mpson
editor

Numerical

Numerical Grid Generation

Editor
Joe F. Thompson

AD A124103
DTIC FILE COPY

This document has been approved for public release and sale; its distribution is unlimited.

DTIC
ELECTE
APR 05 1983
S D
E

North-Holland

83 04 05 164

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0061 / 2. GOVT ACCESSION NO. A127498	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NUMERICAL GRID GENERATION	5. TYPE OF REPORT & PERIOD COVERED FINAL Book
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joe F. Thompson	8. CONTRACT OR GRANT NUMBER(s) AFOSR-ISSA-82-00024
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Aerospace Engineering Mississippi State University Mississippi State, Mississippi 39762	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2304/A3 PE 61102F
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB, DC. 20332	12. REPORT DATE 1982
	13. NUMBER OF PAGES 909
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release - - distribution unlimited

DTIC ELECTE APR 05 1983 E

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Elsevier Science Publsihing Co. Inc, NY, NY 10017, Editor, Joe F. Thompson North Holland. Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and their Use in the Numerical Solution of Partial Differential Equations, April 1982,Nashville Tenn.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The basic ideas of the construction and use of numerically-generated boundary-fitted coordinate systems for the numerical solution of partial computation can be done on a fixed square grid in the rectangular transformed region regardless of the shape of movement of the physical boundaries. A number of different types of configurations for the transformed region and the basic transformation relations from a cartesian system to a general curvilinear system are given. The material of this paper is applicable to all types of coordinate system generation.

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A127 498

Numerical Grid Generation

Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and their Use in the Numerical Solution of Partial Differential Equations
April 1982, Nashville, Tennessee
Sponsored by NASA and AFOSR
Organized by Mississippi State University

Edited by

Joe F. Thompson
Department of Aerospace Engineering
Mississippi State University, Mississippi State, Mississippi

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special

DTIC COPY INSPECTED

NORTH-HOLLAND
New York • Amsterdam • Oxford

© 1982 by Elsevier Science Publishing Co., Inc.
All rights reserved.

Published by:

Elsevier Science Publishing Co., Inc.
52 Vanderbilt Avenue, New York, New York 10017

Sole distributors outside USA and Canada:

Elsevier Science Publishers B.V.
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

The text of this book appears simultaneously in *Applied Mathematics and Computation*, Volumes 10 and 11 (1982).

Library of Congress Cataloging in Publication Data

Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Use in the Numerical Solution of Partial Differential Equations (1982: Nashville, Tenn.)
Numerical grid generation.

Bibliography: p.
1. Numerical grid generation (Numerical analysis)—Congresses. 2. Differential equations, Partial—Numerical solutions—Congresses. 3. Fluid dynamics—Congresses. I. Thompson, Joe F. II. United States. National Aeronautics and Space Administration. III. United States. Air Force. Office of Scientific Research. IV. Mississippi State University. V. Title.
QA377.S966 1982 515.3'53 82-14244
ISBN 0-444-00757-1

Manufactured in the United States of America

Numerical Grid Generation

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENT	xvii
FOREWORD	xix
GENERAL CURVILINEAR COORDINATE SYSTEMS Joe F. Thompson	1
Boundary-Conforming Coordinate Systems	2
Boundary-Value Problem - Physical Region	3
Boundary Value Problem - Transformed Region	3
Orthogonality at the Boundary	4
Transformed Region Configurations	4
Simply-Connected Regions	5
Multiply-Connected Regions	7
Three-Dimensional Regions.	13
Special Grid Points	14
Corners	14
Branch Cuts	15
Derivative Correspondence Across Cuts.	16
Transformation Relations	18
Base Vectors, Tangents, Normals, Area, and Volume	19
Divergence, Gradient, Curl and Laplacian	21
Normal and Tangential Derivatives, Integrals	23
Two-Dimensional Relations	25
Time Derivatives	28
ERROR INDUCED BY COORDINATE SYSTEMS C. Wayne Mastin	31
Derivative Approximations	31
Nonorthogonality	36
Examples	37
BASIC DIFFERENTIAL MODELS FOR COORDINATE GENERATION Z. U. A. Warsi	41
Notation and Basic Formulas	42
Generating Differential Equations Based on Gauss Equations.	45
Fundamental Generating System of Equations	49
Coordinate Redistribution (Concentration).	51
Morphology of A Solution Algorithm	53
Exact Solutions	55
Generating Differential Equations Based on Laplace Equations.	58
Case of Orthogonal Coordinates	63
Generating Differential Equations Based on the Riemann Tensor	66
Case of Orthogonal Coordinates	70
ELLIPTIC GRID GENERATION Joe F. Thompson	79
Elliptic Generation Systems	80
Poisson System	81

Page

General Poisson System 83
Other Systems 85
Systems for Curved Surfaces 86
Control Functions 90
Attraction to Coordinate Lines/Points 90
Attraction to Lines/Points in Space 93
Determination From Related Systems 95
Determination From Boundary Point Distributions 96
Determination From Surface Distributions 99
Automatic Determination 100
Numerical Solution 101
Iterative Solutions 101
Problems 102

CONFORMAL GRID GENERATION 107
David C. Ives

Coordinate Transformations. 108
Flow Simulation Implications 109
Design Implications 110
Complex Variable Notation 110
Conformal Mapping 111
Riemann Sheet Determination 113
Classical Mapping 116
Sequential Mapping 117
Near-Circle to Circle Mapping 117
One-Step Mappings 118
Hinge Point Transformations 121
Multiple Bodies 122
Creating New Mappings 123
Reflection Principle 126
Calculating the Grid 127
Three Dimensions 128
Reference Material 128
Packaged Tools 130

ALGEBRAIC GRID GENERATION 137
Robert E. Smith

Boundary-Fitted Coordinate Transformations 139
Algebraic Grid Generation Methods 144
Transfinite Interpolation 144
The Multisurface Method 150
The Two-Boundary Technique 154
Surface Representation and Parameterization 157
Uniformity. 159
Side Boundary Constraints 164
Grid Generation Topology. 165
Grid Computation 168

Page

TRANSFINITE MAPPINGS AND THEIR APPLICATION TO GRID GENERATION 171
William J. Gordon and Linda C. Thiel

Two-Dimensional Regions 173
Three-Dimensional Solid Structures 177

ORTHOGONAL GRID GENERATION 193
Peter R. Eiseman

Geometry 196
Curves, Surfaces, Tangents, and Metrics 196
Intrinsic Constraints 199
The Restricted Existence of Three-Dimensional Orthogonal Coordinates 201
Analytic and Finite Difference Orthogonality 202
Orthogonal Trajectories 204
Overview 204
The Choice of Nonorthogonal Coordinates 205
The Fundamental Differential Equation 211
An Example of the Fundamental Differential Equation 213
Methods of Generation 215
Field Solutions 222
Overview 222
The Curvature Constraint 222
Coordinates from a Metric Specification 226

PATCHED COORDINATE SYSTEMS 235
P. E. Rubbert and K. D. Lee

Multi-Block Structuring 235
Approaches to Structuring 236
The Overall Grid Generation Process 237
Perimeter Line Discretization (1D) 238
Surface Grids (2D) 238
Volume Grids (3D) 240
Some Observations on Requirements for Grid Quality 242

SOLID MECHANICS APPLICATIONS OF BOUNDARY FITTED COORDINATE SYSTEMS 253
John C. McWhorter

Uniform Shafts - Arbitrary Cross Section 254
Equations for the Membrane Analogy 255
Non-Uniform Shaft, Circular Cross Section, Arbitrary Profile 258
Bending of Arbitrarily Shaped Plates 260
Plate Bending Equations 261
Transformed Plate Equations 262
Solution of Transformed Plate Equations 264
Results for Uniform Shafts 266
Results for Non-Uniform Shafts 270
Results for Plates of Arbitrary Shape 272

Page

COORDINATE SYSTEM CONTROL: ADAPTIVE MESHES 277
J. U. Brackbill

Variational Formulation of the Mesh Generator 278
Solution of the Variational Problem in Two Dimensions 279
The Adaptive Mesh 282
A Numerical Example of Adaptive Zoning 283
Adaptive Zoning of Time-Dependent Problems 284
Adaptivity for Arbitrary Meshes: The Inverse Problem 287
Adaptive Zoning for Moving Boundary Calculations 288

ON APPLICATION OF BODY CONFORMING CURVILINEAR GRIDS FOR FINITE DIFFERENCE SOLUTION OF EXTERNAL FLOW 295
Joseph L. Steger

Conservative Differencing of Transformed Equations 296
Calculations on Curvilinear Grids 303

THE USE OF SOLUTION ADAPTIVE GRIDS IN SOLVING PARTIAL DIFFERENTIAL EQUATIONS 317
Dale A. Anderson and M. M. Rai

The Basic Method 320
Extension to Multidimensional Problems 325
Shock Aligning Schemes 330
Time Requirements 333

ADAPTIVE GRIDDING FOR FINITE DIFFERENCE SOLUTIONS TO HEAT AND MASS TRANSFER PROBLEMS 339
Harry A. Dwyer, Mitchell D. Smooke and Robert J. Kee

Basic System of Equations 340
Adaptive Grid Methods 342
Positive Weight Function Concept 342
Steady-State Problems, Variable Node Method 344
Time-Dependent Problems 345
Coordinate Transformation Methods 345
Linear Algebra Considerations 346
Example Problems 348
Steady Premixed Flames 348
Two-Dimensional Elliptic Boundary Value Problem 350
Unsteady Two-Dimensional Flame Propagation, Coordinate Transformation Method 350

APPLICATION OF CURVILINEAR COORDINATE GENERATION TECHNIQUES TO THE COMPUTATION OF INTERNAL FLOWS 357
Doyle D. Knight

Characteristics of Internal Flows 357
Desirable Characteristics of Curvilinear Coordinates 358
Brief Review of Recent Applications 359
Elliptic Partial Differential Equations 360
Conformal Transformations 361

Page

Algebraic Techniques 361
Other Methods 362
A Method for Generating Two-Dimensional Orthogonal or Nearly Orthogonal Grids with Direct Control of Mesh Spacing 362
Introduction 362
Intermediate Transformation 364
Final Transformation 368
Results 372
A Current Research Problem in Grid Generation for Internal Flows . . 376

SOLUTION OF NONLINEAR WATER WAVE PROBLEMS USING BOUNDARY-FITTED COORDINATE SYSTEMS 385
Henry J. Haussling

Mathematical Theory of Free-Surface Potential Flow 386
Coordinate System Transformation 390
Numerical Techniques 392
Selected Results 395

NUMERICAL MODELING OF ESTUARINE HYDRODYNAMICS ON A BOUNDARY-FITTED COORDINATE SYSTEM 409
Billy H. Johnson

Vertically Averaged Estuarine Hydrodynamics 410
Classification 410
Governing Equations in Cartesian Coordinates 411
Time Averaging for Turbulent Flows 412
Depth Averaging for Nearly Horizontal Flow 412
Required Input Data 414
Horizontal Grids Employed in Hydrodynamic Models 415
Conformal Curvilinear Grids 415
Orthogonal Curvilinear Grids 415
Stretched Rectangular Grids 416
Boundary-Fitted Coordinate Systems 416
Tranformation of Vertically Averaged Equations 419
VAHM - A Vertically Averaged Hydrodynamic Model 422
Computational Grid 423
Differences 423
Boundary Conditions 425
Example Problem 425
Generation of Boundary-Fitted Coordinates 425
Flow Computations 426

AUTOMATED THREE-DIMENSIONAL GRID REFINEMENT ON A MINICOMPUTER 437
P. D. Manhardt and A. J. Baker

Macro Elements 438
Generated Grid Analysis 441
Global Connectivity 442
A 3D Example 443

Page

AUTOMATIC ALGEBRAIC COORDINATE GENERATION 447
Peter R. Eiseman

Multisurface Transformations 448
System Operators 449
The Data Base 449
The Order of Application 450
Direct Surface Generators 450
Geometric Surface Operators 451
Surface Parameterization Operators 452
Surface Generators from Existing Surfaces 453
Transverse Operators 453
Mesh Operators 453
Assembly Operators 454
Data Visualization Operators 455
Applications 455

AUTOMATIC TOPOLOGY GENERATION AND GENERALISED B SPLINE MAPPING 465
A. Roberts

Tesselation of Surfaces 467
Basic Cube Cluster Operations 469
Solid and Null Volumes 471
Topological Displays 472
Basis Function Mapping 476
Integer Spline Mapping 477
Comparison of Generating Functions 478
Topological Singularities 482

THE NUMERICAL DIFFERENTIATION OF DISCRETE FUNCTIONS USING POLYNOMIAL INTERPOLATION METHODS 487
James M. Hyman and Bernard Larrouturou

Interpolation Methods 490
Mapping Methods 495
Software Design 498
Usage 503

SOLUTION OF VISCOUS INTERNAL FLOWS ON CURVILINEAR GRIDS GENERATED BY THE SCHWARZ-CHRISTOFFEL TRANSFORMATION. 507
O. L. Anderson, R. T. Davis, G. B. Hankins, and D. E. Edwards

Analysis 509
Curvilinear Coordinate Analysis 509
Viscous Flow Solution 514
Results and Discussion 517
Comparison of Coordinate Calculations 517
AGT101 Gas Turbine 517
AGT101 Turbine Inlet Duct 518
AGT101 Turbine Exhaust Diffuser 520

Page

TEST PROBLEMS, COORDINATE TRANSFORMATIONS, AND TECHNIQUE FOR NONSTEADY COMPRESSIBLE FLOW ANALYSIS. 525
Jon J. Yagla

Conservation Law Form 525
Finite Difference Equations 526
Coordinate Transformation 528
General Remarks 528
Special Consideration for External Aerodynamics and Transient Field Calculations. 531
Schwarz-Christoffel Transformation 532
Numerical Integration of Schwarz-Christoffel Differential Equation. . 534
Test Problems 536
Application to Shock-on-Shock Problem 543

AN EXPERIENCE IN MESH GENERATION FOR THREE-DIMENSIONAL CALCULATION OF POTENTIAL FLOW AROUND A ROTATING PROPELLER 547
Wen-Huei Jou

Slicing Three-Dimensional Space 548
Two-Dimensional Cascade Meshes 549
General Requirements for Cascade Meshes. 550
C-Type and O-Type Meshes 551
H-Type Mesh. 554

FAST GENERATION OF THREE-DIMENSIONAL COMPUTATIONAL BOUNDARY-CONFORMING PERIODIC GRIDS OF C-TYPE. 563
Djordje S. Dulikravich

Shearing and Stretching in Physical Space 565
Conformal Mapping and Remapping 569
Shearing and Stretching in Computational Space 571
Results 572

CONFORMAL GRID GENERATION FOR MULTIELEMENT AIRFOILS 585
Douglas Halsey

Conformal Mapping of Multielement Airfoils 587
Grids Using a String Mapping 588
Grids Using Streamline/Potential-Line Networks. 590
ϕ-ψ Grids for Streaming Flows 591
ϕ-ψ Grids for Circulatory Fundamental Flows 593
ϕ-ψ Grids for More General Flows 593
Segmented Grids with Specified Boundaries 596
Hybrid Grids 598

CONFORMAL MAPPINGS ONTO MULTIPLY CONNECTED REGIONS WITH SPECIFIED BOUNDARY SHAPES. 601
Andrew N. Harrington

Theoretical Formulation 602
Numerical Formulation 604

Page

Regions Whose Outer Boundaries Are Rectangles 607
Further Possible Modifications 612
Some Experimental Results 614

3-D SOLUTION OF FLOW IN AN INFINITE SQUARE ARRAY OF CIRCULAR TUBES BY USING BOUNDARY-FITTED COORDINATE SYSTEM 619
B. C-J. Chen, T. H. Chien, W. T. Sha, and J. H. Kim

Coordinate Transformation 619
Control Volumes 621
Numerical Methods 623
Applications 625

GENERATION OF BOUNDARY-FITTED COORDINATE SYSTEMS USING SEGMENTED COMPUTATIONAL REGIONS 633
Roderick M. Coleman

Description of the Method 633
Use of the Program 635
General Remarks 636
Examples and Applications 636

GRID GENERATION BY ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS FOR A TRI-ELEMENT AUGMENTOR-WING AIRFOIL. 653
Reese L. Sorenson

The Augmentor-Wing. 653
The "Grape" Grid-Generation Algorithm 654
Application of Grape to the Two Flow Simulation Studies 656
Body-Surface Distribution 659
Results 661

NUMERICAL GENERATION OF COMPOSITE THREE DIMENSIONAL GRIDS BY QUASILINEAR ELLIPTIC SYSTEMS. 667
P. D. Thomas

Three-Dimensional Grids for Simply-Connected Regions 669
Two-Dimensional Grids on Curved Surfaces. 673
Composite Grids and Multiply-Connected Regions. 675
Numerical Results 676
Three-Dimensional Composite Grid for a Wing-Body Combination . . 676
Surface Grids. 677

THREE-DIMENSIONAL GRID GENERATION USING POISSON EQUATIONS. 687
C. F. Shieh

Grid Generation Procedure 688
Grid Spacing Control 690

Page

GENERATION OF THREE-DIMENSIONAL BOUNDARY-FITTED CURVILINEAR COORDINATE SYSTEMS FOR WING/WING-TIP GEOMETRIES USING THE ELLIPTIC SOLVER METHOD . . 695
Frank C. Thames

Form of the Transformation. 697
Generation of the Transformation. 698
Governing Equations. 698
Control Functions. 700
Numerical Solution Technique 703
Discussion of Results 703
Cases Computed and Boundary Geometry 703
Sample Results 705
Effects of the Control Functions 705
Performance of the Numerical Method. 706

NUMERICAL GENERATION OF THREE-DIMENSIONAL COORDINATES BETWEEN BODIES OF ARBITRARY SHAPES. 717
Z. U. A. Warsi and J. P. Ziebarth

Formulation of the Mathematical Model 717
Numerical Solution of the Equations 722

SEMIDIRECT/MARCHING SOLUTIONS AND ELLIPTIC GRID GENERATION 729
Patrick J. Roache

Semidirect/Marching Methods: The Gem Codes 729
Application to Elliptic Grid Generation Equations 730
Accuracy and Timing Tests 731
Continuation Methods for Difficult Geometries 732
Solution Oscillations Near Gradient Boundaries. 733
Sensitivity to Cross Derivatives. 733
Symbolic Manipulation and Grid Generation 734
Future Work 736

2-D ELLIPTIC GRID GENERATION USING A SINGULARITY METHOD AND ITS APPLICATION TO TRANSONIC INTERFERENCE FLOWS 739
Karl D. Klevenhusen

Grid Generation by Singularity Methods. 742
Selection of a Suitable Singularity Method 742
The Singularity Method in the Physical Plane 743
The Singularity Method in the Streamline Plane 744
Grid Generation for Transonic Flow Computation. 747
Grid Spacing 747
The Stagnation Point Problem 748
Application to Transonic Interference Flows. 749

THREE DIMENSIONAL GRID GENERATION USING BIHARMONICS 761
G. R. Shubin, A. B. Stephens, and J. B. Bell

Analytic Formulation. 762
Discretization and Numerical Solution 766
Results 768

Page

MARCHING GRID GENERATION USING PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS. . 775
S. Nakamura

Preliminary Observations. 776
Practical Applications. 777
Grid Spacing Control. 778
Local Orthogonality of Grid Lines 782
Illustration of the Grids Generated 783

ASSESSING THE QUALITY OF CURVILINEAR COORDINATE MESHES BY DECOMPOSING THE JACOBIAN MATRIX 787
G. David Kerlick and Goetz H. Klopfer

Decomposition of the Jacobian Matrix. 788
Laplace's Equation. 791
Truncation Error. 792
Polar Coordinates 793
Analysis Applied to the Solution of the Full Potential Code Tair. . . 794
Numerical Results 796

AN IMPLICIT SCHEME FOR WATER WAVE PROBLEMS 809
Martha B. Aston and J. W. Thomas

Mathematical Formulation. 810
Numerical Grid Generation 812
Numerical Scheme. 815
Numerical Results 816

A VECTORIZED, FINITE-VOLUME, ADAPTIVE-GRID ALGORITHM FOR NAVIER-STOKES . . 819
Peter A. Gnoffo

Finite-Volume Formulation 820
Boundary Conditions 825
Adaptive Grid 825
Results and Discussions 829

IDEALIZED DYNAMIC GRID COMPUTATION OF PHYSICAL SYSTEMS 837
Joshua A. Anyiwo

Strain Field Model of Physical Systems. 838
Sample Computation of Strain Density Fields. 842
Dynamic Grid Generation Using Invariant-Mapping 843
The Tau Computational Space 844
A Grid Dynamism Constraint 845
A Grid Orthogonality Constraint. 846
A Grid Smoothness Constraint 847
A Tau Computational Space Method. 848

EQUIDISTANT MESH FOR GAS DYNAMIC CALCULATIONS. 859
C. M. Ablow

Page

APPLICATIONS AND GENERALIZATIONS OF VARIATIONAL METHODS FOR GENERATING ADAPTIVE MESHES 865
Jeffrey Saltzman and Jeremiah Brackbill

The Variational Formulation in Two Dimensions 866
Generalization of the Variational Formulation to Three Dimensions . . 869
An Application in Two Dimensions. 878

ORTHOGONAL COORDINATE MESHES WITH MANAGEABLE JACOBIAN. 885
C. I. Christov

Governing Equations 885
Method of Solution. 887
Results and Discussion. 890
Generalization for Three Dimensions 892

INDEX 895

ACKNOWLEDGMENT

This symposium was sponsored jointly by NASA Headquarters and the U. S. Air Force Office of Scientific Research under NASA Contract NASW-3552. The contract officers, Dr. Randolph A. Graves of NASA and Major C. Edward Oliver of the Air Force participated in the planning and execution of the meeting throughout the duration of this project. Their interest, encouragement, and contributions were instrumental in bringing this project to fruition.

A number of the papers, because of their expository nature, required more than the usual amount of effort that is involved in the preparation of conference presentations. This effort is evident in the papers and is gratefully acknowledged.

The administrative support of Charles B. Cliett, head of the Department of Aerospace Engineering of Mississippi State University, and Melvin B. Swartzberg, business manager for the College of Engineering, was essential in carrying out this project. Finally, the conscientious effort of the secretarial staff of the Department of Aerospace Engineering, particularly Patricia McFadden, throughout the preparation for the meeting and the assembling of this volume must be noted.

PREVIOUS PAGE IS BLANK

FOREWORD

In the past decade the numerical generation of boundary-conforming curvilinear coordinate systems has provided the key to the development of finite difference solutions of partial differential equations on regions with arbitrarily shaped boundaries. Although much of the impetus for these developments has come from fluid dynamics, the techniques are equally applicable to heat and mass transfer, electromagnetics, solid mechanics, and all other areas involving field solutions.

This symposium was designed to provide a forum for the presentation and interchange of ideas and results on the numerical generation of such coordinate systems and their application in the numerical solution of partial differential equations. A number of specially prepared expository papers was included with the submitted papers. These expository papers are designed to cover the basic techniques involved in the numerical generation of curvilinear coordinate systems and the use thereof in the numerical solution of partial differential equations. These papers thus can serve to provide an introduction to such techniques for all concerned with field solutions and also to acquaint users of such systems with additional techniques.

A boundary-conforming coordinate system is a curvilinear coordinate system having some coordinate line (surface in 3D) coincident with each segment of the boundary of a region. (Such coordinate systems have been variously termed boundary-fitted, body-fitted, surface-fitted, surface-orientated, surface-conforming, etc. in the literature.) When partial differential equations are transformed onto such a coordinate system, finite difference representations can be made using only neighboring points at coordinate line intersections, without need of interpolation, regardless of the boundary shape or even its movement. Even with a moving boundary, all computation thus can always be done on a fixed square grid in the transformed region. This allows quite general codes to be written for the numerical solution of partial differential equations on arbitrary regions. Since the coordinate system itself can also be generated numerically, the complication of boundary shape is thus effectively removed from the problem, and diverse configurations can be treated by a single code, the boundary shape being either an item of input or to be determined by the solution in the case of moving boundaries.

Boundary conforming coordinate systems are generated numerically by determining the values of the physical coordinates (cartesian or otherwise) in the field from the values (and/or angles of intersection) on the boundary. This can be done in two basic ways: (1) by algebraic interpolation from the boundary

PREVIOUS PAGE IS BLANK

values, or (2) by solving a set of partial differential equations with the boundary values as boundary conditions therefor. In general, the coordinate system should have lines concentrated in regions of expected high variation of the physical solution, but the system should be smooth and the spacing should not change too rapidly. Orthogonality is not necessary, but the departure therefrom should not be excessive, although quite large departures can be tolerated. Ultimately the coordinate system should be coupled with the physical solution thereon, so that the coordinate line spacing continually adapts to resolve the developing variations in the physical solution.

The first seventeen papers of this volume are expository in nature, with the topics selected to provide an introduction to the various techniques and considerations involved in the generation and use of boundary-conforming coordinate systems. The general ideas of such coordinate systems and the necessary transformation relations for partial derivatives, integrals, normal and tangent vectors, etc. are given in the first paper (Thompson) of this volume. This paper also introduces various configurations possible for the transformed plane and discusses the use of branch cuts and points requiring special consideration. It is possible for poor distribution or orientation of coordinate lines to introduce error into a numerical solution done thereon, and the determination and control of such error sources is discussed in the second paper (Mastin). This is a particularly important concern since sudden changes in line spacings and excessive skewness of lines can introduce negative numerical diffusion into a solution. These first two papers are relevant to the use of boundary-conforming coordinate systems in general, regardless of the method of generation.

The difference representation of derivatives on general grids is discussed in the paper of Hyman & Larrouturou. When the transformed partial differential equations are written in strong conservative form on a curvilinear coordinate system, i.e., such that the metric coefficients appear inside the derivatives, it is important that the differencing be done in such a way that a uniform solution will be preserved. With the metric coefficients inside the derivatives, it is possible for differences of these coefficients to fail to cancel exactly, thus introducing spurious source terms which cause the solution to drift away from the uniform case. This topic is discussed in the paper by Steger, where some corrective measures are suggested, and in the paper of Gnoffo. In some cases this problem can cause analytical metric coefficients to give poorer solutions than are obtained with coefficients evaluated from difference expressions, as noted in the paper of Mastin.

As noted above, there are two basic approaches to the numerical generation of boundary-conforming coordinate systems: generation from the solution of partial differential equations and generation by algebraic interpolation between boundaries. The first of these includes conformal mapping as well as generation from the solution of general elliptic, parabolic, and hyperbolic systems. A general discussion of appropriate partial differential systems is given in the paper of Warsi, while elliptic systems are specifically covered in the second paper of Thompson. The use of some hyperbolic systems is included in the first paper of Eiseman, and in the paper of Steger, and parabolic systems are applied in the paper of Nakamura.

Particular elliptic systems are discussed in the papers of Rubbert & Lee, Brackbill, Coleman, Sorenson, Thomas, Shieh, Thames, Warsi & Ziebarth, Klevenhusen, Shubin et al., Saltzman & Brackbill, and Christov. Applications of coordinate systems generated from elliptic systems are included in the papers of McWhorter, Knight, Haussling, Johnson, and Chen et al. Applications of a hyperbolic generating system appears in the paper of Steger.

Control of the coordinate line spacing and orientation is exercised through adjustable terms in the partial differential equations through which it is possible to cause coordinate lines and/or points to be attracted to other coordinate lines and/or points or to locations in the physical plane. Some specific procedures for the control of coordinate line spacing are included in the second paper of Thompson with elliptic generating systems. Some general discussion of this topic appears in the paper of Warsi, and the determination from boundary point distribution is discussed in the paper of Thomas.

The generation of coordinate systems by solving partial differential equations requires numerical solution of difference equations on the field. This has generally been done for elliptic systems by iterative procedures such as SOR, ADI, multi-grid techniques, etc. as discussed in the second paper of Thompson. A singularity solution method is used in the paper of Klevenhusen. The paper by Roache discusses a semidirect marching procedure that is applicable to elliptic systems in some cases. Marching techniques, typically involving tri-diagonal solutions, are generally applicable to parabolic and hyperbolic systems as used in the papers of Nakamura and Steger, respectively.

An extensive discussion of conformal transformations is given in the paper of Ives, where a number of techniques are covered and a comprehensive list of references is given. This paper also discusses techniques for exercising some control of the line spacing through subsequent stretching transformations. Further discussions and applications of conformal transformations are given in

the papers of Anderson et al., Yagla, Jou, Dulikravich, Halsey, and Harrington. Regions with multiple internal boundaries are treated in the papers of Halsey and Harrington, as well as in that of Ives.

The other major class of coordinate system generation procedures--algebraic generation systems--utilizes interpolation formulas to determine the values of the coordinates at the points in the field. The paper of Smith and that of Gordon & Thiel provide a general discussion of the techniques involved, and further discussions are given in the papers of Eiseman and in the paper of Roberts, as well as in the paper of Manhardt & Baker. The paper of Gordon & Thiel also provides a general discussion of the theory of the interpolation techniques involved. Control of the coordinate lines with algebraic generation systems is accomplished through stretching functions in the interpolation formulas. The paper of Smith discusses interactive control of the coordinate system using computer graphics.

Although orthogonality is not necessary, certain error terms do vanish for an orthogonal system as may some terms of transformed differential equations. Orthogonality, however, places certain constraints on the point distributions. A comprehensive discussion of the generation of orthogonal coordinate systems is given in the first paper of Eiseman, which treats both algebraic generating systems and systems based on hyperbolic partial differential equations. This paper also includes some basic theoretical aspects of orthogonal systems, as does the paper of Warsi. Orthogonal systems generated from hyperbolic systems are used in the paper of Steger. Orthogonal systems can also be generated from elliptic systems by determining the control functions such as to achieve orthogonality. This approach is discussed in the papers of Knight and Christov.

In order to adequately resolve solution gradients it is necessary that the coordinate system be controlled so that coordinate lines are concentrated in regions of large variations of the solution, but without excessive spacing changes, skewness, or depletion of lines in other regions. Ultimately the coordinate system should be coupled with the physical solution thereon so that the coordinate lines continually adapt to follow regions of developing variations while maintaining smooth coverage of the entire field. This area is now receiving considerable attention, and the papers of Brackbill and Anderson cover some of the basic ideas involved.

Brackbill implements the adaptive control through terms in the elliptic generating system, while Anderson gives a procedure based on an analogy with electrostatic charge attraction which is applicable to any coordinate system. The adaptive procedure of Brackbill is discussed further and applied in the

paper of Saltzman & Brackbill, where striking results for multiple shock reflections are given. In the procedure of Anderson, grid points attract or repel other points in accordance with the deviation of some local error measure from the average measure over the field. Force couples are also employed to cause lines of grid points to align with solution phenomena such as shocks. A related procedure is used in the paper of Gnoffo. Other adaptive control procedures are discussed in the papers of Dwyer et al., Anyiwo, and Ablow.

Related to the subject of coordinate system control is the question of evaluating the quality of a system. As noted above, rapid changes in spacing and excessive skewness can cause errors, as is discussed in the paper of Mastin. The question of quality assessment is addressed directly in the paper of Kerlick, where various aspects of the grid are isolated and some specific assessment procedures are developed. The elliptic generation system of Brackbill (cf. also the paper of Saltzman & Brackbill) includes procedures for balancing the conflicting requirements of smoothness, avoidance of excessive skewness, and line concentration. The requirements placed on the coordinate system can be relaxed somewhat if solution algorithms to be used thereon are "forgiving" as noted in the paper of Rubbert & Lee.

The generation of surface coordinate systems on curved surfaces is discussed in general in the paper of Warsi, and further discussions are given in the paper of Warsi & Ziebarth and that of Thomas. Some discussion of this topic is also included in the second paper of Thompson and in the papers of Eiseman. Surface representation is also discussed in connection with the algebraic generation system in the papers of Smith and Gordon & Thiel.

The general idea of embedding a coordinate system for a subregion within a larger system is a powerful tool for the treatment of more complicated configurations even in two dimensions, particularly with configurations involving multiple interior boundaries. Some continuity must be preserved in the coordinate system at the junctures of the subsystems, certainly in regard to point distribution and preferably also for the slope and the higher derivatives of the coordinate coefficients. The algebraic generation procedures can achieve this by including derivatives in the interpolation formulas, either directly through Hermite interpolation or indirectly through the use of intermediate surfaces. The papers of Smith and Gordon & Thiel, and the second paper of Eiseman, are relevant to this topic. With elliptic generation systems, derivatives can be matched at mesh boundaries by iteratively adjusting the control functions in the equations as in the paper of Sorenson. Also in this regard, the paper of Coleman discusses a code applicable to very general two-dimensional

configurations patched with complete continuity at the junctures. Higher-order elliptic systems, such as in the paper of Shubin et al., allow additional boundary conditions so that derivatives can be matched. The first paper of Eiseman discusses the embedding of two-dimensional orthogonal coordinate systems. The concept of embedding is used with conformal systems in the paper of Halsey.

Complicated three-dimensional configurations are now becoming accessible, being generally treated by patching together coordinate systems generated separately for subregions surrounding particular components. This topic is introduced in the paper of Rubbert & Lee, and further discussions are given in the papers of Roberts and Thomas, as well as in the papers of Manhardt & Baker and Shubin et al.

The construction of three-dimensional systems by stacking two-dimensional systems generated on successive surfaces is included in the papers of Warsi and Thomas, as well as in the second paper of Thompson. An example appears in the paper of Chen et al. The construction of three-dimensional systems from an assembly of two-dimensional conformal systems is included in the papers of Ives, Jou, and Dulikravich. Three-dimensional configurations treated directly without such patching together of subregions or stacking of two-dimension surface systems are discussed in the papers of Warsi & Ziebarth and Thames.

As noted in the paper of Ives, combinations of generation techniques should be considered for general configurations. Thus algebraic stretching can be applied to a coordinate system generated from partial differential equations in order to redistribute the coordinate lines. Similarly, an elliptic generation system could be used to smooth a coordinate system generated algebraically. It might also be useful to generate an overall grid with one type of generation system and then generate finer grids within subregions of the larger grid with another type of generation procedure.

Since the time derivatives can be transformed as well, the computations can still be done on a fixed square grid in the transformed plane even if the boundaries are in motion. The use of such time-dependent systems with a moving free surface boundary is discussed in the paper of Haussling, in the related paper of Coleman, and in the paper of Aston & Thomas.

Applications to fluid mechanics are made in a number of the papers. The paper of McWhorter discusses applications to solid mechanics, specifically the bending of plates and the torsion in shafts. Heat and mass transfer applications with combustion are given in the paper of Dwyer et al. Free surface flow problems are discussed in the papers of Haussling, Coleman, and Aston & Thomas. The paper of Johnson gives applications to estuarine flows involving branching

channels, islands, and also free surfaces simulated with depth-averaging.

Complicated wing/body/nacelle configurations of airplanes are treated in the papers of Rubbert & Lee and Roberts. Wing/wing tip configurations are given in the paper of Thames. Blade/hub configurations are treated in the papers of Jou and Dulikravich. Cascade applications are included in the papers of Ives, Jou, Dulikravich. Other examples are given in the paper of Knight. Applications to external aerodynamics are discussed in the papers of Steger, Yagla, Klevenhusen and Gnoffo. Particular considerations related to internal flows are discussed in the paper of Knight. Other internal flows are treated in the papers of Anderson et al. and Chen et al.

In conclusion, the areas of most importance for further research are the analysis and reduction of error introduced by the coordinate system, the automation of control of coordinate line spacing to achieve both concentration and smoothness, the dynamic coupling of the coordinate system with the physical solution, and the patching together of regions to represent general configurations with sufficient continuity. The increasing interest and progress in the generation and use of numerically generated boundary-fitted coordinate systems are evidenced by several recent conferences devoted to this topic, and the present volume should serve as an introduction to the state-of-the-art in this area for all concerned with the numerical solution of partial differential equations. Coordinate system generation and use can be expected to evolve at an even more rapid pace in the coming years, and it is hoped that this present volume will provide coverage of the developments thus far from which new directions can be charted.

Joe F. Thompson
Mississippi State, MS
May 1982

AD P000966

Copyright 1982 by Elsevier Science Publishing Company, ...
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

GENERAL CURVILINEAR COORDINATE SYSTEMS

JOE F. THOMPSON
Department of Aerospace Engineering, Mississippi State University,
P. O. Drawer A, Mississippi State, Mississippi 39762

ABSTRACT

The basic ideas of the construction and use of numerically-generated boundary-fitted coordinate systems for the numerical solution of partial differential equations are discussed. With such coordinate systems, all computation can be done on a fixed square grid in the rectangular transformed region regardless of the shape or movement of the physical boundaries. A number of different types of configurations for the transformed region and the basic transformation relations from a cartesian system to a general curvilinear system are given. The material of this paper is applicable to all types of coordinate system generation.

INTRODUCTION

Numerical solution of partial differential equations requires some discretization of the field into a collection of points or elemental volumes (cells). The differential equations are approximated as a set of difference equations on this collection, and this set of algebraic equations is then solved for the discrete values of the functions.

Now, although difference expressions can be obtained on a random point distribution, the discretization of the field requires some organization for the solution thereon to be efficient, i.e., it must be possible to efficiently identify the points or cells neighboring the computation site. Furthermore, the discretization must conform to the boundaries of the region in such a way that boundary conditions can be accurately represented. This organization is provided by a coordinate system, and the need for alignment with the boundary is reflected in the choice of cartesian coordinates for rectangular regions, cylindrical coordinates for circular regions, etc.

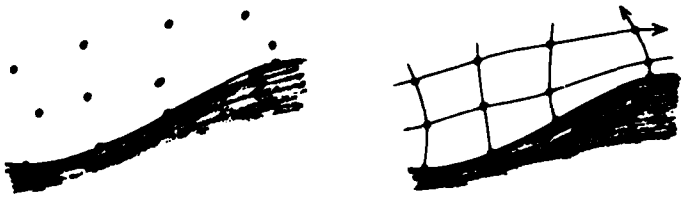


Fig. 1.

The current interest in numerically-generated boundary-conforming coordinate systems arises from this need of organization of the discretization of the field for general regions. Using such coordinate systems, numerical solutions of partial differential equations on physical regions of arbitrary shape can be constructed and codes can be developed that require only changes in the input to treat different physical configurations and boundary shapes.

This paper discusses the basic concepts of such coordinate systems, various configurations of the transformed region, and the transformation relations needed for the numerical solution of partial differential equations thereon. Various procedures for generating these coordinate systems and examples of applications are covered in the following papers of this volume. A general survey of this area has recently been given by Thompson, et al[1]. In the following sections, a two-dimensional region will be considered in most of the discussions for economy of presentation. Generalization to three dimensions will be evident in most cases and will be mentioned specifically only when necessary.

BOUNDARY-CONFORMING COORDINATE SYSTEMS

The basic idea of a boundary-conforming curvilinear coordinate system is to have some coordinate line (in 2D, surface in 3D) coincident with each boundary segment, analogous to the way in which lines of constant radial coordinate coincide with circles in a cylindrical coordinate system. The other curvilinear coordinate will vary along the boundary segment and clearly must do so monotonically, else the same pair of values of the curvilinear coordinates will occur at two different physical points. (It should be clear that the curvilinear coordinate that varies along a boundary segment must have the same direction and range of variation over some opposing segment, e.g., as the angular variable varies from 0 to 2π over both of two concentric circles in cylindrical coordinates).

With the values of the curvilinear coordinates specified on the boundary, it then remains to generate values of these coordinates in the field from these boundary values. There must, of course, be a unique correspondence between the cartesian (or other basis system) and curvilinear coordinates, i.e., the mapping of the physical region onto the transformed region must be one-to-one, so that every point in the physical field corresponds to one, and only one, point in the transformed field, and vice versa. Coordinate lines of the same family must not cross, and lines of different families must not cross more than once.

Boundary-value problem - physical region

The generation of the curvilinear coordinate system may be stated as follows: with the curvilinear coordinates specified on the boundaries, e.g., $\xi(x,y)$ and $\eta(x,y)$ specified for (x,y) on a boundary curve Γ (this specification being a constant value for either ξ or η on each segment of Γ, with a monotonic variation of the other), generate the values, $\xi(x,y)$ and $\eta(x,y)$, in the field bounded by Γ. This is thus a boundary value problem on the physical field with the curvilinear coordinates (ξ,η) as the dependent variables and the cartesian coordinates (x,y) as the independent variables, with boundary conditions specified on curved boundaries.

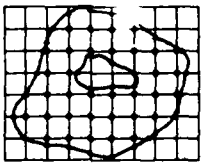


Fig. 2.

(In this discussion the transformation is assumed to be from cartesian coordinates in the physical plane. The transformation can, however, be from any system of coordinates in the physical plane.)

Boundary value problem - transformed region

The problem may be simplified for computation, however, by first transforming so that the physical cartesian coordinates (x,y) become the dependent variables, with the curvilinear coordinates (ξ,η) as the independent variables. Since constant values of one curvilinear coordinate, with monotonic variation of the other, have been specified on each boundary segment, it follows that these boundary segments in the physical field will correspond to vertical or horizontal lines in the transformed field. Also, since the range of variation of the curvilinear coordinate varying along a boundary segment has been made the same over opposing segments, it follows that the transformed field will be composed of rectangular blocks.

The boundary value problem in the transformed field then involves generating the values of the physical cartesian coordinates, $x(\xi,\eta)$ and $y(\xi,\eta)$, in the field from the specified boundary values of $x(\xi,\eta)$ and $y(\xi,\eta)$ on the rectangular boundary of the transformed field, which is formed of segments of constant ξ or η, i.e., vertical or horizontal lines.

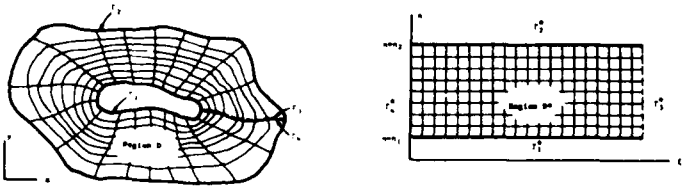


Fig. 3.

The problem is thus much more simple in the transformed plane, since the boundaries there are all rectangular. The computation in the transformed plane thus is on a square grid regardless of the shape of the physical boundaries.

With values of the cartesian coordinates known in the field as functions of the curvilinear coordinates, the network of intersecting lines formed by contours (surfaces in 3D) on which a curvilinear coordinate is constant, i.e., the curvilinear coordinate system, provides the needed organization of the discretization with conformation to the physical boundary.

Orthogonality at the boundary

It is also possible, of course, to apply Neumann boundary conditions, rather than Dirichlet, for the cartesian coordinates on the boundaries of the transformed region, so that coordinate lines intersect the boundary normally. This amounts to leaving the points of intersection of the curvilinear coordinate lines with the physical boundary free to move along that boundary with the angle of intersection, rather than the location, being specified. This condition is applied by requiring that the dot product of the tangent to the intersecting coordinate line with the tangents to the coordinate lines on the boundary vanish. This condition, together with the equation defining the boundary, serves to determine the coordinates of the boundary points in the course of the generation of the coordinate system. More general generation procedures can also be formulated which allow specification of both the coordinates and orthogonality on the physical boundary.

TRANSFORMED REGION CONFIGURATIONS

As noted above, the generation of the curvilinear coordinate system is done by devising a scheme for determination of the field values of the cartesian coordinates from specified values of these coordinates (and/or curvilinear coordinate line intersection angles) on the boundary of the transformed region. Since the boundary of the transformed region is comprised

of horizontal and vertical line segments corresponding to segments of the physical boundary on which a curvilinear coordinate is specified to be constant, it should be evident that the configuration of the coordinate system depends on how this boundary correspondence is made.

Some examples of different configurations are given below from which more complex configurations can be inferred. In these examples only a minimum number of coordinate lines are shown in the interest of clarity for presentation. In all of these examples boundary values of the physical cartesian coordinate (and/or Neumann boundary conditions) are understood to be specified on all boundaries, both external and internal, of the transformed region except for segments indicated by dotted lines. These latter segments correspond to branch cuts in the physical plane as is explained in connection with the occurrence thereof in the examples given. Such re-entrant boundary segments always occur in pairs, the members of which are indicated on each of the configurations shown. Points outside the field across one segment of such a pair are coincident with points inside the field across the other segment in the pair. In most cases an example of an actual coordinate system is given as well, and other examples can be found in the following papers of this volume. References to the use of various configurations may be found in Thompson, et al.[1]

Simply-connected regions

It is natural to define the same curvilinear coordinate to be constant on each member of a pair of generally opposing boundary segments in the physical plane. Thus, a simply-connected region formed by four curves is logically treated by transforming to an empty rectangle:

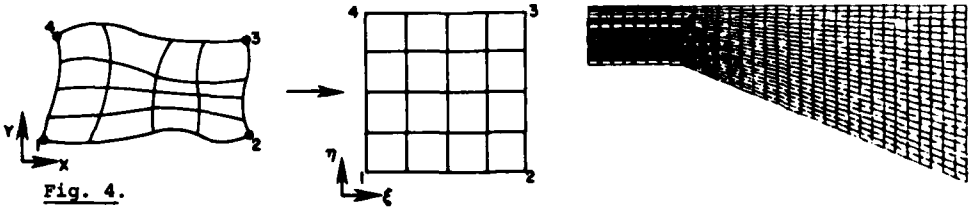


Fig. 4.

Similarly, a generally L-shaped region could remain L-shaped in the transformed region.

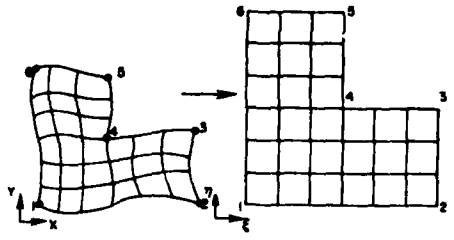


Fig. 5.

The generalization of these ideas to more complicated regions is obvious, the transformed region being composed of contiguous rectangular blocks.

Note, however, that the physical boundary segment on which a single curvilinear coordinate is constant can have slope discontinuities. Therefore, the L-shaped region above could have been considered to be composed of four segments instead of six, so that the transformed region becomes a simple rectangle:

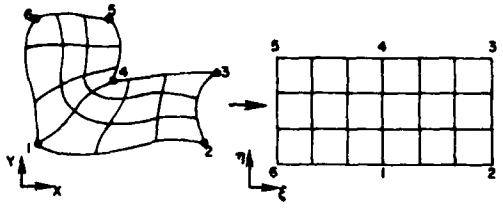


Fig. 6.

Whether or not the boundary slope discontinuity propagates into the field, so that the coordinate lines in the field exhibit a slope discontinuity, depends on how the coordinate system in the field is generated. As is discussed in a later paper in these proceedings, coordinate systems generated as the solution of elliptic partial differential equations do not show such propagation of boundary slope discontinuities into the field.

Also, it is not necessary that boundary slope discontinuities on the physical boundary correspond to corners on the boundary of the transformed region, and a counter-example follows next:

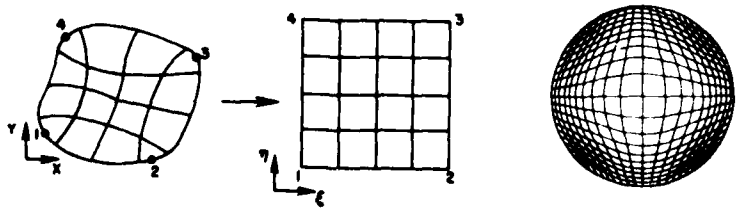


Fig. 7.

In this case, the segment 1-2 on the physical boundary is a line of constant η, while the segment 2-3 is a line of constant ξ.

Since the species of curvilinear coordinate necessarily changes at a corner on the transformed boundary, the identification of a corner with a

point on a smooth physical boundary may require special treatment of such a point in the difference representations used in numerical solutions of the curvilinear coordinate system as discussed later. A point of slope discontinuity on the physical boundary also requires special treatment in difference solutions, since no normal can be defined thereon. This, however, is inherent in the nature of the physical boundary and is not related to the construction of the transformed configuration.

Some slightly more complicated examples of the alternatives introduced above now follow:

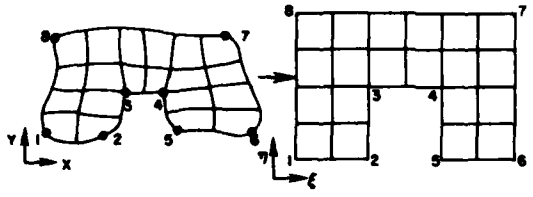


Fig. 8.

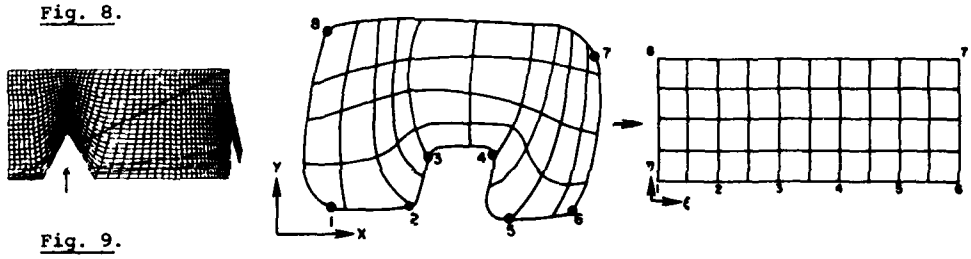


Fig. 9.

Still another alternative in this case would be to collapse the intrusion 2-3-4-5 to a slit in the transformed region:

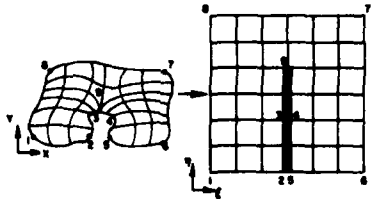


Fig. 10.

Here the physical cartesian coordinates are specified and double-valued on the vertical slit, 2-9-5, in the transformed region. Solution values in a difference solution on such a coordinate system would also be double-valued on the slit, of course.

Multiply-connected regions

With obstacles in the interior of the field, i.e., with interior boundaries, there are still more alternative configurations in the transformed region.

One possibility is to maintain the connectivity of the transformed region the same as that of the physical plane as in the following examples showing two variations of this approach, using interior slabs and slits, respectively, in the transformed region:

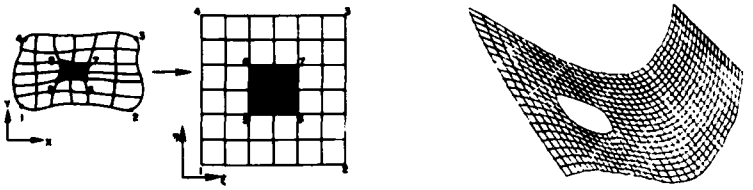


Fig. 11.

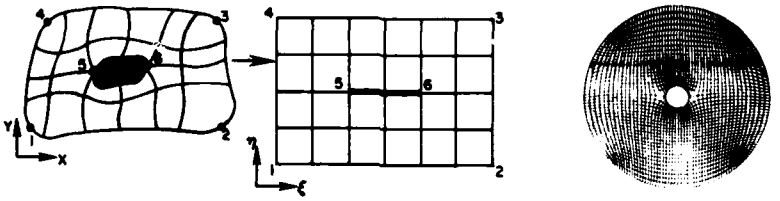


Fig. 12.

Another obvious variation would be to have the slit vertical in the latter case. In the second of these configurations, the points 5 and 6 will require special treatment in difference solutions as is discussed later.

The transformed region could, however, be made simply-connected by introducing a branch cut in the physical region as illustrated below:

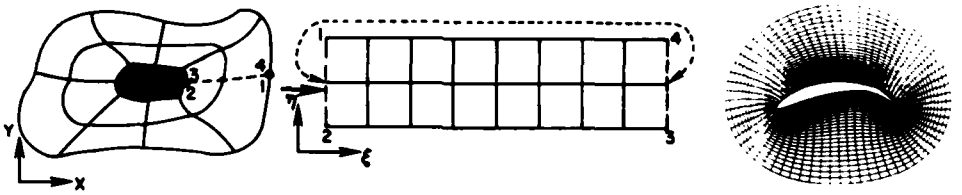


Fig. 13.

Here the coincident coordinate lines 1-2 and 4-3 form a branch cut, which becomes re-entrant boundaries on the left and right sides of the transformed region. All derivatives are continuous across this cut, so that points outside the right side boundary in the transformed region are the same as corresponding points on the same horizontal line inside the left side boundary, and vice versa. Boundary values are not specified on the cut. This cut is, of course, analogous to the coincident 0 and 2π lines in a cylindrical coordinate system.

This type of configuration is often called an O-type. Another possible configuration of this type is as shown below, often called a C-type.

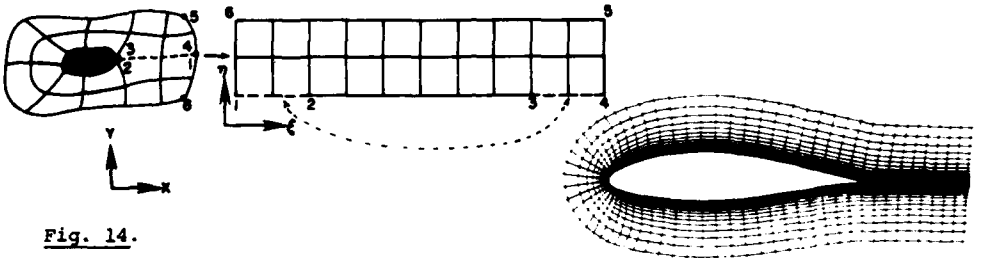


Fig. 14.

Here the two members of the pair of segments forming the branch cut are both on the same coordinate line, and, consequently, points located below the segment 1-2 on the left portion of the bottom of the transformed plane coincide with points above the segment 4-3 located a corresponding distance to the right of the centerline.

Regions of higher connectivity than those shown above are treated in a similar manner. The connectivity may be maintained as in the following illustration:

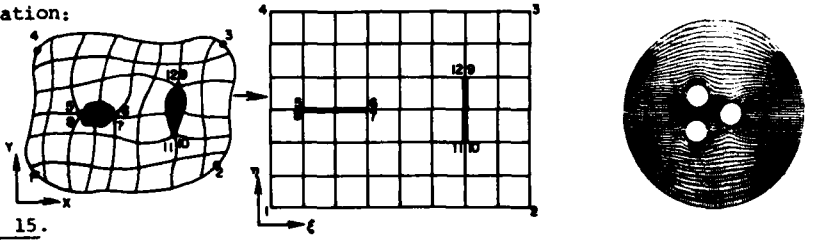


Fig. 15.

Here one slit is made horizontal and one vertical just for generality of illustration. Both could, of course, be of the same orientation. Slabs, rather than slits, could also have been used. The WESCOR Code of Thompson[3] is applicable to two-dimensional regions with any number of interior obstacles and/or boundary intrusions, which are transformed into slits and/or slabs using an elliptic generation system.

With the transformed region made simply-connected we have, using two branch cuts, a configuration related to the O-type shown above for one internal boundary.

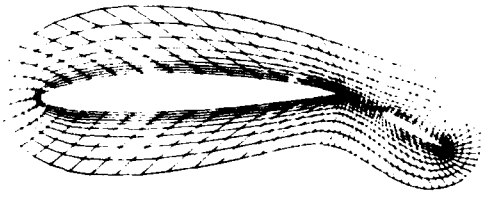
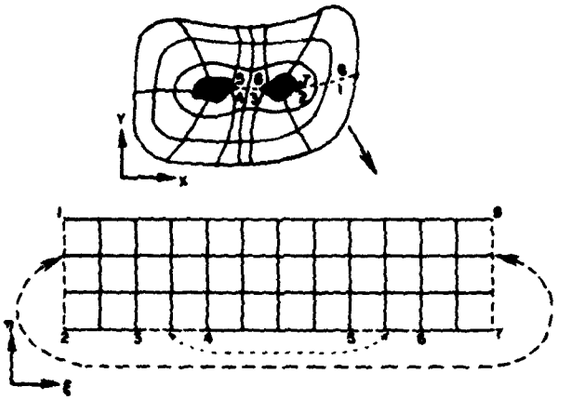


Fig. 16.



Here the pairs 1-2, 8-7 and 3-4, 6-5 are the branch cuts, which form re-entrant boundaries in the transformed region as shown. In this case points outside the right side of the transformed region coincide with points inside the left side, and vice versa. Also, points below the bottom segment 3-4 to the left of the centerline coincide with points above the bottom segment 6-5 an equal distance to the right of the centerline. Again, all derivatives are continuous across both of the cuts.

There are a number of other possibilities for placement of the two cuts on the boundary of the transformed region. The TOMCAT Code of Thompson, et al.,[4] is capable of treating such configurations with any number of interior boundaries and any placement of the boundary segments on the rectangular exterior boundary of the transformed region, using an elliptic generation system. It is not necessary to reduce the connectivity of the region completely; rather, a slit or slab can be used for some of the interior boundaries, while others are placed on the exterior boundary of the transformed region.

In more complicated configurations, one type of coordinate system can be embedded in another. A simple example of this is shown below, where a cylindrical-type of system surrounding an internal boundary is embedded in a system of a more rectangular form:

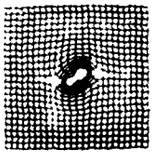
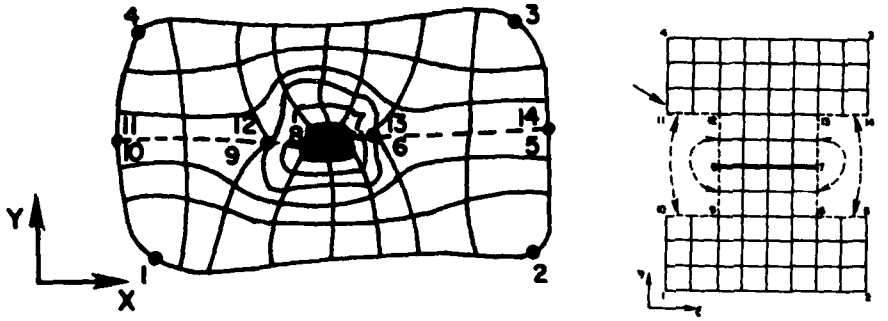


Fig. 17.



Here points below the segment 11-12 are coincident with points below the segment 10-9, and vice versa, with similar correspondence for the pair of segments, 13-14 and 6-5. Points to the left of the segment 8-12 coincide with points to the right of the segment 8-9 located a corresponding distance from 8. Similar correspondence holds for the pair, 7-13 & 7-6. Note that here boundary values are specified on the slit 8-7. The NUMESH code of Coleman[5] is designed to produce embedded systems of this and other types with an elliptic generation system. This code is also discussed in a later paper of this volume.

An example of a C-type system embedded in another C-type system is given next:

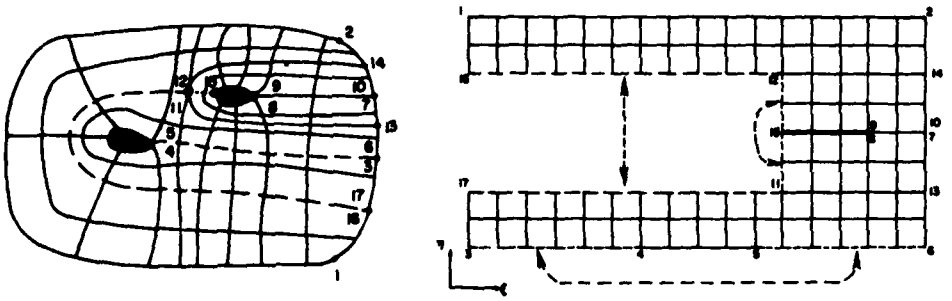


Fig. 18.

Points below 16-12 coincide with points below 17-11 in this case. Points to the left of 15-12 are coincident with points to the right of 15-11 located a corresponding distance from 15. The slit here is formed of the segments 8-15 and 9-15. The coincidents points 11 and 12 here must be taken as a point boundary in the physical plane, i.e., fixed at a specified value. Examples of this type of embedding of coordinates system are given by Sorenson elsewhere in this volume, using extensions of the GRAPE code

(Sorenson[6]) based on an elliptic generation system.

An alternative arrangement of the transformed plane that corresponds to exactly the same coordinate system in the physical plane is as follows:

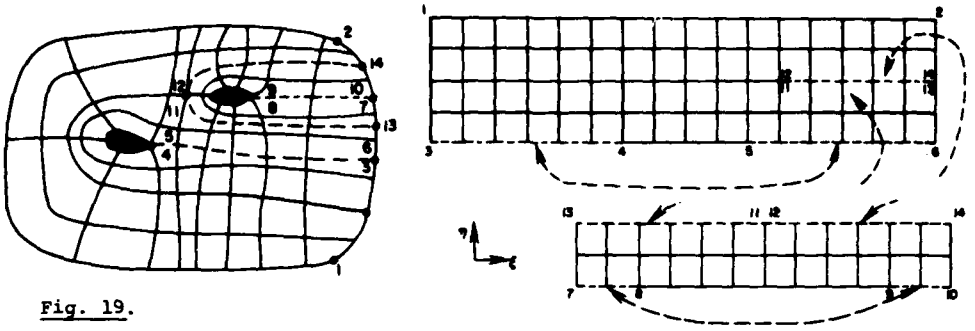


Fig. 19.

Here points below 3-4 coincide with points above 6-5 located a corresponding distance to the right of the centerline. When calculations are made on or above the segment 12-14 on the larger block, points below this segment coincide with points below the corresponding segment on the smaller block. Similarly, when calculations are made on or below the segment 13-11 on the larger block, points above this segment coincide with points below the corresponding segment on the smaller block. Finally, points below the segment 7-8 on the smaller block are coincident with points above the segment 10-9 located a corresponding distance from the centerline. This example illustrates that transformed plane configurations using disjoint, but connected, blocks can be rearranged into a contiguous region which may be more convenient to use.

A more complicated arrangement of cuts, where the species of coordinate changes on a continuous line as the cut is crossed, is illustrated below. The transformed region in this case is composed of three disjoint blocks connected by the cuts.

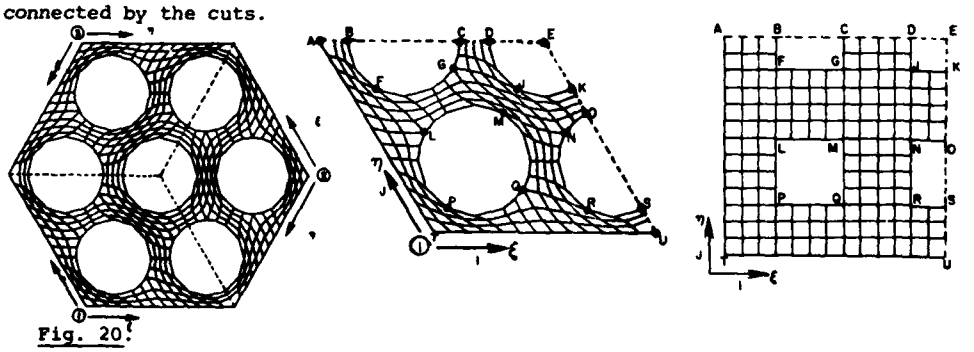


Fig. 20.

Here points outside one section are coincident with corresponding points inside the adjacent section.

As a final configuration for consideration in two dimensions, the following example shows a case with fewer lines on one side of a slab than on the other. It should be noted that this does not necessitate the use of different increments of the curvilinear coordinates in the difference expressions, because these increments always cancel out anyway.

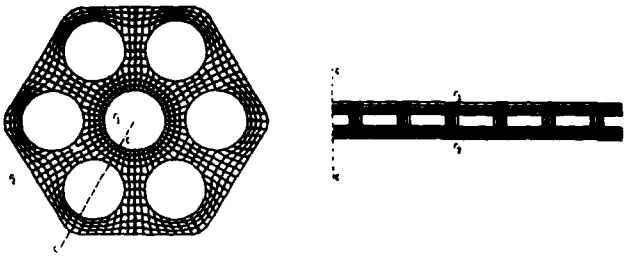


Fig. 21.

Three-dimensional regions

These general concepts illustrated in these examples extend directly to three dimensions. Interior boundaries in the transformed region can become rectangular solids and plates corresponding to the slabs and slits, respectively, illustrated above for two dimensions. An early example of the use of plates was given by Thames in Thompson, et al.[8] The use of three-dimensional configurations comprised of contiguous rectangular solids is illustrated by Coleman in a later paper of this volume. Some three-dimensional regions can be treated by stacking two-dimensional systems generated on planes or surfaces, with corresponding points connected on the successive surface systems.

It is also possible to use branch cuts, as illustrated above for two dimensions, to bring the interior boundaries in the physical region to the exterior boundary of the transformed region. The correspondence between the physical and transformed planes can, however, become much more complicated in three dimensions, and considerable ingenuity may be required to visualize this correspondence. For instance, the simple case of polar coordinates corresponds to a rectangular solid with two opposing sides having the radial coordinate constant thereon, and two re-entrant sides on which the longitude is constant at 0 and 2π, respectively (corresponding to the cut). The remaining two sides correspond to the north and south polar axes, so that an axis opens to cover an entire side. There is thus a line, i.e., the axis, in the physical region that corresponds to an entire side in the transformed region. An example of another arrangement appears below from Mastin and Thompson[9], which is comprised of three disjoint blocks connected by cuts as shown.

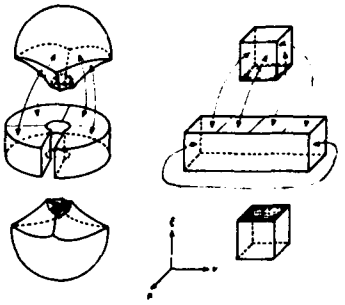


Fig. 22.

SPECIAL GRID POINTS

Several of the configurations discussed above involve boundary points that differ from the usual point formed simply by the intersection of two coordinate lines of different species. Since these points are specified boundary points, they require no special treatment in the generation of the coordinate system with Dirichlet boundary conditions. However, special treatment is necessary at such points with Neumann boundary conditions and in the difference representations for a numerical solution to be done on the coordinate system.

Corners

In the case of a concave (to the field) corner in the transformed plane corresponding to a point on a smooth physical boundary, e.g., points 1, 2, 3, and 4 in Fig. 7, the situation is as illustrated below:

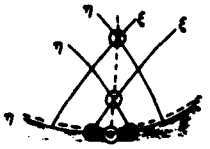


Fig. 23.

Here the problem is that the species of coordinate line changes at the point in question. This case can be treated by considering the dotted lines to represent a local pair of coordinate lines, so that locally the derivatives of one species are taken along the line with the closed circles, while those of the other species are taken along the line with the open circles.

The use of slits in the transformed plane introduces a peculiar point where two lines of the same species meet, e.g., point 9 in Fig. 10:

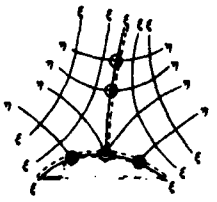


Fig. 24.

Again a logical procedure is to use a local coordinate system formed of the dotted lines, with derivatives of one species being represented locally along the line with the closed circles and those of the other along the line with the open circles. This same situation also occurs for configurations having a boundary segment and a branch cut on the same curvilinear coordinate line, i.e., on the same side of the transformed region as in the C-type configuration of Fig. 14.

A convex corner in the transformed region, such as occurs when rectangular interior boundaries are used therein (points 5, 6, 7, and 8 in Fig. 11) produces the following configuration:

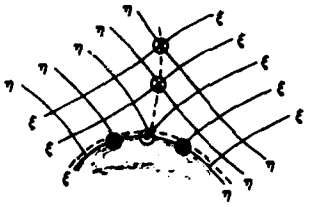


Fig. 25.

Here again a local coordinate system indicated by the dotted lines can be used in the same manner as discussed above.

Branch cuts

Points on re-entrant boundaries in the transformed region, i.e., on branch cuts in the physical region, are not peculiar points in the above sense. Such points, in fact, differ no more from the other field points than do the points on the 0 and 2π lines in a cylindrical coordinate system. Care must be taken, however, to identify the interior points coinciding with the extensions from such points beyond the field. This correspondence was noted above in each of the configurations shown. There are essentially three types of pairs of re-entrant boundaries as shown below in the discussion of derivative correspondence. One exterior point and its corresponding interior point are shown for each case. The converse of the correspondence should be evident in each configuration. Note that while in the first and last cases the coincident points are on the same coordinate line, in the

second case the coincident points are on the same species of line, but at mirror-image distances from the ends of the re-entrant segments.

For the configuration of Fig. 20, involving a change in the coordinate species at the cuts, not only must the directions be taken into account as the cut is crossed but also the coordinate species must be interpreted differently from that established across the cut in order to remain on the same sheet as the cut is crossed. For example, points on an η-line belonging to section 1 but located outside the right side of this region are coincident with points on a ξ-line of region 2 at a corresponding distance below the top of this region.

Derivative correspondence across cuts

Care must be taken at branch cuts to express derivatives of the coordinates correctly in relation to the particular side of the cut on which the site of the computation is located. The existence of branch cuts indicates that the transformed region is multi-sheeted, and computations must remain on the same sheet as the cut is crossed. As noted above, points outside the region across a cut are coincident with points inside the region across the other member of the pair of boundary segments corresponding to the cut in the transformed region. The positive directions of the curvilinear coordinates to be used at these points inside the region across the other member of the pair in some cases are the same as the defined directions there, but in other cases are the opposite directions.

For cuts located on opposing sides of the transformed region, the proper form is simply a continuation across the cut. Thus in the configuration of Fig. 13, with a computation site on the right side of the transformed region, i.e., on the upper side of the cut in the physical plane, we have points to the right of the site (above the cut in the physical plane) coinciding with points to the right of the left side of the transformed region (below the cut in the physical plane) as noted above. When ξ-derivatives and η-derivatives for use to the right of the right side of the transformed region are calculated to the right of the left side, the positive directions of ξ and η are to the right and upward, respectively. This is illustrated below. (In this and the following two figures, the dotted arrows indicate the proper directions to be used at the interior points coincident with the required exterior points, while solid arrows indicate the established directions for the coordinate lines.

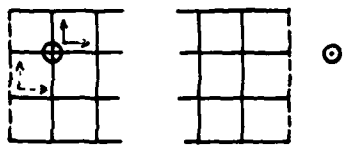


Fig. 26.

With the two sides of the cut both located on the same coordinate line, i.e., on the same side of the transformed region as in the configuration of Fig. 14, however, the situation is not as simple as the above. In this case, when the computation site is on the left branch of the cut in the transformed region (on the lower branch in the physical region), the points below this boundary in the transformed region coincide with points located above the right branch of the cut (above the cut in the physical region), as has been noted earlier. The η-derivatives for use at such points below the left branch are thus calculated at these corresponding points above the right branch. The positive direction of η for purposes of this calculation of derivatives above the right branch for use below the left branch must be taken as downward, not upward. There is a similar reversal in the interpretation of the positive direction of ξ when derivatives are calculated above the right branch for use below the left. These interpretations are illustrated below:

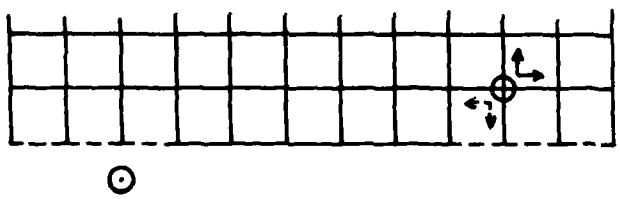


Fig. 27.

Finally, in the configuration of Fig. 17, where the two sides of the cut face each other across a void, there is really no problem of interpretation since the directions in the configuration are treated simply as if the void did not exist. This correspondence is as shown below:

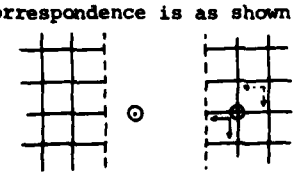


Fig. 28.

In all cases, the interpretation of the positive directions of the curvilinear coordinates must be such as to preserve the direction in the physical region as the cut is crossed. Thus in the case of Fig. 20, where the coordinate species changes at the cut, the situation is more complicated. Here, for example, a ξ-derivative for use in region 1 outside the right side

of that region must be evaluated as a negative η-derivative at a corresponding point below the top of region 2:

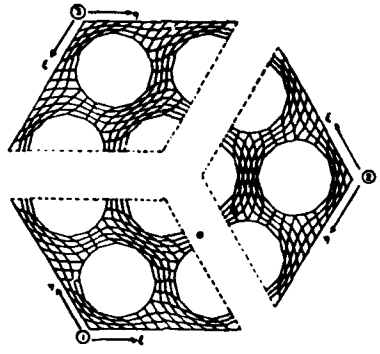


Fig. 29.

TRANSFORMATION RELATIONS

Numerical solutions of partial differential equations on regions of arbitrary shape can be constructed by transforming the equations to the curvilinear coordinate system. All computation then can be done in the transformed region, with the curvilinear coordinates as the independent variables, which is inherently rectangular with a fixed square grid regardless of the shape or motion of the physical boundary. In this section the transformation relations from the cartesian system of the physical plane to a general three-dimensional curvilinear coordinate system are given. In a numerical solution the curvilinear derivatives in these expressions are represented by difference expressions along coordinate lines, as diagrammed below:

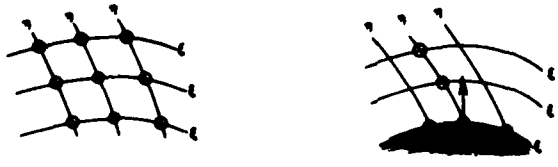


Fig. 30.

All derivative operations may thus be expressed in terms of points on the coordinate lines, using the transformation relations, without need of interpolation. The transformed equations will contain more terms, but the expression of boundary conditions is greatly simplified.

The general framework discussed in this section is set in standard tensor notation. The development of many of the relations given is necessarily omitted in the interest of space, and reference can be made to Eiseman[10] and Warsi[11] for more detail of tensor usage.

The geometric meaning of some of the metric quantities is worth noting here, however: the quantity g_{ij} is proportional to the cosine of the angle between a coordinate line along which the curvilinear coordinate ξ^i varies (the other two coordinates being constant along such a line) and a coordinate line along which ξ^j varies. Thus g_{ij} vanishes for $i \neq j$ for an orthogonal system. The quantity $\sqrt{g_{ii}}$ (no summation) is proportional to the arc length along this coordinate line along which ξ^i varies. Then the arc length along a general curve (not necessarily a coordinate line) is given by

$$(ds)^2 = \sum_i \sum_j g_{ij}\, d\xi^i d\xi^j$$

The quantity $\sqrt{g}$ is the familiar Jacobian of the transformation which measures the volume of a cell in 3D (area in 2D). Finally, the cell aspect ratio, $\sqrt{\frac{g_{ii}}{g_{jj}}}$ (no summation), measures the ratio of the lengths of the cell sides. Throughout the discussions, "ξ^i-line" refers to a line on which the coordinate ξ^i is constant.

Base vectors, tangents, normals, area, and volume

To establish the terminology, consider the following general element bounded by six curved sides, each of which lies on a surface on which one of the curvilinear coordinates ξ^i is constant:

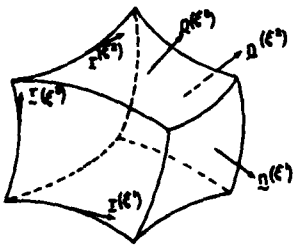


Fig. 31.

Here the unit tangent vector to the line formed by the intersection of surfaces of constant curvilinear coordinate are indicated, as well as the unit normal vectors to such surfaces. The positive direction in each case is in the direction of increasing coordinate values. Note that the tangent, $\underset{\sim}{\tau}^{(\xi^1)}$, is tangent to the line of intersection of the surface of constant ξ^2 and that of constant ξ^3, etc. The normal $\underset{\sim}{n}^{(\xi^1)}$, is perpendicular to the surface of constant ξ^1, etc.

Here the repeated index summation convention will not be used, rather all summations will be explicitly indicated. The values taken by all indices are assumed to be 1, 2, 3, of course.

Base vectors, tangents, and normals. With $\underset{\sim}{r}$ the position vector of a general point (x_1, x_2, x_3), the unit tangent and normal vectors are given by

$$\underset{\sim}{\tau}(\xi^i) = \frac{\dfrac{\partial \underset{\sim}{r}}{\partial \xi^i}}{\left|\dfrac{\partial \underset{\sim}{r}}{\partial \xi^i}\right|} \tag{1}$$

$$\underset{\sim}{n}(\xi^i) = \frac{\dfrac{\partial \underset{\sim}{r}}{\partial \xi^j} \times \dfrac{\partial \underset{\sim}{r}}{\partial \xi^k}}{\left|\dfrac{\partial \underset{\sim}{r}}{\partial \xi^j} \times \dfrac{\partial \underset{\sim}{r}}{\partial \xi^k}\right|} \tag{2}$$

where the indices (i, j, k) occur in cyclic order in Eq.(2). These unit vectors have the directions of the covariant and contravariant base vectors which are defined, respectively, as

$$(\underset{\sim}{a}_i)_j \equiv \frac{\partial x_j}{\partial \xi^i} \tag{3}$$

$$(\underset{\sim}{a}^i)_\ell \equiv \frac{(\underset{\sim}{a}_j \times \underset{\sim}{a}_k)_\ell}{\sqrt{g}} = \frac{\partial \xi^i}{\partial x_\ell} = \frac{\beta_{\ell i}}{\sqrt{g}} \tag{4}$$

where $\sqrt{g}$ is the Jacobian of the transformation,

$$\sqrt{g} = \underset{\sim}{a}_1 \cdot (\underset{\sim}{a}_2 \times \underset{\sim}{a}_3) \tag{5}$$

and, with (ℓ, m, n) cyclic also,

$$\beta_{\ell i} \equiv (\underset{\sim}{a}_j \times \underset{\sim}{a}_k)_\ell = \frac{\partial x_m}{\partial \xi^j}\frac{\partial x_n}{\partial \xi^k} - \frac{\partial x_n}{\partial \xi^j}\frac{\partial x_m}{\partial \xi^k} \tag{6}$$

The elements of the metric tensor are defined in terms of the base vectors as

$$g_{ij} \equiv \underset{\sim}{a}_i \cdot \underset{\sim}{a}_j \tag{7}$$

It is convenient to introduce also the elements of the inverse of the metric tensor:

$$g^{\ell i} = \frac{1}{g} \sum_r \beta_{r\ell}\beta_{ri} = \frac{1}{g}(g_{mj}g_{nk} - g_{mk}g_{nj}) \tag{8}$$

The arc length along the ξ^i line is

$$ds = \left|\frac{\partial \underset{\sim}{r}}{\partial \xi^i}\right| = \sqrt{g_{ii}} \tag{9}$$

The unit tangent and normal may then be written as

$$(\underset{\sim}{\tau}^{(\xi^i)})_j = \frac{(\underset{\sim}{a}_i)_j}{\sqrt{g_{ii}}} = \frac{1}{\sqrt{g_{ii}}} \frac{\partial x_j}{\partial \xi^i} \tag{10}$$

$$(\underset{\sim}{n}^{(\xi^i)})_j = \frac{(\underset{\sim}{a}^i)_j}{\sqrt{g^{ii}}} = \frac{\beta_{ji}}{\sqrt{gg^{ii}}} \tag{11}$$

so that the unit tangent and normal have the directions of the covariant and contravariant base vectors, respectively.

Area. The area of a side of the element lying on a surface of constant ξ^i is

$$dS^{(\xi^i)} = \left|\frac{\partial \underset{\sim}{r}}{\partial \xi^j} \times \frac{\partial \underset{\sim}{r}}{\partial \xi^k}\right| d\xi^j d\xi^k = \sqrt{gg^{ii}}\, d\xi^j d\xi^k \tag{12}$$

Volume. The volume of the element is given by

$$dV = \frac{\partial \underset{\sim}{r}}{\partial \xi^i} \cdot \left(\frac{\partial \underset{\sim}{r}}{\partial \xi^j} \times \frac{\partial \underset{\sim}{r}}{\partial \xi^k}\right) d\xi^i d\xi^j d\xi^k = \sqrt{g}\, d\xi^1 d\xi^2 d\xi^3 \tag{13}$$

Divergence, gradient, curl and Laplacian

With these relations, the Divergence Theorem, $\iiint \underset{\sim}{\nabla} \cdot \underset{\sim}{A} dV = \iint \underset{\sim}{A} \cdot \underset{\sim}{n} dS$, can be applied to the element as follows. Thus

$$\iiint (\underset{\sim}{\nabla} \cdot \underset{\sim}{A})\sqrt{g}\, d\xi^1 d\xi^2 d\xi^3 = \sum_i \left(\iint_{(\xi^i)^+} \underset{\sim}{A} \cdot \underset{\sim}{n}^{(\xi^i)} dS^{(\xi^i)} - \iint_{(\xi^i)^-} \underset{\sim}{A} \cdot \underset{\sim}{n}^{(\xi^i)} dS^{(\xi^i)} \right)$$

where the notation $(\xi^i)^+$ and $(\xi^i)^-$ indicatesthe corresponding sides of the elements that are located at the larger and smaller values of ξ^i, respectively.

Substitution for $\underset{\sim}{n}^{(\xi^i)}$ and $dS^{(\xi^i)}$ from Eq.(11) and (12) then yields

$$\iiint (\nabla \cdot \underset{\sim}{A}) \sqrt{g}\, d\xi^1 d\xi^2 d\xi^3 =$$

$$\sum_i \left(\iint_{(\xi^i)^+} \underset{\sim}{A} \cdot \underset{\sim}{a}^i \sqrt{g}\, d\xi^j d\xi^k - \iint_{(\xi^i)^-} \underset{\sim}{A} \cdot \underset{\sim}{a}^i \sqrt{g}\, d\xi^j d\xi^k \right)$$

where (i, j, k) are cyclic.

Divergence. In the limit, this reduces to the general geometrically conservative expression for the divergence in the curvilinear coordinates:

$$\nabla \cdot \underset{\sim}{A} = \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi^i} (\sqrt{g}\, \underset{\sim}{A} \cdot \underset{\sim}{a}^i) = \frac{1}{\sqrt{g}} \sum_i \sum_j \frac{\partial}{\partial \xi^i} (\beta_{ji} A_j) \tag{14}$$

The geometrically non-conservative form of the divergence is obtained from Eq.(14) by expanding the derivative and noting that

$$\sum_i \frac{\partial}{\partial \xi^i} (\sqrt{g}\, \underset{\sim}{a}^i)_\ell = \sum_i \frac{\partial \beta_{\ell i}}{\partial \xi^i} = 0 \tag{15}$$

Thus the non-conservative form of the divergence is

$$\nabla \cdot \underset{\sim}{A} = \sum_i \underset{\sim}{a}^i \cdot \frac{\partial \underset{\sim}{A}}{\partial \xi^i} = \frac{1}{\sqrt{g}} \sum_i \sum_j \beta_{ji} \frac{\partial A_j}{\partial \xi^i} \tag{16}$$

Gradient. With $A_\ell \equiv f$ and $A_m = A_n = 0$, with (ℓ, m, n) cyclic, we have from Eq.(14) the conservative expression for the gradient:

$$(\nabla f)_\ell = \frac{\partial f}{\partial x_\ell} = \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi^i} (\beta_{\ell i} f) \tag{17}$$

and from Eq.(16) the corresponding non-conservative form

$$(\nabla f)_\ell = \frac{1}{\sqrt{g}} \sum_i \beta_{\ell i} \frac{\partial f}{\partial \xi^i} \tag{18}$$

It may be noted from Eq.(18), with $f = \xi^j$, that

$$\frac{\partial \xi^j}{\partial x_\ell} = \sum_i (\underset{\sim}{a}^i)_\ell \delta_{ij} = (\underset{\sim}{a}^j)_\ell = \frac{1}{\sqrt{g}} \beta_{\ell j} \tag{19}$$

which again identifies the contravariant base vectors as normals to the coordinate surfaces.

Curl. The conservative and non-conservative forms for the curl then are

$$(\nabla \times \underset{\sim}{A}) = \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi^i} (\sqrt{g}\, \underset{\sim}{a}^i \times \underset{\sim}{A}) \tag{20}$$

$$(\nabla \times \underset{\sim}{A})_\ell = \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi^i} (\beta_{mi} A_n - \beta_{ni} A_m)$$

and

$$(\nabla \times \underset{\sim}{A})_\ell = \frac{1}{\sqrt{g}} \sum_i (\beta_{mi} \frac{\partial A_n}{\partial \xi^i} - \beta_{ni} \frac{\partial A_m}{\partial \xi^i}) \tag{21}$$

Laplacian. The forms for the Laplacian then are obtained by substituting $\underset{\sim}{A} = \nabla f$ from Eq.(17) or (18) into Eq.(14) or (16), respectively. Thus the conservative form is

$$\nabla^2 f = \frac{1}{\sqrt{g}} \sum_i \sum_j \sum_k \frac{\partial}{\partial \xi^i} [\frac{1}{\sqrt{g}} \beta_{ki} \frac{\partial}{\partial \xi^j} (\beta_{kj} f)] \tag{22}$$

With the derivatives expanded the non-conservative form is

$$\nabla^2 f = \sum_i \sum_j g^{ij} \frac{\partial^2 f}{\partial \xi^i \partial \xi^j} + \sum_i (\nabla^2 \xi^i) \frac{\partial f}{\partial \xi^i}$$

with

$$\nabla^2 \xi^i = \frac{1}{\sqrt{g}} \sum_j \sum_k \beta_{kj} \frac{\partial}{\partial \xi^j} (\frac{1}{\sqrt{g}} \beta_{ki}) \tag{23}$$

Normal and tangential derivatives, integrals

Normal derivative. From Eq.(11) and (17), the geometrically conservative expression for the normal derivative on a ξ^i surface is

$$(\frac{\partial f}{\partial n})(\xi^i) = \underset{\sim}{n}(\xi^i) \cdot \nabla f$$

$$= \frac{1}{g\sqrt{g_{ii}}} \sum_k \sum_\ell \beta_{ki} \frac{\partial}{\partial \xi^\ell} (\beta_{k\ell} f) \tag{24}$$

while from Eq.(18) the corresponding non-conservative expression is

$$(\frac{\partial f}{\partial n})(\xi^i) = \frac{1}{\sqrt{g_{ii}}} \sum_\ell g^{i\ell} \frac{\partial f}{\partial \xi^\ell} \tag{25}$$

Tangential derivative. The tangential derivative on a ξ^i line in geometrically conservative form is, using Eq.(10) and (17),

$$(\frac{\partial f}{\partial \tau})(\xi^i) = \underset{\sim}{\tau}(\xi^i) \cdot \underset{\sim}{\nabla} f = \frac{1}{\sqrt{g}\sqrt{g_{ii}}} \sum_k \sum_\ell \frac{\partial x_k}{\partial \xi^i} \frac{\partial}{\partial \xi^\ell} (\beta_{k\ell} f) \tag{26}$$

and in non-conservative form,

$$(\frac{\partial f}{\partial \tau})(\xi^i) = \frac{1}{\sqrt{g_{ii}}} \frac{\partial f}{\partial \xi^i} \tag{27}$$

Line integral. The line integral along an increment of a ξ^i line is

$$\int_s T_{kj} dx_\ell = \sum_i T_{kj} \frac{\partial x_\ell}{\partial \xi^i} \Delta \xi^i \tag{28}$$

Surface integral. Returning to the Divergence Theorem, it follows from Eq.(14) that in the limit the surface integral is expressed in conservative form, with A replaced by a tensor, as

$$\sum_j \iint_S T_{kj} n_j dS = \sum_i \sum_j \frac{\partial}{\partial \xi^i} [\beta_{ji} T_{kj}] \Delta \xi^1 \Delta \xi^2 \Delta \xi^3 \tag{29}$$

and the non-conservative form is

$$\sum_j \iint_S T_{kj} n_j dS = \sum_i \sum_j \beta_{ji} \frac{\partial T_{kj}}{\partial \xi^i} \tag{30}$$

Volume integral. From Eq.(13), the volume integral is simply

$$\iiint_V T_{kj} dV = T_{kj} \sqrt{g}\, \Delta \xi^1 \Delta \xi^2 \Delta \xi^3 \tag{31}$$

Two-dimensional relations

In two dimensions we have, with $\underset{\sim}{k}$ the unit vector in the direction of invariance and ξ^3 the curvilinear coordinate in this direction,

$$\frac{\partial \underset{\sim}{r}}{\partial \xi^3} = \underset{\sim}{k}$$

Metric elements. Then with $\xi \equiv \xi^1$, $\eta \equiv \xi^2$ and $x \equiv x^1$, $y \equiv x^2$ for convenience, we have

$$g_{11} = x_\xi^2 + y_\xi^2 \ , \ g_{22} = x_\eta^2 + y_\eta^2 \ , \ g_{33} = 1$$

$$g_{12} = g_{21} = x_\xi x_\eta + y_\xi y_\eta \ , \ g_{13} = g_{31} = g_{23} = g_{32} = 0$$

Then

$$\sqrt{g} = x_\xi y_\eta - x_\eta y_\xi \text{ and } \beta_{11} = y_\eta \ , \ \beta_{22} = x_\xi \ , \ \beta_{33} = \sqrt{g} \ , \ \beta_{12} = -y_\xi$$

$$\beta_{21} = -x_\eta \ , \ \beta_{13} = \beta_{31} = \beta_{23} = \beta_{32} = 0$$

so that

$$g^{11} = \frac{g_{22}}{g} \ , \ g^{22} = \frac{g_{11}}{g} \ , \ g^{33} = 1 \ , \ g^{12} = g^{21} = -\frac{g_{12}}{g}$$

$$g^{13} = g^{31} = g^{23} = g^{32} = 0$$

The following two-dimensional forms then result:

Divergence. (conservative)

$$\underset{\sim}{\nabla} \cdot \underset{\sim}{A} = \frac{1}{\sqrt{g}} \left[(y_\eta A_1 - x_\eta A_2)_\xi + (-y_\xi A_1 + x_\xi A_2)_\eta \right] \tag{32}$$

(non-conservative)

$$\underset{\sim}{\nabla} \cdot \underset{\sim}{A} = \frac{1}{\sqrt{g}} \left[y_\eta (A_1)_\xi - x\,(A_2)_\xi - y_\xi (A_1)_\eta + x\,(A_2)_\eta \right] \tag{33}$$

Gradient. (conservative)

$$f_x = \frac{1}{\sqrt{g}} \left[(y_\eta f)_\xi - (y_\xi f)_\eta \right] \tag{34a}$$

$$f_y = \frac{1}{\sqrt{g}} [-(x_\eta f)_\xi + (x_\xi f)_\eta] \tag{34b}$$

(non-conservative)

$$f_x = \frac{1}{\sqrt{g}} (y_\eta f_\xi - y_\xi f_\eta) \tag{35a}$$

$$f_y = \frac{1}{\sqrt{g}} (-x_\eta f_\xi + x_\xi f_\eta) \tag{35b}$$

Curl (conservative)

$$\nabla \times \mathbf{A} = \frac{\mathbf{k}}{\sqrt{g}} [(y_\eta A_2 + x_\eta A_1)_\xi - (y_\xi A_2 + x_\xi A_1)_\eta] \tag{36}$$

(non-conservative)

$$\nabla \times \mathbf{A} = \frac{\mathbf{k}}{\sqrt{g}} [y_\eta (A_2)_\xi + x_\eta (A_1)_\xi - y_\xi (A_2)_\eta - x_\xi (A_1)_\eta] \tag{37}$$

Laplacian (conservative)

$$\begin{aligned} \sqrt{g}\, \nabla^2 f = & \left[\frac{1}{\sqrt{g}} y_\eta [(y_\eta f)_\xi - (y_\xi f)_\eta] \right. \\ & \left. - \frac{1}{\sqrt{g}} x_\eta [-(x_\eta f)_\xi + (x_\xi f)_\eta] \right]_\xi \\ & + \left[- \frac{1}{\sqrt{g}} y_\xi [(y_\eta f)_\xi - (y_\xi f)_\eta] \right. \\ & \left. + \frac{1}{\sqrt{g}} x_\xi [(-x_\eta f)_\xi + (x_\xi f)_\eta] \right]_\eta \end{aligned} \tag{38}$$

(non-conservative)

$$\begin{aligned} \nabla^2 f = & \frac{1}{g} [(x_\eta^2 + y_\eta^2) f_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta) f_{\xi\eta} \\ & + (x_\xi^2 + y_\xi^2) f_{\eta\eta}] + (\nabla^2 \xi) f_\xi + (\nabla^2 \eta) f_\eta \end{aligned} \tag{39}$$

Normal derivative. (conservative)

$$f_{n}(\xi) = \frac{1}{\sqrt{g}\sqrt{x_\eta^2 + y_\eta^2}} [y_\eta[(y_\eta f)_\xi - (y_\xi f)_\eta] \tag{40a}$$

$$- x_\eta[-(x_\eta f)_\xi + (x_\xi f)_\eta]]$$

$$f_{n}(\eta) = \frac{1}{\sqrt{g}\sqrt{x_\xi^2 + y_\xi^2}} [-y_\xi[(y_\eta f)_\xi - (y_\xi f)_\eta] \tag{40b}$$

$$+ x_\xi[-(x_\eta f)_\xi + (x_\xi f)_\eta]]$$

(non-conservative)

$$f_{n}(\xi) = \frac{1}{\sqrt{g}\sqrt{x_\eta^2 + y_\eta^2}} [(x_\eta^2 + y_\eta^2) f_\xi - (x_\xi x_\eta + y_\xi y_\eta) f_\eta] \tag{41a}$$

$$f_{n}(\eta) = \frac{1}{\sqrt{g}\sqrt{x_\xi^2 + y_\xi^2}} [-(x_\xi x_\eta + y_\xi y_\eta) f_\xi + (x_\xi^2 + y_\xi^2) f_\eta] \tag{41b}$$

Tangential derivative. (conservative)

$$f_{\tau}(\xi) = \frac{1}{\sqrt{x_\eta^2 + y_\eta^2}} [x_\xi[(y_\eta f)_\xi - (y_\xi f)_\eta] - y_\xi[(x_\eta f)_\xi - (x_\xi f)_\eta]] \tag{42a}$$

$$f_{\tau}(\eta) = \frac{1}{\sqrt{x_\xi^2 + y_\xi^2}} [x_\eta[(y_\eta f)_\xi - (y_\xi f)_\eta] - y_\eta[(x_\eta f)_\xi - (x_\xi f)_\eta]] \tag{42b}$$

(non-conservative)

$$f_{\tau}(\xi) = \frac{\sqrt{g}}{\sqrt{x_\eta^2 + y_\eta^2}} f_\xi \tag{43a}$$

$$f_{\tau}(\eta) = \frac{\sqrt{g}}{\sqrt{x_\xi^2 + y_\xi^2}} f_\eta \tag{43b}$$

Line integral.

$$\int_s T_{kj}dx = T_{kj}(x_\xi \Delta\xi + x_\eta \Delta\eta) \tag{44a}$$

$$\int_s T_{kj}dy = T_{kj}(y_\xi \Delta\xi + y_\eta \Delta\eta) \tag{44b}$$

Surface integral.

$$\sum_j \iint_S T_{kj}n_j dS = [(y_\eta T_{k1} - x_\eta T_{k2})_\xi - (y_\xi T_{k1} - x_\xi T_{k2})_\eta]\Delta\xi\Delta\eta \tag{45}$$

Volume integral.

$$\iiint_V T_{kj}dV = T_{kj}\sqrt{g}\,\Delta\xi\Delta\eta \tag{46}$$

Time derivatives

With a moving coordinate system the time derivatives transform as follows:

$$\left(\frac{\partial f}{\partial t}\right)_{x_1,x_2,x_3} = \left(\frac{\partial f}{\partial t}\right)_{\xi^1,\xi^2,\xi^3} - \sum_i \left(\frac{\partial f}{\partial x_i}\right)_{x_j,x_k,t} \left(\frac{\partial x_i}{\partial t}\right)_{\xi^1,\xi^2,\xi^3} \tag{47}$$

with (i, j, k) cyclic and $\frac{\partial f}{\partial x_i}$ given by Eq.(17) or (18). Here it must be noted that the partial time derivative on the left is at a fixed point in the physical region, while that on the right is at a fixed point in the transformed region. The movement of the coordinate system is accounted for by the time derivatives of the cartesian coordinates that appear on the right. It is thus possible to do all computation on a fixed grid in the transformed region regardless of the movement of the boundaries and the grid points in the physical region. A number of references to the use of time-dependent coordinate systems is given in Thompson, et al.[1]

CONCLUSION

The material of this paper is applicable to general boundary-conforming coordinate systems, regardless of how such systems are generated. The

configurations and transformation relations given thus can serve as the basis for the use of any of the systems discussed in this volume. Different configurations of the transformed region are naturally more appropriate to different physical problems. The coordinate lines must be concentrated in regions of large variations in the physical solution to be done on the coordinate system, and the ultimate goal is to have this need sensed and fulfilled automatically as a part of the solution, so that the coordinate systems continuously adjust to adapt to the developing physical solution. These topics are discussed specifically elsewhere in this volume. Further discussions of the concepts of tensor and differential geometry as related to coordinate system generation have been given by Eiseman[10] and Warsi[11].

Using the transformation relations given, all derivatives or integrals in a system of equations to be solved for some physical problem can be expressed with the curvilinear coordinates as the independent variables. A numerical solution can then be constructed in which all computation is done on a fixed square grid in the transformed region regardless of the shape or motion of the physical boundaries. Although much of the impetus for the development of boundary-conforming coordinate systems has come from fluid mechanics, the techniques are applicable to field solutions in all areas. A number of such applications are discussed in the papers of this volume. An extensive survey of coordinate system generation techniques and the use thereof in numerical solutions of partial differential equations has recently been given by Thompson, et al.[1]

ACKNOWLEDGMENTS

These developments were made in the effort under the following sponsorship:

Grant NGR 25-001-055, NASA Langley Research Center

Grant AFOSR 80-0185, U. S. Air Force Office of Scientific Research

Contract DACW39-78-C-0054, U. S. Army Engineer Waterways Experiment Station

REFERENCES

1. Thompson, Joe F., Warsi, Zahir U. A., and Mastin, C. Wayne. "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," Journal of Computational Physics, to be published 1982.
2. Thames, F. C., Thompson, J. F., Mastin, C. W. and Walker, R. L. "Numerical Solutions for Viscous and Potential Flow about Arbitrary Two-Dimenional Bodies Using Body-Fitted Coordinate Systems," Journal of Computational Physics, 24, 245 (1977).

3. Thompson, J. F. "WESCOR - Boundary-Fitted Coordinate Systems for General 2D Regions with Obstacles and Boundary Intrusions."
4. Thompson, J. F., Thames, F. C. and Mastin, C. W. "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, 24, 274 (1977).
5. Coleman, R. M. "NUMESH: A Computer Program to Generate Finite Difference Meshes for Arbitrary Doubly-Connected Two-Dimensional Regions," CMLD-77-05, David W. Taylor Naval Ship Research and Development Center (1977).
6. Sorenson, R. L. "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poisson's Equation," NASA TM-81198 (1980).
7. Sha, W. T., Chen, B. C-J, Cha, J. W., Vanka, S. P., Schmitt, R. C., Thompson, J. F., and Doria, M. L. "Benchmark Rod-Bundle Thermal-Hydraulic Analysis Using Boundary Fitted Coordinates," 1979 Winter Meeting of the American Nuclear Society, San Francisco (1979).
8. Thompson, J. F., Thames, F. C., Shanks, S. P., Reddy, R. N., and Mastin, C. W. "Solutions of the Navier-Stokes Equations in Various Flow Regimes on Fields Containing any Number of Arbitrary Bodies Using Boundary-Fitted Coordinate Systems," Lecture Notes in Physics, 59, 421 (1976).
9. Mastin, C. W. and Thompson, J. F. "Three-Dimensional Body-Fitted Coordinate Systems for Numerical Solution of the Navier-Stokes Equations," AIAA 78-1147, AIAA 11th Fluid and Plasma Dynamics Conference, Seattle (1978).
10. Eiseman, P. R. "Geometric Methods in Computational Fluid Dynamics," ICASE 80-11, NASA Langley Research Center (1980).
11. Warsi, Z. U. A. "Tensors and Differential Geometry Applied to Analytic and Numerical Coordinate Generation," MSSU-EIRS-81-1, Mississippi State University (1981).

AD P000967

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ERROR INDUCED BY COORDINATE SYSTEMS*

C. WAYNE MASTIN
Department of Mathematics and Statistics, Mississippi State University, Drawer MA, Mississippi State, Mississippi.

INTRODUCTION

The choice of a curvilinear coordinate system can have a substantial effect on the error in the numerical solution of a partial differential equation. The truncation error is dependent not only on the higher order derivatives of the solution and the local grid spacing, but also on the rate-of-change of the grid spacing and on the departure of the grid from orthogonality. The effect of a nonuniform grid in one-dimension was analyzed by Kalnay de Rivas [1]. That analysis explained the reason for the poor results obtained by Crowder and Dalton[2] when the grid spacing changed suddenly. The coordinate system influences the smoothness as well as the accuracy of the numerical solution. This fact is evident by recalling the general principal that the smoothness of the solution of a partial differential equation depends on the smoothness of the coefficients.

The coefficients of the equation, in terms of curvilinear coordinates, depend on the derivatives of the functions defining the coordinate system. Examples where lack of smoothness can be traced to the coordinate system appear in the papers by Shang[3], Dwyer, Kee, and Sanders[4] and McCrory and Orszag[5].

This report will analyze the local truncation error in the approximation of first and second order derivatives on a curvilinear grid. Standard second order central differences have been used. An analogous development could be carried out using one-sided or higher order difference approximations. While only two-dimensional grids are considered most results can be extended in an obvious manner to three dimensions.

DERIVATIVE APPROXIMATIONS

The traditional approach in deriving difference equations on a curvilinear coordinate system is by applying the chain rule. The derivatives of a function f in terms of the computational $\xi\eta$ - variables are related to the derivatives with respect to the physical xy - variables by the following equations.

---

*Sponsored by NASA Langley Research Center under Grant No. NSG 1577.

$$f_\xi = x_\xi f_x + y_\xi f_y \tag{1}$$

$$f_{\xi\xi} = x_{\xi\xi} f_x + y_{\xi\xi} f_y + x_\xi^2 f_{xx} + 2x_\xi y_\xi f_{xy} + y_\xi^2 f_{yy} \tag{2}$$

$$f_{\xi\eta} = x_{\xi\eta} f_x + y_{\xi\eta} f_y + x_\xi x_\eta f_{xx} + (x_\xi y_\eta + x_\eta y_\xi) f_{xy} + y_\xi y_\eta f_{yy} \tag{3}$$

Analogous equations hold for f_η and $f_{\eta\eta}$ giving a total of 5 equations from which one can derive expressions for the derivatives of f with respect to x and y in terms of derivatives of f, x, and y with respect to the computational variables ξ and η. The derivatives of f with respect to the computational variables can now be approximated using differences on a uniform rectangular grid. The derivatives of x and y may be computed exactly or approximated, depending on whether the coordinate system is defined analytically or numerically. Since the error analysis is essentially the same in both cases, the subscripts ξ and η may denote either derivatives of differences except where specifically noted.

Suppose a square mesh of unit width is constructed on the computational region. When first order derivatives of f are approximated by central differences, the principal part of the local truncation error for f_ξ is given by

$$\frac{1}{6} f_{\xi\xi\xi}.$$

This derivative can be expanded in terms of physical derivatives. If all third order derivatives are assumed to be small and all other terms retained, we arrive at the following expression for the difference approximation of f_ξ.

$$f_\xi \doteq x_\xi f_x + y_\xi f_y + \frac{1}{2} x_\xi x_{\xi\xi} f_{xx} + \frac{1}{2}(x_\xi y_{\xi\xi} + y_\xi x_{\xi\xi}) f_{xy} + \frac{1}{2} y_\xi y_{\xi\xi} f_{yy} \tag{4}$$

In this equation, and the second order difference approximations to follow, the subscript on the left denotes a difference rather than a derivative. Thus, it is easily shown that the use of (1) in deriving difference approximations results in a truncation error which is O(h), where h is the local grid spacing, provided the following condition is satisfied.

$$\frac{h^2}{J} = O(1)$$

where

$$J = x_\xi y_\eta - x_\eta y_\xi.$$

Note that the validity of this result depends on the first and second order derivatives of x and y being $O(h)$ and the third order derivatives being $O(h^2)$. This restriction does not apply when differences of x and y are used.

A similar expansion for the second order difference approximations $f_{\xi\xi}$ and $f_{\xi\eta}$ can be derived by considering equation (2) and the principal truncation error. Again neglecting terms with third or higher derivatives, we arrive at

$$f_{\xi\xi} \approx x_{\xi\xi}f_x + y_{\xi\xi}f_y + (\tfrac{1}{4}x_{\xi\xi}^2 + x_\xi^2)f_{xx} + (\tfrac{1}{2}x_{\xi\xi}y_{\xi\xi} + 2x_\xi y_\xi)f_{xy} + (\tfrac{1}{4}y_{\xi\xi}^2 + y_\xi^2)f_{yy}, \tag{5}$$

$$f_{\xi\eta} \approx x_{\xi\eta}f_x + y_{\xi\eta}f_y + (x_\xi x_\eta + \tfrac{1}{2}x_{\xi\eta}\nabla^2 x)f_{xx} + (x_\xi y_\eta + x_\eta y_\xi + \tfrac{1}{2}x_{\xi\eta}\nabla^2 y + \tfrac{1}{2}y_{\xi\eta}\nabla^2 x)f_{xy} + (y_\xi y_\eta + \tfrac{1}{2}y_{\xi\eta}\nabla^2 y)f_{yy}. \tag{6}$$

These same relations hold whether derivatives or differences are used provided the following approximation for the Laplacian is understood in (6).

$$\nabla^2 x(\xi,\eta) = \tfrac{1}{2}[x(\xi+1,\eta+1) + x(\xi+1,\eta-1) + x(\xi-1,\eta+1) + x(\xi-1,\eta-1) - 4x(\xi,\eta)]$$

The approximations (5) and (6) presume that third and fourth order derivatives of x and y are of order $O(h^2)$ and $O(h^3)$, respectively. If differences are used (5) holds without restrictions while certain third order differences must be $O(h^2)$ in (6). These approximations imply that the use of (1), (2), and (3) in deriving second order derivative approximations results in a truncation error of $O(1)$.

From the above analysis, it is observed that unless care is exercised in selecting the coordinate system, the usual difference methods for first order partial differential equations will be only first order accurate and those for second order equations will be inconsistent. However, the truncation error for first and second order derivatives can be increased to $O(h^2)$ and $O(h)$, respectively, if the second order derivatives of x and y are assumed to be $O(h^2)$. This effectively limits the rate of change in coordinate line spacing and the curvature of coordinate lines. Alternately, equations (4), (5), and (6) can be used to derive difference approximations for the derivatives of f with the same accuracy. It also follows that truncation errors of $O(h^2)$ and $O(h)$ are the best that can be achieved for first and second order derivatives using only a nine

point difference molecule on an arbitrary grid.

The major factor in the selection of a curvilinear coordinate system is the shape of the physical region. Based on our analysis, other properties would be desireable. From equation (5) one would certainly desire the condition

$$x_{\xi\xi}^2 + y_{\xi\xi}^2 < 4(x_\xi^2 + y_\xi^2)$$

which has a simple geometric interpretation when differences are assumed. This means that the angle formed by connecting the successive points

$$(x(\xi - 1, \eta), y(\xi - 1, \eta))$$
$$(x(\xi,\eta), y(\xi,\eta))$$
$$(x(\xi + 1, \eta), y(\xi + 1, \eta))$$

is greater than 90 degrees. This condition could be checked by inspection of the grid and would most likely be avoided by using a smooth coordinate system and clustering grid points near re-entrant boundary points of the physical region. A more restrictive upper bound on the second order derivatives may be more difficult to enforce. For example, if we consider the simple exponential mapping

$$x = e^{\xi},$$

then

$$x_\xi = x_{\xi\xi}.$$

The grid spacing with this one-dimensional mapping increases by a factor of e at each step. Such rapid expansion of grid spacing should only be used in regions where it is know that the second order partial derivatives of the solution will be relatively small.

We will now mention briefly some similar results which are valid for other commonly used difference approximations. When solving second order equations, terms of the following type are often encountered.

$$(uf_x)_x, (uf_x)_y, (uf_y)_x, (uf_y)_y$$

Each of these can be expressed in terms of computational derivatives by formulas

like

$$(uf_x)_x = \frac{y_\eta}{J}\left[\left(\frac{y_\eta}{J}uf_\xi\right)_\xi - \left(\frac{y_\xi}{J}uf_\eta\right)_\xi\right] - \frac{y_\xi}{J}\left[\left(\frac{y_\eta}{J}uf_\xi\right)_\eta - \left(\frac{y_\xi}{J}uf_\eta\right)_\eta\right].$$

The second order differences needed to form the appropriate difference equation can be expanded by a Taylor series to arrive at formulas similar to (5) and (6). It also follows that the difference approximations of the four second order derivatives may be inconsistent but will be O(h) whenever the second order derivatives of x and y are $O(h^2)$.

Another type of differencing is frequently encountered when solving conservation law equations of the type

$$f_x + g_y = s.$$

The divergence form of the equation is retained in the computational region when it is written as

$$(fy_\eta - gx_\eta)_\xi + (gx_\xi - fy_\xi)_\eta = Js.$$

However, when the computational derivatives are replaced by differences with respect to ξ and η, a truncation error analysis reveals the following approximation.

$$f_x + g_y = \frac{1}{J}\left[(fy_\eta - gx_\eta)_\xi + (gx_\xi - fy_\xi)_\eta\right]$$
$$+ \left(1 - \frac{J^*}{J}\right)f_x + \left(1 - \frac{J^{**}}{J}\right)g_y - \frac{Z^*}{J}g_x - \frac{Z^{**}}{J}f_y + O(h)$$

where

$$J^* = (xy_\eta)_\xi - (xy_\xi)_\eta$$

$$J^{**} = (yx_\xi)_\eta - (yx_\eta)_\xi$$

$$Z^* = (xx_\xi)_\eta - (xx_\eta)_\xi$$

$$Z^{**} = (yy_\eta)_\xi - (yy_\xi)_\eta$$

This expansion is primarily of interest when the coordinate derivatives are

computed numerically, for if the actual derivatives of x and y are inserted, then $J = J^* = J^{**}$ and $Z^* = Z^{**} = 0$. It is clear that the difference between J and J^* (or J^{**}) may be considerable since their evaluation requires different sets of grid points. One only needs to compare values at a point on a ξ = constant coordinate line where there is a large jump in spacing of the ξ = constant lines.

NONORTHOGONALITY

The degree to which nonorthogonality increases the local truncation error is determined by the value of the Jacobian of the mapping. When the derivatives of f with respect to ξ and η are approximated by the differences in (1) and the corresponding equation for f_η, the principal truncation error for f_x, is given by

$$T_x = -\frac{1}{6J}(y_\eta f_{\xi\xi\xi} - y_\xi f_{\eta\eta\eta}).$$

In order to isolate the effect of nonorthogonality, it will be assumed that the local coordinate system is simply a sheared rectangular coordinate system with uniform spacing of h in the ξ-direction and k in the η-direction. Let ϕ and θ denote the angles of inclination of the η = constant and ξ = constant coordinate lines. Under these assumptions it is clear that

$$T_x = -\frac{1}{6\sin(\theta-\phi)}\left[h^2 \sin\theta\left(\cos\theta\frac{\partial}{\partial x} + \sin\phi\frac{\partial}{\partial y}\right)^3 f\right.$$

$$\left. - k^2 \sin\phi\left(\cos\theta\frac{\partial}{\partial x} + \sin\theta\frac{\partial}{\partial y}\right)^3 f\right].$$

Therefore, the factor which causes an increase in truncation error due to nonorthogonoality is $1/\sin(\theta-\phi)$. This is exactly the factor by which the Jacobian is decreased by the shearing of the coordinate lines. Another observation is worth noting. If $\theta, \phi \to \pi/2$, then $|T_x| \to \infty$. This is to be expected since one cannot accurately measure the rate of change of f in the x-direction if there is practically no change in x along either coordinate line. The same development carries over to second order derivatives. In that case, the nonorthogonality introduces a factor of $[\sin(\theta-\phi)]^{-2}$ into the truncation error. In conclusion, a slight degree of nonorthogonality has a negligible effect on truncation error. The rate of increase in truncation error does, however, increase with the degree of nonorthogonality.

EXAMPLES

There is, at present, no completely satisfactory method for estimating the truncation error in the finite difference solution of an arbitrary partial differentia' equation. The estimates used here were calculated from the numerical solution with the third and fourth order derivatives approximated by the standard five-point difference formulas. Consequently, these truncation error estimates often exhibit large point-to-point variations due to the inherent instability in the numerical approximation of higher order derivatives. An alternate method of analyzing truncation error would be to solve the problem on a course and a fine grid. However, this would require considerably more work unless one were using a multi-grid algorithm. It should also be emphasized that the local truncation error does not necessarily represent the actual error in the numerical solution. It will, at best, distinguish regions where the error is larger.

The first example is the numerical solution of Laplace's equation for potential flow about a circular cylinder. The results obtained using the three grids in Figure 1 illustrate the effect of the grid on the accuracy and smoothness of the numerical solution and on the values of the truncation error estimates. The uniform grid (A) gave a smooth approximation of the solution with the maximum absolute values of the error and truncation error estimates occurring at the leading edge as indicated in Figures 2 and 3. Moving grid points into the region near the leading edge, as in grid (B), results in a reduction in both actual and truncation error. Further distortion, as with grid (C), improves accuracy but at the cost of smoothness in the numerical solution. The truncation error was not plotted for this case since the spike in the graph that occurred with grid (B) extended far beyond the bounds of Figure 3 in the case of grid (C). Even though the estimates were of doubtful accuracy, the computed truncation error did perform as anticipated in the above analysis. For this example, the smoothness of the numerical solution must be balanced with accuracy since partial derivatives have to be computed to arrive at values for the pressure and velocity components.

The next example demonstrates the impact of the solution on truncation error as well as grid effects. The stream function - vorticity equations for steady-state viscous flow about a circular cylinder were solved on the two grids indicated in Figure 4. At a Reynold's number of 20, the boundary layer is sufficiently thin so as to produce a large radial variation in vorticity between the forward stagnation point and the point of separation. The choice of a grid can have a measurable influence on the numerical solution as depicted in Figure 5. When compared with the values in Dennis and Chang[6], grid (B) gives more

accurate vorticity values on the forward part of the cylinder. However, the truncation error estimates for both grids were quite large as can be seen in Figure 6. The larger values with grid (B) are a result of the grid points being closer to the surface. When interpolated at common points, both grids gave essentially the same truncation error estimate. It therefore appears that the expected decrease in truncation error resulting from a finer grid is nullified by a more rapid expansion of grid line spacing. Note that changes in grid spacing would have a major influence on the truncation error since the second order derivatives of the vorticity are very large near the surface. In fact, any further contraction of grid lines near the surface resulted in a loss of accuracy and much larger truncation error estimates. When compared with previous numerical and experimental results, grid (B) proved to possess a near optimal distribution of circular coordinate lines for this problem. No results are presented for values on the aft portion of the cylinder since neither grid could adequately model the vortex at the rear of the cylinder.

No smoothing has been used in the computation of truncation error estimates so that the extreme values of the estimates are illustrated. Nonphysical oscillations in numerical solutions, like those encountered in problems with high Reynold's numbers or shocks, would need to be eliminated by smoothing before truncation error estimates are calculated.

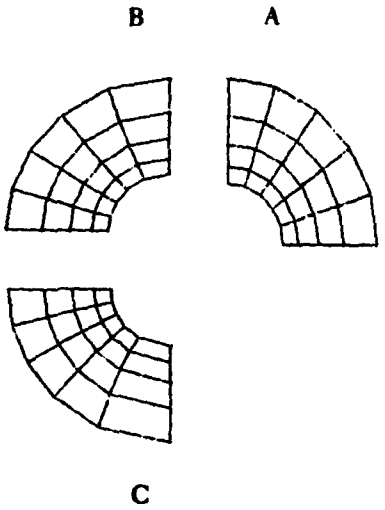


Fig. 1. Quadrants from three grids with different angular distribution of grid points.

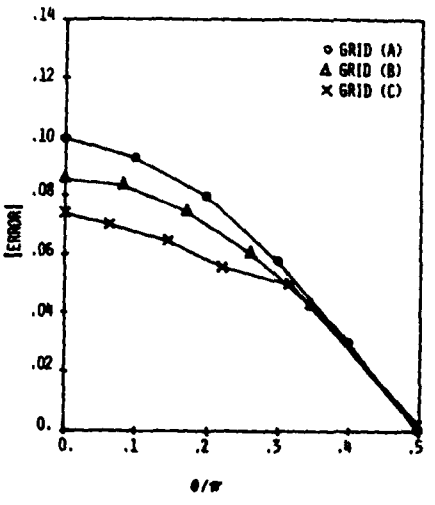


Fig. 2. Absolute value of error in surface potential measured from leading edge.

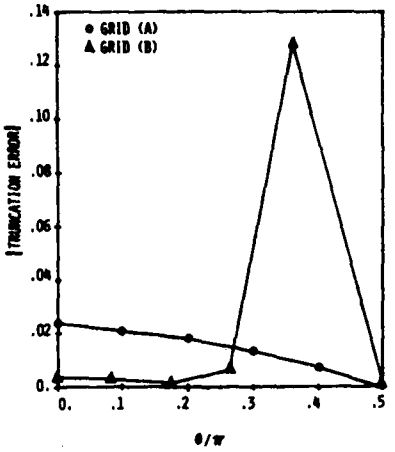


Fig. 3. Absolute value of truncation error estimate at one grid point off the cylinder.

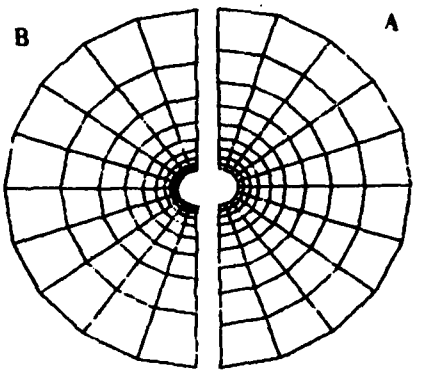


Fig. 4. Two grids with different radial distribution of grid points.

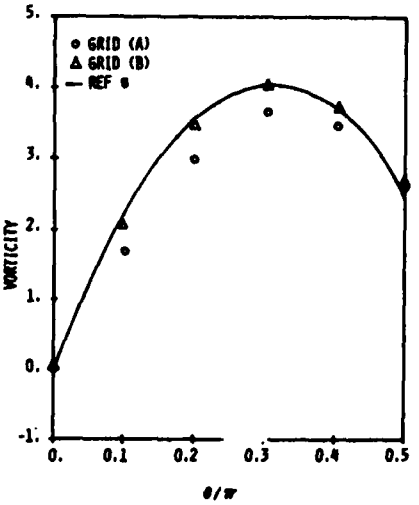


Fig. 5. Comparison of surface vorticity on forward part of cylinder measured from leading edge.

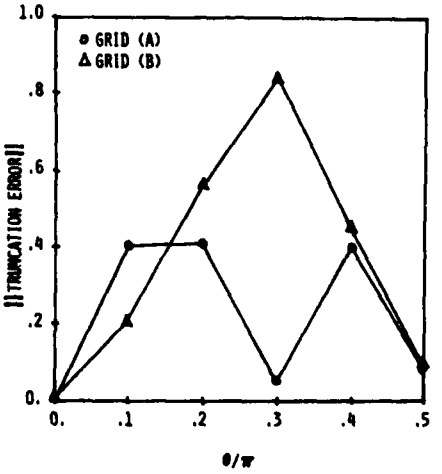


Fig. 6. Norm of truncation error estimate for the stream function and vorticity equations calculated one grid point off the cylinder.

CONCLUSIONS

These examples illustrate the two fundamental sources of truncation error in the numerical solution of partial differential equations on curvilinear coordinate systems. The first is the grid spacing and changes in grid spacing which is measured by the first and second order derivatives (or differences) of the

functions defining the coordinate system. The second source is the higher order derivatives of the solution itself. Large third and fourth order derivatives of the solution must be offset by a fine mesh if accurate results are expected. While the grid effects can be computed precisely, the values of the solution derivatives must be approximated from the numerical solution. Conventional wisdom might cause one to be skeptical of truncation error estimates based on the numerical approximation of higher order derivatives, however, this procedure was successfully used by Pierson and Kutler[7] in a one-dimensional grid generation algorithm. There is still a great need for further work on the accurate estimation of local truncation error and its use in multi-dimensional grid generation algorithms.

REFERENCES

1. Kalnay de Rivas, E. (1972) J. Comput. Phys. 10, 202-210.
2. Crowder, H. J. and Dalton, C. (1971) J. Comput. Phys. 7, 32-45.
3. Shang, J. S. (1981) AIAA-81-0048, AIAA 19th Aerospace Sciences Meeting, St. Louis.
4. Dwyer, H. A., Kee, R. J. and Sanders, B. R. (1980) AIAA J. 18, 1205-1212.
5. McCrory, R. L. and Orszag, S. A. (1980) J. Comput. Phys. 37, 93-112.
6. Dennis, S. C. R. and Chang, G. Z. (1970) J. Fluid. Mech. 42, 471-489.
7. Pierson, B. L. and Kutler, P. (1979) AIAA-79-0272, AIAA 17th Aerospace Sciences Meeting, New Orleans.

AD P000968

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

BASIC DIFFERENTIAL MODELS FOR COORDINATE GENERATION

Z. U. A. WARSI‡
Department of Aerospace Engineering, Mississippi State University, Drawer A, Mississippi State, Mississippi 39762, USA

CONTENTS

§1. Introduction
§2. Notation and Basic Formulas
§3. Generating Differential Equations Based on Gauss Equations
§4. Generating Differential Equations Based on Laplace Equations
§5. Generating Differential Equations Based on the Riemann Tensor

§1. INTRODUCTION

This paper examines in detail the analytical aspects of three distinct methods of coordinate generation based on partial differential equations, in either two or three dimensions. The first method is based on the Gauss equations of a surface under the constraint of the Beltrami's second order equations. These equations have been structured in such a way that an automatic connection is established between the succeeding generated surfaces. The second method is a re-examination of those equations which are based on the inhomogeneous Laplace equations. This analysis reveals a new form for the terms which play a role in the concentration of coordinate lines and in the adaptive coordinate system generation. The third method pertains to a set of equations in the metric coefficients which is obtained by setting the Riemann's curvature tensor to zero.

The problem of generating spatial coordinates by numerical methods is a problem of much interest in practically all branches of engineering and physics. At present a number of techniques are under active development for the generation of two and three-dimensional coordinates in the regions between two or a number of arbitrary shaped bodies. Among these efforts two easily discernable groups can be formed, (i) the methods based on elliptic PDE's, and (ii) algebraic methods. In the first group, a set of inhomogeneous Laplace equations is taken as the basic generating system. These equations are then inverted and solved for the Cartesian coordinates.

---

‡Professor

Some very useful results based on this line of approach started with the work of Winslow[1], have been obtained by Thompson, et al.[2] (TTM method), Steger, et al.[3], Yu[4], Graves[5], and Thomas[6]. For an extensive bibliography refer to Thompson, et al.[7] In the second group of methods, the grid points in space are generated by interpolating and blending functions starting from the given boundary data. This line of approach has been followed by Eiseman[8], Smith, et al.[9], Erickson[10], and others.

In this paper we consider only the analytical aspects of the differential equation's approach to coordinate generation. The main effort here is to present only those results which are of permanent interest to the workers in the field of coordinate generation. The proposed equations in any one of the groups have not been arbitrarily selected to generate some sort of coordinates. These equations are in fact those which every numerically or analytically generated coordinates must satisfy. The reader will find that some large portions of sections 3 and 5 have new results and are based on the work by Warsi[11,12]. In sections 3 and 5 a number of exact solutions have been obtained which can be used to provide a testing ground for different numerical schemes.

§2. NOTATION AND BASIC FORMULAS

In this paper any general curvilinear coordinate system will be denoted by a superscript index notation, such as x^i. However, when an expression has been expanded out in full and there is no need for an index notation then we shall use the symbols

$$x^1 = \xi \ , \ x^2 = \eta \ , \ x^3 = \zeta \ .$$

The rectangular Cartesian coordinates (x, y, z) which determine the position vector $\underset{\sim}{r}$, i.e.,

$$\underset{\sim}{r} = \underset{\sim}{r}(x,y,z)$$

will be denoted by the subscripted variable x_i, where $x_1 = x$, $x_2 = y$, $x_3 = z$.

Two similar indices, one appearing as a subscript and the other as a superscript will always imply summation over the range of index values; e.g., $A_{ij}B^i = A_{1j}B^1 + A_{2j}B^2 + A_{3j}B^3$.

In an Euclidean space (E^2 or E^3), the covariant base vectors $\underset{\sim}{a}_i$ are given by

$$\underset{\sim}{a}_i = \frac{\partial \underset{\sim}{r}}{\partial x^i} \ , \tag{1a}$$

so that

$$\underset{\sim}{a}_1 = \underset{\sim}{r}_\xi \ , \ \underset{\sim}{a}_2 = \underset{\sim}{r}_\eta \ , \ \underset{\sim}{a}_3 = \underset{\sim}{r}_\zeta \ , \tag{1b}$$

where a variable subscript will denote a partial derivative. Using the Riemannian metric, the formula for the length element ds is given by

$$(ds)^2 = g_{ij} dx^i dx^j \ ,$$

where, because of the Euclidean nature of the space the metric coefficients are given by

$$g_{ij} = \underset{\sim}{a}_i \cdot \underset{\sim}{a}_j = \frac{\partial \underset{\sim}{r}}{\partial x^i} \cdot \frac{\partial \underset{\sim}{r}}{\partial x^j} \ . \tag{2a}$$

The coefficients $g_{ij} = g_{ji}$ are the covariant components of the metric tensor. The contravariant components g^{ij} are related with g_{ij} through the equation

$$g^{ij} g_{ik} = \delta^j_k \tag{2b}$$

where δ^j_k (the Kronecker deltas) are the mixed components of the metric tensor. Using (2b) we define the contravariant base vectors as

$$\underset{\sim}{a}^i = g^{ij} \underset{\sim}{a}_j \ . \tag{2c}$$

The quantities g and $\hat{g}$ defin 1 as

$$g = \det(g_{ij}) \ , \tag{3a}$$

$$\hat{g} = \det(g^{ij}) \ , \tag{3b}$$

are related as

$$g\hat{g} = 1 \ . \tag{4}$$

For a three-dimensional space

$$g = g_{11}g_{22}g_{33} + 2g_{12}g_{13}g_{23} - (g_{23})^2 g_{11} - (g_{13})^2 g_{22} - (g_{12})^2 g_{33}. \tag{5}$$

Introducing the quantities,

$$\left.\begin{aligned} G_1 &= g_{22}g_{33} - (g_{23})^2 \ , \\ G_2 &= g_{11}g_{33} - (g_{13})^2 \ , \\ G_3 &= g_{11}g_{22} - (g_{12})^2 \ , \\ G_4 &= g_{13}g_{23} - g_{12}g_{33} \ , \\ G_5 &= g_{12}g_{23} - g_{13}g_{22} \ , \\ G_6 &= g_{12}g_{13} - g_{23}g_{11} \ , \end{aligned}\right\} \tag{6}$$

we have, on solving Eqs. (2b),

$$g^{11} = G_1/g \ , \ g^{22} = G_2/g \ , \ g^{33} = G_3/g \ , \tag{7a}$$

$$g^{12} = G_4/g \ , \ g^{13} = G_5/g \ , \ g^{23} = G_6/g \ . \tag{7b}$$

The space Christoffel symbols of the first and second kind respectively are given by

$$[ij,k] = \frac{1}{2}\left(\frac{\partial g_{ik}}{\partial x^j} + \frac{\partial g_{jk}}{\partial x^i} - \frac{\partial g_{ij}}{\partial x^k}\right) \ , \tag{8a}$$

$$\Gamma^{\ell}_{ij} = g^{k\ell}[ij,k] \ . \tag{8b}$$

Using (8b), we have

$$\frac{\partial \underset{\sim}{a}_i}{\partial x^j} = \Gamma^{\ell}_{ij}\, \underset{\sim}{a}_{\ell} \ . \tag{8c}$$

In the case of a two-dimensional surface embedded in a three-dimensional space, we shall use the Greek indices α, β, etc. (with the exception of ν) with the stipulation that they assume only two values. Thus the surface Christoffel symbols of the first and second kind are respectively given by

$$[\alpha\beta,\delta] = \frac{1}{2}\left(\frac{\partial g_{\alpha\delta}}{\partial x^\beta} + \frac{\partial g_{\beta\delta}}{\partial x^\alpha} - \frac{\partial g_{\alpha\beta}}{\partial x^\delta}\right) \ , \tag{9a}$$

$$\Upsilon^{\sigma}_{\alpha\beta} = g^{\delta\sigma}[\alpha\beta,\delta] \ , \tag{9b}$$

where for the purpose of clarification we have used the symbol Υ (upsilon) to denote the surface Christoffel symbols of the second kind in (9b) and not by Γ as in (8b).

In the process of formulation of a 3D coordinate generation problem, it is helpful to imagine the coordinates of a point in space as the intersection of three distinct surfaces on each of which one coordinate is held fixed. Using the convention of a right-handed coordinate system x^1, x^2, x^3 or ξ, η, ζ, we introduce the notation $\Sigma^{(\nu)}$ as a surface on which the coordinate x^{ν} = const., such that

$\nu = 1$ implies that (x^2, x^3) are in the surface,

$\nu = 2$ implies that (x^3, x^1) are in the surface,

$\nu = 3$ implies that (x^1, x^2) are in the surface.

Thus, the unit normal vector on the surface $\Sigma^{(\nu)}$ is

$$\underset{\sim}{n}^{(\nu)} = (\underset{\sim}{r}_{\alpha} \times \underset{\sim}{r}_{\beta}) / |\underset{\sim}{r}_{\alpha} \times \underset{\sim}{r}_{\beta}| \ , \tag{10}$$

where

$$\left.\begin{aligned} &\nu = 1 : \alpha = 2 \ , \ \beta = 3 \quad (\text{surface } x^1 = \xi = \text{const.}) \ , \\ &\nu = 2 : \alpha = 3 \ , \ \beta = 1 \quad (\text{surface } x^2 = \eta = \text{const.}) \ , \\ &\nu = 3 : \alpha = 1 \ , \ \beta = 2 \quad (\text{surface } x^3 = \zeta = \text{const.}) \ . \end{aligned}\right\} \tag{11}$$

All other quantities and formulas which appear in the rest of the paper have been defined where they first appear. Refer also to Warsi[12] and Eisenhart[13].

§3. GENERATING DIFFERENTIAL EQUATIONS BASED ON GAUSS EQUATIONS

In this section our aim is to develop a method[11] for the generation of 3D coordinates wherein a series of surfaces are generated on each of which two previously designated coordinates vary while the third coordinate remains fixed. This method must also be structured in such a way that the variation

of the third coordinate from one generated surface to the next is fully reflected in the system of generating equations. With this aim, we start from the equations of Gauss[13,14] which for a surface x^ν = const., are given by

$$\underset{\sim}{r}_{\alpha\beta} = \Gamma^{\delta}_{\alpha\beta}\,\underset{\sim}{r}_{\delta} + b_{\alpha\beta}\underset{\sim}{n}^{(\nu)} \quad , \tag{12}$$

where the variations of α, β and the range of δ with ν follows the scheme in (11). The quantities $b_{\alpha\beta}$ are the coefficients of the second fundamental form of the surface. Since on the surface x^ν = const., the vector $\underset{\sim}{n}^{(\nu)}$ is orthogonal to the surface vectors $\underset{\sim}{r}_{\delta}$, hence

$$b_{\alpha\beta} = \underset{\sim}{n}^{(\nu)} \cdot \underset{\sim}{r}_{\alpha\beta} \quad . \tag{13}$$

To fix ideas, we envisage a surface which is formed of the coordinate lines ξ, η and on which ζ = const. Dropping the index ν, Eq. (12) yields the three equations

$$\underset{\sim}{r}_{\xi\xi} = \Gamma^{\delta}_{11}\underset{\sim}{r}_{\delta} + S\underset{\sim}{n} \quad , \tag{14a}$$

$$\underset{\sim}{r}_{\xi\eta} = \Gamma^{\delta}_{12}\underset{\sim}{r}_{\delta} + T\underset{\sim}{n} \tag{14b}$$

$$\underset{\sim}{r}_{\eta\eta} = \Gamma^{\delta}_{22}\underset{\sim}{r}_{\delta} + U\underset{\sim}{n} \tag{14c}$$

where the index δ now varies from 1 to 2, and

$$S = b_{11} \quad , \quad T = b_{12} \quad , \quad U = b_{22} \quad . \tag{15}$$

Here $\underset{\sim}{n}$ is orthogonal to both $\underset{\sim}{r}_{\xi}$ and $\underset{\sim}{r}_{\eta}$, and the coefficients of the first fundamental form of the surface are g_{11}, g_{12}, and g_{22}; each evaluated at ζ = const. Obviously

$$g_{11} = x_\xi^2 + y_\xi^2 + z_\xi^2 \quad , \quad g_{12} = x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta \quad , \quad g_{22} = x_\eta^2 + y_\eta^2 + z_\eta^2 \quad . \tag{16}$$

If Eqs. (14) are considered as the first order partial differential equations in $\underset{\sim}{r}_{\xi}$ and $\underset{\sim}{r}_{\eta}$, then we must also consider the Weingarten equations

$$\underset{\sim}{n}_{\xi} = -b_{1\beta}g^{\beta\gamma}\underset{\sim}{r}_{\gamma} \quad , \tag{17a}$$

$$\underset{\sim}{n}_{\eta} = -b_{2\beta} g^{\beta\gamma} \underset{\sim}{r}_{\gamma} \quad . \tag{17b}$$

If now g_{11}, g_{12}, g_{22}, b_{11}, b_{12}, b_{22} are arbitrarily prescribed then the set of Eqs. (14) and (17), which represent fifteen scalar equations for the nine scalars ($\underset{\sim}{r}_{\xi}$, $\underset{\sim}{r}_{\eta}$, $\underset{\sim}{n}$), form an overdetermined system. Consequently one has to impose the compatability requirements

$$(\underset{\sim}{r}_{\alpha\beta})_{\gamma} = (\underset{\sim}{r}_{\alpha\gamma})_{\beta}$$

for all values of α, β, γ from 1 to 2. This operation leads to the Mainardi-Codazzi equations and the theorema egregium of Gauss which are higher order equations and are not very suitable for the purpose of numerical solution. We therefore return to the Gauss equations (14) and ask the question: Is it possible to develop a method which centers around the Gauss equations and is simple to implement numerically? The answer is in affirmative if we manipulate Eqs. (14) as follows.

Multiplying Eq. (14a) by g_{22}, Eq. (14b) by $-2g_{12}$ and Eq. (14c) by g_{11} and adding the three equations, we get

$$\pounds \underset{\sim}{r} = -[(\Delta_2 \xi)\underset{\sim}{r}_{\xi} + (\Delta_2 \eta)\underset{\sim}{r}_{\eta}]\, G_3 + (g_{22}S - 2g_{12}T + g_{11}U)\underset{\sim}{n} \quad , \tag{18}$$

where $\pounds$ is the second order differential operator,

$$\pounds = g_{22}\partial_{\xi\xi} - 2g_{12}\partial_{\xi\eta} + g_{11}\partial_{\eta\eta} \quad ,$$

and Δ_2 is the second order differential operator of Beltrami. For any surface x^{ν} = const., (refer to the scheme in (11)),

$$\Delta_2^{(\nu)} = \frac{1}{\sqrt{G_\nu}}\left[\partial_\alpha\left\{\frac{1}{\sqrt{G_\nu}}(g_{\beta\beta}\partial_\alpha - g_{\alpha\beta}\partial_\beta)\right\} + \partial_\beta\left\{\frac{1}{\sqrt{G_\nu}}(g_{\alpha\alpha}\partial_\beta - g_{\alpha\beta}\partial_\alpha)\right\}\right] \quad . \tag{19a}$$

In particular for the surface ζ = const. we drop the enclosed superscript and write

$$\Delta_2 = \frac{1}{\sqrt{G_3}} [\partial_\xi \{\frac{1}{\sqrt{G_3}} (g_{22}\partial_\xi - g_{12}\partial_\eta)\}$$

$$+ \partial_\eta \{\frac{1}{\sqrt{G_3}}(g_{11}\partial_\eta - g_{12}\partial_\xi)\}] \ . \tag{19b}$$

It is easy to show by using the definitions of $\Gamma^\delta_{\alpha\beta}$, that

$$\Delta_2\xi = \frac{1}{G_3}(2g_{12}\Gamma^1_{12} - g_{22}\Gamma^1_{11} - g_{11}\Gamma^1_{22}) \ , \tag{20a}$$

$$\Delta_2\eta = \frac{1}{G_3}(2g_{12}\Gamma^2_{12} - g_{22}\Gamma^2_{11} - g_{11}\Gamma^2_{22}) \ . \tag{20b}$$

The system of Eqs. (18) is still untamed and needs suitable constraints. We must also somehow modify the terms S, T, U so as to bring the variation of $\underset{\sim}{r}$ with respect to ζ, as was noted in the opening paragraph of this section. To achieve this objective we consider the Eqs. (8c) which for the surface ζ = const. are

$$\underset{\sim}{r}_{\xi\xi} = \Gamma^1_{11}\underset{\sim}{r}_\xi + \Gamma^2_{11}\underset{\sim}{r}_\eta + \Gamma^3_{11}\underset{\sim}{r}_\zeta \ , \tag{21a}$$

$$\underset{\sim}{r}_{\xi\eta} = \Gamma^1_{12}\underset{\sim}{r}_\xi + \Gamma^2_{12}\underset{\sim}{r}_\eta + \Gamma^3_{12}\underset{\sim}{r}_\zeta \ , \tag{21b}$$

$$\underset{\sim}{r}_{\eta\eta} = \Gamma^1_{22}\underset{\sim}{r}_\xi + \Gamma^2_{22}\underset{\sim}{r}_\eta + \Gamma^3_{22}\underset{\sim}{r}_\zeta \ , \tag{21c}$$

where all the derivatives with respect to ζ are assumed to have been evaluated at ζ = const. Taking the dot product of Eqs. (21) with $\underset{\sim}{n}$ and comparing with Eqs. (13), we find that

$$\left.\begin{aligned} b_{11} = S = \lambda\Gamma^3_{11} \ , \\ b_{12} = T = \lambda\Gamma^3_{12} \ , \\ b_{22} = U = \lambda\Gamma^3_{22} \ , \end{aligned}\right\} \tag{22}$$

where

$$\lambda = \underset{\sim}{n}\cdot\underset{\sim}{r}_\zeta = Xx_\zeta + Yy_\zeta + Zz_\zeta \quad , \tag{23}$$

$$\left.\begin{aligned} X &= (y_\xi z_\eta - y_\eta z_\xi)/\sqrt{G_3} \quad , \\ Y &= (x_\eta z_\xi - x_\xi z_\eta)/\sqrt{G_3} \quad , \\ Z &= (x_\xi y_\eta - x_\eta y_\xi)/\sqrt{G_3} \quad . \end{aligned}\right\} \tag{24}$$

Thus, by using the forms in (22) we have established a connection with the coordinate ζ which changes from one surface to the next. We now rewrite Eq. (18) as

$$\underset{\sim}{\mathcal{L}}\underset{\sim}{r} + [(\Delta_2\xi)\underset{\sim}{r}_\xi + (\Delta_2\eta)\underset{\sim}{r}_\eta]G_3 = \underset{\sim}{n}R \quad , \tag{25}$$

where

$$R = \lambda[g_{11}\Gamma^3_{22} - 2g_{12}\Gamma^3_{12} + g_{22}\Gamma^3_{11}) \quad . \tag{26a}$$

Note that

$$R = G_3(k_1 + k_2) \tag{26b}$$

where $k_1 + k_2$ is twice the mean curvature of the surface.

§3.1 Fundamental generating system of equations

We now impose the following differential constraints on the coordinates ξ and η:

$$\Delta_2\xi = 0 \quad , \tag{27a}$$

$$\Delta_2\eta = 0 \quad , \tag{27b}$$

and take them as the fundamental generating equations for the coordinates in a surface. It must be noted that Δ_2 is not a 2D Laplace operator except when the surface degenerates into a plane having no dependence on z.

It is a well known result in differential geometry that the isothermic coordinates in a surface satisfy Eqs. (27) identically. The isothermic coordinates ξ and η are those orthogonal coordinates in a surface which yield $g_{22} = g_{11}$. The situation here is parallel to the choice of the Laplace equations $\nabla^2\xi = 0$, $\nabla^2\eta = 0$ for the generation of plane curvilinear coordinates.

(e.g., the TTM method[2]), which are also satisfied identically by the conformal coordinates in a plane. This does not mean that the Laplace equations are suitable only for the generation of conformal coordinates. In fact, as is evidenced by the available body of numerical results, the Laplace equations are capable of generating very general coordinates in arbitrary domains. Therefore, there looks to be no apparent reason why Eqs. (27) should not form the basic generating system for general coordinates in a surface. The analytical solutions given in this paper and the numerical results given in Warsi and Ziebarth[15] support this contention.

Having chosen Eqs. (27) as the generating system, the equation for the determination of the Cartesian coordinates, viz., Eq. (25), becomes

$$\mathcal{L}\underset{\sim}{r} = n\underset{\sim}{R} \ . \tag{28}$$

The three scalar equations in expanded form are

$$g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = XR \ , \tag{29a}$$

$$g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = YR \ , \tag{29b}$$

$$g_{22}z_{\xi\xi} - 2g_{12}z_{\xi\eta} + g_{11}z_{\eta\eta} = ZR \ , \tag{29c}$$

where X, Y, Z, and R have been defined in Eqs. (24) and (26). It must be noted that by cyclic permutations, equations similar to Eqs. (29) can be written for the surfaces η = const. and ξ = const. However, only one set, e.g., Eqs. (29), is sufficient provided that we are able to take care of the derivatives $\underset{\sim}{r}_{\zeta}$ appearing in R.

The set of Eqs. (29) form a consistent set of equations for the determination of x, y, z under the prescribed boundary conditions.* For an analytical understanding of these equations we open the differentiations of the metric coefficients in the formulae for Γ^3_{11}, Γ^3_{12}, and Γ^3_{22}. Thus

$$\Gamma^3_{11} = \alpha x_{\xi\xi} + \beta y_{\xi\xi} + \gamma z_{\xi\xi} \ , \tag{30a}$$

$$\Gamma^3_{12} = \alpha x_{\xi\eta} + \beta y_{\xi\eta} + \gamma z_{\xi\eta} \ , \tag{30b}$$

*Refer to comment (i) at the end of the paper.

$$\Gamma_{22}^{3} = \alpha x_{\eta\eta} + \beta y_{\eta\eta} + \gamma z_{\eta\eta} \ , \qquad (30c)$$

where

$$\alpha = (G_5 x_\xi + G_6 x_\eta + G_3 x_\zeta)/g \ ,$$

$$\beta = (G_5 y_\xi + G_6 y_\eta + G_3 y_\zeta)/g \ ,$$

$$\gamma = (G_5 z_\xi + G_6 z_\eta + G_3 z_\zeta)/g \ .$$

Substituting Eqs. (30) in (26) and after arranging the terms we can rewrite Eqs. (29) as a quasilinear system,

$$A_{\alpha\beta}^{ij} \frac{\partial^2 x_j}{\partial x^\alpha \partial x^\beta} = 0 \ , \ i = 1, 2, 3 \qquad (31)$$

where $x_1 = x$, $x_2 = y$, $x_3 = z$ and there is an implicit sum on j from 1 to 3 and on α, β from 1 to 2. The coefficients $A_{\alpha\beta}^{ij}$ depend on the metric coefficients g_{11}, g_{12}, g_{22} and on those geometric quantities which depend only on the first partial derivatives. For example

$$A_{11}^{11} = g_{22}(1-\alpha\lambda X) \ , \ A_{12}^{11} = -2g_{12}(1-\alpha\lambda X) \ , \text{ etc., etc.}$$

Equations (31) are three equations in three unknowns with two independent variables. Refer to Petrovsky[16] for the classifications of such equations.

§3.2 Coordinate redistribution (concentration)

Before discussing the basic solution algorithm for the set of Eqs. (29) it is important to study the effect of a coordinate transformation which produces a nonuniform distribution of coordinates. Again using indexed quantities, let $\bar{x}^\alpha$ be another coordinate system defined as

$$\bar{x}^\alpha = \bar{x}^\alpha(x^1, x^2) \ , \ \alpha = 1, 2 \ ,$$

with

$$\det\left(\frac{\partial \bar{x}^\alpha}{\partial x^\beta}\right) \neq 0 \ .$$

Using x_i to mean either x, y, or z, we have

$$\frac{\partial x_i}{\partial x^\beta} = \frac{\partial x_i}{\partial \bar{x}^\gamma} \frac{\partial \bar{x}^\gamma}{\partial x^\beta} , \quad i = 1, 2, 3 , \tag{32a}$$

$$\frac{\partial^2 x_i}{\partial x^\alpha \partial x^\beta} = \frac{\partial^2 x_i}{\partial \bar{x}^\delta \partial \bar{x}^\gamma} \frac{\partial \bar{x}^\delta}{\partial x^\alpha} \frac{\partial \bar{x}^\gamma}{\partial x^\beta} + \frac{\partial x_i}{\partial \bar{x}^\gamma} \frac{\partial^2 \bar{x}^\gamma}{\partial x^\alpha \partial x^\beta} . \tag{32b}$$

Also,

$$g^{\alpha\beta} = \bar{g}^{\theta\sigma} \frac{\partial x^\alpha}{\partial \bar{x}^\theta} \frac{\partial x^\beta}{\partial \bar{x}^\sigma} . \tag{32c}$$

Now, Eqs. (29) can be written in a compact form as

$$g^{\alpha\beta} \frac{\partial^2 x_i}{\partial x^\alpha \partial x^\beta} = \frac{R}{G_3} X_i , \tag{33}$$

where

$$X_1 = X , \quad X_2 = Y , \quad X_3 = Z ,$$

and

$$g^{11} = g_{22}/G_3 , \quad g^{12} = -g_{12}/G_3 , \quad g^{22} = g_{11}/G_3 . \tag{34}$$

On coordinate transformation we have

$$G_3 = \bar{G}_3/(D^*)^2 , \quad R = \bar{R}/(D^*)^2 , \quad X_i = \bar{X}_i , \tag{35}$$

where

$$D^* = \frac{\partial x^1}{\partial \bar{x}^1} \frac{\partial x^2}{\partial \bar{x}^2} - \frac{\partial x^1}{\partial \bar{x}^2} \frac{\partial x^2}{\partial \bar{x}^1} .$$

Thus Eq. (33) becomes

$$\bar{g}^{\alpha\beta} \frac{\partial^2 x_i}{\partial \bar{x}^\alpha \partial \bar{x}^\beta} + \bar{g}^{\mu\sigma} P^\gamma_{\mu\sigma} \frac{\partial x_i}{\partial \bar{x}^\gamma} = \frac{\bar{R}}{\bar{G}_3} \bar{X}_i , \tag{36}$$

where

$$P^\gamma_{\mu\sigma} = \frac{\partial x^\alpha}{\partial \bar{x}^\mu} \frac{\partial x^\beta}{\partial \bar{x}^\sigma} \frac{\partial^2 \bar{x}^\gamma}{\partial x^\alpha \partial x^\beta} . \tag{37}$$

Using equations similar to (34) in the new coordinate system, Eq. (36) yields the equations

$$\bar{\mathcal{L}}x = \bar{\bar{X}}\bar{\bar{R}} \ , \tag{38a}$$

$$\bar{\mathcal{L}}y = \bar{\bar{Y}}\bar{\bar{R}} \ , \tag{38b}$$

$$\bar{\mathcal{L}}z = \bar{\bar{Z}}\bar{\bar{R}} \ , \tag{38c}$$

where

$$\bar{\mathcal{L}} = \bar{g}_{22}\partial_{\xi\xi} - 2\bar{g}_{12}\partial_{\xi\eta} + \bar{g}_{11}\partial_{\eta\eta} + \bar{P}\partial_{\xi} + \bar{Q}\partial_{\eta} \ , \tag{39}$$

$$\bar{P} = \bar{g}_{22}P^1_{11} - 2\bar{g}_{12}P^1_{12} + \bar{g}_{11}P^1_{22} \ , \tag{40a}$$

$$\bar{Q} = \bar{g}_{22}P^2_{11} - 2\bar{g}_{12}P^2_{12} + \bar{g}_{11}P^2_{22} \ , \tag{40b}$$

and $\bar{X}$, $\bar{Y}$, $\bar{Z}$, and $\bar{R}$ have exactly the same expressions as in (24) and (26a) in the new coordinate system.

The structure of the terms $P^{\gamma}_{\mu\sigma}$ is quite revealing particularly in those situations when it is desired to redistribute an already existing coordinate system x^{α} so as to achieve a desired concentration or expansion of the coordinates $\bar{x}^{\alpha}$. Though still a forcing function behavior for $P^{\gamma}_{\mu\sigma}$ has to be prescribed, the user is at least aware of its structure, that is, it must be composed of the product of two first partial derivatives and a second partial derivative. These considerations may be important in the adaptive coordinate systems. In other cases $P^{\gamma}_{\mu\sigma}$ may be prescribed arbitrarily. One such case has been treated numerically in Ref. 15. (Refer also to §3.)

§3.3 Morphology of A Solution Algorithm

The discussion that follows pertains to the case when it is desired to generate the 3D curvilinear coordinates between two artibrary shaped smooth surfaces. As is shown in Fig. 1, let the surface coordinates of the inner body $\eta = \eta_B$ and of the outer body $\eta = \eta_\infty$ be the <u>same</u> coordinates. Because of the right-handedness of the coordinate triple (ξ,η,ζ), the ordered pair (ζ,ξ) is taken as a positive ordered pair on both the surfaces. Since both the surfaces $\eta = \eta_B$ and $\eta = \eta_\infty$ are known either analytically or numerically, so that

$$\eta = \eta_B : \underset{\sim}{r} = \underset{\sim}{r}_B(\xi,\zeta) \; ; \; \eta = \eta_\infty : \underset{\sim}{r} = \underset{\sim}{r}_\infty(\xi,\zeta) \; , \tag{41}$$

and hence the needed partial derivatives with respect to ξ and ζ are directly available at the surfaces.

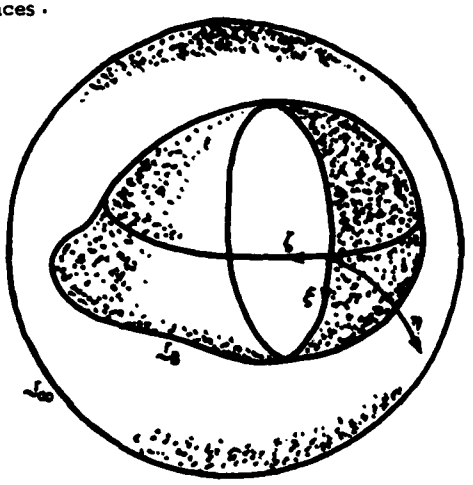


Figure 1. Selection of coordinates on the inner and outer boundaries.

For the computation of $\underset{\sim}{r}_\zeta$ in the field one must first note that the coordinate ζ may not, in general, satisfy the Beltrami's equation $\Delta_2^{(2)}\zeta = 0$. Consequently, $\underset{\sim}{r}_\zeta$ must satisfy the equation

$$\mathcal{L}^{(2)}\underset{\sim}{r} + G_2(\Delta_2^{(2)}\zeta)\underset{\sim}{r}_\zeta = G_2(k_1^{(2)} + k_2^{(2)})\underset{\sim}{n}^{(2)} \; .$$

From this equation we devise a weighted integral formula

$$\underset{\sim}{r}_\zeta = \int [f_1(\eta)(\underset{\sim}{r}_{\zeta\zeta})_B + f_2(\eta)(\underset{\sim}{r}_{\zeta\zeta})_\infty]d\zeta \; . \tag{42a}$$

where

$$(\underset{\sim}{r}_{\zeta\zeta})_{B,\infty} = [\frac{G_2}{g_{11}}(k_1^{(2)} + k_2^{(2)})\underset{\sim}{n}^{(2)} + \frac{2g_{13}}{g_{11}}\underset{\sim}{r}_{\xi\zeta} - \frac{g_{33}}{g_{11}}\underset{\sim}{r}_{\xi\xi}$$

$$- \frac{\sqrt{G_2}}{g_{11}}\{\frac{\partial}{\partial\zeta}(\frac{g_{11}}{\sqrt{G_2}}) - \frac{\partial}{\partial\xi}(\frac{g_{13}}{\sqrt{G_2}})\}\underset{\sim}{r}_\zeta]_{B,\infty} \; , \tag{42b}$$

and

$$f_1(\eta_B) = 1 \; , \; f_1(\eta_\infty) = 0 \; , \; f_2(\eta_B) = 0 \; , \; f_2(\eta_\infty) = 1 \; . \tag{42c}$$

Referring to Fig. 2(a), we now solve Eqs. (29) or (38) for each ζ = const., by prescribing the values of x, y and z on the lower curve C_1 and the upper curve C_2 which represent the curves on B and ∞ respectively. In Fig. 2(b) C_3 and C_4 are the cut lines on which periodic boundary conditions are to be imposed.

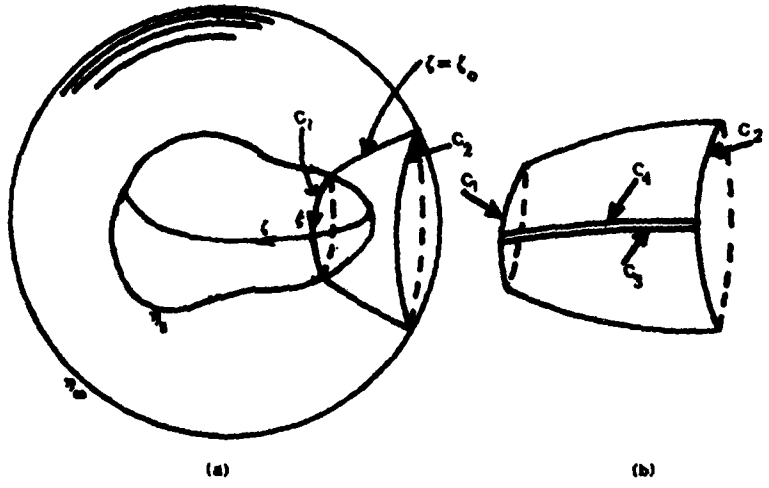


Figure 2.(a) Topology of the given surfaces. (b) Surface to be generated.

§3.4 Exact solutions

The following two examples demonstrate that the proposed set of generating equations (27) or equivalently the set of equations (29) or (38) are consistent and provide nontrivial solutions.

Example 1: Isothermic coordinates on a unit sphere.

Let the surface coordinates of a unit sphere be denoted as ξ, ζ, where the order (ζ,ξ) forms a right-handed system. Since our objective is to provide isothermic coordinates which are orthogonal, we assume

$$x = \psi(\zeta) \ , \ y = f(\zeta)\cos\ \xi \ , \ z = f(\zeta)\sin\ \xi \ , \tag{43a}$$

so that

$$f^2 + \psi^2 = 1 \ . \tag{43b}$$

Calculating the metric coefficients and the surface Christoffel symbols based on the assumed form (43a), we find that the equations $\Delta_2^{(2)}\xi = 0$ and $\Delta_2^{(2)}\zeta = 0$ are satisfied provided that

$$f^2 = \psi'^2 + f'^2 \ . \tag{43c}$$

Eliminating ψ between (43b,c), we get

$$f'^2 = (1-f^2)f^2 ,$$

which on integration yields

$$f(\zeta) = \frac{2e^{\zeta}}{1+e^{2\zeta}} , \quad \psi(\zeta) = \frac{1-e^{2\zeta}}{1+e^{2\zeta}} . \tag{43d}$$

It can be verified that when the solution (43d) is used in (43a) then the resulting metric coefficients g_{11} and g_{33} are equal. Thus the coordinates ξ,ζ are isothermic. The relations between the standard spherical polar coordinates θ,ϕ and the coordinates ξ,ζ are

$$\xi = \phi , \quad \zeta = \ln \tan \frac{\theta}{2} .$$

Refer also to §5.1.1.

Example 2: 3D coordinates between a prolate ellipsoid and a sphere.

We now consider the case of coordinate generation between an inner body $\eta = \eta_B$ which is a prolate ellipsoid and an outer body $\eta = \eta_\infty$ which is a sphere. The coordinates which vary on these two surfaces are ξ and ζ. A curve C_1 on the inner surface designated as $\zeta = \zeta_0$ is

$$x = \tau\cosh \eta_B \cos \zeta_0 , \; y = \tau\sinh \eta_B \sin \zeta_0 \cos \xi, \; z = \tau\sinh \eta_B \sin \zeta_0 \sin \xi. \tag{44a}$$

Similarly the curve C_2 corresponding to $\zeta = \zeta_0$ on the outer surface is

$$x = e^{\eta_\infty} \cos \zeta_0 , \; y = e^{\eta_\infty} \sin \zeta_0 \cos \xi , \; z = e^{\eta_\infty} \sin \zeta_0 \sin \xi . \tag{44b}$$

In order to provide the solution of the present problem with coordinate contraction, we consider Eqs. (38) and assume

$$\xi = \xi(\bar{\xi}) , \quad \eta = \eta(\bar{\eta}) + \eta_B \tag{45}$$

where $\bar{\xi} = 0$ corresponds to $\xi = 0$ and $\bar{\eta} = \bar{\eta}_B$ corresponds to $\eta = \eta_B$. Thus $\xi(0) = 0$, $\eta(\bar{\eta}_B) = 0$. Under the transformation (45), the only nonzero components of $P^{\gamma}_{\mu\sigma}$ are P^1_{11} and P^2_{22}. Writing

$$\lambda(\bar{\xi}) = \frac{d\xi}{d\bar{\xi}} , \quad \theta(\bar{\eta}) = \frac{d\eta}{d\bar{\eta}} .$$

we have

$$P^1_{11} = -\frac{1}{\lambda}\frac{d\lambda}{d\bar{\xi}} , \quad P^2_{22} = -\frac{1}{\theta}\frac{d\theta}{d\bar{\eta}} \tag{46}$$

Based on the forms of the boundary conditions (44a) and (44b) we assume the following forms for x, y, z for $\zeta = \zeta_0$:

$$x = f(\bar{\eta})\cos\zeta_0 \ , \ y = \phi(\bar{\eta})\sin\zeta_0 \cos\xi, \ z = \phi(\bar{\eta})\sin\zeta_0 \sin\xi \ . \tag{47}$$

The boundary conditions for f and ϕ are*

$$f(\bar{\eta}_B) = \tau\cosh\eta_B \ , \ f(\bar{\eta}_\infty) = e^{\eta_\infty} \ , \ \phi(\bar{\eta}_B) = \tau\sinh\eta_B \ , \ \phi(\bar{\eta}_\infty) = e^{\eta_\infty}. \tag{48}$$

Using the expressions in (47) we calculate the various partial derivatives, metric coefficients, and all other data as needed for the Eqs. (38). On substitution we get an equation containing $\sin^2\zeta_0$ and $\cos^2\zeta_0$. Equating to zero the coefficients of $\sin^2\zeta_0$ and $\cos^2\zeta_0$ we obtain

$$\frac{f''}{f'} = \frac{\theta'}{\theta} + \frac{\phi'}{\phi}, \tag{49}$$

$$\frac{\phi''}{\phi'} = \frac{\theta'}{\theta} + \frac{\phi'}{\phi} \ , \tag{50}$$

where a prime denotes differentiation with respect to $\bar{\eta}$. On direct integrations of Eqs. (49) and (50) under the boundary conditions (48), we get

$$f(\bar{\eta}) = Ae^{B\eta(\bar{\eta})} + C \ ,$$

$$\phi(\bar{\eta}) = De^{B\eta(\bar{\eta})} \ ,$$

where

$$A = \tau[(e^{\eta_\infty} - \tau\cosh\eta_B)\sinh\eta_B]/(e^{\eta_\infty} - \tau\sinh\eta_B) \ ,$$

$$B = (\eta_\infty - \ell n\tau\sinh\eta_B)/(\eta_\infty - \eta_B) \ ,$$

$$C = \tau[e^{\eta_\infty}(\cosh\eta_B - \sinh\eta_B)]/(e^{\eta_\infty} - \tau\sinh\eta_B) \ ,$$

$$D = \tau\sinh\eta_B \ .$$

As an application, we take

$$\xi(\bar{\xi}) = a\bar{\xi} \ , \ \eta(\bar{\eta}) = b(\bar{\eta} - \bar{\eta}_B)\kappa^{\bar{\eta}} \ ,$$

*τ and η_B are the parameters of the ellipsoid.

where a, b and κ are constants. Thus

$$\eta(\bar{\eta}) = \frac{(\eta_\infty - \eta_B)(\bar{\eta}-\bar{\eta}_B)}{\bar{\eta}_\infty - \bar{\eta}_B} \kappa^{(\bar{\eta}-\bar{\eta}_\infty)} .$$

By taking a value of κ slightly greater than one ($\kappa = 1.05$) we can have sufficient contraction in the $\bar{\eta}$-coordinate near the inner surface. For the chosen problem since the dependence on ζ is simple, we find that the generated coordinates between a prolate ellipsoid and a sphere are

$$x = [Ae^{B\eta(\bar{\eta})} + C]\cos\zeta,\ y = De^{B\eta(\bar{\eta})}\sin\zeta\cos\xi\ ,\ z = De^{B\eta(\bar{\eta})}\sin\zeta\sin\xi$$

This example shows that the chosen generating system of equations (38) are capable of providing non-isothermic coordinates between a prolate ellipsoid and a sphere.

§4. GENERATING DIFFERENTIAL EQUATIONS BASED ON LAPLACE EQUATIONS

For the purpose of coordinate generation in either two or three dimensions it has become quite popular, particularly after the publication of the TTM method[2], to adopt a system of inhomogeneous Laplace' equations as the generating system. The inhomogeneous terms are completely arbitrary and seemingly there is no guidance from the analytical side as to how they should be chosen. Because of this and due to other basic reasons it is important to reconsider the formulation of the problem of coordinate generation based on Laplace' system of equations from an analytical point of view. The conclusions drawn from these considerations are that the set of Laplace equations

$$\nabla^2 x^i = 0\ ,\ i = 1, 2, 3 \qquad (51)$$

are essentially the basis of the TTM method rather than the set of inhomogeneous equations

$$\nabla^2 \bar{x}^i = P^i(\bar{x}^1, \bar{x}^2, \bar{x}^3)\ ,\ i = 1, 2, 3\ , \qquad (52)$$

where P^i are the specified functions. The reason for this conclusion is that a coordinate transformation from x^i to any other system $\bar{x}^i$, both satisfying the same boundary conditions, automatically gives rise to the set of equations (52) from (51). Thus as soon as the solution of the system of equations (51) under the constraints of a body conforming boundary conditions has been obtained a transformation

$$\bar{x}^i = \bar{x}^i(x^1, x^2, x^3)$$

can redistribute these coordinates in any desired manner.

To formulate the above noted ideas analytically, we consider the formula for the Laplacian of a scalar ϕ in the curvilinear coordinate system,[12,13] which is

$$\nabla^2\phi = g^{ij}\left(\frac{\partial^2\phi}{\partial x^i \partial x^j} - \Gamma^r_{ij}\frac{\partial\phi}{\partial x^r}\right) . \tag{53}$$

If $\phi = x^m$ is any curvilinear coordinate, then from (53) we obtain

$$\nabla^2 x^m = -g^{ij}\Gamma^m_{ij} . \tag{54}$$

If $\phi = x_m$, where x_m is any of the rectangular Cartesian coordinate, $x_1 = x$, $x_2 = y$, $x_3 = z$, then since $\nabla^2 x_m = 0$, we obtain using (53),

$$g^{ij}\frac{\partial^2 x_m}{\partial x^i \partial x^j} + (\nabla^2 x^r)\frac{\partial x_m}{\partial x^r} = 0 . \tag{55}$$

Taking (51) as the basic generating system, we get from (55),

$$g^{ij}\frac{\partial^2 x_m}{\partial x^i \partial x^j} = 0 . \tag{56}$$

Using the formulae stated in §2, we get $Dx_m = 0$, or

$$Dx = 0 , \tag{57}$$

$$Dy = 0 , \tag{58}$$

$$Dz = 0 , \tag{59}$$

where the operator D is given by

$$D = G_1\partial_{\xi\xi} + G_2\partial_{\eta\eta} + G_3\partial_{\zeta\zeta} + 2G_4\partial_{\xi\eta} + 2G_5\partial_{\xi\zeta} + 2G_6\partial_{\eta\zeta} . \tag{60}$$

In two dimensions* $g_{33} = 1$, $\frac{\partial}{\partial \zeta} = 0$, so that D becomes

$$D = g_{22}\partial_{\xi\xi} - 2g_{12}\partial_{\xi\eta} + g_{11}\partial_{\eta\eta} . \tag{61}$$

Let $\bar{x}^i$ be another coordinate system which satisfies the same body conforming boundary conditions as the system x^i, and let

$$\bar{x}^i = \bar{x}^i(x^1, x^2, x^3) \ , \ i = 1, 2, 3 \ ,$$

with

$$\det\left(\frac{\partial \bar{x}^i}{\partial x^j}\right) \neq 0 \ .$$

Then an analysis similar to §3.2 shows that

$$\frac{\partial x_m}{\partial x^j} = \frac{\partial x_m}{\partial \bar{x}^\ell}\frac{\partial \bar{x}^\ell}{\partial x^j} \ ,$$

$$\frac{\partial^2 x_m}{\partial x^i \partial x^j} = \frac{\partial^2 x_m}{\partial \bar{x}^k \partial \bar{x}^\ell}\frac{\partial \bar{x}^k}{\partial x^i}\frac{\partial \bar{x}^\ell}{\partial x^j} + \frac{\partial x_m}{\partial \bar{x}^\ell}\frac{\partial^2 \bar{x}^\ell}{\partial x^i \partial x^j} \ .$$

Using the last expression and the transformation law

$$g^{ij} = \bar{g}^{rn}\frac{\partial x^i}{\partial \bar{x}^r}\frac{\partial x^j}{\partial \bar{x}^n}$$

in Eq. (56) we get

$$\bar{g}^{k\ell}\frac{\partial^2 x_m}{\partial \bar{x}^k \partial \bar{x}^\ell} + \bar{g}^{rn}\, p^\ell_{rn}\frac{\partial x_m}{\partial \bar{x}^\ell} = 0 \ , \tag{62}$$

where

$$p^\ell_{rn} = \frac{\partial x^i}{\partial \bar{x}^r}\frac{\partial x^j}{\partial \bar{x}^n}\frac{\partial^2 \bar{x}^\ell}{\partial x^i \partial x^j} \ , \tag{63}$$

and is symmetric in the lower two indices. If now in Eq. (55) we replace x^i by $\bar{x}^i$, g^{ij} by $\bar{g}^{ij}$ and introduce

---

*Refer to comment (ii) at the end of the paper.

$$\nabla^2 \bar{x}^r = \bar{g}^{mn}\, \bar{\Gamma}^r_{mn}$$

$$= P^r \ ,$$

then it amounts to the same thing as taking the non-homogeneous Laplace equations (52) as the generating system.† Thus we reach the conclusion that essentially Eqs. (51) are the basic generating equations and that any redistribution of the solution of Eqs. (51) gives rise to Eqs. (62).

Transferring the second term of Eq. (62) to the right hand side and using the formulae developed in §2 which are applicable to all coordinate systems, we obtain

$$\bar{D}x_m = -(G_1 P^\ell_{11} + G_2 P^\ell_{22} + G_3 P^\ell_{33} + 2G_4 P^\ell_{12} + 2G_5 P^\ell_{13} + 2G_6 P^\ell_{23}) \frac{\partial x_m}{\partial \bar{x}^\ell} \ , \qquad (64)$$

where $x_m = x$, y, or z, and $\bar{D}$ is the same operator as (60) in the new coordinate system. In two dimensions, Eq. (64) gives rise to the familiar forms[7,17]

$$\bar{D}x = -(\bar{g}_{22} P^1_{11} - 2\bar{g}_{12} P^1_{12} + \bar{g}_{11} P^1_{22}) x_{\bar{\xi}} - (\bar{g}_{22} P^2_{11} - 2\bar{g}_{12} P^2_{12} + \bar{g}_{11} P^2_{22}) x_{\bar{\eta}} \ , \qquad (65a)$$

$$\bar{D}y = -(\bar{g}_{22} P^1_{11} - 2\bar{g}_{12} P^1_{12} + \bar{g}_{11} P^1_{22}) y_{\bar{\xi}} - (\bar{g}_{22} P^2_{11} - 2\bar{g}_{12} P^2_{12} + \bar{g}_{11} P^2_{22}) y_{\bar{\eta}} \ . \qquad (65b)$$

It must be noted that the preceding analysis guides one to a proper selection of the quantities P^ℓ_{rn} for concentrating the coordinate lines in the desired regions. This selection, though still arbitrary, at least suggests that the chosen P^ℓ_{rn} should be something like a product of two first and one second partial derivatives. This idea is important in the adaptive coordinate systems. Furthermore, the preceding analysis also exposes for the first time the existence of the cross derivative quantities P^ℓ_{rn} $(r \neq n)$ which do not appear if one starts from the Eqs. (52) and which may be important in non-orthogonal coordinates. For example, in two dimensions the quantities P^ℓ_{rm} are

$$P^1_{11} = (\xi_{\bar{\xi}})^2 \bar{\xi}_{\xi\xi} + 2\xi_{\bar{\xi}} \eta_{\bar{\xi}} \bar{\xi}_{\xi\eta} + (\eta_{\bar{\xi}})^2 \bar{\xi}_{\eta\eta} \ ,$$

$$P^2_{11} = (\xi_{\bar{\xi}})^2 \bar{\eta}_{\xi\xi} + 2\xi_{\bar{\xi}} \eta_{\bar{\xi}} \bar{\eta}_{\xi\eta} + (\eta_{\bar{\xi}})^2 \bar{\eta}_{\eta\eta} \ ,$$

† Refer to comment (iii) at the end of the paper.

$$P_{22}^1 = (\xi_\eta)^2 \bar{\xi}_{\xi\xi} + 2\xi_\eta \eta_\eta \bar{\xi}_{\xi\eta} + (\eta_\eta)^2 \bar{\xi}_{\eta\eta} ,$$

$$P_{22}^2 = (\xi_\eta)^2 \bar{\eta}_{\xi\xi} + 2\xi_\eta \eta_\eta \bar{\eta}_{\xi\eta} + (\eta_\eta)^2 \bar{\eta}_{\eta\eta} ,$$

$$P_{12}^1 = \xi_{\bar{\xi}} \xi_\eta \bar{\xi}_{\xi\xi} + (\xi_{\bar{\xi}} \eta_\eta + \eta_{\bar{\xi}} \xi_\eta) \bar{\xi}_{\xi\eta} + \eta_{\bar{\xi}} \eta_\eta \bar{\xi}_{\eta\eta} ,$$

$$P_{12}^2 = \xi_{\bar{\xi}} \xi_\eta \bar{\eta}_{\xi\xi} + (\xi_{\bar{\xi}} \eta_\eta + \eta_{\bar{\xi}} \xi_\eta) \bar{\eta}_{\xi\eta} + \eta_{\bar{\xi}} \eta_\eta \bar{\eta}_{\eta\eta} .$$

If $\xi = \xi(\bar{\xi})$ and $\eta = \eta(\bar{\eta})$, then writing

$$\lambda = \frac{d\xi}{d\bar{\xi}} , \quad \theta = \frac{d\eta}{d\bar{\eta}}$$

we get

$$P_{11}^1 = -\frac{1}{\lambda} \frac{d\lambda}{d\bar{\xi}} , \; P_{11}^2 = 0 , \; P_{22}^1 = 0 ,$$

$$P_{22}^2 = = \frac{1}{\theta} \frac{d\theta}{d\bar{\eta}} , \; P_{12}^1 = 0 , \; P_{12}^2 = 0 ,$$

which are exactly the same as have been used in an earlier paper.[17] In this case, writing for brevity

$$P_{11}^1 = P , \; P_{22}^2 = Q ,$$

Eqs. (65) simply become

$$\bar{D}x = -(\bar{g}_{22} P x_{\bar{\xi}} + \bar{g}_{11} Q x_{\bar{\eta}}) , \tag{66a}$$

$$\bar{D}y = -(\bar{g}_{22} P y_{\bar{\xi}} + \bar{g}_{11} Q y_{\bar{\eta}}) . \tag{66b}$$

These equations do not contain the cross derivative terms P_{12}^1, P_{12}^2 because $\bar{\xi}$ and $\bar{\eta}$ have been chosen to be functions of ξ and η respectively.

§4.1 Case of orthogonal coordinates

In general, for the generation of orthogonal coordinates it is not necessary that the coordinate functions should also satisfy the Laplace equations in the xyz-space. In this section after summarizing the basic generating equations for the orthogonal coordinates we have studied the effect of constraining the coordinate functions to be simultaneously harmonic.

The orthogonality conditions are

$$g_{ij} = 0 \text{ for } i \neq j \ . \tag{67}$$

Also, for orthogonal coordinates Eqs. (54) simply become

$$\left.\begin{aligned} \nabla^2\xi &= \frac{1}{\sqrt{g}} \frac{\partial}{\partial \xi} \left(\frac{h_2 h_3}{h_1}\right) \ , \\ \nabla^2\eta &= \frac{1}{\sqrt{g}} \frac{\partial}{\partial \eta} \left(\frac{h_1 h_3}{h_2}\right) \ , \\ \nabla^2\zeta &= \frac{1}{\sqrt{g}} \frac{\partial}{\partial \zeta} \left(\frac{h_1 h_2}{h_3}\right) \ , \end{aligned}\right\} \tag{68}$$

where

$$\nabla^2 = \partial_{xx} + \partial_{yy} + \partial_{zz} \ , \ h_1 = \sqrt{g_{11}} \ , \ h_2 = \sqrt{g_{22}} \ , \ h_3 = \sqrt{g_{33}} \ , \ \sqrt{g} = h_1 h_2 h_3 \ .$$

Proceeding directly from Eq. (55) and using Eqs. (67) and (68) we obtain

$$\Xi x_m = 0 \ , \ m = 1, 2, 3 \ , \tag{69}$$

where

$$\Xi = \frac{\partial}{\partial \xi} \left(\frac{h_2 h_3}{h_1} \frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta} \left(\frac{h_1 h_3}{h_2} \frac{\partial}{\partial \eta}\right) + \frac{\partial}{\partial \zeta} \left(\frac{h_1 h_2}{h_3} \frac{\partial}{\partial \zeta}\right) \ .$$

Note that the operator Ξ and the Laplacian operator ∇^2 are related as

$$\Xi \phi = h_1 h_2 h_3 \nabla^2 \phi \ ,$$

for a scalar ϕ.

Equations (69) are those fundamental equations which every orthogonal coordinate system must satisfy. A program of calculation using Eqs. (67) and (69) along with the definitions of g_{11}, g_{22} and g_{33} can be developed.

§4.1.1 Case of orthogonal coordinates using the Laplace equations

Case I: 3D coordinates.

If the generating system of equations is taken as

$$\nabla^2\xi = 0 \ , \ \nabla^2\eta = 0 \ , \ \nabla^2\zeta = 0 \ , \tag{70}$$

then from Eqs. (68) we find that

$$h_1 = f_2(\xi,\zeta)f_3(\xi,\eta) \ , \ h_2 = f_1(\eta,\zeta)f_3(\xi,\eta) \ , \ h_3 = f_1(\eta,\zeta)f_2(\xi,\zeta) \ , \tag{71}$$

where f_1, f_2, f_3 are arbitrary functions of their arguments. Also the generating system (69) for the Cartesian coordinates becomes

$$g_{22}g_{33}\frac{\partial^2 x_m}{\partial\xi^2} + g_{11}g_{33}\frac{\partial^2 x_m}{\partial\eta^2} + g_{11}g_{22}\frac{\partial^2 x_m}{\partial\zeta^2} = 0 \ , \ m = 1, 2, 3 \ , \tag{72}$$

which because of (71) can also be written as

$$f_1^2(\eta,\zeta)\frac{\partial^2 x_m}{\partial\xi^2} + f_2^2(\xi,\zeta)\frac{\partial^2 x_m}{\partial\eta^2} + f_3^2(\xi,\eta)\frac{\partial^2 x_m}{\partial\zeta^2} = 0 \ , \ m = 1, 2, 3 \ . \tag{73}$$

Case II: 2D coordinates.

For the case of 2D orthogonal coordinates the equations

$$\nabla^2\xi = 0 \ , \ \nabla^2\eta = 0 \ , \tag{74}$$

with the use of Eqs. (68) yield

$$g_{22} = ag_{11} \ ,$$

where a is a constant. The case a = 1 gives the corresponding isothermic coordinates which are conformal. However, by a straight forward coordinate transformation of the isothermic coordinates ξ,η to another coordinates $\bar{\xi},\bar{\eta}$ we can have a coordinate distribution in which $\bar{g}_{22} \neq \bar{g}_{11}$. For, let

$$\xi = \xi(\bar{\xi},\bar{\eta}) \ , \ \eta = \eta(\bar{\xi},\bar{\eta})$$

be an arbitrary orthogonal transformation. Using the chain rule of differentiation, we get

$$\xi_{xx} = \xi_{\bar{\xi}}\bar{\xi}_{xx} + \xi_{\bar{\eta}}\bar{\eta}_{xx} + \xi_{\bar{\xi}\bar{\xi}}(\bar{\xi}_x)^2 + \xi_{\bar{\eta}\bar{\eta}}(\bar{\eta}_x)^2 + 2\xi_{\bar{\xi}\bar{\eta}}\bar{\xi}_x\bar{\eta}_x$$

etc., etc.,

which when used in Eqs. (74) along with the orthogonality condition

$$\bar{\xi}_x\bar{\eta}_x + \bar{\xi}_y\bar{\eta}_y = 0$$

and the formulae

$$\nabla^2\bar{\xi} = \frac{1}{\sqrt{g}}\frac{\partial}{\partial\bar{\xi}}\sqrt{\bar{g}_{22}/\bar{g}_{11}}\ , \tag{75a}$$

$$\nabla^2\bar{\eta} = \frac{1}{\sqrt{g}}\frac{\partial}{\partial\bar{\eta}}\sqrt{\bar{g}_{11}/\bar{g}_{22}}\ , \tag{75b}$$

$$(\bar{\xi}_x)^2 + (\bar{\xi}_y)^2 = \frac{1}{\bar{g}_{11}}\ ,\ (\bar{\eta}_x)^2 + (\bar{\eta}_y)^2 = \frac{1}{\bar{g}_{22}}\ ,\ \bar{g} = \bar{g}_{11}\bar{g}_{22}\ ,$$

yield the equations

$$\sqrt{\bar{g}}\ \nabla^2\xi = \frac{\partial}{\partial\bar{\xi}}\left(\xi_{\bar{\xi}}\sqrt{\bar{g}_{22}/\bar{g}_{11}}\right) + \frac{\partial}{\partial\bar{\eta}}\left(\xi_{\bar{\eta}}\sqrt{\bar{g}_{11}/\bar{g}_{22}}\right) = 0\ , \tag{76a}$$

$$\sqrt{\bar{g}}\ \nabla^2\eta = \frac{\partial}{\partial\bar{\xi}}\left(\eta_{\bar{\xi}}\sqrt{\bar{g}_{22}/\bar{g}_{11}}\right) + \frac{\partial}{\partial\bar{\eta}}\left(\eta_{\bar{\eta}}\sqrt{\bar{g}_{11}/\bar{g}_{22}}\right) = 0\ . \tag{76b}$$

A study of Eqs. (76) suggests that if ξ is only a function of $\bar{\xi}$, and η is only a function of $\bar{\eta}$, e.g.,

$$\xi(\bar{\xi}) = \int \mu(\bar{\xi})d\bar{\xi}\ ,\ \eta(\bar{\eta}) = \int \frac{d\bar{\eta}}{\nu(\bar{\eta})}\ ,$$

then Eqs. (76a,b) are identically satisfied by taking

$$\sqrt{\bar{g}_{11}/\bar{g}_{22}} = \mu(\bar{\xi})\nu(\bar{\eta})\ . \tag{76c}$$

Thus

$$\bar{g}_{11} = \mu^2(\bar{\xi})\nu^2(\bar{\eta})\bar{g}_{22}, \tag{76d}$$

and so the coordinates $\bar{\xi},\bar{\eta}$ are orthogonal but not conformal.

An important result from the preceding analysis is that if the orthogonal coordinates are generated through the solution of the Laplace equations (74) then there exists an infinity of transformations $\xi = \xi(\bar{\xi})$, $\eta = \eta(\bar{\eta})$ in which the ratio $\bar{g}_{11}/\bar{g}_{22}$ is a product of a function of $\bar{\xi}$ and a function of $\bar{\eta}$. This result is not in general true for coordinates not satisfying the Laplace equations.

§5. GENERATING DIFFERENTIAL EQUATIONS BASED ON THE RIEMANN TENSOR

In any given space there are endless possibilities for the introduction of coordinate curves. Each chosen set of curves determines its own metric components. For example, in a Cartesian plane besides introducing rectangular Cartesian coordinates x, y, we also have endless possibilities for introducing either orthogonal or nonorthogonal coordinate curves. However, as is well known, there is a basic differential constraint on the variations of g_{ij}'s irrespective of the coordinate system. Since the curvature of an Euclidean two-dimensional plane is identically zero, the basic differential constraint on the g_{ij}'s is

$$(G_3)^{-1}R_{1212} = \frac{\partial}{\partial\eta}\left(\frac{\sqrt{G_3}}{g_{11}}\Gamma^2_{11}\right) - \frac{\partial}{\partial\xi}\left(\frac{\sqrt{G_3}}{g_{11}}\Gamma^2_{12}\right) = 0\ , \tag{77}$$

where ξ,η are any arbitrary coordinate curves in the plane. Thus no matter which coordinate system is introduced in a plane, the corresponding matrics g_{ij} must satisfy Eq. (77). Equation (77) has also been used as the basic generating equation for the generation of orthogonal coordinates in a plane[18]. In general, the Riemann curvature tensor R_{rjnp} defined as, [12,13]

$$R_{rjnp} = \frac{1}{2}\left(\frac{\partial^2 g_{rp}}{\partial x^j \partial x^n} + \frac{\partial^2 g_{jn}}{\partial x^r \partial x^p} - \frac{\partial^2 g_{rn}}{\partial x^j \partial x^p} - \frac{\partial^2 g_{jp}}{\partial x^r \partial x^n}\right)$$

$$+ g^{ts}([jn,s]\,[rp,t] - [jp,s][rn,t]) \tag{78}$$

defines the components of the curvature tensor of any general space. If the space is N-dimensional, then the number of components R_{rjnp} are given by

$$\frac{N^2}{12}(N^2-1)\ .$$

Thus for N = 2 there is one distinct surviving component stated in Eq. (77). However, for N = 3, it has six distinct components

$$R_{1212},\ R_{1313},\ R_{2323},\ R_{1213},\ R_{1232},\ R_{1323}\ .$$

If the 3D-space is Euclidean, then its curvature is zero, so that the six equations

$$R_{1212} = 0\ ,\ R_{1313} = 0\ ,\ R_{2323} = 0\ ,$$
$$R_{1213} = 0\ ,\ R_{1232} = 0\ ,\ R_{1323} = 0 \tag{79}$$

determine the differential constraints for the six metric coefficients g_{ij} in any coordinate system introduced in an Euclidean space. These equations in the expanded form are as follows:

$$R_{1212} = \frac{\partial^2 g_{11}}{\partial \eta^2} - 2\frac{\partial^2 g_{12}}{\partial \xi \partial \eta} + \frac{\partial^2 g_{22}}{\partial \xi^2} + 2g^{ts}([22,s][11,t] - [12,s][12,t]) = 0\ , \tag{80a}$$

$$R_{1313} = \frac{\partial^2 g_{11}}{\partial \zeta^2} - 2\frac{\partial^2 g_{13}}{\partial \xi \partial \zeta} + \frac{\partial^2 g_{33}}{\partial \xi^2} + 2g^{ts}([33,s][11,t] - [13,s][13,t]) = 0\ , \tag{80b}$$

$$R_{2323} = \frac{\partial^2 g_{22}}{\partial \zeta^2} - 2\frac{\partial^2 g_{23}}{\partial \eta \partial \zeta} + \frac{\partial^2 g_{33}}{\partial \eta^2} + 2g^{ts}([33,s][22,t] - [23,s][23,t]) = 0\ , \tag{80c}$$

$$R_{1213} = \frac{\partial^2 g_{11}}{\partial \eta \partial \zeta} - \frac{\partial^2 g_{12}}{\partial \xi \partial \zeta} - \frac{\partial^2 g_{13}}{\partial \xi \partial \eta} + \frac{\partial^2 g_{23}}{\partial \xi^2} + 2g^{ts}([23,s][11,t] - [12,s][13,t]) = 0, \tag{80d}$$

$$R_{1232} = \frac{\partial^2 g_{22}}{\partial \xi \partial \zeta} - \frac{\partial^2 g_{12}}{\partial \eta \partial \zeta} - \frac{\partial^2 g_{23}}{\partial \xi \partial \eta} + \frac{\partial^2 g_{13}}{\partial \eta^2} + 2g^{ts}([22,s][13,t] - [23,s][12,t]) = 0, \tag{80e}$$

$$R_{1323} = \frac{\partial^2 g_{33}}{\partial \xi \partial \eta} - \frac{\partial^2 g_{13}}{\partial \eta \partial \zeta} - \frac{\partial^2 g_{23}}{\partial \xi \partial \zeta} + \frac{\partial^2 g_{12}}{\partial \zeta^2} + 2g^{ts}([33,s][12,t] - [23,s][13,t]) = 0, \tag{80f}$$

where [ij,k] are the Christoffel symbols of the first kind defined in (8a).

Equations (80) are those consistent set of partial differential equations which must always be satisfied by the metric coefficients g_{ij}. In the 3D case Eqs. (80) are six equations in six unknowns and, therefore, they form a closed system of equations. In contrast, for the 2D case there is only one equation (Eq. (77)) and three unknowns g_{11}, g_{12}, g_{22} and therefore some constraints

are needed to turn Eq. (77) (such as orthogonality[18]) into a solvable equation. This author is not aware of any numerical solution of the complete set of equations (80), though there are some possibilities of developing solution algorithms using Eqs. (80) as the core equations. For example, in the problem of obtaining the 3D coordinates for the configuration of Fig. 1, one can judiciously choose g_{11}, g_{13}, and g_{33} based on the given boundary data for the whole field and then solve Eqs. (80) for the remaining coefficients g_{22}, g_{23}, and g_{12}. It should also be noted that in any physical problem, e.g., the Navier-Stokes problem, one only needs the metric coefficients and their derivatives (Christoffel symbols), which become available after solving Eqs. (80). Nevertheless, for graphical and other purposes, one also needs the functions $x(\xi,\eta,\zeta)$ etc.

To obtain the Cartesian coordinates on the basis of the available g_{ij}'s, we introduce the unit base vectors $\underset{\sim}{\lambda}_i$ as

$$\underset{\sim}{\lambda}_i = \underset{\sim}{a}_i/\sqrt{g_{ii}} \ , \text{ no sum on i.} \tag{81}$$

Let the components of $\underset{\sim}{\lambda}_i$ along the rectangular Cartesian axes be denoted as u_i, v_i, w_i, so that

$$\underset{\sim}{\lambda}_i = (u_i, v_i, w_i) \ ,$$

where

$$\left.\begin{aligned} u_1 &= x_\xi/\sqrt{g_{11}} \ , \ v_1 = y_\xi/\sqrt{g_{11}} \ , \ w_1 = z_\xi/\sqrt{g_{11}} \ , \\ u_2 &= x_\eta/\sqrt{g_{22}} \ , \ v_2 = y_\eta/\sqrt{g_{22}} \ , \ w_2 = z_\eta/\sqrt{g_{22}} \ , \\ u_3 &= x_\zeta/\sqrt{g_{33}} \ , \ v_3 = y_\zeta/\sqrt{g_{22}} \ , \ w_3 = z_\zeta/\sqrt{g_{33}} \ . \end{aligned}\right\} \tag{82}$$

Knowing u_i, v_i, w_i, it is possible to evaluate the Cartesian coordinates through the line integrals

$$\underset{\sim}{r} = \int (\underset{\sim}{\lambda}_1\sqrt{g_{11}}\, d\xi + \underset{\sim}{\lambda}_2\sqrt{g_{22}}\, d\eta + \underset{\sim}{\lambda}_3\sqrt{g_{33}}\, d\zeta) \ . \tag{83}$$

The determination of u_i, v_i, w_i is a separate problem which we now consider. First of all using (81) in Eq. (8c), we get a system of first order partial differential equations

$$\frac{\partial \underset{\sim}{\lambda}_i}{\partial x^j} = \underset{\sim}{\lambda}_1 \left(\frac{g_{11}}{g_{ii}}\right)^{\frac{1}{2}} \Gamma^1_{ij} + \underset{\sim}{\lambda}_2 \left(\frac{g_{22}}{g_{ii}}\right)^{\frac{1}{2}} \Gamma^2_{ij}$$

$$+ \underset{\sim}{\lambda}_3 \left(\frac{g_{33}}{g_{ii}}\right)^{\frac{1}{2}} \Gamma^3_{ij} - \frac{\underset{\sim}{\lambda}_i}{2g_{ii}} \frac{\partial g_{ii}}{\partial x^j} , \tag{84}$$

where, as before, there is no sum on the repeated index i. Equations (84) form a system of 27 first order PDE's in nine independent variables u_i, v_i, w_i. This system of equations is overdetermined and thus its solvability should depend on certain compatibility conditions. According to a theorem on the overdetermined system of equations[19], if the compatibility conditions hold then the solution of Eqs. (84) exists and is unique. The conditions

$$\frac{\partial^2 \underset{\sim}{\lambda}_i}{\partial x^m \partial x^j} = \frac{\partial^2 \underset{\sim}{\lambda}_i}{\partial x^j \partial x^m} \tag{85}$$

for all values of i, m, and j are the compatibility conditions. To prove (85) we use Eq. (8c), which on cross differentiation yields

$$\frac{\partial^2 \underset{\sim}{a}_i}{\partial x^m \partial x^j} - \frac{\partial^2 \underset{\sim}{a}_i}{\partial x^j \partial x^m} = R^{\ell}_{.imj} \underset{\sim}{a}_{\ell} , \tag{86}$$

where $R^{\ell}_{.imj}$ is the Riemann-Christoffel curvature[12] tensor and is related with the Riemann's tensor $R_{ijk\ell}$. Evidently in our present case $R^{\ell}_{.imj} = 0$, since the space is Euclidean. Inserting (81) in (86) we find that Eq. (85) are identically satisfied.

It is interesting to note that for a two-dimensional curvilinear coordinate system there is no need to solve the system of equations such as (84). In this case the single differential equation with $G_3 = g$

$$R_{1212} = \sqrt{g}\left[\frac{\partial}{\partial \eta}\left(\frac{\sqrt{g}\ \Gamma^2_{11}}{g_{11}}\right) - \frac{\partial}{\partial \xi}\left(\frac{\sqrt{g}\ \Gamma^2_{12}}{g_{11}}\right)\right] = 0$$

implies the existence of a single function $\alpha(\xi,\eta)$ such that

$$\alpha_\xi = \frac{-\sqrt{g}}{g_{11}}\ \Gamma^2_{11} , \quad \alpha_\eta = \frac{-\sqrt{g}}{g_{11}}\ \Gamma^2_{12} .$$

Consequently

$$u_1 = \cos\alpha,\ v_1 = -\sin\alpha,\ u_2 = \cos(\alpha-\theta)\ ,\ v_2 = -\sin(\alpha-\theta)\ ,$$

where α is the angle made by the tanget to the coordinate line η = const. in a clockwise sense with the x-axis, and

$$\cos\theta = g_{12}/\sqrt{g_{11}g_{22}}$$

is known.

§5.1 Case of orthogonal coordinates

We again return to the case of 3D orthogonal coordinates. Refer also to §4.1. Under the constraint of orthogonality,

$$\left.\begin{aligned} &g_{12} = g_{13} = g_{23} = 0\ ,\ [12,3] = [13,2] = [23,1] = 0\ , \\ &\Gamma^3_{12} = \Gamma^2_{13} = \Gamma^1_{23} = 0\ ,\ g = g_{11}g_{22}g_{33}\ , \end{aligned}\right\} \tag{87}$$

the set of equations (80) reduce somewhat. They are

$$\frac{\partial}{\partial\xi}\left(\frac{1}{\sqrt{g_{11}g_{22}}}\frac{\partial g_{22}}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(\frac{1}{\sqrt{g_{11}g_{22}}}\frac{\partial g_{11}}{\partial\eta}\right) + \frac{1}{2g_{33}\sqrt{g_{11}g_{22}}}\frac{\partial g_{11}}{\partial\zeta}\frac{\partial g_{22}}{\partial\zeta} = 0\ , \tag{88a}$$

$$\frac{\partial}{\partial\xi}\left(\frac{1}{\sqrt{g_{11}g_{33}}}\frac{\partial g_{33}}{\partial\xi}\right) + \frac{\partial}{\partial\zeta}\left(\frac{1}{\sqrt{g_{11}g_{33}}}\frac{\partial g_{11}}{\partial\zeta}\right) + \frac{1}{2g_{22}\sqrt{g_{11}g_{33}}}\frac{\partial g_{11}}{\partial\eta}\frac{\partial g_{33}}{\partial\eta} = 0\ , \tag{88b}$$

$$\frac{\partial}{\partial\eta}\left(\frac{1}{\sqrt{g_{22}g_{33}}}\frac{\partial g_{33}}{\partial\eta}\right) + \frac{\partial}{\partial\zeta}\left(\frac{1}{\sqrt{g_{22}g_{33}}}\frac{\partial g_{22}}{\partial\zeta}\right) + \frac{1}{2g_{11}\sqrt{g_{22}g_{33}}}\frac{\partial g_{22}}{\partial\xi}\frac{\partial g_{33}}{\partial\xi} = 0\ , \tag{88c}$$

$$\frac{\partial^2 g_{11}}{\partial\eta\partial\zeta} = \frac{1}{2}\frac{\partial g_{11}}{\partial\eta}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\zeta} + \frac{1}{g_{22}}\frac{\partial g_{22}}{\partial\zeta}\right) + \frac{1}{2g_{33}}\frac{\partial g_{11}}{\partial\zeta}\frac{\partial g_{33}}{\partial\eta}\ , \tag{88d}$$

$$\frac{\partial^2 g_{22}}{\partial\xi\partial\zeta} = \frac{1}{2}\frac{\partial g_{22}}{\partial\zeta}\left(\frac{1}{g_{22}}\frac{\partial g_{22}}{\partial\xi} + \frac{1}{g_{33}}\frac{\partial g_{33}}{\partial\xi}\right) + \frac{1}{2g_{11}}\frac{\partial g_{11}}{\partial\zeta}\frac{\partial g_{22}}{\partial\xi}\ , \tag{88e}$$

$$\frac{\partial^2 g_{33}}{\partial\xi\partial\eta} = \frac{1}{2}\frac{\partial g_{33}}{\partial\xi}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\eta} + \frac{1}{g_{33}}\frac{\partial g_{33}}{\partial\eta}\right) + \frac{1}{2g_{22}}\frac{\partial g_{22}}{\partial\xi}\frac{\partial g_{33}}{\partial\eta}\ , \tag{88f}$$

which are the Lame's equations.

§5.1.1 The case of isothermic coordinates.

Isothermic coordinates* in a surface embedded in a 3D Euclidean space are those coordinates in which the metric coefficients g_{11} and g_{33} in the surface η = const. are equal. That is, the element of length ds on η = const. is given by

$$(ds)^2_{\eta=\text{const.}} = g_{11}\,[(d\xi)^2 + (d\zeta)^2]\,,$$

where ξ,ζ are chosen to be the surface coordinates. Setting

$$g_{33} = g_{11}, \text{ and } g_{22} = F(\eta)$$

in Eqs. (88), we obtain the basic equations for g_{11}, which are

$$\frac{\partial}{\partial\xi}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\xi}\right) + \frac{\partial}{\partial\zeta}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\zeta}\right) + \frac{1}{2Fg_{11}}\left(\frac{\partial g_{11}}{\partial\eta}\right)^2 = 0\,, \tag{89a}$$

$$\frac{\partial}{\partial\eta}\left(\frac{1}{\sqrt{Fg_{11}}}\frac{g_{11}}{\partial\eta}\right) = 0\,, \tag{89b}$$

$$\frac{\partial}{\partial\zeta}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\eta}\right) = 0\,, \tag{89c}$$

$$\frac{\partial}{\partial\xi}\left(\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial\eta}\right) = 0\,. \tag{89d}$$

It can easily be verified that the only solution of Eqs. (89c,d) is

$$g_{11} = [a+P(\eta)]^2 f(\xi,\zeta)\,,\ a = \text{const.} \tag{90}$$

Thus from (89b)

$$F(\eta) = \left(\frac{dP}{d\eta}\right)^2\,. \tag{91}$$

Substituting (90) and (91) in Eq. (89a), the differential equation for

*Refer to the comment (iv) at the end of the paper.

$f(\xi,\zeta)$ becomes

$$\frac{\partial}{\partial\xi}\left(\frac{1}{f}\frac{\partial f}{\partial\xi}\right) + \frac{\partial}{\partial\zeta}\left(\frac{1}{f}\frac{\partial f}{\partial\zeta}\right) + 2f = 0 \ . \tag{92}$$

In Kreyszig[14] we have the result that if in a portion of a surface isothermic coordinates can be introduced then that portion of the surface can conformally be mapped onto a plane. Thus in effect the solution of Eq. (92) provides that mapping function which conformally maps a surface onto a plane. As a verification of the above conclusion, we verify that the function

$$f = \frac{4e^{2\zeta}}{(1+e^{2\zeta})^2} \tag{93}$$

is a solution of Eq. (92). This function is related with the isothermic coordinates on a sphere. Using the parametric equationsof a sphere

$$x = [a+P(\eta)]\cos\theta,\ y = [a+P(\eta)]\sin\theta\sin\phi,\ z = [a+P(\eta)]\sin\theta\cos\phi$$

and writing

$$\xi = \phi\ ,\ \zeta = \ln\tan\frac{\theta}{2}\ ,$$

where $0 < \phi < 2\pi$ and $0 < \theta < \pi$, we obtain

$$g_{33} = g_{11} = \frac{4(a+P)^2e^{2\zeta}}{(1+e^{2\zeta})^2}\ .$$

Thus the equations

$$\left.\begin{aligned} x &= \frac{(a+P)(1-e^{2\zeta})}{1+e^{2\zeta}}\ , \\ y &= \frac{2(a+P)e^{\zeta}\sin\xi}{1+e^{2\zeta}}\ , \\ z &= \frac{2(a+P)e^{\zeta}\cos\xi}{1+e^{2\zeta}} \end{aligned}\right\} \tag{94}$$

represent a sphere of radius $a+P(\eta)$ in terms of the isothermic coordinates ξ,ζ in the surface. Since $P(\eta)$ is an arbitrary function of η, we have the capability of prescribing a suitable function $P(\eta)$ to achieve any sort of

contraction or expansion in the field. It looks that the representation (94) should prove useful in the computational problems associated with a sphere.

Comment (i):

As a further justification for the consistency of the set of Eqs. (29) it has been shown below that these equations can be combined to obtain the equation for a surface $z = z(x,y)$ in the well known form

$$\alpha z_{xx} - 2\beta z_{xy} + \gamma z_{yy} = 2HM \ , \tag{i}$$

where

$$2H = k_1+k_2 = R/G_3 \ , \ M = 1+p^2+q^2 \ , \ p = z_x, \ q = z_y \ ,$$

$$\alpha = (1+q^2)/\sqrt{M} \ , \ \beta = pq/\sqrt{M} \ , \ \gamma = (1+p^2)/\sqrt{M} \ .$$

First note the following definitions and identities:

$$G_3 = g_{11}g_{22} - (g_{12})^2 \ , \ X = -p/\sqrt{M} \ , \ Y = -q/\sqrt{M} \ , \ Z = 1/\sqrt{M} \ ,$$

$$\Delta_1(x,x) = (1-X^2)G_3 \ , \ \Delta_1(x,y) = -XYG_3 \ , \ \Delta_1(y,y) = (1-Y^2)G_3 \ ,$$

where (ii)

$$\Delta_1(a,b) = g_{22}a_\xi b_\xi - g_{12}(a_\xi b_\eta + a_\eta b_\xi) + g_{11}a_\eta b_\eta \ .$$

Calculating $z_{\xi\xi}$, $z_{\xi\eta}$, $z_{\eta\eta}$ from z_ξ, z_η, substituting these expressions in Eq. (29c) while using the equations in (ii) and Eqs. (29a,b) we recover Eq. (i) given above.

We now compare the equations obtained by Thomas[6] with those of Warsi[11]. Thomas' equations in the present notation are

$$\mathfrak{L}x + 2pG_3H/\sqrt{M} = 0 \ , \ \mathfrak{L}y + 2qG_3H/\sqrt{M} = 0 \ , \text{where } G_3 = (x_\xi y_\eta - x_\eta y_\xi)^2 M \ , \tag{iii}$$

which are exactly the same as Eqs. (29a,b) of this paper. It must, however, be pointed out that the derivation of Eqs. (iii) involves four steps[6]: (a) orthogonality of ζ with ξ,η, (b) vanishing of the curvature of the ζ-lines, (c) elimination of an arbitrary parameter (which may be zero), (d) prescription of $z(x,y)$ for the surface to be generated.

Comment (ii):

In two dimensions another differential system is provided by a first order Beltrami equation [20], which in the complex form is

$$f_{\bar{z}} - H(z,\bar{z})f_z = 0 \ , \tag{i}$$

where

$$f = f(z,\bar{z}) \ ,$$

$$z = x + iy \ , \ \bar{z} = x - iy \ , \ i = \sqrt{-1} \ .$$

Writing

$$f(z,\bar{z}) = \xi(x,y) + i\eta(x,y) \ ; \ H(z,\bar{z}) = \mu(x,y) + i\nu(x,y) \ , \tag{ii}$$

we obtain the following two real equations from (i):

$$-\eta_x = \beta\xi_x + \gamma\xi_y \ , \tag{iii}$$

$$\eta_y = \alpha\xi_x + \beta\xi_y \ , \tag{iv}$$

where

$$\alpha = [(1-\mu)^2 + \nu^2]/\Delta \ , \ \beta = -2\nu/\Delta \ , \ \gamma = [(1+\mu)^2 + \nu^2]/\Delta \ , \ \Delta = 1-(\mu^2 + \nu^2) \ .$$

Note that

$$\alpha\gamma - \beta^2 = 1 \ ,$$

$$\alpha + \gamma = 2(2-\Delta)/\Delta$$

A quasiconformal mapping becomes conformal when $H = 0$, or equivalently $\alpha = \gamma = 1$, $\beta = 0$. The resulting equations are then the Cauchy-Riemann equations

$$\xi_x = \eta_y \ , \ \xi_y = -\eta_x \ ,$$

and then $f(z)$ is an analytic function in the domain D.

Equations (iii) and (iv) can be inverted so that only the partial derivatives of x and y appear. Thus

$$x_\eta = \beta x_\xi - \alpha y_\xi \ , \qquad \text{(v)}$$

$$y_\eta = \gamma x_\xi - \beta y_\xi \ . \qquad \text{(vi)}$$

For Eqs. (v) and (vi) it is important to write α, β, γ in terms of the metric coefficients, which are[7,12],

$$\Delta = 4\sqrt{g}/[2\sqrt{g} + (g_{11} + g_{22})] \ ,$$

$$\alpha + \gamma = (g_{11} + g_{22})/\sqrt{g} \ ,$$

$$\alpha,\gamma = \left[g_{11} + g_{22} \mp \{(g_{11} + g_{22})^2 - 4(1 + \beta^2)g\}^{\frac{1}{2}}\right]/2\sqrt{g} \ .$$

Comment (iii):

As is expected, Eq. (82) can be reduced to the form

$$\bar{g}^{k\ell} \frac{\partial^2 x_m}{\partial \bar{x}^k \partial \bar{x}^\ell} + (\nabla^2 \bar{\xi}^r) \frac{\partial x_m}{\partial \bar{x}^r} = 0$$

by using the formula

$$\frac{\partial^2 \bar{x}^\ell}{\partial x^i \partial x^j} = \Gamma^p_{ij} \frac{\partial \bar{x}^\ell}{\partial x^p} - \bar{\Gamma}^\ell_{qr} \frac{\partial \bar{x}^q}{\partial x^i} \frac{\partial \bar{x}^r}{\partial x^j}$$

in the expression for $\dfrac{\partial^2 x_m}{\partial x^i \partial x^j}$. However, for gaining a new insight into the structure of the redistribution terms it looks profitable to keep the form (62) with P^ℓ_{rn} defined in (63).

Comment (iv):

Generation of isothermic coordinates can also be achieved by the method detailed in Ref. 14. Let $\bar{x}^1$ and $\bar{x}^2$ be some sort of coordinates introduced in a portion of the surface (for example from the subroutine developed by Craidon[21]), and let x^1, x^2 be the desired isothermic coordinates. Then

$$x^i = x^i(\bar{x}^1, \bar{x}^2) \ .$$

Because of x^i being isothermic, we have

$$g_{22} = g_{11} \ .$$

Using the transformation law for the covariant and the contravariant metric tensor components we get

$$\frac{\partial x^1}{\partial \tilde{x}^i} = \tilde{g}_{ij} \tilde{\varepsilon}^{jk} \frac{\partial x^2}{\partial \tilde{x}^k} , \tag{i}$$

$$\frac{\partial x^2}{\partial \tilde{x}^i} = -\tilde{g}_{ij} \tilde{\varepsilon}^{jk} \frac{\partial x^1}{\partial \tilde{x}^k} \tag{ii}$$

where

$$\tilde{\varepsilon}^{jk} = \frac{1}{\sqrt{\tilde{g}}} e_{jk} ,$$

$$e_{11} = 0 , \; e_{22} = 0 , \; e_{12} = +1 , \; e_{21} = -1 .$$

From Eqs. (i) and (ii) we find the second order differential equations

$$\frac{\partial}{\partial \tilde{x}^i} (\sqrt{\tilde{g}} \, \tilde{g}^{ij} \frac{\partial x^k}{\partial \tilde{x}^j}) = 0 , \tag{iii}$$

where k = 1, 2. Note that in the Eqs. (i) - (iii) the indices range over the values 1, 2.

Equations (iii) provide two linear uncoupled equations for the determination of the isothermic coordinates, since the values $\tilde{g}^{ij}$ of $\tilde{g}^{ij}$ are known a' priori.

CONCLUSIONS

Three distinct methods of numerical coordinate generation based on PDE's have been analyzed in detail. In the two newly proposed methods, viz., the methods discussed in §§3 and 5, some useful results have been obtained by looking at the generating system of equations as a system of forcing differential relations among the metric coefficients g_{ij}. For example, in the method of §3 and g_{ij}'s are forced to satisfy Eqs. (27) (refer also to their forms in Eqs. (20)). In the method of §5, the g_{ij}'s naturally satisfy Eqs. (80) since the space is intrinsically Euclidean. In the TTM method discussed in §4 the generating Laplace or Poisson equations also amount to specifying a set of differential constraints on the g_{ij}'s.

In the process of obtaining the above noted results a number of other results and equations have been obtained which should be satisfied by all

coordinate systems. For example, the orthogonal coordinates in an Euclidean space must satisfy Eqs. (69), (88), and the nonorthogonal coordinates must satisfy Eqs. (80), no matter which method is used to generate them. In effect all these results provide enough material for proposing more efficient calculation algorithms for the coordinate generation on a computer.

ACKNOWLEDGMENTS

The author gratefully acknowledges the continuing research support of the Air Force Office of Scientific Research through Grant No. AFOSR 80-0185.

REFERENCES

1. Winslow, A. M. (1967) J. Comp. Phy., 2 pp. 149-172.
2. Thompson, J. F., Thames, F. C., and Mastin, C. W. (1974) J. Comp. Phy., 15, pp. 299-319.
3. Steger, J. L. and Sorensen, R. L. (1979) J. Comp. Phy., 33, pp. 405-410.
4. Yu, N. J. (1980) AIAA Paper 80-1391.
5. Graves, R. A. (1980) NASA Tech. Memo. 80131.
6. Thomas, P. D. (1981) AIAA Paper 81-0996.
7. Thompson, J. F., Warsi, Z. U. A. and Mastin, C. W. J. Comp. Phy. (to appear).
8. Eiseman, P. R. (1979) J. Comp. Phy., 33, pp. 118-150.
9. Smith, R. E. and Weigel, B. L. (1980) AIAA Paper 80-0192.
10. Erickson, L. E. (1980) NASA Cp-2166, pp. 253-264.
11. Warsi, Z. U. A. (1980) Report MSSU-EIRS-80-7, Mississippi State University Engineering and Industrial Research Station.
12. Warsi, Z. U. A. (1981) Report MSSU-EIRS-81-1, Mississippi State University Engineering and Industrial Research Station.
13. Eisenhart, L. P. (1947) An Introduction to Differential Geometry with Use of the Tensor Calculus. Princeton University Press, Princeton, N.J.
14. Kreyszig, E. (1959) Differential Geometry. University of Toronto Press, Toronto.
15. Warsi, Z. U. A. and Ziebarth, J. P. These proceedings.
16. Petrovsky, I. G. (1954) Lectures on Partial Differential Equations. Interscience Publishers, Inc., New York.
17. Warsi, Z. U. A. and Thompson, J. F. (1976) Report MSSU-EIRS-ASE-77-1, Mississippi State University Engineering and Industrial Research Station.
18. Warsi, Z. U. A. and Thompson, J. F. (1982) Math. Comp., 38. To be published.
19. Stoker, J. J. (1969) Differential Geometry. Wiley-Interscience, New York, pp. 392-395.
20. Ahlfors, L. V. (1966) Lectures on Conformal Mappings. D. Van Nostrand Co., Inc., Princeton, N.J.
21. Craidon, C. B. (1975) NASA TMX-3206.

AD P000969

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ELLIPTIC GRID GENERATION

JOE F. THOMPSON
Department of Aerospace Engineering, Mississippi State University,
P. O. Drawer A, Mississippi State, Mississippi 39762

ABSTRACT

Various types of generating systems for boundary-conforming coordinate systems based on the numerical solution of systems of elliptic partial differential equations are discussed. Particular emphasis is given to the determination of functions in these equations which control the distribution of the curvilinear coordinate lines in the field.

INTRODUCTION

In general, the generation of a boundary-conforming coordinate system is accomplished by a determination of the values of the curvilinear coordinates in a region from specified values (and/or slopes of the coordinate lines) on the boundary of the region. One coordinate will be constant on each segment of the physical boundary curve (surface in 3D), while the other varies monotonically along the segment.

The equivalent problem in the transformed region is the determination of values of the physical (cartesian or other) coordinates in the interior of the transformed region from specified values on the boundary of the transformed region, as discussed in the first paper of this volume. This is a more reasonable problem for computation, since the boundary of the transformed region is comprised of horizontal and vertical segments, so that this region is composed of rectangular blocks which are contiguous, at least in the sense of being joined by re-entrant boundaries (branch cuts).

Now the generation of field values of a function from boundary values can be done in various ways, e.g., by interpolation between the boundaries, etc., and several methods are discussed elsewhere in this volume. The solution of such a boundary-value problem, however, is a classic problem of partial differential equations, so that it is logical to take the coordinates to be solutions of a system of partial differential equations. If the coordinate points (and/or slopes) are specified on the entire closed boundary of the region, the choice of equations must be elliptic, while if the specification is on only a portion of the boundary the choice would be hyperbolic. This latter case would occur, for instance, when an inner boundary of a region is

PREVIOUS PAGE IS BLANK

specified, but a surrounding outer boundary is arbitrary. The present discussion, however, treats the general case of a completely specified boundary, which requires an elliptic partial differential system.

ELLIPTIC GENERATION SYSTEMS

The extremum principles, i.e., that extrema of solutions cannot occur within the field, exhibited by some elliptic systems, can serve to guarantee a one-to-one mapping between the physical and transformed regions. Thus, since the variation of the curvilinear coordinate along a physical boundary segment must be monotonic, and is over the same range over facing boundary segments, it clearly follows that the extrema of the curvilinear coordinates must occur on the boundaries of the physical field and not in the interior. (Note that it is the extremum principles of the partial differential system in the physical plane, i.e., with the curvilinear coordinates as the dependent variables, that is relevant since it is the curvilinear coordinates, not the cartesian coordinates, that must be constant or monotonic on the boundaries.)

Another important property in regard to coordinate system generation is the inherent smoothness that prevails in the solutions of elliptic systems. Furthermore, boundary slope discontinuities are not propagated into the field. There are then a number of advantages to using a system of elliptic partial differential equations as a means of coordinate system generation. The historical progress of the choice of elliptic systems for this purpose has been traced in Thompson, et al.[1] Consequently, in the interest of space, references to all earlier work will not be made here. Numerous examples of the generation and application of coordinate systems generated from elliptic partial differential equations are covered in the above reference, as well as in other papers in the present volume.

Laplace system. The most simple elliptic partial differential system, and one that does exhibit an extremum principle and considerable smoothness, is the Laplace system

$$\nabla^2 \xi = 0 \tag{1a}$$

$$\nabla^2 \eta = 0 \tag{1b}$$

This generation system guarantees a one-to-one mapping, c.f., Mastin & Thompson[2], for boundary-conforming curvilinear coordinate systems on general closed boundaries.

With this generating system the coordinate lines will tend to be equally spaced in the absence of boundary curvature because of the strong smoothing effect of the Laplacians, but will become more closely spaced over convex boundaries and less so over concave boundaries as is illustrated below:

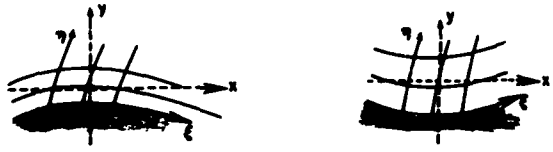


Fig. 1.

In the first case shown here, we have $\eta_{xx} > 0$ because of the convex (to the interior) curvature of the lines of constant η (η-lines). Therefore, by Fig. 1a, it follows that $\eta_{yy} < 0$, and hence the spacing between the η-lines must increase with y. The η-lines thus will tend to be more closely spaced over such a convex boundary segment. For concave segments, illustrated in Fig. 1b, we have $\eta_{xx} < 0$, so that η_{yy} must be positive, and hence the spacing of the η-lines must decrease outward from this boundary.

Poisson system

Control of the coordinate line distribution in the field can be exercised by generalizing the elliptic generating system to Poisson equations:

$$\nabla^2 \xi = P \tag{2a}$$

$$\nabla^2 \eta = Q \tag{2b}$$

This system still possesses an extremum principle if the inhomogeneous functions do not change sign in the field. It should be noted, however, that the presence of an extremum principle is a sufficient, but not necessary, condition for a one-to-one mapping, so that some latitude can be taken in the form of the inhomogeneous functions. This system is the basis of the TOMCAT code of Thompson, et al.[3]

Effect of control functions. Since a negative value of the control function would tend to make η_{yy} more negative, it follows that negative values of Q will tend to cause the coordinate line spacing in the cases shown above to increase more rapidly outward from the boundary. Generalizing, negative values of the control function Q will cause the η-lines to tend to move toward lines with lower values of η, while negative values of P will cause ξ-lines to tend to move toward lines having lower values of ξ. These effects are illustrated below for an η-line boundary.

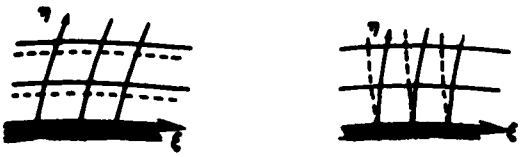


Fig. 2.

Note that since the boundary values are fixed, the ξ-lines cannot change the intersection with the boundary. The effect of the control function P at the boundary in this case is thus to change the angle of intersection, causing the ξ-lines to rotate toward lines with lower values of ξ. These effects are illustrated in the following figures:

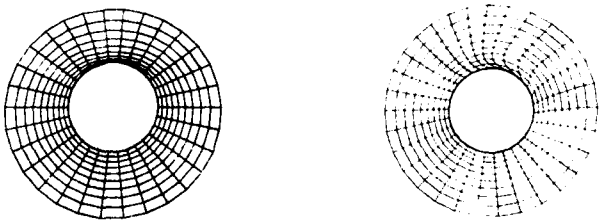


Fig. 3.

Here the ξ-lines are radial and the η-lines are circumferential. In the left illustration the control function Q is locally non-zero near a portion of the inner boundary, while in the right figure P is locally non-zero.

In general, a negative value of the Laplacian of one of the curvilinear coordinates causes the lines on which that coordinate is constant to move toward lines having lower values of that coordinate. For coordinate lines intersecting a boundary, this has the effect of causing such lines to rotate toward lines having lower values of this coordinate while maintaining fixed intersections with the boundary. Positive values of the Laplacian naturally result in the opposite effect, i.e., displacement toward lines having higher values of the coordinate.

Effect of boundary point distribution. Because of the strong smoothing tendencies that are inherent in the Laplacian operator, in the absence of the control functions, i.e., with P = Q = 0, the coordinate lines will tend to be generally equally spaced far from the boundaries regardless of the boundary point distribution. For example, the simple case of a coordinate system comprised of horizontal and vertical lines in a rectangular region cannot be obtained as a solution of Eq.(2) with P = Q = 0 unless the boundary points

are equally spaced. This is easily seen since with vertical ξ-lines we have $\xi_{yy} = 0$. If these lines are not equally spaced then ξ_{xx} cannot vanish, and hence the Laplacian cannot be zero.

Thus, if the coordinate lines in the interior of the region are to have the same general spacing as the point distributions on the boundaries which these lines connect, it is necessary to evaluate the control functions to be compatible with the boundary point distribution. Continuing the simple example of the rectangular field with horizontal and vertical coordinate lines, since $\xi_{yy} = \eta_{xx} = 0$, Eq.(2) reduce to

$$\xi_{xx} = P$$

$$\eta_{yy} = Q$$

where P and Q cannot vanish if the point distribution is not uniform on the horizontal and vertical boundaries, respectively. These effects are illustrated in the figures below. Here the control functions are zero in the left figure.

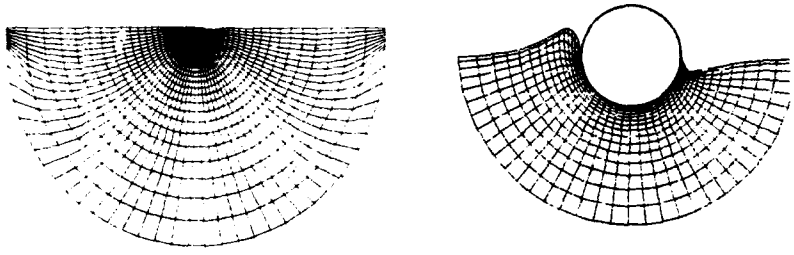


Fig. 4.

Note that although the spacing is not uniform on the semi-circular outer boundary in this figure, the angular spacing is essentially uniform away from this boundary. By contrast non-zero control functions in the right figure, evaluated from the boundary point distribution, cause the field spacing to follow that on the boundary. This evaluation of the control functions from the boundary point distribution is discussed more fully in a later section.

General Poisson system

Generalizing the Poisson system of Eq.(2) to three-dimensions, we have, with the cartesian and curvilinear coordinates denoted, respectively, as x_i and ξ^i,

$$\nabla^2 \xi^i = P^i(\xi^j, \frac{\partial \xi^j}{\partial x_k}, x_k) \qquad (3)$$

where i, j, and k all range over 1-3. In the transformed plane, this system becomes, using Eq.(23) of the first paper in this volume,

$$\sum_{i=1}^{3} \left(\sum_{j=1}^{3} g^{ij} \frac{\partial^2 x_k}{\partial \xi^i \partial \xi^j} + P^i \frac{\partial x_k}{\partial \xi^i} \right) = 0 \tag{4}$$

where

$$g^{ij} = \frac{1}{g}(g_{km}g_{\ell n} - g_{kn}g_{\ell m}) \tag{5}$$

with (i, k, ℓ) and (j, m, n) cyclic and

$$g_{ij} = \underset{\sim}{r}_{\xi^i} \cdot \underset{\sim}{r}_{\xi^j} \tag{6}$$

$$g = [\underset{\sim}{r}_{\xi^1} \cdot (\underset{\sim}{r}_{\xi^2} \times \underset{\sim}{r}_{\xi^3})]^2 \tag{7}$$

Here $\underset{\sim}{r}$ is the general position vector in cartesian coordinates.

Note that if P^i is not a function of the coordinate derivatives, $\frac{\partial \xi^j}{\partial x_k}$, the system is linear in the physical region, cf. Eq.(2), but quasi-linear in the transformed region, cf. Eq.(4). The equations are thus more complicated in general in the transformed region, but this is overshadowed by the great simplification in the boundary conditions that arises from the fact of the rectangular boundaries in the transformed region.

Alternate form. Eq.(4) can be made to reduce to a particularly simple form in one dimension by taking P^i to be of the form

$$P^i = g^{ii}P^i \tag{8}$$

(No summation is implied here. All summations are explicitly indicated throughout this paper.) Eq.(4) then becomes

$$\sum_{i=1}^{3} \sum_{j=1}^{3} g^{ij} \left(\frac{\partial^2 x_k}{\partial \xi^i \partial \xi^j} + \delta_{ij} P^i \frac{\partial x_k}{\partial \xi^i} \right) = 0 \tag{9}$$

This form becomes particularly simple in one dimension, since then we have

$$\frac{x_{\xi\xi}}{x_\xi} = -P \tag{10}$$

which can be integrated analytically if P is a function only of ξ to give

$$x(\xi) = x_0 + a\int_0^{\xi} \exp[-\int_0^{\xi'} P(\xi'')d\xi'']d\xi' \tag{11}$$

with $a = (x_\xi)_0$. In general, the function P^i may depend on the same variables as did p^i in its original definition. The magnitude of the function P^i should be some orders of magnitude smaller than that of the function p^i because of the multiplication of the former by g^{ii}, which is a measure of the ratio of the arc length of a cell side to the cell volume.

Vector form. In vector form, Eq.(9) can be written as

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij}(\underset{\sim}{r}_{\xi^i\xi^j} + \delta_{ij}P^i\underset{\sim}{r}_{\xi^i}) = 0 \tag{12}$$

In two dimensions we have, with $\underset{\sim}{k}$ the unit vector in the direction of invariance and ξ^3 the curvilinear coordinate in this direction,

$$\underset{\sim}{r}_{\xi^3} = \underset{\sim}{k} \quad \text{and} \quad \underset{\sim}{r}_{\xi^1} \cdot \underset{\sim}{r}_{\xi^3} = \underset{\sim}{r}_{\xi^2} \cdot \underset{\sim}{r}_{\xi^3} = 0$$

Then Eq.(12) reduces to the following, with $P \equiv P^1$ and $Q \equiv P^2$, for a two-dimensional plane surface:

$$|\underset{\sim}{r}_\eta|^2(\underset{\sim}{r}_{\xi\xi} + P\underset{\sim}{r}_\xi) - 2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta)\underset{\sim}{r}_{\xi\eta} + |\underset{\sim}{r}_\xi|^2(\underset{\sim}{r}_{\eta\eta} + Q\underset{\sim}{r}_\eta) = 0 \tag{13}$$

Other systems

Two other forms of elliptic system that have been considered in the literature are Eq.(2) with the control functions taken of the forms

$$P^i = gP^i \quad \text{Godunov \& Prokopov}^4 \tag{14}$$

and

$$P^i = -\frac{1}{D}\underset{\sim}{\nabla}D \cdot \underset{\sim}{\nabla}\xi^i \quad \text{Winslow}^5 \tag{15}$$

This latter choice puts Eq.(2) in the form of a diffusion equation with a variable diffusitivity:

$$\underset{\sim}{\nabla} \cdot (D\underset{\sim}{\nabla}\xi^i) = 0 \tag{16}$$

in which the "diffusitivity," D, becomes the control function.

Elsewhere in this volume Brackbill discusses elliptic generating systems developed from a variational approach based on minimizing the integral of

$$[(\nabla\xi)^2 + (\nabla\eta)^2] + \lambda_v(w\sqrt{g}) + \lambda_o(\nabla\xi \cdot \nabla\eta)^2$$

over the field. The Euler equations for this functional provide the elliptic partial differential system for ξ and η, which consists of two quasilinear equations in all six second derivatives, with coefficients that are quadratic functions of the first derivatives. The term involving w causes the mesh to adjust so that the product, wg, is more nearly constant over the mesh. The last term in the functional serves to minimize the departure from orthogonality, while the first term, which contributes the Laplacian to the Euler equations, regulates the smoothness. Larger values of λ_v and λ_o give added weight to the corresponding features in the solution. The non-negative weight function, w(x,y), may be taken to be a measure of some physical solution variation or magnitude, or may measure the truncation error in some manner. Obviously, the mesh will tend to be fine where this function is large.

Systems for curved surfaces

A two-dimensional coordinate system can be generated on a curved surface, but the curvature of the surface must be taken into account.

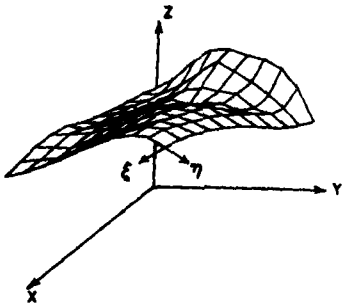


Fig. 5.

Two approaches are discussed below. The first requires specification of the surface while the second determines the surface along with the coordinate system thereon using information from specified bounding surfaces intersected by the surface in question.

Specified surfaces (Thomas[6]). Consider for definiteness a surface of constant ζ and assume, for the moment, that the coordinate lines emanating from this surface are normal to the surface, i.e., $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\zeta = \underset{\sim}{r}_\eta \cdot \underset{\sim}{r}_\zeta = 0$. Then Eq.(12) reduces to the following on the surface:

$$g^{11}(\underset{\sim}{r}_{\xi\xi} + P\underset{\sim}{r}_\xi) + g^{22}(\underset{\sim}{r}_{\eta\eta} + Q\underset{\sim}{r}_\eta) + g^{33}(\underset{\sim}{r}_{\zeta\zeta} + R\underset{\sim}{r}_\zeta) + 2g^{12}\underset{\sim}{r}_{\xi\eta} = 0 \quad (17)$$

Let the ζ-surface be specified by $z = f(x,y)$. Then on the surface

$$z_\xi = f_x x_\xi + f_y y_\xi \tag{18a}$$

$$z_\eta = f_x x_\eta + f_y y_\eta \tag{18b}$$

and

$$z_{\xi\xi} = f_x x_{\xi\xi} + f_y y_{\xi\xi} + f_{xx} x_\xi^2 + f_{yy} y_\xi^2 + 2f_{xy} x_\xi y_\xi \tag{18c}$$

$$z_{\eta\eta} = f_x x_{\eta\eta} + f_y y_{\eta\eta} + f_{xx} x_\eta^2 + f_{yy} y_\eta^2 + 2f_{xy} x_\eta y_\eta \tag{18d}$$

$$z_{\xi\eta} = f_x x_{\xi\eta} + f_y y_{\xi\eta} + f_{xx} x_\xi x_\eta + f_{yy} y_\xi y_\eta + f_{xy}(x_\xi y_\eta + x_\eta y_\xi) \tag{18e}$$

Now eliminating R between the x and z components of Eq.(17) we have

$$g^{11}(x_{\xi\xi} + Px_\xi) + g^{22}(x_{\eta\eta} + Qx_\eta) + g^{33}x_{\zeta\zeta} + 2g^{12}x_{\xi\eta}$$

$$= \frac{x_\zeta}{z_\zeta}\left[g^{11}(z_{\xi\xi} + Pz_\xi) + g^{22}(z_{\eta\eta} + Qz_\eta) + g^{33}z_{\zeta\zeta} + 2g^{12}z_{\xi\eta}\right] \tag{19}$$

Eliminating R between the y and z components yields a similar equation with x replaced by y.

In order to treat the ζ-derivatives, assume for the moment that the curvature of the coordinate lines crossing the surface vanishes thereon. Now the principal curvature of a line is given by

$$C\underset{\sim}{N} = \frac{d\underset{\sim}{T}}{ds}$$

where $\underset{\sim}{T}$ is the tangent to the line, s is arc length along the line, and $\underset{\sim}{N}$ is a unit vector normal to $\underset{\sim}{T}$, called the principal normal to the line. The tangent to a ζ-line is given by $\underset{\sim}{T} = \underset{\sim}{r}_\zeta/|\underset{\sim}{r}_\zeta|$, and the arc length is $ds = |\underset{\sim}{r}_\zeta|d\zeta$, so that

$$C\underset{\sim}{N} = \frac{1}{|\underset{\sim}{r}_\zeta|}\frac{d}{d\zeta}\left(\frac{\underset{\sim}{r}_\zeta}{|\underset{\sim}{r}_\zeta|}\right) = \frac{\underset{\sim}{r}_{\zeta\zeta}}{|\underset{\sim}{r}_\zeta|^2} - \frac{(\underset{\sim}{r}_\zeta \cdot \underset{\sim}{r}_{\zeta\zeta})\underset{\sim}{r}_\zeta}{|\underset{\sim}{r}_\zeta|^4} \tag{20}$$

The vanishing of the curvature of the ζ-lines crossing the surface is then expressed by

$$|\underset{\sim}{r}_\zeta|^2 \underset{\sim}{r}_{\zeta\zeta} - (\underset{\sim}{r}_\zeta \cdot \underset{\sim}{r}_{\zeta\zeta})\underset{\sim}{r}_\zeta = 0 \tag{21}$$

which yields

$$x_{\zeta\zeta} = \frac{x_\zeta}{z_\zeta} z_{\zeta\zeta}$$

$$y_{\zeta\zeta} = \frac{y_\zeta}{z_\zeta} z_{\zeta\zeta} \tag{22}$$

Also from the orthogonality conditions for the ζ-line crossing the surface, i.e., $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\zeta = \underset{\sim}{r}_\eta \cdot \underset{\sim}{r}_\zeta = 0$, we have

$$\frac{x_\zeta}{z_\zeta} = -\frac{1}{J}(z_\xi y_\eta - z_\eta y_\xi)$$

$$\frac{y_\zeta}{z_\zeta} = -\frac{1}{J}(x_\xi z_\eta - x_\eta z_\xi) \tag{23}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$. Then, using Eq.(18), it follows that

$$\frac{x_\zeta}{z_\zeta} = -f_x$$

$$\frac{y_\zeta}{z_\zeta} = -f_y \tag{24}$$

Now substitution of Eq.(22) causes the second ζ-derivaties in Eq.(19) to cancel. Then we have, using Eq.(18) for the derivatives of z in this equation and the analogous equation for y,

$$Dx + J^2 f_x G = 0$$

$$Dy + J^2 f_y G = 0 \tag{25}$$

where

$$Dx \equiv |\underset{\sim}{r}_\eta|^2 (x_{\xi\xi} + Px_\xi) + |\underset{\sim}{r}_\zeta|^2 (x_{\eta\eta} + Qx_\eta) - 2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta) x_{\xi\eta} \tag{26a}$$

$$Dy \equiv |\underset{\sim}{r}_\eta|^2 (y_{\xi\xi} + Py_\xi) + |\underset{\sim}{r}_\xi|^2 (y_{\eta\eta} + Qy_\eta) - 2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta) y_{\xi\eta} \tag{26b}$$

$$G \equiv [(1+f_y^2) f_{xx} + (1+f_x^2) f_{yy} - 2f_x f_y f_{xy}] / (1 + f_x^2 + f_y^2) \tag{27}$$

Note that all the derivatives of f, and hence G, can be evaluated from the defining function f(x,y) of the surface. Note also that, again using Eq.(18) for the z derivatives, we have

$$|\underset{\sim}{r}_\eta|^2 = (1+f_x^2)x_\eta^2 + (1+f_y^2)y_\eta^2 + 2f_x f_y x_\eta y_\eta \tag{28a}$$

$$|\underset{\sim}{r}_\xi|^2 = (1+f_x^2)x_\xi^2 + (1+f_y^2)y_\xi^2 + 2f_x f_y x_\xi y_\xi \tag{28b}$$

$$\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta = (1+f_x^2)x_\xi x_\eta + (1+f_y^2)y_\xi y_\eta + f_x f_y (x_\xi y_\eta + x_\eta y_\xi) \tag{28c}$$

Eq.(25) can thus be used to generate a two-dimensional coordinate system on a specified curved surface. These equations are applicable to a surface specified by the equation z = f(x,y). Analogous equations can, of course, be inferred for other surfaces. The assumptions made in obtaining these equations all relate to the behavior of the coordinate lines crossing the surface, i.e., that these lines are normal to the surface and have vanishing curvature at the surface. The curvature of the surface is taken account of through the terms involving the function G, without which Eq.(25) reduce to the plane two-dimensional equation (13).

A three-dimensional coordinate system can be constructed by connecting corresponding points on a sequence of surfaces on which two-dimensional systems have been generated by Eq.(25), but the equation of each surface would have to be specified, of course. The resulting three-dimensional system will not necessarily actually have the coordinate lines crossing the surfaces normally with vanishing curvature, however, since the successive surfaces are specified independently.

Gaussian surfaces. Another approach to the generation of three-dimensional coordinate systems by generating two-dimensional systems on surfaces is given by Warsi elsewhere in this volume, using the Gaussian equations for a surface. In contrast to the development of Eq.(25), here the surface is not specified. Therefore, three coupled two-dimensional equations result on the surface of constant ζ,

$$Dx - XR = 0 \tag{29a}$$

$$Dy - YR = 0 \tag{29b}$$

$$Dz - ZR = 0 \tag{29c}$$

where D is again defined by Eq.(26), and (X, Y, Z) are the components of the normal to the surface,

$$X = (y_\xi z_\eta - y_\eta z_\xi)/\sqrt{J} \tag{30a}$$

$$Y = (x_\eta z_\xi - x_\xi z_\eta)/\sqrt{J} \tag{30b}$$

$$Z = (x_\xi y_\eta - x_\eta y_\xi)/\sqrt{J} \tag{30c}$$

$$R = (Xx_\zeta + Yy_\zeta + Zz_\zeta)(g_{11}\Gamma^3_{22} - 2g_{12}\Gamma^3_{12} + g_{22}\Gamma^3_{11}) \tag{31}$$

and Γ^i_{jk} are the space Christoffel symbols:

$$\Gamma^i_{jk} = \frac{1}{2}\sum_{i=1}^{3} g^{\ell i}\left(\frac{\partial g_{j\ell}}{\partial x^k} + \frac{\partial g_{k\ell}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^\ell}\right) \tag{32}$$

In this procedure the ζ-derivatives, rather than the equation of the surface, must be specified. This can be done evaluating these derivatives on specified bounding surfaces intersecting the surface in question at its edges, and interpolating these boundary values onto the surface for use in Eq.(31). Again three-dimensional coordinate systems can be constructed by connecting corresponding points on the successive surfaces.

CONTROL FUNCTIONS

As discussed above, negative values of the control function P^i will cause the ξ^i coordinate lines to concentrate toward lines with lower values of ξ^i. Several approaches to the determination of the control functions are discussed below.

Attraction to coordinate lines/points

This effect was utilized in the TOMCAT Code of Thompson, et al.[3] to achieve attraction of coordinate lines to other coordinate lines and/or points by taking the form of the control functions to be, in 2D,

$$P(\xi,\eta) = -\sum_{i=1}^{n} a_i \,\mathrm{sign}(\xi - \xi_i)\exp(-c_i|\xi-\xi_i|) - \sum_{i=1}^{m} b_i \,\mathrm{sign}(\xi - \xi_i)\exp\{-d_i[(\xi - \xi_i)^2 + (\eta - \eta_i)^2]^{\frac{1}{2}}\} \tag{33}$$

and an analogous form for $Q(\xi,\eta)$ with ξ and η interchanged. (Here the subscripts identify particular ξ-coordinate lines and are not to be confused with the superscripts used to refer to the curvilinear coordinates in general.) In this form, the control functions are functions only of the curvilinear coordinates.

In the P function, the effect of the amplitude a_i is to attract ξ-coordinate lines toward the ξ_i-line, while the effect of the amplitude b_i is to attract ξ-lines toward the single point (ξ_i, η_i). Note that this attraction to a point is actually attraction of ξ-lines to a point on another ξ-line, and as such acts normal to the ξ-line through the point. There is no attraction of η-lines to this point via the P function. In each case the range of the attraction effect is determined by the decay factors, c_i and d_i. With the inclusion of the sign changing function, the attraction occurs on both sides of the ξ-line or the (ξ_i, η_i) point, as the case may be. Without this function, attraction occurs only on the side toward increasing ξ, with repulsion occurring on the other side. A negative amplitude simply reverses all of the above-described effects, i.e., attraction becomes repulsion and vice versa. The effect of the Q function on η-lines follows analogously.

In the case of a boundary that is an η-line, positive amplitudes in the Q function will cause η-lines off the boundary to move closer to the boundary, assuming that η increases off the boundary. The effect of the P function will be to alter the angle at which the ξ-lines intersect the boundary, since the points on the boundary are fixed, with the ξ-lines tending to lean in the direction of decreasing ξ. These effects are evident in Figures 2 and 3 above. Further examples are given below.

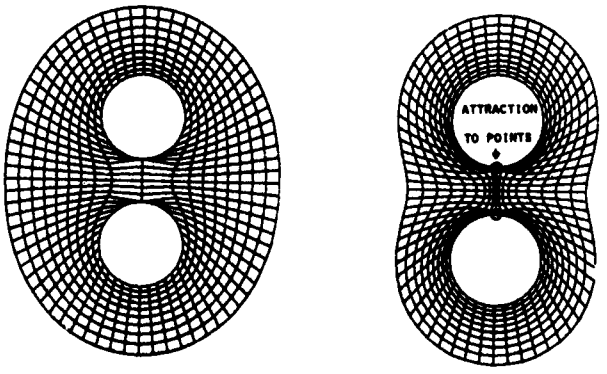


Fig. 6.

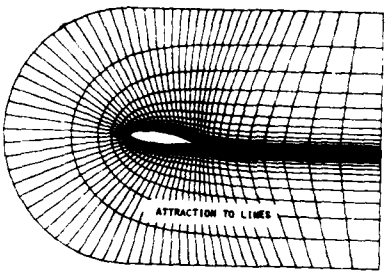


Fig. 7.

The first two figures show the result of attraction to the two circled points in comparison with the case with no control function. The other figure illustrates strong attraction to the coordinate line coincident with the inner boundary and the branch cut in this C-type system. If the boundary is such that η decreases off the boundary, then the amplitudes in the Q function must be negative to achieve attraction to the boundary. In any case, the amplitudes a_i cause the effects to occur all along the boundary (as in the last figure above), while the effects of the amplitudes b_i occur only near selected points on the boundary (cf. Fig. 3 above.)

Effect of cuts. In configurations involving branch cuts, the attraction lines and/or points in this type of evaluation of the control functions should be considered to exist on all sheets. In the O-type configuration shown in Fig. 13 (cf. also Fig. 26) of the first paper of this volume, where the two sides of the cut are on opposite sides of the transformed region, the control function P for attraction to the ξ_i-line would strictly involve an infinite summation over k, with ξ_i replaced by $\xi_i + k\Delta\xi$ where $\Delta\xi$ is the jump in ξ at the cut. Thus ξ_i in Eq.(33) would be replaced by $\xi_i + k\Delta\xi$ and the right side would be summed from $k = -\infty$ to $+\infty$. However, because of the exponential decay the terms usually decrease rapidly as k increases, so that only two terms in the k summation really need be included. Note that since there is no jump in η across the cut in this configuration, the evaluation of Q is affected by this cut only through the replacement of ξ_i as above in the term for the point attraction, with summation over k of only this part of the right side.

For the C-type configuration of Fig. 14 (cf. also Fig. 27) of the first paper, with the two sides of the cut on the same side of the transformed region, η is reflected in the cut, so that in the evaluation of the control function Q, two terms should appear for each i, in the second of which η_i would be replaced by $-\eta_i + 2\eta_c$ where η_c is the value of η on the cut. Again, however, the contribution of the second term may be small because of the exponential decay.

Here the evaluation of the control function P is affected by the cut only through the point attraction part, with η_i replaced as above.

The third type of cut illustrated in the first paper, Fig. 18 (cf. also Fig. 28), for which the two sides of the cut face across a void of the transformed region, is the most simple since here the jump in η across the void is simply to be removed from the distance to the attraction line by replacing η_i with $\eta_i - \Delta\eta$ in both the control functions. There is no additional summation in this case.

The case in Fig. 20 of the first paper, where the coordinate species changes sign at the cut, requires individual attention at each cut. For example, the contribution to the control functions in region 1 at a point (ξ,η) from an attraction site (ξ_i,η_i) in region 2 would be evaluated using distances of $(\xi - \xi_{max}) + (\eta_{max} - \eta_i)$ and $(\eta - \xi_i)$, in place of ξ_i and η_i, respectively.

Attraction to lines/points in space

If the attraction line and/or the attraction points are in the field, rather than on a boundary, then the above attraction is not a fixed line or point in space, since the attraction line or points are themselves determined by the solution of the generation system and hence are free to move. It is, of course, also possible to take the control functions to be functions of x and y, instead of ξ and η, and thus achieve attraction to fixed lines and/or points in the physical field. This case becomes somewhat more complicated, since it must be ensured that coordinate lines are not attracted parallel to themselves. The following development is from Thompson[7].

Recall that in the above discussion, η-lines are attracted to other η-lines, and ξ-lines are attracted to other ξ-lines. It is unreasonable, of course, to attempt to attract η-lines to ξ-lines, since that would have the effect of collapsing the coordinate system. When, however, the attraction is to be to certain fixed lines in the physical region, defined by curves $y = f(x)$, care must be exercised to avoid attempting to attract coordinate lines to specified curves that cut the coordinate lines at large angles. Thus, in the figure below,

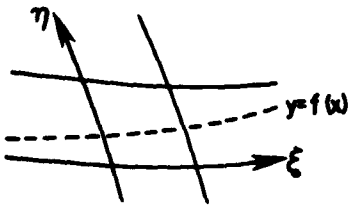


Fig. 8.

it is unreasonable to attract ξ-lines to the curve f(x), while it is natural to attract the η-lines to f(x).

However, in the general situation, the specified line f(x) will not necessarily be aligned with either a ξ or η line along its entire length. Since it is unreasonable to attract a line tangentially to itself, some provision is necessary to decrease the attraction to zero as the angle between the coordinate line and the given line f(x) approaches 90°. This can be accomplished by multiplying the attraction function by the cosine of the angle between the coordinate line and the line f(x). It is also necessary to change the sign on the attraction function on either side of the line f(x). This can be done by multiplying by the sine of the angle between the line f(x) and the vector to the point on the coordinate line.

These two purposes can be accomplished as follows. Let a general point on the ξ-line be located by the vector $\underset{\sim}{R}(x,y)$, and let the attraction line $y = f(x)$ be specified by the collection of points $S(x_i,y_i)$, $i = 1, 2, --, n$. Let the unit tangent to the attraction line be $\underset{\sim}{t}(x_i,y_i)$, and the unit tangent to a ξ-line be $\underset{\sim}{\tau}^{(\xi)}$. Then with $\underset{\sim}{k}$ the unit vector normal to the two dimensional plane, and with reference to the following figure,

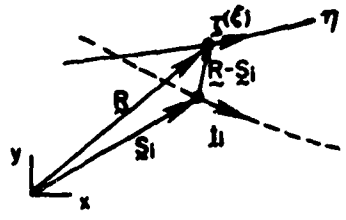


Fig. 9.

the control functions P(x,y) and Q(x,y) may logically be taken as

$$P(x,y) = -\sum_{i=1}^{n} a_i(\underset{\sim}{t}_i \cdot \underset{\sim}{\tau}^{(\xi)}) \frac{[\underset{\sim}{t}_i \times (\underset{\sim}{R} - \underset{\sim}{S}_i)] \cdot \underset{\sim}{k}}{|\underset{\sim}{R} - \underset{\sim}{S}_i|} \exp(-d_i|\underset{\sim}{R} - \underset{\sim}{S}_i|) \qquad (34)$$

The equation for Q simply has ξ replaced by η in the above. These functions depend on x and y through both $\underset{\sim}{R}$ and $\underset{\sim}{\tau}^{(\xi)}$ or $\underset{\sim}{\tau}^{(\eta)}$, and thus must be recalculated at each point as the iterative solution proceeds. This form of coordinate control will therefore be more expensive to implement than that based on attraction to other coordinate lines.

There is no real distinction between "line" and "point" attraction with this type of attraction. "Line" attraction here is simply attraction to a group of points that form a line, f(x). If line attraction is specified, then

the tangent to the line f(x) is computed from the adjacent points on the line. If point attraction is specified, then the "tangent" must be input for each point. The tangents to the coordinate lines are computed from relations given in the first paper of this volume,

$$\underset{\sim}{\tau}^{(\xi)} = \frac{1}{\sqrt{g_{22}}} (i x_\eta + j y_\eta) \tag{35a}$$

$$\underset{\sim}{\tau}^{(\eta)} = \frac{1}{\sqrt{g_{11}}} (i x_\xi + j y_\xi) \tag{35b}$$

Effect of cuts. The presence of branch cuts introduces no complication with this type of attraction since the distances involved are in terms of the cartesian coordinates, rather than the curvilinear coordinates. This form of attraction makes the control functions dependent on both the curvilinear and cartesian coordinates, and thus attraction to space lines and/or points involves more complicated equations in the transformed region than does attraction to other coordinate lines and/or points, since for the former the coefficients of the first derivatives are functions of the dependent variables.

Determination from related systems

If the cartesian coordinates were known in the field, i.e., the coordinate system had already been generated, the control functions used could obviously be determined at each point by substituting these known values in the generation system equations and solving algebraically for the control functions. More important, control functions for use in a general case might be determined from a more simple case of related geometry. The effect of the control would be qualitatively the same in the more general case.

Thus, the control functions for a case of a rectangular physical region with coordinate lines parallel to the sides are given by one-dimensional equations as in Eq.(10) which are, with ξ-lines parallel to the y-axis and η-lines parallel to the x-axis,

$$P = -\frac{x_{\xi\xi}}{x_\xi}, \quad Q = -\frac{y_{\eta\eta}}{y_\eta} \tag{36}$$

for use in Eq.(13). The effect of the use of these control functions, evaluated from arc lengths along the boundaries, in related, but more general, regions would be qualitatively the same.

Similarly, for an annular physical region bounded by two concentric circles, a solution of Eq.(13) of the form $x = r(\eta)\cos\theta(\xi)$, $y = r(\eta)\sin\theta(\xi)$ exists for the control functions, given by

$$P = -\frac{\theta''}{\theta'}\,,\quad Q = \frac{r'}{r} - \frac{r''}{r'} \tag{37}$$

which may be verified by direct substitution. If the control functions determined by substituting desired radial and angular point distributions in Eq.(37) are used in a more general case with opposing η-line boundaries, the line spacing will be qualitatively the same as these point distributions. This topic is treated in more general form in the next section.

Results of the two applications discussed above are shown in the following figures, the first of which illustrates the application of the one-dimensional evaluation from arc length along the boundary of a simply-connected region, while in the second the evaluation is from Eq.(37) using concentric circles of diameters equal to the maximum chord of the true inner and outer boundaries.

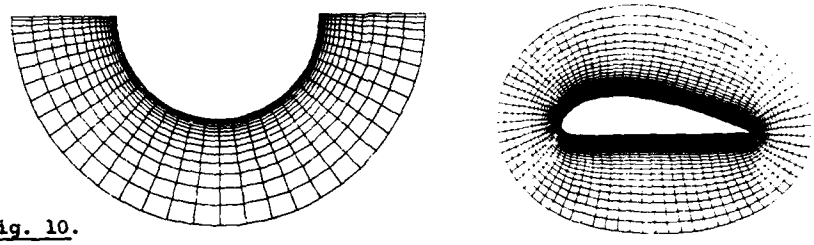


Fig. 10.

Another approach would be to use a different generation procedure, e.g., algebraic, to generate a preliminary coordinate system for the same configuration from which to determine the control functions by substitution in Eq.(13). (The control functions could be smoothed before use in the elliptic generation system if desired.) In this way some of the advantageous features of other generation systems could be employed, while using the inherent smoothness of the elliptic systems to produce a final system without slope discontinuities in the coordinate lines.

Determination from boundary point distributions

The more general question of the determination of the control functions so as to reflect the boundary point distributions directly in the spacing of the coordinate lines in the field for general regions is still under study. A reasonable approach in some cases is that of Thomas[6], which is outlined below and discussed at greater length by Thomas elsewhere in this volume.

Determination from edge distributions. The projection of the two-dimensional vector equation (13) in the direction tangent to a line of constant η is obtained by taking the dot product with $\underset{\sim}{r}_\xi$. If, for the moment, the coordinate lines are assumed to be orthogonal on this η-line, so that $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta$ vanishes thereon, this projection will be

$$|\underset{\sim}{r}_\eta|^2 (\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\xi} + P|\underset{\sim}{r}_\xi|^2) + |\underset{\sim}{r}_\xi|^2 (\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\eta\eta}) = 0 \tag{38}$$

Then, solving for the control function P, we have

$$P = -\frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\xi}}{|\underset{\sim}{r}_\xi|^2} - \frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\eta\eta}}{|\underset{\sim}{r}_\eta|^2} \tag{39}$$

Now the derivative of arc length along the η-line is

$$s_\xi^{(\eta)} = |\underset{\sim}{r}_\xi| \tag{40}$$

so that the first term in this expression for P is

$$\frac{s_{\xi\xi}^{(\eta)}}{s_\xi^{(\eta)}} = \frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\xi}}{|\underset{\sim}{r}_\xi|^2} \tag{41}$$

and is thus related to the rate of change of the arc length spacing along the η-line. This term can be evaluated from the point distribution, $\underset{\sim}{r}(\xi)$, along the η-line.

In regard to the other term in P, the principal curvature of the ξ-lines crossing the η-line is given by Eq.(20) with ζ replaced by η. Then, under the assumption that the coordinate lines are orthogonal on the η-line so that $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta = 0$ and $\underset{\sim}{N} = \frac{\underset{\sim}{r}_\xi}{|\underset{\sim}{r}_\xi|}$, we have for this curvature

$$C^{(\xi)} = \frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\eta\eta}}{|\underset{\sim}{r}_\xi||\underset{\sim}{r}_\eta|^2} \tag{42}$$

Thus the other term in the expression above for P is the product of the arc length spacing, $s_\xi^{(\eta)} = |\underset{\sim}{r}_\xi|$, along the η-line with the local curvature $C^{(\xi)}$ of the ξ-lines crossing the η-line. Since this term involves η-derivatives, it cannot be evaluated from the point distribution on the η-line. The distribution of the local curvature of the ξ-lines must be specified along the η-line, either by direct specification or by interpolation between points where a specified ξ-line is available, e.g., boundary lines on the ends of the η-line.

We thus have a procedure for evaluating the control function P along an η-line if the point distribution thereon is specified and if the distribution of local curvature of the ξ-lines crossing this η-line is also specified, either directly or by interpolation between known ξ-lines:

$$P = -\frac{s_{\xi\xi}^{(\eta)}}{s_{\xi}^{(\eta)}} - s_{\xi}^{(\eta)} C^{(\xi)} \tag{43}$$

using Eq.(41) and (42) above. (Recall that this evaluation is made under the assumption that the coordinate lines are orthogonal on the η-line.)

A similar equation can be obtained for the evaluation of the control function Q along a ξ-line:

$$Q = -\frac{s_{\eta\eta}^{(\xi)}}{s_{\eta}^{(\xi)}} - s_{\eta}^{(\xi)} C^{(\eta)} \tag{44}$$

where now the arc length is along the ξ-line and the curvature is that of the η-lines crossing this line:

$$\frac{s_{\eta\eta}^{(\xi)}}{s_{\eta}^{(\xi)}} = \frac{\underset{\sim}{r}_{\eta} \cdot \underset{\sim}{r}_{\eta\eta}}{|\underset{\sim}{r}_{\eta}|^2} \tag{45}$$

$$C^{(\eta)} = \frac{\underset{\sim}{r}_{\eta} \cdot \underset{\sim}{r}_{\xi\xi}}{|\underset{\sim}{r}_{\eta}|\,|\underset{\sim}{r}_{\xi}|^2} \tag{46}$$

If the control functions are evaluated in this manner along pairs of facing boundaries in the transformed region, values for these functions in the interior can be obtained by interpolation between the corresponding facing boundaries, i.e., interpolation for P between η-line boundaries and for Q between ξ-line boundaries as illustrated below.

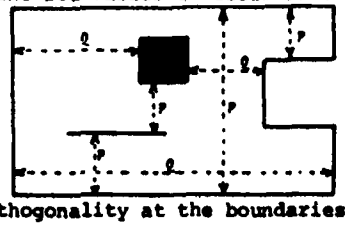


Fig. 11.

Note that although orthogonality at the boundaries was assumed in the development of the above relations for the control functions thereon, there has be n no enforcement of this condition. Therefore, the resulting coordinate system will not necessarily be orthogonal anywhere. Control functions determined in this manner will, however, serve to project the influence of

the boundary point distributions into the field, so that the line spacing in the field will generally follow that on the boundaries.

Determination from surface distributions

These ideas may be extended to three dimensions, with the control functions first being determined from the point distribution on the edges of a surface on which a curvilinear coordinate is constant in the manner discussed above for the two-dimensional case. These edge values are then interpolated onto the surface, and a two-dimensional solution is done for the coordinate system on the surface as discussed above. Then the new control functions are determined from the resulting point distribution on the surface. When this has been done for all the boundary surfaces, the surface control functions are interpolated into the interior of the three-dimensional region.

The surface solution requires a version of the two-dimensional equations that takes into account the curvature of the surface, such as Eq.(25) given above. As in the two-dimensional case, the control function P for use in Eq.(25) can be determined from the projection of Eq.(17) along a coordinate line of constant η, with the momentary assumption that the coordinate lines are orthogonal on this line. Thus dotting $\underset{\sim}{r}_\xi$ into Eq.(17), and taking $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta = 0$, as well as the previously assumed condition of $\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\zeta = 0$, we have

$$P = -\frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\xi}}{|\underset{\sim}{r}_\xi|^2} - \frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\eta\eta}}{|\underset{\sim}{r}_\eta|^2} - \frac{\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\zeta\zeta}}{|\underset{\sim}{r}_\zeta|^2} \tag{47}$$

But by Eq.(21) we have

$$\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\zeta\zeta} = \frac{\underset{\sim}{r}_\zeta \cdot \underset{\sim}{r}_{\zeta\zeta}}{|\underset{\sim}{r}_\zeta|^2} \underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\zeta = 0$$

so that the expressions for the control functions, P and Q, reduce to the expressions obtained in the two-dimensional case, Eq.(43) and (44).

With the points known on a ζ-surface, values of the control function thereon can be determined by forming the projection of Eq.(17) along lines of constant ξ and along lines of constant η on this surface, again assuming that the ζ-lines emanating from the surface are normal thereto. Thus, dotting $\underset{\sim}{r}_\xi$ and $\underset{\sim}{r}_\eta$ into Eq.(17), we have

$$\begin{pmatrix} |\underset{\sim}{r}_\eta|^2 & \underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta \\ \underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta & |\underset{\sim}{r}_\xi|^2 \end{pmatrix} \begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} \dfrac{2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta)(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\eta})}{|\underset{\sim}{r}_\xi|^2} - \dfrac{|\underset{\sim}{r}_\eta|^2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\xi\xi})}{|\underset{\sim}{r}_\xi|^2} - \underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_{\eta\eta} \\ \dfrac{2(\underset{\sim}{r}_\xi \cdot \underset{\sim}{r}_\eta)(\underset{\sim}{r}_\eta \cdot \underset{\sim}{r}_{\xi\eta})}{|\underset{\sim}{r}_\eta|^2} - \dfrac{|\underset{\sim}{r}_\xi|^2(\underset{\sim}{r}_\eta \cdot \underset{\sim}{r}_{\eta\eta})}{|\underset{\sim}{r}_\eta|^2} - \underset{\sim}{r}_\eta \cdot \underset{\sim}{r}_{\xi\xi} \end{pmatrix} \tag{48}$$

which can be solved for P and Q on the ζ-surface.

In like manner, values of the control functions P and R can be determined on an η-surface from a point distribution thereon, and the functions Q and R can be determined on a ξ-surface. The function P in the interior of the three-dimensional region then is determined by interpolation of the values on facing η-surfaces. The functions Q and R in the interior are determined in a similar manner using ξ and ζ surfaces and η and η surfaces, respectively.

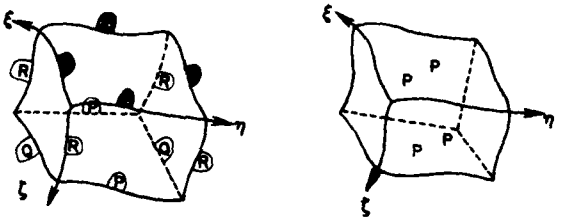


Fig. 12.

Automatic determination

Another approach to the determination of the control functions is to interatively adjust their values until some desired specification is achieved. Thus in Sorenson[8], in two dimensions, the control functions P and Q of Eq.(2) are iteratively adjusted to achieve a specified spacing of the first coordinate line from the boundary and a specified angle of intersection at the boundary. Here the control functions are taken to be of the form

$$P(\xi,\eta) = p(\xi)\exp(-a\eta) + r(\xi)\exp[-c(\eta_{max} - \eta)] \tag{49a}$$

$$Q(\xi,\eta) = q(\xi)\exp(-b\eta) + s(\xi)\exp[-d(\eta_{max} - \eta)] \tag{49b}$$

With the desired intersection angle at the boundary and the spacing of the first line off the boundary specified at each boundary point

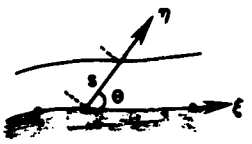


Fig. 13.

(and with a priori choices for the decay factors), the code determines the values of the functions of ξ in the control functions automatically as follows: specification of the intersection angle and first line spacing at the boundary points determines x_η and y_η at each boundary point, values of x_ξ and y_ξ being known from the boundary point distribution. Since $x_{\xi\eta}$ and $y_{\xi\eta}$ can then be determined at each boundary point, and $x_{\xi\xi}$ and $y_{\xi\xi}$ are known from the boundary point distribution, it remains only to determine $x_{\eta\eta}$ and $y_{\eta\eta}$ at each boundary point in order to evaluate the functions of ξ in (4) at each boundary point. With assumed values for these functions, the system of Eq.(2) is solved on the field. Values of $x_{\eta\eta}$ and $y_{\eta\eta}$ at each boundary point are then determined from this solution on the field, and new values of the functions of ξ are calculated at each boundary point. This process is repeated until convergence. This procedure is utilized in the GRAPE code and its extensions which is discussed by Sorenson elsewhere in this volume.

NUMERICAL SOLUTION

The elliptic generating system of equations can be solved numerically in a variety of ways. The first step is generally to replace the derivatives with difference expressions, and second-order central differences are most widely used. (There have been some solutions using first-order, directed one-sided differences for the first derivatives, keyed to the sign of the control functions.) The representation of difference expressions across branch cuts is discussed in the first paper of this volume. The solution is thus cast on the square grid of the rectangular transformed region. Values of the cartesian coordinates (or other basis system) and/or the coordinate line slopes are specified on the boundary of the transformed region, except on the re-entrant portion corresponding to the branch cuts. The non-linear difference equations can then be solved by iteration.

Iterative solutions

The most simple and the easiest iterative procedure to apply in general configurations is point SOR (cf. Thompson[3]). The convergence for Eq.(12)

has been found to be rapid and dependable for a wide variety of configurations using overrelaxation. The optimum acceleration parameters and the convergence rate decrease as the control functions increase in magnitude. Some consideration has been given to the calculation of a field of locally-optimum acceleration parameters (cf. Thompson[3]), but the predicted values generally tend to be too high, and the desired increases in convergence rate were not obtained. Since the equations are nonlinear, it is necessary that the initial guess lie in a neighborhood of the solution. A logical and versatile procedure is to use linear interpolation between closest facing boundary segments for the initial guess as illustrated below.

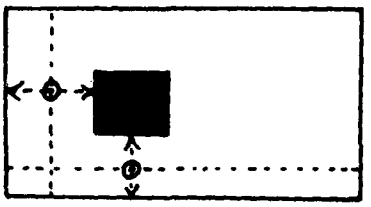


Fig. 14.

The generating system has also been solved using SLOR-iteration as in Sorenson[8], ADI iteration (cf. Ghia, et al.[9]), the strongly-implicit procedure SIP (Ghia & Ghia[10]), and by multi-grid procedures as in Camarero and Younis[11].

Problems

Since the coordinate lines tend to concentrate near a convex boundary, very sharp convex corners may cause problems with the convergence of iterative solutions of the generation equations. These equations are nonlinear, and therefore convergence of an iterative procedure requires that the initial guess be within some neighborhood of the solution. With control functions designed to cause attraction to the boundary, it is possible for the coordinate lines to overlap a very sharp convex corner during the course of the iteration even though a solution with no overlap exists.

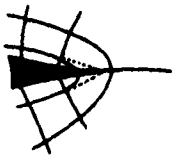


Fig. 15.

This problem may be handled by first converging the solution with the coordinate lines artificially locked off the corner. Thus, if newly calculated values of the cartesian coordinates at a point during the iteration would cause this point to move farther from its present location

than the distance to the adjacent point on the curvilinear coordinate line to the corner, then these new values are replaced by the average of the coordinates of the old point and the adjacent point. After convergence, this lock is removed and final convergence to the solution is obtained. Note that this problem does not arise when the curvilinear coordinate line emanating from the corner is the same as that on the boundary, as in the C-type configuration of Fig. 14 of the first paper of this volume,

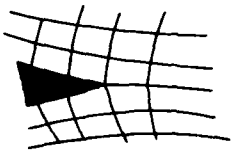


Fig. 16.

since then the lines do not wrap around the corner.

With very large cell aspect ratio, e.g., for $g_{11} >> g_{22}$, the generation equation is dominated by the term containing the second derivative along the curvilinear coordinate line on which the shorter arc length lies. This causes the cartesian coordinates to tend strongly toward averages of adjacent points on this line during the course of the iteration. Therefore, when strong control functions are used to attract coordinate lines to the boundary in a C-type configuration,

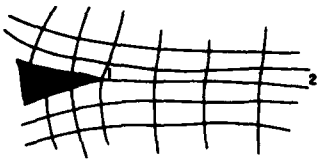


Fig. 17.

the points on the line 1 - 2 are very slow to move from the initial guess during the iteration. Convergence in such a case is very slow, and it is expedient to artificially fix the points on such a line as if it were a boundary. This will cause the coordinate lines crossing this line to have discontinuous slopes at this line, but since the spacing along these crossing lines is very small, the error thus incurred in difference solutions on the coordinate system is small.

CONCLUSION

The generation of boundary-conforming coordinate systems through numerical solution of elliptic partial differential equations has great versatility and is capable of treating a wide range of configurations without requiring special adaptation. This type of generation system provides smooth coordinate

systems and does not propagate boundary slope discontinuities into the field. The coordinate line distribution can be controlled through functions in the elliptic equations, and effective means for the evaluation of such functions have been developed. The control functions can be automatically determined to control both the line spacing off a boundary and the angle of intersection of the lines with the boundary. These generating systems can be quickly and effectively applied in the numerical solution of partial differential equations for physical problems. The necessary transformation relations therefor are given in the first paper of this volume, where various possible configurations of the transformed plane and the treatment of branch cuts and other special points are also discussed. As noted at the beginning, further discussions and applications of elliptic generation systems appear in other papers of this volume. Further development is needed particularly in three dimensions and in improvement of the automation of the determination of the control functions, especially for more complicated configurations.

ACKNOWLEDGMENTS

These developments were made in the effort under the following sponsorships:

Grant NGR 25-001-055, NASA Langley Research Center

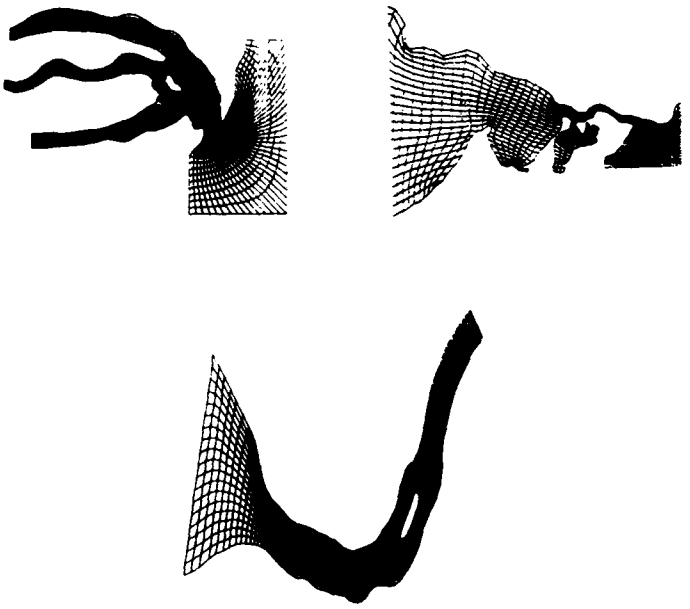
Grant AFOSR 80-0185, U. S. Air Force Office of Scientific Research

Contract DACW39-78-C-0054, U. S. Army Engineer Waterways Experiment Station

REFERENCES

1. Thompson, J. F., Zahir U. A. Warsi, and C. Wayne Mastin. "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," Journal of Computational Physics, to appear in mid-1982.
2. Mastin, C. W. and Thompson, J. F. "Elliptic Systems and Numerical Transformations," J. Math. Anal. App., 62, 52 (1978).
3. Thompson, J. F., Thames, F. C., and Mastin, C. W. "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, 24, 274 (1977).
4. Godunov, S. K. and Prokopov, G. P. "The Use of Moving Meshes in Gas-Dynamical Computations," USSR Comp. Math. Math. Phys., 12, 182 (1972).
5. Winslow, Alan M. "Adaptive Mesh Zoning by the Equipotential Method," UCID-19062, (1981).
6. Thomas, P. D. "Construction of Composite Three Dimensional Grids from Subregion Grids Generated by Elliptic Systems," AIAA Computational Fluid Dynamics Conference, Palo Alto, 24 (1981).
7. Thompson, J. F. "WESCOR - Boundary-Fitted Coordinate Code for General 2D Regions with Obstacles and Boundary Intrusions," Contractor Report, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Mississippi (1982).

8. Sorenson, R. L. "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poisson's Equation," NASA TM-81198, (1980).
9. Ghia, U., Ghia, K. N. and Studerus, C. J. "Use of Surface-Oriented Coordinates in the Numerical Simulation of Flow in a Turbine Cascade," Lecture Notes in Physics, 59, 197 (1976).
10. Ghia, U. and Ghia, K. N. "Boundary-Fitted Coordinates for Regions with Highly-Curved Boundaries and Reentrant Boundaries," Numerical Grid Generation Techniques, NASA CP-2166, 295 (1980).
11. Camarero, R. and Younis, M. "Efficient Generation of Body-Fitted Coordinates for Cascades Using Multigrid," AIAA Journal, 18, 487 (1980).



AD P000970

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

Conformal Grid Generation

by

David C. Ives
United Technologies Corporation
Pratt & Whitney Aircraft Group
East Hartford, Connecticut 06108

ABSTRACT

This paper treats conformal mapping as it relates to the generation of grids to be used for flow simulation. Classical and contemporary mapping methods are discussed and compared in some detail. Numerous suggestions are included to help the reader avoid common pitfalls, and the mapping methods believed most promising are identified.

INTRODUCTION

Conformal mapping is a versatile component of the spectrum of grid generation techniques, yet its use is sometimes avoided by investigators who feel uneasy about its implementation due to unfamiliarity. The object of this paper is to put conformal mapping into perspective, to discuss alternate implementations, and to induce the reader to use conformal mapping, when appropriate, as one component of a grid generation process.

Although this paper discusses the generation of grids using conformal mapping techniques, the grids themselves are not restricted to being conformal or orthogonal. Algebraic techniques are considered for the latter stages of a mapping sequence, and elliptic techniques are considered for "filling-in" the interior of a grid once the boundary correspondences have been obtained through a mapping sequence. The emphasis is on grids on a surface, rather than throughout a volume, since a conformal mapping is basically a surface-to-surface correspondence.

This paper starts with a discussion of the basic differences between algebraic, orthogonal, and conformal coordinate systems. Some implications of these differences are considered as they relate to flow simulation and design considerations. A brief review of complex variable notation and various types of conformal mappings is followed by a discussion of the problem of multiple valuedness, which can be the most troublesome aspect of conformal mapping from a computer implementation viewpoint. Various mapping techniques in current use are reviewed and techniques useful for creating new conformal mappings are outlined. The generation of a grid, once the mapping is known, is discussed.

PREVIOUS PAGE
IS BLANK

Suitable reference material is listed by problem type and packaged computational tools are described. The closing remarks highlight techniques that are believed to be the most promising in this evolving field.

COORDINATE TRANSFORMATIONS

This section illustrates some of the differences between general, independent algebraic, orthogonal, and conformal coordinate systems.

A general two-dimensional transformation between (x,y) physical coordinates and (ξ,η) computational coordinates (where (ξ,η) are taken to be orthogonal throughout this paper) can be written as

$$x = x(\xi,\eta)$$
$$y = y(\xi,\eta)$$

or in differential form in terms of a Jacobian matrix as

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{1}$$

where, from the chain rule, the four independent parameters a, b, c, and d are given by

$$a = \left.\frac{\partial x}{\partial \xi}\right|_{\eta}$$
$$b = \left.\frac{\partial x}{\partial \eta}\right|_{\xi}$$
$$c = \left.\frac{\partial y}{\partial \xi}\right|_{\eta}$$
$$d = \left.\frac{\partial y}{\partial \eta}\right|_{\xi} .$$

Here dx and dy are components of a differential vector dZ in the physical plane, while $d\xi$ and $d\eta$ are orthogonal (perpendicular) components of a differential vector $d\zeta$ in a computational plane. For a general algebraic transformation, dx and dy are not orthogonal, and their magnitudes need not be equal when the magnitudes of $d\xi$ and $d\eta$ are equal. An elliptic technique as in Ref. [1], for example, falls into the above class if orthogonality is not enforced.

A less general algebraic transformation (referred to as an independent algebraic transformation) composed of independent transformations in each direction can be written as

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} h & 0 \\ 0 & g \end{bmatrix} \cdot \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{2}$$

where h and g are two independent parameters. For this transformation, dx and dy are orthogonal but their magnitudes are not equal when the magnitudes of $d\xi$ and $d\eta$ are equal.

An orthogonal differential coordinate transformation can be written in a similar manner as

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} h & 0 \\ 0 & g \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{3}$$

where h, g, and θ are three independent parameters, θ is the angle between the dx and $d\xi$ directions, and dx is orthogonal to dy.

A conformal differential coordinate transformation can be written as

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} h & 0 \\ 0 & h \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{4}$$

where h and θ are two independent parameters. For this transformation, dx and dy are orthogonal and their magnitudes are equal when the magnitudes of $d\xi$ and $d\eta$ are equal. Thus the magnification h (called the metric or mapping modulus) between the (x,y) and (ξ,η) planes is not a function of direction; it is just a function of location. A conformal transformation is also angle preserving in a local sense, so that if two differential vectors meet at the angle α in the (ξ,η) plane, they also meet at the angle α in the (x,y) plane.

FLOW SIMULATION IMPLICATIONS

A grid can affect the storage and speed of a flow simulation. This section touches on some of the aspects of this dependence of a flow simulation on the grid.

The computer storage required for a grid was once an important consideration for flow simulation. For a finite difference solution, the independent parameters in the differential coordinate transforms above were usually stored (for an orthogonal grid) or regenerated as required (for a grid constructed using only two independent algebraic stretchings). For a limited set of flow problems, which fortunately includes two-dimensional transonic potential flow, it turns out that a conformal mapping requires the storage of only the metric, h, at each grid point in addition to the potential, ϕ. The x and y physical plane coordinates and the parameter θ in Eq. (4) are not needed throughout the grid in this case, due to the way the gradient operator transforms under a conformal mapping, so only two quantities need be stored for each grid point.

This is one reason why conformal grids were popular in the early days of computers for finite difference solutions using a velocity potential. Purely algebraic grids were also popular as they could be regenerated as required, so that only the potential need be stored at each grid point. The rapidly declining cost of computer memory has made storage requirements a minor issue* at many installations for two-dimensional and axisymmetric inviscid flow problems; in fact, a modern finite volume technique may routinely store twenty to thirty quantities at each grid point. The storage requirements of a grid system therefore need not be a deciding factor in the choice of a grid system.

The computer time required by a flow algorithm depends on the nature of the grid system. For example, when a conformal grid (or an orthogonal grid using a conformal step followed by independent algebraic stretchings) is used, a byproduct is the solution for the two-dimensional incompressible flow as discussed later. This incompressible solution may often be used with a compressibility transformation as an accurate first guess for an iterative solution of the compressible problem, reducing the computer time needed to converge the iterative process. In addition, a suitable conformal mapping often automatically provides a two-dimensional concentration of the mesh in regions of high gradient (because the mapping modulus is independent of direction for a conformal mapping), without simultaneously producing a one-dimensional concentration where not needed. This allows the use of fewer grid points, which results in faster calculations.

DESIGN IMPLICATIONS

A number of modern aerodynamic design methods, Refs. [2,3], wherein the pressure is specified and the body shape giving this pressure is calculated, use conformal transformation techniques. The computer programs required for such a conformal mapping based aerodynamic design process are minor variations of the conformal mapping programs required to generate an orthogonal grid. For self-consistency, one might as well use the same routines for both. This may prove to be a significant justification for conformal grid generation.

COMPLEX VARIABLE NOTATION

Complex variable notation allows us to write the differential conformal transformation, Eq. (4), in a compact form as

---

* In some computing installations, the charging algorithm recognizes this fact so that the charge for a calculation only depends on the CPU time and not on the product of CPU time and storage.

$$dx + idy = he^{i\theta}(d\xi+id\eta)$$

or $$dZ = H\, d\zeta$$

or $$Z = \int H\, d\zeta$$

or $$Z = F(\zeta)$$

where $$Z = x + iy$$

and $$H = he^{i\theta}$$

and $$\zeta = \xi + i\eta \quad .$$

Thus given the relation $Z = F(\zeta)$ we can easily calculate the mapping modulus as $h = \left|\frac{dZ}{d\zeta}\right| = |F'(\zeta)|$. Complex notation greatly simplifies the use of conformal transformation techniques, and is well supported* on most modern computing systems.

CONFORMAL MAPPING

Conformal mapping provides a surface correspondence that is not limited to planar surfaces. Conformal mappings from non-planar surfaces, such as from spheres or general cylindrical surfaces, to a plane have been employed by cartographers under the title of projections. We now will discuss some characteristics of these different types of mappings, starting with some simple planar mappings.

Planar mappings, relating points in two different planes, are the most familiar. An important planar mapping is the bilinear transform given by

$$\frac{(Z-a)}{(Z-b)}\,(c-a) = \frac{(\zeta-A)}{(\zeta-B)}\,(C-A) \quad , \qquad (5)$$

where the points $Z = a$, b, and c correspond to the points $\zeta = A$, B, and C respectively. It can be shown that Eq. (5) transforms circles or straight lines in the Z plane to circles or straight lines in the ζ plane. A region at infinity can be treated as a point in this equation. This mapping illustrates that one may specify the correspondence of only three specific points between planes (i.e., one may sssign a, b, c and A, B, C in an arbitrary manner). The bilinear transformation is often used after another conformal mapping step to produce a canonical (or standardized) contour for the next step.

---

* A breakup into real and imaginary parts, and then using real arithmetic only, can produce a two to one speed improvement on IBM systems. This means that inner loops for heavily used routines should be considered for such a breakup.

Often a contour is conformally mapped onto a unit circle and then the conformal transformation

$$\begin{aligned} \omega &= \ell n(\zeta) \\ &= \ell n(re^{i\theta}) \\ &= \ell nr + i\theta \end{aligned} \tag{6}$$

is applied, followed by the independent algebraic transformation

$$\begin{aligned} R &= e^{\ell nr} = r \\ \theta &= \theta \end{aligned} \tag{7}$$

to map the interior of the unit circle to the interior of the rectangle given by

$$0 \le R \le 1 \quad ,$$

$$0 \le \theta \le 2\pi \quad .$$

A uniform grid in this rectangle plane may then be constructed and mapped back to the physical plane where the resulting grid will be orthogonal.

A simple conformal mapping to map a general axisymmetric cylindrical surface onto a plane has been variously reported in the literature and is well recognized overseas, Refs. [4, 5, 6, 7, 8, 9], but evidently not at present in the United States. This mapping is given by

$$dZ = r\, d\zeta \quad , \tag{8}$$

where

$dZ = ds + ird\theta$, $d\zeta = d\xi + id\eta$, and

s = arc length along cylinder in axial direction $= \int [1 + \left(\frac{dr}{dx}\right)^2]^{\frac{1}{2}} dx$.

The modulus of this transformation is simply r, the radius in the cylindrical coordinate system (x,r,θ) describing the physical surface. For a constant radius axisymmetric cylinder, the radius r drops out of the flow equations and one can think of simply unwrapping the cylinder, but in fact one is implicitly performing a conformal mapping. For an axisymmetric conical surface, the above relation becomes

$$Z = e^{\zeta \sin\phi} \quad , \tag{9}$$

where ϕ is the angle between the axis and the cone surface. One obvious use of cylindrical mappings is in the generation of grids for quasi three-dimensional blade-to-blade flow simulations for axial, mixed (meaning both significant radial and axial flow components), and centrifugal flow turbomachines. For $\phi = \pi/2$, Eq. (9) reduces to Eq. (6) which has been used in Ref. [10] to map a centrifugal impeller with near log-spiral blades onto a two-dimensional cascade of blades.

The requirements for constructing a useful geographical or astronomical map (i.e: local angles are preserved and the local scale factor does not depend on orientation) are identical to the definition of a conformal mapping. Thus, the mappings developed by cartographers to represent a spherical surface by a planar map apply directly. Among such maps are stereographic and Mercator projections, and a more general projection used for star maps, Ref. [9]. Stereographic projections, invented by the Greek astronomer Ptolemy as cited in Ref. [11], were used to develop grid systems for the computation of supersonic flow over conical bodies, Ref. [12], and more general bodies, Ref. [13]. A spiral groove spherical bearing lubrication study, Ref. [14], also used stereographic projection. Stereographic projections have been known for 18 centuries; the recent innovation was in how to use them for flow calculations.

RIEMANN SHEET DETERMINATION

The bane of conformal mapping is the possibility of getting on the wrong Riemann sheet, or equivalently, choosing an inappropriate root. Development of conformal mapping computer programs would be substantially easier were it not for the multivalued nature of many mappings. This section discusses techniques that have proven useful for determining the appropriate root.

To illustrate the problem, consider the mapping

$$Z = \zeta^k \tag{10a}$$

where k is a real number. We can express ζ in polar coordinates as

$$\zeta = re^{i(\theta+2\pi n)} , \tag{10b}$$

where n is any integer, since

$$e^{i2\pi n} = 1 .$$

Then

$$Z = r^k e^{ik\theta} e^{ik2\pi n} .$$

For a square root transformation k = 1/2, and the $e^{ik2\pi n}$ term is 1 if n is even and -1 if n is odd so that there are two different solutions for Z for the same ζ (while there is only one value of ζ for each value of Z). This multivaluedness can be visualized in terms of Riemann sheets, with each different value of ζ lying on a different sheet as illustrated in Fig. 1. For k = 1/3, there are three values of Z for each value of ζ. For irrational values of k, there are an infinite number of values of Z for each value of ζ.

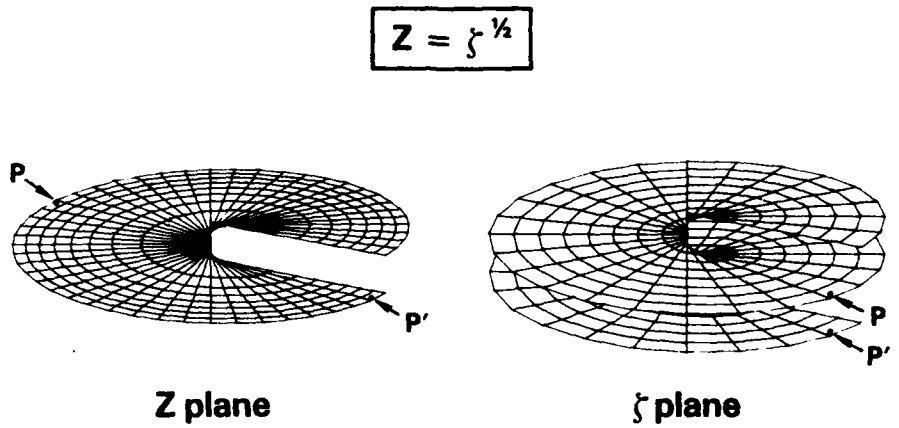


FIGURE 1. RIEMANN SHEETS

A computer implementation of $Z = \zeta^{1/2}$ using the CSQRT complex square root FORTRAN function will return only the root having a positive real part; the other root will be the negative of this root. It is up to the investigator to provide program logic to choose which of these two roots is appropriate.

For k = 1/2, which is rather common, physical reasoning can often be used to select the appropriate value of Z. In particular, mappings using the principle of reflection (covered later) produce quadratic equations with two roots which are image points with respect to a circle. In this case one may choose the root either inside or outside the circle based on physical reasoning. For the square root transformation, as often used in airfoil or wing grid generation, or the transformation,

$$Z = \zeta + e^{\zeta} \quad , \tag{11}$$

which also has two roots and is often used for inlet grid generation, the root selection is rather simple. Referring to Figs. 2 and 3, the initial value at "A" is the root with a negative real value, and as one traverses the airfoil in a clockwise direction and the inlet in a counterclockwise direction, the root to the right of, and (of those remaining, if any) nearest to, the previous root is chosen. This simple algorithm has worked well for a large variety of airfoil and inlet grid calculations.

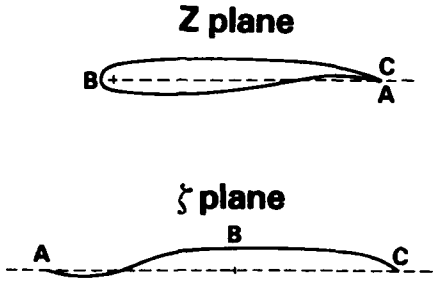


FIGURE 2. AIRFOIL MAPPING

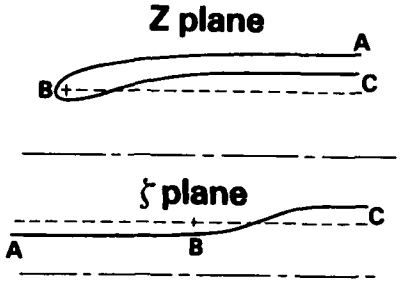


FIGURE 3. NACELLE MAPPING

For a transformation involving an irrational power of k near (but not exactly) 1/2, there are many values of Z that lie near each other and the selection is less obvious. The von Karman-Trefftz transformation, Ref. [15], given by

$$\frac{\zeta-a}{\zeta-b} = \left[\frac{Z-A}{Z-B}\right]^k \tag{12}$$

which transforms an airfoil in the Z plane into a near circle in the ζ plane, is one such case. When $k = \frac{1}{2}$, the trailing edge is cusped and Eq. (12) becomes the Joukowski transformation.

One technique to deal with the root problem is that of "tracking", where one chooses the value of n such that $(\theta+2\pi n)$ in Eq. (10b) is closest to the value of $(\theta+2\pi n)$ for a nearby point. The value of θ is obtained from the real and imaginary parts of ζ using the FORTRAN ATAN2 function, which returns a value of θ between $-\pi$ and π. In this manner one "tracks" the argument $(\theta+2\pi n)$ continuously along a curve and chooses n to make this argument continuous. There remains the problem of how to choose the root for the initial point. This may often be accomplished from knowledge of the behavior of the transformation at the point in the mapped plane which corresponds to the far field in the physical plane.

Another method of root selection is to introduce a "branch cut" based on physical reasoning, and then preassign n on each side of the cut. The choice of root for a reflection mapping as described earlier is equivalent to using a circle for the cut. For many problems the cut may be taken as a straight line and is chosen so as to not cross the boundary being mapped. Some mappings may require more than one cut, as illustrated in Refs. [16] and [17]. Using a branch cut can be considered as a static technique to preassign n on a global basis, while tracking is a dynamic technique to determine n on a local basis.

It is not unusual, for complicated mappings, to occasionally encounter root selection errors in a conformal mapping program which had been thought to be resistant to such effects. This usually happens when attempting to map a geometry significantly different from, but still in the same class as, the geometry used to develop the program. For simple mappings (i.e: those quoted so far in this discussion with the possible exception of Eq. (12)) root selection problems are not expected if the above techniques have been carefully implemented. In any case, an error is easily detected by defining a uniform grid in the rectangular computational plane and mapping this grid back to the physical plane where it is plotted. If the physical plane grid is continuous, does not overlap, and does not exhibit open sections or jumps, the roots have probably been selected correctly. It is important to visually inspect the grid created for each new geometry before it is used for a flow calculation if there is any doubt on this point.

CLASSICAL MAPPING

The classical technique of mapping consists of mapping a contour (an airfoil, a cascade, an inlet, etc.) to a near-circle by a single transformation or a sequence of simple transformations. The near-circle is then mapped to a circle by a near-circle to circle transformation, such as proposed by Theodorsen and Garrick, Ref. [18].

The curve to near-circle transformation(s) often require a substantial level of ingenuity and luck, and even then can fall prey to root selection errors. In simple cases, such as for an isolated airfoil or an inlet, the near-circle is actually near to being a circle. For other cases (inlet with centerbody far inside or far outside, turbine cascade, small gap/chord compressor cascade), the near-circle may not even resemble a circle. When this happens, the investigator must either find a better mapping to a near-circle (which may not be easy in practice, even after many years of experience), or change to a nonclassical technique such as discussed later in the section on one-step mappings.

SEQUENTIAL MAPPINGS

For a complicated mapping of the form $Z = f(\zeta)$, roots are often more easily selected and the mapping is more easily created, if the process is broken up into an equivalent sequence of simpler mappings. For instance, the von Karman-Trefftz transformation of Eq. (12) can be restated as the sequence

$$\omega = \frac{Z-A}{Z-B} \quad , \tag{13a}$$

$$\eta = \omega^k \quad , \tag{13b}$$

$$\zeta = \frac{a-b\eta}{1-\eta} \quad . \tag{13c}$$

The bilinear transformations, Eqs. (13a) and (13c) require no root selection, while Eq. (13b) requires keeping track of only a single angle. Other schemes may require the separate tracking of the angles related to both the $(Z-A)^k$ and the $(Z-B)^k$ term.

The logarithmic transformation, Eq. (6), and its inverse

$$\zeta = e^Z \tag{14}$$

are the key elements of conformal mappings for a cascade of airfoils, such as occur in turbomachinery. For this case, there are an infinite number of identical airfoils in the Z plane, all of which map onto the same contour in the ζ plane. When mapping from the ζ plane to the Z plane, the correct root is the one that "tracks" a nearby point; the other roots will be different by an integral multiple of 2π in the vertical (or imaginary axis) direction. Most of the cascade mappings which the author has seen, Refs. [19], [20], [21] and [22], can be restated as a simple sequence* with Eq. (14) (possibly combined with a bilinear transformation) as the first element.

As a general principle, it is highly recommended for mappings of the form $Z = f(\zeta)$ that a complicated mapping be restated as a sequence of simpler mappings for actual implementation in a computer program.

NEAR-CIRCLE TO CIRCLE MAPPING

If an orthogonal grid is desired, the mapping of a near-circle to a circle is often chosen as one-step of the mapping process. Although there are now a plethora of ways to accomplish this mapping**, the classical technique is that of using the Theodorsen-Garrick transformation, Ref. [18], given by

---

* The author has not yet been successful in breaking up the Garrick transform, Ref. [23], into a simple form.

** Including panel techniques, Schwartz-Christoffel techniques, and elliptic techniques with orthogonal boundary control.

$$\frac{Z}{\zeta} = e^{\sum_{j=0}^{j=N}(a_j+ib_j)\zeta^j} \tag{15a}$$

or, in its derivative form by

$$\frac{dZ}{d\zeta} = e^{\sum_{j=0}^{j=N}(a_j+ib_j)\zeta^j} \tag{15b}$$

What is frequently not appreciated is that when Eqs. (15a, 15b) are implemented in the required iterative manner (see Ref. [18 or 24] for details), the iteration may not necessarily converge unless the near-circle is sufficiently "near" to being a circle. In particular, there is a requirement on the maximum value* allowed for $\left|\frac{d(\ell nr)}{d\theta}\right|$ for the mapping in Eq. (15a), where r and θ are the polar coordinates describing the near-circle. For the vast majority of airfoils, the classical von Karman-Trefftz transformation produces a near-circle that meets the above requirement. For turbine cascades, on the other hand, the above requirement is usually not met. It is possible to underrelax the iteration, but the required underrelaxation parameter may need to be quite small for Eq. (15a) to ensure convergence, resulting in unrealistic computation time requirements. The form given by Eq. (15b) is much less sensitive, and with an underrelaxation factor of .5 has converged even for a case where r was a multiple-valued function of θ, so that $\left|\frac{d\ell nr}{d\theta}\right|$ was infinite! Obviously, the convergence criterion for Eq. (15b) is substantially different than that derived for Eq. (15a) in Ref. [25]. In short, if one uses the derivative form, the chances for success are much higher. Of course, the derivative form requires the integration of a complex function, but an excellent technique using a spline exists as discussed later. Both of the above forms can employ fast Fourier techniques, as in Refs. [24, 26], to efficiently evaluate the series $\sum(a_j+ib_j)\zeta^j$ at evenly spaced increments on the unit circle in the ζ plane and thus keep the required computing time modest.

ONE-STEP MAPPINGS

A one-step mapping maps a contour onto a canonical shape, such as a circle or the real axis, in a single step. Such transformations are usually given in derivative form, namely $\frac{dZ}{d\zeta} = f(\zeta)$.

---

* For a relaxation parameter of unity, this maximum value is .2955.

The classic Schwarz-Christoffel transformation for a polygon with N straight sides takes the form

$$\frac{dZ}{d\zeta} = \prod_{j=1}^{j=N} (\zeta - b_j)^{k_j} \quad , \tag{16}$$

while for a polygon with curved sides, Davis, Ref. [27], uses a differential form of the product term in Eq. (16) so that

$$\frac{dZ}{d\zeta} = e^{\frac{1}{\pi}\int \ell n(\zeta - b)\, d\beta} \quad , \tag{17}$$

where β is an angle related variable. Davis employs a composite integration formula to resolve the curvature effects. Equation (16), as used by Skulsky, Ref. [28], can be considered a subset of the Davis technique. The Davis technique has been successfully employed in Refs. [12], [29], and [30] to map a wide range of complicated configurations.

An alternative is to use the form

$$\frac{dZ}{d\zeta} = g(\zeta) e^{\sum_{j=0}^{j=N} (a_j + ib_j)\zeta^j} \quad , \tag{18}$$

where $g(\zeta)$ is chosen to resolve angles or general behavior*, while the exponential term accounts for the curvature. This form can take advantage of fast Fourier techniques to evaluate the coefficients a_j and b_j.

One example of such a mapping is given by

$$\frac{dZ}{d\zeta} = \left[1 - \frac{1}{\zeta}\right]^k e^{\sum_{j=0}^{j=N} (a_j + ib_j)\zeta^{-j}} \quad , \tag{19}$$

which is used in Ref. [26] to map an airfoil in the Z plane to a circle in the ζ plane. Another mapping of this type is given by

---

* The $g(\zeta)$ function is easily constructed using the Schwarz-Christoffel technique, as illustrated in Kober, Nehari, or Milne-Thomson, Refs. [31], [32] or [33].

$$\frac{dZ}{d\zeta} = -\frac{(\zeta - iR_0)}{(1-iR_0)(\zeta+1)^2(\zeta-1)} e^{\sum_{j=0}^{j=N}(a_j+ib_j)\zeta^j} \quad , \tag{20}$$

where

$$R_0 = 1 + 2\sqrt{\pi\epsilon} \quad ,$$

and

$$\epsilon = \frac{\text{nose radius}}{\text{inlet interior radius}} \quad .$$

This mapping was developed by the author to map the region exterior to a semi-infinite inlet and above the centerline to the interior of a circle as illustrated in Fig. 4. A similar form, suggested by Jameson, Ref. [34], given by

$$\frac{dZ}{d\zeta} = \frac{(R_0^2+\zeta^2)}{(\zeta+1)^3(1-\zeta)} e^{\sum_{j=0}^{j=N}(a_j+ib_j)\zeta^j} \quad , \tag{21}$$

was used to map an inlet and centerline to a half circle as illustrated in Fig. 5.

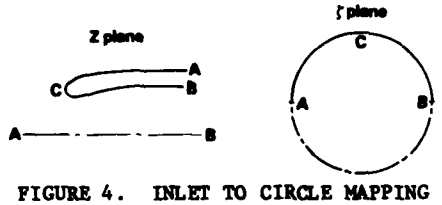


FIGURE 4. INLET TO CIRCLE MAPPING

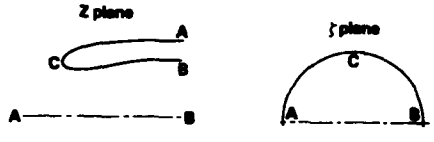


FIGURE 5.
INLET TO HALF CIRCLE MAPPING

The latter mapping did not adequately resolve the exterior inlet contour far downstream (but was good otherwise), while a mapping based on Eq. (20) adequately resolved the inlet in all areas. The reason is that the evenly spaced point distribution (as required by fast Fourier techniques) on the ζ plane circle is too sparse when transformed back to the Z plane to accurately represent the nacelle far downstream for Eq. (21). This example illustrates the need to choose the correct canonical domain. Of course, the two canonical domains are related in a simple quadratic manner through the transformation

$$\frac{Z-1}{Z+1} = i\cdot\left[\frac{\zeta-1}{\zeta+1}\right]^2 \quad , \tag{22}$$

which transforms a unit circle in the Z plane to a unit upper half circle (and to the real axis from -1. to 1.) in the ζ plane as illustrated in Fig. 6.

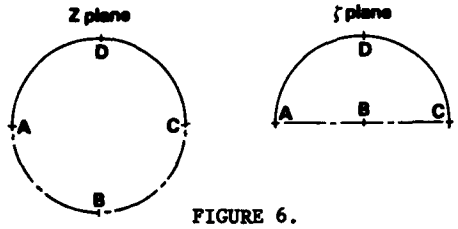


FIGURE 6.
CIRCLE TO HALF CIRCLE MAPPING

The one-step mapping of Eq. (18) is far simpler to program* than the conventional classic technique described earlier, converges stably and rapidly, and is easy to modify later for a new class of geometries. If one wishes to use fast Fourier techniques for conformal mapping, Eq. (18) is recommended as the method of choice rather than using a sequence resulting in a near-circle followed by a derivative transform like Eq. (15b), especially since Eqs. (18) and (15b) differ only by a very few lines of computer code.

It is recommended that one should not expect the Fourier series in Eqs. (15a), (15b), or (18) to resolve a slope discontinuity by the sheer brute force of a large number of Fourier terms. One might "get away" with doing so for small discontinuities in slope (e.g., 5 degrees) where a high accuracy near the discontinuity is not required, but for larger slope discontinuities the accuracy and convergence of the transformation will suffer. It is easy to remove such slope discontinuities using the hinge point transformations covered next or by incorporating slope discontinuities into the $g(\zeta)$ term in Eq. (18) so that the Fourier series only has to resolve a function with a continuous derivative.

HINGE POINT TRANSFORMATIONS

At the other extreme from one-step mappings, the hinge point transformations due to Moretti and Hall, Refs. [17] and [35], break the problem up into a large number of sequential applications of a single mapping. For problems with a plane of symmetry, Moretti applied a von Karman-Trefftz transformation to each pair of slope discontinuities in turn, to produce a smooth near-circle. Hall

---

* Note that the computer program in Ref. [26] can be easily modified to implement a mapping using Eq. (18), instead of using Eq. (19) for the airfoil.

repeatedly applied the transformation $Z = (\zeta-\zeta_0)^k$ (Eq. (10a)), where ζ_0 and k have been chosen to remove the left-most remaining slope discontinuity at each stage, to produce a smooth near half plane. Once a near-circle or near half plane is obtained, a non-conformal shearing or a conformal mapping can be used to produce a canonical domain.

MULTIPLE BODIES

Techniques to map multiple bodies can be classified as simultaneous, sequential, iterative, and periodic. This section discusses these techniques.

A simultaneous multiple body mapping maps two or more bodies simultaneously to near-canonical or canonical domains. The single mapping

$$\prod_{j=1}^{j=N}\left[\frac{\zeta-\zeta_{Tj}}{\zeta-\zeta_{Nj}}\right] = \prod_{\ell=1}^{\ell=N}\left[\frac{Z-Z_{T\ell}}{Z-Z_{N\ell}}\right]^{k_\ell} , \tag{23}$$

taken from Ref. [24] simultaneously maps N airfoils to N near-circles. The Garrick transformation in Ref. [36] simultaneously maps two concentric near-circles to two concentric circles.

A sequential multiple body mapping maps two contours to canonical contours by first mapping one contour to a canonical contour, then mapping the second contour to a canonical contour while preserving the nature of the first canonical contour. Two examples of this are given in Ref. [24], where transformations are given to map an airfoil to a near-circle (or to map a near-circle to a circle) while keeping a nearby circle a circle. Another example is illustrated in Fig. 7, where a centerbody is mapped onto the real axis using a technique similar to that in Ref. [37], and then a nearby inlet is mapped to a circle, using the Eq. (20), while keeping the real axis a straight line.

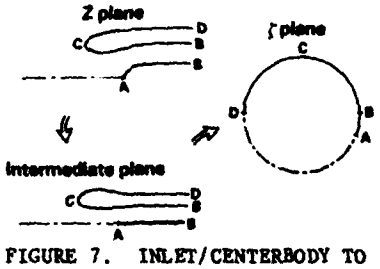


FIGURE 7. INLET/CENTERBODY TO CIRCLE MAPPING SEQUENCE

After doing a number of problems both simultaneously and sequentially, the author has found that a sequential multiple body technique usually works better in practice than a simultaneous technique since the branch cuts required to resolve the root selection problem are simpler. This more than balances the increase in computer time* required for a sequential approach.

An iterative technique to map multiple contours to canonical contours has recently been developed by Halsey, Ref. [38]. In this technique each contour is individually mapped to a circle in sequence, with no special constraints on holding the other contour shapes. By repeatedly cycling through all the contours one at a time, the contours are all mapped to circles to engineering accuracy. That this process converges is not particularly surprising; what is surprising is that only about five cycles through all the contours are required to achieve four digit accuracy for practical cases.

A periodic mapping maps a finite or infinite number of identical (but otherwise displaced or rotated) contours onto overlayed (but otherwise identical) contours on different Riemann sheets. As an example, the logarithmic transformation of Eq. (6), when appropriately scaled, maps a cascade of airfoils in the ω plane onto a single (highly distorted) airfoil-like contour in the ζ plane. The transformation $Z = \zeta^N$ (which is a subset of Eq. (10a)), where N is an integer, maps a region with N angular periodic boundaries in the Z plane onto a region with one periodic boundary in the ζ plane. By these means, a periodic configuration can be reduced to a form wherein the periodicity condition reduces essentially to a continuity condition (except for circulation-type terms associated with branch cuts).

CREATING NEW MAPPINGS

This section includes a discussion of a few general techniques that can be used to create new conformal mappings, and some restrictions to keep in mind while doing so.

The first step when considering the development of a new mapping is to verify that the desired mapping has not already been created. The book by Kober, Ref. [31], includes an extensive survey of mappings developed up to the mid 1940's, and should be reviewed first. The reference material in Table 1, included later in this paper, can be consulted by topic. If none of the above include the desired mapping, a careful literature search may be warranted. Only when the investigator is reasonably sure of not "reinventing the wheel" should a new mapping be attempted.

* A 2-body sequential approach may typically require twice the computer time of a simultaneous approach.

If the complex potential ω for the two-dimensional incompressible flow over (or through) a body is known, i.e., if

$$\omega = \phi + i\psi = f(Z)$$

is known, then the mapping from the Z plane to the ω plane (with coordinates (ϕ,ψ)) is known. Thus, knowing the incompressible flow is equivalent to knowing the mapping from the Z plane to a rectangle in the ω plane, and vice versa. This principle can be used to construct conformal mappings. In particular, a "panel method" potential flow program can be used to construct a conformal mapping, Ref. [39].

Another technique for the creation of new mappings can be summarized by the description "guess and plot." Using this technique a function having appropriate zeroes, poles, and singularities is guessed*, and its effect is determined by plotting a transformed contour or a transformed grid. About one third of the functions this author has guessed over the years have had the appropriate action, indicating that this technique is more viable than might be thought at first glance.

Since a conformal mapping is simply a functional relationship, if one knows

$$Z = f(\omega) \quad ,$$

and

$$\zeta = g(\omega) \quad ,$$

then one has

$$Z = f(g^{-1}(\zeta)) \quad .$$

In short, if one knows how to map two different contours to the same contour, then one can map one of these contours onto the other, Refs. [31] and [40]. Put another way, a conformal mapping of a conformal mapping is a conformal mapping. One then begins to think in terms of building up a sequence of mappings, with each step bringing a contour closer to a canonical contour such as a unit circle. Thus by combining known mappings in a new sequence, one can construct new mappings.

In generating an orthogonal grid with conformal mapping, there are two processes involved. The first process is to conformally map the desired contour (i.e: an airfoil, or inlet, etc. ...) to a rectangle (or equivalently to a circle or half plane). The computing time required for this step is often

---

* These singularities, poles, and zeroes are placed at points on the contour where the slope is discontinuous, and when necessary (e.g., near regions of high convex curvature of the contour), inside the contour along a line joining the local contour point of maximum curvature and its center of curvature. How far along this line depends on the total slope change nearby, and the singularity powers also depend on these slope changes.

nearly linearly* proportional to L, the number of input points. The second process is classically to map an orthogonal rectangular grid in the mapped plane back to the physical plane. This involves a computing time proportional to MxN, the total number of grid points. It is common for MxN to be substantially larger than L. This implies that the forward transformations can be somewhat inefficient (i.e., implicit) but the inverse transformations should be efficient (i.e., explicit), if possible. This should be kept in mind when creating mappings that are to be used to transform a grid back to the physical plane.

When creating new mappings, one principle to be followed is "do not create singularities** within the flow field"; however, singularities are often necessary and completely acceptable within a body, on a bounding surface, or at images of "infinity." An example of creating a singularity within the flow field is shown in Fig. 8, taken from Ref. [19]. Figure 9 from this same reference shows a grid system with the same number of grid points which is better suited for flow computations. The above principle was used in Ref. [24] to determine some mapping parameters that were otherwise undetermined. With another choice of parameters, the mappings therein looked something like those in Fig. 10 rather than like those in Fig. 11 where the parameters were chosen to obey this principle.

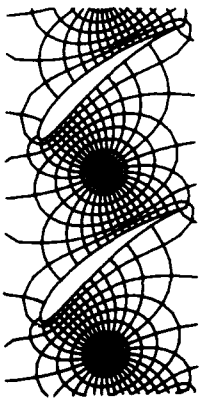


FIGURE 8. GRID SYSTEM WITH SINGULARITY IN FLOW FIELD

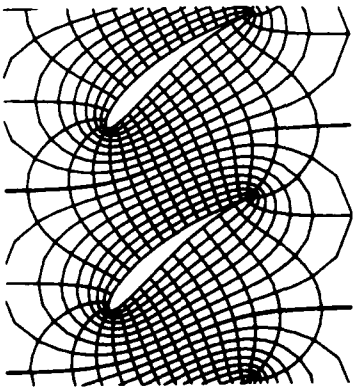


FIGURE 9. GRID SYSTEM WITHOUT SINGULARITY IN FLOW FIELD

* Actually the operation count goes as $Lx\ell n_2 L$, as dictated by using fast Fourier techniques for Eqs. (15a), (15b), or (18).

** A singularity occurs when $\frac{dZ}{d\zeta} = 0$ or infinity.

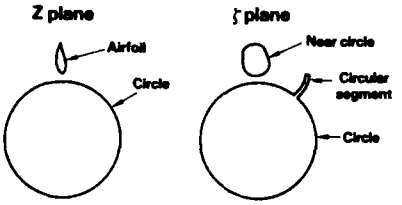


FIGURE 10. INCORRECT CHOICE OF MAPPING CONSTANTS

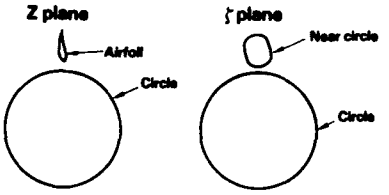


FIGURE 11. CORRECT CHOICE OF MAPPING CONSTANTS

REFLECTION PRINCIPLE

It is possible to map a contour to a desired shape while simultaneously keeping a nearby straight line straight by using a reflection principle. If the relation

$$f(Z) = g(\zeta)$$

conformally maps a contour from the Z plane to the ζ plane in a desirable manner, then the relations

$$f(Z)\cdot\bar{f}(Z) = g(\zeta)\cdot\bar{g}(\zeta) \quad , \tag{24a}$$

and

$$\frac{f(Z)}{\bar{f}(Z)} = \frac{g(\zeta)}{\bar{g}(\zeta)} \quad , \tag{24b}$$

where the superscript "-" denotes the complex conjugate operator, will both have a "similar"* effect but will preserve the shape of the real axis. A proof of the real axis shape preservation is outlined in Appendix A. Examples involving mappings symmetric with respect to a line appear in Refs. [16] and [17]. An extension of the above concept allows the construction of mappings which preserve a circle by utilizing operators involving image points with respect to circles rather than the complex conjugate operator. Examples of circle preserving mappings are contained in Refs. [24] and [36]. The question of whether to use the product mapping, Eq. (24a), or the ratio mapping, Eq. (24b), can be resolved by simply programming both and then choosing the one which produces the most desirable effect.

---

* "Similar" is taken here to mean not identical, but generally of the same nature.

CALCULATING THE GRID

Recently, Sockol and Adamczyk in Refs. [39] and [41] have observed that the generation of an orthogonal grid can be broken down into two independent steps, namely;

1) calculation of the boundary point correspondence between the physical and computational planes by means of conformal mapping or some other orthogonality preserving technique, and
2) the generation of a grid given this boundary point correspondence.

This observation is to conformal grid generation as the finite volume (or finite element) technique is to flow calculations; it breaks the grid generation problem up into two nearby independent steps just as the finite volume technique breaks a flow simulation up into two nearby independent steps (namely the generation of a grid and the solution of the flow equations on the grid). This allows a great deal of flexibility to be attained. In short, what is the most efficient, simple, or general technique required to complete step #1 (which is basically a boundary operation) is not necessarily the same as the most efficient, simple, or general technique required to complete step #2 (which is basically an area operation). This represents a major step forward in orthogonal grid generation. This section reviews some methods used to complete step #2.

One obvious way to generate an orthogonal grid is to invert the conformal mapping used in step #1, and then map an orthogonal grid constructed in the computational plane back to the physical plane, as mentioned earlier. If one has used an algebraic stretching in the direction tangential to the body to create the computational grid, fast Fourier techniques cannot be easily employed in the mapping inversion. Depending on the mapping complexity, the number of Fourier terms, and the presence of implicit expressions in the inverse mapping, the creation of a grid at NxM points can require the order of CxLxNxM operations, where L is the number of Fourier terms and C is a relatively large constant. In practical terms, this can require about six seconds of time on an IBM 3081 computer for a typical* case. For a non-stretched isolated airfoil mapping, where fast Fourier techniques can be used in the inverse mapping, a typical grid can be generated in about two seconds of IBM 3081 computer time, once the conformal mapping is known.

---

* For the purposes of this paper "typical" means a 128x32 grid using 256 Fourier terms for an inlet/centerbody configuration, unless otherwise specified. The IBM 3081 computing time quoted can be taken as roughly equivalent to an equal amount of CDC 7600 computer time.

Another way to generate an orthogonal grid is to solve a Laplace problem (with Dirichlet boundary conditions) on the computational grid for both x and y (the physical plane coordinates), which are known on the boundary as described in Refs. [39] and [41]. Since the Laplacian operator will retain orthogonality, the (x,y) values thus calculated will describe an orthogonal grid in the physical plane. A fast Poisson solver, Ref. [42], with an operation count of $CxNxMx\ell n_2N$, where C is a relatively small constant, may be used to solve the Laplace problem. For a typical grid, this will require about two seconds on an IBM 3081 computer. This solver is sufficiently general even for use when independent stretchings are used following a conformal mapping to generate the rectangular grid in the computational plane.

THREE DIMENSIONS

Conformal mapping is basically a surface technique, but it can be used as one component of a three-dimensional grid generation system. One example is shown in Fig. 12, taken from Ref. [16], where a three-dimensional inlet/centerbody grid was constructed using conformal mapping in each circumferential plane, or slice, followed by independent algebraic stretchings to construct an orthogonal grid in the sliced plane, one of which is shown. The resultant three-dimensional grid is orthogonal in two directions, but not in the third. Similar techniques are used to construct near-orthogonal grids in Refs. [13], [43], and [44].

REFERENCE MATERIAL

The following table summarizes a limited amount of reference material for conformal mapping of aerodynamic configurations. The book by Kober, Ref. [31], is particularly useful when attempting to map a new configuration. The paper by Moretti, Ref. [45], is highly recommended as an alternate review of the field.

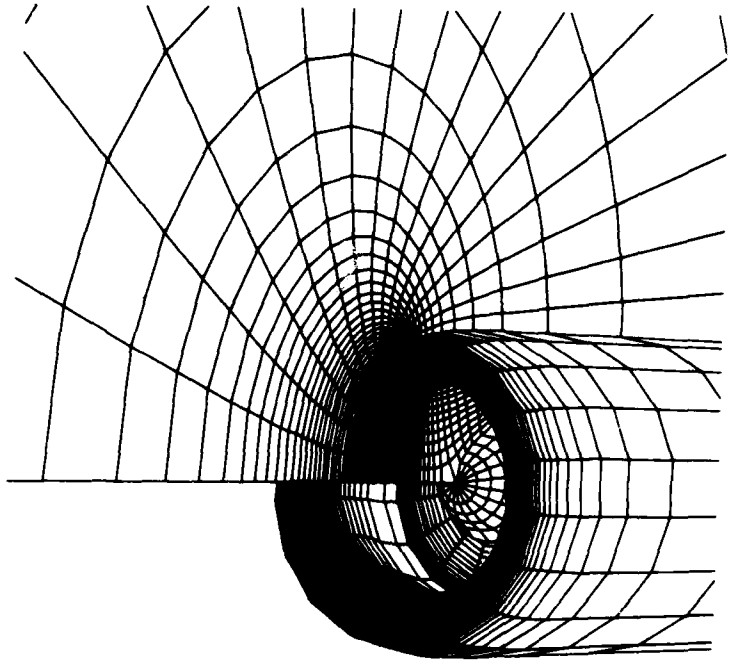


FIGURE 12. PERSPECTIVE VIEW OF
THREE-DIMENSIONAL COORDINATE SYSTEM

TABLE 1 - A LIMITED SET OF CONFORMAL MAPPING REFERENCES.

Subject	References
General - Papers	27, 28, 40, 45, 46, 47, 48
General - Books	11, 31, 32, 33
Airfoil	18, 24, 26
Airfoil with Spoiler	49
Two Piece Airfoil	24, 36, 38
Multiple Bodies	24, 38
Inlet	16, 50, 51
Inlet with Centerbody	16
Axisymmetric Body	37
Cascade	19, 20, 21, 22, 23, 39, 41
Non-Planar	4, 5, 6, 7, 8, 9, 10, 12, 13, 14
Symmetric	16, 17
Ducts	27, 29, 30

PACKAGED TOOLS

To accomplish simple conformal transformations, one does not need much in the way of supporting tools. For more complex transformations, a number of packaged tools can substantially simplify program development. Among these tools are spline fits, fast Fourier transforms, efficient matrix techniques, fast Poisson solvers, near-circle to circle transformation packages, and good graphics. The need for, use, and availability of these packages is covered in this section.

Mapping procedures often require interpolation between points. A spline fit is a good way to accomplish such an interpolation, Ref. [52]. In addition, it may be necessary to numerically integrate a function. Such an integration can be accomplished by analytic integration of a spline passed through the function values. A spline routine accomplishing both of these objectives is listed in a FORTRAN program form on pages 279-281 of Ref. [26]. By simply declaring all floating point variables in this routine as COMPLEX, this program is suitable for integrating a complex function. The derivative form of transformations of Eqs. (15b), (16), (17), and (18) require such an integration.

A fast Fourier transform, such as given in Ref. [26], is a key element of an efficient implementation of transformations using Eqs. (15a), (15b), or (18). The program listing contained within pages 202 to 240 of Ref. [26] uses fast Fourier techniques, and makes a good starting point.

The LINPACK package available from the Society for Industrial and Applied Mathematics (SIAM), Ref. [53], contains linear equation solvers that can be useful in mapping operations. If one uses a near-circle to circle transform at unevenly spaced points in the circle plane (namely at points corresponding to the input points in the physical plane), fast Fourier techniques cannot be used. By using LINPACK routines on a fast computer (i.e: IBM 3081) moderate computation times (about one minute of CPU time) are required for a representation involving one or two hundred points. A typical FFT mapping time for the same number of points is two seconds.

Fast Poisson solvers are available for filling-in the grid, as mentioned previously. A rather versatile fast Poisson solver is reported in Ref. [42].

Good graphics routines for plotting intermediate contours and grids have, in practice, been found to be the most important element of the process of inventing new mappings, or new combinations of mappings. By using extensive graphics, a defective mapping is soon discovered with little effort. This allows a wider range of functions to be tried (guessed) in the given amount of time allowed for completion of a project. The graphics package must be general

enough to plot both contours and grids and should be accessible by a simple subroutine call. It is not unusual to plot fifty different intermediate results during the development of a mapping sequence such as that in Ref. [16]. Of course, each plotting call is sequentially commented out, but not deleted, during the development process. Such an extensive use of graphics greatly simplifies debugging, but would be unbearably tedious without a simple calling sequence. Since graphics tends to be tailored to a particular computer installation, a sufficiently versatile and easy to use package often must be written locally and thus is not usually available to the public.

CLOSING REMARKS

Modern finite volume flow solvers do not require an orthogonal grid, but a near-orthogonal grid is usually beneficial. Often a simple conformal transformation, followed by independent algebraic transformations, can be used to generate such a near-orthogonal grid with little difficulty to be expected in a computer program implementation. This is a good way to get involved with conformal mapping.

The conformal mapping of a contour onto a canonical contour is far easier to accomplish using a one-step technique based on Eq. (18), as opposed to classically mapping a contour onto a near-circle and then mapping the near-circle onto a circle. This is especially true since Ref. [26] contains a one-step conformal mapping computer program which is easily modified to map new geometries. This one-step mapping technique is faster, simpler, more resistant to root selection problems, and more stable than the classical technique.

It is not necessary to generate the grid using the inverse of the transform employed to map the contour to a canonical contour. In fact, it would seem that use of a fast Poisson solver to generate the grid using the known boundary correspondence offers a flexibility, simplicity, and economy that may not be surpassed by other methods.

REFERENCES

1. Thompson, J.F. and Mastin, C.W. (1980) "Grid Generation Using Differential Systems Techniques," NASA Conference Publication No. 2166 on Numerical Grid Generation Techniques, pp. 37-72, Hampton, Va.
2. McFadden, G.B. (1979) "An Artificial Viscosity Method for the Design of Supercritical Airfoils," Report COO-3077-158, Courant Mathematics and Computing Laboratory, New York University, N.Y., N.Y.
3. Volpe, G. and Melnik, R.E. (1981) "The Role of Constraints in the Inverse Design Problem for Transonic Airfoils," Paper No. 81-1233, AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto, California.
4. Young, L. (1958) "Runners of Experimental Turbomachines," Engineering, pp. 376-378.

REFERENCES (continued)

5. Prasil, F. (1926) "Techniche Hydrodynamik," Appendix 4, J. Springer, Berlin.
6. Senoo, Y. and Yoshiyuki, N. (1971) "A Blade Theory of an Impeller with an Arbitrary Surface of Revolution," Trans. ASME Journal of Engineering for Power, pp. 454-460.
7. Wislicenus, G. (1947) "Fluid Mechanics of Turbomachinery," McGraw-Hill, New York.
8. Lewis, R.A. (1964-1965) "Internal Aerodynamics of Turbo-Machines," Proc. Instn. Mech. Engrs., Vol. 179, Pt. 1, pp. 1115-1128.
9. Weatherburn, C.E. (1955) "Differential Geometry of Three-Dimensions," Volume 1, Cambridge at the University Press.
10. Kumar, T.C.M. and Rao, Y.V.N. (1977) "Theoretical Investigation of Pressure Distributions Along the Surfaces of a Thin Blade of Arbitrary Geometry of a Two-Dimensional Centrifugal Pump Impeller," ASME Journal of Fluids Engineering, pp. 531-542.
11. Polya, G. and Latta, G. (1974) "Complex Variables," John Wiley & Sons, New York, N.Y.
12. Grossman, B. (1979) "Numerical Procedure for the Computation of Irrotational Conical Flows," AIAA Journal, Vol. 17, No. 8, Article No. 78-1213R, pp. 828-837.
13. Grossman, B. and Siclari, M.J. (1980) "The Non-Linear Supersonic Potential Flow Over Delta Wings," Paper No. 80-0269, AIAA 18th Aerospace Sciences Meeting, Pasadena, California.
14. Murata, S., Miyake, Y. and Kawabata, N. (1980) "Exact Two-Dimensional Theory of Spherical Spiral Groove Bearings," ASME Journal of Lubrication Technology, Vol. 102, pp. 430-438.
15. von Karman, T. and Trefftz, E. (1918) "Potential-stromung um gegebene Tragflachenquerschnitte," Zeitschrift fur Flugtechnische Moforluftsch, Vol. 9, pp. 111-116.
16. Ives, D.C. and Menor, W.A. (1981) "Grid Generation for Inlet and Inlet-Centerbody Configurations Using Conformal Mapping and Stretching," Paper No. 81-0997, AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, Calif.
17. Moretti, G. (1976) "Conformal Mappings for Computations of Steady, Three-Dimensional, Supersonic Flows," Numerical/Laboratory Computer Methods in Fluid Mechanics, ASME.
18. Theodorsen, T. and Garrick, I.E. (1933) "General Potential Theory of Arbitrary Wing Sections," NACA TR 452.
19. Ives, D.C. and Liutermoza, J.F. (1977) "Analysis of Transonic Cascade Flow Using Conformal Mapping and Relaxation Techniques," AIAA Journal, Vol. 15, pp. 647-652.
20. Howell, A.R. (1948) "A Theory of Arbitrary Airfoils in Cascade," The Philosophical Magazine, Vol. 39, pp. 913-927.
21. Legendre, R. (1972) "Work in Progress in France Related to Computation of Profiles for Turbomachine Blades by Hodograph Method," American Society of Mechanical Engineers, Paper 72-GT-41.
22. Frith, D.A. (1973) "Inviscid Flow Through a Cascade of Thick, Cambered Airfoils, Part 1 - Incompressible Flow," American Society of Mechanical Engineers, Paper 73-GT-84.
23. Garrick, I.E. (1944) "On the Plane Potential Flow Past a Lattice of Arbitrary Airfoils," NACA Rept. 688.
24. Ives, D.C. (1976) "A Modern Look at Conformal Mapping, Including Multiply Connected Regions," AIAA Journal, Vol. 14, pp. 1006-1011.
25. Warschawski, S.E. (1945) "On Theodorsen's Method of Conformal Mapping of Nearly Circular Regions," Quarterly of Applied Mathematics, Vol. 3, pp. 12-28.

REFERENCES (continued)

26. Bauer, F., Garabedian, P., Korn, D. and Jameson, A. (1975) "Supercritical Wing Sections II," Vol. 108, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, New York.
27. Davis, R.T. (1979) "Numerical Methods for Coordinate Generation Based on Schwarz-Christoffel Transformation," Paper No. 79-1463, AIAA 4th Computational Fluid Dynamics Conference, Williamsburg, Va.
28. Skulsky, R.S. (1966) "A Conformal Mapping Method to Predict Low-Speed Aerodynamic Characteristics of Arbitrary Slender Re-entry Shapes," AIAA Journal of Spacecraft, Vol. 3, pp. 247-253.
29. Sridhar, K.P. and Davis, R.T. (1981) "A Schwarz-Christoffel Method for Generating Internal Flow Grids," Symposium on Computers in Flow Prediction and Fluid Dynamic Experiments, pp. 35-44, ASME Winter Annual Meeting, Washington, D.C.
30. Anderson, O.L., et al (1982) "Solution of Viscous Internal Flows on Curvilinear Grids Generated by Schwarz-Christoffel Transformation," Symposium on the Numerical Generation of Curvilinear Coordinate Systems and use in the Numerical Solution of Partial Differential Equations, Nashville, Tenn.
31. Kober, H. (1957) "Dictionary of Conformal Representations," Dover Publications, Inc., New York.
32. Nehari, Z. (1952) "Conformal Mapping," Dover Publications, Inc., New York.
33. Milne-Thomson, L.E. (1960) "Theoretical Hydrodynamics," The MacMillan Company, New York, N.Y.
34. Jameson, A. (1979) Private Communication.
35. Hall, D.W. (1980) "A Three-Dimensional Body-Fitted Coordinate System for Flow Field Calculations on Asymmetric Nosetips," NASA Conference Publication No. 2166 on Numerical Grid Generation Techniques, pp. 315-328, Hampton, Va.
36. Garrick, I.E. (1936) "Potential Flow About Arbitrary Biplane Wing Sections," Rept. 542, NACA.
37. Arlinger, B. (1980) "Axisymmetric Transonic Flow Computations Using a Multigrid Method," Proceedings of Seventh International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, Vol. 141, pp. 55-60, Springer-Verlag, New York, N.Y.
38. Halsey, N.D. (1979) "Potential Flow Analysis of Multielement Airfoils Using Conformal Mapping," AIAA Journal, Vol. 17, No. 12, pp. 1281-1288.
39. Adamczyk, J.J. (1980) "2D Grid Code for Rotor and Stator Blade Sections Using Electrostatic Analogy - O-Type and C-Type," NASA Conference Publication No. 2165 on Numerical Grid Generation Techniques, pp. 129-142, Hampton, Va.
40. Caughey, D. (1978) "A Systematic Procedure for Generating Useful Conformal Mappings, Int. J. Numerical Methods in Eng., 12, p. 1651.
41. Sockol, P.M. (1980) "2D Grid Code for Rotor and Stator Blade Sections - C-Type," NASA Conference Publication No. 2166 on Numerical Grid Generation Techniques, pp. 437-448, Hampton, Va.
42. Swarztrauber, P.N. and Sweet, R. (1975) "Efficient Fortran Subprograms for the Solution of Elliptic Partial Differential Equations," Tech. Note NCAR-TN/IA-109, National Center for Atmospheric Research, Boulder, Colorado.
43. Jameson, A. (1974) "Iterative Solution of Transonic Flows Over Airfoils and Wings, Including Flows at Mach 1," Communications on Pure and Applied Mathematics, Vol. XXVII, pp. 283-309.
44. Jameson, A. and Caughey, D.A. (1977) "A Finite Volume Method for Transonic Potential Flow Calculations," Paper No. 77-635, AIAA 3rd Computational Fluid Dynamics Conference.

REFERENCES (continued)

45. Moretti, G. (1980) "Grid Generation Using Classical Techniques," NASA Conference Publication No. 2166 on Numerical Grid Generation Techniques, pp. 1-35, Hampton, Va.
46. Laura, P.A.A., (1975) "A Survey of Modern Applications of the Method of Conformal Mapping," Revista de la Union Mathematica Argentina, Vol. 27, pp. 167-179.
47. Garrick, I.E. (1949) "Conformal Mapping in Aerodynamics, with Emphasis on the Method of Successive Conjugates," Symposium on Construction and Applications of Conformal Maps, National Bureau of Standards, Applied Mathematics Series, Vol. 18, pp. 137-147.
48. Smith, R.E. (1980) ed., Numerical Grid Generation Techniques, NASA Conference Publication No. 2166, Hampton, Va.
49. Rossow, V.J. (1973) "Conformal Mapping for Potential Flow About Airfoils with Attached Flap," AIAA Journal of Aircraft, Vol. 10, pp. 60-62.
50. Arlinger, B.G. (1975) "Calculation of Transonic Flow Around Axisymmetric Inlets," Paper No. 75-80, AIAA 13th Aerospace Sciences Meeting, Pasadena, California.
51. Caughey, D.A. and Jameson, A. (1976) "Accelerated Iterative Calculation of Transonic Nacelle Flowfields," AIAA Paper 76-100, Washington, D.C.
52. Ahlberg, J.H., Nilson, E.N. and Walsh, J.L. (1967) The Theory of Splines and Their Applications, Academic Press, New York, pp. 9-16.
53. Dongarra, J.J. (1979) "LINPACK User's Guide," Society of Industrial and Applied Mathematics, Philadelphia, Pa.

APPENDIX A

PRODUCT MAPPING

Consider the product mapping

$$f(Z)\cdot\bar{f}(Z) = g(\zeta)\cdot\bar{g}(\zeta) = \omega = \phi + i\psi \quad .$$

The real axis in the Z plane can be specified by the relation $Z = \bar{Z}$, so on the real axis

$$\begin{aligned} f(Z)\cdot\bar{f}(Z) &= f(Z)\cdot\bar{f}(\bar{Z}) \\ &= f(Z)\cdot\overline{f(Z)} \\ &= \text{pure real} = \phi + i\psi \quad . \end{aligned}$$

Therefore, the real axis $Z = \bar{Z}$ maps to the real axis $\psi = 0$ in the ω-plane. In a similar manner, the real axis $\zeta = \bar{\zeta}$ also maps to the real axis $\psi = 0$ in the ω plane. Thus the real axis $Z = \bar{Z}$ maps to the real axis $\zeta = \bar{\zeta}$.

RATIO MAPPING

Consider the ratio mapping

$$\frac{f(Z)}{\bar{f}(Z)} = \frac{g(\zeta)}{\bar{g}(Z)} = e^{\omega} = e^{\phi+i\psi} \quad .$$

APPENDIX A (continued)

The real axis in the Z plane is specified by $Z = \bar{Z}$, so on the real axis,

$$\ell n \frac{f(Z)}{\bar{f}(Z)} = \ell n \frac{f(Z)}{\bar{f}(\bar{Z})}$$

$$= \ell n \frac{f(Z)}{\overline{f(Z)}}$$

$$= \text{pure imaginary} = \phi + i\psi \quad .$$

Therefore, the real axis $Z = \bar{Z}$ maps to the imaginary axis $\phi = 0$ in the ω plane. In a similar manner, the real axis $\zeta = \bar{\zeta}$ also maps to the imaginary axis $\phi = 0$ in the ω plane. Thus the real axis $Z = \bar{Z}$ maps to the real axis $\zeta = \bar{\zeta}$.

AD P000971

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ALGEBRAIC GRID GENERATION

ROBERT E. SMITH
NASA Langley Research Center
Hampton, VA 23665

ABSTRACT

Three methods are described for transforming grids in bounded two- and three-dimensional physical domains into a uniform grid in a rectangular computational domain. The methods are based on mathematical interpolation functions and do not require the solution of differential equations or the use of complex variables. They are simply referred to as algebraic methods and are called transfinite interpolation, the multisurface method, and the two-boundary technique. The primary advantage of the methods is that they provide explicit control of physical grid shape and physical grid spacing. Secondly, they require relatively few computations. Consequently, the application of interactive computer graphics in conjunction with the methods is advocated for rapid generation of grids. The basic mathematical structure of each method is described, and particular attention is given to surface representation, parameter variable generation, and control function generation. Physical boundary topology and grid derivative requirements are presented.

NOMENCLATURE

$\vec{a},\vec{b},\vec{c}$	vector valued representation of surface points
$\vec{A},\vec{B},\vec{C}$	vector valued representation of surface derivatives
e	function relating normalized arc length to the computational variable spanning between surfaces
E	magnitude of vectors tangent to a spanning function
f,g,h	control functions
$\vec{F}$	vector valued representation of the physical domain
$\vec{F}_1,\vec{F}_2$	intermediate vector valued representations of the physical domain
G	integral of interpolants
J,J^{-1}	Jacobian matrix, inverse Jacobian matrix
$\hat{K}$	control parameter in a grid concentration function
L,N,M	number of defining points in the I, J, and K directions in the physical domain
r,s,t	normalized arc lengths
S	a set defining points on a surface

PREVIOUS PAGE IS BLANK

S_C	a set defining points in the computational domain
S_P	a set defining points in the physical domain
$\vec{S}$	vector valued representation of a surface
T	orthogonality magnitude coefficient
$\vec{V}$	vector tangent to a piecewise linear curve
u,v,w	dependent variables from the physical domain
x,y,z	physical coordinates
X,Y,Z	physical coordinate functions
α,β,γ	blending functions
δ	kronecker delta function
λ	blending function for linear interpolation
μ	blending function for cubic interpolation
ξ,η,ζ	computational coordinates
$\Delta\xi,\Delta\eta,\Delta\zeta$	increments for a uniform computational grid

Superscripts

n,m	nth, mth partial derivative

Subscripts

I,J,K	index for known points in the physical domain
k	index for surfaces
ℓ	index

INTRODUCTION

The numerical solution of partial differential equations about irregular geometries and with varying characteristic scales has created the need for coordinate systems and associated transformations that reflect both geometric and physical requirements. Coordinate transformations can complicate the basic equations of motion,[1,2] but they can simplify the application of boundary conditions and refine solution accuracy in critical regions. The process of finding coordinate transformations in discrete representations is called "grid generation," and in this paper three algebraic grid generation methods are examined. The methods are transfinite interpolation, the multisurface method, and the two-boundary technique. Interpolation formulas in terms of homotopic mappings[3] and constraints in terms of point positions and/or derivatives are the essential elements of the techniques.

Transfinite interpolation[4] described by Gordon and Hall in the early 1970's is a highly generalized algebraic grid generation method. It is an outgrowth of methods of surface definition[5] pioneered by Steven Coons. Transfinite interpolation is applied through a series of univariate interpolations where

blending functions and the associated parameters (point position and/or derivatives) determine a grid. Eriksson[6] and Rizzi and Eriksson[7] have adapted the original transfinite interpolation formulation to use only exterior boundary descriptions and derivatives of certain boundaries. They have also incorporated exponentials into the blending functions to concentrate the grid near an exterior boundary. The developers of the GIM code[8,9] use transfinite interpolation for grid generation. They define boundaries in terms of algebraic geometric formulas (linear segments, circular arc, conic, etc.) and use linear blending functions for the interior grid computation.

The multisurface method[10,11] developed by Peter Eiseman provides formulas for grid definition based on grid descriptions of two boundary surfaces and an arbitrary number of intermediate control surfaces. Choosing interpolants (defined similar to blending functions) and the placement of the control surfaces determines grid shape and spacing. The multisurface method has been used by Eiseman in numerous applications[12,13] but most notably for computing grids about turbine cascades.[12]

The two-boundary technique[1,2,14] described by this author is based on the description of two exterior boundaries and the application of either linear or hermite cubic polynomial interpolation to compute the interior grid. For cubic interpolation, surface derivatives combined with magnitude coefficients control the orthogonality of the grid at and near the boundaries. Kowalski,[15] applying the two-boundary technique, extended the derivative magnitude coefficients to a functional form for variable orthogonality control.

Four additional topics are discussed. They are surface parameterization, grid spacing control, grid topology, and grid computation. These topics compliment the basic mathematical structure of the algebraic grid generation methods and should be considered in their application. An introduction to boundary-fitted coordinate systems, which sets the stage for grid generation, precedes the description of the algebraic methods.

BOUNDARY-FITTED COORDINATE TRANSFORMATIONS

The motivation for discrete coordinate transformations or grid generation is the numerical solution of partial differential equations. Numerical solutions are obtained by either finite difference or finite element techniques, however, the emphasis here is directed at finite difference methods. Normally, equations of motion are derived relative to a Cartesian coordinate system. When boundary conditions must be applied on irregular subdomains of the Cartesian coordinate system, and when there are regions within the subdomain

that have rapidly varying solutions, it is desirable to transform the equations to a more appropriate coordinate system. An ideal coordinate system for obtaining numerical solutions is a boundary-fitted coordinate system where the physical boundaries of an irregular subdomain transform into exterior boundaries of a rectangular region, and where regions of rapid change are amplified. If the bounded subdomain of the Cartesian coordinate system is called the physical domain, then the rectangular coordinate system where a solution is obtained is called the computational domain (Fig. 1). A transformation between the two

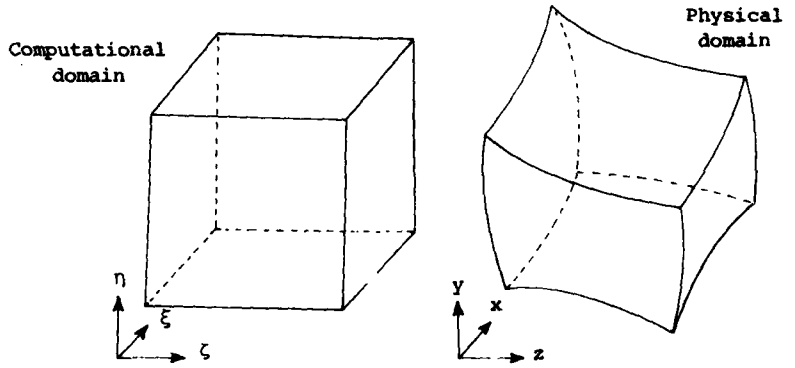


Fig. 1. Computational domain--physical domain

domains is a unique single valued functional relation. This is represented symbolically by letting x, y, and z be coordinates in the physical domain and ξ, η, and ζ be coordinates in the computational domain, then

$$\xi = \xi(x,y,z), \quad \eta = \eta(x,y,z), \quad \zeta = \zeta(x,y,z),$$

and conversely

$$x = x(\xi,\eta,\zeta), \quad y = y(\xi,\eta,\zeta), \quad z = z(\xi,\eta,\zeta).$$

The bounds of the computational domain are defined by

$$0 \leq \xi \leq 1,$$

$$0 \leq \eta \leq 1,$$

$$0 \leq \zeta \leq 1.$$

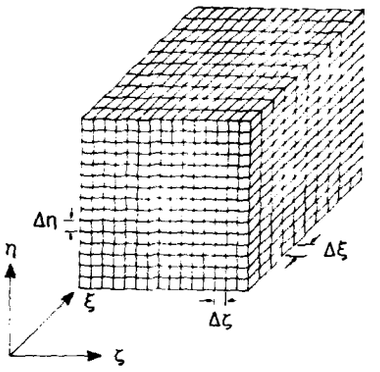


Fig. 2. Computational grid

A uniform grid (Fig. 2) is superimposed onto the computational domain by letting

$$\Delta\xi = \text{constant}_1,$$

$$\Delta\eta = \text{constant}_2,$$

$$\Delta\zeta = \text{constant}_3.$$

Given the functional relations

$$x = x(\xi,\eta,\zeta), \quad y = y(\xi,\eta,\zeta), \text{ and } \quad z = z(\xi,\eta,\zeta)$$

the uniform grid in the computational domain is transformed to a corresponding grid in the physical domain.

In order to transform the equations of motion, partial derivatives with respect to the independent variables x, y, and z must be transformed to partial derivatives with respect to the variables ξ, η, and ζ. For example, if u, v, and w are velocities in the x, y, and z directions in the equations of motion, then the first derivatives of u, v, and w with respect to x, y, and z are transformed to first derivatives with respect to ξ, η, and ζ by chain differentiation. That is

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \zeta} \\ \frac{\partial v}{\partial \xi} & \frac{\partial v}{\partial \eta} & \frac{\partial v}{\partial \zeta} \\ \frac{\partial w}{\partial \xi} & \frac{\partial w}{\partial \eta} & \frac{\partial w}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix}.$$

The matrix

$$J = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix}$$

is the Jacobian matrix of the transformation. If the functional relations

$$\xi = \xi(x,y,z), \quad \eta = \eta(x,y,z), \text{ and } \quad \zeta = \zeta(x,y,z)$$

are known, then the Jacobian matrix can be directly found by differentiating the functions. It is not necessary, however, to explicitly know ξ, η, and ζ as functions of x, y, and z to determine the Jacobian matrix. The inverse Jacobian matrix

$$J^{-1} \equiv \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

can be obtained by differentiating the functions

$$x = x(\xi,\eta,\zeta), \quad y = y(\xi,\eta,\zeta) \text{ and } \quad z = z(\xi,\eta,\zeta)$$

with respect to ξ, η, and ζ. With these derivatives

$$J = \frac{\text{Transposed of Cofactor } (J^{-1})}{|J^{-1}|},$$

where $|J^{-1}|$ is the Jacobian determinate.

$$|J^{-1}| = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

$$= \frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) - \frac{\partial x}{\partial \eta}\left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi}\right)$$
$$+ \frac{\partial x}{\partial \zeta}\left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi}\right)$$

Thus

$$J = \frac{1}{|J^{-1}|}\begin{bmatrix} \left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) & -\left(\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) & \left(\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\right) \\ -\left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi}\right) & \left(\frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \xi}\right) & -\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi}\right) \\ \left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi}\right) & -\left(\frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi}\right) & \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right) \end{bmatrix}$$

provided $|J^{-1}| \neq 0$. Higher derivative analysis can be pursued in a similar fashion. For more information on the transformation of partial differential equations the reader is referred to Reference 11.

It is rare to find algebraic expressions of the computational coordinates as functions of the physical coordinates. The preferred approach is to express the physical domain as a function of the computational domain and differentiate the physical grid with respect to the computational grid. It is very important that derivative evaluation be performed and

incorporated into the finite difference approximation of the equations of motion in such a manner that geometrically-induced errors are not created. References 17 and 18 address this subject.

ALGEBRAIC GRID GENERATION METHODS

In this section three similar algebraic grid generation techniques are discussed. The objective is to outline the mathematical structure so that the techniques can be compared for common salient features and individual merit. Each case is based upon the computational domain and the physical domain presented in the previous section. Intermediate functions which enhance the control of grid spacing may also be postulated.

Transfinite interpolation

Probably the most recent and comprehensive description of transfinite interpolation for application such as those arising in computational fluid dynamics is presented by Rizzi and Eriksson.[7] This description generally follows their format. A transformation from the computational domain to the physical domain is a vector-valued function

$$\vec{F}(\xi,\eta,\zeta) = \begin{bmatrix} x(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) \end{bmatrix} \tag{1}$$

where

$$0 \le \xi \le 1,$$
$$0 \le \eta \le 1,$$
$$0 \le \zeta \le 1.$$

The first formulation of transfinite interpolation, which we call the "point method" is based on knowing a sparsely-organized set of points in the computational domain $S_C = \left\{ \xi_{IKJ},\ \eta_{IJK},\ \zeta_{IJK} \right\}_{\substack{I=1 \\ J=1 \\ K=1}}^{\substack{I=L \\ J=N \\ K=M}}$ and the corresponding point set in

the physical domain $S_P = \left\{ x_{IJK},\ y_{IJK},\ z_{IJK} \right\}_{\substack{I=1 \\ J=1 \\ K=1}}^{\substack{I=L \\ J=N \\ K=M}}$. Note that the set S_C is not the uniform computational grid. Each point in the computational domain is at the intersection of three perpendicular planes, and in the physical domain each point is at the intersection of three surfaces (Fig. 3). The vector-valued representation on each surface is given by

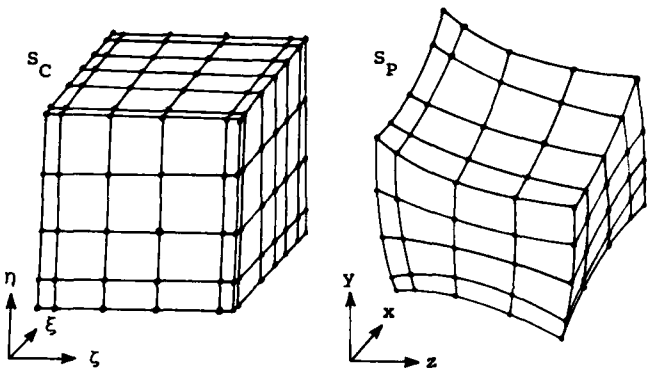


Fig. 3. Transfinite interpolation--point description

$$\vec{F}(\xi_I,\eta,\zeta) = \begin{bmatrix} x(\xi_I,\eta,\zeta) \\ y(\xi_I,\eta,\zeta) \\ z(\xi_I,\eta,\zeta) \end{bmatrix} = \vec{a}_I(\eta,\zeta),\ I=1\ldots L, \tag{2a}$$

$$\vec{F}(\xi,\eta_J,\zeta) = \begin{bmatrix} x(\xi,\eta_J,\zeta) \\ y(\xi,\eta_J,\zeta) \\ z(\xi,\eta_J,\zeta) \end{bmatrix} = \vec{b}_J(\xi,\zeta),\ J=1\ldots N, \tag{2b}$$

$$\vec{F}(\xi,\eta,\zeta_K) = \begin{bmatrix} x(\xi,\eta,\zeta_K) \\ y(\xi,\eta,\zeta_K) \\ z(\xi,\eta,\zeta_K) \end{bmatrix} = \vec{c}_K(\xi,\eta),\ K=1\ldots M. \tag{2c}$$

Interpolation between the points in the physical domain is performed by defining a set of blending functions:

$$\alpha_I(\xi);\ I=1\ldots L,$$

$$\beta_J(\eta);\ J=1\ldots N,$$

$$\gamma_K(\zeta);\ K=1\ldots M,$$

with the conditions

$$\alpha_I(\lambda_i) = \delta_{I\ell},\quad \ell=1\ldots L$$

$$\beta_J(\eta_\ell) = \delta_{J\ell},\quad \ell=1\ldots N$$

$$\gamma_K(\zeta_\ell) = \delta_{K\ell},\quad \ell=1\ldots M$$

where

$$\delta_{I\ell} = 0,\ I \neq \ell,$$

$$\delta_{I\ell} = 1,\ I = \ell,$$

$$\delta_{J\ell} = 0,\ J \neq \ell,$$

$$\delta_{J\ell} = 1,\ J = \ell,$$

$$\delta_{K\ell} = 0,\ K \neq \ell,$$

$$\delta_{K\ell} = 1,\ K = \ell.$$

The transfinite interpolation method is the application of the recursive algorithm

$$\vec{F}_1(\xi,\eta,\zeta) = \sum_{I=1}^{L} \alpha_I(\xi)\vec{a}_I(\eta,\zeta), \tag{3a}$$

$$\vec{F}_2(\xi,\eta,\zeta) = \vec{F}_1(\xi,\eta,\zeta) + \sum_{J=1}^{N} \beta_J(\eta)\left[\vec{b}_J(\xi,\zeta) - \vec{F}_1(\xi,\eta_J,\zeta)\right], \tag{3b}$$

$$\vec{F}(\xi,\eta,\zeta) = \vec{F}_2(\xi,\eta,\zeta) + \sum_{K=1}^{M} \gamma_K(\zeta)\left[\vec{c}_K(\xi,\zeta) - \vec{F}_2(\xi,\eta,\zeta_K)\right]. \tag{3c}$$

Each step of the algorithm is a univariate interpolation in one of three possible directions, or the steps can be combined into a single equation. Also, if it is assumed that $\vec{F}(\xi,\eta,\zeta)$ is continuous, the order of the interpolation direction is not important. Obviously, a large quantity of geometric information is required to define a grid. Deriving appropriate blending functions is the key element and it can vary from one grid problem to another.

Eriksson[5] uses only the outer boundary surfaces (Fig. 4) and out of surface derivatives at certain boundaries to define an interior grid. This is reasonable since normally a great deal of geometric information is known at bounding surfaces, but not always away from them. Eriksson's presentation is as follows and is referred to as the "outer surface" method.

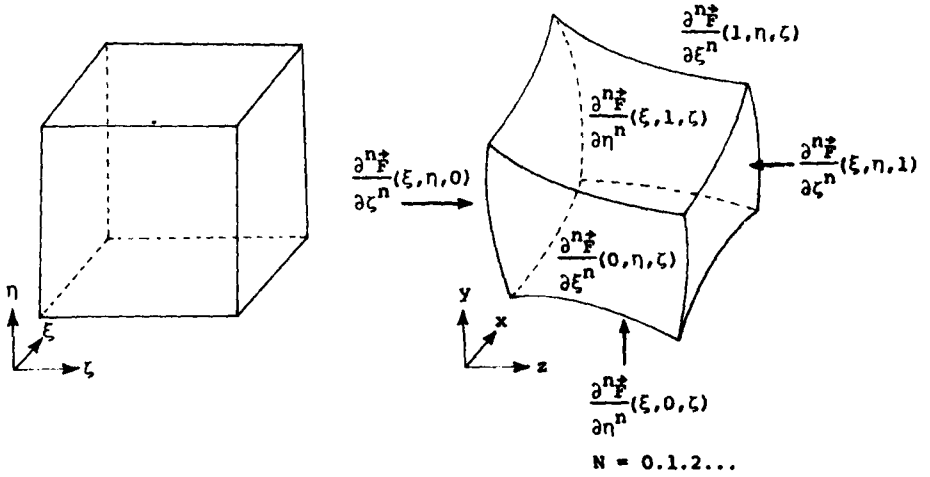


Fig. 4. Transfinite interpolation--outer surface description

Let

$$\frac{\partial^n \vec{F}}{\partial \xi^n}(\xi_\ell,\eta,\zeta) = \vec{A}^n_\ell(\eta,\zeta), \qquad \begin{matrix}\ell=1,2\\ n=0,1\ldots P\end{matrix}$$

$$\frac{\partial^n F}{\partial \eta^n}(\xi,\eta_\ell,\zeta) = \vec{B}_\ell^n(\xi,\zeta), \qquad \begin{matrix}\ell=1,2\\ n=0,1\ldots Q\end{matrix}$$

$$\frac{\partial^n F}{\partial \zeta^n}(\xi,\eta,\zeta_\ell) = \vec{C}_\ell^n(\xi,\eta), \qquad \begin{matrix}\ell=1,2\\ n=0,1\ldots R\end{matrix}$$

A set of blending functions is defined by

$$\alpha_\ell^{(n)}(\xi) \qquad \ell=1,2, \quad n=0,1\ldots P$$

$$\beta_\ell^{(n)}(\eta) \qquad \ell=1,2, \quad n=0,1\ldots Q$$

$$\gamma_\ell^{(n)}(\zeta) \qquad \ell=1,2, \quad n=0,1\ldots R$$

with the conditions

$$\frac{\partial^m \alpha_\ell^{(n)}(\xi)}{\partial \xi^m} = \delta_{\ell 1}\,\delta_{nm}$$

$$\frac{\partial^m \beta_\ell^{(n)}(\eta)}{\partial \eta^m} = \delta_{\ell 1}\,\delta_{nm}$$

$$\frac{\partial^m \gamma_\ell^{(n)}(\zeta)}{\partial \zeta^m} = \delta_{\ell 1}\,\delta_{nm}$$

and where the δ functions are defined by

$$\delta_{ij} = 0, \quad i \neq j, \quad \delta_{ij} = 1, \quad (i = j).$$

The transfinite interpolation algorithm becomes

$$\vec{F}_1(\xi,\eta,\zeta) = \sum_{\ell=1}^{2}\sum_{n=0}^{P} \alpha_\ell^{(n)}\ \vec{A}_\ell^n(\eta,\zeta) \tag{4a}$$

$$\vec{F}_2(\xi,\eta,\zeta) = \vec{F}_1(\xi,\eta,\zeta) + \sum_{\ell=1}^{2}\sum_{n=0}^{Q} \beta_\ell^{(n)}(\eta)\left[\vec{B}^n(\xi,\zeta) - \frac{\partial^n \vec{F}_1}{\partial \eta^n}(\xi,\eta_\ell,\zeta)\right] \tag{4b}$$

$$F(\xi,\eta,\zeta) = \vec{F}_2(\xi,\eta,\zeta) + \sum_{\ell=1}^{2}\sum_{n=0}^{R} \gamma_\ell^{(n)}(\zeta)\left[\vec{C}_\ell^n(\xi,\eta) - \frac{\partial^n \vec{F}_2}{\partial \zeta^n}(\xi,\eta,\zeta_\ell)\right] \tag{4c}$$

The boundary sets are

$$\left\{x_{\ell JK}, y_{\ell JK}, z_{\ell JK}\right\}_{\substack{K=1\\J=1\\\ell=1}}^{\substack{K=M\\J=N\\\ell=2}}, \quad \left\{0, \eta_{JK}, \zeta_{JK}\right\}_{\substack{K=1\\J=1}}^{\substack{K=M\\J=N}}, \quad \left\{1, \eta_{JK}, \zeta_{JK}\right\}_{\substack{K=1\\J=1}}^{\substack{K=M\\J=N}},$$

$$\left\{x_{I\ell K}, y_{I\ell K}, z_{I\ell K}\right\}_{\substack{K=1\\\ell=1\\I=1}}^{\substack{K=M\\\ell=2\\I=L}}, \quad \left\{\xi_{IK}, 0, \zeta_{IK}\right\}_{\substack{K=1\\I=1}}^{\substack{K=M\\I=L}}, \quad \left\{\xi_{IK}, 1, \zeta_{IK}\right\}_{\substack{K=1\\I=1}}^{\substack{K=M\\I=L}},$$

$$\left\{x_{IJ\ell}, y_{IJ\ell}, z_{IJ\ell}\right\}_{\substack{\ell=1\\J=1\\I=1}}^{\substack{\ell=2\\J=N\\I=L}}, \quad \left\{\xi_{IJ}, \eta_{IJ}, 0\right\}_{\substack{J=L\\I=L}}^{\substack{J=N\\I=L}}, \quad \left\{\xi_{IJ}, \eta_{IJ}, 1\right\}_{\substack{J=1\\I=1}}^{\substack{J=N\\I=L}}.$$

Also, outward derivatives at certain of these boundaries as well as the blending functions are required for this formulation. Again the choice of the blending functions is critical to the successful application of the method.

The multisurface method

The multisurface method developed by Peter Eiseman is a procedure for generating coordinates between an inner boundary surface $\vec{S}_1(\xi,\zeta)$ and an outer boundary surface $\vec{S}_N(\xi,\zeta)$. An arbitrary number of internal surfaces $\vec{S}_2(\xi,\zeta)\ldots\vec{S}_{N-1}(\xi,\zeta)$ are introduced to control the coordinate representation between $\vec{S}_1(\xi,\zeta)$ and $\vec{S}_N(\xi,\zeta)$ (Fig. 5). Each surface representation is such that

$$\vec{S}_k(\xi,\zeta) = \begin{bmatrix} x_k(\xi,\zeta) \\ y_k(\xi,\zeta) \\ z_k(\xi,\zeta) \end{bmatrix}, \quad k=1....N.$$

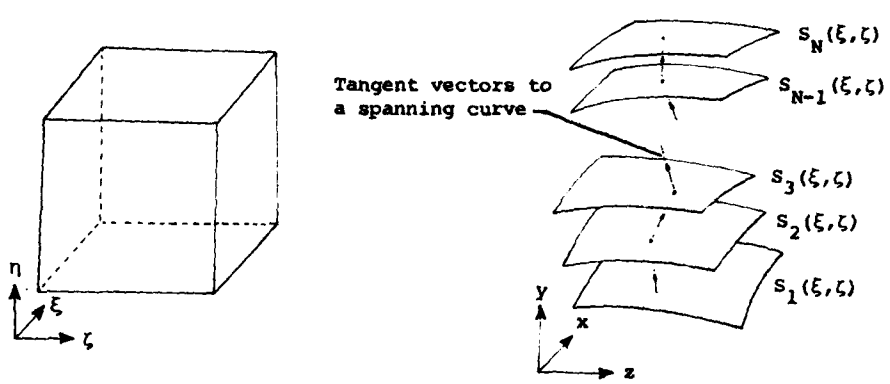


Fig. 5. The multisurface method

The physical domain can be written as

$$\vec{F}(\xi,\eta,\zeta) = \begin{bmatrix} x(\vec{S}_1(\xi,\zeta), \vec{S}_2(\xi,\zeta)\ldots\vec{S}_N(\xi,\zeta),\eta) \\ y(\vec{S}_1(\xi,\zeta), \vec{S}_2(\xi,\zeta)\ldots\vec{S}_N(\xi,\zeta),\eta) \\ z(\vec{S}_1(\xi,\zeta), \vec{S}_2(\xi,\zeta)\ldots\vec{S}_N(\xi,\zeta),\eta) \end{bmatrix} \tag{5}$$

where

$$0 \le \xi \le 1, \quad 0 \le \eta \le 1, \quad 0 \le \zeta \le 1.$$

The variable η is the independent variable spanning between surfaces. With this introduction the description of the multisurface method generally follows that presented in Reference 16.

It is assumed that the set of surfaces described above are ordered from bounding surface to bounding surface, and for a fixed ξ and ζ there is a corresponding point on each surface. The intermediate surfaces are not coordinate surfaces, but instead are surfaces which are used to establish a field of tangent vectors to the coordinate curve spanning across the surfaces. For the time being, it is assumed that the bounding surfaces are coordinate surfaces. A smooth interpolation connecting the bounding surfaces results in a smooth vector field of tangent directions but with unspecified magnitudes. A unique vector field of tangents is obtained by correctly choosing magnitudes which on integration fit precisely the bounding surfaces. This is demonstrated with the vector field of tangents given by

$$\vec{V}_k(\xi,\zeta) = E_k\left[\vec{S}_{k+1}(\xi,\zeta) - \vec{S}_k(\xi,\zeta)\right] \qquad k=1..N, \text{ (Fig. 6).} \tag{6}$$

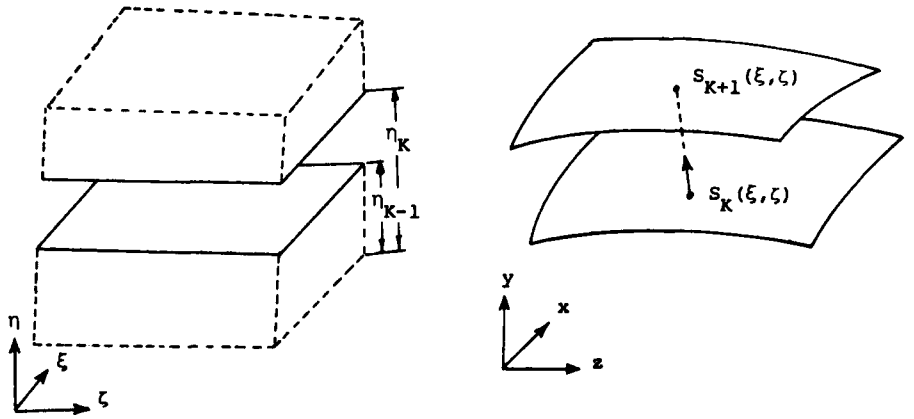


Fig. 6. Tangents to a piecewise linear curve and a partition of the spanning variable from the computational domain

The coefficients E_k are scalars which determine the magnitude of the vectors but not the direction. Using the independent variable η for the spanning direction, a partition $\eta_1 < \eta_2 \ldots < \eta_{N-1}$ can be specified in correspondence with the tangents in Eq. (6). The partitioned variable can be used to represent the tangents as discrete vector-valued functions which map η_k into $\vec{V}_k(\xi,\zeta)$. The first derivative of $\vec{F}(\xi,\eta,\zeta)$ with respect to η is given by

$$\frac{\partial \vec{F}}{\partial \eta}(\xi,\eta,\zeta) = \sum_{k=1}^{N-1} \psi_k(\eta)\ \vec{v}_k(\xi,\zeta) \tag{7}$$

where

$$\psi_k(\eta_\ell) = \delta_{k\ell},$$

and

$$\delta_{k\ell} = 0 \quad k \neq \ell,$$

$$\delta_{k\ell} = 1 \quad k = \ell.$$

The interpolants $\psi_k(\eta)$ are defined exactly like the blending functions in Eq. (3) but here they are used to describe a derivative function and multiply a tangent vector field. Integrating Eq. (7) with an initial η and $\vec{S}_1(\xi,\zeta)$ yields

$$\vec{F}(\xi,\eta,\zeta) = \vec{S}_1(\xi,\zeta) + \sum_{k=1}^{N-1} E_k G_k(\eta) \left[\vec{S}_{k+1}(\xi,\zeta) - \vec{S}_k(\xi,\zeta)\right] \tag{8}$$

where

$$G_k(\eta) = \int_{\eta_1}^{\eta} \psi_k(x)\ dx.$$

If the magnitudes E_k are chosen so that each $E_k G_k(\eta_{N-1}) = 1$, then the evaluation of Eq. (8) at η_{N-1} reduces to $S_N(\xi,\zeta)$. This allows Eq. (8) to be expressed

$$\vec{F}(\xi,\eta,\zeta) = \vec{S}_1(\xi,\zeta) + \sum_{k=1}^{N-1} \frac{G_k(\eta)}{G_k(\eta_{N-1})} \left[\vec{S}_{k+1}(\xi,\zeta) - \vec{S}_k(\xi,\zeta)\right] \tag{9}$$

which is referred to as Eiseman's general multisurface transformation.

The basic ingredients of the multisurface method are the partition $\eta_1 < \eta_2 \ldots < \eta_{N-1}$, the interpolents ψ_k and the surfaces $\vec{S}_k(\xi,\zeta)$. Choosing ψ_k to be polynomials of degree N in η, the curve connecting the bounding surfaces is of degree $N + 1$. In a systematic fashion

$$\psi_k(\eta) = \prod_{\substack{i=1 \\ i\neq k}}^{N-1} (\eta - \eta_i).$$

An example is a three surface transformation $(N - 1 = 2)$ and with $\eta_1 = 0$, $\eta_2 = 1$, $\psi_1 = 1 - \eta$, and $\psi_2 = \eta$ then

$$G_1(\eta) = \eta - \frac{\eta^2}{2}, \quad G_2(\eta) = \frac{\eta^2}{2},$$

$$\vec{F}(\xi,\eta,\zeta) = \left[1 - \eta(2 - \eta)\right]\vec{S}_1(\xi,\zeta) + \left[\eta(2 - \eta) - \eta^2\right]\vec{S}_2(\xi,\zeta) + \eta^2\vec{S}_3(\xi,\zeta),$$

or

$$\vec{F}(\xi,\eta,\zeta) = \sum_{i=1}^{3} \beta_k(\eta)\vec{S}_k(\xi,\zeta),$$

where

$$\beta_1(\eta) = 1 - 2\eta + \eta^2,$$

$$\beta_2(\eta) = 2\eta - 2\eta^2,$$

$$\beta_3(\eta) = \eta^2.$$

Comparing the multisurface method with transfinite interpolation (Eq. (3)), the multisurface method requires interpolants $\psi_k(\eta)$ and one set of surfaces, and allows interpolation in only one coordinate direction. It is apparent that blending functions can be derived from Eiseman transformation formulas

starting with interpolants. It is important to remember that the most difficult aspect of algebraic grid generation is the determination of functions (blending functions interpolants, etc.) which control a grid. The emphasis in the multi-surface development is on deriving interpolants which provide satisfactory control.

The two-boundary technique

The two-boundary technique has been described by Smith[1,2] and Smith and Weigel.[13] The technique has common characteristics with Eriksson's formulation of transfinite interpolation where position and derivatives on exterior boundaries along with blending functions are used to define the physical domain. For the two-boundary technique blending functions are specified to be linear and cubic polynomials as described by Coons.[5] These blending functions can also be derived from Eiseman's two-surface definition and a special modification of the four-surface definition.[16] Later control functions are incorporated to further enhance grid spacing control.

The technique is based on defining two nonintersecting surfaces $\vec{s}_1(\xi,\zeta)$ and $\vec{s}_2(\xi,\zeta)$ (Fig. 7) where

$$\vec{s}_1(\xi,\zeta) = \begin{bmatrix} x_1(\xi,\zeta) \\ y_1(\xi,\zeta) \\ z_1(\xi,\zeta) \end{bmatrix}, \qquad \vec{s}_2(\xi,\zeta) = \begin{bmatrix} x_2(\xi,\zeta) \\ y_2(\xi,\zeta) \\ z_2(\xi,\zeta) \end{bmatrix}$$

and

$$0 \le \xi \le 1, \qquad 0 \le \zeta \le 1.$$

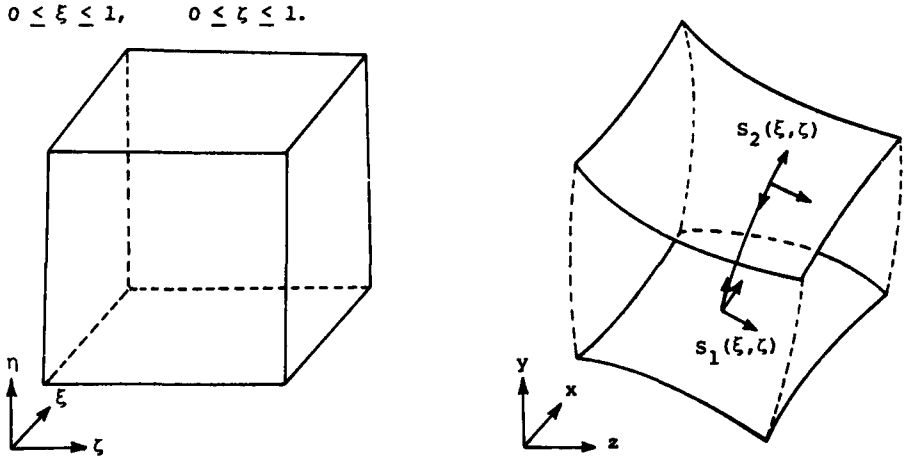


Fig. 7. The two-boundary technique

The physical domain is expressed

$$\vec{F}(\xi,\eta,\zeta) = \begin{bmatrix} x(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) \end{bmatrix} = \begin{bmatrix} x(\vec{s}_1(\xi,\zeta), \vec{s}_2(\xi,\zeta),\eta) \\ y(\vec{s}_1(\xi,\zeta), \vec{s}_2(\xi,\zeta),\eta) \\ z(\vec{s}_1(\xi,\zeta), \vec{s}_2(\xi,\zeta),\eta) \end{bmatrix} .$$

Explicit forms of the two-boundary technique are linear and hermite cubic interpolation. The linear form is

$$\vec{F}(\xi,\eta,\zeta) = \sum_{k=1}^{2} \lambda_k(\eta)\vec{s}_k(\xi,\zeta) \tag{10}$$

where

$$\lambda_1(\eta) = 1 - \eta,$$

$$\lambda_2(\eta) = \eta.$$

The cubic formulation is

$$\vec{F}(\xi,\eta,\zeta) = \sum_{k=1}^{2} \mu_k(\eta)\vec{s}_k(\xi,\zeta) + \sum_{k=1}^{2} \mu_{k+2}(\eta)\left[\frac{\partial\vec{s}_k}{\partial\xi}(\xi,\zeta) \times \frac{\partial\vec{s}_k}{\partial\zeta}(\xi,\zeta)\right] \tag{11}$$

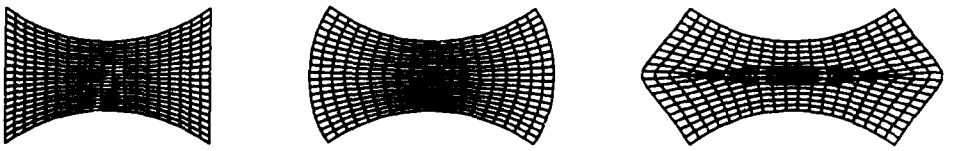
where

$$\begin{aligned} \mu_1(\eta) &= 2\eta^3 - 3\eta^2 + 1, \\ \mu_2(\eta) &= -2\eta^3 + 3\eta^2, \\ \mu_3(\eta) &= \eta^3 - 2\eta^2 + 1, \\ \mu_4(\eta) &= \eta^3 - \eta^2, \\ 0 \le \eta &\le 1. \end{aligned} \tag{12}$$

the cross product of surface derivatives or normal derivatives at the boundaries is given by

$$\frac{\partial \vec{S}_k}{\partial \xi}(\xi,\zeta) \times \frac{\partial \vec{S}_k}{\partial \zeta}(\xi,\zeta) = T_k \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial x_k}{\partial \xi}(\varepsilon,\zeta) & \frac{\partial y_k}{\partial \xi}(\varepsilon,\zeta) & \frac{\partial z_k}{\partial \xi}(\varepsilon,\zeta) \\ \frac{\partial x_k}{\partial \zeta}(\varepsilon,\zeta) & \frac{\partial y_k}{\partial \zeta}(\varepsilon,\zeta) & \frac{\partial z_k}{\partial \zeta}(\varepsilon,\zeta) \end{vmatrix}, \quad k=1,2. \tag{13}$$

The constants T_k control the magnitudes of the normal derivatives of the boundaries. For nonzero T_k a grid resulting from this formulation is orthogonal at the boundaries $\vec{S}_k(\xi,\zeta)$. Increasing the magnitudes T_k forces the effect of orthogonality further into the interior of the physical grid. If the magnitudes are too large, the grid becomes double-valued and is unsatisfactory (Fig. 8). Kowalski allows T_k to be a function of ξ and ζ $(T_k(\xi,\zeta))$ which allows variable effect of orthogonality over the domain.



No orthogonality magnitude

Satisfactory orthogonality magnitude

Unsatisfactory orthogonality magnitude

Fig. 8. Grid orthogonality

The key ingredients for the two-boundary technique, as it is presented here, are two nonintersecting bounding surfaces, normal magnitude constants or normal magnitude functions. It is later shown that additional ingredients are control functions which govern the spacing of a grid. The following section deals with surface representation which is used to define the bounding surfaces.

SURFACE REPRESENTATION AND PARAMETERIZATION

Surface representation is an important aspect of the algebraic grid generation methods. Normally, the initial description of a surface is in terms of an organized point set $S = \left\{ z_{IK}, y_{IK}, z_{IK} \right\}_{I=1}^{I=L}{}_{K=1}^{K=M}$ (Fig. 9). It is desirable to find a functional representation

$$\vec{S}(r,t) = \begin{bmatrix} x(r,t) \\ y(r,t) \\ z(r,t) \end{bmatrix}$$

with two independent variables r and t, which contains S, and related to independent variables from the computational domain (i.e., ξ and ζ). A process that is recommended for many grid generation problems is:

1. parameterize the data set S with normalized arc length or approximate normalized arc length (Fig. 10);

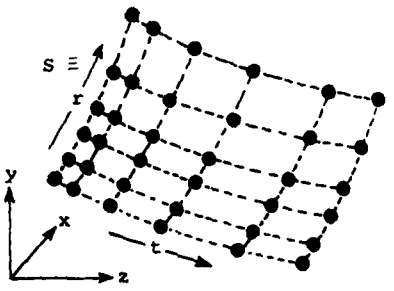


Fig. 9. Surface representation

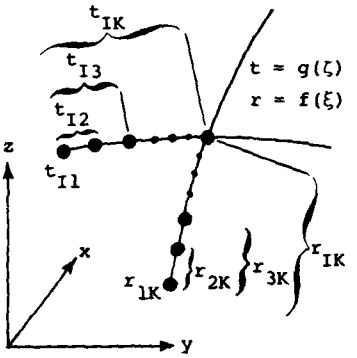


Fig. 10. Arc length parameterization

2. construct single valued functions relating the arc lengths to the computational coordinates (Fig. 10); and

3. interpolate the data set S with a bidirectional interpolation procedure such as bicubic splines with the arc lengths being the independent variables (Fig. 11).

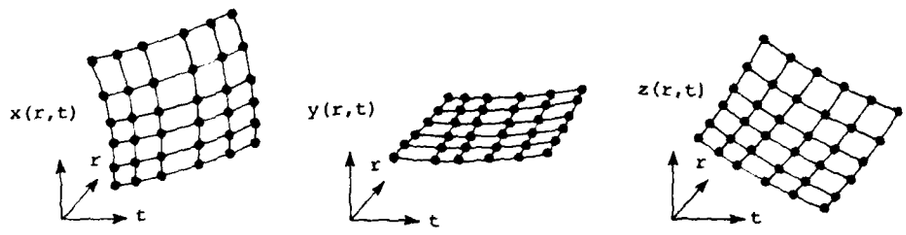


Fig. 11. Bi-directional interpolation

Approximate arc lengths are computed from the set S by

$$r_{IK} = r_{I-1K} + \left[(x_{IK} - x_{I-1K})^2 + (y_{IK} - y_{I-1K})^2 + (z_{IK} - z_{I-1K})^2\right]^{1/2}$$

$$t_{IK} = t_{IK-1} + \left[(x_{IK} - x_{IK-1})^2 + (y_{IK} - y_{IK-1})^2 + (z_{IK} - z_{IK-1})^2\right]^{1/2}$$

$$r_{11} = t_{11} = 0,$$

$$I = 1 \ldots L,$$

$$K = 1 \ldots M.$$

Normalized approximate arc lengths are

$$r_{IK} = \frac{r_{IK}}{r_{LK}}, \qquad t_{IK} = \frac{t_{IK}}{t_{IM}}, \qquad 0 \le r_{IK} \le 1, \qquad 0 \le t_{IK} \le 1.$$

After forming the sets $S_x = \left\{x_{IK}, r_{IK}, t_{IK}\right\}_{\substack{K=1 \\ I=1}}^{\substack{K=M \\ I=L}}$, $S_y = \left\{y_{IK}, r_{IK}, t_{IK}\right\}_{\substack{K=1 \\ I=1}}^{\substack{K=M \\ I=L}}$ and $S_z = \left\{z_{IK}, r_{IK}, t_{IK}\right\}_{\substack{K=1 \\ I=1}}^{\substack{K=M \\ I=L}}$ three bidirectional interpolations can be performed

for $x(r,t)$, $y(r,t)$ and $z(r,t)$. The intermediate variables r and t are defined on the unit interval, and likewise ξ and ζ are defined on the unit interval. Constructing single-valued functions $r = f(\xi)$ and $t = g(\zeta)$ has the effect of controlling the location of an arbitrary grid point on the surface. Each surface can have different functions relating the computational variables to the parametric variables. The functions $f(\xi)$ and $g(\zeta)$ are called "control functions" and are discussed in more detail at a later point.

UNIFORMITY

In the previous section computational variables are related to parametric variables describing surfaces. In a similar but more complex manner the computational variable spanning between surfaces can be parameterized relative to arc length. This is particularly desirable for the two-boundary technique. The variable η used in the cubic blending functions for spanning between the boundary surfaces (Eq. (12)) is a mathematical entity and has no physical meaning other than it represents a coordinate from the computational domain. Uniformly discretizing the variable η for a fixed ξ, ζ, and T_k yields from Eq. (10) coordinates in the physical domain along a space curve (Fig. 12).

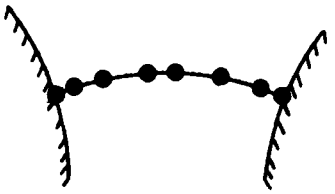


Fig. 12. Natural distribution of grid points along a spanning curve

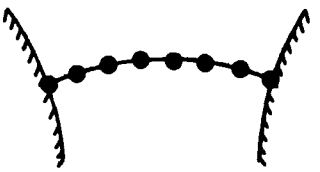


Fig. 13. A uniform distribution of grid points with respect to arc length along a spanning curve

In addition to defining the shape of the space curve, a distribution of points (grid points) along the curve is specified. This distribution is fixed unless some control function replaces η in the blending functions. Before applying control it is often desirable to initially specify a uniform reference distribution of grid points along the space curve connecting the boundaries (Fig. 13). One approach for obtaining a uniform distribution is to compute the normalized arc length or approximate normalized arc length (s) along the space curve. This establishes an empirical relation between the

variable s and the variable η. Uniformly discretizing s, performing a single-valued interpolation for η, and substituting into the blending functions yields uniform distributions of grid points along the curve. Alternately, the blending functions can be redefined in terms of s where now $s = e(\eta)$ and

$$\mu_1(\eta) = 2s^3 - 3s^2 + 1,$$

$$\mu_2(\eta) = 2s^3 + 3s^2,$$

$$\mu_3(\eta) = s^3 - 2s^2 + s,$$

$$\mu_4(\eta) = s^3 - s^2,$$

$$s = \eta, \quad s = e(\eta).$$

This procedure is complex because it has two steps and the necessity for an additional interpolation. Control of the grid spacing distribution relative to the uniform distribution can be accomplished by constructing a single valued function on the unit interval such that

$$s = h[e(\eta)].$$

The function $h[e(\eta)]$ is a control function for the spanning direction.

GRID SPACING CONTROL

The spacing of a grid in the phyiscal domain is primarily affected by how the computational coordinates are incorporated into the blending functions, interpolants, or surface constraints. Eiseman presents what he calls "piecewise local control" through the derivation of interpolants and the reader is referred to References 11 and 16 for this approach. Another approach is the construction of control functions which are embedded in the blending functions or surface constraints. Control functions are demonstrated using the two-boundary technique in two dimensions and with cubic blending functions.

The relationship between the computational domain and the physical domain for the two-boundary technique in two dimensions is given by

$$x(\xi,\eta) = x_1(r_1)\mu_1(s) + x_2(r_2)\mu_2(s) + T_1 \frac{dy_1}{dr_1}(r_1)\mu_3(s) + T_2 \frac{dy_2}{dr_2}(r_2)\mu_4(s),$$

(14)

$$y(\xi,\eta) = y_1(r_1)\mu_1(s) + y_2(r_2)\mu_2(s) - T_1 \frac{dx}{dr_1}(r_1)\mu_3(s) - T_2 \frac{dx}{dr_2}(r_2)\mu_4(s),$$

and

$$\mu_1(s) = 2s^3 - 3s^2 + 1,$$

$$\mu_2(s) = -2s^3 + 3s^2,$$

$$\mu_3(s) = s^3 - 2s^2 + s,$$

$$\mu_4(s) = s^3 - s^2,$$

where

$x_1(r_1), y_1(r_1)$ ≡ position on the first boundary as a function of normalized arc length along the boundary

$x_2(r_2), y_2(r_2)$ ≡ position on the second boundary as a function of normalized arc length along the boundary

$\frac{dx_1}{dr_1}(r_1), \frac{dy_1}{dr_1}(r_1)$ ≡ first derivative along the first boundary with respect to normalized arc length along the boundary

$\frac{dx_2}{dr_1}(r_2), \frac{dy_2}{dr_2}(r_2)$ ≡ first derivative along the second boundary with respect to normalized arc length along the boundary

T_1, T_2 ≡ normal derivative magnitudes for the respective boundaries

$r_1 = f_1(\xi)$ ≡ normalized arc length along the first boundary

$r_2 = f_2(\xi)$ ≡ normalized arc length along the second boundary

$s = h[e(\eta)]$ ≡ arc length along the grid curves connecting the two boundaries

ξ, η
$0 \leq \xi \leq 1$
$0 \leq \eta \leq 1$
≡ coordinates from the computational domain

Uniformly discretizing ξ and η and given the other quantities described above a corresponding grid is generated in the physical domain from Eq. (14).

For grid curves connecting the two boundaries (Fig. 14), their relationship and spacing relative to their neighboring grid curves is based on position, derivatives, and derivative magnitudes at the two boundaries, and the blending functions. Given that the blending functions are the same for all grid curves, the spacing between the curves is only a function of boundary information. The boundary positions and derivatives are a function of normalized arc lengths which are in turn written as functions of the computational coordinate ξ. It is the functions $f_1(\xi)$ and $f_2(\xi)$ that ultimately control the spacing between grid curves. When there is relatively low slope in these functions (Fig. 14), there is concentration of grid curves, and when there is relatively high slope the grid curves are dispersed (Fig. 15). In a similar manner

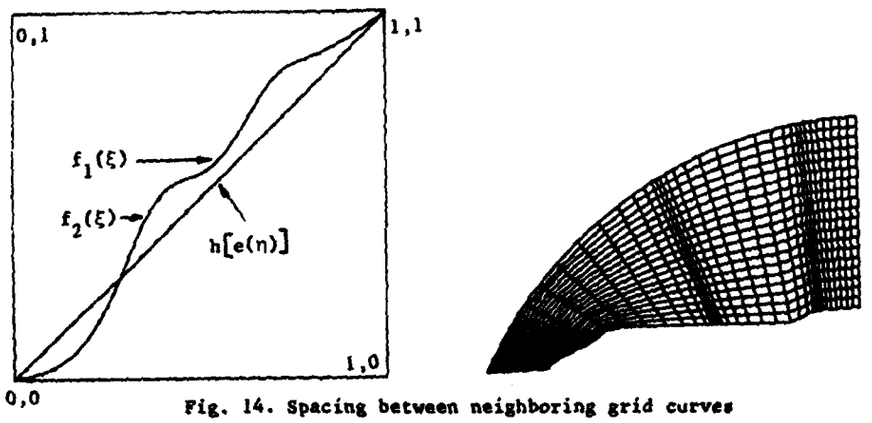


Fig. 14. Spacing between neighboring grid curves

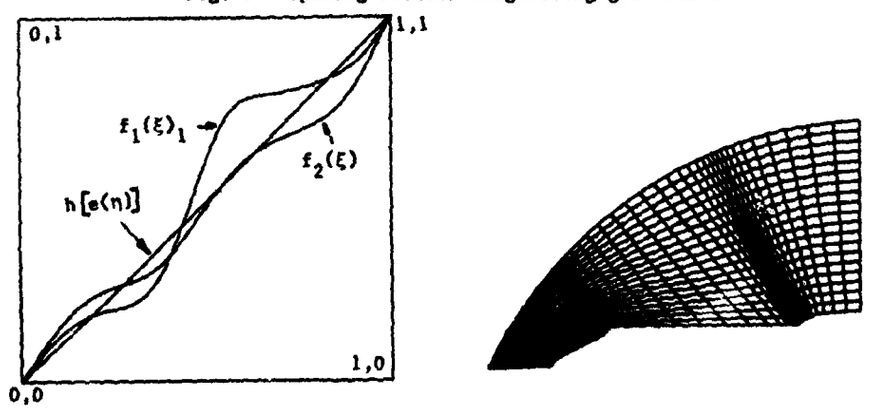


Fig. 15. Effect of control functions

the grid points along a grid curve are distributed by the blending functions. A control function $h[e(\eta)]$ relating η to the normalized arc length determines the final grid point distribution along the grid curve (Fig. 16).

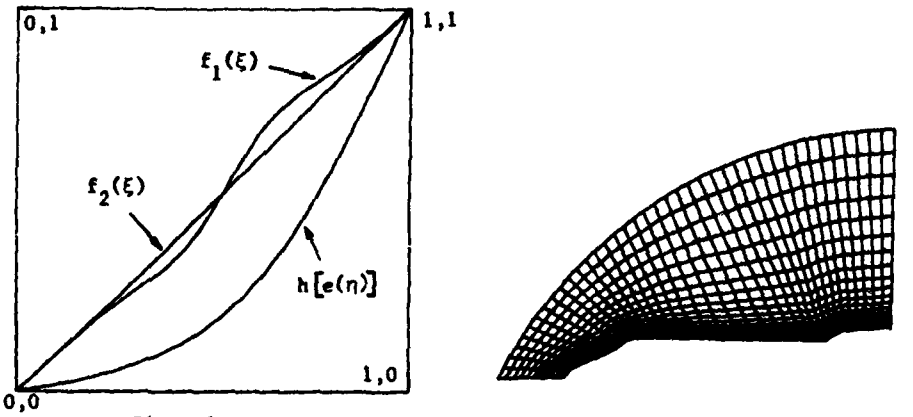


Fig. 16. Control of grid points along grid curves

The functions $f_1(\xi), f_2(\xi)$ and $h[e(\eta)]$ are called control functions. They should be single valued, smooth, and have smooth derivatives. Another condition is that the functions are defined on the unit square (Fig. 17).

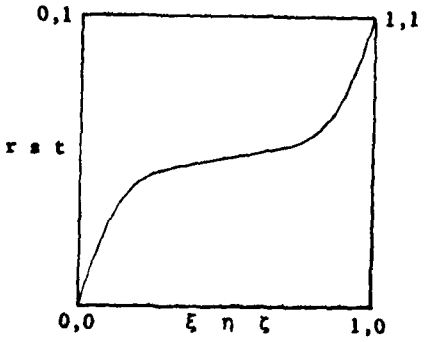


Fig. 17. Domain for the definition of control functions

The control functions can be analytic functions such as

$$r_1 = \frac{e^{\hat{K}\xi} - 1}{e^{\hat{K}} - 1}$$

where the parameter $\hat{K}$ controls the concentration of grid curves near the first boundary. In general, analytic functions are restrictive relative to where control can be applied. Another approach for arbitrary control is the use of smoothing spline functions on the unit square. This approach is described in Reference 15.

SIDE BOUNDARY CONSTRAINTS

The two-boundary technique can be constrained by boundaries intersecting the two primary boundaries. This is accomplished by applying the technique as previously described, and then applying the recursive formulas of transfinite interpolation. An example with one side boundary constraint is demonstrated (Fig. 18a). The two-boundary technique is performed with the formulation

$$\vec{F}_1(\xi,\eta,\zeta) = \vec{F}(\vec{S}_1^1(f_1(\xi),g_1(\zeta)),\vec{S}_2^1(f_2(\xi),g_2(\xi)),h(\eta)).$$

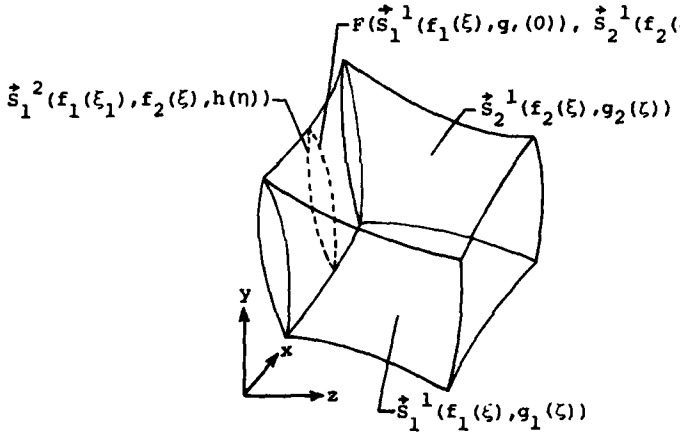


Fig. 18a. Step one in the two-boundary technique

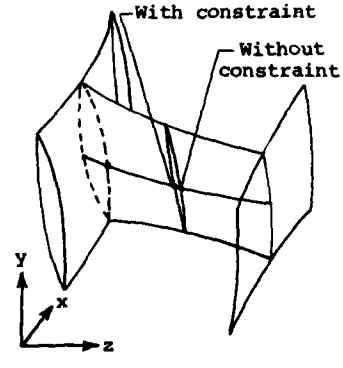


Fig. 18b. Side boundary constraint

The second step is from the transfinite interpolation formulation with a linear blending function (Fig. 18b)

$$\vec{F}(\xi,\eta,\zeta) = \vec{F}_1(\xi,\eta,\zeta) + (1-\zeta)\left[\vec{S}_1^2(f_1(\xi),f_2(\xi),h(\eta)) - \vec{F}(\vec{S}_1^1(f_1(\xi),g_1(0)),\vec{S}_2^1(f_2(\xi),g_2(0)),h(\eta))\right].$$

GRID GENERATION TOPOLOGY

The algebraic grid generation techniques that are presented are defined with the assumption that a uniform rectangular computational domain transforms into a physical domain. Also, exterior boundaries of the computational domain transform into boundaries on the physical domain. Consequently the topology of the physical domain strongly influences how a grid generation technique is applied. It is obvious that single six-sided box (computational domain) or a square in two dimensions is not going to transform into all physical domains. Further, in certain cases, transformations can only be made by introducing singularities. Problems most often arise when there are closed boundaries and in this section some topological considerations are described.

For boundaries in two dimensions, there are two primary types of physical domains that transform from a square computational domain. They are O-type domains and C-type domains, and are more commonly referred to as O-grids and C-grids. Several two-dimensional domains are described schematically in Figure 19 with corresponding boundary numbers in the computational and physical domains. Also, multiple computational and physical domains can be coupled with a common boundary. A resulting grid as well as grid derivatives should be continuous across a common boundary. If there is a discontinuity, special consideration should be taken in the finite difference procedure for the solution of the equations of motion.

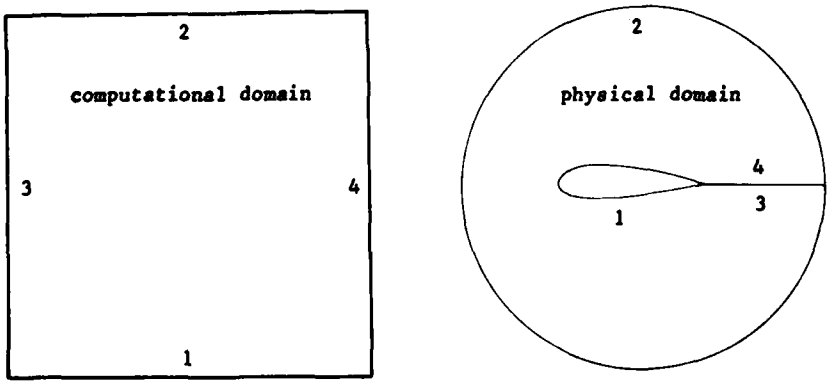


Fig. 19a. Simple O-grid

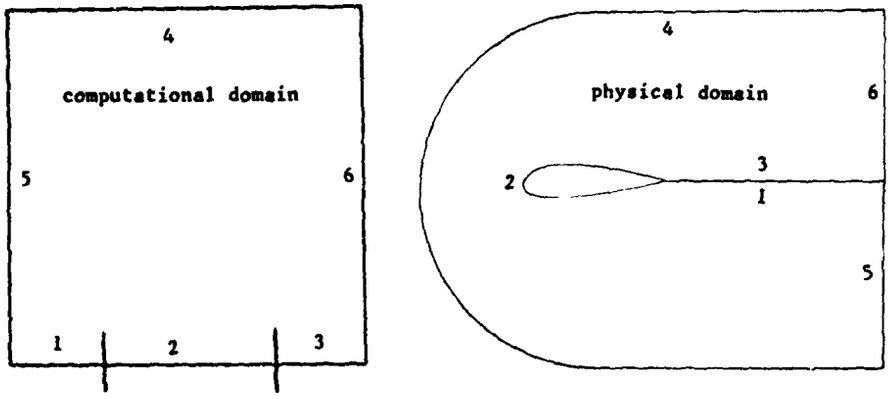


Fig. 19b. Simple C-grid

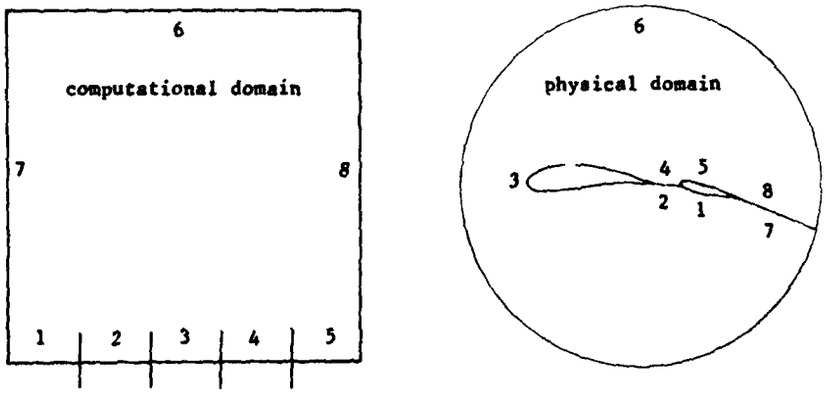


Fig.19c. Multiple body O-grid

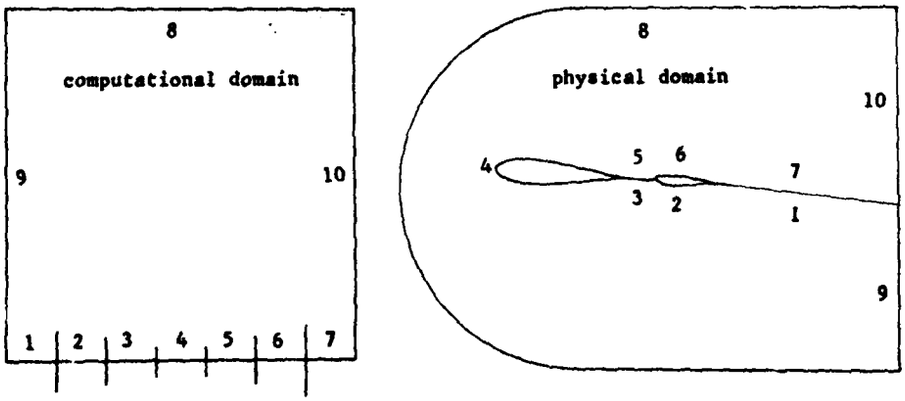


Fig. 19d. Multiple body C-grid

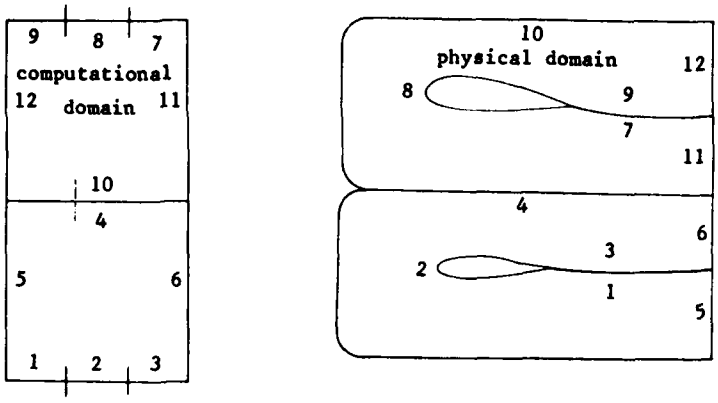


Fig. 19e. Combination domain C-grid

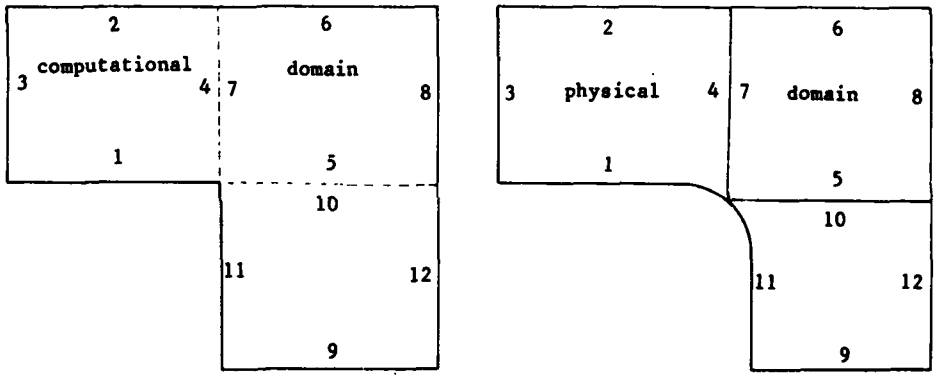


Fig. 19f. L-shape domain

For closed boundaries in three dimensions two suitable types of domains are O-O domains and C-O domains. The topologies associated with closed boundary domains in three dimensions is more complex than for two dimensions. A primary reason is that a planar surface from the computational domain will not transform into a closed three-dimensional surface without introducing singularities. Rizzi and Eriksson[6] extensively discuss the problems of three-dimensional closed surface topology and associated singularities and the reader is referred to Reference 6.

A final note on topology and singularities is that every effort should be made to place an unavoidable singularity in a region where there is little change in the basic equations of motion. Near a singularity there are large changes in the derivatives of the computational coordinates with respect to

the physical coordinates. These large changes between neighboring grid points can lead to inaccuracy and/or possibly instability in a finite difference procedure for the solution of the equations of motion.

GRID COMPUTATION

Algebraic grid generation methods generally require a large quantity of input data. These data can be divided into two types: "fixed data" and "variable data." Fixed data describes quantities such as bounding surfaces, whereas variable data describes quantities such as internal control surfaces of control functions. Interactive computer graphics is ideally suited for this type of application. Variable data which control a grid can be modified with cursor or numeric input at a graphics terminal, the resulting grid and grid derivatives visually observed, and the data again modified until satisfactory grid characteristics are achieved.

The algebraic grid generation methods work well in an interactive environment because the computation is explicit with no iteration necessary for a given grid solution. What is necessary, however, is that the communication rate between the computer and graphics terminal be high (9600 baud or greater) because of the large number of line segments that must be displayed for a grid. An aside point for the algebraic methods that are discussed is that the grid characteristics can be worked out on a relative small grid (few grid points) and the resulting control directly applied to compute a larger grid.

DISCUSSION AND CONCLUSIONS

The three algebraic grid generation techniques that are described are basically interpolation procedures with several common characteristics. Blending functions, interpolants, or control functions along with physical geometric constraints govern the transformation of a uniform rectangular computational grid into a physical grid. The methods are relatively simple to understand, they are explicit and do not require extensive computational effort, and they have a high degree of generality.

Next to the computational procedure, the most important consideration for grid generation is the topology of the physical domain. For three dimensions where there are closed boundaries, singularities are introduced. Care should be taken relative to where singularities exist and the corresponding effect in the finite difference procedure for the solution of the equations of

motion. Grid topology is not extensively discussed in this paper, but its importance is emphasized.

Another consideration is that grid derivatives must be smooth. This means that each step in a grid generation method must produce a smooth result. Wiqqles is one step propagate into the next step and finally into the grid.

A final point is that the use of interactive computer graphics for grid generation is highly advantageous. Until the time is reached when grid generation is truely coupled with the equations of motion and the grid control is adaptive, instantaneous human intervention in the control process in the next best approach. Since algebraic grid generation methods are explicit and require relatively few computations, they work very well in an interactive environment. This implies that the development of computer applications software is an important aspect of algebraic grid generation.

REFERENCES

1. Smith, R. E., Two-Boundary Grid Generation for the Solution of the Three-Dimensional Navier-Stokes Equations, Ph.D. Dissertation, Old Dominion University, May 1981.
2. Smith, R. E., "Two-Boundary Grid Generation for the Solution of the Three-Dimensional Navier-Stokes Equations," NASA TM-83123, May 1981.
3. Dugundji, James, Topology, Allyn and Bacon Inc., 1968.
4. Gordon, W. J. and Hall, C. A., "Construction of Curvilinear Coordinate Systems and Application to Mesh Generation," International Journal for Numerical Methods in Engineering, Vol. 7, 461-477, 1973.
5. Coons, S. A., "Surfaces for Computer Aided Design of Space Forms," MAC TR-41, Contract No. AF-33(6000-42859) MIT, June 1967.
6. Eriksson, Lars-Erik, "Three-Dimensional Spline-Generated Coordinate Transformations for Grids Around Wing-Body Configurations," Numerical Grid Generation Techniques, NASA CP 2166, 1980.
7. Rizzi, A. and Eriksson, L. E., "Transfinite Mesh Generation and Damped Euler Equation Algorithm for Transonic Flow Around Wing-Body Configurations," AIAA 5th Computational Fluid Dynamics Conference, June 1981, Palo Alto, California.
8. Anderson, P. G. and Spradley, L. W., "Finite-Difference Grid Generation by Multivariate Blending Function Interpolation," Numerical Grid Generation Techniques, NASA CP 2166, 1980.
9. Spradley, L. W. Stalnaker, J. F., and Ratliff, A. W., "Computation of Three-Dimensional Viscous Flows with the Navier-Stokes Equations," AIAA Paper 80-1348, AIAA 13th Fluid and Plasma Dynamics Conference, Snowmass, Colorado.
10. Eiseman, P. R., "A Multi-Surface Method of Coordinate Generation," Journal of Computational Physics, Vol. 33, No. 1, October 1979.
11. Eiseman, P. R., "Geometric Methods in Computational Fluid Dynamics," ICASE Report No. 81-11, April 1980.
12. Eiseman, P. R., "A Coordinate System for a Viscous Transonic Cascade Analysis," Journal of Computational Physics, Vol. 29, 1978.
13. Eiseman, P. R., "Three-Dimensional Coordinates About Wings," Fourth AIAA Computer Fluid Dynamics Conference, Williamsburg, Virginia, July 1979.

14. Smith, R. E. and Weigel, B. L., "Analytic and Approximate Boundary-Fitted Coordinate Systems for Fluid Flow Simulation," AIAA Paper 80-0192, AIAA 18th Aerospace Sciences Meeting, January 1980, Pasadena, California.
15. Kowalski, E. J., "Boundary-Fitted Coordinate Systems for Arbitrary Computational Regions," Numerical Grid Generation Techniques, NASA CP 2166, 1980.
16. Eiseman, P. R. and Smith, R. E., "Mesh Generation Using Algebraic Techniques," Numerical Grid Generation Techniques, NASA CP 2166, 1980.
17. Steger, J. L., Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Applications to Airfoils," AIAA Paper 77-665, AIAA 10th Fluid and Plasma Dynamics Conference, Albuquerque, New Mexico, June 1977.
18. Hindman, R. G., "Geometrically Induced Errors and their Relationship to the Form of the Governing Equations and the Treatment of Generalized Mappings," AIAA Paper 81-1008, AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, California, June 1981.
19. Smith, R. E., Kndlinski, R. A., and Everton, E. L., "A Grid Spacing Control Technique for Algebraic Grid Generation Methods," AIAA Paper 82-0226, AIAA 20th Aerospace Sciences Meeting, January 1982, Orlando, Florida.

AD P000972

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

TRANSFINITE MAPPINGS AND THEIR APPLICATION TO GRID GENERATION

WILLIAM J. GORDON AND LINDA C. THIEL
Department of Mathematical Sciences, Drexel University,
Philadelphia, PA 19104 USA

SUMMARY

The two essential ingredients of any boundary value problem are the field equations which describe the physics of the problem and a set of relations which specify the geometry of the problem domain. Mesh generators or grid generators are preprocessors which decompose the problem domain into a large number of interconnected finite elements or curvilinear finite difference stencils. A number of such techniques have been developed over the past decade to alleviate the frustration and reduce the time involved in the tedious manual subdividing of a complex-shaped region or 3-D structure into finite elements. Our purpose here is to describe how the techniques of bivariate and trivariate "blending function" interpolation, which were originally developed for and applied to geometric problems of computer-aided design of sculptured surfaces and 3-D solids, can be adapted and applied to the geometric problems of grid generation. In contrast to other techniques which require the numerical solution of complex partial differential equations (and, hence, a great deal of computing), the transfinite methods proposed herein are computationally inexpensive.

1. INTRODUCTION

Over the past decade, a number of schemes have been developed for automating the generation of finite element and curvilinear finite difference grids. Among these, the *transfinite mapping* technique of Gordon and Hall[5] has been shown to have a number of advantages (cf. [6],[7]). Some of these are:

1. Exact modeling of boundaries
2. Minimal input effort
3. Automatic node connectivity definition
4. Well-suited to interactive graphics implementation
5. Good correlation between boundary nodes and interior mesh
6. Computationally efficient
7. Easy extension to three dimensions.

We use the term "*transfinite*" to describe this class of techniques since, unlike classical methods of higher dimensional interpolation which match the *primitive function* $\vec{F}$ at a finite number of points, the transfinite methods match

$\vec{F}$ at a nondenumerable number of points. In fact, as we shall see below, transfinite mappings of the plane match $\vec{F}$ along entire curve segments, while transfinite mappings in Euclidean 3-space can match $\vec{F}$ exactly on the six faces of a curvilinear parallelpiped.

To begin, we recall the geometric interpretation of the graph of a vector-valued function of two independent variables s and t

$$\vec{F}(s,t) = [x_1(s,t), x_2(s,t), \ldots, x_n(s,t)]^T. \tag{1}$$

As the variables s and t range over a domain S in the s,t-plane R^2, $F(s,t)$ traces out a region R in Euclidean n-space, E^n. That is, $\vec{F}$ maps regions in R^2 into regions in E^n

$$\vec{F}: \; R^2 \to E^n. \tag{2}$$

For two-dimensional problems, we shall be concerned with continuous transformations $\vec{F}$ which map the unit square $S = [0,1]x[0,1]$ one-to-one onto a simply connected, bounded region R in E^2 or E^3. Such maps can be thought of as topological distortions of the planar region S onto the two-dimensional manifold R, which is either a planar region ($R \subset E^2$) or a surface embedded in 3-space ($R \subset E^3$). In either case, a one-to-one (univalent) mapping $S \to R$ is equivalent to the introduction of a *curvilinear co-ordinate system* on R. The curve of constant *generalized co-ordinate* $s = s^*$ is the image $\vec{F}(s^*,t)$ of the line $s = s^*$ in S.

Similarly, the curve $\vec{F}(s,t^*)$ is the set of all points in R with generalized co-ordinate $t = t^*$. Thus, the point $\vec{F}(s^*,t^*)$ on R is said to have generalized co-ordinates (s^*,t^*), and, since the mapping $S \to R$ is univalent, any point $P \in R$ can be uniquely referenced by its generalized co-ordinates.

If S is the unit cube $[0,1]x[0,1]x[0,1]$ in the s,t,u-parameter space R^3 and R is a bounded region in Euclidean 3-space, then a one-to-one mapping $\vec{F}$ of S onto R can be envisioned as a topological distortion of the cube into R. Such a mapping of $R^3 \to E^3$ generates a curvilinear co-ordinatization of the solid R so that each point of R may be referenced by its *generalized coordinates* (s,t,u).

For bounded, simply connected planar domains R, one could of course generate an orthogonal co-ordinatization by means of a conformal mapping of R onto a canonical region such as a square or a circle. *However, from a practical standpoint, the construction of a conformal map is equivalent to the solution of Laplace's equation and is thus contrary to the goal of computational simplicity.* In contrast, the transfinite mappings described below are relatively simple to construct and implement for a wide variety of regions, and are computationally inexpensive.

2. TWO-DIMENSIONAL REGIONS

We first consider the case in which S is the unit square $[0,1]x[0,1]$. Let us postulate the *existence* of a *primitive function* F which maps S onto R. It should be remarked that the function $\vec{F}$ is a fiction which we introduce only for notational simplicity and convenience. In practice, the only thing we are given is the geometric description of R in terms of its boundary, and it is the task of the analyst to cast this information into a form appropriate to the mapping formulas considered below. This, however, is not difficult to do and can be implemented by a computer subroutine.

Generically, $\vec{F}$: $R^2 \to E^2$ should be thought of as a continuous vector-valued function of the two independent variables s and t such that $\vec{F}$: $\partial S \to \partial R$. For example, consider the following mapping:

$$\vec{F}(s,t) = \begin{pmatrix} x(s,t) \\ y(s,t) \end{pmatrix} = \begin{pmatrix} 4st(1-s)(1-t) + (1+t/\sqrt{2}) \cos \frac{s\pi}{2} \\ (1+t/\sqrt{2}) \sin \frac{s\pi}{2} \end{pmatrix} \tag{3}$$

This maps the unit square $[0,1]x[0,1]$ onto the quarter annulus R shown in Fig.1. The perimeter of the unit square maps onto the perimeter of R, and lines of constant s and constant t map onto the curvilinear co-ordinate system illustrated. In other words, each of the curves shown in the figure is a curve of *generalized co-ordinate s = const. or t = const.*

Our problem is to construct a univalent (one-to-one) function $\vec{U}$: $S \to R$ which matches $\vec{F}$ on the boundary of S, i.e.

$$\left.\begin{aligned} \vec{U}(0,t) &= \vec{F}(0,t), \ \vec{U}(s,0) = \vec{F}(s,0) \\ \vec{U}(1,t) &= \vec{F}(1,t), \ \vec{U}(s,1) = \vec{F}(s,1) \end{aligned}\right\} \tag{4}$$

A function $\vec{U}$ which interpolates to $\vec{F}$ at a non-denumerable set of points as in (4) will be termed a *transfinite interpolant* of $\vec{F}$.

To explain the notion of transfinite mapping, we shall find it convenient to rely on the *algebraic theory* of approximation developed in [3] and [4]. In this paper, by a *projector* P we shall mean an idempotent linear operator whose domain is the linear space F of all continuous functions defined on S and whose range is a subspace of F. The above interpolation problem (4) can be viewed as a search for a projector P such that $\vec{U} = P[\vec{F}]$ is a univalent map of $S \to R$ which satisfies the desired interpolatory properties. $\vec{U}$ is termed the *projection* of $\vec{F}$ or the *image of* $\vec{F}$ *under* P.

Suppose now that ϕ_0, ϕ_1 and ψ_0, ψ_1 are four univariate functions which satisfy the *cardinality conditions*

$$\left.\begin{aligned}\phi_i(k) &= \delta_{ik} \equiv \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} && \text{for } i,k = 0,1 \\ \psi_j(\ell) &= \delta_{j\ell} && \text{for } j,\ell = 0,1 \end{aligned}\right\} \quad ; \tag{5}$$

and consider the *projectors* P_s and P_t defined by

$$\left.\begin{aligned} P_s[\vec{F}] &\equiv \phi_0(s)\vec{F}(s_0,t) + \phi_1(s)\vec{F}(s_1,t) \\ P_t[\vec{F}] &\equiv \psi_0(t)\vec{F}(s,t_0) + \psi_1(t)\vec{F}(s,t_1) \end{aligned}\right\}^* \quad . \tag{6}$$

Then, the *product* projection

$$P_sP_t[\vec{F}] = \sum_{i=0}^{1}\sum_{j=0}^{1} \phi_i(s)\psi_j(t)\vec{F}(s_i,t_j) \tag{7}$$

interpolates to $\vec{F}$ at the four corners of $[0,1]\times[0,1]$ and the Boolean sum projection

$$(P_s \oplus P_t)[\vec{F}] \equiv P_s[\vec{F}] + P_t[\vec{F}] - P_sP_t[\vec{F}] \tag{8}$$

interpolates to $\vec{F}$ on the *entire boundary* of $[0,1]\times[0,1]$. These properties of the functions (7) and (8) may be readily verified by evaluating the right-hand sides for the appropriate values of s and t and recalling the cardinality properties (5); see also [3] or [4].

The functions ϕ_i and ψ_j in the above formulae are as yet unspecified except for their values at the points $s_0 = t_0 = 0$ and $s_1 = t_1 = 1$. They are commonly referred to as *'blending functions'* and the function $\vec{U} = (P_s \oplus P_t)[\vec{F}]$ is termed a blended interpolant. The simplest choice for the blending functions in (5) is the set of four linear functions

$$\left.\begin{aligned} \phi_0(s) &= 1 - s, & \psi_0(t) &= 1 - t \\ \phi_1(s) &= s, & \psi_1(t) &= t \end{aligned}\right\} \quad . \tag{9}$$

The vector-valued bivariate function $\vec{U} \equiv (P_s \oplus P_t)[\vec{F}]$ obtained by using (8) and (9) is termed the *bilinearly blended interpolant* of $\vec{F}$, or the *transfinite bilinear interpolant* of $\vec{F}$. Explicitly, it is given by

$$\begin{aligned} \vec{U}(s,t) = (1-s)\vec{F}(0,t) &+ s\vec{F}(1,t) + (1-t)\vec{F}(s,0) + t\vec{F}(s,1) \\ &- (1-s)(1-t)\vec{F}(0,0) - (1-s)t\vec{F}(0,1) \\ &- s(1-t)\vec{F}(1,0) - st\vec{F}(1,1) \quad . \end{aligned} \tag{10}$$

This function has the properties that $\vec{U} = \vec{F}$ on the perimeter of the unit square $[0,1]\times[0,1]$. This was first demonstrated by S.A. Coons in [2].

Figure 2 illustrates the mappings induced by the projectors (6)-(8) on a

*The reader should verify that both the operators P_s and P_t are, in fact, projectors, i.e., they are linear and idempotent.

region R with blending functions given by (9). It is readily seen that $P_s[\vec{F}]$ and $P_t[\vec{F}]$ each match only two opposing boundary segments, $P_sP_t[\vec{F}]$ matches only the corners of R, but $(P_s \oplus P_t)[\vec{F}]$ does, in fact, interpolate the complete perimeter of R. The mappings $P_s[\vec{F}]$ and $P_t[\vec{F}]$ are sometimes referred to as "linear lofting", in analogy with the drafting procedure known as lofting. $P_sP_t[\vec{F}]$, which is linearly ruled in both directions, is termed bilinear, and $(P_s \oplus P_t)[\vec{F}]$ is properly termed *bilinearly blended*. The three projectors P_s, P_t and $P_s \oplus P_t$ are all *transfinite projectors* since they interpolate $\vec{F}$ at a nondenumerable number of points.

Other examples of transfinite mappings obtained using equation (10) are shown in Figures 3,4 and 5.

We can generalize the above notions in two ways: first, we may consider mappings of $R^2 \to E^n$ for general n; and secondly, we may interpolate $\vec{F}$ *not only* on the boundary of the region $R = \{(x_1,x_2,\ldots,x_n)^T = \vec{F}(s,t): 0 \le s, t \le 1\}$, but also along other 'flow lines' or constant generalized co-ordinate lines. To this end, let $0 < s_0 < s_1 < \ldots < s_M = 1$ and $0 = t_0 < t_1 < \ldots < t_N = 1$, and let $\{\phi_i(s)\}_{i=0}^M$ and $\{\psi_j(t)\}_{j=0}^N$ be functions satisfying

$$\phi_i(s_k) = \delta_{ik},\ \psi_j(t_\ell) = \delta_{j\ell} \tag{11}$$

$0 \le i, k \le M$, $0 \le j, \ell \le N$. *Now define the projections*

$$\left.\begin{aligned} P_s[\vec{F}] &\equiv \sum_{i=0}^{M} \phi_i(s)\vec{F}(s_i,t) \\ P_t[\vec{F}] &\equiv \sum_{j=0}^{N} \psi_j(t)\vec{F}(s,t_j) \end{aligned}\right\} \tag{12}$$

The *product projection*

$$P_sP_t[\vec{F}] \equiv \sum_{i=0}^{M}\sum_{j=0}^{N} \phi_i(s)\psi_j(t)\vec{F}(s_i,t_j) \tag{13}$$

interpolates to $\vec{F}$ on the finite point set $\{(s_i,t_j)\}_{i=0,j=0}^{M,N}$ while the *Boolean sum* or *transfinite interpolant*

$$(P_s \oplus P_t)[\vec{F}] \equiv P_s[\vec{F}] + P_t[\vec{F}] - P_sP_t[\vec{F}] \tag{14}$$

interpolates to $\vec{F}$ along the M+N+2 lines $s = s_i$, $0 \le i \le M$ and $t = t_j$, $0 \le j \le N$. That is, if $\vec{U}(s,t) \equiv (P_s \oplus P_t)[\vec{F}]$ then

$$\begin{aligned} &\vec{U}(s,t_j) = \vec{F}(s,t_j),\ 0 \le j \le N \\ &\vec{U}(s_i,t) = \vec{F}(s_i,t),\ 0 \le i \le M \end{aligned} \tag{15}$$

If $M = N = 1$ and $s_0 = t_0 = 0$, $s_1 = t_1 = 1$, (14) reduces to the *transfinite bilinear interpolant* (10). If $M = N = 2$ and $s_0 = t_0 = 0$, $s_1 = t_1 = 1/2$, $s_2 = t_2 = 1$, then using the blending functions

$$\begin{aligned} \phi_0(s) &= 2(s-1/2)(s-1), & \psi_0(t) &= 2(t-1/2)(t-1) \\ \phi_1(s) &= 4s(1-s), & \psi_1(t) &= 4t(1-t) \\ \phi_2(s) &= 2s(s-1/2), & \psi_2(t) &= 2t(t-1/2) \end{aligned} \tag{16}$$

in (14) yields the *biquadratically blended interpolant* of $\vec{F}$ along the *six lines* $s = 0, 1/2, 1$ and $t = 0, 1/2, 1$.

There are cases in which bilinear transfinite mapping techniques will not produce a satisfactory curvilinear grid. This typically happens on regions R which are so grossly distorted that there is either too great a variation in the size of grid elements or protions of the generalized co-ordinate curves actually map outside the region ("overspill", cf. [5]). An example is shown in Figure 6. Here, a bilinear transfinite mapping was executed with the result that some constant generalized co-ordinate lines overspilled the region.

There are basically three ways to deal with such difficulties. First, one may decompose the overly complex region into two or more simpler subregions and map each of these separately. Although this approach generally works well, there may be problems at the interfaces between subregions, since the generalized co-ordinate lines will have slope discontinuities there. One way of handling this difficulty is to employ higher degree blending functions, e.g., cubic Hermite blending functions.

A second way of attempting to achieve a satisfactory transformation is to reparametrize the boundary segments of R by, for example, introducing monotonic transformations of the independent variables s and/or t.

Another way of dealing with complex regions is to introduce *auxiliary constraints* into the transformation problem. Since the paramount objective is to obtain a one-to-one (invertible) mapping of S onto R, the analyst is perfectly free to enforce whatever additional constraints he feels will guarantee the invertibility of the mapping. In our experience, we have generally found it adequate to specify, as an auxiliary constraint, the image (i.e., the mapped position) of a single interior point of S. That is, we identify where inside R we desire to map a selected point in the interior of S. For simplicity, suppose the point in S whose image position we want to constrain is the mid-point of the square, $s = t = 1/2$. We want to force this point to map into the point in R having co-ordinates (α,β). Let $\vec{U}$ be the bilinearly blended function of (10). Then, the following transformation maps ∂S onto ∂R and maps $(1/2,1/2)$ onto the point (α,β):

$$\vec{V}(s,t) = \vec{U}(s,t) + 16s(1-s)t(1-t)\left[\begin{pmatrix}\alpha\\ \beta\end{pmatrix} - \vec{U}(1/2,1/2)\right]. \tag{17}$$

To verify this, note that along the perimeter of $[0,1]x[0,1]$ the right-hand side

reduces to just $\vec{U}(s,t)$, which maps ∂S exactly onto ∂R, as desired. For $(s,t) = (1/2,1/2)$, the right-hand side reduces to just (α,β). More generally, the point $(s,t) = (a,b)$ can be mapped into (α,β) by the formula:

$$\vec{V}(s,t) = \vec{U}(s,t) + \frac{s(1-s)t(1-t)}{a(1-a)b(1-b)}\left[\begin{pmatrix}\alpha\\ \beta\end{pmatrix} - \vec{U}(s,t)\right] . \tag{18}$$

As an example, Figure 6 shows a region R for which the bilinearly blended transformation (10) does not give an invertible mapping of S onto R. It is intuitively clear that the image of the point $(s,t) = (1/2,1/2)$ has mapped too far to the right. Therefore, we enforce the auxiliary constraint that the point $(1/2,1/2)$ in S should map onto the point $(.494,.119)$ in R. Figure 7 illustrates the result of the transformation obtained using (17).

In [5], Gordon and Hall propose the use of more general auxiliary constraints. For example, instead of just a single point, they consider mappings for which lines of constant s or t are forced to map onto preselected generalized co-ordinate curves in the domain R. Such curves may arise naturally as interfaces between subregions of R, or they may be determined by the analyst on geometric grounds. The transformation formulas appropriate to these constrained maps are given by equation (14).

If the region R is basically triangular, in contrast to quadrilateral, transfinite interpolation techniques over triangles may be more appropriate. The theory for such "trilinearly blended" methods was developed in [1]. The details of these techniques as applied to grid generation may be found in [6] and [7].

As a practical matter, the curves bounding R may not be easily represented as closed-form mathematical expressions. Nevertheless, the above results still apply if the boundary curves are represented as discrete point sets, i.e., piecewise linear curves. For a fuller discussion of *"discretized transfinite mappings"* see [6] and [7]. Surfaces in Euclidean 3-space are handled in precisely the same way as 2-D regions. All that need be done is to add the third co-ordinate functions Z(s) and Z(t) to the x and y components. A discussion of surface decomposition techniques is given in [7].

3. THREE-DIMENSIONAL SOLID STRUCTURES

The purpose of this section is to outline the extensions to 3-dimensions of the bivariate transfinite mapping techniques discussed above. To begin, we consider the following three projectors:

$$\begin{aligned} P_s[\vec{F}] &= (1-s)\vec{F}(0,t,u) + s\vec{F}(1,t,u) \\ P_t[\vec{F}] &= (1-t)\vec{F}(s,0,u) + t\vec{F}(s,1,u) \\ P_u[\vec{F}] &= (1-u)\vec{F}(s,t,0) + u\vec{F}(s,t,1) \ . \end{aligned} \tag{19}$$

In these expressions, the "primitive function" $\vec{F}$ is a vector-valued function of the three independent parameters s,t and u. As s,t and u range over the unit cube, $\vec{F}$ maps out the solid volume R. Depending upon the co-ordinate system used, the three components of $\vec{F}$ may correspond to Cartesian, spherical, cylindrical, toroidal, etc. co-ordinates. As a practical matter, the co-ordinate system employed should be that which most appropriately fits the geometry and topology of the problem domain.

Quite clearly, the three projectors in (19) correspond, respectively, to linear blending (lofting) in s,t and u. Now, however, the geometric entities on the right-hand side of the expressions are not curves, as in (6), but rather *surfaces*. For example, as t and u range over the parameter domain [0,1]x[0,1], the vector-valued function F(0,t,u) traces out a surface in Euclidean 3-space corresponding to one of the six faces of the solid volume R.

It is useful to consider the pairwise products of the above three projectors. For instance, the product of the first and the second is

$$P_s(P_t[\vec{F}]) = (1-s)(1-t)\vec{F}(0,0,u) + (1-s)t\vec{F}(0,1,u) + s(1-t)\vec{F}(1,0,u) + st\vec{F}(1,1,u) \tag{20}$$

The right-hand side of this expression contains four expressions which refer to the *edges* of the object under consideration. By evaluating the expression along the four edges (s,t) = (0,0),(0,1),(1,0) and (1,1), it can be verified that the trivariate function $P_sP_t[\vec{F}]$ does, in fact, match $\vec{F}$ along these edges. (It may be easily demonstrated that $P_sP_t[\vec{F}] = P_tP_s[\vec{F}]$, i.e., the projectors commute, just as in the bivariate case.)

We have seen that the projectors P_s,P_t and P_u each interpolate the two opposing faces of the solid volume described by $\vec{F}(s,t,u)$, and that products of pairs of these projectors interpolate to the *edges* of the solid. If we take the product of all three of the projectors in (19), we obtain the expression:

$$\begin{aligned} P_sP_tP_u[\vec{F}] = {} & (1-s)(1-t)(1-u)\vec{F}(0,0,0) + (1-s)(1-t)u\vec{F}(0,0,1) \\ & + (1-s)t(1-u)\vec{F}(0,1,0) + (1-s)tu\vec{F}(0,1,1) \\ & + s(1-t)(1-u)\vec{F}(1,0,0) + s(1-t)u\vec{F}(1,0,1) \\ & + st(1-u)\vec{F}(1,1,0) + stu\vec{F}(1,1,1). \end{aligned} \tag{21}$$

The right-hand side of this expression contains values of $\vec{F}$ which refer to the eight *corners* of the region R. It does, in fact, interpolate to these eight corners. Since (21) is linear in each of the three parameters s,t and u, it is termed a *trilinear interpolant*. The graph of the trilinear interpolant is a six-sided polyhedron which passes through the vertices of R; it is simply the 3-D

generalization of a quadrilateral in Euclidean 2-space.

In the 2-dimensional case, one starts with the two projectors P_s and P_t of (6) and, by combination, generates a total of four; namely, P_s, P_t, P_sP_t and $P_s \oplus P_t$. In three dimensions, the situation is much more complex and the variety of possible projectors much richer. In [3] and [4], it is shown that under the two binarary operations of operator multiplication and Boolean ($\oplus$) addition, any collection of commutative projectors forms a *distributive lattice*. Space does not permit going into the details of this theory, but we can illustrate some of the results in the trivariate case. Without proof, we state that there are 21 distinct projectors which can be formed by multiplication and $\oplus$ addition of the three projectors P_s, P_t and P_u. In addition to those displayed above, the following are *examples:*

$$
\begin{aligned}
&P_s \oplus P_t = P_s + P_t - P_sP_t \\
&P_s \oplus P_tP_u = P_s + P_tP_u - P_sP_tP_u \\
&P_sP_t \oplus P_u = P_sP_t + P_u - P_sP_tP_u \\
&P_s \oplus P_t \oplus P_tP_u = P_s + P_t - P_sP_t \\
&P_sP_t \oplus P_tP_u \oplus P_uP_s = P_sP_t + P_tP_u + P_uP_s - 2P_sP_tP_u \\
&P_s \oplus P_t \oplus P_u = P_s + P_t + P_u - P_sP_t - P_tP_u - P_uP_s + P_sP_tP_u
\end{aligned}
\qquad (22)
$$

One should note that, because of the *idempotency* and linearity of the projectors, much cancellation occurs. For instance, one has $P_s \oplus P_sP_t = P_s + P_sP_t - P_sP_sP_t = P_s$, which means that nothing is achieved by taking the $\oplus$ sum of P_s and P_sP_t. This is because the interpolation properties of P_sP_t are a subset of those of the projector P_s.

An aspect of the theory developed in [3] and [4] is that there is an *isomorphism* between the distributive lattice of projectors and the associated distributive lattice of their precision sets. In other words, if we know the expression for a certain projector, then we can determine the point set on which it interpolates by replacing operator multiplication by set intersection and $\oplus$ addition by set union. For example, the *precision set* (set of points on which it interpolates) of P_s consists of the two faces of R defined by $s = 0$ and $s = 1$; and similarly for P_t and P_u. Thus, it follows from the isomorphism that the precision sets of the projectors in (22) are, respectively, given by the following expressions in which S_s, S_t and S_u are the precision sets of P_s, P_t and P_u:

$$
\begin{aligned}
&S_s \cup S_t \\
&S_s \cup (S_t \cap S_u) \\
&(S_s \cap S_t) \cup S_u \\
&S_s \cup S_t \cup (S_t \cap S_u) \\
&(S_s \cap S_t) \cup (S_t \cap S_u) \cup (S_u \cap S_s) \\
&S_s \cup S_t \cup S_u
\end{aligned}
\tag{23}
$$

If one thinks of the precision sets, it is obvious that the weakest ("*algebraically minimal*") of all projectors is the triple product $P_s P_t P_u$ and the "*algebraically maximal*" projector is the Boolean sum of all three: $P_s \bullet P_t \bullet P_u$. All other possible projectors are algebraically in-between these two.

Which of this miriad of 3-D projectors one uses in practice is a matter of what data is given; or, more precisely, *where* the data is given. In other words, one is given the precision set and must use the isomorphism "backward" to infer the appropriate projector (i.e., interpolation formula). For instance, if one does, in fact, know the exact shapes of all six of the bounding surfaces of R, then the appropriate mapping equation is the full-blown expression $(P_s \bullet P_t \bullet P_u)[\vec{F}]$, which explicitly involves all faces, all edges and all corners. At the other extreme, one may only know the co-ordinates of the eight corners of R. In this case, one would use the transformation equation (21).

In practice, the situation is usually somewhere in-between, i.e., the data is seldom as complete as required by the maximal projector $P_s \bullet P_t \bullet P_u$ or as scant as only the eight vertices needed in $P_s P_t P_u[\vec{F}]$. The examples given below assume that the function $\vec{F}$ is known (i.e., data is given) on the 12 edges of R. In this case, the revelant transformation equation or mapping formula is:

$$
\begin{aligned}
&(P_s P_t \bullet P_t P_u \bullet P_u P_s)[\vec{F}] \\
&\quad = (1-s)(1-t)\vec{F}(0,0,u) + (1-s)t\vec{F}(0,1,u) \\
&\quad + s(1-t)\vec{F}(1,0,u) + st\vec{F}(1,1,u) \\
&\quad + (1-t)(1-u)\vec{F}(s,0,0) + (1-t)u\vec{F}(s,0,1) \\
&\quad + t(1-u)\vec{F}(s,1,0) + tu\vec{F}(s,1,1) \\
&\quad + (1-s)(1-u)\vec{F}(0,t,0) + (1-s)u\vec{F}(0,t,1) \\
&\quad + s(1-u)\vec{F}(1,t,0) + su\vec{F}(1,t,1) \\
&\quad - 2[(1-s)(1-t)(1-u)\vec{F}(0,0,0) + (1-s)(1-t)u\vec{F}(0,0,1) \\
&\quad + (1-s)t(1-u)\vec{F}(0,1,0) + (1-s)tu\vec{F}(0,1,1) \\
&\quad + s(1-t)(1-u)\vec{F}(1,0,0) + s(1-t)u\vec{F}(1,0,1) \\
&\quad + st(1-u)\vec{F}(1,1,0) + stu\vec{F}(1,1,1)].
\end{aligned}
\tag{24}
$$

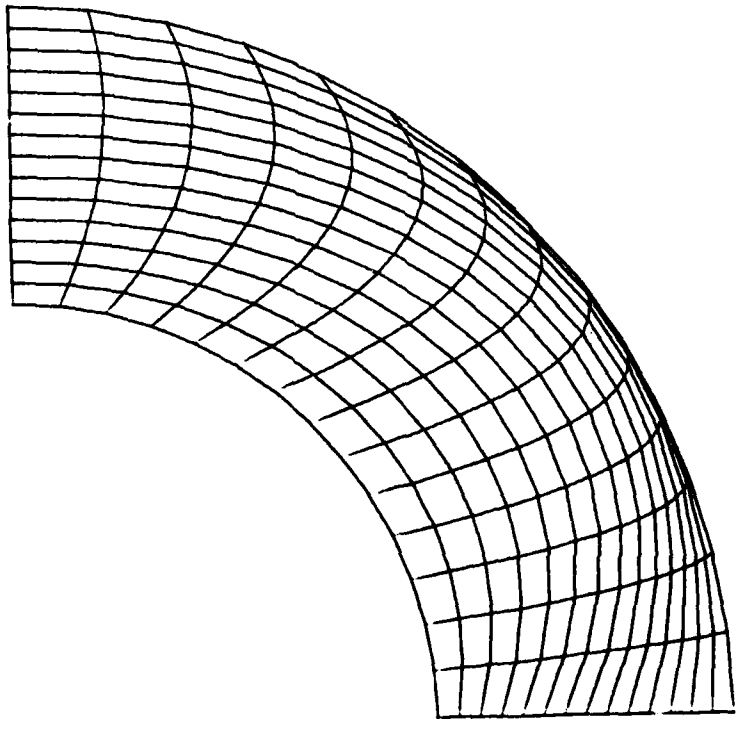
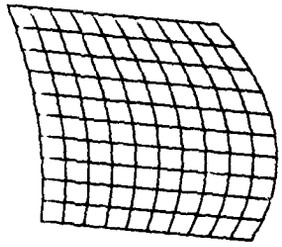


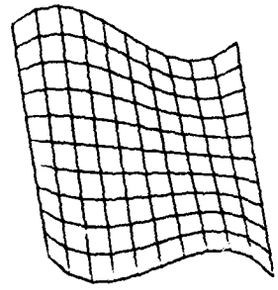
Figure 1

The curvilinear co-ordinate system induced by the following mapping:

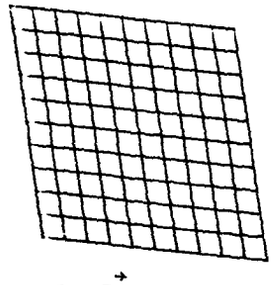
$$\vec{F}(s,t) = \begin{pmatrix} x(s,t) \\ y(s,t) \end{pmatrix} = \begin{pmatrix} 4st(1-s)(1-t)+(1+t/\sqrt{2})\cos\frac{s\pi}{2} \\ (1+t/\sqrt{2})\sin\frac{s\pi}{2} \end{pmatrix}$$



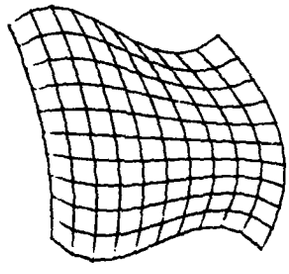
$P_s[\vec{F}]$



$P_t[\vec{F}]$



$P_s P_t[\vec{F}]$



$(P_s \bullet P_t)[\vec{F}]$

Figure 2

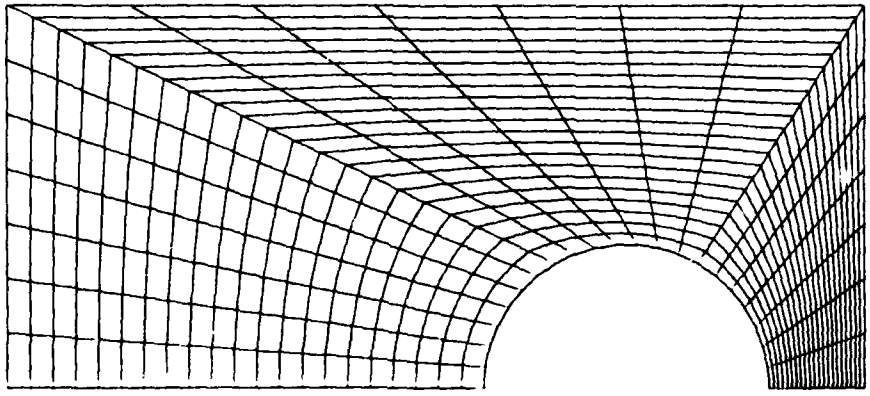


Figure 3

The curvilinear co-ordinate system generated by the following parametrization:

$$\vec{F}(0,t) = \begin{bmatrix} -2.5t + .5 \\ -2 \end{bmatrix}, \quad \vec{F}(1,t) = \begin{bmatrix} 0.5t + 2 \\ -2 \end{bmatrix}$$

$$\vec{F}(s,0) = \begin{bmatrix} 1.25 - .75 \cos(\pi s) \\ -2 + .75 \sin(\pi s) \end{bmatrix}$$

$$\vec{F}(s,1) = \begin{bmatrix} \begin{cases} -2, & 0.0 \le s < .33 \\ -6.5 + 13.5s, & .33 \le s < .66 \\ 2.5, & .66 \le s \le 1 \end{cases} \\ \begin{cases} -2 + 6s, & 0.0 \le s < .33 \\ 0.0, & .33 \le s < .66 \\ 4 - 6s, & .66 \le s \le 1 \end{cases} \end{bmatrix}.$$

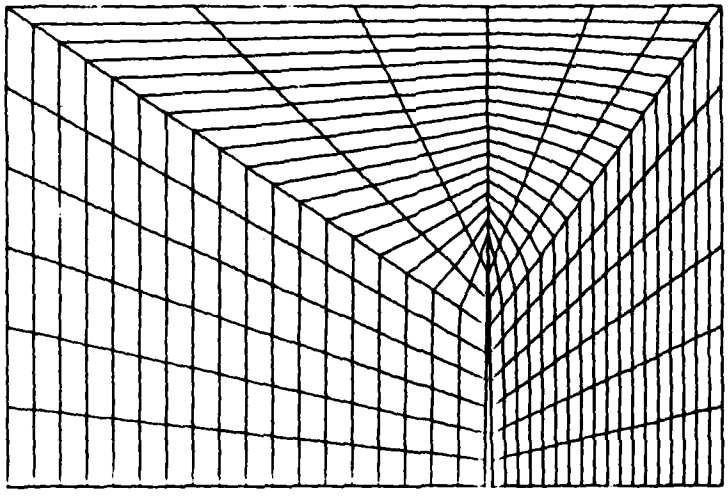


Figure 4

The transfinite map of a region with a crack via the following parametrization:

$$\vec{F}(0,t) = \begin{bmatrix} 1 - 3t \\ -3 \end{bmatrix}, \quad \vec{F}(1,t) = \begin{bmatrix} 1.05 + 1.45t \\ -3 \end{bmatrix}$$

$$\vec{F}(s,0) = \begin{bmatrix} 1 + .05s \\ \begin{cases} -3 + 3s, & 0.0 \le s \le 0.5 \\ -3s, & 0.5 \le s \le 1 \end{cases} \end{bmatrix},$$

$$\vec{F}(s,1) = \begin{bmatrix} \begin{cases} -2, & 0.0 \le s < .33 \\ -12.7 + 41.4s - 27.9s^2, & .33 \le s < .66 \\ 2.5, & .66 \le s \le 1 \end{cases} \\ \begin{cases} -3 + 9s & 0.0 \le s < .33 \\ 0.0 & .33 \le s < .66 \\ 6 - 9s & .66 \le s \le 1 \end{cases} \end{bmatrix}.$$

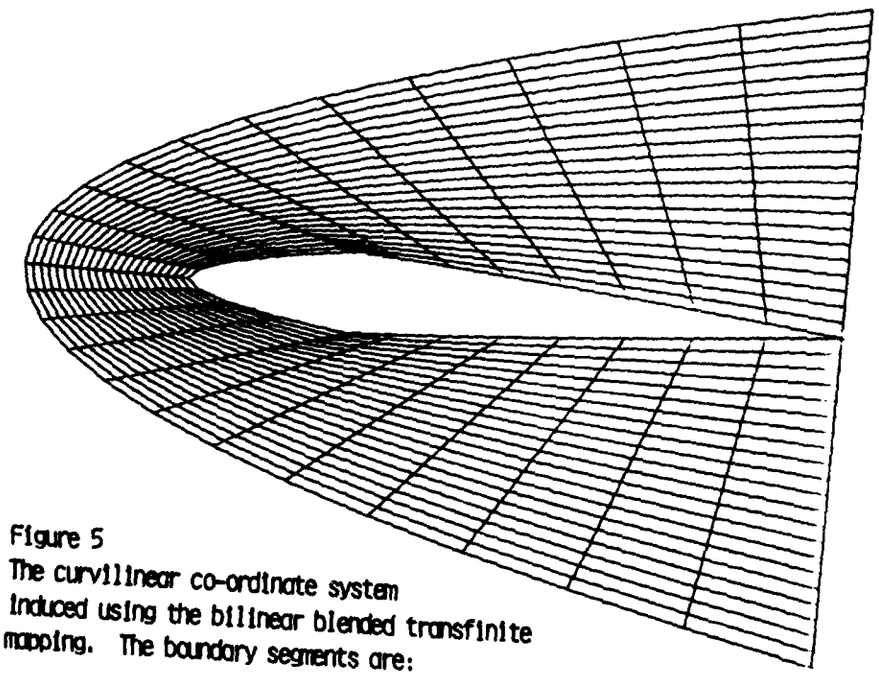


Figure 5
The curvilinear co-ordinate system induced using the bilinear blended transfinite mapping. The boundary segments are:

$$\vec{F}(0,t) = \begin{bmatrix} 4 \\ -4t \end{bmatrix}, \quad \vec{F}(1,t) = \begin{bmatrix} 4 \\ 4t \end{bmatrix},$$

$$\vec{F}(s,0) = \begin{bmatrix} \begin{cases} 4 - 11.5s + 7.5s^2, & 0.0 \le s < .33 \\ 9 - 36s + 36s^2, & .33 \le s < .66 \\ -3.5s + 7.5s^2, & .66 \le s \le 1 \end{cases} \\ \begin{cases} -1.125s + 1.125s^2, & 0.0 \le s < .33 \\ -.75 + 1.5s & .33 \le s < .66 \\ 1.125s - 1.125s^2 & .66 \le s \le 1 \end{cases} \end{bmatrix},$$

$$\vec{F}(s,1) = \begin{bmatrix} 4 - 20s + 20s^2 \\ -2 + 4s \end{bmatrix}.$$

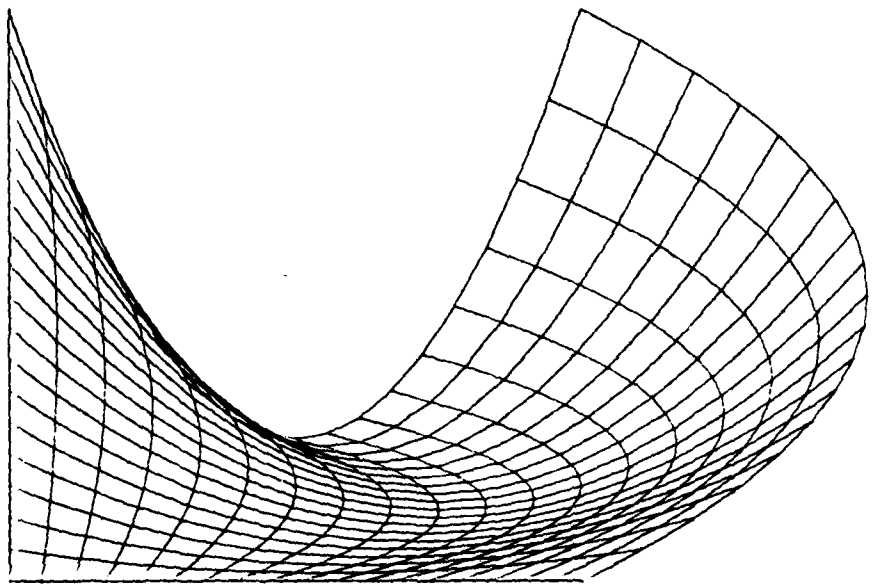


Figure 6
Bilinear blending here yields a non-univalent map. The parametrization of the boundary is:

$$\vec{F}(0,t) = \begin{bmatrix} 0 \\ t \end{bmatrix}, \quad \vec{F}(1,t) = \begin{bmatrix} 1 + 2t - 2t^2 \\ t \end{bmatrix},$$

$$\vec{F}(s,0) = \begin{bmatrix} s \\ 0 \end{bmatrix}, \quad \vec{F}(s,1) = \begin{bmatrix} s \\ 1 - 3s + 3s^2 \end{bmatrix}.$$

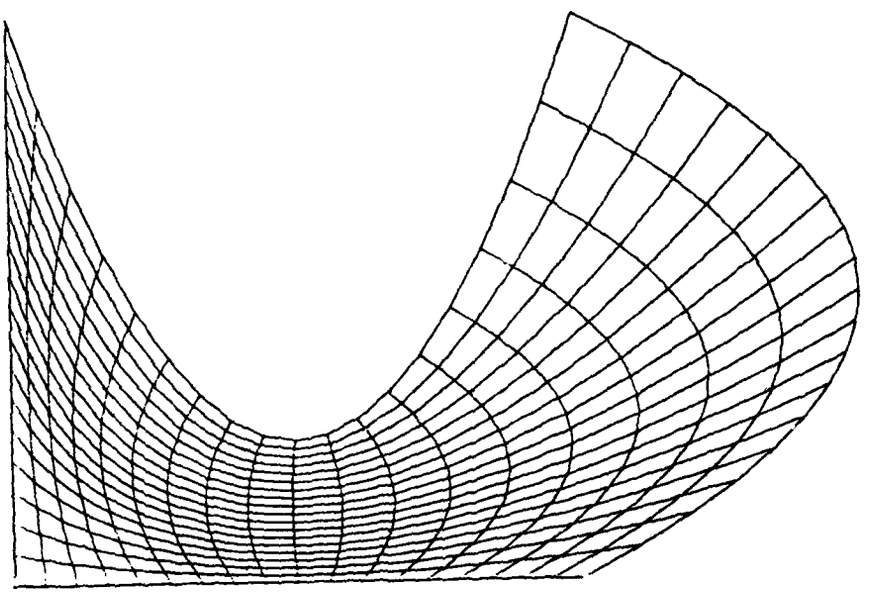


Figure 7
A curvilinear co-ordinate system with no "overspill" is achieved via (17). Here, the point (1/2,1/2) in the s,t-plane is mapped onto the point (.494, .119) in the x,y-plane.

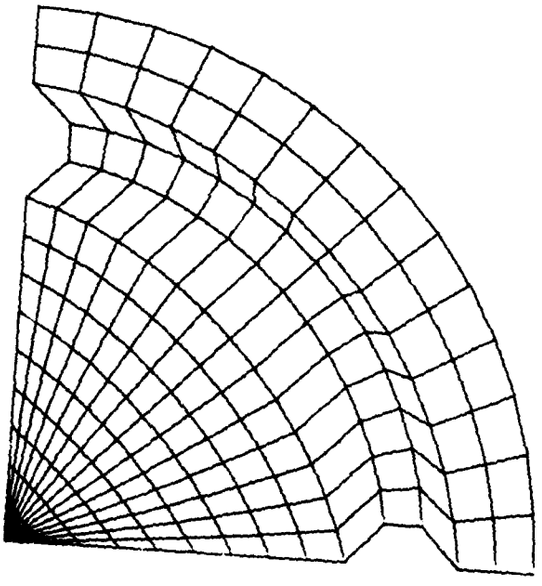


Figure 8

This mapping was generated via (18) by mapping the point (11/14,1/2) in the s,t-plane onto the point (.326,.334) in the x,y-plane. The boundary segments are:

$$\vec{F}(0,t) = \begin{bmatrix} -3 \\ -3 \end{bmatrix}, \quad F(1,t) = \begin{bmatrix} -3+6\cos(\pi t/2) \\ -3+6\sin(\pi t/2) \end{bmatrix}$$

$$\vec{F}(s,0) = \begin{bmatrix} 6s-3 \\ -3+\sqrt{.25-(6s-4.5)^2}, & .67 \le s \le .83 \\ -3, & \text{otherwise} \end{bmatrix}$$

$$\vec{F}(s,1) = \begin{bmatrix} -3+\sqrt{.25-(6s-4.5)^2}, & .67 \le s \le .83 \\ -3, & \text{otherwise} \\ 6s-3 \end{bmatrix}$$

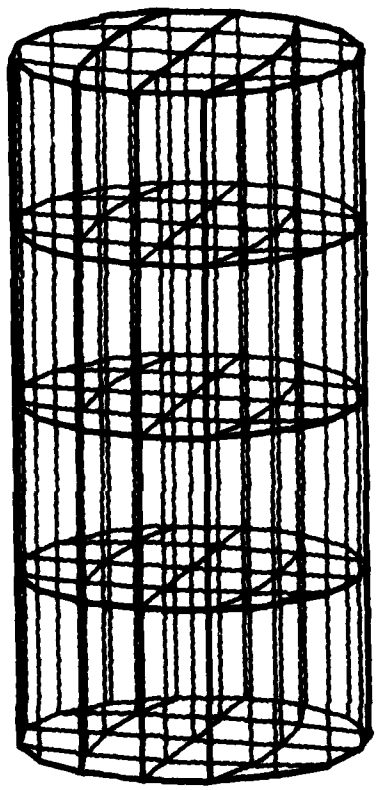


Figure 9

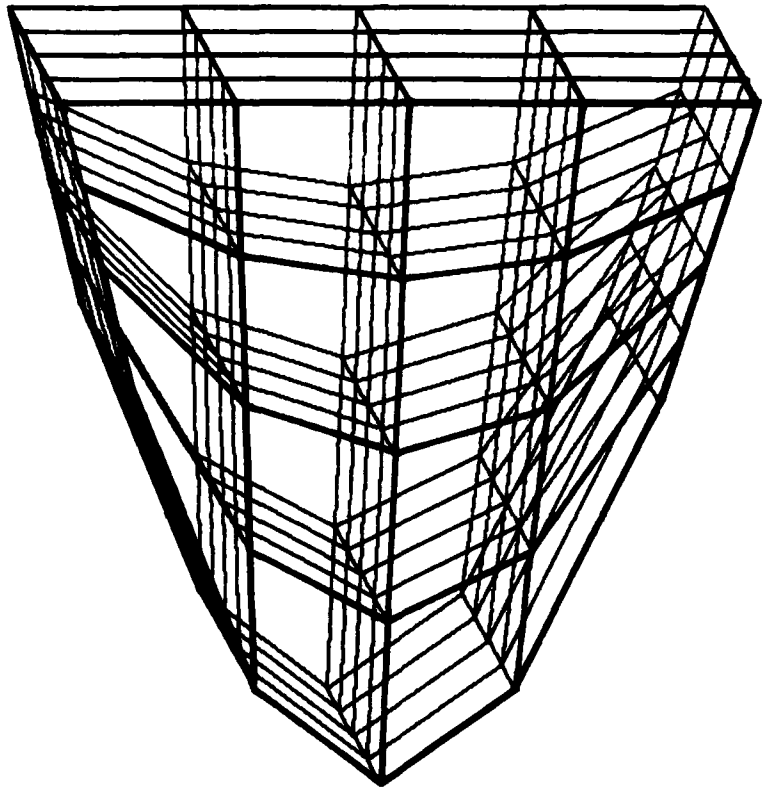


Figure 10

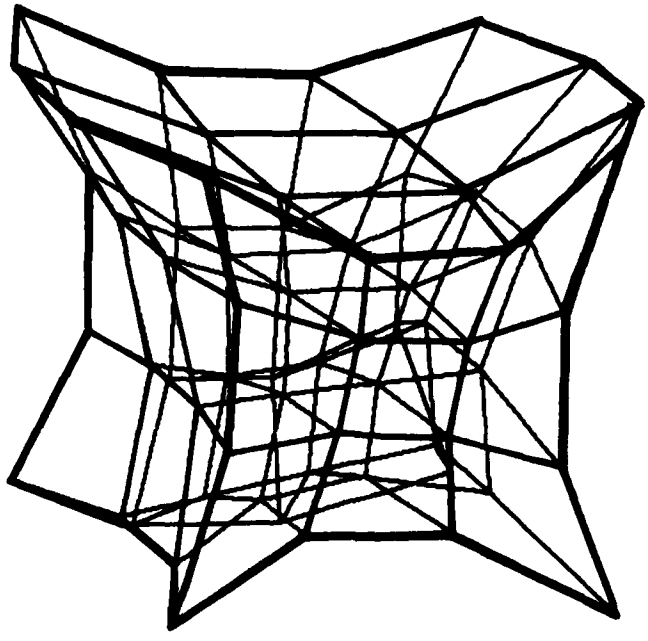


Figure 11

ACKNOWLEDGEMENTS

This work was supported in part by the U.S. Office of Naval Research and the Air Force Office of Scientific Research under contract #N00014-80-C0176 to Drexel University.

The authors wish to thank Debra DeLise-Hughes for her assistance in the preparation and typing of this paper, and Andrew Galardi for his help with some of the figures.

REFERENCES

1. G. Birkhoff, W.J. Gordon, R.E. Barnhill, "Smooth Interpolation in Triangles", J. Approx. Theory, 8, 114-128 (1973).
2. S.A. Coons, "Surfaces for Computer-Aided Design of Space Forms", Project MAC, Design Div., Dept. Mech. Engng., Mass. Inst. Tech. (1964). Available from: Clearinghouse for Federal Scientific-Technical Information, National Bureau of Standards, Springfield, VA, U.S.A.
3. W.J. Gordon, "Blending-Function Methods for Bivariate and Multivariate Interpolation and Approximation", SIAM J. Numer. Anal. 8, 158-177 (1971).
4. W.J. Gordon, "Distributive Lattices and the Approximation of Multivariate Functions", in Approximations with Special Emphasis on Spline Functions, Academic Press, New York, 1969, pp. 223-277.
5. W.J. Gordon, C.A. Hall, "Construction of Curvilinear Coordinate Systems and Applications to Mesh Generation", Inst. J. Num. Meth. Engng., 7, 461-477 (1973).
6. R.B. Haber, M.S. Shepard, J.F. Abel, R.H. Gallagher, and D.P. Greenberg, "A General Two-Dimensional Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings", Int. J. Num. Meth. Engng., 17 (7), 1015-1044 (1981).
7. R. Haber, J.F. Abel, "Discrete Transfinite Mappings for the Description and Meshing of Three-Dimensional Surfaces Using Interactive Computer Graphics", Int. J. Num. Meth. Engng., 18, 41-66 (1982).

AD P000973

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ORTHOGONAL GRID GENERATION

Peter R. Eiseman
Department of Applied Physics and Nuclear Engineering, Columbia University, New York, New York 10027

INTRODUCTION

Coordinate transformations have been a key element in most numerical solutions to partial differential equations when either the solution or the region exhibits some geometric complexity. The grids from the transformations are chosen to represent the region boundaries with coordinate curves or surfaces and to adequately resolve the solution by clustering points, curves, or surfaces. The coordinates can be used separately or in a composite fashion to appropriately discretize a configuration with a topological complication in addition to that of basic geometry. In either case, the discretization is well structured since the space of computational variables is rectilinear. Solution procedures developed in a Cartesian setting can then be applied along with the corresponding simplicity in the organization of computational data.

To gain the clear advantages of a well-structured discretization, the original partial differential equations must be expressed relative to the grid. The result is usually an increase in the complexity of the equations. This increase is greatest with nonorthogonal coordinates, is fairly mild with orthogonality and is the least with conformal transformations.

To choose between the various types of coordinates, ~~we~~ must ~~first consider~~ be considered which constraints are needed for a given problem. The fundamental constraint for a general region is its boundary geometry. When the coordinates match the boundary, the need for boundary interpolation disappears and the grid is also aligned with the desired solution near the boundary. Without any further requirement in the two-dimensional case, conformal systems are usually the best. In addition to boundary geometry, however, the pointwise distribution along the boundary is often required as a further constraint. The distribution is a boundary coordinate system or systems, which together with the geometry forms a complete boundary representation. When the representation is arbitrarily prescribed, conformal transformations are not applicable because of analytic continuation. As the next simplest case, orthogonal coordinates are preferable. In two-dimensions they are generally applicable on both planes and curved surfaces. In three-dimensional regions, orthogonal systems are severely restricted and are not generally applicable. The best that can be done in the general content is to bound such regions with orthogonal systems so that full

orthogonality can be specified at the boundaries. Further boundary constraints can also be imposed with specified derivatives so that rates of entry or exit from a region can be given. In any dimension, the capability to create a smoothly assembled composite mesh for topologically complex configurations would be achieved. In addition to the various boundary constraints, significant advantages can be obtained under the constraint that points, curves, or surfaces be clustered in some location within the region. The purpose is usually to more fully resolve the numerical solution of a given problem with a fixed number of mesh points. Additional advantages can also be achieved with the constraint that a certain desireable mesh structure be smoothly embedded within the region. Under all of these conditions, orthogonal coordinates can usually be obtained but are restricted to two-dimensional situations.

To impose any of the constraints for any desired purpose, the element of control must be established within the mesh generation process. The degree of available control depends upon the type of coordinates and is quite substantial for two-dimensional orthogonal systems. Once established, the mesh controls can be applied directly in advance or with a prescribed evolution to follow the expected path of solution behavior. Moreover, they can also be applied from the evolutionary solution properties to produce algorithms that are dynamically adapted to the solution.

Due to the possible control, the inherent simplicity, and the improved accuracy in numerical simulations, we have been encouraged to develop methods for generating orthogonal coordinate systems on planes and on curved surfaces. To unify our discussion of the various methods, the surface metric will be introduced with the understanding that Euclidian planes are included as special cases. The required geometric background is included here in a brief preliminary section so that our development is essentially self-contained. The metric is also important since it contains the basic information about any coordinate system and is used to relate the given partial differential equations to the coordinates. In terms of the metric, we can easily distinguish between conformal, orthogonal, and nonorthogonal systems. For orthogonal systems, the cross terms in the metric vanish; for conformal, the remaining terms are all equal. Any deviation from the orthogonal metric is a direct measure of the corresponding deviation of the coordinates from orthogonality. Since coordinate orthogonality enters a given system of partial differential equations only through the metric, the accurate rendition of orthogonality depends solely upon the metric. When the metric for a coordinate grid is evaluated analytically, the accuracy comes from the underlying transformation. With a numerical

evaluation, it comes solely from the grid which then motivates us to consider grids which are numerically orthogonal. The choice between analytic and numerical evaluations depends upon the application. As an example, the use of conservation law form would bias us towards numerical evaluation and hence numerical orthogonality.

The various methods of orthogonal coordinate generation can be reasonably compared on the basis that a maximal amount of control can be exercised at the least possible cost. The cost is in terms of computational and human efficiency. Computational efficiency depends upon speed and storage; human efficiency, upon robustness and the number of parameters that must be specified. The amount of control is in terms of the number of constraints that can be applied. With additional constraints, the number of specified parameters must increase; but in the interest of human efficiency, must not increase beyond what is required for the exercise of control. In most cases, the element of control is the most important factor since it yields the capability to obtain grids that meet desireable specifications. The cost usually becomes important only when the coordinates must be frequently generated as would occur when an evolutionary process must be followed.

The methods for orthogonal coordinate generation can be split into the categories of orthogonal trajectory methods, field solutions, and marching techniques. Orthogonal trajectory methods yield a great amount of control at a low cost and are currently the best available methods. The trajectories are generated with respect to a given family of curves which is usually taken from a nonorthogonal transformation. When the nonorthogonal transformation is chosen to match a complete boundary representation on all four boundaries of a region and to have orthogonality at two opposing boundaries, the trajectories can be generated in parallel to the boundaries that have orthogonality. This yields an orthogonal system that conforms to the geometry of all boundaries and has pointwise distributions specified on three out of four of them. In addition, the clustering controls on the chosen family of curves from the nonorthogonal transformation are directly carried over into the orthogonal system. To improve upon this situation, we should first be able to give the fourth boundary a pointwise distribution. A direct extension of orthogonal trajectories may be applicable here in the spirit of a global iterative cycle in which the fixed family of curves is continually adjusted until the desired pointwise distribution is attained. A more direct and potentially advantageous approach is to consider field solutions where the coordinates are generated by elliptic partial differential equations with Dirichlet boundary conditions. In addition

to the specification of pointwise distributions on up to all four boundaries, there is also the important potential to create general clustering controls which would improve upon the clustering from orthogonal trajectory methods. In cases where there is only one boundary that is important, marching methods have been developed to march away from the boundary. In comparison with orthogonal trajectory methods, the need to specify a family of curves is removed and is usually replaced by a hopefully simpler prescription that may demand less of the user. In all of the methods, the metric plays a fundamental role, and its use in field and marching techniques is a key to success. Moreover, these techniques may be considered together on the basis of various conditions imposed upon the metric.

GEOMETRY

Curves, Surfaces, Tangents, and Metrics

To readily define geometric objects in up to three dimensions, we will consider a fixed three-dimensional Euclidian space where points are described by the position vector

$$\vec{c} = (c_1, c_2, c_3) \tag{1}$$

with Cartesian components c_i. A general curve or surface is then defined by

$$\vec{c}(\vec{x}) = (c_1(\vec{x}), c_2(\vec{x}), \vec{c}_3(\vec{x})) \tag{2}$$

where the coordinate vector $\vec{x}$ is given by $\vec{x} = x_1$ for curves and $\vec{x} = (x_1, x_2)$ for surfaces. A curve that is contained in a surface is given by $\vec{x} = \vec{x}(t) = (x_1(t), x_2(t))$ to yield a position vector

$$\vec{c}(t) = (c_1(\vec{x}(t)), c_2(\vec{x}(t)), c_3(\vec{x}(t)) \tag{3}$$

for parameter t. In particular, coordinate curves on the surface are given when either $x_1 = t$ and x_2 is constant or x_1 is constant and $x_2 = t$ for the respective x_1 and x_2 directions. For the curves and surfaces, sufficient smoothness will be assumed so that all derivatives exist and are continuous.

For a surface curve section about $\vec{c}(t_o)$, the increment $\Delta\vec{c} = \vec{c}(t) - \vec{c}(t_o)$ is given by the secant vector from $\vec{c}(t_o)$ to $\vec{c}(t)$ which defines the approximate rate of change $\Delta\vec{c}/\Delta t$ in the same direction. An illustration is given in Fig. 1.

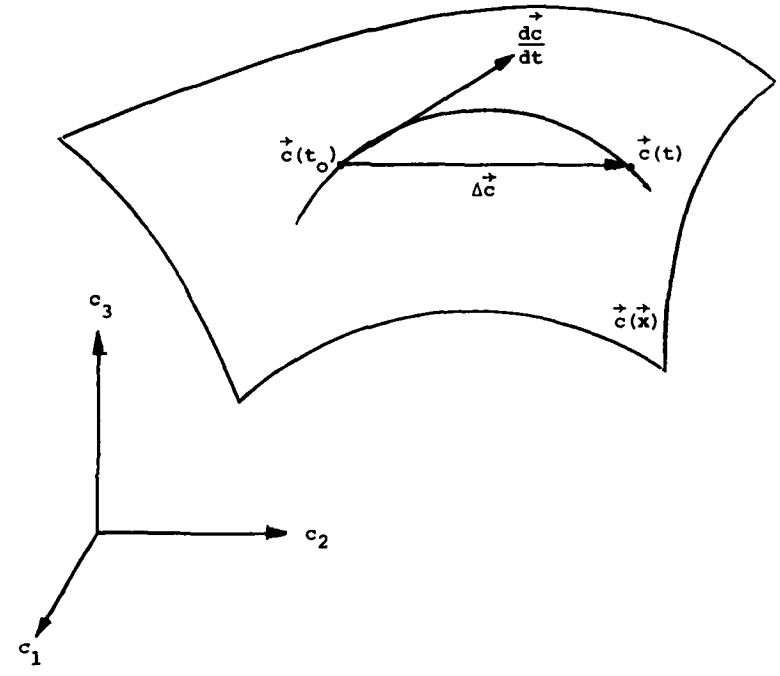


Fig. 1. The natural tangent to a surface curve.

In the figure, the indicated limit

$$\left.\frac{d\vec{c}}{dt}\right|_{t = t_o} = \underset{t \to t_o}{\text{Lim}} \frac{\vec{c}(t) - \vec{c}(t_o)}{t - t_o} \tag{4}$$

is clearly (assuming smoothness) a tangent vector to the surface curve at the point $\vec{c}(t_o)$ and with a magnitude equal to the rate of travel along the curve.

When the surface curves are coordinate curves, we get the tangents

$$\vec{e}_1 = \frac{\partial\vec{c}}{\partial x_1}$$

and (5)

$$\vec{e}_2 = \frac{\partial\vec{c}}{\partial x_2}$$

which we call the natural tangents for the respective coordinate directions. Now returning to the secant $\Delta\vec{c}$, we see that its length is an approximation to

curve arc length. Then, in the limit, we get the differential element of arc length ds from

$$\begin{aligned}(ds)^2 &= d\vec{c} \cdot d\vec{c} \\ &= \left(\frac{\partial \vec{c}}{\partial x_i}\, dx_i\right) \cdot \left(\frac{\partial \vec{c}}{\partial x_j}\, dx_j\right) \\ &= (\vec{e}_i \cdot \vec{e}_j)\, dx_i\, dx_j \\ &= g_{ij}\, dx_i\, dx_j \\ &= g_{11}\,(dx_1)^2 + 2\, g_{12}\, dx_1\, dx_2 + g_{22}\,(dx_2)^2\end{aligned} \tag{6}$$

where the repeated indices represent sums from 1 to 2. The implied summation is a standard notation that we will continue to use. The expansion for ds is just a rule for distance measurement along the surface with respect to its coordinate representation. The functions $g_{ij} = \vec{e}_i \cdot \vec{e}_j$ define the rule which is called the metric. For distance measurement $s = s_i$ along a coordinate curve for x_i, the only nonzero coordinate differential is dx_i and the metric becomes $ds_i = \sqrt{g_{ii}}\ dx_i$ or simply $ds_i = \|\vec{e}_i\|\, dx_i$ in correspondence with the view of the secant approximation. The angle θ between the coordinate curves is also obtained from the metric since

$$\cos\theta = \left(\frac{\vec{e}_1}{\|\vec{e}_1\|}\right) \cdot \left(\frac{\vec{e}_2}{\|\vec{e}_2\|}\right) = \frac{g_{12}}{\sqrt{g_{11}\, g_{22}}} \tag{7}$$

The coordinates are orthogonal when $\cos\theta$ vanishes which occurs when $g_{12} = 0$. Consequently, the metric for two-dimensional orthogonal coordinates is of the form

$$(ds)^2 = g_{11}\,(dx_1)^2 + g_{22}\,(dx_2)^2 \tag{8}$$

where the diagonal coefficients need not be equal. Here, a square element defined by $dx_1 = dx_2 = dx$ is mapped onto a rectangular surface element with sides $ds_1 = \sqrt{g_{11}}\ dx$ and $ds_2 = \sqrt{g_{22}}\ dx$. The element aspect ratio is just

$$\frac{ds_2}{ds_1} = \sqrt{\frac{g_{22}}{g_{11}}} \tag{9}$$

When the aspect ratio is unity, differential squares map to differential squares. The orthogonal metric then has equal diagonal entries and the coordinates are called isothermal parameters or simply the system is called conformal.

In the orthogonal case, squares of area $(dx)^2$ are mapped to rectangles of area $ds_1\ ds_2 = \sqrt{g_{11}\ g_{22}}\ (dx)^2$. The Jacobian J of the transformation is easily identified as $\sqrt{g_{11}\ g_{22}}$ which in the more specific conformal case is just $J = g_{11} = g_{22}$. In the more general nonorthogonal case, $J = \sqrt{g}$ where $g = \det(g_{ij}) = g_{11}\ g_{22} - g_{12}^2$.

Intrinsic Constraints

In general, we have seen that the metric is the rule for distance, angle, and area measurements within a surface; as a consequence, it is intrinsic to the surface. Moreover, any quantity that depends only upon the metric is also intrinsic. As a rule for measurements with a surface, the metric is independent of any specific coordinate representation. To change the representation from (x_1, x_2) to (y_1, y_2) coordinates we have

$$(ds)^2 = g_{ij}\ dx_i\ dx_j = g_{ij}\ \frac{\partial x_i}{\partial y_\ell}\ dy_\ell\ \ \frac{\partial x_j}{\partial y_k}\ dy_k = \bar{g}_{\ell k}\ dy_\ell\ dy_k \tag{10a}$$

so that

$$\bar{g}_{\ell k} = g_{ij}\ \frac{\partial x_i}{\partial y_\ell}\ \frac{\partial x_j}{\partial y_k} \tag{10b}$$

are the coffecients attached to (y_1, y_2). In general, any intrinsic quantity is independent of coordinates.

The Gaussian curvature [1] of a surface is a scalar quantity and was shown by Gauss to be intrinsic. He was very proud of this result and called it "Theorema egregium" which means "extraordinary theorem." Explicitly, the Gaussian curvature K is given by

$$K = \frac{1}{4\sqrt{g}} \begin{vmatrix} g_{11} & \frac{\partial g_{11}}{\partial x_1} & \frac{\partial g_{11}}{\partial x_2} \\ g_{12} & \frac{\partial g_{12}}{\partial x_1} & \frac{\partial g_{12}}{\partial x_2} \\ g_{22} & \frac{\partial g_{22}}{\partial x_1} & \frac{\partial g_{22}}{\partial x_2} \end{vmatrix} - \frac{1}{2\sqrt{g}} \left\{ \frac{\partial}{\partial x_2} \left[\frac{\frac{\partial g_{11}}{\partial x_2} - \frac{\partial g_{12}}{\partial x_1}}{\sqrt{g}} \right] - \frac{\partial}{\partial x_1} \left[\frac{\frac{\partial g_{12}}{\partial x_2} - \frac{\partial g_{22}}{\partial x_1}}{\sqrt{g}} \right] \right\} \tag{11}$$

With orthogonal coordinates, $g_{12} = 0$ and

$$K = \frac{1}{2\sqrt{g}} \left\{ \frac{\partial}{\partial x_1}\left(\frac{1}{\sqrt{g}} \frac{\partial g_{22}}{\partial x_1}\right) + \frac{\partial}{\partial x_2} \left(\frac{1}{\sqrt{g}} \frac{\partial g_{11}}{\partial x_2}\right)\right\} \tag{12a}$$

or equivalently,

$$K = \frac{1}{2\sqrt{g_{11}\, g_{22}}} \left\{ \frac{\partial^2}{\partial x_1^{\,2}} (\log g_{22}) + \frac{\partial^2}{\partial x_2^{\,2}} (\log g_{11}) \right\} \tag{12b}$$

In conformal coordinates, also $g_{11} = g_{22}$ and thus

$$K = \frac{1}{2\sqrt{g}} \left(\frac{\partial^2}{\partial x_1^{\,2}} + \frac{\partial^2}{\partial x_2^{\,2}} \right) (\log\sqrt{g}) \tag{12c}$$

In Euclidian space, Cartesian coordinates can be defined which is equivalent to also setting $g = 1$ and thus

$$K = 0 \tag{12d}$$

as would be expected for a flat space.

When surface coordinates are changed from old coordinates (x_1, x_2) to new coordinates (y_1, y_2), the Gaussian curvature can be computed in each to give K_{old} and K_{new} respectively which by coordinate invariance yields the equality

$$K_{new} = K_{old} \tag{13}$$

which is a constraint on the new choice of metric. To create new orthogonal coordinates on a surface region described by old generally nonorthogonal coordinates, the constraint becomes

$$\frac{\partial}{\partial y_2}\left(\frac{1}{\sqrt{g}} \frac{\partial g_{11}}{\partial y_2}\right) + \frac{\partial}{\partial y_1} \left(\frac{1}{\sqrt{g}} \frac{\partial g_{22}}{\partial y_1}\right) = 2\sqrt{g}\ K_{old} \tag{14}$$

with the g_{ij}'s attached to the (y_1, y_2) rather than (x_1, x_2). When the surface curvature is known in advance, K_{old} can be used directly. As examples, $K_{old} = b^{-2}$ for a sphere of radius b and $K_{old} = 0$ for a Euclidian

plane. In the Euclidian case, the constraint on two-dimensional orthogonal coordinates then becomes

$$\frac{\partial}{\partial y_1}\left(\frac{1}{\sqrt{g_{11}\,g_{22}}}\frac{\partial g_{22}}{\partial y_1}\right)+\frac{\partial}{\partial y_2}\left(\frac{1}{\sqrt{g_{11}\,g_{22}}}\frac{\partial g_{11}}{\partial y_2}\right)=0 \tag{15}$$

The Restricted Existence of Three-Dimensional Orthogonal Coordinates

As in the two-dimensional case, distance measurement in three dimensions is given by a metric

$$(ds)^2 = g_{ij}\,dx_i\,dx_j \tag{16}$$

where the indices now run from 1 to 3 rather than 1 to 2. Orthogonallity is given by

$$g_{12} = g_{13} = g_{23} = 0 \tag{17}$$

and in the context of Euclidian space the coordinates are called a triply orthogonal system of surfaces.

The number of triply orthogonal systems is, however, greatly limited. Dupin [1] in 1813 showed that the intersections between the surfaces were lines of curvature of the surfaces. For any surface of such a system, this means that its orthogonal surface coordinates are of a very special type. As lines of curvature, each coordinate curve follows either a maximum of minimum curvature direction. For example, on an axis of an ellipsoid, the maximum direction points towards the smallest remaining axis while the minimum points towards the other. If they are equal, the choice of orthogonal directions is arbitrary and the location is called an umbilic. A surface where all points are umbilic is a sphere. On a sphere we would then be able to choose orthogonal coordinate systems at will. As part of a triply orthogonal system, two families of surfaces would be the cones from the origin and through the respective family of curves on the sphere. The final family would be all other spheres that are concentric to the given sphere. Unfortunately, we could not choose a nonspherical outer boundary or even a nonconcentric spherical one.

The restriction is clearly more severe for general surfaces. As an example, three-dimensional boundary layer coordinates over a two-dimensional surface would be useful in a number of situations but only exists (as shown by

Darboux [1]) when the surface coordinates are lines of curvature. Under a further requirement of grid clustering along the surface, clustering would have to appear where it is not needed. In addition, if only a portion of the surface is needed, then its boundaries must also be lines of curvature which is not generally possible. Because of the restrictions to such specialized cases, three-dimensional orthogonal coordinates will not be available for most problems with nontrivial geometry. As a consequence, only two-dimensional orthogonal systems are of real interest. In the threedimensional case, the primary interest is with orthogonality only on one family of surfaces or only on boundaries.

Analytic and Finite Difference Orthogonality

When the curvilinear variables of a transformation are uniformly discretized and are subsequently mapped by the transformation, we obtain a curvilinear grid. Without any further use of the given transformation, all of the metric data must be evaluated directly from the grid. In particular, orthogonality would then be determined only by the grid. The evaluation should also be consistent with the overall numerical scheme. The consistency is particularly evident in cases where conservation law forms are used and where it is important to correctly model internal flux balances. Typically, the choice is between central or one-sided differences. For our discussion, central differences will be used with the understanding that other forms of differencing would follow the same pattern.

The uniform discretization of the curvilinear variables is given by the coordinate vector $\vec{x}(i, j) = (a + i\,\Delta x_1, b + j\,\Delta x_2)$ for increments Δx_k and for constants a and b. The curvilinear surface grid is then given by $\vec{c}(i,j) = \vec{c}(\vec{x}(i,j))$ from Eq. 2. With respect to the grid, the central difference approximation to the natural tangents from Eq. 5 are second order accurate and are given by

$$\vec{E}_1(i, j) = \frac{\vec{c}(i + 1, j) - \vec{c}(i - 1, j)}{2\,\Delta x_1} \tag{18}$$

and

$$\vec{E}_2(i, j) = \frac{\vec{c}(i, j + 1) - \vec{c}(i, j - 1)}{2\,\Delta x_2}$$

Their dot product yields the central difference metric

$$G_{ki} = E_k \cdot E_i \tag{19}$$

which approximates the metric g_{ki} of the analytically defined transformation. With only the grid data, the transformed partial differential equations would be expressed in terms of the approximate metric which would be from Eq. 19 in the case of central differences. Whether or not the metric appears in the explicit formulation, the effect of orthogonality comes entirely from G_{ki}. The condition for central difference orthogonality is just $G_{12} = 0$ and is illustrated in Fig. 2.

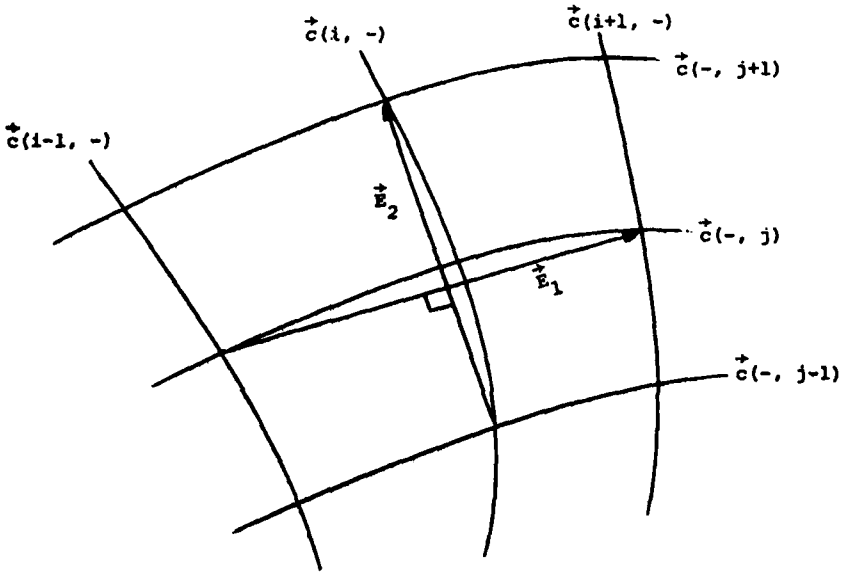


Fig. 2. Central difference orthogonality.

As an example, the polar coordinate transformation $\vec{c}(\vec{x}) = (x_1 \cos x_2, x_1 \sin x_2, 0)$ takes a uniform discretization of $\vec{x}$ into a uniform polar grid. By symmetry, the central difference metric can be computed under the assumption that the angle x_2 is 0. At $x_2 = 0$, we have $\vec{E}_1 = (1, 0, 0)$ and $\vec{E}_2 = \frac{x_1}{\Delta x_2} (0, \sin\Delta x_2, 0)$ which yields central difference orthogonallity $G_{12} = 0$ that is then valid for the entire mesh. This corresponds exactly with analytic orthogonality where the computations yield $g_{12} = 0$. The

correspondence, however, is often not exact. For example, if the angle x_2 is replaced by a nonlinear distribution function, then G_{12} is nonvanishing and g_{12} is vanishing. That is, we have analytic orthogonallity without precise central difference orthogonality. Conversely, it is also possible to have central difference orthogonality without analytic orthogonality.

When an analytically orthogonal transformation renders upon discretization a grid with a significant departure from finite difference orthogonallity, the recommended approach would then be to use the transformation itself to get the natural tangents and the subsequent metric data. The computation would involve either analytic evaluation or the use of a temporary grid refinement. With both computations, the information which is missing in the given grid is supplied. The value of this information must, of course, be balanced against other considerations before we decide to get it.

ORTHOGONAL TRAJECTORIES

Overview

The generation of orthogonal coordinates by means of orthogonal trajectories depends upon a specified family of curves and a specified pointwise distribution along one of the curves. Curves in the family must not overlap other members and must be ordered in a monotone fashion so that each is positioned to follow all previous numbers and to precede all others. If all of the curves were given an aligned and normalized arc length parameterization, then a system of coordinates would result where the curves are just a family of coordinate curves. Suitable families are then obtained from prescribed nonsingular transformations. The transformations are nonorthogonal, for otherwise, the desired system would already be given. Controls over the chosen family of coordinate curves come from the nonorthogonal transformation, are quite extensive, and have been well-developed within the general context of coordinate generation. To complete the specified data, we must select the curve on which the pointwise distribution is to be prescribed. If the curve is internal, then the orthogonal coordinates can be generated in the opposite directions away from the curve. This effectively splits the problem into two separate problems. As a consequence, the selected curve can be considered as a boundary curve which for our purposes is an initial boundary. From the pointwise distribution on the initial boundary, the orthogonal trajectories are obtained by integration across the field. They end upon the opposing boundary as a target giving it a pointwise distribution. The boundary will thus be referred to as a target boundary. The overall process is illustrated in Fig. 3 where the trajectories are the oriented curves from initial to

target boundaries. The generation of coordinates to obtain various families of curves will be discussed. This will be followed by a development of the fundamental differential equation for orthogonal trajectories along with an example. Then the various specific methods will be examined.

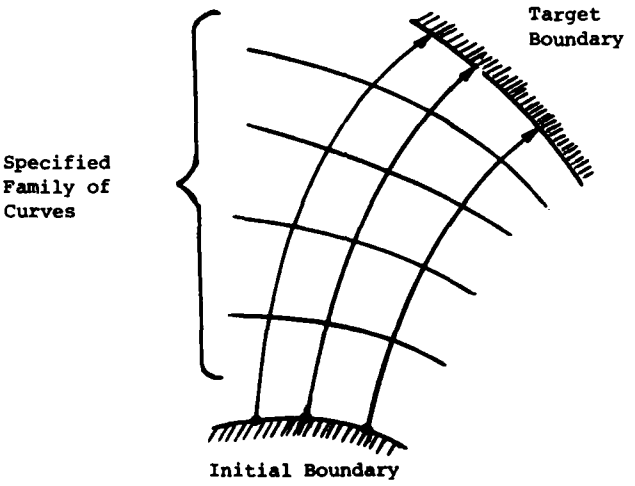


Fig. 3. Orthogonal Trajectory Methods

The Choice of Nonorthogonal Coordinates

When the desired orthogonal system only has two prescribed boundaries and a pointwise distribution on one of them, the shearing transformation yields the simplest way to get the required family of curves. On a surface, the two boundaries are curves of the form described in Eq. 3. Assuming the same parameter z_2 for each curve, let corresponding curves in the coordinate space $\vec{x} = (x_1, x_2)$ be denoted by $\vec{a}(z_2)$ and $\vec{b}(z_2)$. The shearing transformation is then given by $\vec{c}(\vec{x})$ from Eq. 2 where

$$\vec{x} = (1 - z_1)\ \vec{a}(z_2) + z_1\ \vec{b}(z_2) \tag{20}$$

for $0 \leq z_1 \leq 1$. As z_1 varies from 0 to 1, the family of curves varies from $\vec{c}(\vec{a}(z_2))$ to $\vec{c}(\vec{b}(z_2))$ which are the prescribed boundaries. The basic construction here is in the (x_1, x_2)-plane and is depicted in Fig. 4.

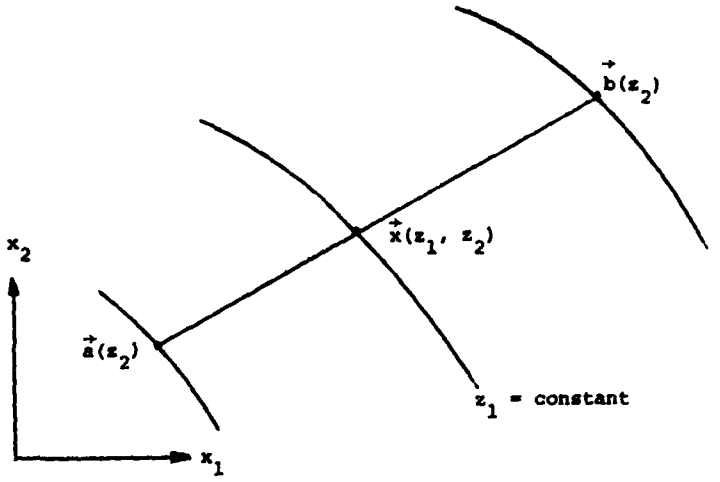


Fig. 4. The Shearing Transformation

The shearing process is also equivalent to the application of an operator P_1 that acts upon any surface coordinates between the two curves and is defined by

$$P_1[\vec{c}(\vec{x})] = \vec{c}((1 - z_1)\ \vec{a}(z_2) + z_1\ \vec{b}(z_2)) \tag{21}$$

We then easily find that the repeated application P_1^2 is just P_1 and hence that P_1 is a projector of any surface coordinates with the two given boundaries into a sheared coordinate system with the same two boundaries. This projector, however, yields a smooth curve distribution which is uniform only in the $\vec{x}$-plane. To obtain uniformity on the surface, the z_1- variable would be taken as the normalized arc length along the surface curves of shortest distance that connect the surface boundaries at each fixed z_2. From a geometric viewpoint, the direct surface construction would also be preferred since the resultant projector is independent of the given surface representation in terms of $\vec{x}$-coordinates. Although the shearing process would be accurately represented with respect to the surface, the inherent simplicity would be lost. As a consequence, will consider only projectors such as P_1 in Eq. 21 which are developed relative to the x-plane.

In the $\vec{x}$-plane immediate alternative possibilities are provided by sheared conformal transformations and by elliptic partial differential methods. With the sheared conformal approach, a contour is coarsely approximated with a

simple conformal transformation that serves to unfold it roughly onto, for example, a horizontal axis. A shearing transformation is then applied between the unfolded contour and a horizontal line to yield upon composition a coordinate system which fits the contour and is approximately conformal. A notable example is given by the sheared Schwartz-Christoffel method of Caughy [2]. With the elliptic partial differential equation approach, the Poisson system presented by Thompson, Thames, and Mastin [3] is directly applicable and also provides immediate clustering controls over the family of desired curves.

When the desired orthogonal coordinates are to conform with geometry on two boundaries and with pointwise distributions on three boundaries, more control is needed in the nonorthogonal transformation. At a basic level, we must be able to obtain a family of coordinate curves which connect two opposing boundaries with specified geometry and which are orthogonal to each of them. Then the connecting curves define the geometry of the remaining two boundaries in correspondence with the end points on the two given boundaries. Taking one of the connecting curve boundaries as an initial boundary, orthogonal trajectories are computed towards the other as a target boundary. Because of the existing boundary orthogonality at the two specified boundaries, they would already be trajectories; hence, would be preserved under the accurate computation of trajectories. This process is depicted in Fig. 5.

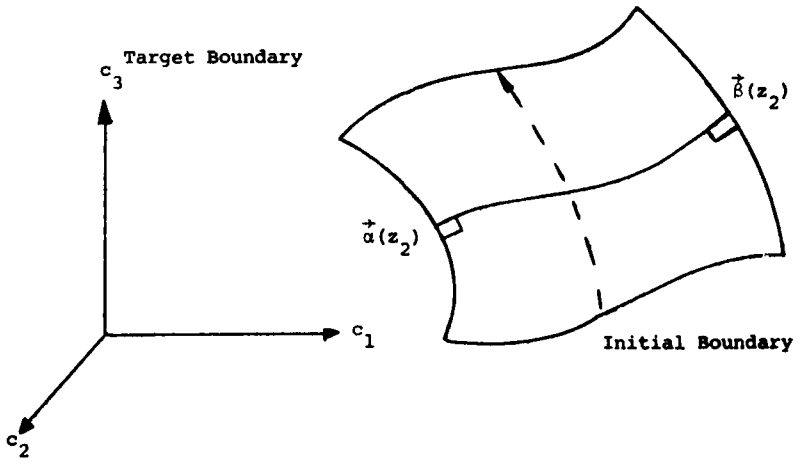


Fig. 5. Nonorthogonal surface coordinates with orthogonality on two prescribed opposing boundaries.

The specified boundaries on surface are the curves denoted by $\vec{\alpha}(z_2)$ and $\vec{\beta}(z_2)$ which are respectively images of curves $\vec{a}(z_2)$ and $\vec{b}(z_2)$ in the $\vec{x}$-plane. The specified orthogonality conditions on the surface are pulled back to the $\vec{x}$-plane where they are not orthogonality conditions but are more general derivative conditions. In the plane $\vec{c} = (x_1, x_2, 0)$ the two conditions coincide. The derivative condition is obtained by using the known boundary data in the fundamental differential equation for orthogonal trajectories. Its derivation will be presented in the next section.

In the $\vec{x}$-plane, coordinates which fit the boundaries $\vec{a}(z_2)$ and $\vec{b}(z_2)$ and which meet the desired derivative conditions can be simply obtained from the multisurface transformation [4] where further controls on the family of connecting curves are also readily available. In two-dimensions, the constructive surfaces in the transformation are curves $\vec{w}_k(z_2)$ which are ordered from boundary $\vec{a}(z_2) = \vec{w}_1(z_2)$ to boundary $\vec{b}(z_2) = \vec{w}_N(z_2)$ with the intermediate curves $\vec{w}_2(z_2)$, ..., $\vec{w}_{N-1}(z_2)$ available for control. At each value of z_2, a piecewise linear connecting curve is determined by joining the successive points $\vec{w}_k(z_2)$ with straight line segments. In correspondence with the N-1 straight line segments, a partition $r_1 < r_2 < \ldots < r_{N-1}$ for the assumed transverse coordinate z_1 is prescribed. In further correspondence, a sequence of interpolation functions $\psi_k(z_1)$ is defined to vanish at all partition points except r_k as k = 1,2, ..., N-1. The general N-surface transformation is then given by

$$\vec{x}(z_1, z_2) = \vec{w}_1(z_2) + \sum_{k=1}^{N-1} \frac{G_k(z_1)}{G_k(r_{N-1})} [\vec{w}_{k+1}(z_2) - \vec{w}_k(z_2)] \tag{22}$$

where

$$G_k(z_1) = \int_{r_1}^{z_1} \psi_k(z)\, dz$$

It is an easy matter to check that

$$\vec{x}(r_1, z_2) = \vec{a}(z_2)$$
$$\vec{x}(r_{N-1}, z_2) = \vec{b}(z_2) \tag{23}$$

and

$$\frac{\partial \vec{x}}{\partial z_1}(r_k, z_2) = \frac{\psi_k(r_k)}{G_k(r_{N-1})}\left[\vec{w}_{k+1}(z_2) - \vec{w}_k(z_2)\right]$$

for k = 1, 2, ..., N-1. From a substitution for the derivatives, we obtain the alternative Hermite form

$$\vec{x}(z_1, z_2) = \vec{w}_1(z_2) + \sum_{k=1}^{N-1} \frac{G_k(z_1)}{\psi_k(r_k)} \frac{\partial \vec{x}}{\partial z_1}(r_k, z_2) \tag{24}$$

With either form, an operator P_1 is defined by

$$P_1 \vec{c}(\vec{x}) = \vec{c}(\vec{x}(z_1, z_2)) \tag{25}$$

and is seen to be a projector ($P_1^2 = P_1$). Any coordinates on the surface which meet the surface data that has been pulled back to the $\vec{x}$-plane data of Eq. 23 are projected by P_1 into the multisurface coordinates. When global (N-2)nd degree polynominal interpolants ψ_k are used, the N-surface transformation is given by an (N-1)st degree polynominal in z_1. Here, each surface corresponds to an extra degree of freedom. With 2 of them, we get the shearing transformation of Eq. 20 and the corresponding projector given in Eq. 21. With 4 of them and the partition $0 = r_1 < r_2 = \frac{1}{2} < r_3 = 1$, we get the cubic Hermite projector where

$$\vec{x}(z_1, z_2) = (1 - 3 z_1^2 + 2 z_1^3)\vec{x}(0, z_2) + z_1^2 (3 - 2 z_1)\vec{x}(1, z_2) \tag{26}$$

$$+ z_1(1 - z_1)^2 \frac{\partial \vec{x}}{\partial z_1}(0, z_2) - z_1^2(1 - z_1)\frac{\partial \vec{x}}{\partial z_1}(1, z_2)$$

The cubic Hermite transformation of Eq. 26 has been studied by Eiseman [4], Smith and Weigel [5], and Kowalski [6]. The associated projector also meets the minimal conditions that are demanded of the connecting curves that are illustrated in Fig. 5. The control required to meet more conditions is available with larger N. Since N distinct conditions are enough to determine an (N-1)st degree polynomial, the polynomial N-surface transformation can be cast into the general Hermite form.

$$\vec{x}(z_1, z_2) = \sum_{j=1}^{J} \sum_{i=0}^{I_j} \phi_{ij}(z_1) \frac{\partial^i}{\partial z_1^i} \vec{x}(t_j, z_2)$$

with

$$N = \sum_{j=1}^{J} (1 + I_j) \tag{27}$$

and polynomials $\phi_{ij}(z_1)$ satisfying

$$\frac{\partial^k}{\partial z_1^k} \phi_{ij}(t_\ell) = \delta_{ki} \delta_{\ell j}$$

for $t_1 = r_1$, $t_J = r_{N-1}$, and arbitrarily selected points t_j in between. The most direct and useful application of the Hermite form is the specification of extra conditions only at the boundaries. For more precise controls over the curves, we must return to the selection of the multisurface interpolants $\psi_k(z_1)$ and take them to be local piecewise-polynominal functions. The added precision comes from the resultant local dependence on constructive surfaces and from the use of lower order polynomials in the construction of the ψ_k. Details are presented by Eiseman [7, 8] and Eiseman and Smith [9]. One application of the extra control, here, would be to approximate the desired geometry of the lateral boundaries which were the unspecified initial and target boundaries of Fig. 5.

When an exact specification is required of all boundaries, the simplest way to get suitable coordinates is to use Boolean sums of projectors as presented by Gordon [10] and by Gordon and Hall [11]. In addition to the projectors P_1 that we developed for the z_1-variable, similar projectors P_2 can also be defined for the z_2-variable. The Boolean sum of the projectors is defined by

$$P_1 \oplus P_2 = P_1 + P_2 - P_1 P_2 \tag{28}$$

and is itself a projector if P_1 and P_2 commute ($P_1P_2 = P_2P_1$). With $P_1 \oplus P_2$ as a projector, we obtain surface coordinates $(P_1 \oplus P_2)\, \vec{c}(\vec{x})$ which conform to all boundaries. At a minimum, orthogonality is needed on two opposing boundaries so that they are reproduced as orthogonal trajectories. To meet the requirement, P_1 is taken as the cubic Hermite projector (Eq. 26 inserted into Eq. 25) while P_2 is taken as the shearing projector (of Eq. 21 with an interchange of variables). These projectors commute and thus define suitable coordinates.

Watford [12] used these coordinates in the plane for his orthogonal trajectory method. When more control is desired, either the general Hermite form of Eq. 27 or the local multisurface interpolants (for Eq. 22) may be used. With the general Hermite forms in both projectors, mixed derivatives occur and, as examined by Gregory [13], commutativity only comes from continuity at the corners of the region. Barnhill [14] presents two methods to remove the problem which are known as Little's square and Brown's square respectively. Direct applications of the general Hermite form have been considered by Rizzi and Eriksson [15] for higher order specified data on the boundaries. With the general form of the multisurface transformation given in Eq. 22, commutativity comes from the specification of the control curves. When they form a control net, commutativity is immediate and the local interpolants can be applied directly. Otherwise, the same considerations as in the general Hermite case must be examined.

As an alternative to the above algebraic constructions, suitable coordinates can also be readily obtained by iterative elliptic partial differential equation method of Sorenson and Steger [16]. The iteration is done globally on the whole system of equations. In each cycle, the source terms of the Poisson system from Thompson, Thames, and Mastin [3] are updated. The cycles also include the main solution step for the inverse Poisson system. Thus, the additional cost of iteration is minimal. Upon convergence, derivative conditions can be satisfied at two opposing boundaries. As a consequence, boundary orthogonality can be specified.

The Fundamental Differential Equation

Once a suitable nonorthogonal coordinate system has been selected for the surface, the differential equation for orthogonal trajectories can be formulated with respect to the desired family of coordinate curves. Rather than specific formulations, we will develop the fundamental differential equation from which the others will follow as special cases. Without loss of generality, the given nonorthogonal coordinates can be taken as $\vec{x} = (x_1, x_2)$ for the surface $\vec{c}(\vec{x})$ of Eq. 2. This is only the assumption that the right coordinates were selected in the original surface representation.

The orthogonal coordinates to be generated from the trajectories will be denoted by $\vec{y} = (y_1, y_2)$ and defined on the domain $a_1 \leq y_1 \leq b_1$, $a_2 \leq y_2 \leq b_2$. The original coordinate curves with constant x_2 will also be assumed as coordinate curves in the orthogonal system, and thus, will constitute the specified family of curves. When the family of curves are given by constant

values of y_2, the desired transformation takes the form

$$x_1 = x_1(y_1, y_2)$$
$$x_2 = x_2(y_2) \tag{29}$$

and corresponds to the pointwise distribution $x_1(y_1, a_2)$ on the initial curve $\vec{c}(x_1(y_1, a_2), x_2(a_2))$. With the metric coefficients $\bar{g}_{ij}$ for the (y_1, y_2)-coordinates, the orthogonality condition for the transformation is just $\bar{g}_{12} = 0$. Under the change of coordinate rule given in Eq. 10, the orthogonality condition expands into

$$g_{11} \frac{\partial x_1}{\partial y_1} \frac{\partial x_1}{\partial y_2} + g_{12} \left(\frac{\partial x_1}{\partial y_1} \frac{\partial x_2}{\partial y_2} + \frac{\partial x_2}{\partial y_1} \frac{\partial x_1}{\partial y_2} \right) + g_{22} \frac{\partial x_2}{\partial y_1} \frac{\partial x_2}{\partial y_2} = 0 \tag{30}$$

where the metric coefficients g_{ij} are attached to (x_1, x_2). From the form of the transformation given in Eq. 29, the derivative $(\partial x_2/\partial y_1)$ vanishes and the condition reduces to

$$\left(g_{11} \frac{\partial x_1}{\partial y_2} + g_{12} \frac{\partial x_2}{\partial y_2} \right) \frac{\partial x_1}{\partial y_1} = 0 \tag{31}$$

If $(\partial x_1/\partial y_1)$ also vanishes, then from Eq. 10b we get $\bar{g}_{11}$ which together with $\bar{g}_{12} = 0$ yields a vanishing Jacobian $J = \sqrt{\bar{g}_{11}\, \bar{g}_{22} - \bar{g}_{12}^2}$ and thus a coordinate singularity. For nonsingular coordinates the isolated derivative can then be factored out of Eq. 31 to give

$$g_{11} \frac{\partial x_1}{\partial y_2} + g_{12} \frac{\partial x_2}{\partial y_2} = 0 \tag{32}$$

The derivative $(\partial x_2/\partial y_2)$ in the equation must be the total derivative (dx_2/dy_2) which directly follows from the form of the transformation given in Eq. 29. With nonsingularity, it must also not vanish. As a result, $x_2(y_2)$ is monotone, the inverse function $y_2(x_2)$ exists, and it is also monotone. The orthogonal trajectories are the curves with y_1 taken as a constant which for a particular trajectory may be denoted by $\bar{y}_1$. Upon substitution, the transformation reduces to $x_1 = x_1\,(\bar{y}_1, y_2\,(x_2))$ which for simplicity can be denoted by $x_1(x_2)$. With one application of the chain rule, Eq. 32. becomes

$$\frac{dx_1}{dx_2} = - \frac{g_{12}}{g_{11}} \tag{33}$$

which is the fundamental ordinary differential equation for orthogonal trajectories. The equation is always well-defined for nonsingular coordinates since $g_{11} > 0$. The initial conditions are just $x_1(e) \equiv x_1(\bar{y}_1, a_2)$ for each $\bar{y}_1$ and where $e = x_2(a_2)$. Upon integration,

$$x_1(\bar{y}_1, y_2) = x_1(\bar{y}_1, a_2) - \int_e^{x_2(y_2)} \frac{g_{12}}{g_{11}} \, dz \tag{34}$$

which explicitly defines the trajectories for a given $x_2(y_2)$. The trajectories are also the characteristics to the fundamental hyperbolic partial differential equation which is given by Eq. 32 in inverse form. With Jacobian J, the inverse derivative expressions

$$\frac{\partial x_1}{\partial y_2} = - \frac{1}{J} \frac{\partial y_1}{\partial x_2} \quad \text{and} \quad \frac{\partial x_2}{\partial y_2} = \frac{1}{J} \frac{\partial y_1}{\partial x_1}$$

are inserted into Eq. 32 to yield

$$\frac{\partial y_1}{\partial x_2} - \frac{g_{12}}{g_{11}} \frac{\partial y_1}{\partial x_1} = 0 \tag{35}$$

which has characteristic directions given by the fundamental ordinary differential equation, Eq. 33.

An Example of the Fundamental Differential Equation

To illustrate how the fundamental differential equation is applied and how it relates to other derivations, we will consider a simple useful example. Let f be a nonnegative function of a single variable and consider the shearing transformation

$$\vec{c} = (x_1, x_2 f(x_1), 0) \tag{36}$$

with curvilinear coordinates $p \leq x_1 \leq q$ and $0 \leq x_2 \leq 1$. For constant values of x_2, we get a family of curves through which we wish to obtain orthogonal trajectories. The curves are in the plane $c_3 = 0$ and are depicted in Fig. 6.

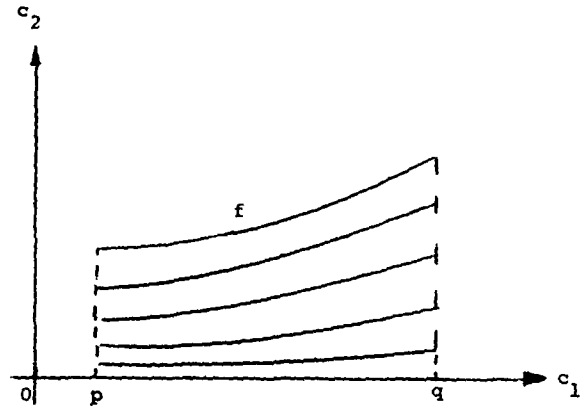


Fig. 6. Family of curve from shearing transformation.

From the definition in Eq. 5, the natural tangents to the coordinate curves are the respective x_k-derivatives of $\vec{c}$ and are given by

$$\vec{e}_1 = (1,\ x_2\ f'(x_1),\ 0)$$
$$\vec{e}_2 = (0,\ f(x_1),\ 0) \tag{37}$$

where f' denotes the derivative of f. The metric coefficients are defined in Eq. 6 as $g_{ij} = \vec{e}_i \cdot \vec{e}_j$ and in our specific sheared system become

$$g_{11} = 1 + [x_2\ f'(x_1)]^2$$
$$g_{12} = x_2\ f(x_1)\ f'(x_1) \tag{38}$$
$$g_{22} = [f(x_1)]^2$$

By substitution, the fundamental ordinary differential equation (Eq. 33) becomes

$$\frac{dx_1}{dx_2} = -\frac{x_2\ f(x_1)\ f'(x_1)}{1 + [x_2\ f'(x_1)]^2} \tag{39}$$

has a unique solution when f' satisfies a Lipschitz condition.

To obtain a comparison with direct Cartesian formulations, we will write the fundamental equation in terms of the Cartesian components $c_1 = x_1$ and $c_2 = x_2\, f(x_1)$. The inverse transformation is then given by $x_1 = c_1$ and $x_2 = c_2/f(c_1)$. This, in turn, is differentiated to yield the differentials

$$dx_1 = dc_1 \tag{40}$$

$$dx_2 = \frac{dc_2 - x_2\, f'(c_1)\, dc_1}{f(c_1)}$$

Upon direct substitution, the fundamental equation (Eq. 39) becomes

$$\frac{dc_2}{dc_1} = -\frac{1}{x_2\, f'(c_1)} \tag{41}$$

which could have been directly inferred on the basis of negative inverse slopes to the curves $c_2 = x_2\, f(c_1)$.

In the Cartesian formulation, a further variant with slopes is considered in Berger and Curtis [17]. The family of curves is defined in the functional form

$$\phi(x_2, c_1, c_2) = 0 \tag{42}$$

A normal increment (dc_1, dc_2) should then be proportional to the gradient $\nabla\phi = \left(\frac{\partial\phi}{\partial c_1}, \frac{\partial\phi}{\partial c_2}\right)$ which means that the slopes are equal; hence we have

$$\frac{dc_2}{dc_1} = \frac{\left(\frac{\partial\phi}{\partial c_2}\right)}{\left(\frac{\partial\phi}{\partial c_1}\right)} \tag{43}$$

For our example, we take $\phi = c_2 - x_2\, f(c_1)$ and from Eq. 43 arrive at the result of Eq. 41.

Methods of Generation

The various orthogonal trajectory methods can be separated into the categories which correspond to either a continuous or discrete specified family of curves. With the continuous specification, the fundamental differential equation or one of its equivalent forms is usually used. The family of curves is usually generated from algebraic coordinate transformations which are preferable because a smooth normal field must be readily defined everywhere so that the fundamental differential equation is also

available everywhere. Primary advantages with the continuous family are that the orthogonal metric coefficients can be accurately computed regardless of the resultant grid point locations and that the accuracy in the determination of the locations can be made arbitrarily high in either a global or local sense. The correct locations are especially important in cases where the trajectory must reproduce a boundary which was intentionally taken to be orthogonal to the specified family of curves. The objective there is to generate orthogonal coordinates with specified boundary data on three out of four boundaries. The accurate computation of metric data comes from the coordinate change rule of Eq. 10 and from the geodesic curvature [1] of the coordinate curves under consideration. As in the case with Gaussian curvature (Eqs. 11-15), geodesic curvature is also an intrinsic quantity. For coordinate curves in x_1, it is given by

$$K_1 = -\frac{1}{2\sqrt{g g_{11}^3}} \left\{ g_{11} \frac{\partial g_{11}}{\partial x_2} + g_{12} \frac{\partial g_{11}}{\partial x_1} + 2 g_{11} \frac{\partial g_{12}}{\partial x_1} \right\}$$

and in x_2, by

$$K_2 = \frac{1}{2\sqrt{g g_{22}^3}} \left\{ g_{22} \frac{\partial g_{22}}{\partial x_1} + g_{12} \frac{\partial g_{22}}{\partial x_2} - 2 g_{22} \frac{\partial g_{21}}{\partial x_2} \right\} \tag{44}$$

For the orthogonal coordinate system (y_1, y_2) with metric $h_1^2 = \bar{g}_{11}$ and $h_2^2 = \bar{g}_{22}$, the curvatures reduce to

$$K_1 = -\frac{1}{h_1 h_2} \frac{\partial h_1}{\partial y_2} \tag{45}$$

$$K_2 = \frac{1}{h_1 h_2} \frac{\partial h_2}{\partial y_1}$$

Since the family of curves with constant x_2 was retained as constant y_2 curves in the form of the transformation (Eq. 29), the curvature K_1 along each member of the family is independent of the coordinate system. As a consequence, K_1 can be analytically computed at any location in $\vec{x}$-coordinates (Eq. 44) and must be equal to the corresponding expression in $\vec{y}$-coordinates (Eq. 45). In the expression, h_2 can also be computed analytically in terms of (x_1, x_2). Under a change of coordinates, the metric coefficient $\bar{g}_{22}$ can be expanded in terms of

g_{ij} by the rule given in Eq. 10. When the orthogonality condition of Eq. 32 is used to eliminate the derivative $(\partial x_1/\partial y_2)$ in the expansion, we get

$$h_2 = \sqrt{\frac{g}{g_{11}}}\ \frac{dx_2}{dy_2} \tag{46}$$

which is suitable for analytic evaluation. With the evaluations for K_1 and h_2, the curvature equation becomes

$$\frac{dh_1}{dx_2} = -\left(K_1\sqrt{\frac{g}{g_{11}}}\right) h_1 \tag{47}$$

This can be solved simultaneously with the fundamental differential equation (Eq. 33) to accurately produce values of the metric h_1 and h_2 along with the locations. The calculation for h_1 is then effectively decoupled from parallel solution data which otherwise would have been required.

Blottner and Moreno [18] and Blottner and Ellis [19] have considered applications to surfaces of the general form

$$\vec{c} = (r(x_1, x_2)\cos x_2,\ r(x_1, x_2)\sin x_2,\ x_1) \tag{48}$$

which included particular studies for ellipsoids, paraboloids, and spheres. In each case, they were interested in the creation of approximate boundary layer coordinates. From Darboux's Theorem [1], exactness is not possible unless the resultant orthogonal system reproduces lines of curvature. In their study, Eqs. 33 and 47 were derived for the constant x_2-curves on the surfaces of Eq. 48. The equations were then solved for both locations and metric data.

In another study, Watford used the fundamental equation (Eq. 33) and the metric equation (Eq. 47) for the generation of planar coordinates. The specified family of curves came from the nonorthogonal coordinates defined by the shearing projector of Eq. 21, the Hermite projector of Eq. 26, and the Boolean sum (Eq. 28) of the two. On application, the metric equation was written for ΔS_1 rather than h_1 by using the relationship $ds_1 = h_1\ dy_1$.

With a further study of the planar case, Davies [20] considered the fundamental equation in the hyperbolic partial differential equation form of Eq. 35. His basic solution was implicit and was interlaced with a characteristic analysis. Unlike the others, interpolation was required along

the constant y_2-curves and the metric data was determined solely by finite differences rather than by Eq. 47.

When the specified family of curves is to be given in discrete form, the associated nonorthogonal coordinate transformation can be continuously or discretely defined and can be obtained by any readily available method. The objective is to now determine an orthogonal correspondence between successive curves which would be a good approximation to the corresponding analytically defined trajectories had we been able to fill in the spaces between curves with the underlying nonorthogonal coordinates.

The method of Graves and McNally [21], [22] provides us with a simple and effective technique for the generation of orthogonal trajectories in the above discrete case. The technique is a predictor-corrector process and is illustrated in Fig. 7.

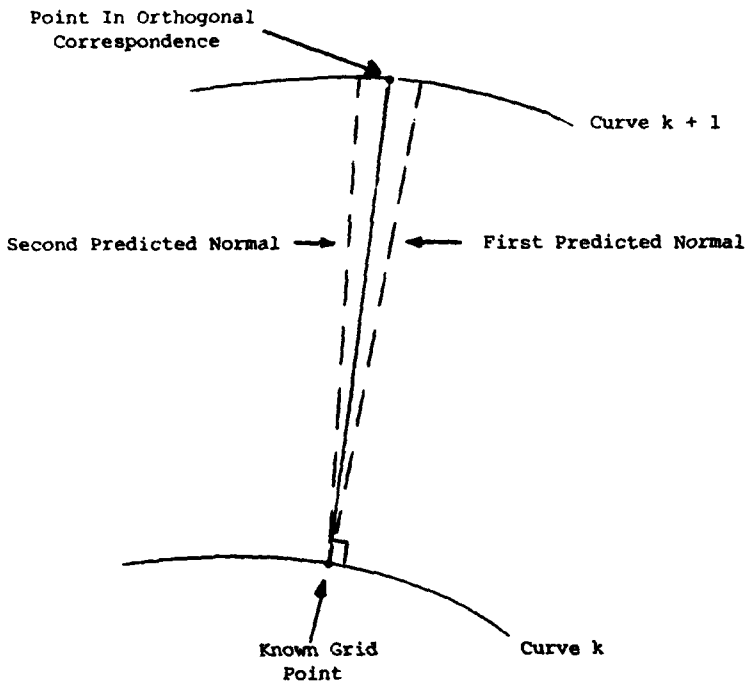


Fig. 7. Graves-McNally Predictor-Corrector Process

There it is desired to advance the orthogonal trajectory from a known grid point location on curve K up to a grid point in orthogonal correspondence on

curve K + 1. The first step is an Euler predictor in which the point of intersection is found between curve K + 1 and the normal line to curve K from the known grid point. Then the normal direction from curve K + 1 at the point of intersection is used to obtain a second predicted normal line from the known grid point. The process is completed when the point of intersection for the second predicted normal is found on curve K + 1 and is averaged with the first intersection point. The result is the desired point which is in orthogonal correspondence with the known grid point location that we started with on curve K.

With the view that the orthogonal correspondence depends upon the missing continuum of curves, Potter and Tuttle [23] have proposed a method in which the continuum is approximated by an assumption on the normal directions. In particular, the normal directions are represented by a vector field of unknown magnitude which is then completely determined under the assumption that it is divergence free. The divergence free assumption means that the normal field has no sources or sinks. The mathematical implication is that the cell aspect ratio (Eq. 9) is of the form

$$\sqrt{\frac{g_{11}}{g_{22}}} = a(y_1)\ b(y_2) \tag{49}$$

for some functions a and b. In terms of potential functions defined by $a = dA/dy_1$ and $b = dB/dy_2$, Laplace equations are obtained and solved between the successive curves K and K + 1. At the outset, the initial locations, and also normal directions, are known on curve K. On curve K + 1, the family of normal directions are known and the pointwise locations are desired to determine the orthogonal correspondence. In between, the postulated vector field is smoothly filled in and conforms to the known family of directions on curves K and K + 1. From the solution, which is done only on the given curves, a unique orthogonal correspondence is obtained. The divergence free assumption, however, is not valid for all orthogonal coordinate systems. For example, between concentric circles of a polar coordinate system, the assumption fails since the normal vector field is always of unit magnitude and enters the inner circle over a smaller arc length than the outer circle on which it exits. Although the assumption is not met, the form of the metric aspect ratio (Eq. 49) is retained. Moreover, with logarithmic scaling in the radial direction, the normal fluxes can be balanced on concentric circles. Here, it is evident that magnitudes of the normal vector field are also uniquely determined. In general, the aspect ratio form of Eq. 49 is not

retained. To illustrate this fact, we consider the example which lead to Eq. 41 and take $f(x_1) = x_1^2/2$ which yields a case where the orthogonal trajectories can be computed analytically. With initial data at $c_2 = 0$ given by $c_1 = 1 + y_1$ for $0 \leq y_1 \leq 1$, we readily obtain the cell aspect ratio

$$\sqrt{\frac{g_{11}}{g_{22}}} = \frac{y_2 \sqrt{2(R + 1)}}{R - 1} \tag{50}$$

where

$$R = \sqrt{1 + 2\, y_2^2 (1 + y_1^2)}$$

Because of the radicals, it is clearly impossible to manipulate the expression into the product form of Eq. 49.

With the work by Ghia, Hodge, and Hankey [24] we return to the linear constructive process between curves K and K + 1. In contrast to the method of Graves and McNally, their method is based upon an approximation of $g_{12} = 0$. In the discrete notation of Eq. 18, the first step is to find the index j for which the expression

$$\left[\vec{c}(j, k + 1) - \vec{c}(i, k)\right] \cdot \left[\frac{\vec{E}_1(i, k) + \vec{E}_1(j, k + 1)}{\quad}\right] \tag{51}$$

changes sign. Here, i designates the fixed point on curve K, and j is the index for successive points on curve K + 1. The point corresponding to the sign change would yield a rough approximation to orthogonality. In a second step, the approximation is improved by linear interpolation over the interval on which the sign change occurred.

To alleviate the orthogonality errors, it appeared to the present author that a leap frog technique would be advantageous. Consequently, the leap from method was developed and applied to some examples which were graphically very similar to the results from the previous discrete methods, but were orthogonal to machine accuracy in the central difference metric $G_{k\ell}$ of Eq. 19. We shall now describe the method which is also illustrated in Fig. 8. The first step is to compute the central difference natural tangent $\vec{E}_1(i, K)$ which is given in Eq. 18 and which is at the ith point

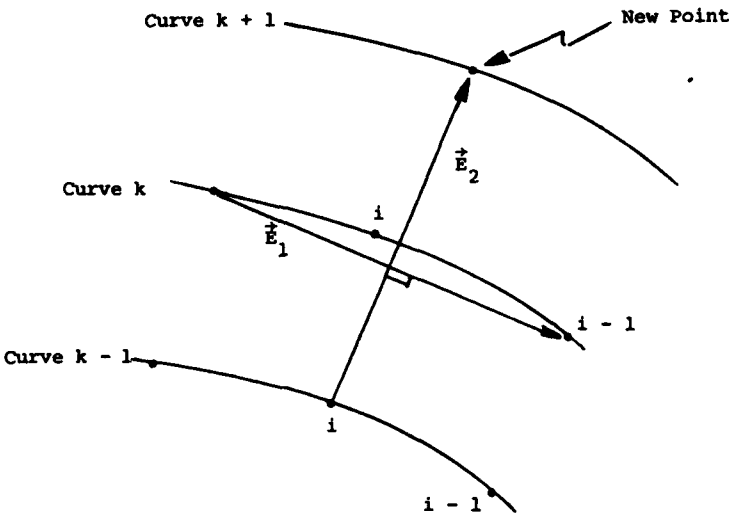


Fig. 8. The Leap-Frog Method

on curve K. In the second step, a line which is orthogonal to $\vec{E}_1(i, K)$ is constructed from point i on the previous curve K - 1. The third and final step is to find the point of intersection on curve K + 1. The new point on the ith orthogonal trajectory is the intersection point. Since the previous curve is always needed for the leap frog method, the first curve from the boundary must be treated differently. This is done by using point i on curve K rather than on curve K + 1. Then the resulting orthogonal grid is exactly orthogonal with respect to central differences at internal points and with respect to mixed forward and central differences on the initial boundary. On the target boundary, by contrast, nothing can be claimed.

When the continuous family of curves is considered, all of the above discrete methods can be used on a succession of discretizations. Here we suppose that a fixed discretization is required along with accuracy demands that exceed the level available from the resulting grid. The desired level can then be obtained from grid refinements. In the limit, the results converge upon the analytically defined systems, and for accuracy, become more competitive with methods which directly use the fundamental differential equation (Eq. 33). The accuracy is especially important if a prescribed boundary is to be reasonably preserved as an orthogonal trajectory. Otherwise, the specified boundary could be lost as the trajectory either

leaves or goes interior to the region. In conjunction, there is also the small technical problem of either modifying the methods near such boundaries or else extending the specified family of curves to cover the small boundary deviations. In addition, the accuracy is also required to extract the metric data from the analytically defined orthogonal system which is being approximated. The intended application is clearly for cases in which an analytic or nearly analytic evaluation of metric data is desired. In this context, the finite difference orthogonality from the leap-frog method would not be realized.

FIELD SOLUTIONS

Overview

With the orthogonal trajectory methods, we have seen that orthogonal coordinates can be efficiently generated and be made to conform to a specified pointwise distribution on three out of four boundaries. To specify a distribution on the fourth boundary, we would require some global interative procedure in which the previously specified and fixed family of curves would have to be continually updated. On such problems, a more attractive and less complex approach would then be to obtain a method for the whole field at once. The result would be field solutions to systems of elliptic partial differential equations which arise from specified forms of the metric. The metric specifications over a field also lead to methods for another class of problems. These are called marching methods since the grid is generated by marching away from the single specified boundary. The class of problems are those where the exact form of the far field boundary is not needed and where more importance is placed upon minimizing the amount of sensitive user specified data. Here, we shall discuss both elliptic and marching methods under the general framework of metric specifications.

The Curvature Constraint

The first and most basic metric specification is just the orthogonality condition $g_{12} = 0$. Without any further condition, a unique system of orthogonal coordinates cannot be found. This is generally evident from the curvature constraint of Eq. 14 where there is the immediate freedom to choose either g_{11} or g_{22} or a single relationship between them. More specifically, we shall illustrate the nonuniqueness with a concrete example obtained by orthogonal trajectories. For the example, consider the nonorthogonal coordinates (x_1, x_2) for $\vec{c}(\vec{x})$ of Eq. 2 which are defined by

$$c_1 = x_1$$

$$c_2 = (x_2 - x_2^2)\,(3x_1^2 - 2x_1^3) + x_2^2 \qquad (52)$$

$$c_3 = 0$$

for $0 \leq x_k \leq 1$. In the (c_1, c_2)-plane, the coordinates cover the unit box and are depicted in Fig. 9. The constant x_1 curves are vertical lines.

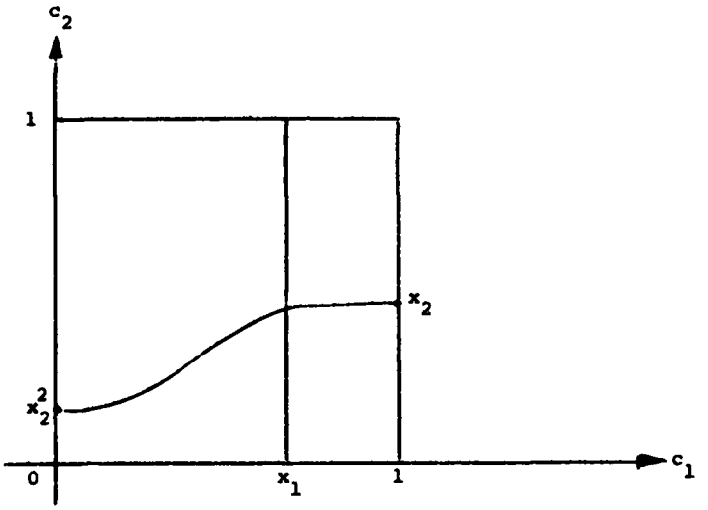


Fig. 9. Curves for nonuniqueness example.

The constant x_2-curves enter the boundaries $c_1 = 0$ and $c_1 = 1$ with zero slope and with the respective c_2 locations x_2^2 and x_2. When orthogonal trajectories are constructed from a uniform distribution on the top of the box ($c_2 = 1$), the resulting orthogonal coordinates for the box are uniformly distributed at both $c_1 = 1$ and $c_2 = 1$. When the system is reflected through both c_1 and c_2 axes successively, a nontrivial orthogonal coordinate system is obtained with a uniform pointwise distribution on all boundaries for the larger box $-1 \leq c_k \leq -1$. The system is also seen to be singular at the origin. However, with the same uniform boundary distribution, the larger box is also clearly covered by Cartesian coordinates. To summarize the example, the condition $g_{12} = 0$ does not yield enough information to distinguish between the Cartesian and constructed systems for the larger box. Moreover, there is no control over singularity.

Haussling and Coleman [25] presented a method that was based entirely on the condition $g_{12} = 0$. Because of the above nonuniqueness, the choice of orthogonal coordinates is determined by the various specific aspects of the numerical method such as the initial conditions for an iteration. The method, however, did produce orthogonal grids for cases with specified pointwise distributions for four boundaries. To get a system which could be solved for each coordinate, they obtained two equations by differentiating $g_{12} = 0$ in each direction y_k. With the known orthogonallity at the region corners, the two equations are equivalent to $g_{12} = 0$. In discrete form, a linear combination of the two equations had to be used in order to insure that the difference molecule had a nonzero central coefficient.

Unlike the nonunique situation above, the available degree of freedom to impose an additional metric condition can be used to establish controls on the orthogonal coordinate generation process. With an assumed condition, the full metric will be considered. For simplicity, we will concentrate on the planar case with fixed Cartesian coordinates $\vec{c} = (x_1, x_2, 0)$ for Eq. 2. The extension to curved surfaces follows a similar pattern and will be examined subsequently. Now let $\bar{g}_{ij}$ be the metric for (x_1, x_2) which is given by $\bar{g}_{11} = \bar{g}_{22} = 1$ and $\bar{g}_{12} = 0$ since the coordinates are Cartesian. With the specified metric g_{ij} for the desired orthogonal coordinates (y_1, y_2), the change of coordinate rule for metric coefficients becomes

$$\left(\frac{\partial x_1}{\partial y_1}\right)^2 + \left(\frac{\partial x_2}{\partial y_1}\right)^2 = g_{11} \tag{53}$$

$$\frac{\partial x_1}{\partial y_1}\frac{\partial x_1}{\partial y_2} + \frac{\partial x_2}{\partial y_1}\frac{\partial x_2}{\partial y_2} = 0$$

$$\left(\frac{\partial x_1}{\partial y_2}\right)^2 + \left(\frac{\partial x_2}{\partial y_2}\right)^2 = g_{22}$$

which must be solved for $x_1 = x_1(y_1, y_2)$ and $x_2 = x_2(y_1, y_2)$. The general rule is given in Eq. 10b with a notational interchange of $\bar{g}_{ij}$ and g_{ij}. The metric equations here are clearly satisfied by the polar formulation

$$\frac{\partial x_1}{\partial y_1} = \sqrt{g_{11}}\cos\alpha \qquad \frac{\partial x_1}{\partial y_2} = \sqrt{g_{22}}\sin\alpha$$

$$\frac{\partial x_2}{\partial y_1} = -\sqrt{g_{11}}\sin\alpha \qquad \frac{\partial x_2}{\partial y_2} = \sqrt{g_{22}}\cos\alpha \tag{54}$$

A solution then exists if the differential forms

$$w_1 = \sqrt{g_{11}}\,\cos\alpha\;\; dy_1 + \sqrt{g_{22}}\,\sin\alpha\;\; dy_2$$
$$w_2 = -\sqrt{g_{11}}\,\sin\alpha\;\; dy_1 + \sqrt{g_{22}}\,\cos\alpha\;\; dy_2 \tag{55}$$

respectively correspond to chain rule expansions for dx_1 and dx_2 so that $w_1 = dx_1$ and $w_2 = dx_2$. This means that w_1 and w_2 must be exact differentials. By the Poincare-Lemma [26], they are exact if they are closed which by definition means that $dw_1 = 0$ and $dw_2 = 0$ where d is the exterior derivative. In terms of the derivatives defined in Eq. 54, the closure condition is equivalent to

$$\frac{\partial}{\partial y_1}\left(\frac{\partial x_k}{\partial y_2}\right) = \frac{\partial}{\partial y_2}\left(\frac{\partial x_k}{\partial y_1}\right) \tag{56}$$

Upon substitution, the equality of mixed derivatives altogether yield two equations which reduce to

$$\frac{\partial \alpha}{\partial y_1} = \frac{1}{\sqrt{g}}\frac{\partial g_{11}}{\partial y_2}$$
$$\frac{\partial \alpha}{\partial y_2} = -\frac{1}{\sqrt{g}}\frac{\partial g_{22}}{\partial y_1} \tag{57}$$

The corresponding differential form is then given by

$$w_3 = \frac{1}{\sqrt{g}}\frac{\partial g_{11}}{\partial y_2}\,dy_1 - \frac{1}{\sqrt{g}}\frac{\partial g_{22}}{\partial y_1}\,dy_2 \tag{58}$$

For an angular function $\alpha(y_1, y_2)$ to exist such that $w_3 = d\alpha$, the differential form w_3 must be closed which, as before, is equivalent to the equality of mixed derivatives. Upon substitution, the equality becomes

$$\frac{\partial}{\partial y_1}\left(\frac{1}{\sqrt{g}}\frac{\partial g_{22}}{\partial y_1}\right) + \frac{\partial}{\partial y_2}\left(\frac{1}{\sqrt{g}}\frac{\partial g_{11}}{\partial y_2}\right) = 0 \tag{59}$$

which is just the metric constraint of Eq. 15 that results from the fact that Gaussian curvature vanishes for a plane. The expected curvature constraint

means that there is only one degree of freedom in the choice of metric. Once chosen, the constraint is the integrability condition which determines the remainder of the metric. Then the metric can be used to obtain the solution by means of line integrals of the differentials. The integrals are well-defined since the integrands are exact differentials which then yield path independence. From a given point $\vec{y}_o = (y_{10}, y_{20})$ to a new point $\vec{y} = (y_1, y_2)$, the line integrals are given by

$$\begin{bmatrix} x_1(\vec{y}) \\ x_2(\vec{y}) \end{bmatrix} = \begin{bmatrix} x_1(\vec{y}_o) \\ x_2(\vec{y}_o) \end{bmatrix} + \int_{\vec{y}_o}^{\vec{y}} \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \sqrt{g_{11}}\, dy_1 \\ \sqrt{g_{22}}\, dy_2 \end{bmatrix} \tag{60}$$

for Cartesian locations where the rotational angles are determined from

$$\alpha(\vec{y}) = \alpha(\vec{y}_o) + \int_{\vec{y}_o}^{\vec{y}} \frac{1}{\sqrt{g}} \left[\frac{\partial g_{11}}{\partial y_2} dy_1 - \frac{\partial g_{22}}{\partial y_2} dy_2 \right] \tag{61}$$

The metric development and the realization in terms of line integrals also extends to curved surfaces. For the extension, we return to the general metric rule on coordinate changes given in Eq. 10b. Since the rule produces a quadradic form for the derivatives, the associated symmetric matrix can be diagonalized by a unitary transformation. Since the matrix is also positive definite, square roots of diagonal entries can be taken and unitary matricies can be accordingly modified to give a transformation into the form of Eq. 53. The transformed derivatives are linear combinations of the original coordinate derivatives and can be expressed in the polar formulation of Eq. 54 which satisfies the basic metric rule. With the associated differential forms, exactness is achieved subject to the curvature integrability constraint for curved surfaces. Accordingly, the line integrals can be defined in the same manner as before.

Coordinates from a Metric Specification

The available degree of freedom to specify the metric can be used in various forms. The first form that we will consider is the case where orthogonal coordinates are attached to the streamlines of an isentropic steady

gas flow. This case was developed by Schindler [27] and is given here in basic form without his examples and further extensions. The equations for an isentropic steady gas are given by

$$\begin{aligned} &\frac{\partial}{\partial x_1}(\rho u_1) + \frac{\partial}{\partial x_2}(\rho\, u_2) = 0 \\ &\rho\left[u_1 \frac{\partial u_1}{\partial x_2} + u_2 \frac{\partial u_1}{\partial x_2}\right] + \frac{\partial p}{\partial x_1} = 0 \\ &\rho\left[u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2}\right] + \frac{\partial p}{\partial x_2} = 0 \\ &p\,\rho^{-\gamma} = p_o\,\rho_o^{\gamma} \end{aligned} \tag{62}$$

where (x_1, x_2) are Cartesian coordinates, (u_1, u_2) is velocity, ρ is density, p is pressure, and γ is the ratio of specific heats. In terms of the sound speed $c = \sqrt{\frac{\gamma p}{\rho}}$, the last equation in Eq. 62 is automatically satisfied when

$$\begin{aligned} p &= p_o \left(\frac{c}{c_o}\right)^{\frac{2\gamma}{\gamma-1}} \\ \rho &= \rho_o \left(\frac{c}{c_o}\right)^{\frac{2}{\gamma-1}} \end{aligned} \tag{63}$$

When orthogonal coordinates (y_1, y_2) are chosen such that streamlines are the constant y_2 curves, the gas equations Eq. 62 become

$$\begin{aligned} &\frac{\partial}{\partial y_1}\left(\sqrt{g_{22}}\; a\; c^{\frac{2}{\gamma-1}}\right) = 0 \\ &\frac{\partial}{\partial y_1}\left(\frac{a^2}{2} + \frac{c^2}{\gamma-1}\right) = 0 \\ &\frac{\partial}{\partial y_2}\left(\frac{a^2}{2} + \frac{c^2}{\gamma-1}\right) = \frac{a}{\sqrt{g_{11}}}\,\frac{\partial(a\sqrt{g_{11}})}{\partial y_2} \end{aligned} \tag{64}$$

where $a = \sqrt{u_1^2 + u_2^2}$ is the velocity of the gas. Upon integration, we have

$$\begin{aligned} &\sqrt{g_{22}}\; a\; c^{\frac{2}{\gamma-1}} = A(y_2) \\ &\frac{a^2}{2} + \frac{c^2}{\gamma-1} = \frac{1}{2} B(y_2)^2 \\ &a\sqrt{g_{11}}\, \exp\left\{-\int \frac{BB'}{a^2}\, dy_2\right\} = R(y_1) \end{aligned} \tag{65}$$

where A, B, and R are constants of integration. Under the transformation $\bar{y}_1 = \int R\, dy_1$ and $\bar{y}_2 = \int A\, dy_2$, however, we get $g_{11} = R^2\, \bar{g}_{11}$ and $g_{22} = A\, \bar{g}_{22}$, and therefore, can safely assume that R = 1 and A = 1. The remaining integration constant is the Bernoulli function $B(y_2)$ which we specify. In terms of the Mach number $M = \frac{a}{c}$, the second equation of Eq. 65 yields

$$a = B(y_2)\, \frac{M}{\sqrt{M^2 + \frac{2}{\gamma-1}}}$$

$$c = B(y_2)\, \frac{1}{\sqrt{M^2 + \frac{2}{\gamma-1}}} \tag{66}$$

On substitution into the other two equations, the metric is given by

$$\sqrt{g_{11}} = \frac{1}{M}\sqrt{M^2 + \frac{2}{\gamma-1}}\ \exp\left\{\frac{2}{\gamma-1}\int_{y_{20}}^{y_2} \frac{B'(y)}{B(y)}\ \frac{1}{M^2}\ dy\right\}$$

$$\sqrt{g_{22}} = \frac{1}{M}\left\{M^2 + \frac{2}{\gamma-1}\right\}^{\frac{\gamma+1}{2(\gamma-1)}} \tag{67}$$

After the Bernoulli function $B(y_2)$ has been specified, we get a single differential-integral equation for the Mach number $M = M(y_1, y_2)$ by substitution into the curvature integrability condition given in Eq. 59. The orthogonal coordinate system attached to the flow is then obtained from the line integrals of Eqs. 60 and 61. The available degree of freedom, here, was consumed by the Bernoulli function.

In another study, Warsi and Thompson [28] consumed the degree of freedom with an additional Laplace equation and considered general orthogonal coordinates for annular regions in the plane. Following Potter and Tuttle [23], they assumed that the desired normal field in the direction from inner to outer loop is divergence-free, and as a consequence, that the cell aspect ratio has the product form given in Eq. 49. Then under the transformation $\bar{y}_1 = \int a\, dy_1$ and $\bar{y}_2 = \int \frac{1}{b}\, dy_2$ with a and b from Eq. 49, the orthogonal metric becomes $\bar{g}_{11} = \bar{g}_{22}$. The curvature integrability constraint becomes $\nabla^2 \log \bar{g}_{11} = 0$ which is of the conformal type from Eq. 12c. To complete the system, the Laplace equation $\nabla^2 y_1 = 0$ is also derived from the above assumption.

For cases with the arbitrary boundaries, Eiseman [29] proposed a method where metrices that fit the boundary data are specified. On the two boundaries where y_2 is constant, g_{11} can be evaluated directly by finite differences, g_{22} can be evaluated only at the endpoints (i.e. region corners), and g_{22} must be specified elsewhere as a rate of entry or exit into or out of the region. On the other two boundaries, the roles of the indices 1 and 2 are interchanged. With the metric given on the boundary, transfinite interpolation (Eq. 28 or [10]) is used to continue it smoothly over the entire region. Local clustering controls are then established by the addition of functions which vanish on the boundaries. Functions, such as B-spline basis elements, are ideally suited for this purpose since they can be used in (tensor) product form to vanish everywhere except on a local region of interest. The overall continuation, however, must be positive everywhere, or else a coordinate singularity would result. Now with g_{11} and g_{22} defined everywhere, it would appear that more than the available degrees of freedom were consumed. But on returning to the change of coordinate rule (Eq. 10b) to derive the desired equations for the coordinates, we will find that in the planar case only the cell aspect ratio $\sqrt{g_{11}/g_{22}}$ appears as the specified quantity. From the rule of Eq. 53, the second equation for $g_{12} = 0$ can be split into the Cauchy-Riemann form

$$\frac{\partial x_2}{\partial y_2} = \frac{1}{h}\frac{\partial x_1}{\partial y_1} \qquad (68)$$

$$-\frac{\partial x_2}{\partial y_1} = h\frac{\partial x_1}{\partial y_2}$$

for some positive function h. By use of the other two equations from Eq. 53, we get

$$g_{11} = \left(\frac{\partial x_1}{\partial y_1}\right)^2 + \left(\frac{\partial x_2}{\partial y_1}\right)^2 \qquad (69)$$

$$= \left(h\frac{\partial x_2}{\partial y_2}\right)^2 + \left(-h\frac{\partial x_1}{\partial y_2}\right)^2$$

$$= h^2 g_{22}$$

and hence

$$h = \sqrt{\frac{g_{11}}{g_{22}}} \qquad (70)$$

From the Cauchy-Riemann form in Eq. 68, we can directly compute mixed derivatives for x_2 by differentiating the first equation with respect to y_1 and the second with respect to y_2. A similar computation can also be done for x_1 after h is brought onto the other side of the equations. Upon the equality of corresponding mixed derivatives, we obtain the coordinate generating system

$$\begin{aligned} Dx_1 &= 0 \\ Dx_2 &= 0 \end{aligned} \tag{71}$$

where

$$D = \frac{\partial}{\partial y_1}\frac{1}{h}\frac{\partial}{\partial y_1} + \frac{\partial}{\partial y_2} h \frac{\partial}{\partial y_2}$$

which can be identified as the Laplace operator in (y_1, y_2) and which is linear when h is assumed to be known as it is here.

An alternative coordinate coordinate generating system can also be used with the specified h. To obtain it, we use $g_{11} = h^2 g_{22}$ from Eq. 69 to eliminate g_{11} in the curvature integrability constraint of Eq. 59. The result is the elliptic partial differential equation

$$\frac{\partial}{\partial y_1}\left(\frac{1}{h}\frac{\partial F}{\partial y_1}\right) + \frac{\partial}{\partial y_2}\left(h\frac{\partial F}{\partial y_2}\right) + 2\frac{\partial^2 h}{\partial y_2} = 0 \tag{72}$$

that must be solved for $F = \log g_{22}$ subject to Dirichlet boundary conditions. Upon solution, $g_{22} = e^F$ and $g_{11} = h^2 e^F$ can be used in the line integrals of Eqs. 60-61 to get the coordinates.

Under the observation that the gradient of a Cartesian coordinate x_k is a constant vector field which is then divergenceless, Ryskin and Leal [30] also arrived at the Laplace coordinate generating system of Eq. 71. However, they specified h only in cases where at least one boundary was free to move. There, the pointwise distributions were not specified, but instead were determined by boundary conditions consistent with the Cauchy-Riemann form of Eq. 68 in a pattern very much in the spirit of conformal methods. The boundary movement came with a global iterative cycle. In cases where the boundary and its pointwise distribution were to be fixed, they made h conform to the boundary data in the same manner as in Eiseman [29]; but in contrast, the coordinate equations (Eq. 71) were also put into an iterative loop under the assumption of a nonlinear system. The Laplace system of Eq. 71 clearly becomes nonlinear if h as an unknown is expanded in terms of the metric form given in Eq. 53, and subsequently, inserted into the system. From this viewpoint, the boundary conformity of h is applied at each cycle with boundary

data taken from the current grid in the evolution towards convergence. However, with the evolution of h, the rates of entry or exit into or out of the region become dependent upon the numerical method. The values of h are constant only at the region corners where the boundary data determines it uniquely. The result is that h is method dependent everywhere except at corners. This means that they have essentially arrived at the same situation as in the study of Haussling and Coleman [25] where nonuniqueness can result as in the example associated with Fig. 9. To summarize each cycle, the solution determines boundary h which, in turn, determines the solution.

The primary motivation for the iterative cycles came from the supposition that h could have been selected as unity, in which case, the coordinates would be conformal and would, therefore, have all of the limitations arising from analytic continuation properties. With h chosen to fit the boundary data, a result of h = 1 may not be a real limitation, however, because the pointwise distributions on the boundary may already be consistent with a conformal transformation. This would also be consistent with Eq. 72 and the line integral approach. An affirmative conclusion, however, has yet to be drawn between the noniterative position in Eiseman [29] and the fully iterative one in Ryskin and Leal [30]. Here, it remains to consider various applications to cases with arbitrary data on four boundaries. It would not be surprising if a method based upon a controlled evolution of h should arise.

Rather than a specification of cell aspect ratios as above, a natural quantity to specify is the cell volume which is given by the Jacobian $J = \sqrt{g_{11}\, g_{22}}$. Then g_{11} can be eliminated from the curvature integrability condition of Eq. 59 with $g_{11} = J^2/g_{22}$ in the same manner that we did to get Eq. 72 in the case of specified h. The result is a partial differential equation for g_{22} that is hyperbolic rather than elliptic. As a consequence, specified volume methods must be solved by marching the grid away from a single body. Steger and Sorenson [31] have constructed a direct method from specified J and $g_{12} = 0$ which in expanded form are given by

$$\frac{\partial x_1}{\partial y_1}\frac{\partial x_2}{\partial y_2} - \frac{\partial x_2}{\partial y_1}\frac{\partial x_1}{\partial y_2} = J \tag{73}$$

$$\frac{\partial x_1}{\partial y_1}\frac{\partial x_1}{\partial y_2} + \frac{\partial x_2}{\partial y_1}\frac{\partial x_2}{\partial y_2} = 0$$

A number of interesting grids were generated with this method. A distinct advantage comes from a smaller demand upon the user to specify sensitive data. In comparison with orthogonal trajectory methods there is no need to specify a family of curves which for concave regions would require some cleverness in

order to avoid unwanted clustering on an outer boundary. In its place, a user must specify cell volumes from a simple model grid which is tailored to the application and results in an unclustered outer boundary. The tailoring comes from the same arc length and pointwise distribution for the model body which is connected to a desireable far field structure. The disadvantages of the method are that boundary discontinuities propagate and that the outer boundary cannot be prescribed. In an earlier method, Starius [32] considered a technique based upon the Cauchy-Riemann form of Eq. 68 where h was chosen to make the system hyperbolic. This, however, could not be marched beyond a narrow band from the boundaries. A more complete survey of hyperbolic methods can be found in Thompson, Warsi, and Mastin [33].

CONCLUSION

Subject to various constraints, orthogonal coordinate generation techniques have been examined under the unifying context of the associated metric. The metric was seen to be the basic ingredient for both the generation process and the application of the results.

REFERENCES

1. Laugwitz, D. (1965) Differential and Riemannian Geometry. Academic Press.
2. Caughey, D. A. (1978) Int. J. Num. Meth. Engr. 12, 1651.
3. Thompson, J. F., Thames, F. C. and Mastin, C. W. (1977) J. Comp. Phys. 24, 274.
4. Eiseman, P.R. (1979) J. Comp. Phys. 33, 118.
5. Smith, R. E. and Weigel, B. L. (1980) AIAA Paper 80-0192, AIAA 18th Aerospace Sciences Meeting, Pasadena.
6. Kowalski, E. J. (1980) NASA CP2166, 331.
7. Eiseman, P. R. Coordinate Generation with Precise Controls. ICASE Report 80-30, to appear in J. Comp. Phys.
8. Eiseman, P. R. Lecture Notes in Physics 141. Springer-Verlag, 176.
9. Eiseman, P. R. and Smith R. E. (1980) NASA CP2166, 73.
10. Gordon, W. J. (1971) SIAM J. Num. Anal. 8, 158.
11. Gordon, W. J. and Hall C. A. (1973) Int. J. Num. Meth. in Eng. 7, 461.
12. Watford, R. W. (1978) Imp. College of Science and Tech. London SW7 2BX, Rept. FS/78/43.
13. Gregory, J. A. (1974) Computer Aided Geometric Design. Academic Press, p. 71.

14. Barnhill, R. E. (1977) Mathematical Software III. Academic Press, p. 69.
15. Rizzi, A. and Eriksson L. E. (1981) AIAA Paper 81-0999. Fifth CFD Conf.
16. Sorenson, R. L. and Steger, J. L. (1980) NASA CP2166, 449.
17. Berger, B. S. and Curtis, D. M. ASME J. Appl. Mechanics. In press.
18. Blottner, F. G. and Moreno, J. B. (1980) NASA CP2166, 189.
19. Blottner, F. G. and Ellis, M. A. (1973) Sandia Lab. Res. Rept. SLA-73-0366.
20. Davies, C. W. (1981) J. Comp. Phys. 39, 164.
21. Graves, R.A. (1980) NASA CP2166, 307.
22. McNally, W. D. (1972) NASA TND-6766.
23. Potter, D. E. and Tuttle, G. H. (1973) J. Comp. Phys. 13, 483.
24. Ghia, U., Hodge, J. K., and Hankey, W. L. (1977) AFFDL-TR-77-117, Air Force Flight Dynamics Lab., WPAFB.
25. Haussling, H. J. and Coleman, R. M. (1981) J. Comp. Phys. 43, 373.
26. Flanders, H. (1963) Differential Forms with Applications to the Physical Sciences. Academic Press.
27. Schindler, G. M. The Intrinsic Geometrical and Physical Properties of Plane Steady Gas Flow. SIAM J. Appl. Math., (1982), 42,10.
28. Warsi, Z. U. A. and Thompson, J. F. (1980) NASA CP2166, 516.
29. Eiseman, P. R. (1979) ICASE Int. Rpt. No. 11.
30. Ryskin, G. and Leal, L. G. Submitted to J. Comp. Phys.
31. Steger, J. L. and Sorenson, R. L. (1980) NASA CP2166, 463.
32. Starius, G. (1977) Numer. Math. 28, 242.
33. Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W. Boundary-Fitted Coordinate Systems for Numerical Solutions of PDE's--A Review, J. Comp. Phys., to appear.

AD P000974

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

PATCHED COORDINATE SYSTEMS

P. E. RUBBERT and K. D. LEE
Boeing Commercial Airplane Company
Seattle, Washington 98124

INTRODUCTION

Numerical grid generation has been an excellent tool for producing curvilinear body-fitted coordinate systems. Curvilinear coordinate systems or grids are commonly used in the solution of partial differential equations in domains surrounding arbitrary geometrical boundary shapes, as reviewed in Ref. 1. Body-fitted grids are particularly advantageous in the treatment of surface boundary conditions, and usually yield a degree of simplicity in the logic required to solve the hosted partial differential equations.

In practice, numerical grid generation usually involves transformation of the physical domain of interest into a geometrically simple domain, such as a rectangular block or assembly of blocks. The solution of grid generation equations in the simple domain produces the coordinates of a corresponding grid in the physical domain, subject to a variety of grid control procedures aimed at producing favorable grid characteristics. This process is usually straightforward when the topology of the physical domain is simple enough to allow transformation to a single rectangular domain. But when dealing with geometrically and topologically complex domains such as surround an aircraft configuration, (the authors' work has dealt principally with the solution of partial differential equations governing the flow field about aircraft configurations), the total issue of grid generation becomes more complex. The domain in general cannot be mapped into a single block. The configuration surface geometry itself may be nonanalytic, and these features will be manifest in any grid surrounding such complex boundary shapes.

MULTI-BLOCK STRUCTURING

One approach to organizing coordinate systems for complex domains, espeically in three dimensions, is to divide the domain into a number of geometrically simple subdomains, termed blocks. The grid within each block can then remain of simple character, such as rectangular. We term the grid

PREVIOUS PAGE IS BLANK

comprised of the assembly of such subdomain grids a block structured or multi-block grid. Figure 1 shows both single and multi-block grids in the domain about an airfoil.

The multi-block concept provides a general capability for surface-fitted grids in simple or complex domains. Figures 2 and 3 demonstrate typical multi-block representations of a wing/body/nacelle/strut configuration. Each component of the solid configuration is mapped into a simple rectangular block in the computational space. The computational domain then consists of the remainder of space surrounding the configuration blocks. This space can be divided in an obvious way into an assembly of rectangular blocks.

In the transformation, physical corners appearing at intersections between components (such as between wing and fuselage) are kept as corners in computational space. However, the block structure introduces some new types of special points as seen in the figures. A point on the smooth boundary surface in physical space can appear as a corner point in computational space (termed a "fictitous corner"). Some gridlines may merge together as shown at the wing-tip (termed a "collapsed edge"). It is therefore necessary that the hosted algorithm be adaptable to these aspects of the grid.

APPROACHES TO STRUCTURING

At least two approaches are commonly used to generate multi-block grids. One is to generate the grid separately within each rectangular block. In this case, certain of the block boundary surfaces no longer correspond to boundary surfaces of the original problem. They are just boundaries separating one block from another, and we call them field boundaries. Nevertheless, solution of the grid generation equations require grid boundary conditions on these field boundaries, and these are usually supplied by the user. Grids generated in this manner are termed patched grids or patched coordinate systems. There is no built-in feature leading to any degree of grid continuity across a field boundary separating one block from another. Some control of grid continuity is available through proper choice of grid boundary conditions on the field boundaries.

Another approach to generating multi-block grids is to solve the grid generation equations in the entire block-structured domain as a single grid generation problem. This adds a degree of complexity in that the grid generation equations must be solved in a domain comprised of an assembly of blocks, rather than one block at a time. In our experience, this has not been

difficult to implement. One consequence of this approach is that the locations of block boundaries in physical space (field faces) emerge as a solution of the grid generation problem rather than an input. Also, all grids will be analytic across field faces. We term grids generated in this manner as directly solved multi-block grids.

Some of the differences between these two approaches are illustrated in Figures 4 and 5. In practice, a combination of these two approaches frequently provides the degree of flexibility desired for many problems of interest.

Figure 5 illustrates some characteristic differences between patched and directly-solved multi-block grids. With patching, the block boundaries, $\overline{CE}$ and $\overline{CF}$, are supplied as boundary conditions to the 3D grid generation process. Grid analyticity across block boundaries is not guaranteed since volume grids on both sides of a block boundary are generated separately. Grid control features are therefore sometimes implemented to achieve a degree of analyticity across block boundaries. In the directly solved multi-block system, grids are automatically analytic across all block boundaries in the field, for example, $\overline{GI}$ and $\overline{GJ}$, but grids may still be nonanalytic across surface perimeter lines. However, the grid spacing near fictitious corners cannot be as easily controlled as in the patched system, leading to grids that are less desirable in this aspect. In practice, both methods of grid generation have merits and limitations.

The issues of grid generation cannot be addressed without considering the adaptability of the hosted algorithms, i.e., the algorithms for solving the partial differential equations. Some algorithms require smooth analytic grids. Others do not. The adaptability of the hosted algorithm therefore imposes requirements on grid quality, and vice versa. For many classes of problems there can exist a tradeoff between hosted algorithm requirements and grid requirements. The two are intimately related. The present paper offers some philosophy on grid versus hosted algorithm requirements, as well as a discussion of multi-block grid generation techniques.

THE OVERALL GRID GENERATION PROCESS

3D grid generation via patching is shown in Figure 6 where the illustrations represent 2D cuts for a wing/body configuration both in physical and computational space. The procedure starts with dividing physical space into six-sided blocks by considering the structure of configuration

components. The next step is to discretize the block perimeter lines which connect block corner points. This 1D-like discretization is illustrated by dots through which the perimeter lines penetrate the plane of the figure. The 1D discretization provides boundary conditions for a subsequent 2D-like grid generation producing grids covering the block boundary surfaces. These in turn serve as boundary conditions to produce 3D volume grids filling each block. Grid control can be achieved at any of the above four steps, i.e., through the block pattern, perimeter discretization, surface grid generation and/or volume grid generation.

Grid generation via the direct method is shown in Figure 7. Since the coordinates of field block boundaries emerge as part of the solution to grid generation in the direct method, we divide only the natural boundaries such as configuration surfaces and exterior boundaries into four-sided patches in order to map the configuration components into rectangular blocks in the transformed space. Then the perimeter lines enclosing patches on physical surfaces are discretized appropriately in a way to produce a good grid distribution. The subsequent 2D-like grid generation is carried out only on the natural boundary surface patches. Next, volume grids are generated at once everywhere in the computational domain. Therefore, the domain where the grid generation equations are solved is not a simple rectangular box; it is a large box containing an interior cutout comprising the rectangular blocks representing the configuration.

PERIMETER LINE DISCRETIZATION (1D)

The discretization of perimeter lines can be controlled by using different spacing distributions. Uniform spacing, algebraic or geometric stretching, and distribution proportional to a reference distribution can be applied along the arc length or along each coordinate separately to provide good boundary fitting behavior.

SURFACE GRIDS (2D)

We identify two different types of block boundary surfaces; configuration surface and field block boundary. The 2D discretization of configuration surfaces is perhaps the most difficult task in 3D grid generation. The grid points must lie on the specified geometry and display characteristics of smoothness, etc., acceptable to the hosted solution procedure. The surface grids should also provide good behavior at block boundary abutments with

neighboring surface patches. Field block boundary discretization is an easier problem in that the block boundary need not be coincident with a specified geometry except at the edges. Only smoothness and proper discretization are usually required.

2D grid generation of a curved configuration surface in 3D space usually cannot be achieved in one step due to the constraint that the grid points should lie on a specified geometry. A promising approach is the use of a parametric transformation. The 3D surface in (x,y,z) coordinates is defined in terms of 2D parametric coordinates, say (u,v). The discretization process is then carried out in the parametric coordinates by using either analytical or numerical grid generation schemes. Then the Cartesian coordinates of the grid points are extracted. For example, a parametric transformation is commonly used in the discretization of wing surfaces. The chordwise lines are mapped into one parametric coordinate using a parabolic transformation and the spanwise lines into the other parametric coordinate by a shearing transformation. It then becomes a simple matter to specify a discretization that produces smooth grids of desired quality.

Another approach is the use of projection. Discretization is obtained first on a simpler geometric surface by numerical or analytical grid generation schemes. Then the discretized points are projected onto the real, curved surface geometry to obtain the grid coordinates. For example, a discretization of a fuselage surface can be obtained by generating grids on a circular cylinder and projecting them radially into the surface definition.

On a field block boundary, grid points are not usually required to fall on a defined surface. Therefore, one can adopt a relatively simple numerical grid generation scheme. The scheme we use is an extension of the 2D grid generation method developed by Thompson, et al (Ref. 1).

In the standard Thompson approach for planar surface discretization, the 2D coordinates, $\overrightarrow{X} = (x,y)$ are extracted from the solution of the following grid generation equations subject to specified Dirichlet boundary conditions. With indices (ξ, η) as independent variables, they are

$$(\nabla\xi)^2(\overrightarrow{X}_{\xi\xi}+P\overrightarrow{X}_{\xi}) - 2(\nabla\xi\cdot\nabla\eta)\overrightarrow{X}_{\xi\eta} + (\nabla\eta)^2(\overrightarrow{X}_{\eta\eta}+Q\overrightarrow{X}_{\eta}) = 0 \qquad (1)$$

where

$$(\nabla\xi)^2 = \overline{X}_\eta^2 / h^2$$
$$(\nabla\eta)^2 = \overline{X}_\xi^2 / h^2$$
$$\nabla\xi\cdot\nabla\eta = \overline{X}_\xi \cdot \overline{X}_\eta / h^2$$

and h is the transformation Jacobian which can be factored out in the equations. The extension for use on nonplanar curved field block boundaries involves the introduction of a third index, ζ, which remains constant on the surface. The same grid generation equations are used with $\overline{X} = (x,y,z)$, i.e., three equations instead of two. Accordingly $\overline{X}_\xi = (x_\xi, y_\xi, z_\xi)$ in the nonplanar case instead of $\overline{X}_\xi = (x_\xi, y_\xi)$ in the planar case, etc.

The grid control parameters, P and Q, are defined along perimeters whose coordinates are given (Ref. 2). That is, P is derived from constant η boundaries and Q from constant ξ boundaries by letting respectively

$$\overline{X}_{\xi\xi} + P\overline{X}_\xi = 0 \qquad (2)$$

$$\overline{X}_{\eta\eta} + Q\overline{X}_\eta = 0.$$

Any one coordinate or arc length distribution can be chosen to define the grid control parameters. Also one can use the control parameter defined along one perimeter for the whole domain or interpolate linearly between two facing boundaries. Interpolation gives smoothly varying grids whereas the one-side choice reduces grid nonanalyticity of grid lines across corresponding block boundaries.

VOLUME GRIDS (3D)

Volume grid generation inside a block is a straightforward and relatively easy task compared to surface discretization. One approach is to solve the system of nonlinear 3D grid generation equations introduced by Thompson *subject to Dirichlet boundary conditions*

$$(\nabla\xi)^2(\overline{X}_{\xi\xi}+P\overline{X}_\xi) + (\nabla\eta)^2(\overline{X}_{\eta\eta}+Q\overline{X}_\eta) + (\nabla\zeta)^2(\overline{X}_{\zeta\zeta}+R\overline{X}_\zeta)$$
$$+ 2(\nabla\xi\cdot\nabla\eta)\overline{X}_{\xi\eta} + 2(\nabla\xi\cdot\nabla\zeta)\overline{X}_{\xi\zeta} + 2(\nabla\eta\cdot\nabla\zeta)\overline{X}_{\eta\zeta} = 0 \qquad (3)$$

where

$$\nabla\xi = (\vec{X}_\eta \times \vec{X}_\zeta)/h$$

$$\nabla\eta = (\vec{X}_\zeta \times \vec{X}_\xi)/h$$

$$\nabla\zeta = (\vec{X}_\xi \times \vec{X}_\eta)/h.$$

Again the transformation Jacobian h can be factored out. Similar to surface grid generation, the grid control parameters, P, Q, and R, are extracted from known boundary data. For example, the control parameter R is defined along four perimeter lines with constant ξ and η indices by letting

$$\vec{X}_{\zeta\zeta} + R\vec{X}_\zeta = 0. \tag{4}$$

Then it is interpolated through ξ and η indices or fixed by a choice of indices to give consistent grid control at block boundaries. Other control parameters can be extracted similarly using Equation (2).

The grid generation equations, either surface or volume, are solved iteratively by using successive line overrelaxation (SLOR) or alternating direction implicit (ADI) methods. The computational costs for the 3D generation of patched coordinate systems is less than for directly solved systems because these solution methods are faster on smaller domains. This cost can be offset by the added work required to separately generate the field surface grids.

Another approach to volume grid generation is the use of linear grid generation equations (Ref. 3). Instead of updating the nonlinear coefficients appearing in the grid generation equations, Eqns. (1) and (3), the coefficients can be established on boundaries and interpolated throughout the domain. The cross derivative terms appearing in the grid generation equations can be deleted for simplicity. This leaves a very simple, linear, elliptic set of grid generation equations to solve. Our limited experience with both methods is that the latter is less costly whereas the former seems to offer more direct and effective means for grid control.

SOME OBSERVATIONS ON REQUIREMENTS FOR GRID QUALITY

In the real world of complex three-dimensional problems such as the domain surrounding a fairly complete airplane, there exists a conflicting set of issues and requirements. The developer of hosted algorithms would like grids that are smooth, analytic, and topologically simple, with grid spacing in accordance with the length scales of the hosted problem of interest. The customer who will use the capability being produced wants to get by with the easiest grid to generate, with minimal concern about grid irregularities and their possible impact on the accuracy and reliability of the hosted solution. And management generally wants the least expensive approach.

There is a tradeoff between demands placed on grid quality and demands imposed by the hosted algorithms. Some hosted algorithms demand smooth, analytic grids. Others are more forgiving of the grid. For many 2D problems, the flexibility exists to go either way.

However, 3D grids frequently cannot be analytic in real life engineering problems. An example illustrating this is shown in Figure 8, which depicts a surface discretization used in an actual flow field calculation (requiring only a surface and not a volume grid). Current trends are to seek solutions of hosted algorithms requiring volume grids, and it is easy to visualize that the volume grid about this airplane would not be smooth and analytic everywhere. Smooth, analytic volume grids cannot be obtained from nonsmooth nonanalytic surface grids since surface grids become boundary conditions for the volume grid generation. For this configuration and many others like it, smooth surface grids are judged to be impossible. For example, Figure 9 shows the fairing at a wing/body juncture. The surface itself has creases and slope discontinuities, and their influence on the hosted solution is a desired quantity that cannot be smoothed by fairing a smooth grid over such areas. Nonanalytic coordinate lines and nonsmooth grid stretching cannot be avoided when the geometry involves angular abutment between different components.

Hence, the authors are led to the opinion that for complex geometries such as that shown, one must demand hosted algorithms that are forgiving of grid irregularities.

Our experience indicates that hosted algorithms which work reliably on nonsmooth grids can be developed if that is recognized as a goal in algorithm development. For example, a higher order panel method was developed for solving the 3-D Laplace equation (Ref. 4) with that as a goal. It worked so well that acceptable hosted solutions were produced from discretizations

defined by a random number generator (Fig. 10). Finite element methods that give good solutions on nonsmooth grids are well known. Figure 11 demonstrates algorithm compatability to grid irregularities in solving the potential flow over a cylinder (Ref. 3), using a cell-oriented finite volume formulation designed to be forgiving. Resolved are the algorithm issues concerning fictitious corners and nonanalytic coordinates across block boundaries appearing in multi-block grids.

Forgiving algorithms have been developed for a variety of other equations in aerodynamics. Figure 12 shows a panel method solution to the wave equation for a spindle paneled in a random manner (Ref. 5). The forgiving algorithms cannot only adapt to grid irregularities but also provide capabilities for real life geometry (Fig. 13, Ref. 6). Forgiving and unforgiving algorithms are compared in Figure 14 for the Helmholtz equation.

Demands for forgiving algorithms have also extended into the solution of complex nonlinear transonic flows. Figure 15 shows transonic solutions obtained from a forgiving algorithm for the nonlinear, mixed-flow potential equation (Ref. 7). Patched multi-block grids were generated by allowing intentionally severe grid nonanalyticity across block boundaries to test the robustness of the hosted algorithm. The accuracy of the results is remarkable, even when fictitious corners are located inside of the supersonic region. The discrepancy in shock location is due to the use of different upwind bias in supersonic cells and is unrelated to the grid. 3D transonic results using multi-block grids are reported in Ref. 8.

SUMMARY

In many cases of practical interest, the irregularities present in the boundary surface render the achievement of smooth, regular volume grids an impossible task. One must require hosted algorithms that are compatible with grid irregularities. Evidence is accumulating that this can generally be achieved if made an algorithm design requirement.

The use of patched coordinate systems is a feasible and systematic way of generating orderly grids for complex configurations. They appear to offer an acceptable degree of grid control. And requirements for grid control are really not very stringent. The fact that one must have forgiving hosted algorithms to handle complex geometries at all means that a modest amount of grid control is all that is usually needed to achieve an acceptable grid.

REFERENCES

1. Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W. (1982) Boundary-fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review, to be published in Journal of Computational Physics.
2. Thomas, P. D. and Middlecoff, J. F. (1980) Direct Control of the Grid Point Distribution in Methods Generated by Elliptic Equations," AIAA Journal, Vol. 18, No. 6.
3. Lee, K. D., et al (1980) Grid Generation for General Three-Dimensional Configurations, Numerical Grid Generation Techniques, NASA Conference Publication 2166, p. 355-366.
4. Johnson, F. T. and Rubbert, P. E. (1975) Advanced Panel-Type Influence Coefficient Methods Applied to Subsonic Flows, AIAA Paper 75-50.
5. Ehlers, F. E., Johnson, F. T., and Rubbert, P. E. (1976) A Higher-Order Panel Method for Linearized Supersonic Flow, AIAA Paper 76-381.
6. Cenko, A., et al (1981) PAN AIR Applications to Weapons Carriage and Separation, J. of Aircraft, Vol. 18, No. 2, p. 128-134.
7. Lee, K. D. and Rubbert, P. E. (1980) Transonic Flow Computations using Grid Systems with Block Structure, 7th International Conference on Numerical Methods in Fluid Dynamics, p 266-271.
8. Lee, K. D. (1981) 3-D Transonic Flow Computations using Grid Systems with Block Structure, AIAA 5th Computational Fluid Dynamics Conference.

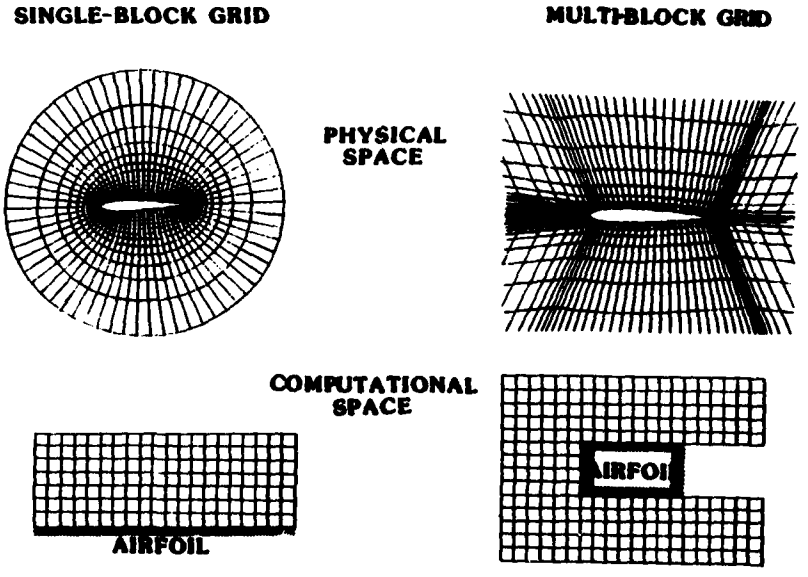


Fig. 1 Terminology; single-block grid versus multi-block grid.

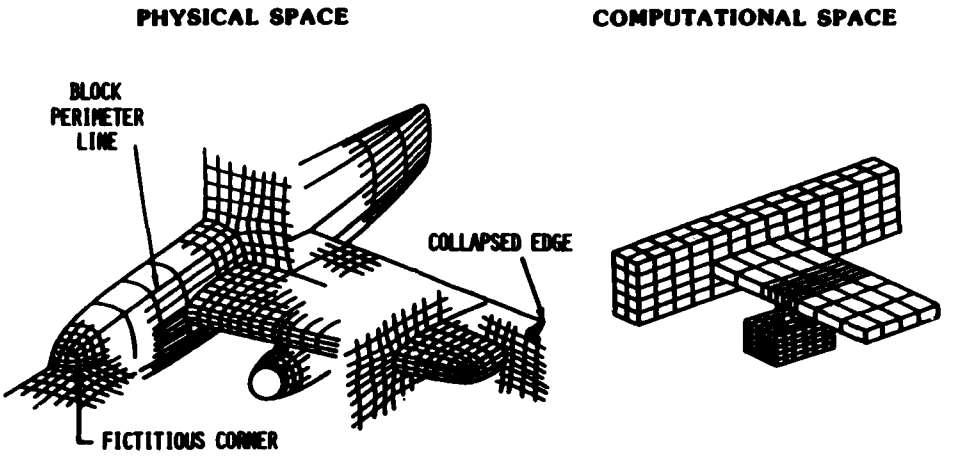


Fig. 2 Multi-block representation of a wing/body/nacelle/strut configuration.

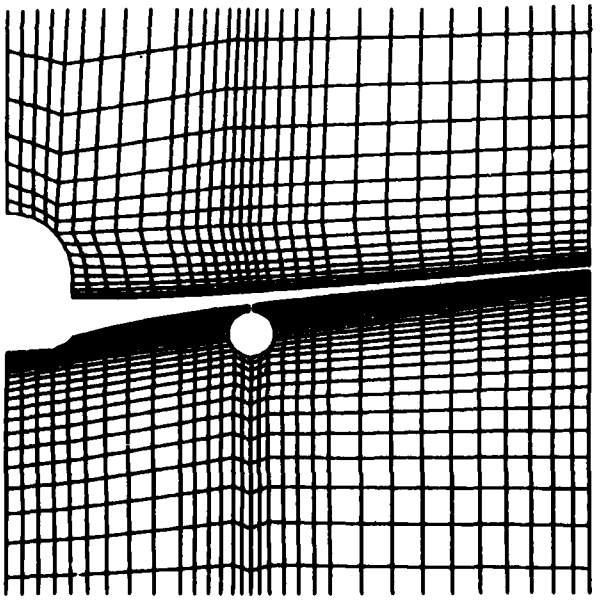


Fig. 3 Example of a 3D multi-block grid for a wing/body/nacelle/strut.

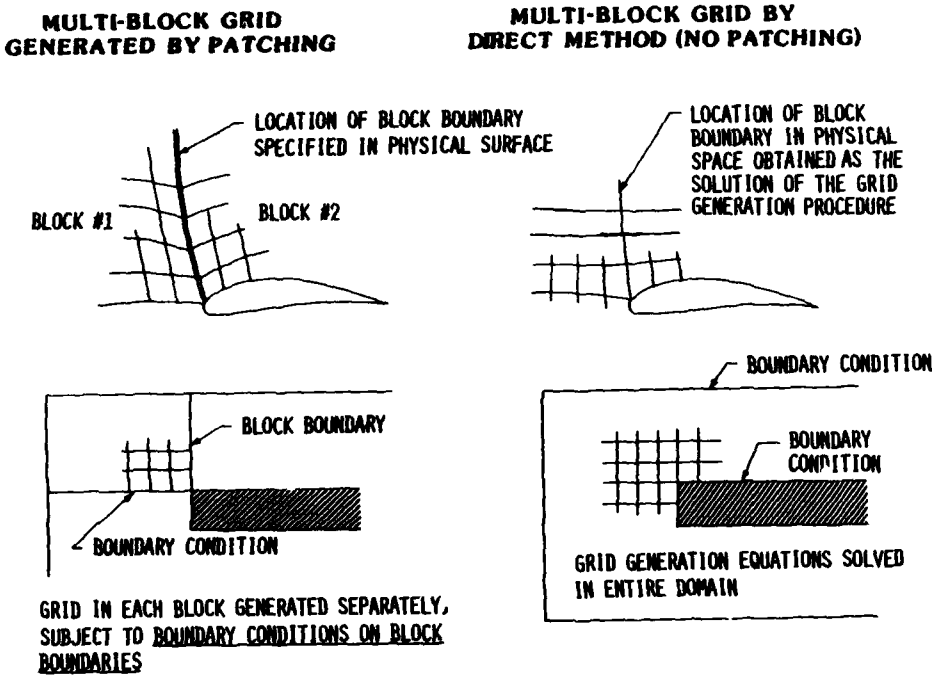


Fig. 4 Multi-block grids; patched versus directly solved.

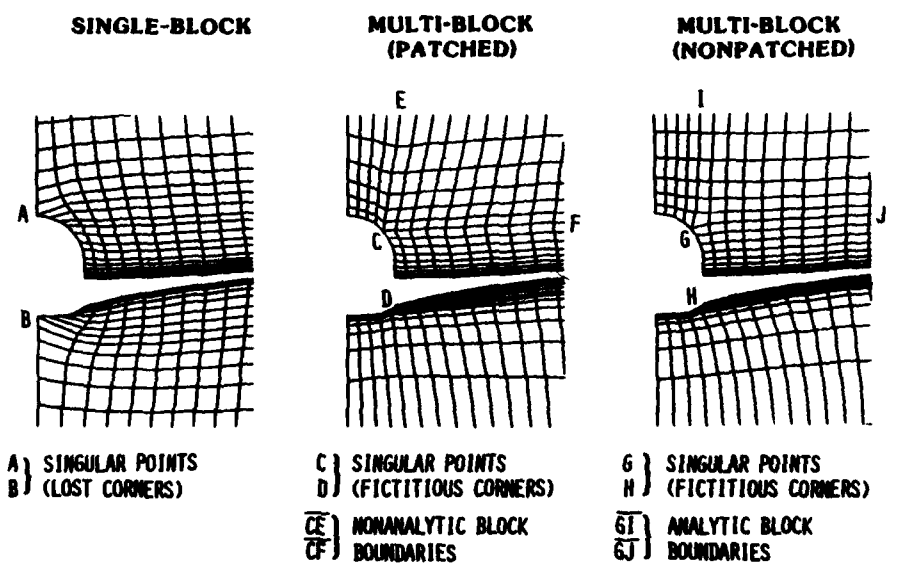
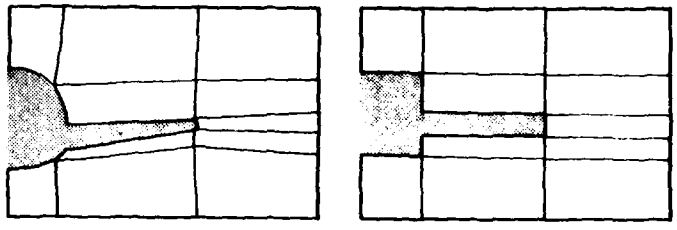
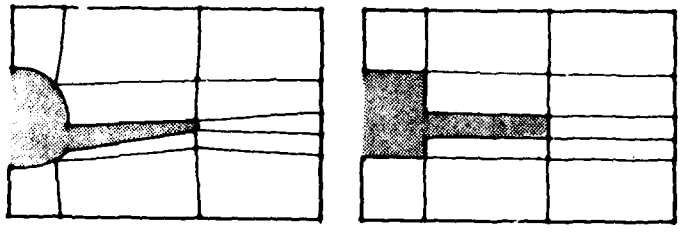


Fig. 5 Comparisons of grid characteristics for a wing/body configuration.

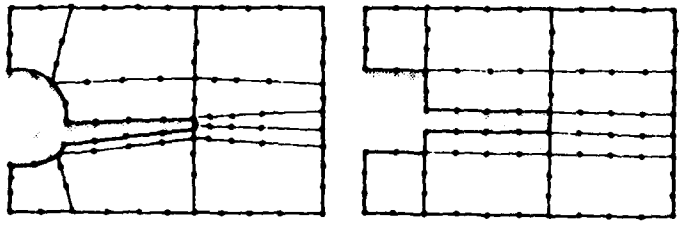
STEP 1: DIVIDE PHYSICAL SPACE INTO 6-SIDED BLOCKS



STEP 2: DISCRETIZE PERIMETER LINES



STEP 3: GENERATE 2-D GRIDS ON ALL BLOCK SURFACES



STEP 4: GENERATE VOLUME GRIDS WITHIN EACH BLOCK

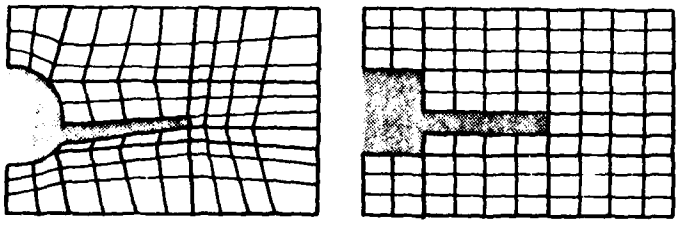
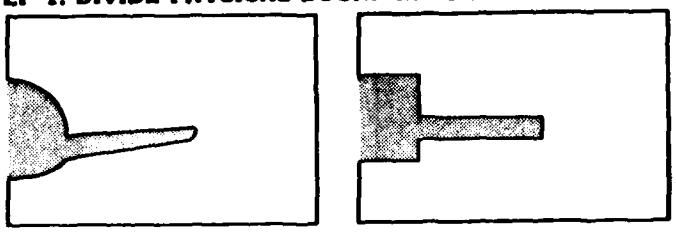
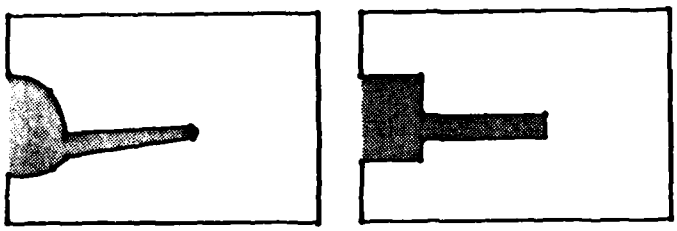


Fig. 6 Grid generation process via patching (illustrations represent 2D cuts for a wing/body).

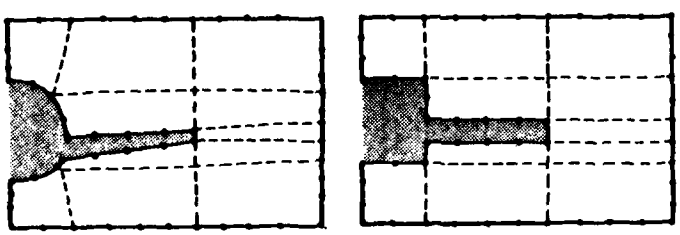
STEP 1: DIVIDE PHYSICAL BOUNDARIES INTO 4-SIDED PATCHES



STEP 2: DISCRETIZE PERIMETER LINES OF PATCHES ON PHYSICAL SURFACES



STEP 3: GENERATE 2-D GRID ON PATCHES (PHYSICAL SURFACES AND EXTERIOR BOUNDARIES ONLY)



STEP 4: GENERATE VOLUME GRID EVERYWHERE AT ONCE

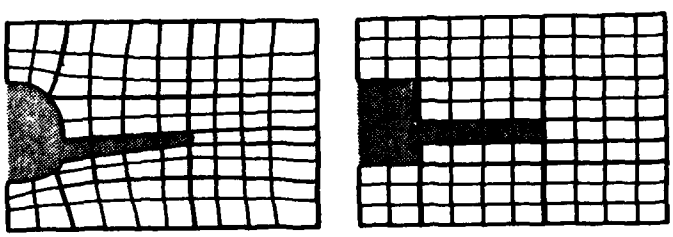


Fig. 7 Grid generation process via the direct method (illustrations represent 2D cuts for a wing/body).

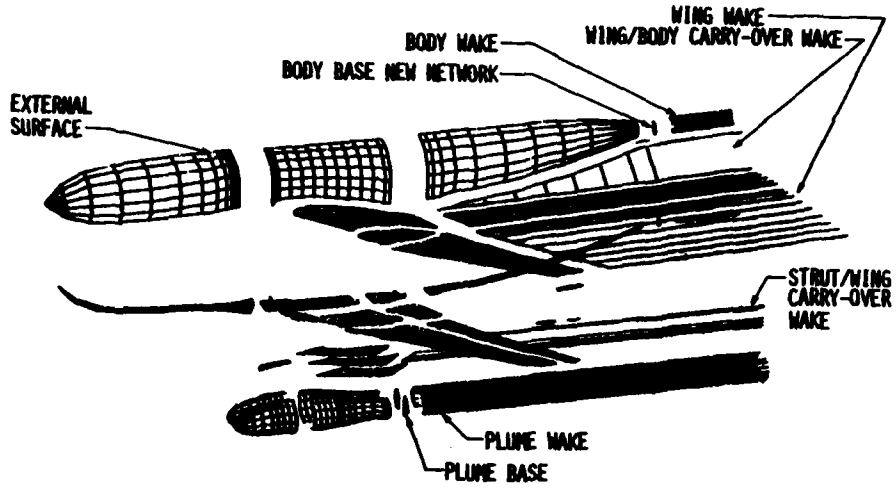


Fig. 8 Complexity of geometry in real life engineering problems.

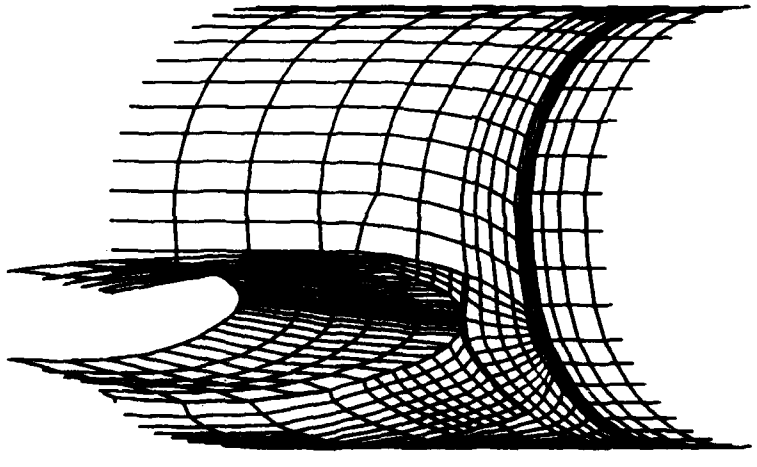


Fig. 9 Fairing at a wing/body juncture; smooth surface grids are impossible.

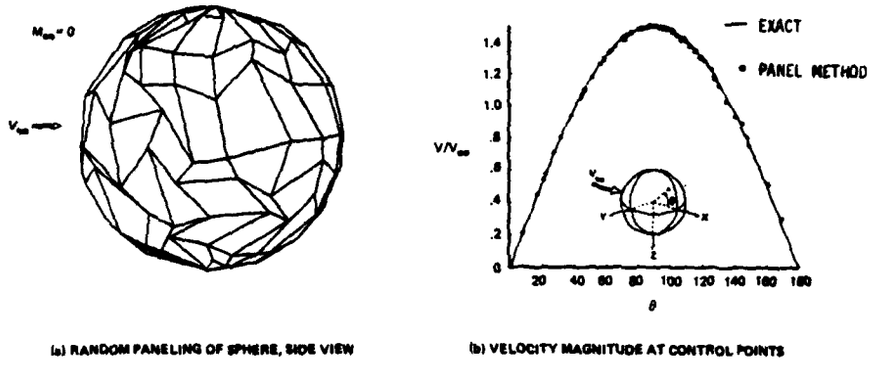


Fig. 10 Example of a forgiving algorithm; a higher order panel method for the Laplace equation.

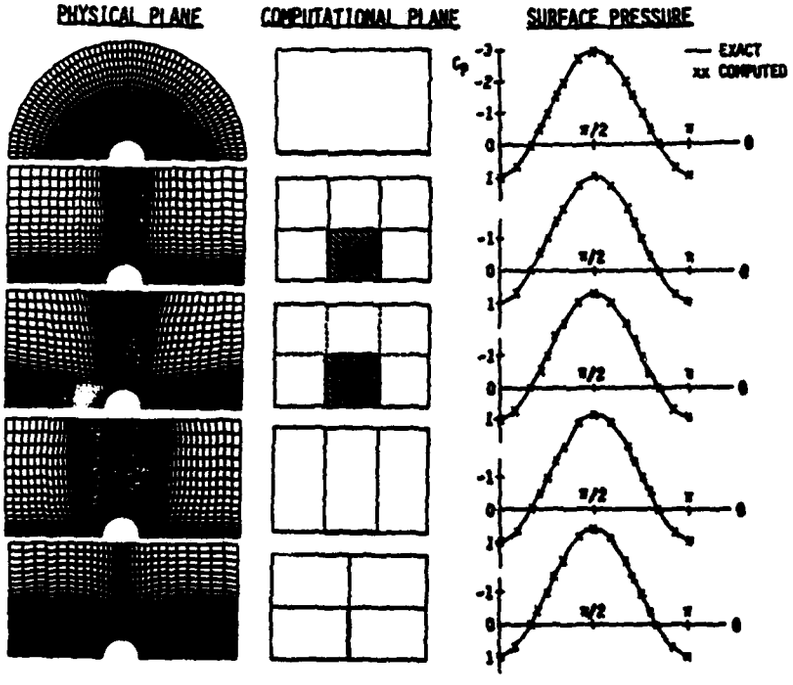
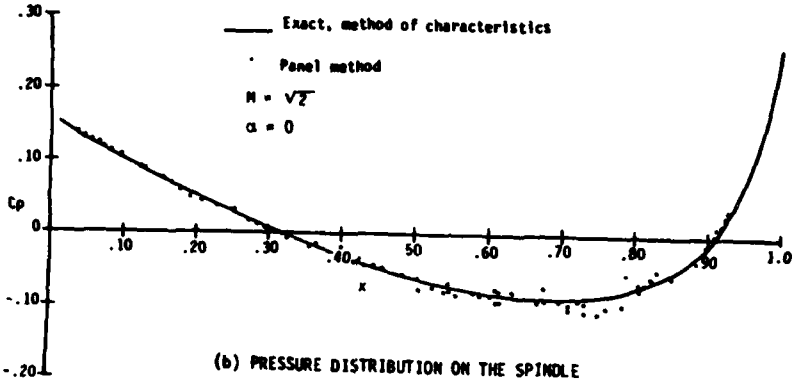


Fig. 11 Example of a forgiving algorithm; a field grid method for the Laplace equation.

(a) RANDOM PANELING OF A SPINDLE



(b) PRESSURE DISTRIBUTION ON THE SPINDLE

Fig. 12 Example of a forgiving algorithm; a panel method for the wave equation.

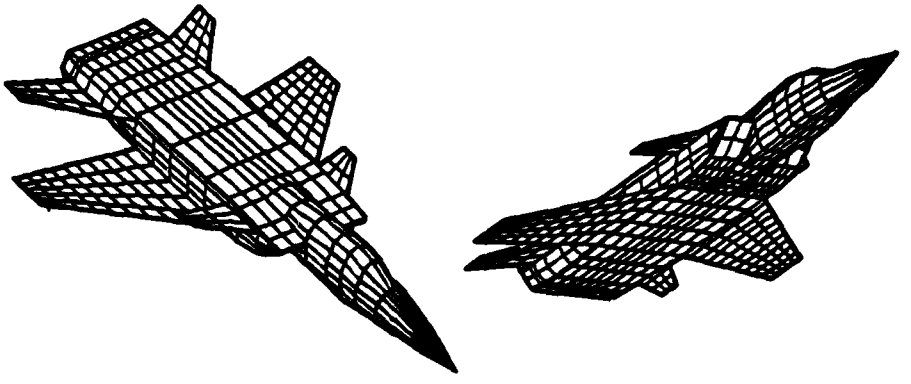
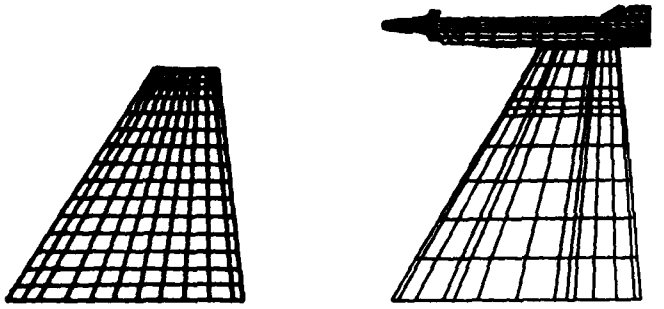


Fig. 13 Forgiving algorithms provide capabilities for real life geometry.



GEOMETRY CONSTRAINTS IMPOSED BY ALGORITHM TECHNOLOGY NOT DESIGNED TO BE FORGIVING

LATER NUMERICAL TECHNOLOGY

Fig. 14 Capability limits imposed by forgiving and unforgiving algorithms for the Helmholtz equation.

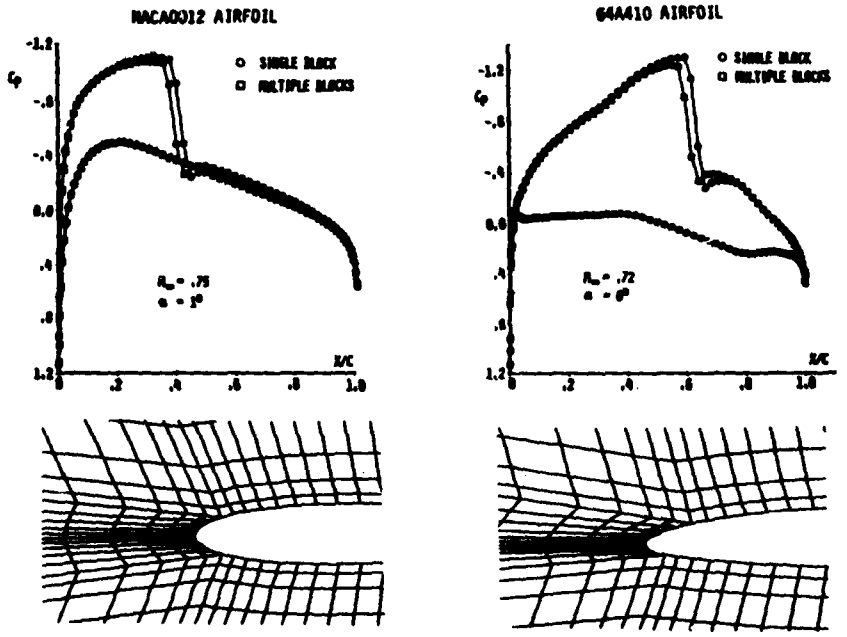


Fig. 15 Transonic solutions obtained from a forgiving algorithm.

AD P000975

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

SOLID MECHANICS APPLICATIONS OF BOUNDARY FITTED COORDINATE SYSTEMS

JOHN C. McWHORTER
Department of Aerospace Engineering, Mississippi State University, P. O. Drawer A, Mississippi State, Mississippi 39762

ABSTRACT

A numerical method which utilizes the boundary fitted coordinate method is presented and applied to the solution of three solid mechanics problems. The three problems considered are the elastic torsion of uniform shafts of arbitrary cross section, the elastic torsion of non-uniform shafts of arbitrarily varying circular cross section, and the bending of thin isotropic elastic plates of arbitrary shape with simple or clamped boundaries. The boundary fitted coordinate method is utilized to transform the arbitrary simply connected or multiply connected region under study onto a fixed rectangular domain where computations are easily done. The governing equations and boundary conditions are transformed and solved on the rectangular domain by SOR iteration. Numerical results for all three problems show close agreement with analytical solutions, although there is local error introduced when the coordinate system is severely skewed. Numerical results check closely with experimental results obtained for problems which have no analytical solution.

INTRODUCTION

The concept of boundary fitted coordinates can be applied to many boundary value problems, and it is the purpose of this paper to present the results of three applications in the general area of solid mechanics. The three applications are (1) the torsion of uniform shafts of arbitrary cross section, (2) the torsion of non-uniform shafts of arbitrary profile, and (3) the bending of thin plates of arbitrary shape. Each application has a different governing equation and different boundary conditions, and each abounds with problems having exact analytical solutions which provide a comparison with the numerical results generated by the application of the boundary fitted coordinate method.

The solid mechanics theory for each application is presented briefly to explain the governing equation and boundary conditions. The governing equation and boundary conditions are then transformed for solution in the boundary fitted coordinate system. Transforming the equations is independent of the particular problem to be solved and must be done only once for each application. The governing equation and boundary conditions are expressed as finite differences

in the transformed coordinate system and the resulting set of algebraic equations is solved by SOR iteration. Thus all computations, both to generate the boundary fitted coordinate system and to solve the governing equation, are done on a rectangular grid of uniform spacing with no interpolation necessary on the boundaries. This leads to the ability to generate a "black box" code where the only input necessary is the shape of the shaft for torsion problems. Plate problems also require a description of the transverse loading and an identification of the type of boundary condition at each boundary point describing the plate shape.

A general code was developed for torsion problems which accurately calculated the torsional shear stress in both uniform and variable profile shafts. A general code for plate problems was developed for simple and clamped boundaries. No work was done on plates with free edges. Comparison between classical solutions and numerical solutions generated by these codes are presented and show excellent agreement in most cases.

The coordinate system generation scheme used in this work is the same as that developed by Thompson, et al.[1] Generally, the coordinate lines were attracted to regions of suspected high gradients, and some work was done to determine the effect of different coordinate spacings on the accuracy of the results.

UNIFORM SHAFTS - ARBITRARY CROSS SECTION

The torsion of uniform shafts is a classical elasticity problem which has been solved analytically for many geometric shapes both directly and by conformal transformation. However, certain classes of cross sections with re-entrant corners cannot be solved exactly. Since the governing equation and boundary conditions are simple, this problem was worked first to demonstrate the utility and validity of the boundary fitted coordinate method.

In terms of a warping function the governing equation is the Laplace equation and the boundary condition is that the tangential derivative of the warping function is known on the boundary. If a modification to the warping function is made, the governing equation becomes the Poisson equation, and on the boundary the modified warping function is zero. This is the same condition that exists for a thin membrane stretched over a hole of the shape of the shaft and inflated by a uniform pressure and is known as the membrane analogy for the torsion of uniform shafts. It can be shown that the torque on the shaft is proportional to the volume under the inflated membrane and that the shear stress in the shaft is proportional to the normal derivative of the deflected membrane and is directed normal to the direction of maximum gradient.

If interior holes are present in the shaft cross section, the membrane analogy still holds if a weightless plate of the shape of the interior hole is "floated" on the deflected membrane and held level by distributed couples around the plate. The elevation of this plate must be found by a force balance on the plate in which the pressure below the plate is balanced by the membrane tension integrated around the perimeter of the plate. A more detailed description of the membrane analogy is given by Den Hartog.[2]

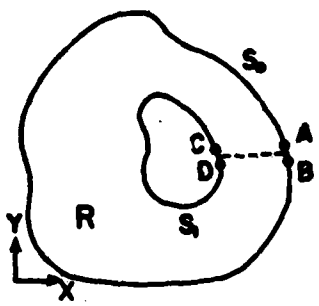


Fig. 1. Physical plane.

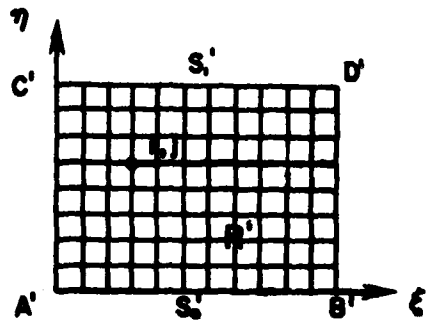


Fig. 2. Transformed plane.

Equations for the membrane analogy

Figures 1 and 2 show the cross section of a typical shaft in the physical and transformed coordinate systems. In the physical system the governing equations and boundary conditions are as follows:

$$\nabla^2 w = -2 \quad \text{on } R \tag{1a}$$

$$w = 0 \quad \text{on } S_o \tag{1b}$$

$$F \equiv \int_{S_1} \frac{\partial w}{\partial n} dS_1 - 2A = 0 \quad \text{on } S_1 \tag{1c}$$

$$\tau_x = \frac{\partial w}{\partial y}/J_t \tag{2a}$$

$$\tau_y = -\frac{\partial w}{\partial x}/J_t \tag{2b}$$

where J_t is the torsion constant and is given by

$$J_t = 2\iint w \, dx \, dy \, . \tag{3}$$

After transformation (for details see Thames[3]) these equations become:

$$\alpha w_{\xi\xi} - 2\beta w_{\xi\eta} + \gamma w_{\eta\eta} + \tau w_{\xi} + \sigma w_{\eta} = -2J^2 \text{ on } R' \tag{4a}$$

$$w = 0 \quad \text{on } S'_o \tag{4b}$$

$$F \equiv \int_{S'_1} \frac{1}{J}(\beta w_{\xi} - \gamma w_{\eta})d\xi - 2\int_{S'_1} y(\xi)x_{\xi}d\xi = 0 \quad \text{on } S'_1 \tag{4c}$$

where

$$\alpha \equiv x_{\eta}^2 + y_{\eta}^2 \tag{5a}$$

$$\beta \equiv x_{\xi}x_{\eta} + y_{\xi}y_{\eta} \tag{5b}$$

$$\gamma \equiv x_{\xi}^2 + y_{\xi}^2$$

$$J = x_{\xi}y_{\eta} - x_{\eta}y_{\xi} \tag{5d}$$

$$\sigma \equiv [y_{\xi}(Dx) - x_{\xi}(Dy)]/J \tag{5e}$$

$$\tau \equiv [x_{\eta}(Dy) - y_{\eta}(Dx)]/J \tag{5f}$$

where

$$Dx = \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \tag{5g}$$

$$Dy = \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} \tag{5h}$$

Since the values of x and y are known from the coordinate transformation at each grid point in the (ξ,η) transformed field, equations (5) can be computed by expressing the derivatives with respect to ξ and η as finite differences on the rectangular grid of the transformed field. This must be done only once for a particular transformation. Next the derivatives in equation (4a) are expressed as central finite differences in the transformed field, which produces an algebraic equation at each node point in the transformed field.

$$\begin{aligned} w_{i,j} = {} & D_1(w_{i-1,j+1} + w_{i+1,j-1} - w_{i+1,j+1} - w_{i-1,j-1}) \\ & + D_2 w_{i,j+1} + D_3 w_{i-1,j} + D_4 w_{i+1,j} + D_5 w_{i,j-1} + D_6 \end{aligned} \tag{6}$$

where

$$D_1 = \beta/4(\alpha+\gamma) \tag{7a}$$

$$D_2 = (\gamma+\sigma)/2(\alpha+\gamma) \tag{7b}$$

$$D_3 = (\alpha-\tau)/2(\alpha+\gamma) \tag{7c}$$

$$D_4 = (\alpha+\tau)/2(\alpha+\gamma) \tag{7d}$$

$$D_5 = (\gamma-\sigma)/2(\alpha+\gamma) \tag{7e}$$

$$D_6 = J^2/(\alpha+\gamma) \tag{7f}$$

If the shaft is solid, the transformation from the physical to transformed field does not have a branch cut as shown in Figure 1, and the outer surface S_o maps onto all four sides of the transformed field so that boundary condition (1c) is not needed. Boundary condition (1b) is then applied on all boundaries in the transformed field and equations (6) can be solved by SOR techniques subject to zero value of w on all boundaries. Thus the only difference from one problem to another lies in the generation of the coordinate system for the particular shaft cross section.

For a shaft with a hole in the interior, boundary condition (1c) must be satisfied by an iterative procedure as follows. Assume a value w_B for the deflection of the weightless plate which is the value of w along the boundary S_1. Knowing that the value of w on A'C' must match the corresponding value on B'D', equations (6) can be solved by SOR iteration. The resultant force F on the weightless plate from the inflating pressure and the upward components of the membrane tension can be calculated from (4c) by numerical integration along C'D'. If F is different from zero, the value of w_B must be adjusted to decrease F to zero. This was easily accomplished by Newton iteration.

Another method of solution which avoided iteration and produced the same result is outlined as follows: It is desired to solve (1a) subject to (1b) and (1c) so let $w = w_o + w_B w_1$ where w_B is as defined above. Now if $\nabla^2 w_o = -2$ on R with $w_o = 0$ on S_o and S_1, and if $\nabla^2 w_1 = 0$ on R with $w_1 = 0$ on S_o and $w_1 = 1$ on S_1, then $w = w_o + w_B w_1$ is a solution to (1a) and satisfies boundary condition (1b). Boundary condition (1c) is satisfied if

$$\int_{S_1} \frac{\partial w_o}{\partial n}\, ds + w_B \int_{S_1} \frac{\partial w_1}{\partial n}\, ds = 2A.$$

Solutions for w_o and w_1 are obtained as explained above with known boundary values, and the integrals in (1c) can be evaluated and w_B calculated.

After the membrane deflection has been found the shear stresses are evaluated by transforming equations (2a), (2b), and (3) and evaluating them in the transformed field. Transformation of (2a), (2b), and (3) yields,

$$\tau_x = (x_\xi w_\eta - x_\eta w_\xi)/J\, J_t \tag{8a}$$

$$\tau_y = -(y_\eta w_\xi - y_\xi w_\eta)/J\, J_t \tag{8b}$$

$$J_t = 2\iint J\, w\, d\xi\, d\eta + 2A\, w_B\ , \tag{9}$$

where Aw_B is the volume under the weightless flat plate and the double integral is the volume under the rest of the membrane. The magnitude and direction of the shear stress can be found from its x and y components and plotted either as contours of constant stress in the physical field or as shear stress vectors.

NON-UNIFORM SHAFT, CIRCULAR CROSS SECTION, ARBITRARY PROFILE

The torsion of circular shafts of variable diameter along their length is an axi-symmetric problem in the theory of elasticity, and it is a two-dimensional problem in cylindrical coordinates as shown in Figures 3 and 4. A detailed treatment of the problem can be found in Timoshenko[4].

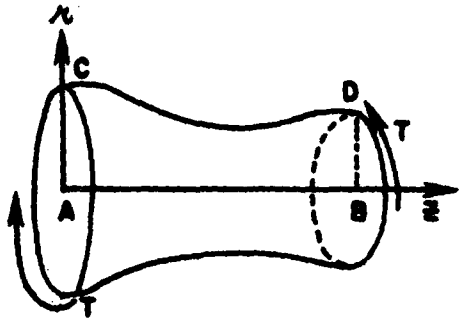


Fig. 3. Non-uniform shaft of arbitrarily varying circular cross section.

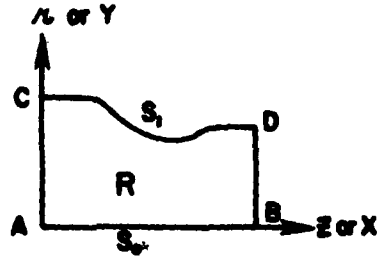


Fig. 4. Two-dimension cylindrical coordinates.

The governing equations can be simplified by the introduction of a stress function to yield

$$\phi_{rr} - \frac{3}{r}\phi_r + \phi_{zz} = 0 \quad \text{on R} \tag{10}$$

where

$$\tau_{r\theta} = -\frac{1}{r^2}\phi_z \tag{11a}$$

and

$$\tau_{z\theta} = \frac{1}{r^2}\phi_r \,. \tag{11b}$$

Torques are applied to the ends of the shaft by the specification of a shear stress distribution. In this work all shafts transitioned to a uniform radius at each end so that the linear stress distribution of a uniform shaft could be impressed as a boundary condition at each end. Due to uniform radius at the ends $\tau_{r\theta} = 0$ and due to a linear distribution of $\tau_{z\theta}$, $\tau_{z\theta} = \frac{1}{r^2}\phi_r = kr$ where k is any constant. Integrating, $\phi = .25kr^4$ at each end. To obtain the torque on the shaft integrate $\tau_{z\theta}$ over the end so that

$T = \int_0^R 2\pi r^2 \tau_{z\theta} dr = \int_0^R 2\pi r^2 (\frac{1}{r^2}\phi_r) dr$. Integrating,

$T = 2\pi \int_0^R \frac{\partial \phi}{\partial r} dr = 2\pi(\phi_R - \phi_0)$. For simplicity choose a unit torque and set ϕ_o equal to zero. This gives $\phi_R = \frac{1}{2\pi}$ as the value of the stress function along the outer surface of the shaft and zero down the axis of the shaft.

Now on the ends of the shaft $\phi = .25kr^4$ and since $\phi_R = \frac{1}{2\pi} = .25kR^4$, $k = \frac{2}{\pi R^4}$ which yields $\phi = \frac{1}{2\pi}(\frac{r}{R})^4$ for the boundary condition on the ends of the shaft.

Now recognize that r and z in Figure 4 can be changed to y and x to match the notation for which equations (5) apply. Next a coordinate system is generated for the shaft profile shown in Figure 4 which is transformed into Figure 2. Transforming equation (10) yields

$$\alpha\phi_{\xi\xi} - 2\beta\phi_{\xi\eta} + \gamma\phi_{\eta\eta} + (\tau + \frac{3J}{y}x_\eta)\phi_\xi + (\sigma - \frac{3J}{y}x_\xi)\phi_\eta = 0 \tag{12}$$

When this equation is expressed as central finite differences of ϕ, an equation similar to (6) results with ϕ exchanged for w. Because of the presence of $-\frac{3}{r}\phi_r$ in equation (10), equations (7b) through (7f) are modified as given below,

$$D_1 = \beta/4(\alpha+\gamma) \tag{13a}$$

$$D_2 = (\gamma+\sigma - \frac{3J}{y} x_\xi)/2(\alpha+\gamma) \tag{13b}$$

$$D_3 = (\alpha-\tau - \frac{3J}{y} x_\eta)/2(\alpha+\gamma) \tag{13c}$$

$$D_4 = (\alpha+\tau + \frac{3J}{y} x_\eta)/2(\alpha+\gamma) \tag{13d}$$

$$D_5 = (\gamma-\sigma + \frac{3J}{y} x_\xi)/2(\alpha+\gamma) \tag{13e}$$

$D_6 = 0$ (due to the right side of equation (10) being zero).

The algebraic equations in nodal values of ϕ are solved by SOR iteration with the boundary values of stress function being specified.

Substituting y for r and x for z, equations (11) for the shear stresses become

$$\tau_{r\theta} = -\frac{1}{y^2}\frac{\partial\phi}{\partial x}$$

$$\tau_{z\theta} = \frac{1}{y^2}\frac{\partial\phi}{\partial y}$$

which when transformed become

$$\tau_{r\theta} = -(y_\eta\phi_\xi - y_\xi\phi_\eta)/J\,y^2 \tag{14a}$$

$$\tau_{z\theta} = (x_\xi\phi_\eta - x_\eta\phi_\xi)/J\,y^2$$

The stress components in the physical field can be computed in the transformed field by expressing the derivatives of x, y, and ϕ in equations (14) as finite differences in the transformed field. Shear stress contours in the physical field can be plotted knowing $\tau_{r\theta}$ and $\tau_{z\theta}$.

BENDING OF ARBITRARILY SHAPED PLATES

The bending of thin isotropic flat plates by transverse loading is governed by the biharmonic equation. In this work the biharmonic equation is broken up into two coupled Poisson equations largely because of the algebriac difficulty encountered in transforming higher derivatives. Recently, however, there has been developed a way of accomplishing the transformation of derivatives automatically.

Boundary conditions considered in this work are simply supported and clamped boundaries which are either curved or straight. These conditions require second derivatives of plate deflection. The free edge condition requires third derivatives and was not considered.

The biharmonic equation and boundary conditions were non-dimensionalized by dividing the physical coordinates by a characteristic length a and by dividing the transverse plate loading by a characteristic loading q_o. Bending and twisting moments divided by $q_o a^2$ and plate deflections multiplied by flexual rigidity D and divided by $q_o a^4$ are non-dimensionalized.

Plate bending equations

Governing equation. The biharmonic equation in terms of non-dimensional deflection w is $\nabla^4 w = q/q_o$. Letting $\nabla^2 w = F$, then $\nabla^2 F = q/q_o$, and the biharmonic equation is reduced to two coupled Poisson equations whose solution requires values of deflection and quantity F on the boundary. The physical meaning of the quantity F can be found from the following moment-curvature relations for a plate (moments and deflection are non-dimensional).

$$M_x = \frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2} \tag{15a}$$

$$M_y = \frac{\partial^2 w}{\partial y^2} + \nu \frac{\partial w^2}{\partial x^2} \tag{15b}$$

$$M_{xy} = (1-\nu) \frac{\partial^2 w}{\partial x \partial y} \tag{15c}$$

adding

$$M_x + M_y = (1+\nu)\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right)$$

and

$$\frac{M_x + M_y}{(1+\nu)} = \nabla^2 w = F \ . \tag{16}$$

It can also be shown that the sum of the bending moments at a point in a plate is an invariant so that $M_x + M_y = M_n + M_{nt} = (1+\nu)F$.

Boundary conditions, simply supported edges. For a plate with simply supported edges and straight boundaries, $w = 0$ and $F = 0$ on the boundaries. However, if the boundary is curved and simply supported, $w = 0$ and the normal moment M_n is zero. Since the tangential moment is unknown F cannot be specified on the boundary and $M_n = 0$ must be used as the second boundary

condition. For a unit normal vector at a point on the plate boundary making an angle θ counterclockwise with the x axis, the normal moment about the plate edge is

$$M_n = M_x \cos^2\theta + M_y \sin^2\theta - 2M_{xy} \sin\theta \cos\theta \ . \tag{17a}$$

Substituting equations (15)

$$M_n = (\nu - \nu \cos^2\theta + \cos^2\theta) \frac{\partial^2 w}{\partial x^2} + (\nu - \nu \sin^2\theta + \sin^2\theta) \frac{\partial^2 w}{\partial y^2} +$$

$$(1-\nu)\sin 2\theta \frac{\partial^2 w}{\partial x \partial y} \tag{17b}$$

Boundary conditions, clamped edges. For plates with clamped edges the boundary deflection is zero and the normal slope of the plate is zero.

$$\frac{\partial w}{\partial n} = 0 \tag{18}$$

Transformed plate equations

The transformation of the two coupled Poisson equations yields expressions similar to equations (4a) and (5). Likewise the expression for normal moments contains second derivatives of w and for generality the expression below which contains the coupled Poisson equations and normal moment was transformed.

$$g(x,y) = A_{11} \frac{\partial^2 w}{\partial x^2} + A_{12} \frac{\partial^2 w}{\partial x \partial y} + A_{22} \frac{\partial^2 w}{\partial y^2} \tag{19}$$

For $F = \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}$, $A_{11} = A_{22} = 1$, $A_{12} = 0$, $g = F$, and

for $\frac{q(x,y)}{q_o} = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2}$, $A_{11} = A_{22} = 1$, $A_{12} = 0$, $g = q/q_o$,

and $M_n(x,y) = [\nu + (1-\nu) \cos^2\theta] \frac{\partial^2 w}{\partial x^2} + [(1-\nu) \sin 2\theta] \frac{\partial^2 w}{\partial x \partial y} +$

$[\nu + (1-\nu) \sin^2\theta] \frac{\partial^2 w}{\partial y^2}$ where $g = M_n$ and

$$A_{11} = \nu + (1-\nu) \cos^2\theta \ , \tag{20a}$$

$$A_{12} = (1-\nu) \sin 2\theta \ , \tag{20b}$$

$$A_{22} = \nu + (1-\nu)\sin^2\theta \tag{20c}$$

Transforming equation (19) yields

$$g(\xi,\eta) = C_1 w_{\xi\xi} + C_2 w_{\xi\eta} + C_3 w_{\eta\eta} + C_4 w_{\xi} + C_5 w_{\eta} \tag{21}$$

where

$$C_1 = (A_{11} y_\eta^2 - A_{12} x_\eta y_\eta + A_{22} x_\eta^2)/J^2 \tag{22a}$$

$$C_2 = (-2A_{11} y_\xi y_\eta + A_{12}(x_\xi y_\eta + x_\eta y_\xi) - 2A_{22} x_\xi x_\eta)/J^2 \tag{22b}$$

$$C_3 = (A_{11} y_\xi^2 - A_{12} x_\xi x_\xi + A_{22} x_\xi^2)/J^2 \tag{22c}$$

$$C_4 = (A_{11} B_1 + A_{12} B_2 + A_{22} B_3)/J^2 \tag{22d}$$

$$C_5 = (A_{11} B_4 + A_{12} B_5 + A_{22} B_6)/J^2 \tag{22e}$$

$$C_6 = \frac{1}{2(C_1 + C_3)} \tag{22f}$$

and

$$B_1 = y_\eta y_{\xi\eta} - y_\xi y_{\eta\eta} + (y_\xi y_\eta J_\eta - y_\eta^2 J_\xi)/J \tag{23a}$$

$$B_2 = x_\xi y_{\eta\eta} - x_\eta y_{\xi\eta} + (x_\eta y_\eta J_\xi - x_\xi y_\eta J_\eta)/J \tag{23b}$$

$$B_3 = x_\eta x_{\xi\eta} - x_\xi x_{\eta\eta} + (x_\xi x_\eta J_\eta - x_\eta J_\xi)/J \tag{23c}$$

$$B_4 = y_\xi y_{\xi\eta} - y_\eta y_{\xi\xi} + (y_\xi y_\eta J_\xi - y_\xi^2 J_\eta)/J \tag{23d}$$

$$B_5 = x_\eta y_{\xi\xi} - x_\xi y_{\xi\eta} + (x_\xi y_\xi J_\eta - x_\eta y_\xi J_\xi)/J \tag{23e}$$

$$B_6 = x_\xi x_{\xi\eta} - x_\eta x_{\xi\xi} + (x_\xi x_\eta J_\xi - x_\xi^2 J_\eta)/J \tag{23f}$$

$$J_\xi = y_\eta x_{\xi\xi} - y_\xi x_{\xi\eta} - x_\eta y_{\xi\xi} + x_\xi y_{\xi\eta} \tag{23g}$$

$$J_\eta = y_\eta x_{\xi\eta} - y_\xi x_{\eta\eta} - x_\eta y_{\xi\eta} + x_\xi y_{\eta\eta} \tag{23h}$$

Notice that expressions (22) and (23) reduce to equations (5) for $A_{11} = A_{22} = 1$ and $A_{12} = 0$.

The angle θ in equations (20) is found from expressions for the unit normal in the transformed plane.

$$\theta = \arccos(-\sigma y_\xi/\sqrt{\gamma}) \text{ for lines of constant } \eta \text{ and} \tag{24a}$$

$$\theta = \arccos(\sigma y_\eta/\sqrt{\alpha}) \text{ for lines of constant } \xi. \tag{24b}$$

The normal derivative of deflection in the transformed plane is given by

$$\frac{\partial w}{\partial n} = \sigma(\gamma w_\eta - \beta w_\xi)/J\sqrt{\gamma} \text{ for lines of constant } \eta \text{ and} \tag{25a}$$

$$\frac{\partial w}{\partial n} = \sigma(\alpha w_\xi - \beta w_\eta)/J\sqrt{\alpha} \text{ for lines of constant } \xi. \tag{25b}$$

σ is +1 for the top and right side and σ is -1 for the bottom and left side of the computational field to insure a positive outward normal.

Solution of transformed plate equations

Equation (21) is the transformed governing Poisson equation $\nabla^2 w = F$ when $g = F$. Substituting F for w and setting $g = q/q_o$ gives the transform of $\nabla^2 F = q/q_o$. Expressing the derivatives as central finite differences yields the following coupled algebraic equations:

$$F_{i,j} = D_1(F_{i-1,j+1} - F_{i+1,j+1} + F_{i+1,j-1} - F_{i-1,j-1}) \tag{26a}$$

$$+ D_2F_{i,j+1} + D_3F_{i-1,j} + D_4F_{i+1,j} + D_5F_{i,j-1} - D_6\frac{q}{q_o}$$

Similarly

$$w_{i,j} = D_1(w_{i-1,j+1} - w_{i+1,j+1} - w_{i-1,j-1} + w_{i+1,j-1}) \tag{26b}$$

$$+ D_2W_{i,j+1} + D_3W_{i-1,j} + D_4W_{i+1,j} + D_5W_{i,j-1} - D_6F_{i,j}$$

where

$$D_1 = -1/4\ C_3C_6 \tag{27a}$$

$$D_2 = (C_2 + 1/2\ C_5)C_6 \tag{27b}$$

$$D_3 = (C_1 - 1/2\ C_4)C_6 \tag{27c}$$

$$D_4 = (C_1 + 1/2\ C_4)C_6 \tag{27d}$$

$$D_5 = (C_2 - 1/2\ C_5)C_6 \tag{27e}$$

$$D_6 = J^2C_6 \tag{27f}$$

If the values of w and F are specified on the plate boundary, equations (6) can be solved by point SOR iteration methods. The normal slope can be computed at each boundary point by expressing the derivatives of w in equations (25) as forward, central, or backwards finite differences depending on the location of the boundary point in the transformed field. The calculation of the normal moment at a boundary point requires first the angle between the outward normal and the x axis. This angle is evaluated from equations (24) and used in equations (20) to obtain the coefficients A_{11}, A_{12}, and A_{22} of the deflection derivatives in the physical plane. Using these values of A_{11}, A_{12}, and A_{22} the coefficients of equation (21) can be computed by evaluating equations (23) and (22). The actual expression for normal moment depends on the location of the boundary point. Forward, backward, and central finite differences are used to compute the derivatives of w on the four sides and four corners of the transformed field. Then the boundary moment is computed from equation (21).

The computer code for the solution of plate deflection problems requires the identification of the type of boundary condition present. This identifier selects either the normal moment equation or the normal slope equation for the boundary value to be satisfied. Starting values of w and F are chosen for the plate and the governing equations are solved. Next, the boundary value of slope or moment is calculated and compared with the value actually present on the boundary. This comparison dictates a change in the value of F on the boundary and the procedure is repeated until convergence is obtained.

In this work it is assumed that the value of F at a boundary point influences only the slope or moment at that point. A simple iteration scheme was used to drive the calculated boundary value B to the desired value B_{BDY}. In the equation below p is the iterate number, and δ is a relaxation parameter.

$$F^{p+1} = F^p - \delta(B^p - B_{BDY})\frac{F^p - F^{p-1}}{B^p - B^{p-1}}. \tag{28}$$

There are more accurate and more efficient iteration schemes than this, but it works well enough to demonstrate the validity of the general method. There is certainly influence on all boundary points due to changing any one value of

F, and for efficiency of calculation a global iteration scheme should be used along with line SOR.

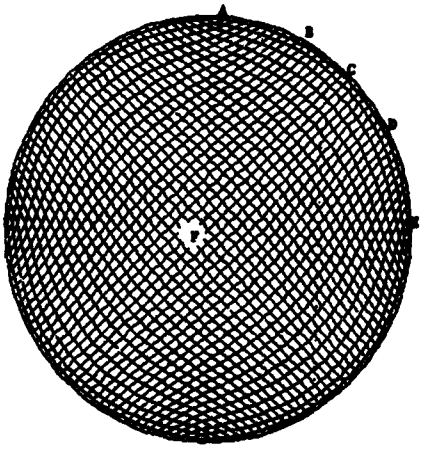


Fig. 5a. Coordinate system with uniform boundary spacing.

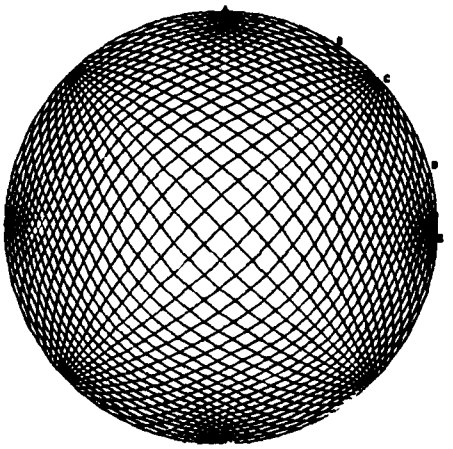


Fig. 5b. Coordinate system with an exponential boundary spacing.

RESULTS FOR UNIFORM SHAFTS

Four coordinate systems were generated for a circular shaft of unit radius subjected to a unit torque. Two coordinate systems were simply connected, one having a uniform spacing and one having an exponential spacing of boundary points as shown in Figures 5a and 5b. These coordinate systems have singularities at the four points which transform into the corners of the rectangular transformed field. The stress cannot be computed at these points but could be taken as the average of the stresses at points to either side. Considerable skewness of the coordinate system occurred near the corner points. The stress distribution along line CF in Figure 5a which has orthogonal coordinates was linear, and along line AF which has skew coordinates the stress distribution deviated somewhat from linear. The value of the maximum stress differed from the exact value by .42% at point C, by .59% at point B, and by -.68% at point A. This error can be accounted for largely by a torsion constant .375% too small. Trapezoidal integration which gives a value smaller than the actual volume was used to find the volume under the deflected membrane.

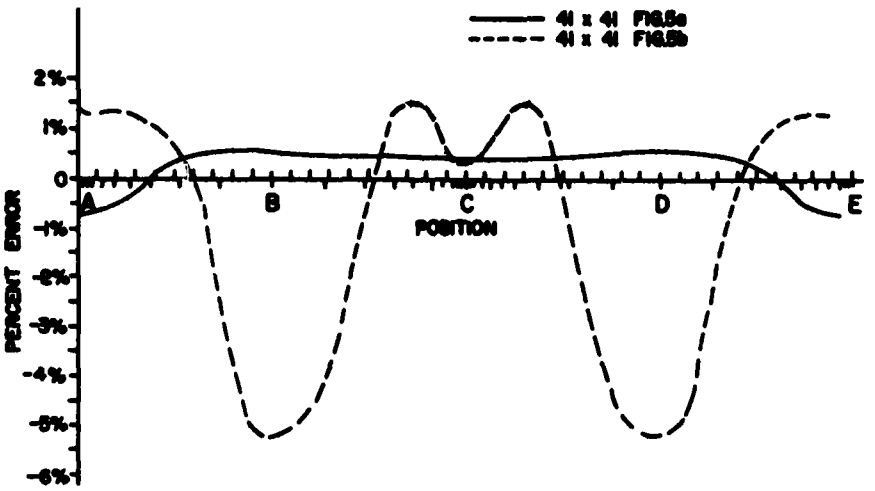


Fig. 5c. A comparison of surface shear stress error for two simply connected coordinate systems showing the effect of coordinate spacing and skewness.

Figure 5c shows a plot of stress error in relation to the coordinate spacing for the simply connected coordinate systems in Figures 5a and 5b. One sees 5% error in the region of large spacing around points B and D for coordinate system 5b. Skewness effects can be seen also.

By introducing a small hole at the center of the shaft an orthogonal coordinate system was generated for the doubly connected region as shown in Figure 6a. A constant value of stress around the outer boundary was .3% in error. Another small hole (diameter .01) was introduced at one half the radius of the shaft (r = .5), and a coordinate system was generated which was not orthogonal as is shown in Figure 6b. The coordinate spacing is uniform on the boundary for both of these cases which should have constant stress around the outer boundary. Figure 6c shows the effect of coordinate skewness on the error. Even though both systems have 51 radial coordinates, the radial spacing near the boundary for the system in 6b is twice as great as for the system in 6a, and this is reflected in the accuracy in Figure 6c near points A and B where the skewness is not large.

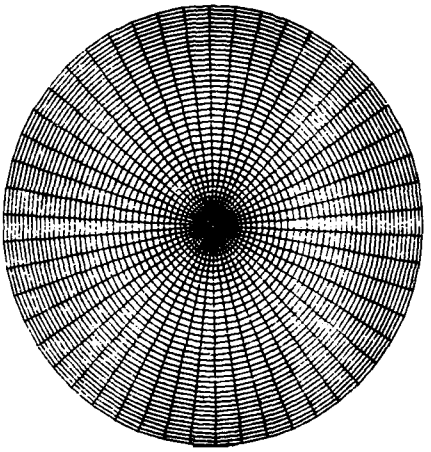


Fig. 6a. Orthogonal coordinate system with uniform boundary coordinate spacing.

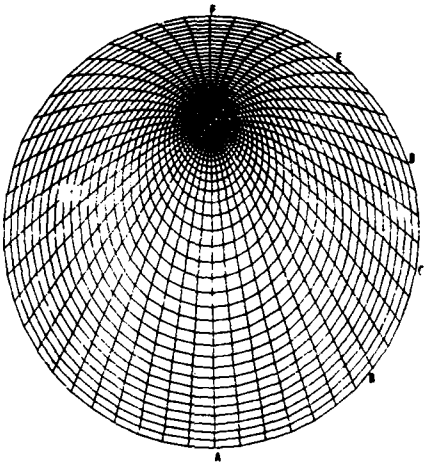


Fig. 6b. Non-orthogonal coordinate system with uniform boundary coordinate spacing.

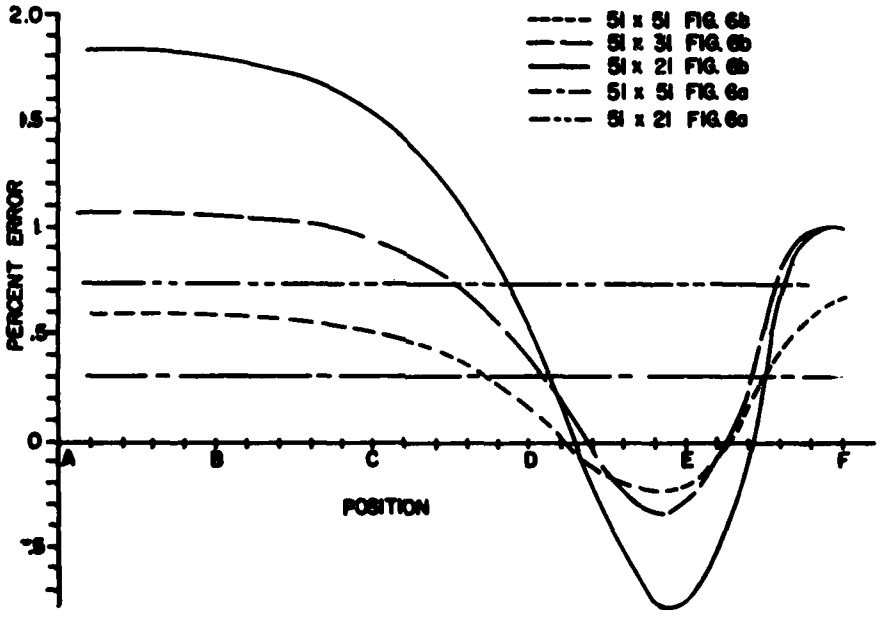


Fig. 6c. A comparison of surface shear stress error for five multiply connected coordinate systems showing the effect of coordinate spacing and skewness.

A plot of stress along line FA in Figure 6b shows a linear distribution with a spike at the interior hole located at r = .5. The stress on the sides of the hole is twice the stress at r = .5 remote from the hole, just as predicted by Kelvin's fluid flow analogy. Figure 7 shows the distribution of stress around the small hole, and it differs from a sinusoidal distribution because of the non-uniformity of the "flow field" around the hole.

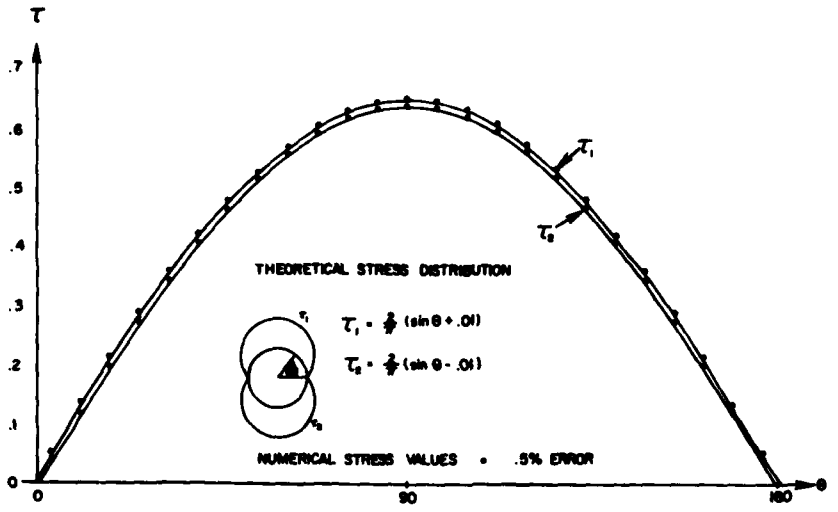


Fig. 7. A comparison of theoretical and numerical shear stresses around a small hole at mid radius of a circular shaft.

Figure 8a shows coordinate systems generated for a hollow circular shaft with a keyseat cut into it. The inner and outer radii, keyseat dimensions, and keyseat radius were chosen to conform to an example from Timoshenko[5]. The circular fillet in the keyseat had three points for one coordinate system and five points for another. Stress calculations gave values of 2.43 and 2.67 for the stress concentration at the fillets which compare with the value 3.4 given by Timoshenko[5]. More closely spaced coordinates would probably increase the stress on the fillet slightly. The value given by Timoshenko was determined experimentally by a fluid flow measurement and could easily be in error. Figure 8b shows a plot of membrane deflection contours around the keyseat, and Figure 8c shows the stress contours which clearly indicate stress concentration around the keyseat fillets.

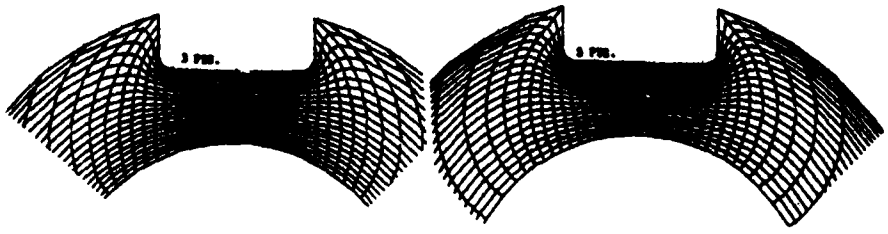


Fig. 8a. Coordinate systems for the region around the keyseat in a hollow circular shaft.

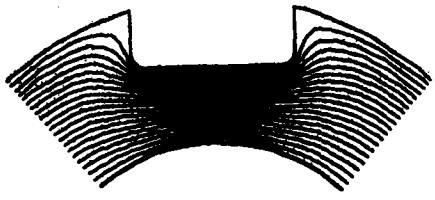


Fig. 8b. Membrane deflection contours around the keyseat in Fig. 8a.

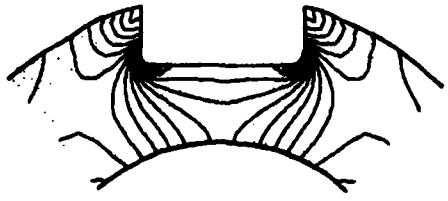


Fig. 8c. Shear stress contours around the keyseat in Fig. 8a.

RESULTS FOR NON-UNIFORM SHAFTS

Exact solutions exist for the stress function in uniform and tapered shafts. Rather coarse coordinate systems were generated for these two shafts, and calculations for the stress functions yielded a quartic distribution of stress function and a linear distribution of stress. The surface stresses were computed within .48% of the exact value for the uniform shaft and within .67% of the exact value for the tapered shaft using 20 radial coordinates. Due to the presence of a $\frac{1}{r}$ term in the governing differential equation, it was necessary to use a locally optimum acceleration parameter to obtain convergence by SOR iteration.

As a further test of the boundary fitted coordinate method, the problem of two uniform shafts of different diameters connected by a circular fillet was considered. Coordinate systems were generated for six different shafts, and stresses were computed and compared with measurements made by Jacobsen[6], who used an electrical analogy to experimentally measure the stress concentration in shafts. A coordinate system typical of the six generated for the

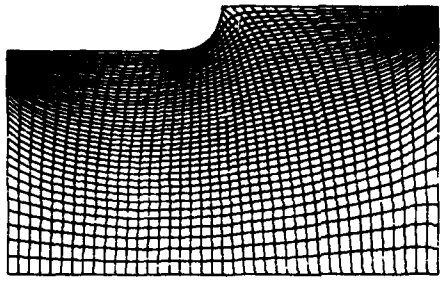


Fig. 9a. Coordinate system for a stepped shaft whose diameters are connected by a circular fillet.

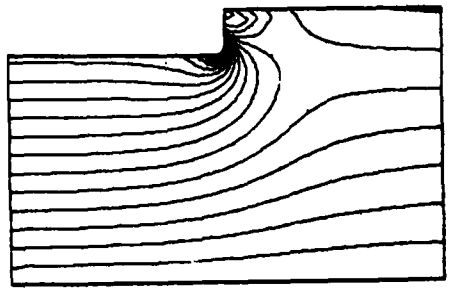


Fig. 9b. Shear stress contours around the circular fillet connecting a stepped shafts' two diameters.

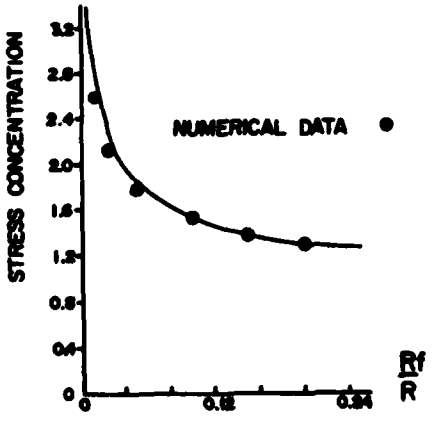


Fig. 9c. Comparison of numerical with experimental stress concentration factors for stepped shafts.

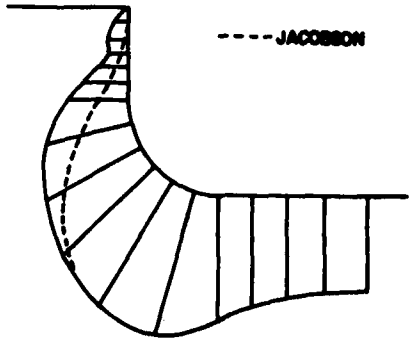


Fig. 9d. Comparison of numerical with experimental shear stress along a circular fillet in a stepped shaft.

shafts is shown in Figure 9a, and a typical set of stress contours is shown in Figure 9b. The maximum stress was found and converted to a stress concentration factor and compared with experimental values in Figure 9c. Agreement is excellent except for small fillet radii, and Jacobsen did not believe the accuracy of his results in this region due to the difficulty of measuring electrical potential differences over small lengths. Figure 9d shows stress

distributed along the fillet for one shaft. The agreement with Jacobsen's data is excellent except in the corner above the fillet where no effort was made to attract coordinate lines (see Figure 9a) since the stress is known to be zero in external corners. Results for four shafts with circular grooves of various radii and depths, and for a shaft of two diameters with a linear transition between the diameters are presented in Young[7].

RESULTS FOR PLATES OF ARBITRARY SHAPE

The deflection of plates involved coupled Poisson equations and boundary conditions somewhat more complicated than those used in the torsion of shafts. In spite of the complexities of simultaneous governing equations and second derivative boundary conditions, accurate plate deflections were computed.

Initial efforts to confirm the accuracy of the computer code were directed toward the solution of classical plate problems which had Dirichlet boundary conditions, namely zero deflection and zero $F(=\nabla^2 w)$. These conditions hold for any simply supported polygonal plate. Other problems such as circular plates or circular sector plates had zero deflection along the boundary and a known constant F (circular plates) or known variable distribution of F (circular sector plates) along the boundary. When the boundary values of w and F were prescribed, a point SOR iteration of the coupled governing equations converged to the analytical solution with an accuracy which depended on number of coordinates (grid size), coordinate spacing, and skewness of the coordinate system. When the coordinate spacing was small, errors of .01% or smaller were obtained, which supports an earlier statement that the error in the torsion

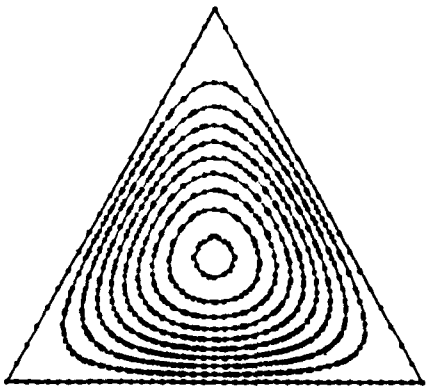


Fig. 10a. Comparison of numerical and theoretical deflection contours for a simply supported equilateral plate uniformly loaded.

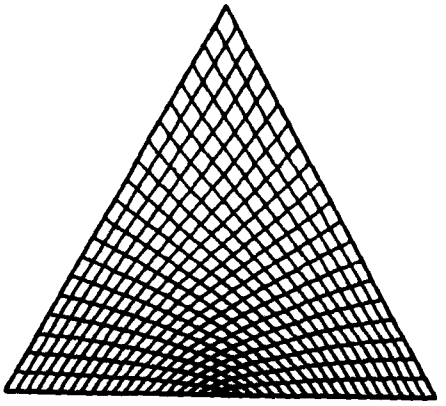


Fig. 10b. Coordinate system for a simply connected equilateral triangular plate.

of uniform shafts or arbitrary cross section was due primarily to the determination of the torsion constant by numerical integration of the volume under the deflected membrane.

A typical comparison of deflection contours is presented in Figure 10a, which shows the close agreement between numerical and analytical contours for an equilateral plate uniformly loaded and simply supported. The coordinate system for this plate is shown in Figure 10b.

Having confirmed that the boundary fitted coordinate method would yield correct results for field values of deflection and F when the true values of boundary deflection and boundary F were specified, the next step was to determine the accuracy of calculations for boundary slope and boundary moment. For plates without skew coordinates (square or rectangular shape) the calculated slope at the boundary and boundary edge moments were as much as 3% in error, although the deflection and F at the plate center were nearly exact. The coordinate system in Figure 5a was used for circular plates, and it has regions of severe skewness around the four corner points. The boundary slope and boundary moment values showed an error which was definitely related to the skewness. Variations from 1% to 25% of the exact value of slope were computed. The effects of skewness on the stress in a shaft were compared in Figure 6c for several coordinate systems. Stress is obtained from first derivatives, while boundary moment is obtained from second derivatives, upon which the effect of skewness is more pronounced.

For the solution of an arbitrary shape to be possible, the unknown boundary values of F must be found which cause the edge slope or edge moment to be zero according to simple or clamped boundary conditions. This was attempted iteratively and met with good success if skewness was not present in the coordinates. For example, the values of F along the boundary of a square plate were varied until the edge moment was zero. A central deflection error of 1% and central F value error of .8% were obtained. However, when the same procedure was attempted for a circular plate with the coordinate system of Figure 5a (very skewed), the central deflection error was 2.4%, and central F value error was 1.9%. Skewness would have made the solution impossible except that for the circular plate it was known that a constant value of F existed on the boundary, and it was not necessary to find a different F at each boundary point, as was necessary for the square plate.

Skewness is much less pronounced for doubly connected regions so a circular plate with a circular hole was investigated. This doubly connected region had an orthogonal coordinate system, with radial coordinate lines attracted near

the boundaries. The inner and outer boundary values of F, which are constant around each boundary due to symmetry, were varied until the slope was zero for clamped edges and until the edge moment was zero for simply supported edges. The results were compared with an analytical solution by Georgian[8]. In each case the boundary values are satisfied, but the field values are in error about 3% at the point of maximum deflection. This is to be expected since for square plates the boundary values were calculated to only 3% accuracy for a nearly exact deflection field. This error generally improves with a denser grouping of coordinate lines (decreasing grid size) on the boundary.

To demonstrate the application of the boundary fitted coordinate method to other shapes, an ellipse with an interior hole and a triangle with an interior hole were investigated. Coordinate systems for these plates are shown in Figures 11a and 11b, and they are clearly non-orthogonal, but the skewness is not severe. The plates were simply supported on all edges and were uniformly loaded. The variation of boundary F was found which satisfied the condition of zero edge moment, and contours of plate deflection are shown in Figures 12a and 12b. It is estimated that the maximum deflection error is less than the 3% error obtained for the circular plate with circular hole, since there are more coordinates for the elliptical plate than for the circular plate.

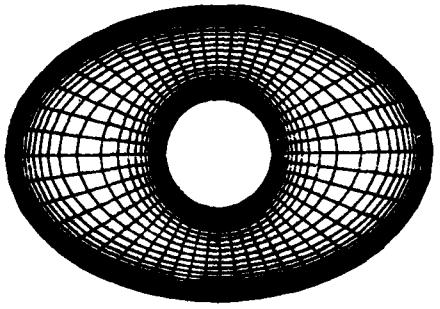


Fig. 11a. Coordinate system for an ellipse with a circular interior hole.

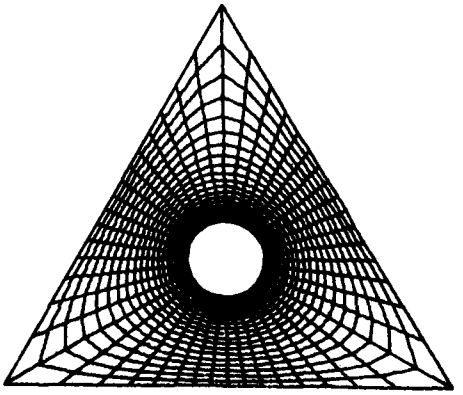


Fig. 11b. Coordinate system for an equilateral triangle with a circular interior hole.

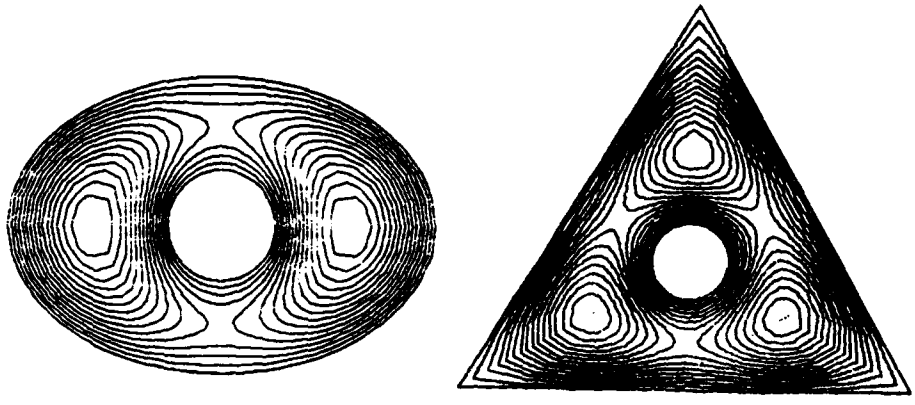


Fig. 12. Deflection contours for uniformly loaded plates with interior and exterior boundaries simply supported.

CONCLUSIONS

The boundary fitted coordinate method simplifies the boundary condition description for solid mechanics problems, while complicating the governing differential equations somewhat. This is an overall advantage since governing equations are easily represented as finite difference expressions, and great difficulties are eliminated by having coordinate lines coincident with the boundaries. A distinct disadvantage of the method lies in its introduction of skew coordinates which have an adverse effect on the accuracy of calculations. It appears that the problems introduced by skewness can be overcome somewhat by a finer grid spacing or more numerous coordinates. An advantage of the method is the ease with which problems of different geometry can be worked. Practically the only difference from one problem to another lies in the generation of a coordinate system for the region under consideration.

The results obtained indicate that the boundary fitted coordinate method yields accurate results and is relatively easy to apply to solid mechanics problems which can be formulated in terms of a field equation. The boundary fitted coordinate method should be extendable to elasticity problems and those problems which have moving boundaries or perhaps regions of changing shape (yield zone for example). It is the ability of the boundary fitted coordinate method to follow regions of changing shape which should be exploited in future work.

REFERENCES

1. Thompson, J. F., Thames, F. C., and Mastin, C. W. (1977) NASA CR-2729.
2. Den Hartog, J. P. (1952) Advanced Strength of Materials. McGraw-Hill Book Company, New York and London, pp. 1-48.
3. Thames, F. C. (1975) Dissertation. Mississippi State University, pp. A1-A8.
4. Timoshenko, S. (1934) Theory of Elasticity. McGraw-Hill Book Company, New York and London, pp. 276-284.
5. Timoshenko, S. (1941) Strength of Materials. D. Van Nostrand Company, New York, p. 326.
6. Jacobsen, L. A. (1925) Transactions of the ASME, 47, pp. 619-638.
7. Young, C. (1980) Thesis. Mississippi State University.
8. Georgian, J. C. (1957) Journal of Applied Mechanics, 24, pp. 306-310.

AD P000976

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

COORDINATE SYSTEM CONTROL: ADAPTIVE MESHES

J. U. BRACKBILL
Applied Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545

INTRODUCTION

In the numerical solution of transient fluid flow problems, the flow variables are calculated at a finite number of points. Typically, the points form ordered arrays which are joined together in some systematic way to form a computation mesh. On this mesh, the flow equations are approximated by finite differences, which are then marched in time.

It is intuitively obvious that the greater the number of points in the computation mesh, the more accurate the numerical solution will be. For one thing, the accuracy of the difference equations depends on the fineness of the mesh. As importantly, the accuracy of the solution depends on the resolution of flow gradients. Where the flow gradients are largest, the absolute error in the difference approximation to derivatives is largest.[1] In addition, more numerical diffusion must be introduced to maintain sufficient smoothness of the solution for nonlinear stability.[2]

When the flow gradients vary from place to place and the mesh spacing is constant, the numerical errors are largest where flow gradients are least well resolved. When mesh points can be added, errors are reduced most efficiently by adding points only in regions of strong gradients.[3] Similarly, when the number of mesh points is fixed, greater overall accuracy can often be obtained by concentrating mesh points where gradients are strong and dispersing them where gradients are weak.[4]

An algorithm which concentrates and disperses points automatically based on the numerical solution of the flow equations is described as adaptive. Here, an adaptive algorithm is derived from a minimum principle, and its application to transient flow problems in two dimensions is described.

The major topics discussed are the variational formulation of a mesh generator with interior control, its use in adaptively rezoning a time-dependent problem, and the solution of the inverse problem to allow adaptive zoning of arbitrary initial meshes. Some of the discussion is reproduced from Ref. 1, but the application to time-dependent problems is emphasized.

VARIATIONAL FORMULATION OF THE MESH GENERATOR

For the solution of finite difference equations on a computation mesh, the data is typically stored in ordered arrays of numbers, $\phi(i,j)$ in which the indices $i = 1, \ldots, M$; $j = 1, \ldots, N$, give not only the location of the data in computer memory, but also the physical relationship betwen the data at one vertex $\underset{\sim}{x}(i,j)$ and another, $\underset{\sim}{x}(i',j')$.

In formulating the mesh generator problem mathematically, it is useful to view the mesh, whose vertices are $\underset{\sim}{x}(i,j)$, as the image of a mapping $\underset{\sim}{x}(\xi,\eta)$ in which the points corresponding to integer values of the natural coordinates, ξ and η, are the mesh vertices. The image of the mesh in natural coordinates is a uniform, rectilinear mesh with mesh spacing $\Delta\xi = \Delta\eta = 1$. A mesh generator determines the mapping $\underset{\sim}{x}(\xi,\eta)$.

Obviously, the differential properties of the mapping are reflected in the properties of the computation mesh. For example, $[(\partial x/\partial\xi)^2 + (\partial y/\partial\xi)^2]^{1/2}$ along a level curve of η is related to mesh spacing between vertices with the same index j. Similarly, the volume of computational cells is related to the Jacobian, J, of the mapping,

$$J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \quad ,$$

and the orthogonality of the mesh is related to the scalar, $\nabla\xi \cdot \nabla\eta$, which is zero when conjugate lines of the mesh are orthogonal.

Integrals over the computation mesh can be written which measure these properties of the mapping. The global smoothness of the mapping (the variation in mesh spacing along level curves of ξ and η) is measured by the integral,

$$I_s = \int_D \left[(\nabla\xi)^2 + (\nabla\eta)^2\right] dV \quad .$$

The orthogonality of the mapping is measured by,

$$I_o = \int_D (\nabla\xi \cdot \nabla\eta)^2 J^3 \, dV$$

and the weighted volume variation is measured by,

$$I_v = \int_D wJ \, dV \quad ,$$

where $w = w(x,y)$ is given.

The smoothest mapping can be obtained by minimizing I_s, the most orthogonal mapping by minimizing I_o', and the mapping with specified variation of J by minimizing I_v.[1,4]

A useful mesh generator results when a combination of I_o', I_v and I_s is minimized as in the penalty method.[5] That is, the integral I is minimized,

$$I = I_s + \lambda_v I_v + \lambda_o' I_o' \quad ,$$

where λ_v and λ_o are positive constants of O(1).

SOLUTION OF THE VARIATIONAL PROBLEM IN TWO DIMENSIONS

To derive the Euler equations for the variational problem formulated in the preceeding section, it is first convenient to interchange dependent and independent variables using the relations.

$$\xi_x = \frac{+y_\eta}{J} \quad , \qquad \xi_y = \frac{-x_\eta}{J} \quad , \qquad \eta_x = \frac{-y_\xi}{J} \quad , \qquad \eta_y = \frac{+x_\xi}{J} \quad .$$

After interchanging variables, the smoothness measure can be written,

$$I_s = \int_1^M \int_1^N d\xi \, d\eta \, \frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{J} \quad ,$$

for which the corresponding Euler equations are,

$$\left(\frac{\partial}{\partial x} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial x_\xi} - \frac{\partial}{\partial \eta}\frac{\partial}{\partial x_\eta}\right)\left[\frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{J}\right] = 0 \quad ,$$

$$\left(\frac{\partial}{\partial y} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial y_\xi} - \frac{\partial}{\partial \eta}\frac{\partial}{\partial y_\eta}\right)\left[\frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{J}\right] = 0 \quad .$$

The measure of volume variation, after interchanging dependent and independent variables, can be written,

$$I_v = \int_1^M \int_1^N d\xi\, d\eta\ wJ^2 \quad ,$$

for which the Euler equations are,

$$\frac{\partial w}{\partial x} J^2 - 2\,[J(w_\xi y_\eta - w_\eta y_\xi) + w(J_\xi y_\eta - J_\eta y_\xi)] = 0 \quad ,$$

and

$$\frac{\partial w}{\partial y} J^2 - 2\,[J(x_\xi w_\eta - x_\eta w_\xi) + w(x_\xi J_\eta - x_\eta J_\xi)] = 0 \quad .$$

Similarly, the orthogonality measure, I_o', can be written

$$I_o' = \int_1^M \int_1^N d\xi\, d\eta\ (x_\xi x_\eta + y_\xi y_\eta)^2 \quad ,$$

for which the Euler equations are

$$b_{o1}x_{\xi\xi} + b_{o2}x_{\xi\eta} + b_{o3}x_{\eta\eta} + a_{o1}y_{\xi\xi} + a_{o2}y_{\xi\eta} + a_{o3}y_{\eta\eta} = 0 \quad ,$$

and

$$a_{o1}x_{\xi\xi} + a_{o2}x_{\xi\eta} + a_{o3}x_{\eta\eta} + c_{o1}y_{\xi\xi} + c_{o2}y_{\xi\eta} + c_{o3}y_{\eta\eta} = 0 \quad .$$

The mesh generator equations can be written in a form convenient for computation by collecting the coefficients of the highest derivatives of x and y with respect to ξ and η and writing,

$$b_1x_{\xi\xi} + b_2x_{\xi\eta} + b_3x_{\eta\eta} + a_1y_{\xi\xi} + a_2y_{\xi\eta} + a_3y_{\eta\eta} = -J^2 \frac{1}{2w} \frac{\partial w}{\partial x} \quad , \qquad (1)$$

$$a_1 x_{\xi\xi} + a_2 x_{\xi\eta} + a_3 x_{\eta\eta} + c_1 y_{\xi\xi} + c_2 y_{\xi\eta} + c_3 y_{\eta\eta} = -J^2 \frac{1}{2w} \frac{w}{y} \quad . \tag{2}$$

The coefficients are reproduced here from Ref. 1 for completeness:

$$a = a_s + \lambda_v a_v + \lambda_o' a_o \quad ,$$

$$b = b_s + \lambda_v b_v + \lambda_o' b_o \quad , \text{ and}$$

$$c = c_s + \lambda_v c_v + \lambda_o' c_o \quad .$$

$$a_{s1} = -A\alpha \quad ; \quad b_{s1} = B\alpha \quad ; \quad c_{s1} = C\alpha$$

$$a_{s2} = 2A\beta \quad ; \quad b_{s2} = -2B\beta \quad ; \quad c_{s2} = -2C\beta$$

$$a_{s3} = -A\gamma \quad ; \quad b_{s3} = B\gamma \quad ; \quad c_{s3} = C\gamma$$

where

$$A = x_\xi y_\xi + x_\eta y_\eta \quad , \quad B = y_\xi^2 + y_\eta^2 \quad , \quad C = x_\xi^2 + x_\eta^2 \quad ,$$

and

$$\alpha = \frac{x_\eta^2 + y_\eta^2}{J^3} \quad , \quad \beta = \frac{x_\xi x_\eta + y_\xi y_\eta}{J^3} \quad , \quad \gamma = \frac{x_\xi^2 + y_\xi^2}{J^3} \quad .$$

$$a_{v1} = -x_\eta y_\eta \quad , \quad b_{v1} = y_\eta^2 \quad , \quad c_{v1} = x_\eta^2 \quad ,$$

$$a_{v2} = x_\xi y_\eta + x_\eta y_\xi \quad , \quad b_{v2} = -2y_\xi y_\eta \quad , \quad c_{v2} = -2x_\xi x_\eta \quad ,$$

$$a_{v3} = -x_\xi y_\xi \quad , \quad b_{v3} = y_\xi^2 \quad , \quad c_{v3} = x_\xi^2 \quad .$$

$$a_{o1} = x_\eta y_\eta \quad , \quad b_{o1} = x_\eta^2 \quad , \quad c_{o1} = y_\eta^2 \quad ,$$

$$a_{o2} = x_\xi y_\eta + x_\eta y_\xi \quad , \quad b_{o2} = 2(2x_\xi x_\eta + y_\xi y_\eta) \quad , \quad c_{o2} = 2(x_\xi x_\eta + 2y_\xi y_\eta) \, .$$

The generator equations form a set of coupled, quasi-linear, partial differential equations of second order which are only slightly more complex than in the original Winslow algorithm.

Algebraic equations at each node result from the substitution of the differences for derivatives. The system of equations is solved by a Jacobi iteration in which the values of $x_{i,j}$ and $y_{i,j}$ are treated as parameters. The iterative solution of these equations will give new values of $x_{i,j}$ and $y_{i,j}$ which satisfy the Euler equations above. A more complete discussion of the numerical solution of the Euler equations is given in Refs. 1 and 6.

THE ADAPTIVE MESH

By choosing an appropriately defined weight function, w(x,y), the mesh generator can be made part of an algorithm to adapt a computation mesh dynamically to data generated by the solution of finite difference equations.

The weight function must depend on the data in such a way that numerical errors are reduced. An obvious approach is to minimize the truncation error for a particular difference approximation. While this may be optimal, detailed analysis of a given difference equation usually yields a very complex expression for the truncation error whose minimization may have no clear connection to the variational forms above.

However, a simpler approach yields a practical, yet useful, algorithm. As has been noted already, numerical errors are reduced significantly when flow gradients are well resolved. When there are sufficient mesh points in each gradient length of the function being represented, the smoothness of the function assures that truncation errors and numerical diffusion necessary for nonlinear stability are reasonable.

The choice of weight function, w, is simplified by noting that as λ_v becomes large, the solution to the mesh generator can be written,

$$wJ^2 = \text{const.}$$

Where ϕ, the function being represented, is positive, a weight function with correct dimensionality is given by,

$$w = \left(\frac{1}{\phi}\nabla\phi\right)^4 \quad , \tag{3}$$

so that wJ^2 is dimensionless. Thus, where w is large, corresponding to steep gradients, J will be small, corresponding to increased resolution.

Although the choice of weight function is straightforward, there are a number of problems of a practical nature which must be solved. The gradient of ϕ is calculated from the numerical data by finite differences and may accentuate the roughness of the data. Variations in mesh size occur in a discrete way in moving from one cell to the next and may increase truncation error. Finally, there are constraints on maximum and minimum zone sizes. The minimum zone size determines the time step, and thus the cost of a calculation. The maximum zone size cannot be greater than the entire mesh.

To address these problems, the algorithm must be modified. The weight function calculated from the data is smoothed to spread the influence of a single data point over a region of the mesh. The coefficient, λ_v, is typically chosen to be O(1) so that the influence of the smoothness integral results in a smooth variation in cell size from zone to zone. Finally, w is scaled between maximum and minimum values corresponding to prescribed maximum and minimum zone sizes.

A NUMERICAL EXAMPLE OF ADAPTIVE ZONING

A simple example illustrating the effect of adaptive zoning in two dimensions is afforded by considering the steady solution of a convection diffusion equation in two dimensions,

$$\nabla\cdot(\phi\underset{\sim}{U}) - \mu\nabla^2\phi = 0 \quad ,$$

where $\underset{\sim}{U} = \hat{r}U(r)$ is given, and μ is a positive constant. On an infinite domain, the first integral of the equation is simply.

$$\underset{\sim}{U}\phi - \mu\nabla\phi = 0 \quad .$$

One can see clearly that when μ is small, where $\underset{\sim}{U}$ is finite $\nabla\phi/\phi$ will be very large. Choosing U(r) to be a function which is non-zero only in an annulus of width δr at r_0 will result in the formation of a boundary layer at r_0.

To demonstrate the increased accuracy of difference equations with adaptive zoning the solution ϕ is given analytically, and the error is defined as the

value of the RHS of the first integral when $\nabla\phi$ is approximated by a finite difference.

The result of a numerical calculation on a square domain, $0 \leq x \leq 1$, $0 \leq y \leq 1$, with $r = 0$ at $x = y = 1/2$, $r_0 = 1/4$, and $U_0 = 12$ is shown in Fig. 1. Where there is rapid variation of ϕ, the cells in the adapted mesh are small in the figure.

An important result is shown in Fig. 2a where the variation of I_v, I_s and the error, ϵ, with λ_v is shown.

Clearly, minimizing I_v reduces the error since both I_v and ϵ decrease together as λ_v increases. The decrease in I_v results in a small increase in I_s corresponding to a departure from Winslow's mesh. The decrease in ϵ as λ_v increases from 0 to 1 is dramatic; from 25% to 3.5%. The conclusion is that with increasing adaptivity, substantial increases in accuracy are obtained. The effect of varying the size of the mesh with and without adaptivity, shown in Fig. 2b, indicates that equivalent accuracy is obtained with the adaptive mesh with one tenth as many cells as in the nonadaptive mesh.

ADAPTIVE ZONING OF TIME-DEPENDENT PROBLEMS

Time dependent flow calculations can be adaptively zoned simply and cheaply by the following algorithm. At the beginning of each time step, the weight function is calculated from the numerical data and processed as described above. The coordinates of the mesh are stored, and new coordinates are calculated from the mesh generator and separately stored. Either by interpolation or by the solution of transport equations in conservation form,[7] the data are transferred from the old mesh to the new one, and the new mesh replaces the old mesh in memory.

The algorithm is explicit. The new mesh is generated with data from the previous time step and does not anticipate changes which may occur during the course of the time step. This splitting of time advancement of the flow equations and time advancement of the mesh is more accurate when the solution changes slowly from time step to time step, as expressed by the inequality,

$$\frac{1}{\phi}\frac{\partial \phi}{\partial t}\,\Delta t < 1 \quad .$$

One might object that this condition may be violated locally, especially in singular perturbation problems. For example, a shock may move a distance comparable to its width in a time step resulting in large changes in the solution in the neighborhood of the shock. With this in mind, one can understand

the value of smoothing the weight function so that the fine zoning induced by strong gradients in the shock front extends some distance to either side of the shock front. Thus, a more relevant condition for the time step is given by the rate of change of the weight function,

$$\frac{1}{w}\frac{\partial w}{\partial t}\Delta t < 1.$$

When the weight function is varying too rapidly, additional smoothing of the data is often required (with some consequent loss in adaptivity).

When the inequality above is satisfied, the displacement of the mesh over a time step is relatively small, and the mesh at the beginning of the time step very nearly satisfies the mesh generator equations. After only two or three iterations, the solution is usually sufficiently good that further improvement is unnecessary. There is also a physical argument why this is so. The change in the flow solution over a time step at a given point depends only on the flow variables at neighboring points. So that the mesh generator can respond on the same time scale as the flow solution, the domain of dependence for each point in the mesh must be at least as large. Since each iteration of the mesh generator increases the domain of dependence by one cell in every direction, two or three iterations are sufficient.

The value of adaptive gridding in time-dependent flow problems can be identified by considering the problem of numerical diffusion. In all flow calculations, the approximation to convective transport introduces numerical diffusion. The amount of diffusion can be estimated by truncation error analysis.[2] For example, when convective transport in one dimension is approximated by an implicit difference equation,[8]

$$^{n+1}\phi_j = {}^{n}\phi_j - \frac{\delta t}{2\delta x}\left({}^{n+1}(\phi u)_{j+1} - {}^{n+1}(\phi u)_{j-1}\right) ,$$

where $^{n+1}\phi_j = \phi(j\delta x,(n+1)\delta t)$, the differential equation approximated by the difference quation to $O(\delta x^3,\delta t^2)$ is,

$$\frac{\partial \phi}{\partial t} + \frac{\partial}{\partial x}(\phi u) = \kappa' \frac{\partial^2 \phi}{\partial x^2} ,$$

where κ' is the numerical diffusion,

$$\kappa' = \frac{\delta t}{2} u^2 - \frac{\delta x^2}{2} \frac{\partial u}{\partial x} \quad .$$

When it is required for accuracy's sake that $u\delta t \leq \delta x$, κ' can be written,

$$\kappa' \approx \frac{1}{2} \frac{\delta x^2}{\delta t} \left(1 - \frac{\delta x}{u} \frac{\partial u}{\partial x}\right) \quad .$$

Clearly, when flow gradients are not resolved and $\delta x(\partial u/\partial x) > u$, κ' is negative and numerical instability results.[2] Analysis of other, standard difference approximations yields similar results.

There are a number of approaches to maintaining stability. Historically, an explicit numerical viscosity has been added to the flow equations to smooth gradients and make them resolvable. However, the consequent reduction of maximum, representable Reynolds numbers is currently considered unacceptable. Several other approaches which do not introduce excessive additional diffusion are now available. One can calculate the negative diffusion locally from truncation error analysis, and compensate by adding a comparable positive diffusion. One can control diffusion locally using flux corrected transport. Alternatively, one can improve the resolution of flow gradients through adaptive gridding. In this case adaptive gridding not only increases the accuracy of numerical calculations, but also the stability. Of course, adaptive gridding is compatible with flux corrected transport.[9]

The effect of adaptive gridding is illustrated by the numerical solution of Fisher's equation in a moving frame,

$$\frac{\partial \phi}{\partial t} + \underset{\sim}{U} \cdot \nabla \phi = \kappa \nabla \cdot \nabla \phi + \phi(1 - \phi)/\tau \quad ,$$

where $\underset{\sim}{U}_o$ is the frame velocity, κ the diffusion coefficient, and τ the reaction time. This single equation contains many of the effects modelled by a system of equations describing diffusion of chemically reacting materials.

In one dimension, the solution to Fisher's equation corresponds to a steady front moving with speed $v = (\kappa/\tau)^{1/2}$. Across the front, whose thickness is proportional to $\kappa^{1/2}$, ϕ increases from zero to one.[10]

Since Fisher's equation will be solved on a moving grid, it is convenient to integrate the equation over a volume whose boundary is moving with local velocity $\underset{\sim}{U}$,

nature of the numerical diffusion, the reduction results from better resolution of the flow gradients.

ADAPTIVITY FOR ARBITRARY MESHES: THE INVERSE PROBLEM

To apply adaptivity to a non-uniform mesh, one must solve the following inverse problem: consider a mesh given by the solution of the mesh generator equations with $\lambda_v = 0$. This mesh satisfies the generator equations with $\lambda_v \neq 0$ only for some particular $w(x,y)$, which must be calculated.

In a typical problem, one has no a priori knowledge of the location of boundary layers or shocks. However, the boundaries of the domain or embedded structures may be represented most conveniently using the mesh generator with $\lambda_v = 0$, or by a simple functional mapping $x(\xi,\eta)$, $y(\xi,\eta)$. To adaptively zone a calculation using this mesh as the initial mesh, one must calculate the corresponding $w_I(x,y)$.

The weight function $w_I(x,y)$ can be calculated from the Euler equations most conveniently when they are written in a form which explicitly displays their dependence on w and its derivatives. This form can be derived from Eqs. (1) and (2) by collecting coefficients of w, $\partial w/\partial x$ and $\partial w/\partial y$ to obtain,

$$S_1 + wR_1 + \lambda_v J^2 \frac{\partial w}{\partial x} = 0 \quad ,$$

and

$$S_2 + wR_2 + \lambda_v J^2 \frac{\partial w}{\partial y} = 0 \quad ,$$

where S_1, S_2, R_1 and R_2 are easily evaluated using the definitions of the coefficients and are functions of the given mesh coordinates.

These equations are easily integrated along level curves of the computation mesh. Noting that w_ξ and w_η are given by the chain rule, the derivatives can be written,

$$w_\xi = -\{(S_1 + wR_1)x_\xi + (S_2 + wR_2)y_\xi\}/\lambda_v J^2 \quad , \tag{4}$$

and

$$w_\eta = -\{(S_1 + wR_1)x_\eta + (S_2 + wR_2)y_\eta\}/\lambda_v J^2 \quad . \tag{5}$$

On level curves of the mesh where ξ is constant, one integrates the equation for w_η, and vice versa for mesh lines where η is constant. So that the resulting w is single valued, one must require that the difference equations satisfy the condition,

$$\langle w_{\xi\eta} \rangle - \langle w_{\eta\xi} \rangle = 0 \quad ,$$

where the brackets denote difference equations.

A simple example taken from a recent calculation of magnetic reconnection in the earth's magnetosphere is shown in Fig. 7. The mesh is generated in the upper half plane with level curves of ξ corresponding to nested ellipses whose eccentricity increases from zero at the center to some large value on the outside. In addition, the distance between ellipses increases geometrically from the center outward.

With $\lambda_v \neq 0$ and w calculated from Eqs. (4) and (5) above, the mesh which satisfies the generator equations is as shown. With w equal to a constant, the mesh would collapse onto the center of curvature.[6]

ADAPTIVE ZONING FOR MOVING BOUNDARY CALCULATIONS

In many moving boundary calculations, the ability to redistribute points along the boundary increases the accuracy and reliability of the calculations. For example, in the modeling the Rayleigh-Taylor instability at the interface between two fluids regions of large curvature may develop which require additional points to resolve. When the total number of points along the interface is fixed, the points may be redistributed along the curve using the adaptive mesh generator.

Where the curve is parameterized by s, so that $x = x(s)$, $y = y(s)$ where $s = s(\xi)$, the analogue of the smoothness integral along the curve may be written,

$$I_s = \int \xi_s^2 \, ds \quad .$$

To alter the distribution of points along the boundary, the integral,

$$I_R = \int w s_\xi^2 \, d\xi \quad ,$$

is minimized with given weight function w(s). When w is proportional to the curvature,

$$\frac{\partial}{\partial t}\int_V \phi \, dV = -\int_s (\hat{n} \cdot \underset{\sim}{F}) \, ds + \int_V (\phi(1-\phi)/\tau) \, dV \quad ,$$

where $\underset{\sim}{U}$ is the displacement of the grid over a time step and where $\underset{\sim}{F}$, the transport across the surface of the moving control volume due to convection and diffusion is given by,

$$\underset{\sim}{F} = \{(\underset{\sim}{U}_o - \underset{\sim}{U}) - \kappa\nabla\}\phi$$

As noted by Thompson, et al.,[6] conservation form is preserved in natural coordinates,

$$\nabla \cdot \underset{\sim}{F} = \{(F_1 y_\eta - F_2 x_\eta)_\xi + (-F_1 y_\xi + F_2 x_\xi)_\eta\}/J$$

where F_1 and F_2 are the x and y components of $\underset{\sim}{F}$. In this form, difference equations in natural coordinates are easily derived.

Using an algorithm which combines donor and interpolated donor cell differencing in the correct proportion to guarantee positive diffusion,[7] Fisher's equation is solved on the domain $0 \le x \le L$, $0 \le y \le L$ with $\phi = \epsilon > 0$ in a small neighborhood of $(x,y) = (0.2L, 0.2L)$ initially, and $\underset{\sim}{U} = (v,v)$, where $v = (\kappa/\tau)^{1/2}$, a circular front expands to eventually fill the domain. With $(\kappa\tau)^{1/2} = 10^{-2}L$, the thin front shown in Fig. 4 results at $t = 0.4L/(\kappa/\tau)^{1/2}$.

With an Eulerian mesh, the results with a mesh of 625 cells shown in Fig. 3 are typical of a calculation with significant numerical diffusion. The speed of the front is greatest toward the lower left corner, and the front actually overtakes the incoming flow and stops only at the boundary. When the mesh size is increased to 5625 cells, the results shown in Fig. 4 are satisfactory. The speed of propagation is approximately correct and the shape of the front is circular.

With an adaptive mesh and weight function given by Eq. (3), the results with 625 cells are shown in Figs. 5 and 6, and are similar to the finely zoned Eulerian calculation with nearly ten times as many cells. The speed of propagation is nearly correct, and the shape is roughly circular. (With finer zoning, there is further improvement as one might expect.)

With the same number of zones, evidently the adaptively zoned calculation introduces less numerical diffusion than the Eulerian one. Because of the

$$\frac{1}{R^2} = \left(\frac{\partial^2 x}{\partial s^2}\right)^2 + \left(\frac{\partial^2 y}{\partial s^2}\right)^2 \quad .$$

points along the curve are concentrated in regions of greatest curvature.

The Euler equations for combined smoothing and curvature resolution terms can be written,

$$0 = s_{\xi\xi} + \frac{2\lambda_R}{R^2}\left(s_{\xi\xi} - \frac{1}{R} R_\xi s_\xi\right) \quad .$$

A simple numerical example of the effect of adaptive zoning on an interface is shown in Fig. 8. In this example, points are distributed at equal intervals along the curve $y = \sin^4 x$ when $\lambda_R = 0$, and are concentrated where the curvature is largest when $\lambda_R = 100$. Note that the area under the curves is the same. Displacements of points along the curves are constrained to preserve the area under the curve.

CONCLUSIONS

The discussion of the formulation and properties of the adaptive mesh algorithm is as complete as space permits. Some additional details and examples are given in Refs. 1 and 9. Other aspects of the method, especially regarding the use of adaptive gridding in realistic calculations, are described in Refs. 11 and 12.

As should be clear from the discussion, adaptive gridding is most useful for singular perturbation problems where localized regions with large gradients develop spontaneously. Often, choosing the correct weight function is not trivial, and, sometimes, it is appropriate to cause the mesh to respond to structure in several dependent variables simultaneously.

An interesting new direction for adaptive meshes is described in Ref. 12. There, an adaptively zoned, particle-in-cell method is presented. Numerical diffusion of material properties is eliminated, and changes in scale are accommodated. The method appears to have significant potential for multi-material calculations.

ACKNOWLEDGEMENTS

Work performed under the auspices of the US Department of Energy.

REFERENCES

1. J. U. Brackbill and J. S. Saltzman, Adaptive Zoning for Singular Problems in Two Dimensions, J. Comp. Phys., to be published.
2. C. W. Hirt, J. Comp. Physics., 2, 339 (1968).
3. I. Babuska and W. Rheinboldt, SIAM J. Num. Anal., 15 (1978).
4. N. N. Yanenko, V. M. Kovenya, V. D. Lisejkin, V. M. Fonin, and E. V. Vorozhtsov, Proc. 6th Int'l. Conf. on Num. Meth. in Fluid Dynamics, Lecture Notes in Physics, 90, 565 (1979).
5. R. Courant, Trans. Amer. Math. Soc., 50, 40 (1941).
6. Joe F. Thompson, Frank C. Thames and C. Wayne Mastin, J. Comp. Phys., 15, 299 (1974).
7. C. W. Hirt, A. A. Amsden and J. L. Cook, J. Comp. Phys., 14, 227 (1974).
8. J. U. Brackbill, Meth. Comp. Phys., 16, 1 (1976).
9. J. J. Saltzman and J. Brackbill, Applications and Generalizations of Variational Methods for Generating Adaptive Meshes, these proceedings.
10. J. Canosa, IBM Journal of Research and Development, 17, 307 (1973).
11. R. D. Milroy and J. U. Brackbill, Numerical Studies of a Field Reversed Theta Pinch Plasma, Phys. of Fluids (to appear).
12. J. U. Brackbill and H. M. Ruppel, A Method for Adaptively Zoned, Particle-in-Cell Calculations in Two Dimensions, submitted J. Comp. Phys.

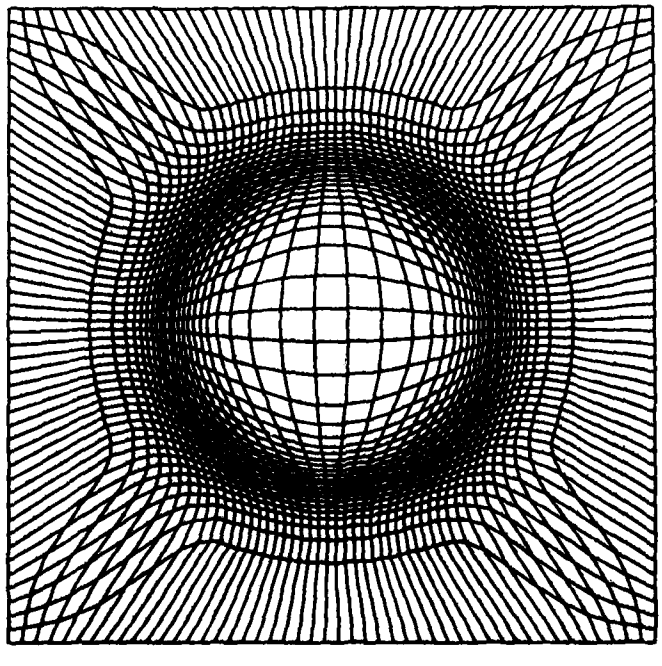


Fig. 1. In the adaptive mesh, the zones are concentrated in an annulus where there is the greatest variation in the solution.

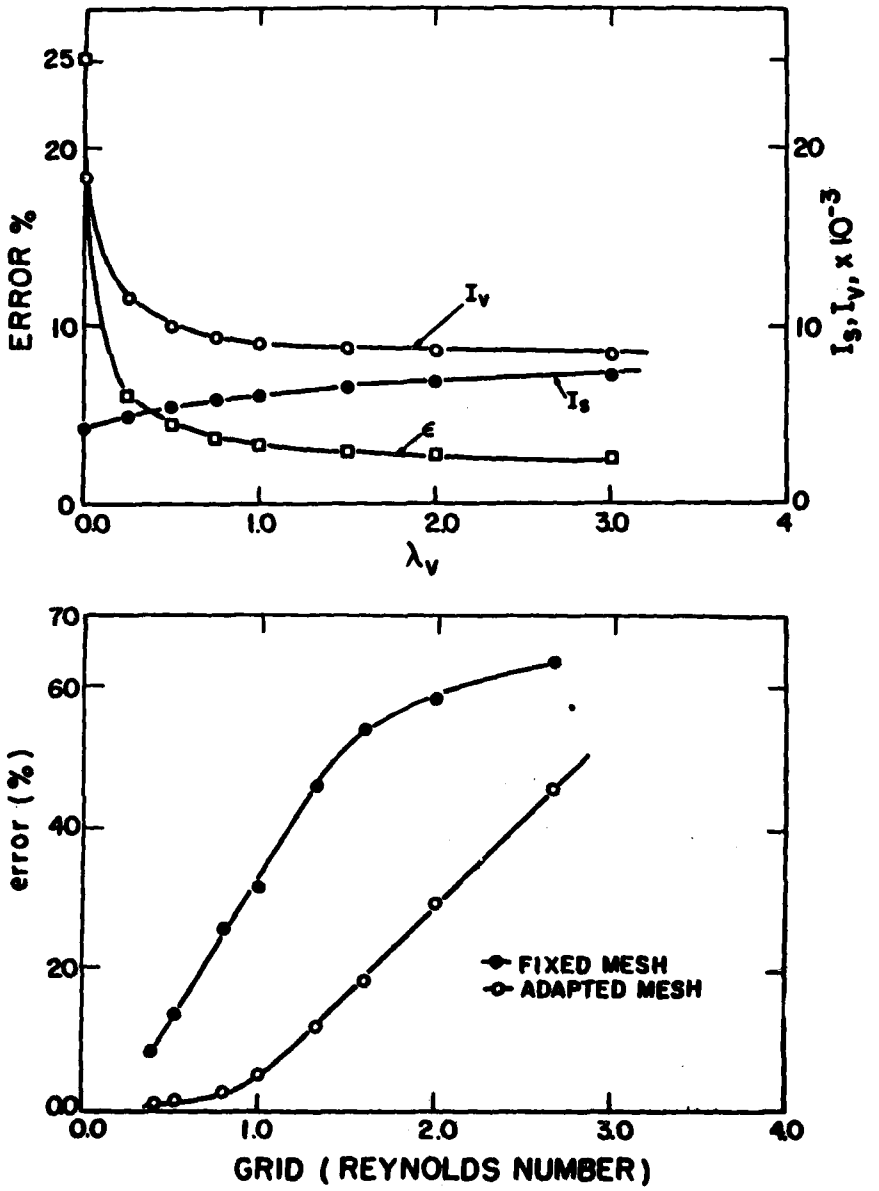


Fig. 2. The error in the finite difference approximation and the integral of weighted volume decrease together in (a) with increasing adaptivity. The error in an adaptive mesh shown in (b) is much smaller than in a fixed Eulerian mesh, especially for R_e greater than one. R_e, the grid Reynolds number, is the ratio of the average mesh spacing to smallest gradient length.

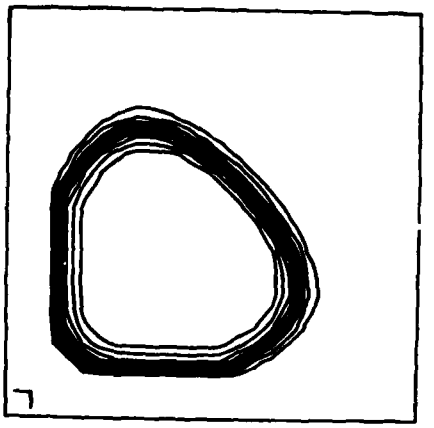


Fig. 3. The solution of Fisher's equation on a 625 zone Eulerian mesh is depicted. Contours of constant ϕ are concentrated in the front across which ϕ increases from 0 outside to 1 inside. The distortion of the front from the correct, circular shape is evident.

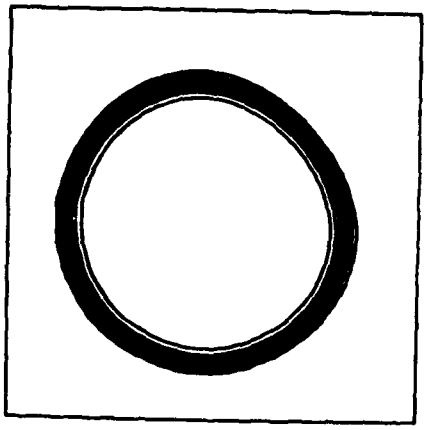


Fig. 4. With a 5625 zone Eulerian mesh, a solution with a more circular front and correct propagation speed is obtained.

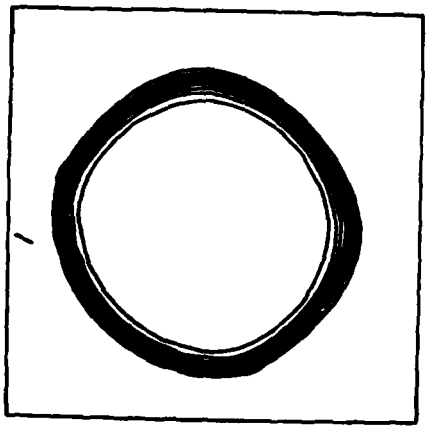


Fig. 5. With a 625 zone adaptive mesh, a comparable solution to one with a 5625 zone Eulerian mesh is obtained. The shape is nearly circular and the propagation speed is approximately correct.

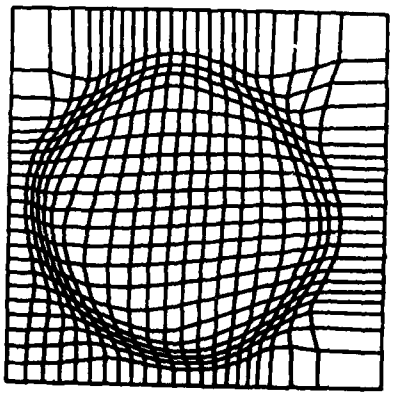


Fig. 6. The mesh corresponding to Fig. 5 is shown. Note the concentration of zones in the front.

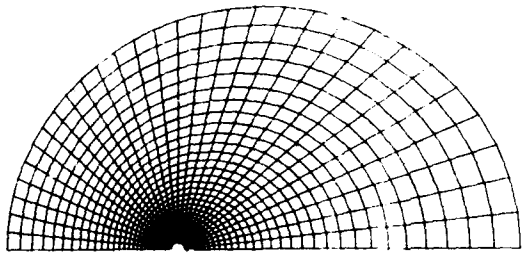
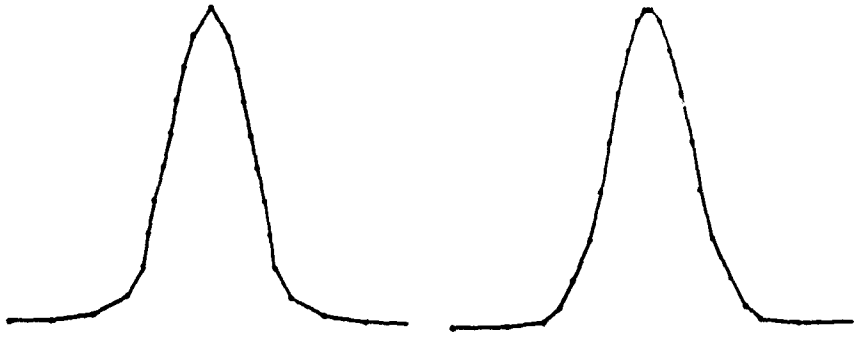


Fig. 7. A computation mesh for a simulation of reconnection in the earth's magnetosphere is shown. The grid results from the solution of the inverse problem with adaptivity implemented.



(a) (b)

Fig. 8. In (a), points are equally spaced along the curve. In (b), the points are concentrated in regions of high curvature.

AD P000977

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ON APPLICATION OF BODY CONFORMING CURVILINEAR GRIDS FOR FINITE DIFFERENCE SOLUTION OF EXTERNAL FLOW

Joseph L. Steger

Department of Aeronautics and Astronautics

Stanford University, Stanford, CA 94305

I. INTRODUCTION

Finite difference practitioners frequently make use of arbitrary coordinate transforms and introduce body conforming curvilinear grid systems. The coordinate transforms may either be built in globally in mappings from physical space to computational space, or they may be built in locally in the finite volume sense. The advantages of using body conforming curvilinear grids in finite difference flow field simulation include the following: Body conforming grids simplify the application of boundary conditions insofar that grid lines will coincide with the body boundary. Curvilinear grids may be clustered to flow field action regions to improve solution accuracy. Body conforming grids may allow simplification of the governing equations. Such grids can also help maintain a well-ordered system of algebraic equations suitable for vector-computer processing or approximate factorization-implicit techniques.

The task of generating suitable body conforming curvilinear grids is not an easy one. The grids should be generated in an automatic manner requiring minimum user input. Yet the user will wish to maintain considerable control of where points will be distributed along the boundary surface and how they are clustered in the interior field. Moreover the grid must be tailored in some degree to the numerical algorithm because some numerical algorithms are more sensitive than others to grid smoothness, skewness, and stretching.

Although use of body conforming curvilinear grids can offer the advantages cited previously their careless application can lead to difficulties. This is particularly true when the governing equations are differenced in conservative (i.e., divergence) form and transform metric terms are brought inside the difference operators. Then as noted above, some numerical algorithms are far more mesh-sensitive than others, and numerical accuracy

and computational efficiency can be affected by how rapidly a grid changes or how far away it is from orthogonality.

The subject of this paper is not the generation of body conforming curvilinear grids; rather it is the use of such grids in finite difference applications. In Section II of this paper the difficulties of solving the transformed equations in conservative form are discussed. In Section III various experiences are cited to suggest that considerable computational efficiency can yet be gleaned by further improvements of the grid. Concluding remarks follow in Section IV.

II. CONSERVATIVE DIFFERENCING OF TRANSFORMED EQUATIONS

a) Background

In aerodynamics applications we frequently try to difference the governing equations in conservative or divergence form. Conservative form differencing is preferred because it best maintains the correct weak solution of the governing equations. Thus if a shock wave is captured by simply solving the difference equations (as opposed to fitting a shock wave discontinuity into the difference equations), then the speed, location and jump of the shock can only be correct if the partial differential equations are in conservative form. The difference equations must also satisfy the divergence relation, at least in the vicinity of the shock. The conservative form of the equations may also be desirable for its numerical properties. For example, nonlinear equations in conservative form can be more cleanly linearized about a previous state than those in nonconservative form. This can be advantageous in implicit marching procedures in order to avoid iterative solutions of nonlinear equations with each marching step.

Let the conservation-law-form of the equations be represented in Cartesian coordinates as

$$\partial_t Q + \partial_x F + \partial_y G = 0 \tag{1}$$

where for simplicity only two dimensions are considered. This strict divergence form of the equations can be maintained for new independent variables

$$\begin{aligned} \xi &= \xi(x, y, t) \\ \eta &= \eta(x, y, t) \\ \tau &= t \end{aligned} \tag{2}$$

as (*c.f.* [1])

$$\partial_\tau \hat{Q} + \partial_\xi \hat{F} + \partial_\eta \hat{G} = 0 \tag{3}$$

where

$$\hat{Q} = J^{-1} Q \tag{4a}$$

$$\hat{F} = J^{-1}(\xi_t Q + \xi_x F + \xi_y G) \tag{4b}$$

$$\hat{G} = J^{-1}(\eta_t Q + \eta_x F + \eta_y G) \tag{4c}$$

and where J is the transform Jacobian

$$J = \xi_x \eta_y - \xi_y \eta_x \tag{5}$$

For a thermally perfect gas Q, F, and G may represent the Cartesian inviscid and viscous flux quantities for conservation of mass, momentum and energy. For example, for inviscid flow

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{pmatrix}, \quad G = \begin{pmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{pmatrix} \tag{6}$$

where ρ is fluid density, u, v and w are Cartesian velocities components, p is pressure and e is given by

$$e = (\gamma - 1)^{-1} p + \frac{1}{2}\rho(u^2 + v^2) \tag{7}$$

Alternately in the case of compressible potential flow

$$Q = \rho, \quad F = \rho\phi_x, \quad G = \rho\phi_y \tag{8}$$

and $\rho = \rho(\phi)$ is determined by the Bernoulli relation.

Although the transformed governing equations (3) are more complex than (1), apparent simplicity is returned to the inviscid flow equations with introduction of the unscaled

contravariant velocities

$$U = \xi_t + \xi_x u + \xi_y v = \xi_x(u - \dot{x}) + \xi_y(v - \dot{y}) \tag{9a}$$

$$V = \eta_t + \eta_x u + \eta_y v = \eta_x(u - \dot{x}) + \eta_y(v - \dot{y}) \tag{9b}$$

For example, the transformed conservation of mass relation is given by

$$\partial_\tau(\rho/J) + \partial_\xi(\rho U/J) + \partial_\eta(\rho V/J) = 0 \tag{10}$$

and does not appear too much more complex than the Cartesian form. Moreover, if $\eta = 0$ coincides with the body boundary surface then flow boundary conditions such as tangency and no slip are especially elegant and are expressed as

$$V = 0 \quad (tangency) \tag{11}$$

$$U, V = 0 \quad (no\ slip) \tag{12}$$

To motivate further discussion is it noted that the transformed equations (3) can be derived by first performing chain rule expansions on the terms of (1). For example

$$F_x = \xi_x F_\xi + \eta_x F_\eta + \tau_x F_\tau \tag{13}$$

where $\tau_x = 0$. The equations are then scaled by J^{-1} and metrics are brought back inside the operators using differentiation by parts. For example,

$$\frac{\xi_x}{J}\partial_\xi F = \partial_\xi\left(\frac{\xi_x F}{J}\right) - F\partial_\xi\left(\frac{\xi_x}{J}\right)$$

and so on. Terms are then collected to give equation (3) as well as combinations of metric terms that will be found to be zero. That is

$$\begin{aligned}\partial_\tau \hat{Q} + \partial_\xi \hat{F} + \partial_\eta \hat{G} = Q\left[\partial_\xi\left(\frac{\xi_t}{J}\right) + \partial_\eta\left(\frac{\eta_t}{J}\right) + \partial_\tau\left(\frac{1}{J}\right)\right] + F\left[\partial_\xi\left(\frac{\xi_x}{J}\right) + \partial_\eta\left(\frac{\eta_x}{J}\right)\right] \\ + G\left[\partial_\xi\left(\frac{\xi_y}{J}\right) + \partial_\eta\left(\frac{\eta_y}{J}\right)\right]\end{aligned} \tag{14}$$

All such right-hand-side combinations of metric terms are found to be zero because of the relations

$$\begin{aligned} \xi_x/J &= y_\eta & \eta_x/J &= -y_\xi \\ \xi_y/J &= -x_\eta & \eta_y/J &= x_\xi \\ \xi_t &= -x_\tau \xi_x - y_\tau \xi_y & \eta_t &= -x_\tau \eta_x - y_\tau \eta_y \end{aligned} \tag{15}$$

for example,

$$F\left[\partial_\xi\left(\frac{\xi_x}{J}\right) + \partial_\eta + \left(\frac{\eta_x}{J}\right)\right] \quad = \quad F(y_{\xi\eta} - y_{\eta\xi}) = 0 \tag{16}$$

and so on.

The point of all this is that the metric quantities have been worked inside the differential operators. This is possible because combinations of metric terms such as

$$\partial_\xi\left(\frac{\xi_y}{J}\right) + \partial_\eta\left(\frac{\eta_y}{J}\right) = 0 \tag{17}$$

are found. Note also that unlike equation (1), equation (3) has been scaled by J^{-1}.

b) Metric Differencing

The fact that the transformed governing equations now have the metrics brought inside the difference operations can lead to numerical errors. This is because the metric variation is now being differenced along with the flow field quantity. In typical external aerodynamics applications, the flow quantities far from the body are essentially constant or uniform. Difference terms should therefore be zero, and this will be true if equation (1) is differenced on a uniform mesh. For a nonuniform mesh the transformed equations will not yield zero in regions of constant flow, however, unless proper differences of the metric identities are zero, that is from (14)

$$\delta_\xi(\xi_x/J) + \delta_\eta(\eta_x/J) = 0 \tag{18a}$$

$$\delta_\xi(\xi_y/J) + \delta_\eta(\eta_y/J) = 0 \tag{18b}$$

$$\delta_\xi(\xi_t/J) + \delta_\eta(\eta_t/J) + \delta_\tau(1/J) = 0 \tag{18c}$$

where δ_ξ, δ_η and δ_τ are the difference operators used in the solution algorithm for (3). If, for example, the metrics ξ_x/J, etc., could be exactly evaluated (as they can be in, say,

a cylindrical coordinate), then equations (18) will not be exactly zero but will be zero to within the order of accuracy of the operators δ_ξ, etc. For a rapid variation of the metrics and for large grid spacing—a phenomenon frequently occurring in aerodynamic applications in the far field—this error can be very appreciable. It can, in fact, add a error source to the equations that can overwhelm the solution accuracy. However, if the metrics themselves are differenced such that equations (18) are exactly zero, then this error is controlled. For example, in two dimensional steady flow relations (18a) and (18b) become using (15)

$$\delta_\xi(y_\eta) - \delta_\eta(y_\xi) = 0 \tag{19a}$$

$$-\delta_\xi(x_\eta) + \delta_\eta(x_\xi) = 0 \tag{19b}$$

If y_η and y_ξ are differenced as $\delta_\eta y$ and $\delta_\xi y$ where δ_η and δ_ξ are the same difference operators used in the solution algorithm for (3), then the metric identities (19) exactly difference to zero. The importance of satisfying these relations was pointed out in [2] in which three point central spatial differences were used in the solution algorithm [3] as well as for the metric quantities.

In three dimensions it becomes more difficult to exactly difference the metric identity relations. For steady or simple unsteady grid motion, Pulliam and Steger [4] introduced an averaging process for the steady terms that works for any difference operator that can be differenced in parts as

$$\delta(uv) = (\mu v)(\delta u) + (\mu u)(\delta v) \tag{20}$$

where μ is an averaging operator. An example of (20) is given by

$$\nabla uv = (\mu v)(\nabla u) + (\mu u)\nabla v \tag{21}$$

where $\nabla u = u_j - u_{j-1}$, $\mu u = \frac{u_j + u_{j-1}}{2}$, etc. In extended work Thomas and Lombard [5] correctly treated the unsteady metric variations and cleverly simplified the calculation of the spacial metric terms. They also coined the term "metric conservation law" to describe the fact that the metric relations (18) must be differenced to be zero.

As one introduces higher order central or one-sided spatial difference operators, or uses predictor-corrector schemes, it becomes more and more difficult to correctly satisfy

the metric relations (18). This ultimately lead [4] to an approximate cancellation method that relies on solving the differenced equations in a simple perturbation form:

$$\delta_t(\hat{Q}-\hat{Q}_\infty)+\delta_\xi(\hat{F}-\hat{F}_\infty)+\delta_\eta(\hat{G}-\hat{G}_\infty)=0 \tag{22}$$

In the far field $\hat{F}\to\hat{F}_\infty$, etc., and any consistent difference scheme is satisfied regardless of how rapidly the metrics vary. When F appreciably varies from F_∞, etc., it is assumed that the grid is sufficiently smooth and refined so that the metric error is no greater than the error of differencing the flux terms. In external flow applications this process has worked quite well, including successful use with the potential flow equation (cf. [6]).

A direct means of cancelling the metric errors is the straightforward one of subtracting the error, for example, for a stationary grid

$$\delta_t\hat{Q}+\delta_\xi\hat{F}+\delta_\eta\hat{G}=F(\delta_\xi y_\eta-\delta_\eta y_\xi)+G(-\delta_\xi x_\eta+\delta_\eta x_\xi) \tag{23}$$

This puts the equation into a weak conservation law form that in principle does not degrade the shock capturing properties of the scheme. It could, however, contribute to a mild source-term weak instability that would be alleviated somewhat by spatial averaging of the right-hand side F and G flux terms. These right-hand-side flux terms can also be approximately evaluated at ∞ in somewhat duplication of (22) above.

The whole problem of differencing the metrics has been avoided in [7-10]. In this approach the Cartesian equations are expanded by chain rule and then simply left that way. That is

$$\partial_\tau Q+\xi_t\partial_\xi Q+\eta_t\partial_\eta Q+\xi_x\partial_\xi F+\eta_x\partial_\eta F+\xi_y\partial_\xi G+\eta_y\partial_\eta F=0 \tag{24}$$

and is called the quasi-linear form by Shamorth and Gibeling [10] or chain-rule conservation form by Hindman [9]. Although the Jacobian is never divided through, this form is somewhat similar to the weak conservation law form (23) just discussed, particularly so with averaging of F and G. It should also properly capture the correct jump relations. The chain-rule conservation form may well be a good compromise to differencing the transformed equations in conservative form. For certain algorithms (e.g., Beam-Warming [3]) it appears to require more work than using the strong conservation law form with free stream correction.

c) Perturbation Form Digression

The above idea of subtracting out the free stream metric variation, equation (22), discussed previously is a special case of perturbing the solution about a known function which in some sense is also a nearby or approximate solution. Let $Q = Q_0 + Q'$ where Q_0 is the nearby or approximate solution and let Q' be the perturbation. The terms of equation (3) can be rewritten as,

$$\partial_\tau \hat{Q} = \partial_\tau \hat{Q}_0 + \partial_\tau \hat{Q}'$$
$$\partial_\xi \hat{F}(Q) = \partial_\xi \hat{F}(Q_0) + \partial_\xi (\hat{F}(Q) - \hat{F}(Q_0))$$
$$etc.$$

Then assuming functions of Q_0 are sufficiently simple to be very accurately (or exactly differentiated) with operators $\bar{\delta}$, the differencing of equation (3) can be represented as

$$\begin{aligned} \delta_\tau \hat{Q}' + \delta_\xi \Big[\hat{F}(Q) - \hat{F}(Q_0)\Big] + \delta_\eta \Big[\hat{G}(Q) - \hat{G}(Q_0)\Big] \\ = -\Big[\bar{\delta}_\tau \hat{Q}_0 + \bar{\delta}_\xi \hat{F}(Q_0) + \bar{\delta}_\eta \hat{G}(Q_0)\Big] \end{aligned} \tag{25}$$

where δ_τ, δ_ξ, and δ_η represent the algorithm difference operators and $\bar{\delta}_\tau$, $\bar{\delta}_\xi$, $\bar{\delta}_\eta$, represent the very accurate differencing. In the case $Q_0 = Q_\infty$ the right-hand side is analytically zero and equation (25) is identical to equation (22).

Such a perturbation form of the differenced equation has been proposed in internal spin-up problems to remove the axisymmetric variation from the dependent Cartesian velocity variable [11,12]. It might also be used in problems in which certain fine details might be otherwise lost in a coarse grid. For example, a nonuniform incoming flow profile can be represented in Q_0 that would otherwise be lost in a far field coarse grid. Assuming in this case that Q_0 satisfies the Euler equations, the right-hand side of (25) is identically zero. Calculations using this particular technique for incoming inviscid shear flows have been tested by Buning and Steger [13]. Although not yet tried, Q_0 might be chosen as an approximate solution, in which case the right hand side of (25) would not be zero. In regions in which $Q \to Q_0$, one could hope to use a much coarser grid without losing solution accuracy.

III. CALCULATIONS ON CURVILINEAR GRIDS

Finite difference and related finite volume calculations using body fitted curvilinear grid systems have been carried out for some time. Solution variables have included velocity potential, stream function, and the primitive variables. Computed results include incompressible and compressible flow around airfoils, projectiles, cascades, inlets, wings, wing-body combinations, etc. In some cases the body deforms with time (e.g., airfoil with moving aileron) and occasionally solutions for multiple non-connected bodies appear (e.g., airfoil with detached flap). No attempt will be made to review this work—the interested reader will find much of the material in the AIAA Journal, the Proceedings of the International Conference on Numerical Methods in Fluid Mechanics, Computers and Fluids, and the Journal of Computational Physics. What is apparent from this literature is that while we are becoming more adept at solving the flow about complex configurations, considerable computational efficiencies are yet to be obtained.

In order to illustrate points to be discussed later, the results of a finite difference simulation, due to Nicolet *et al.* [14] for flow about an X-24C configuration is reproduced in Figs. 1 to 3. A head-on view of the X-24C is indicated in Fig. 1, while Fig. 2 shows typical views of the grid fit between the body surface and an analytically fit outer bow shock. The grid in this case is generated with a hyperbolic partial differential equation grid generation scheme [15]. The overall three dimensional grid is formed by generating two dimensional grids at each station along the body as the solution progresses by marching the steady parabolized Navier-Stokes (PNS) equations. The hyperbolic grid generation scheme is fast enough to be used within the flow field marching scheme. Moreover, each two dimensional grid is itself generated using the same kind of numerical algorithm used for solving the PNS equations—a sort of conservation of numerical algorithm knowledge. Computed surface pressure and heat transfer at a station just prior to the beginning of the wing are compared to experimental data in Fig. 3.

In carrying out the preceding calculation or any similar calculation on a generalized grid it is found that the solution accuracy depends on the grid. This is not surprising because unless one has a very fine mesh throughout the field, an accurate solution will require that grid points be clustered to the flow field action regions—the change of gradient

regions. The grid in Fig. 2, for example, is exceedingly fine near the body surface in order to resolve viscous gradients along the wall. (The less than perfect agreement with the experimental heating rate shown in Fig. 3 is apparently not due to inadequate resolution in this direction). The outer grid line also coincides with the bow shock, and points are clustered along the body, for example to resolve the cross flow expansion around the chime (i.e., lower right corner in Fig. 2) when the vehicle is at angle of attack (here at 6°). Otherwise no other attempt was made in this calculation to adapt the grid to computed flow field gradients.

Besides proper grid clustering, the smoothness of the grid, the skewness of the grid, and sometimes the aspect ratio of the grid can affect the accuracy of a numerical solution or the efficiency with which it is obtained. The grids shown in Fig. 2 are nearly orthogonal close to the body surface and they have a smooth, gradual variation. These grids would be judged felicitously. However, the quality of a grid seems to be hard to quantify because various numerical algorithms appear to behave differently to the properties of grid smoothness, skewness, and stretching. Numerical algorithms that use a very compact stencil of points to evaluate fluxes and metrics, for example, generally seem to be less sensitive to grid spacings that change rapidly or even discontinuously. Thus computors using the MacCormack finite volume method for Navier-Stokes equations sometimes change the grid spacing by a factor or 2 or 4 in a given region. Such a change is not allowed, for example, when using high order central spacial differencing operators.

Some numerical algorithms appear very sensitive to mesh cell aspect ratio, i.e., the ratio of Δx to Δy or $(x_\xi^2 + y_\xi^2)^{1/2}$ to $(x_\eta^2 + y_\eta^2)^{1/2}$. Thus Jameson [16] in developing a mult?xrid relaxation algorithm for the transonic potential equation abandoned SLOR iterative methods. In its place he used an alternating-direction-implicit scheme as the multigrid iterative solver because it is less sensitive to cell aspect ratio. The very efficient approximate-factorization-implicit relaxation scheme of Holst [17] appears to degrade if uniform fine grid spacing is used along the body, prompting Holst to generate his grids with this constraint in mind. For his approximate-factorization scheme the grid shown in Fig. 4 is much preferred to that shown in Fig. 5. A user of a standard alternating-direction-implicit relaxation scheme, however, may very well opt instead for the grid of Fig. 5 over

that of Fig. 4 simply because of its finer grid spacing near the body and presumably greater accuracy.

Avoiding certain undesirable grid properties such as skewness can lead to more complex computer programs and perhaps other difficulties. The cascade C-grid shown in Fig. 6 for example is too highly skewed. While the implicit Beam-Warming algorithm for the Euler or Navier-Stokes equations functions on such a grid, it runs far from optimum. An alternative grid to that shown in Fig. 6 might use an overlapped or patched grid system. For example, the overlapped grid system shown schematically in Fig. 7 is suggested because each grid is easy to generate and has minimum distortion. However, such a grid system requires extensive modification of existing numerical algorithms and computer programs. This is because certain grid points will have to be flagged off, and grid interfaces will have to be joined without degrading numerical stability. Nevertheless, overlapped or patched grid systems will ultimately be needed as body boundary configurations become more complex, for example, in computing flow about a wing with engine nacelles.

Finally, it should be remarked that the effect of a poorly spaced grid will sometimes not be observed until the data is displayed or utilized in a different way. The unpublished result due to Seidel [18] that is shown in Fig. 8 is an example. The plots of generalized pitching moment versus reduced frequency show an essentially exact solution (dashed line) and a low frequency transonic small disturbance finite difference solution with the nonlinear terms removed. The flow is about a flat plate subjected to an angle of attack pulse. The small oscillations shown in the finite difference result are a significant error in a flutter calculation. They were traced back to a discontinuous change of grid stretching more than a chord away from the airfoil, and were eliminated by using a smoothly stretched grid throughout.

IV. CONCLUDING REMARKS

Finite difference methods coupled with body conforming curvilinear grid systems are being used to solve a variety of complex flow fields. Current numerical algorithms and grids are tuned to flow field applications that can be computed in reasonable times on

present day machines. These numerical algorithms rely on sparse-equation time-accurate or iterative-solution methods that function best on well ordered grids.

Curvilinear body conforming grids have made modern finite difference schemes into practical engineering tools. They simplify the application of boundary conditions and allow flow field gradients to be resolved in an orderly manner. However, one must be careful in differencing transformed equations, especially when the equations are in conservative form and transform metrics are brought inside the difference operators. Finite difference algorithms are also sensitive to grid smoothness, skewness and stretching with some algorithms being much more adversely affected than others.

As finite difference methods are extended to more complex geometries, it becomes obvious that more than one grid system will have to be used. Exactly how multiple grids should best be joined, patched, or overset together remains a research topic, but a number of approaches will likely give satisfactory results. The simultaneous development of multiple grid systems and finite difference schemes suitable for multiple grids is underway and will be a major pacing item in computational fluid dynamics.

Acknowledgements: This work was partially supported by Army Research Office Contract DAAG29-81-K-0013 and Air Force Flight Dynamics Contract F33615-81-K-3020.

REFERENCES

1. Viviand, H., Conservative forms of gas dynamic equations, La Recherche Aerospatiale 1 (1974), 65–68.

2. Steger, J.L., Implicit finite-difference simulation of flow about arbitrary two-dimensional geometries, AIAA Journal *16* (1978), 679–686.

3. Beam, R., and R.F. Warming, An implicit finite-difference algorithm for hyperbolic systems in conservation-law-form, Journal of Computational Physics *22* (1976), 87–110.

4. Pulliam, T.H. and J.L. Steger, Implicit finite-difference simulations of three-dimensional compressible flow, AIAA Journal *18* (1980), 159–167.

5. Thomas, P.D. and C.K. Lombard, Geometric conservation law and its application to flow computations on moving grids, AIAA Journal *17* (1979), 1030–1037.

6. Steger, J.L., Implicit finite difference simulation of inviscid and viscous compressible flow, Symposium on Transonic, Shock and Multidimensional Flows, Madison, Wisconsin, May 13-15, 1981.

7. Shan, J.S. and Hankey, W.L., Numerical Solution of the Navier-Stokes Equations for a Three-Dimensional Corner, AIAA J. *15* (1977), 1575-1582.

8. Lerat, A. and Slides, J., Numerical Calculation of Unsteady Transonic Flows. Paper presented at AGARD Meeting of Unsteady Airloads in Separated and Transonic Flow, Lisbon, April 1977.

9. Hindman, R.G., Geometrically induced errors and their relationship to the form of the governing equations and the treatment of generalized mappings, Proceedings of the AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, CA. 1981.

10. Shamroth, S.J. and Gibeling, H.J., Navier-Stokes Solution of the Turbulent Flowfield About an Isolated Airfoil, AIAA J. *18*, December 1980.

11. Steger, J.L., A computer program for the internal flow of spin-stabilized liquid-filled shells. Flow Simulations Contract Report 79-03, submitted to ARRADCOM, Oct. 1979.

12. Chakravarthy, S.R., Numerical simulation of laminar incompressible flow within liquid filled shells, Rockwell International Science Center Report SC5248.2FRD submitted to ARRADCOM, Oct. 1981.

13. Buning, P.G., and J.L. Steger, Simulation of the two-dimensional Euler equations with generalized coordinate transformation using flux vector splitting. AAIA Paper No. 82-0971 to be presented June, 1982.

14. Nicolet, W.E., S. Shanks, G. Srinivasan, and J.L. Steger, Flowfield predictions about lifting entry vehicles. AIAA Paper 82-0026, 1982.

15. Steger, J.L., and D.S. Chaussee, Generation of body-fitted coordinates using hyperbolic partial differential equations, SIAM J. Sci. Stat. Comput. *1* (1980), 431–437.

16. Jameson, A., Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method. AIAA Paper No. 79-1458. Proceedings of AIAA Computational Fluid Dynamics Conference, Williamsburg, VA. July 1979.

17. Holst, T.L., and D. Brown, Transonic airfoil calculations using solution-adaptive grids. AIAA Paper No. 81-1010-CP. 1981.

18. Seidel, David A., Private Communication, NASA Langley Research Center, Hampton, VA, 1982.

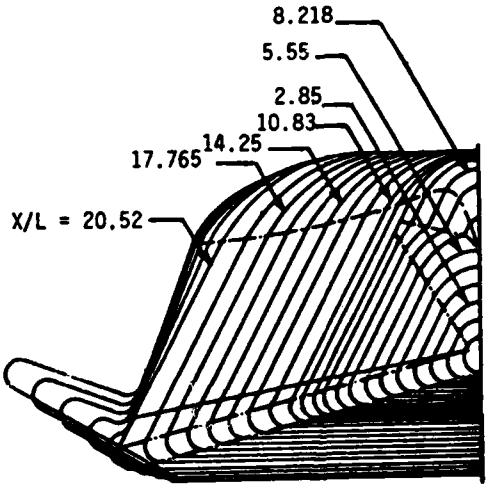


Fig. 1 X-24C configuration.

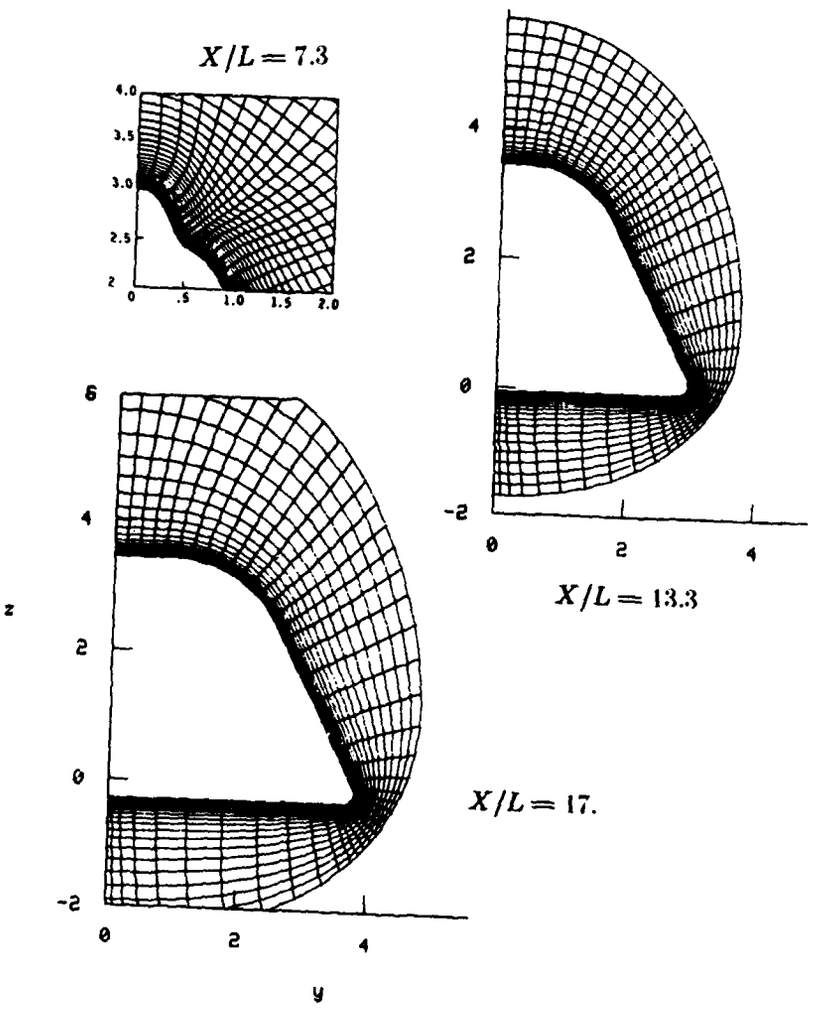


Fig. 2 Typical grids for X-24C configuration.

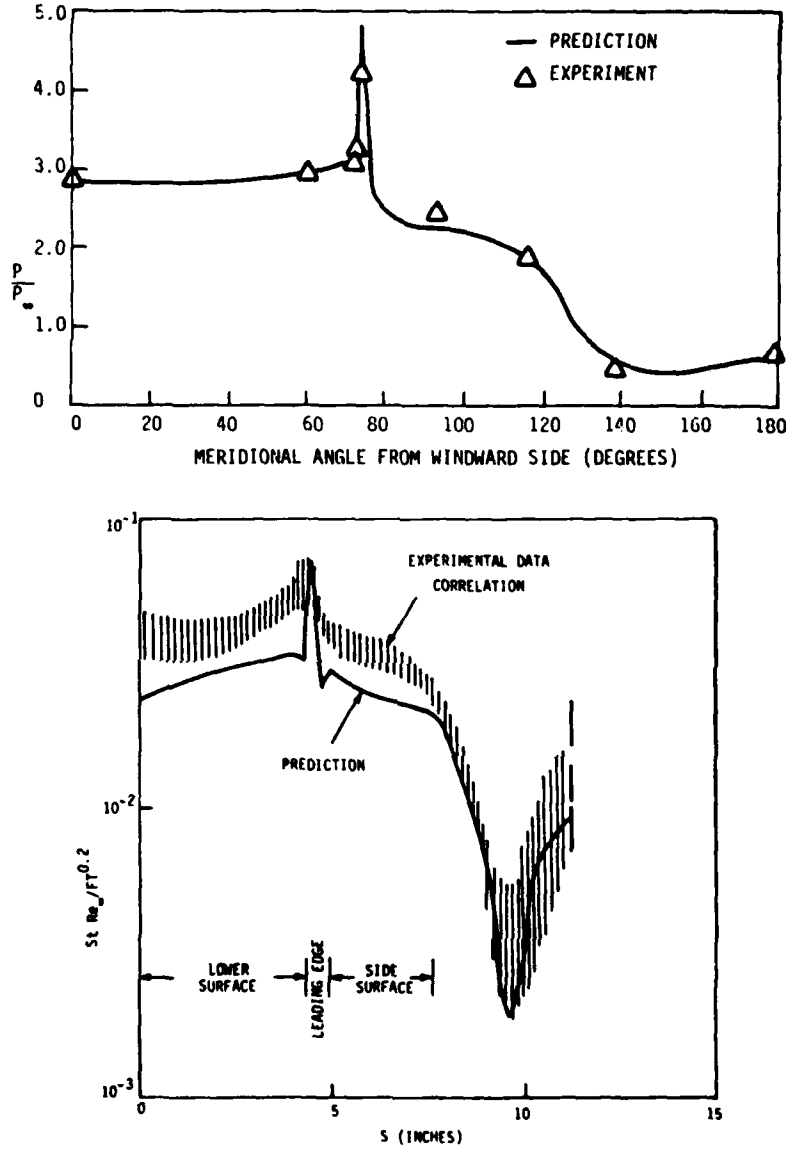


Fig. 3 Solution results obtained using parabolized Navier-Stokes equations for X-24C model at $M_\infty = 5.95$, $\alpha = 6°$, $Re = 5 \times 10^6$/ft at $x = 20.52$ inches.

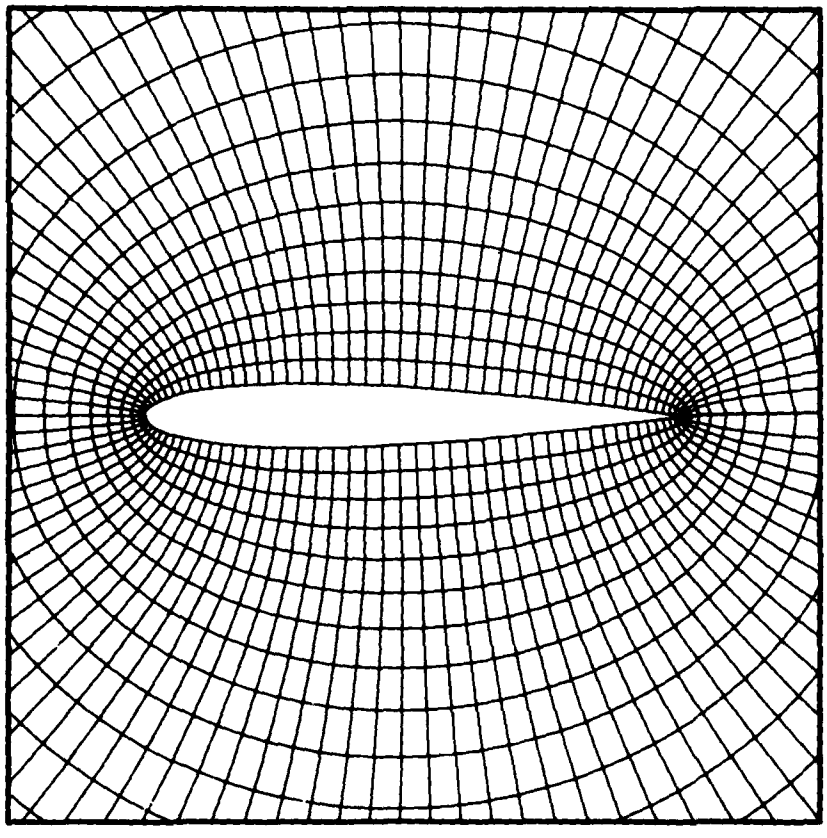


Fig. 4 Low aspect ratio grid used for inviscid flow calculations.

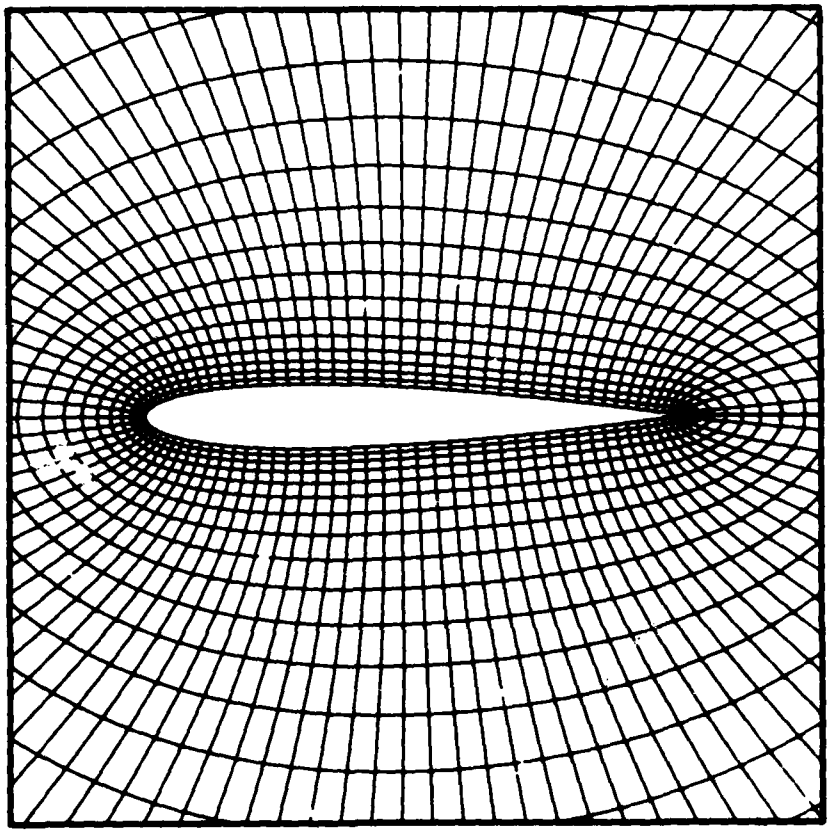


Fig. 5 Inviscid flow grid with uniform mesh spacing adjacent to the body.

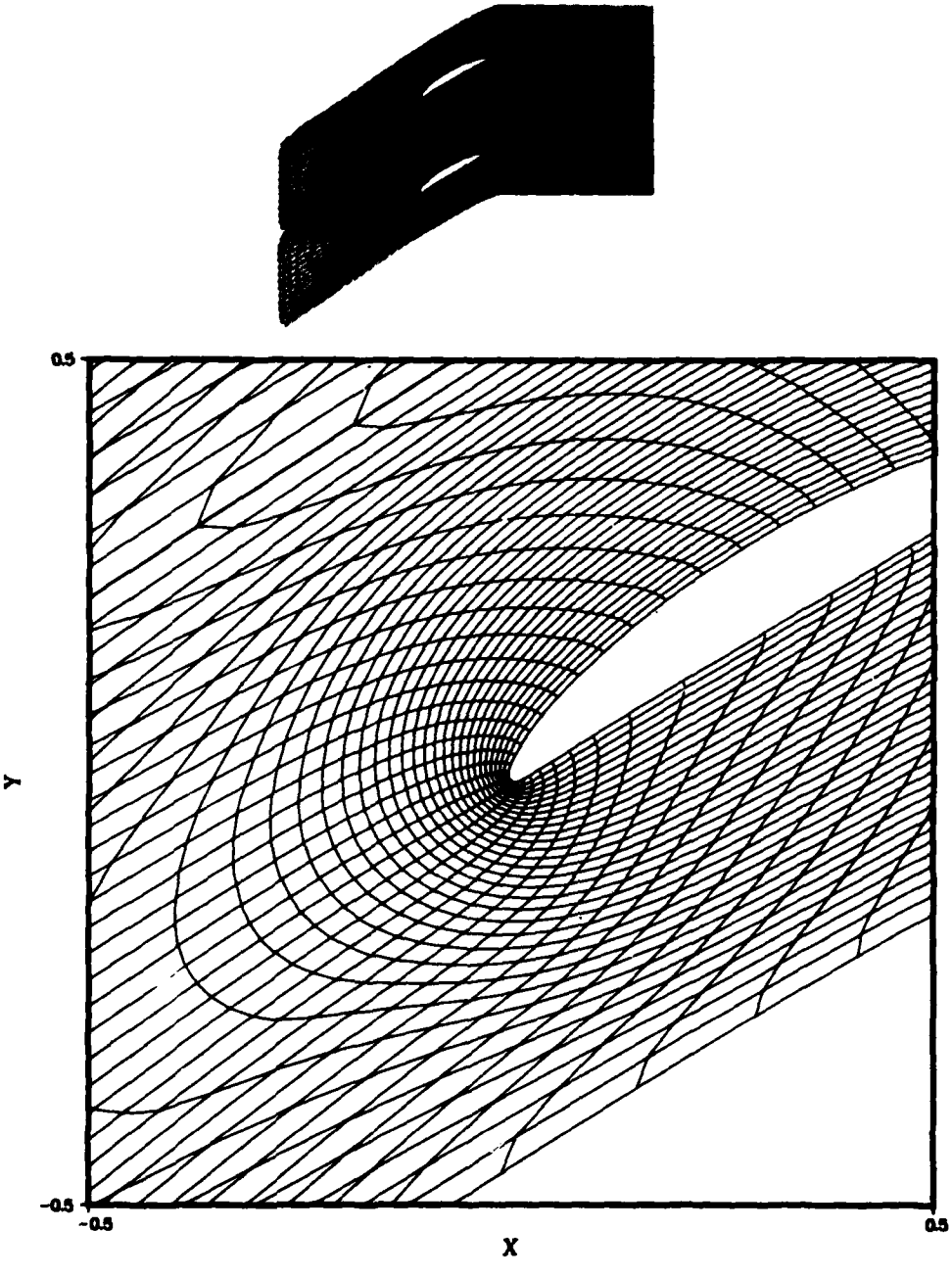


Fig. 6 Highly skewed C-grid for cascade.

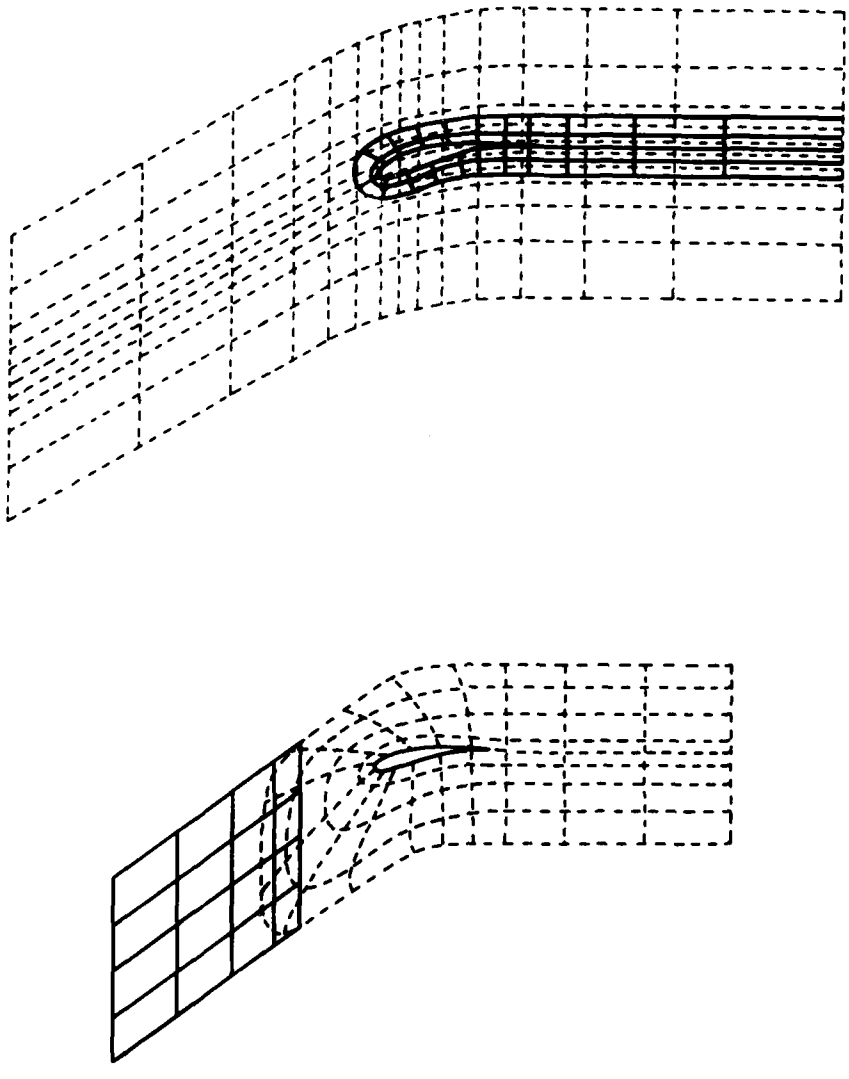


Fig. 7 Schematic indicating how overlapped grids could be used for cascade flow applications to remove grid distortion.

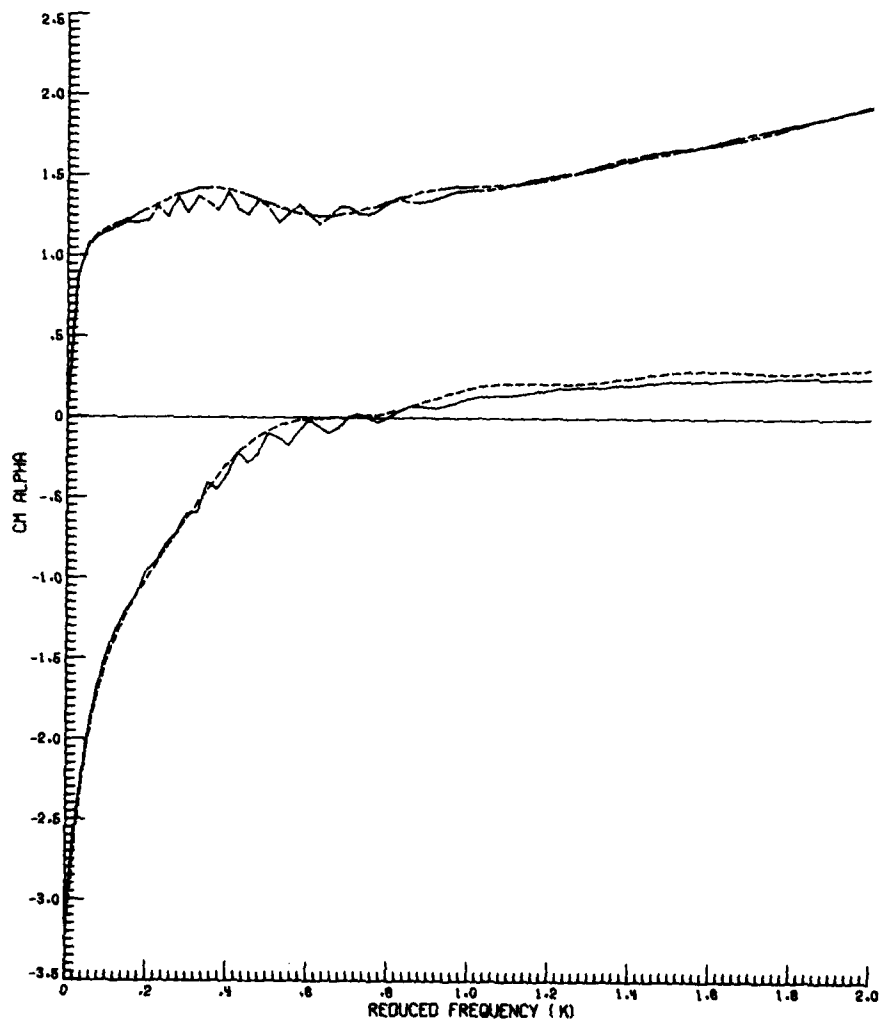


Fig. 8 Generalized pitching moment versus reduced frequence showing oscillations due to a poorly stretched grid.

AD P000978

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

THE USE OF SOLUTION ADAPTIVE GRIDS IN SOLVING PARTIAL DIFFERENTIAL EQUATIONS

DALE A. ANDERSON AND M. M. RAI
Department of Aerospace Engineering and Computational Fluid Dynamics Institute, Iowa State University, Ames, Iowa

ABSTRACT

The grid point distribution used in solving a partial differential equation using a numerical method has a substantial influence on the quality of the solution. An adaptive grid which adjusts as the solution changes provides the best results when the number of grid points available for use during the calculation is fixed. Basic concepts used in generating and applying adaptive grids are reviewed in this paper, and examples illustrating applications of these concepts are presented.

INTRODUCTION

One of the first steps in computing the numerical solution of a partial differential equation is that of choosing a coordinate system. Using this coordinate system, a grid is generated providing placement of mesh points on the region of interest in physical space. The numerical solution is usually calculated in a computational space which is related to the physical domain by a transformation. This transformation is shown schematically in Fig. 1. For simplicity, the computational domain is usually rectangular, although the region under consideration in physical space may be of arbitrary shape. Whenever possible, the physical boundaries are selected to simplify application of boundary conditions or provide some other advantage in the computation.

The distribution of grid points in physical space is established initially and usually doesn't change during the course of the calculations. However, the grid can always be restructured by interrupting the calculations after any number of steps. In order to construct a mesh which is appropriate for a specific problem, a knowledge of the solution is required at the outset. For example, high mesh point densities are desirable in high gradient regions, but if the locations of these regions are unknown, it is difficult to establish a suitable grid. Unfortunately, we seldom know the exact details of a solution before it is computed and even for cases where approximate results are known, unexpected behavior frequently occurs. Construction of an appropriate grid without an a priori knowledge of the solution is an uncertain job at best.

Since the best grid for a given problem depends on the solution, a technique

for moving grid points as the solution changes is desirable. In the ideal case, an adaptive grid scheme which provides the grid as part of the solution would be best. Numerous methods for generating grids have been developed and recently, new techniques for constructing adaptive grids have been used. A comprehensive review of all of these techniques will not be presented here. The purpose of this paper is to review in detail the methods of Rai and Anderson[1,2,3] and Anderson and Rai[4] and demonstrate the application of these schemes to various problems.

A number of different approaches can be used to construct solution adaptive grids. However, all schemes can be classified as methods which determine grid speeds or methods which determine the transformation metrics or physical coordinates. An example demonstrating the transformation of the first-order linear wave equation can be used to illustrate this point. Consider the equation

$$\frac{\partial u}{\partial t} + c\,\frac{\partial u}{\partial x} = 0 \tag{1}$$

where (x,t) are the independent variables and c is the constant wave speed. Suppose a real nonsingular mapping to computational space, (ξ,τ), is given by the relationships

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x,t) \end{aligned} \tag{2}$$

The linear wave equation in computational coordinates is of the form

$$\frac{\partial u}{\partial \tau} + (\xi_t + c\xi_x)\,\frac{\partial u}{\partial \xi} = 0 \tag{3}$$

When Eq. (3) is solved using a numerical method, the solution is obtained in the computational plane and the equation is integrated with respect to τ. Some means of evaluating the ξ_t and ξ_x terms is required before this integration can be carried out. The central problem of adaptive grid generation is that of establishing the transformation $\xi = \xi(x,t)$ required in the integration of Eq. (3).

The two terms in Eq. (3) related to the transformation from physical to computational space have a distinctly different nature. The ξ_t term is the grid speed. While it is true that the values of ξ do not change in computational space, this partial derivative is evaluated at a fixed x position and is related to the grid speed in physical space. This relationship is given by the expression

$$x_\tau = -\xi_t/\xi_x \tag{4}$$

which is much easier to visualize as the x-component of the grid speed. The metric coefficient, ξ_x, is also the Jacobian of the transformation in this simple example and we note that

$$x_\xi = 1/\xi_x \tag{5}$$

In view of the fact that both the grid speed and the transformation metric appear in Eq. (3), two ways of establishing the mapping $\xi = \xi(x,t)$ are available. The first is to evaluate the metric by using some law governing the stretching or compression in the mapping and determine the corresponding point locations in physical space by integrating the metric. This approach is easy to understand since the Jacobian relates the arc length elements in this one-dimensional example. Once the new point locations are known, the grid speed is determined by using the past history of the grid point positions. The second approach is to directly evaluate the grid speed. This is done by using an auxiliary equation for the grid speed which provides the desired control on point motion. Once the grid speed is determined, the new grid point positions in physical space are obtained by integration, and the metrics are computed by simply using the arc length ratios.

Both methods of establishing $\xi(x,t)$ have advantages and disadvantages. Techniques which determine ξ_x initially are easier to understand because the meaning of the metric is clear. The coupling of the grid dynamics to the solution of the partial differential equation lags in time although positive control over point location is always maintained. It is also difficult to extend these schemes to multidimensional problems unless grid generators, such as that of Thompson et al.[5], are used. Techniques which directly provide the grid speed are easy to use in multidimensional problems, and the dynamic motion of the grid does not lag the solution of the partial differential equation. On the other hand, control of grid point motion is not easily maintained. The choice of which technique to use depends on the philosophy employed by the investigator in viewing the grid generation problem.

The problem addressed in this paper using the concept of an adaptive grid scheme may be stated as follows: How should a fixed number of grid points be distributed to reduce the error, improve resolution, or otherwise enhance the quality of the numerical solution of a partial differential equation?

The approach favored here for use in redistributing mesh points is to determine the grid speeds directly. Once these grid speeds are known, the coordinates at the next level are easily found by integration. In addition to the choice of technique for moving points, a number of other considerations must be

made when considering construction of an adaptive grid.

1. The grid must evolve as part of the solution of the problem.
2. Grid points must move due to both boundary motion and changes in the interior solution.
3. The grid speed equations should be as simple as possible.
4. The grid speed equations must account for the elliptic nature of the problem.
5. The resulting grid must reduce error, provide better resolution, or otherwise improve the solution.
6. The adaptive grid scheme must be easily extended to any number of dimensions.

With these thoughts in mind, we now consider the development of a technique for moving grid points.

THE BASIC METHOD

In order to describe the basic adaptive grid technique as simply as possible, we consider a one-dimensional problem with the transformation from physical to computational space given by Eq. (2). Let $|e|$ represent the absolute value of any quantity to be reduced at a point such as the truncation error and let the average value over all mesh points be denoted by $|e|_{av}$. The quantity $|e|$ will be referred to as the error even though it may represent any variable used to drive the adaptive grid. When a numerical solution is computed on a grid with a fixed number of mesh points, we assume the error can be reduced by using more points in regions of large error and assigning fewer points to regions of small error. Assume that the best grid is one that has the same error at each mesh point. This grid can be constructed in the computational domain by assuming that points where $|e|$ is larger than $|e|_{av}$ attract other points and points where $|e|$ is less than $|e|_{av}$ repel other points. Every point is assumed to induce a velocity at every other point where the magnitude and direction depend upon the local excess error. We also assume that the greater the distance between two points, the less they influence each other. This can be achieved by assuming a $1/r^n$ law as an attenuation factor.

For two points, A and B, we write the grid velocity at B due to error at point A in computational space as

$$V_{BA} = K \frac{|e|_A - |e|_{av}}{r_{BA}^n}$$

where r_{BA} is the distance from B to A, n is the power controlling the

attenuation, and K is a proportionality factor. Figure 2 shows the points A and B along with the velocity induced at point B. The grid speed in physical space is obtained from V_{BA} by using the expressions

$$V_{BA} = -\xi_t$$
$$x_\tau = -\xi_t/\xi_x \tag{6}$$

For a problem with N points we may write

$$x_{\tau_j} = \frac{K}{\xi_{x_j}}\left[\sum_{i=j+1}^{N}(|e|_i - |i|_{av})/r_{i,j}^n - \sum_{i=1}^{j-1}(|e|_i - |e|_{av})/r_{i,j}^n\right] \tag{7}$$

The summation is split and a sign change occurs because the positive direction for induced velocity switches depending on whether the point is to the left or right of point j. Equation (7) can be viewed like a gravitational force equation where the grid speed plays the role of force and the excess error divided by ξ_x plays the role of mass. This may also be compared to the force on a test charge in an electrostatic field.

In constructing either a fixed or adaptive grid, it is important that grid lines do not cross. In this example, two points in computational space will not collapse into a single point in physical space for two reasons.

1. The force which drives the grid

$$g = |e| - |e|_{av} \tag{8}$$

switches signs when two points approach each other at close range and this creates a repulsive effect.

2. The ξ_x term becomes very large as two points get close together leading to a very stiff system. For this reason, even if points continue to attract each other, they will never reach the same physical point.

The term $|e|$ has been referred to as a truncation error in the above discussion. The truncation error for a numerical solution depends on the order of the numerical method used. If a quantity other than truncation error is used, the grid generation scheme can be applied in exactly the same way by using a different choice for $|e|$. If resolution of high gradient regions is desired, then the gradient may be used. This flexibility in selecting the driving force is an advantage of the technique.

The scaling factor K in the grid speed equation is necessary because no physical law relates grid speed, excess error, and distance. This factor is chosen so the velocity at any point does not exceed a preset maximum $|\xi_t|_{max}$.

In order to calculate K, we compute the grid speed at all points assuming K = 1, then recalculate K using the expression

$$K = \frac{|\xi_t|_{max}}{|\xi_t|_{computed\ max}} \tag{9}$$

The grid speeds at every point are now rescaled and written as

$$(\xi_t)_{rescaled} = K(\xi_t)_{calculated} \tag{10}$$

During the calculations, the driving force, Eq. (8), at each point becomes smaller, and K must be increased in order to maintain the preset maximum grid speed. The maximum value of K is also preset and once this maximum is reached, the grid point velocities die out very rapidly. Specifying a large value of K_{max} results in small values of the excess error in the grid. The selection of K_{max} is done on an empirical basis. If a value of K_{max} which is too large is used, the result is an oscillation of the grid, while a small value severely restricts the grid point speeds. The maximum grid speed is also determined in a somewhat arbitrary fashion. The most common value of $(\xi_t)_{max}$ used is 1.0.

An example of results produced using the adaptive grid ideas presented here is given by the numerical solution of the unsteady viscous Burgers' equation

$$u_t + uu_x = \mu u_{xx} \tag{11}$$

with initial conditions

$$u(0,x) = \begin{cases} 1 & x = 0 \\ 0 & 0 < x \leq 1 \end{cases} \tag{12}$$

and the boundary conditions

$$\begin{aligned} u(t,0) &= 1 \\ u(t,1) &= 0 \end{aligned} \tag{13}$$

This problem has the steady-state solution

$$u = \hat{u} \tanh\left[\frac{\hat{u}Re}{2}(1 - x)\right] \tag{14}$$

where

$$Re = \frac{1}{\mu} \tag{15}$$

and $\hat{u}$ is a solution of the equation

$$\frac{\hat{u} - 1}{\hat{u} + 1} = \exp(-\hat{u}Re) \tag{16}$$

Figure 3 presents results obtained for this problem with Re = 4. The error measure used to drive the grid in this case was either u_ξ or u_ξ/ξ_x. Using first derivative information directly addresses the question of resolution and less directly the error reduction issue. In this example, MacCormack's[6] second-order scheme was used to integrate Burgers' equation. If the finite-difference equation is expanded using Taylor series, we obtain the modified partial differential equation which explicitly includes the truncation error terms due to the discretization. The form of the lowest-order error term depends on the order of the finite-difference method being used. The lowest-order truncation error term in this case includes both a third derivative and fourth derivative term.

If higher derivatives of the solution are formed using finite-difference methods, the result is a very noisy error estimate. As a result, large amounts of smoothing or some other means of avoiding wild swings in data used to drive the grid must be employed. Grid oscillations and excessive clustering of points can be prevented by damping the grid speeds permitted by limiting the change in the Jacobian at each point. Application of this idea prevents excessive stretching of the mesh and the associated errors in the transformation metrics.

Such a method for controlling the mesh can be implemented by computing the Jacobian at all points initially and storing these values. At the end of each integration step, the Jacobian J_i^K is formed at every point, and the ratio R_i is formed.

$$R_i = \begin{cases} J_i^K/J_i^1 & \text{if} \quad J_i^K/J_i^1 > 1 \\ J_i^1/J_i^K & \text{if} \quad J_i^K/J_i^1 < 1 \end{cases} \tag{17}$$

If the ratio R_i at any point exceeds a preset maximum value, R_{max}, the grid speeds are exponentially damped. If

$$R_{i_{max}} = \text{maximum}(R_i) \text{ over all points} \tag{18}$$

then

$$\xi_{t_i} = D\xi_{t_{i_{calculated}}} \tag{19}$$

where

$$D = \exp\left[-\beta\left(\frac{R_{i_{max}}}{R_{max}}\right)^2\right] \tag{20}$$

The use of this damping factor provides strong control over grid point motion. In the results using u_ξ shown in Fig. 3, there is a simpler way of preventing excess stretching. In this case, smoothing can be applied by using

$$\bar{u} = fu + (1 - f)(1 - x) \qquad 0 \leq f \leq 1 \tag{21}$$

and

$$g = |\bar{u}_\xi| - |\bar{u}_\xi|_{av} \tag{22}$$

This provides excellent grid point control for this problem. Using this approach requires that the best value of f be computed. It is also apparent that the technique is problem dependent. However, the Jacobian damping scheme is not problem dependent and is highly recommended.

The results using u_ξ/ξ_x as an error estimate are also shown in Fig. 3. No damping of any kind was used to obtain the error curves shown. The grid driving force used was of the form

$$g = |u_\xi/\xi_x| - |u_\xi/\xi_x|_{av} \tag{23}$$

This particular form of the driving force results from considering the truncation error in the second-order central difference approximation to a first derivative in physical space. This error may be written in the form

$$\text{error}(u_x) \propto \frac{\Delta x^2}{6}\left(u_\xi \xi_{xxx} + 3\xi_x \xi_{xx} u_{\xi\xi} + \xi_x^3 u_{\xi\xi\xi}\right) \tag{24}$$

This is similar to the truncation error term which appears in the modified partial differential equation and should represent a good error estimate. If it is assumed that $\xi(x,t)$ is nearly linear in x, second and third derivatives of ξ may be neglected. Suppose we also assume that $|u_{\xi\xi\xi}|$ can be replaced by $|u_\xi|$ in the error estimate. Using these approximations, the error in computing u_x may be written as

$$|\text{error}(u_x)| \simeq |u_\xi/\xi_x| \tag{25}$$

Consistent with these assumptions, the larger the Reynolds number, the more results deteriorate when Eq. (23) is used as a grid driving force. In the high gradient regions at the viscous front, the grid stretching is very nonlinear and the higher derivatives of the mapping are important. However, the use of the driving force given by Eq. (23) provides significant reduction in error without any smoothing. This eliminates any empiricism in arriving at an optimum smoothing technique.

EXTENSION TO MULTIDIMENSIONAL PROBLEMS

The grid generation scheme presented above for a one-dimensional problem can easily be extended to two or three space dimensions with no difficulty. Consider a problem in two space dimensions and time, and let the physical coordinates be (x,y,t) and the computational coordinates be (ξ,η,τ) where

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x,y,t) \\ \eta &= \eta(x,y,t) \end{aligned} \tag{26}$$

As in the one-dimensional problem, we assume that the grid speeds in computational space are given by ξ_t and η_t and are determined by prescribing a certain grid driving function. Once these quantities are known, the grid speeds in physical space can be determined from the transformation equations and may be written

$$\begin{aligned} x_\tau &= \frac{\eta_t \xi_y - \xi_t \eta_y}{J} \\ y_\tau &= \frac{\xi_t \eta_x - \xi_x \eta_t}{J} \end{aligned} \tag{27}$$

where

$$J = \xi_x \eta_y - \eta_x \xi_y \tag{28}$$

The collapse of two mesh points into one and the overlap of grid lines are prevented for the same reasons noted in the one-dimensional case.

In order to compute the grid speeds in computational space, we require the calculation of $|e^\xi|$ and $|e^\eta|$ for each point where e^ξ and e^η represent error estimates in the ξ and η directions. The quantities $|e^\xi|_{av}$ for every η = constant line and $|e^\eta|_{av}$ for every ξ = constant line must also be computed. The grid speed equations then become

$$\begin{aligned} (-\xi_t)_{i,j} &= K_1 \sum_{\ell=1}^{M} \left[\sum_{k=i+1}^{N} \frac{|e^\xi|_{k,\ell} - |e^\xi|_{av_\ell}}{r^n} - \sum_{k=1}^{i-1} \frac{|e^\xi|_{k,\ell} - |e^\xi|_{av_\ell}}{r^n} \right] \\ (-\eta_t)_{i,j} &= K_2 \sum_{k=1}^{N} \left[\sum_{\ell=j+1}^{M} \frac{|e^\eta|_{k,\ell} - |e^\eta|_{av_k}}{r^n} - \sum_{\ell=1}^{j-1} \frac{|e^\eta|_{k,\ell} - |e^\eta|_{av_k}}{r^n} \right] \end{aligned} \tag{29}$$

where

$$r = \sqrt{(i-k)^2 + (j-\ell)^2} \tag{30}$$

assuming $\Delta\xi = \Delta\eta = 1$ in the computational domain. Point motion can be controlled and excessive stretching or compression of the grid is avoided by using damping based upon the forms given in Eq. (19) with the D factor for both ξ_t and η_t determined from the Jacobian of the transformation. In two space dimensions, the Jacobian represents the mesh area ratios rather than the arc lengths noted in the one-dimensional case.

BOUNDARY POINT SPEEDS

Points near the boundaries and the boundary points need special treatment when an adaptive grid is used. Points can be made to move along a constant η line by specifying $\eta_t = 0$. A similar procedure can be used for ξ = constant lines. This permits grid points to move along boundaries or selected surfaces. If this procedure is followed on boundaries, undesirable behavior appears in the resulting grid. This is most easily demonstrated by an example.

The two-dimensional linearized viscous Burgers' equation is of the form

$$u_t + u_x + u_y = \mu(u_{xx} + u_{yy}) \tag{31}$$

If this equation is solved on the unit square with the boundary conditions

$$u(x,0,t) = 1 + \frac{[1 - \exp(Re(x - 1))]}{1 - \exp(-Re)}$$

$$u(0,y,t) = 1 + \frac{[1 - \exp(Re(y - 1))]}{1 - \exp(-Re)}$$

$$u(x,1,t) = u(1,y,t) = 1 \tag{32}$$

an analytic solution for the steady state is known and is given by

$$u = 1 + \frac{[1 - \exp(Re(x - 1))][1 - \exp(Re(y - 1))]}{(1 - \exp(-Re))^2} \tag{33}$$

where

$$Re = 1/\mu$$

An adaptive grid was generated for this two-dimensional problem. Gradients were used to drive the grid and the first derivatives were calculated directly from the given analytical solution. The grid obtained using this approach is shown in Fig. 4. The distortion near the boundaries is evident. These results were computed by forcing grid points to move tangential to the boundaries by setting either ξ_t or η_t equal to 0. While the grid produced using this boundary procedure is shaped as expected, motion of the points near the boundary requires more careful study.

Consider a one-dimensional problem where the quantity $|e| - |e|_{av}$ is nearly constant close to the boundary. If we compute the velocity of the second grid point with K = 1 and n = 1, [see Eq. (7)],

$$-(\xi_t)_2 = [|e| - |e|_{av}]\left[\frac{1}{r} + \frac{1}{2r} + \frac{1}{3r} + \ldots - \frac{1}{r}\right] \tag{34}$$

which results in a nonzero grid speed. This places an unnatural constraint on grid speed which produces the grid distortion shown in Fig. 4. This problem is eliminated by introducing a set of pseudo points outside the boundary and setting

$$|e|_j = |e|_{2-j} \qquad j = 0, -1, \ldots \tag{35}$$

and including the pseudo points in calculating the grid speeds. The condition

$$(\xi_t)_1 = 0 \tag{36}$$

on the boundary is automatically satisfied by this scheme. This procedure was used to compute the grid for the same Burgers' equation problem where grid distortion was noted. Figure 5 shows the result. While the results in this case are not grossly different, the addition of the reflected boundary with pseudo points certainly produces a smoother grid. This boundary point treatment is necessary when an adaptive grid is used in more complex problems. An exact solution for Burgers' equation is known for this example and comparison of error for adaptive vs. nonadaptive grids can be made. However, these comparisons will be omitted in the interest of brevity. They show the same trends as the one-dimensional problem and results are presented in detail in Rai's Ph.D. dissertation.[7]

The two simple examples outlined above demonstrate the technique used for constructing an adaptive grid. When a fluid dynamics problem is solved, numerous complexities that are not present in simple examples always appear. A good test case to demonstrate this is the inviscid supersonic flow over a cylinder. In computing the results for this problem, the unsteady equations of motion are integrated in time until the steady-state solution is achieved. The bow shock is fit as a boundary and the flow field was computed using the SCM[8] method for solving hyperbolic partial differential equations. Results were computed on 10 x 10 point fixed and adaptive grids and compared to a 19 x 19 point fixed grid solution. The fixed grids were generated by equally spacing points along the cylinder, drawing rays perpendicular to the cylinder to the boundary shock wave, and then dividing each ray into equal segments.

This example provides a number of new complexities which must be accounted

for when using an adaptive grid. These include fixed curved boundaries, moving boundaries, solution of a system of partial differential equations, and the associated difficulty of not having a predominant variable that can be used to drive the grid. In order to understand these differences, it is worthwhile to examine this problem in some detail.

The Euler equations in two space dimensions and time may be written in the form

$$\vec{W}_t + [A]\vec{W}_x + [B]\vec{W}_y = 0 \tag{37}$$

where

$$\vec{W} = \begin{bmatrix} \rho \\ u \\ v \\ p \end{bmatrix} \quad [A] = \begin{bmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & 1/\rho \\ 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & u \end{bmatrix} \quad [B] = \begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 1/\rho \\ 0 & 0 & \gamma p & v \end{bmatrix}$$

In these expressions, p and ρ are the pressure and density, u and v represent the velocity components in the x and y directions, and γ is the ratio of specific heats. If a transformation such as that specified in Eqs. (26) is made, the governing equations may be written

$$\vec{W}_\tau + [\hat{A}]\vec{W}_\xi + [\hat{B}]\vec{W}_\eta = 0 \tag{38}$$

where

$$[\hat{A}] = \xi_t[I] + \xi_x[A] + \xi_y[B] \tag{39}$$

$$[\hat{B}] = \eta_t[I] + \eta_x[A] + \eta_y[B] \tag{40}$$

and I is the identity matrix.

The SCM scheme used to calculate a steady-state solution to this problem is second-order accurate except at the solid surface boundaries where it is a first-order scheme. Since this renders the solution formally first-order accurate, the second derivative of $\vec{W}$ represents a good estimate of the truncation error. If we assume that errors in calculating the metrics are small, the error in calculating W_τ may be represented as

$$e(\vec{W}_\tau) = -[\hat{A}]e(W_\xi) - [\hat{B}]e(W_\eta) \tag{41}$$

where e( ) represents the error in calculating its argument. We let

$$e^\xi = |[\hat{A}]e(\vec{W}_\xi)|$$
$$e^\eta = |[\hat{B}]e(\vec{W}_\eta)| \tag{42}$$

and note that

$$|e(\vec{W}_\tau)| \leq |[\hat{A}]e(\vec{W}_\xi)| + |[\hat{B}]e(\vec{W}_\eta)| \leq e^\xi + e^\eta \tag{43}$$

In these expressions e^ξ and e^η represent the errors that are used to produce grid motion in the respective directions and the ℓ^1 norm is used where noted. The truncation error is assumed to be proportional to the second derivative which is consistent with the first-order accuracy (overall) of the method used to solve the problem. We write this as

$$e(\vec{W}_\xi) \propto \vec{W}_{\xi\xi}$$

$$e(\vec{W}_\eta) \propto \vec{W}_{\eta\eta}$$

which provides the error terms for driving the grid as

$$e^\xi = |[\hat{A}]\vec{W}_{\xi\xi}|$$

$$e^\eta = |[\hat{B}]\vec{W}_{\eta\eta}| \tag{44}$$

This analysis assumes that excessive stretching is avoided in the transformation. This is assured by using the Jacobian damping outlined in Eqs. (17) - (20) for both grid speeds. The use of this technique prevents both excessive stretching and contraction and provides a means of exerting positive control over the grid point motion. For the cylinder problem, a value of R_{max} of 1.03 was used. For this value of R_{max}, only small distortions of the grid occur. If a larger value of R_{max} is used, the adaptive grid changes significantly.

Figure 6 shows the fixed grid used in computing the flow field. This grid also forms the initial mesh for the adaptive grid calculations. Figure 7 shows the converged adaptive grid. As noted, there are only small changes in the grid geometry for this problem and the similarity of Figs. 6 and 7 verifies this. Figure 8 shows the converged adaptive grid using a 16 x 10 point mesh and an R_{max} of 6.0. The increase in clustering for larger values of R_{max} is evident. Figure 9 shows the error in total enthalpy along the body surface for grids. In the solution for inviscid supersonic flow over a cylinder, the correct total enthalpy is known. Consequently, the comparison of the total enthalpy computed from the numerical result with the correct known value is a good measure of the quality of the solution. As can be seen, the adaptive grid does significantly reduce the error on the body.

The motion of the shock boundary does not require any special treatment in the adaptive grid routine. The shock is propagated separately using the correct jump and shock speed equations, and the adaptive grid scheme is applied

independently. Since the body boundary is curved some special considerations must be given to point motion on the surface. Using the pseudo point idea described earlier provides a zero grid point speed normal to the body. However, integration of the tangential velocity will slightly displace the grid points from the body. At the end of each integration of the grid speed equations, the position of the boundary points must be corrected to insure that they continue to lie on the curved body.

SHOCK ALIGNING SCHEMES

The technique for moving grid points presented in the previous sections can be used to create a grid using a variety of driving mechanisms. The flexibility afforded by directly establishing the grid speed, as opposed to other adaptive grid schemes, provides a way of moving points to satisfy any desired requirement. A demonstration of shock aligning grids for use with shock-capturing methods will demonstrate this point.

When numerical solutions of systems of hyperbolic partial differential equations are obtained, provision must be made for the occurrence of discontinuities. In inviscid high speed flow problems, these discontinuities are shock waves. Lax[9] proposed that numerical solutions of fluid flow problems with shocks should be computed by using the conservation law form of the governing equations. The solution for the flow across any shocks can then be computed without any special treatment. This approach has been used with success since Lax originally proposed the idea. However, solutions for flow through shock waves obtained using shock-capturing techniques are characterized by dispersive errors when second-order finite-difference methods are used, and the solution is spread over several mesh intervals when dissipative schemes are used. The typical oscillations which occur are due primarily to the fact that the mesh is not aligned with the shock. Under these conditions, the conservative variables are not continuous across the shock and differencing these quantities across the shock gives rise to oscillations. In order to understand this phenomena, we need to examine the conditions which must hold across shock waves.

The conservative form of the inviscid equations for two-dimensional steady supersonic flow may be written

$$\frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} = 0 \tag{45}$$

where

$$\vec{E} = [\rho u,\ p + \rho u^2,\ \rho uv]^T$$

and

$$\vec{F} = [\rho v,\ \rho uv,\ p + \rho v^2]^T$$

The fluid variables have the same meaning as before. In addition to Eq. (45), the integrated form of the energy equation is used. When Eq. (45) is solved using finite-difference methods, the solution exhibits dispersive behavior at shock waves because the conservative variables are discontinuous. Since a solution with a shock mathematically represents a weak solution of Eq. (45), the condition which must be satisfied at the shock may be written[10]

$$[\vec{E}]\cos\alpha_1 + [\vec{F}]\cos\alpha_2 = 0 \tag{46}$$

where the square brackets represent the jump in the function across the discontinuity. The direction cosines of the normal to the shock and the x and y axes are denoted by $\cos\alpha_1$ and $\cos\alpha_2$ as shown in Fig. 10. If a coordinate system is selected in such a way as to align one of the coordinates with the shock, say $\alpha_2 = 90°$, the jump condition given in Eq. (46) becomes

$$[\vec{E}]\cos\alpha_1 = 0 \tag{47}$$

This shows that the E vector is continuous across the shock since the jump in $\vec{E}$ is zero for $\cos\alpha_1 \neq 0$. For any other coordinate orientation, $\vec{E}$ is discontinuous and we would not expect finite differences across such a discontinuity to yield a well-behaved result. If an adaptive grid system can be constructed that automatically aligns one of the coordinates with the shock, better solutions of Eq. (45) would be expected.

Consider a line segment on a ξ = constant line between two points A and B (see Fig. 11). Let the midpoint of this segment be O and suppose a shock wave occurs between points A and B. Let h be any variable which changes discontinuously through the shock. An adaptive grid which aligns with the shock can be constructed using this information. If C is another point in computational space, we define

$$(-\xi_t)_C = \frac{K|h_\xi|_O|h_\eta|_O(-1)^k}{r_{OC}^n} \tag{48}$$

where $(\xi_t)_C$ is the grid speed at C due to the changes in h along line AB, $|h_\xi|_O$ and $|h_\eta|_O$ are the absolute values of the gradients in h at point O in the ξ and η directions, K and n are constants, and r_{OC} is the distance between points O and C in computational space. The integer, k, determines the direction of point motion and is given by

$$k = \begin{cases} 1 & \operatorname{sgn}(h_\xi/h_\eta)\operatorname{sgn}(\eta_o - \eta_c) < 0 \\ 2 & \operatorname{sgn}(h_\xi/h_\eta)\operatorname{sgn}(\eta_o - \eta_c) > 0 \end{cases} \tag{49}$$

As in the previous adaptive grid schemes, the r_{oc}^n term limits the radius of influence of various points and restricts the grid alignment to a limited region. As a line segment rotates more nearly parallel to h = constant surfaces, the gradient in one direction decreases, and we must increase the value of K in order to maintain reasonable grid speeds. As previously noted, K is increased to maintain a maximum grid speed until a maximum value of K is reached. This maximum value is usually chosen to be 5 to 10 times the value of K for the initial calculations. If K is too large, the grid may oscillate and the oscillations die out very slowly.

The effectiveness of the aligning grid scheme was evaluated by solving a unit problem. This consisted of specifying the dependent variable, h, on the unit square by the expression

$$h(x,y) = \frac{1}{2}\left[1 + \frac{\tanh\left[\frac{4}{\beta}\,(x - x_m(y))\right]}{\tanh(2)}\right] \qquad \begin{matrix} 0 \le x \le 1 \\ 0 \le y \le 1 \end{matrix} \tag{50}$$

where β is a constant and x_m is a prescribed function of y. Along any y = constant line, h increases monotonically and over 95% of the change in h occurs in the region given by

$$x_m(y) - \beta/2 \le x \le x_m(y) + \beta/2 \tag{51}$$

Different shock-like regions can be obtained by choosing different functions for $x_m(y)$. Figures 12 and 13 show the grid created using x_m as a linear function or a quadratic function of y. The shaded regions correspond to those defined by Eq. (51). The alignment of the grid is excellent near the center of the regions where the maximum gradients of h occur and not as good near the boundaries. From these curves it is apparent that the alignment is a local effect which quickly attenuates with distance from the high gradient region.

To demonstrate the effectiveness of the shock aligning scheme in conjunction with shock-capturing finite-difference methods, the flow field due to a straight oblique shock in a uniform supersonic freestream was computed. Figure 14 shows the location of the shock wave in a uniform grid. The angle of misalignment is twenty degrees. The flow is from the upper part of the picture toward the bottom at a freestream Mach number of 2.0 with the angle between the freestream and the shock wave set at 50°.

The two-dimensional unsteady Euler equations govern inviscid supersonic

flow. In the interest of conserving space, they will not be repeated here. The Euler equations in conservation form were written in (x,y,t) coordinates and transformed to (ξ,η,τ) using a transformation

$$\xi = \xi(x,y,t)$$

$$\eta = \alpha y \tag{52}$$

$$\tau = t$$

where α is a constant. The Euler equations in this transformed system were integrated in time using MacCormack's method until the steady-state solution was obtained. The grid points were moved using the shock aligning expression given in Eq. (47) using the static pressure as the h variable. The position of the shock wave in the converged grid is shown in Fig. 15. Again the alignment is excellent. Figures 16 and 17 show the pressure distributions at y = 0.208 and y = 0.0 for both aligning and nonaligning grids. The solutions using the shock aligning grid are much better than those computed on a uniform mesh. The absence of dispersive oscillations and the resolution of the shock are striking. These results show that the grid can be aligned with a high gradient region as well as clustered by using a slightly different form of the grid speed equation. The results for the simple aligning problems studied offer encouragement in applying aligning schemes to more sophisticated problems with multiple shocks.

TIME REQUIREMENTS

The computer time required to solve a problem using an adaptive grid is usually higher than that required using a fixed mesh when explicit techniques are employed. This is because the allowable step sizes are reduced where fine mesh clustering occurs. In the solution of the one-dimensional Burgers' equation, the high Re cases take roughly three times longer than the fixed mesh solutions while the two-dimensional problems differ by approximately a factor of two. Since the problem under consideration in these cases is very simple, a large portion of the computer time is consumed by the grid calculation. As the complexity of the problem increases and systems of equations are solved, the time spent on grid generation compared to that solving the equations is drastically reduced. It should also be noted that the time required to compute a solution does not always increase. In many cases, we observe more rapid convergence when an adaptive grid is used resulting in lower overall computer time requirements. This behavior is limited to the more sophisticated examples such as the blunt body problem. When these types of problems are solved,

uncertainty exists concerning the total time required. Sometimes the fixed mesh solution is faster, and sometimes the adaptive grid solution takes less time.

CONCLUDING REMARKS

Methods for constructing solution adaptive grids which directly determine grid speeds have been reviewed. The grid speed is assumed to depend upon local truncation errors throughout the mesh and the new grid point locations are obtained by integrating the grid speed equations.

Results presented show that significant error reduction is achieved when adaptive grids are used. Solutions for simple one- and two-dimensional problems as well as more complex fluid flows were presented. The flexibility of the grid generation technique was demonstrated by constructing a shock aligning grid for use with shock-capturing methods. Solutions obtained with this scheme alleviated the dispersive error, usually associated with calculations through shock waves.

Although the results obtained using the adaptive grid schemes reviewed in this paper are satisfactory, significant improvements remain to be made. One area where improvements are needed is in establishing error estimates. Since the quality of an adaptive grid is usually based at least to some extent on local truncation errors, better grids can be constructed if better error estimates are available. Better techniques for deriving grid speeds are also needed. These techniques should be more responsive to local solution changes and still provide a stable, error reducing grid. The empiricism required in using adaptive grid schemes should also be eliminated and methods that are completely automated should be developed.

Since the adaptive grid field is new, everyone should be encouraged to develop new ideas and techniques which look promising. New concepts should be carefully studied with the hope of constructing better methods for computing solutions to partial differential equations.

ACKNOWLEDGMENTS

Figures 6, 7, 9, 12-17 from AIAA papers 81-0114 and 81-1012 are reproduced with the permission of the American Institute of Aeronautics and Astronautics.

Preparation of this paper was supported by NASA Langley Research Center through NASA Cooperative Agreement NCCI-17.

REFERENCES

1. Rai, Man Mohan and Anderson, D.A., "Grid evolution in time asymptotic problems," J. Comput. Phys. 43(1981), pp. 327-344.
2. Rai, M.M. and Anderson, D.A., "Application of adaptive grids to fluid-flow problems with asymptotic solutions," AIAA Journal 20(1982), pp. 496-502.
3. Rai, M.M. and Anderson, D.A., "The use of adaptive grids in conjunction with shock-capturing methods," Presented at the AIAA 5th Computational Fluid Dynamics Conference; AIAA Paper 81-1012 (June 1981).
4. Anderson, D.A. and Rai, M.M., "A new approach to solution adaptive grids," Presented at the ASME Winter Meeting published in proceedings of the ASME/AIAA Symposium on Computers in Flow Predictions and Fluid Dynamics Experiments (November 1981), p. 87.
5. Thompson, J.F., Thames, F.C., and Mastin, C.M., "Automatic numerical generation of body-fitted curvilinear coordinate systems for field containing any number of arbitrary two-dimensional bodies," J. Comput. Phys. 15(1974), pp. 229-319.
6. MacCormack, R.W., "The effect of viscosity in hypervelocity impact cratering," Presented at the AIAA Hypervelocity Impact Conference; AIAA Paper 69-354 (May 1969).
7. Rai, M.M., "A philosophy for construction of solution adaptive grids," Ph.D. Dissertation, Aerospace Engineering Department, Iowa State University, Ames, Iowa (1982).
8. Chakravarthy, S.R., Anderson, D.A., and Salas, M.D., "The split coefficient matrix method for hyperbolic systems of gas dynamic equations," AIAA Paper 80-0268 (January 1980).
9. Lax, P.D., "Weak solutions of nonlinear hyperbolic equations and their numerical computation," Comm. Pure and Applied Math. 7(1954), pp. 159-193.
10. Whitham, G.B., Linear and Nonlinear Waves (Wiley, New York, 1974).

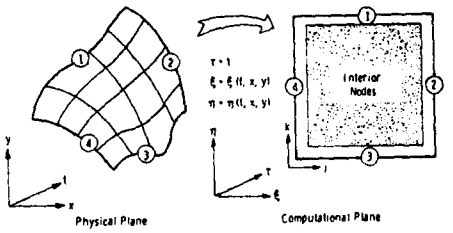


Fig. 1. General mapping between physical and computational space.

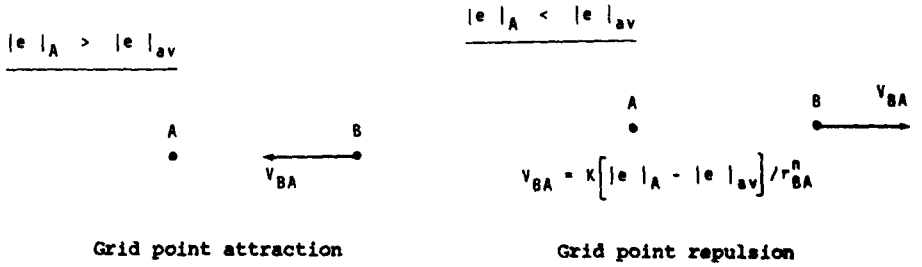


Fig. 2. Mesh point speed definition.

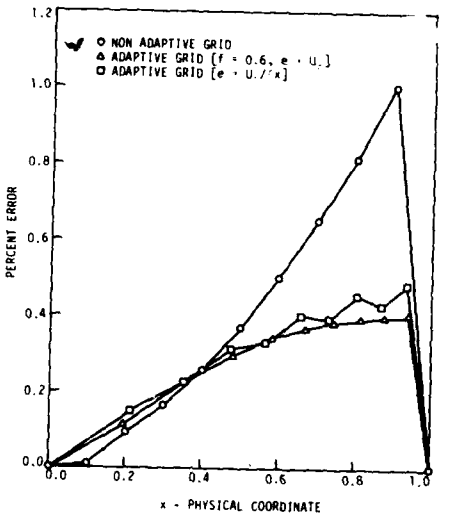


Fig. 3. Comparison of errors for Burgers' equation, Re = 4.

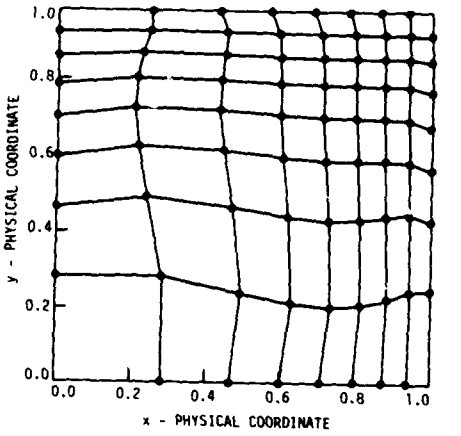


Fig. 4. Converged grid using non-reflected boundary points.

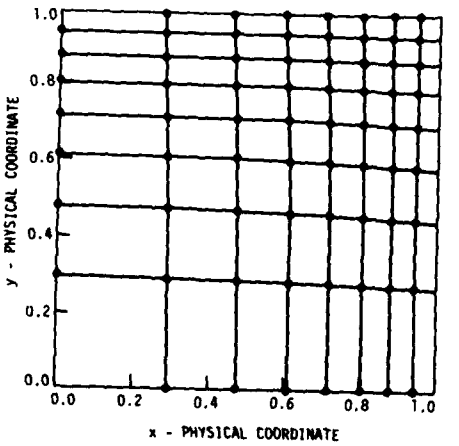


Fig. 5. Converged grid using reflection at the boundaries.

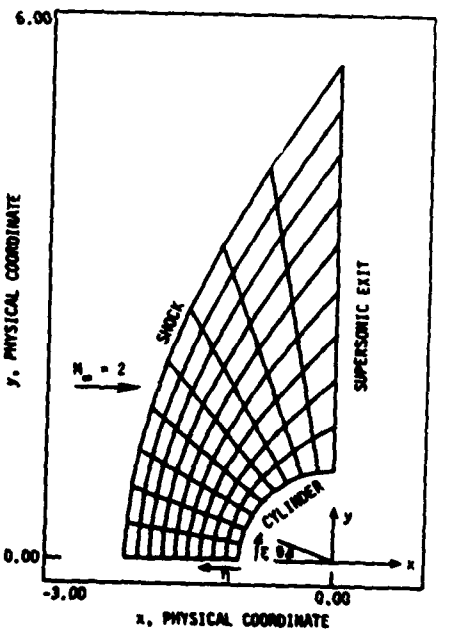


Fig. 6. Fixed grid for the cylinder.

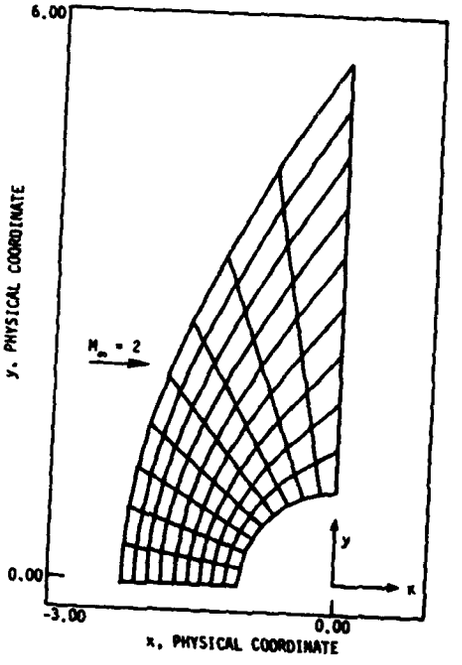


Fig. 7. Converged adaptive grid for the cylinder, $R_{max} = 1.03$.

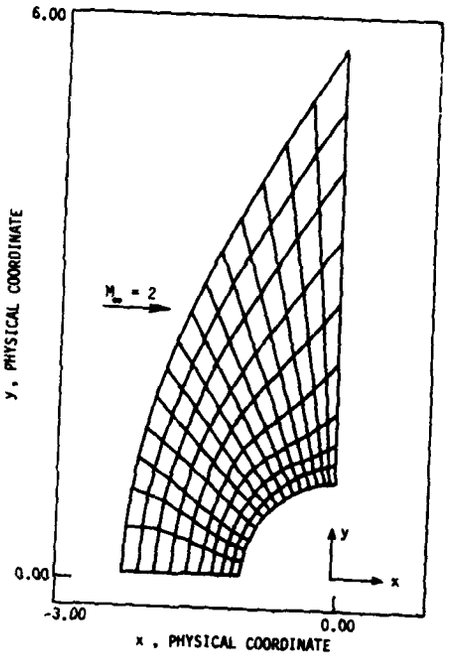


Fig. 8. Converged adaptive grid for cylinder, $K_{max} = 6.0$.

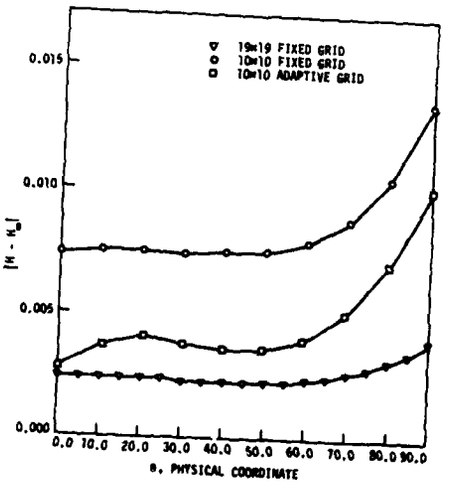


Fig. 9. Comparison of errors in total enthalpy.

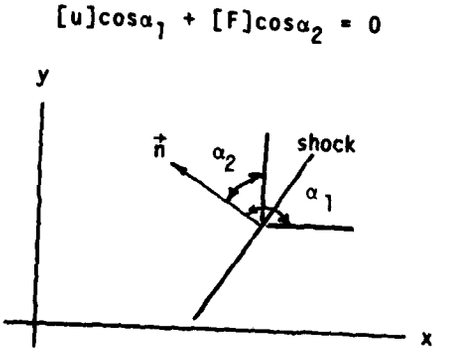


Fig. 10. Shock wave and surface normal orientation.

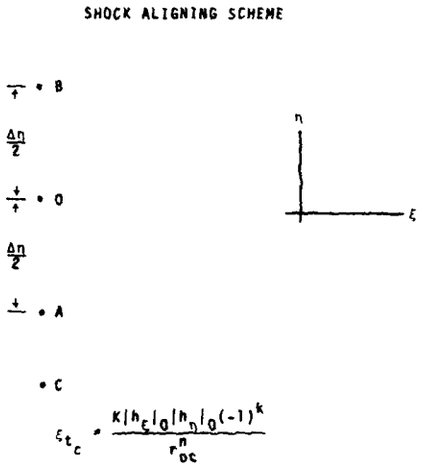


Fig. 11. Grid point speed for shock aligning scheme.

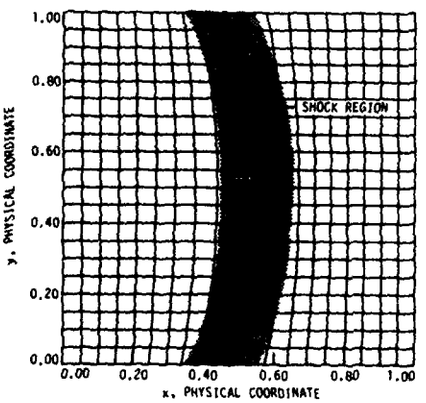


Fig. 13. Converged grid using curved high gradient region.

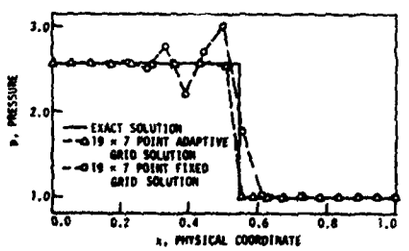


Fig. 16. Pressure comparison at y = 0.208.

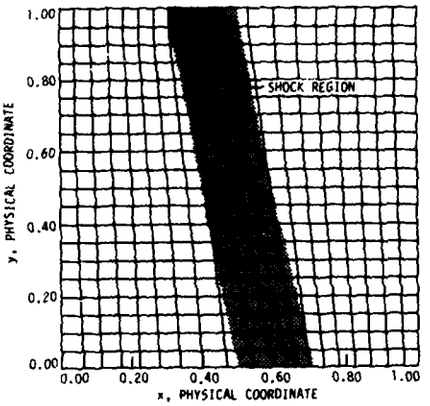


Fig. 12. Converged grid for straight high gradient region.

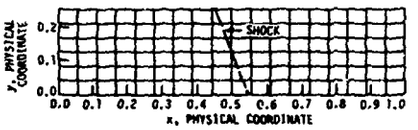


Fig. 14. Fixed grid for the straight oblique shock problem.

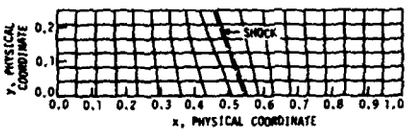


Fig. 15. Converged grid and shock location for the oblique shock problem.

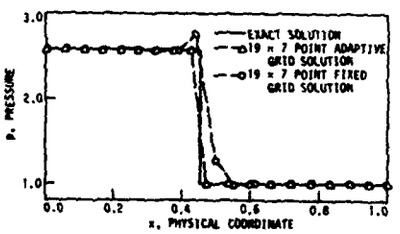


Fig. 17. Pressure comparison at y = 0.0.

AD P000979

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ADAPTIVE GRIDDING FOR FINITE DIFFERENCE SOLUTIONS TO HEAT AND MASS TRANSFER PROBLEMS*

HARRY A. DWYER†, MITCHELL D. SMOOKE‡ AND ROBERT J. KEE‡
†University of California, Mechanical Engineering Department, Davis, CA 95616, USA; ‡Sandia National Laboratories, Applied Mathematics Division, Livermore, CA 94550, USA

INTRODUCTION

Our purpose in writing this paper is to review some of our recent work in the calculation of optimal meshes for the solution of parabolic and elliptic partial differential equations (PDE). We first explain our strategies for the adaptive placement of mesh points. In addition, we make some speculation as to promising avenues for future research in mesh adaptation. Finally, we discuss examples of the application of adaptive gridding to problems of heat and mass transfer. We draw these examples from our work in combustion modeling.

In obtaining numerical solutions to PDEs, the spatial derivatives are often approximated by discrete representations on a mesh network. The accuracy of any numerical solution depends in an important way on the relationship of the location of the mesh points to changes in the dependent variables. Our objective is to investigate finite difference methods in which the mesh networks adapt themselves dynamically to obtain accurate solutions. Such methods represent an important advance in overcoming a major shortcoming of traditional fixed mesh methods which are often unable to resolve accurately steep fronts or sharp peaks.

Our research in adaptive meshing follows two avenues. One is to employ a fixed number of mesh points and to move their location by coordinate transformations. The other is to add or subtract mesh points as needed. In either case the positioning of the mesh depends on one or more important characteristic of the solution. We attempt to equidistribute this characteristic between each mesh interval. For example, equidistribution of the arc-length of the solution has the effect of concentrating mesh points in steep gradients. Taking the coordinate transformation approach, the original equations are recast so that the new independent variable becomes the arc-length. Then, in addition to solving the original equations in the transformed variables, a set of equations is also solved to describe the movement of the original physical coordinates relative to the new transformed variables. When adding and subtracting grid points (the variable node approach) we specify the maximum value of the equidistributed characteristic (say arc-length) allowed over any mesh interval, and continue to add points until this criterion is satisfied. The latter approach is closer to that used in ODE software where as many timesteps as needed are

* Research sponsored by the United States Department of Energy, Office of Basic Energy Sciences.

taken to bring the local truncation error to within prespecified tolerances. However, while this approach may be needed to insure high accuracy for PDEs, it can suffer from limitations of computer storage and time. While we are developing two approaches for adaptive meshing, we believe that the research will ultimately lead to a combination of the methods.

Our variable node method stems largely from our development of methods to solve systems of stiff and unstable nonlinear boundary value problems. Such systems occur frequently in modeling energy systems. Our applications have been principally in combustion chemistry, particularly in the investigation of complex chemical kinetics in premixed flames. Our coordinate transformation work was initially used to track moving flame fronts, and more recently to investigate droplet combustion. In all our work we are concerned with solving simultaneously a relatively large number of PDEs; in the case of the premixed flames, 30 coupled PDEs are typical.

Our work draws on earlier work in both the aeronautical and the boundary value problem literature. From the former we take the ideas of generalized non-orthogonal coordinate transformations and boundary-fitted coordinate systems.[1,2] From the latter we take the ideas of equidistribution of weight functions and error control strategies. Generally speaking, the boundary value problem literature has more theory on which to base methods, but the problems are simpler inasmuch as they are one-dimensional.

BASIC SYSTEM OF EQUATIONS

The solutions to the physical problems which are presented in this paper cover a range of flow and chemical systems. However, in all of the problems there is the common simplification of uncoupling the fluid mechanics from the heat and mass transfer. For some systems, such as steady flame propagation, the simplification is natural to the problem, while in others, it is more artificial. In either case, it does allow for a clear understanding of the problems caused in grid adaptation, when heat and mass transfer as well as chemical reactions are considered.

The system of reaction-diffusion equations which describe the problems in this paper are:

$$\frac{\partial}{\partial t}(\rho Z_m)+\frac{\partial}{\partial x}(\rho u Z_m)+\frac{\partial}{\partial y}(\rho v Z_m)=\frac{\partial}{\partial x}(D_m\frac{\partial Z_m}{\partial x})+\frac{\partial}{\partial y}(D_m\frac{\partial Z_m}{\partial y})+\dot{\omega}_m \qquad (1)$$

where Z is the dependent variable vector (temperature, and species mass fractions), and $\dot{\omega}_m$ is the vector representing the the chemical source terms:

$$Z=(T,Y_1,Y_2,\ldots,Y_K)^t \qquad (2)$$

$$\dot{\omega}=(\dot{\omega}_T,\dot{\omega}_1,\dot{\omega}_2,\ldots\dot{\omega}_K)^t \qquad (3)$$

In these equations the following notation has been employed: ρ - mass density, u - velocity in x-direction, v - velocity in the y-direction and D_m - the diffusive transport coefficient for

the m^{th} equation. The details of the source terms and velocity fields are described when the physical examples and results are presented later in the paper.

For some of the results presented, the above equation set is transformed into generalized non-orthogonal coordinates (τ, ξ and η):

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = \frac{\partial \hat{R}}{\partial \xi} + \frac{\partial \hat{S}}{\partial \eta} + \hat{H} \tag{4}$$

where,

$$\hat{Q} = \frac{Z_m}{J}$$

$$\hat{E} = \frac{Z_m}{J}(\xi_t + \rho u \xi_x + \rho v \xi_y)$$

$$\hat{F} = \frac{Z_m}{J}(\eta_t + \rho u \eta_x + \rho v \eta_y)$$

$$\hat{R} = \frac{D_m}{J}\left(\frac{\partial Z_m}{\partial x}\xi_x + \frac{\partial Z_m}{\partial y}\xi_y\right)$$

$$\hat{S} = \frac{D_m}{J}\left(\frac{\partial Z_m}{\partial x}\eta_x + \frac{\partial Z_m}{\partial y}\eta_y\right)$$

$$\hat{H} = \frac{\dot{\omega}_m}{J}$$

The transformation metrics, or areas and volumes, are given by:

$$J = \frac{1}{x_\xi y_\eta - x_\eta y_\xi}$$

$$\xi_x = J y_\eta, \qquad \xi_y = -J x_\eta, \qquad \xi_t = -x_t \xi_x - y_t \xi_y$$
$$\eta_x = -J y_\xi, \qquad \eta_y = J x_\xi, \qquad \eta_t = -x_t \eta_x - y_t \eta_y$$

We readily see that the resulting equations are more complex; however, with some additional programming a much more valuable tool is obtained. In the above form it is quite easy to implement body-oriented coordinates for arbitrary-shaped bodies, as is often done for flow over airfoils. However, the major advantage of these transformations in our work is the ease with which coordinate adaptation can be utilized.

Even though the governing equations are somewhat simplified compared with the Navier-Stokes equations, they still encompass a large selection of important problems. Moreover, they include a rich and disparate collection of physical time scales. As a result, they are likely to have solutions with regions which need adaptive gridding to be resolved accurately. For example, the effects of the following time scales will be illustrated in the results presented

$\Delta t_U = L/U$, Velocity convection scale

$\Delta t_\nu = L^2/\nu$, Viscous diffusion scale

$\Delta t_\alpha = L^2/\alpha$, Heat conduction scale

$\Delta t_{Dm} = L^2/D_m$, Mass diffusion scale

$\Delta t_{\dot{\omega}_m}$ = Reaction rate scale

As these scales become disparate (depending on the particular problem), steep gradients in space and time develop. Without adaptation, the numerical integration methods can be pushed to dramatic failure. It is our purpose to present methods which resolve these scales in space and time in an efficient and accurate manner.

ADAPTIVE GRID METHODS

We consider both steady and transient problems. The steady problems are elliptic boundary value problems, while the transient problems are parabolic initial-boundary value problems. At each time step of a transient problem, however, an elliptic boundary value problem must be solved. Therefore, our meshing strategies share the same essential features regardless if the application is steady or time-dependent.

During the last fifteen years many varied methods have been developed to attempt to choose optimal grid spacings on which to solve two-point boundary value problems. When these problems are solved using an initial value method (such as shooting), the adaptive meshing is done automatically and accurately. Variable-step initial value problem software is used to adjust the step size as the integration proceeds in order to control the local truncation error. Unfortunately, many problems in combustion are unstable to initial value methods,[3] and therefore global solution methods must be used.

Kautsky and Nichols[4] point out that many of the adaptive mesh selection procedures used for global solution methods can be interpreted as equidistributing a positive weight function. On the interval [0,L], one attempts to determine a mesh M

$$M = (0 = x_1 < x_2 < \ldots < x_M = L)$$

such that the weight function achieves a given constant value over each subinterval. Among the various approaches developed, White[5] has discussed equidistribution of the arc-length of the solution; Pereyra and Sewell[6] have equidistributed the local truncation error and Smooke[3] has chosen to equidistribute both the change in the discrete solution and its gradient. Other methods for choosing appropriate meshes for two-point boundary value problems have been investigated, for example, by Russell and Christiansen,[7] Ablow and Schecter,[8] de Rivas,[9] and Denny and Landis.[10]

Positive Weight Function Concept

Following the notation of Kautsky and Nichols, we say that the mesh M is equidistributed on [0,L] with respect to the non-negative weight function f and the constant W if

$$\int_{x_j}^{x_{j+1}} f \, dx = W, \quad j = 1, 2, \ldots, M-1 \tag{5}$$

Similarly $\mathcal{M}$ is called sub-equidistributing on [0,L] with respect to f and W if

$$\int_{z_j}^{z_{j+1}} f\, dz \leq W, \quad j = 1, 2, \ldots, M-1 \tag{6}$$

Strategies for determining an optimal mesh for two-point boundary value problems can be implemented either implicitly or explicitly. In the implicit approach, the weight function depends directly upon the solution. As a result, the original boundary value problem is converted into an augmented system in which the dependent variables and the mesh are computed simultaneously. In the explicit approach, the weight function does not depend upon the current solution. Instead, it depends upon a previously calculated solution. For both linear and nonlinear boundary value problems, the implicit approach requires that one solve a nonlinear two-point boundary value problem. Thus implicit equidistribution techniques do not preserve the linear-nonlinear character of the original problem. Moreover, even for nonlinear problems the augmented system is usually more difficult to solve than the original problem. Explicit equidistribution techniques, on the other hand, preserve the linear-nonlinear character of the original two-point boundary value problem.

Our experience has led us to consider explicit equidistribution methods. We have found that as the number of dependent variables increases, or the problem becomes more nonlinear, the selection of a mesh by equidistributing an implicit weight function is less practical than by equidistributing a weight function based on the solution from a previous grid. The approach we have chosen to determine an adaptive grid for premixed flame problems is similar to the method used by Pearson[11] in solving scalar boundary layer problems. We attempt to equidistribute the difference in the components of the discrete solution and the difference in the gradient of the components of the discrete solution between adjacent mesh points. That is we seek to obtain a mesh $\mathcal{M}$ such that

$$\int_{z_j}^{z_{j+1}} \left|\frac{dZ_i}{dz}\right| dz \leq \delta \left(\max_{0 \leq s \leq L} |Z_i| \right) \quad \begin{matrix} j = 1, 2, \ldots, M-1 \\ i = 1, 2, \ldots, K+1 \end{matrix}, \tag{7}$$

and

$$\int_{z_j}^{z_{j+1}} \left|\frac{d^2Z_i}{dz^2}\right| dz \leq \gamma \left(\max_{0 \leq s \leq L} \left|\frac{dZ_i}{dz}\right| \right) \quad \begin{matrix} j = 1, 2, \ldots, M-1 \\ i = 1, 2, \ldots, K+1 \end{matrix}, \tag{8}$$

where Z is the dependent variable vector, δ and γ are small numbers less than one and the values of $\max|Z_i|$ and $\max|dZ_i/dz|$ are obtained from a converged numerical solution on a previously determined mesh.

A potential disadvantage of the method described so far is that the mesh may not be smoothly varying. For example, the ratio of consecutive mesh intervals may differ by several orders of magnitude. This can adversly affect the accuracy of the solution as well as the convergence properties of the method. As a result, we impose the added constraint that the mesh be locally bounded, i.e., the ratio of adjacent mesh intervals must be bounded above and

below by constants. We require that

$$\frac{1}{C} \leq \frac{h_j}{h_{j-1}} \leq C, \quad j = 2, 3, \ldots, M \tag{9}$$

where $h_j = x_j - x_{j-1}$ and C is a constant less than one. Such a "buffering" of the mesh tends to smooth out rapid changes in the size of the mesh intervals.

We note here an analogy to the approach followed for time step selection in predictor-corrector ODE software. In these codes some estimate of the local truncation error is made. One way to measure the error is by comparing the difference between a certain order predictor and the same order corrector. Their difference is related to the local truncation error incurred in taking a time step. The time step is then adjusted such that this error is below a prespecified level. Possible ways to measure truncation error are to use different differencing formulas, or to use the same difference formulas but on different meshes.

Certainly the equidistribution and control of local truncation error is the most conservative and accurate approach to mesh adaptation. However, in many cases it may be more costly than necessary. For instance, if only integrated properties of the solution are of interest (e.g. flame speed or surface drag) then perhaps less attention need be paid to truncation error everywhere in the flow field. For problems with strong nonlinearities it may even be preferable to equidistribute something related to the local truncation error rather than the error itself. In particular, we have seen that weight functions based on higher derivatives (to more closely match truncation error) have led to instabilities. Moreover, if the differencing scheme is first order then the local truncation error is proportional to second derivatives of the solution. Thus, the weight function in Eq. (8) is proportional to the local truncation error. As a result we believe that weight functions similar to those in Eqs. (7), (8) and (10), are perfectly adaquate for many problems.

Steady-State Problems, Variable Node Method

After discretization, the governing equations form a nonlinear system of algebraic equations. We solve this system by a damped modified Newton method.[12] First the equations are solved on a uniformly spaced coarse mesh (3-5 subintervals). The values of $\max|Z_i|$ and $\max|dZ_i/dx|$ are then evaluated. We next test the inequalities in Eq. (7) and Eq. (8) for each of the $K+1$ components of Z at each node of the coarse mesh. If either of the inequalities is not satisfied, a grid point is inserted at the midpoint of the interval in question. Once a new mesh has been obtained, we check to see whether it is locally bounded. If it is not, a grid point is inserted at the midpoint of the intervals in which the inequality in Eq. (9) is not satisfied. The previously converged numerical solution is interpolated onto this new mesh and the result serves as an initial solution estimate for Newton's method on this finer grid. The governing equations are solved on the new mesh and the process continues until the inequalities in Eqs.

(7), (8) and (9) are satisfied. Note that if we had refined the mesh by using only the inequality in Eq. (7), we would resolve high gradient regions but would have difficulty resolving regions of high curvature (for example the local maxima of sharp peaks in the solution).

Most of the ideas discussed above can be extended to the solution of multi-dimensional elliptic boundary value problems. We are just beginning to explore methods to obtain optimal grids for two-dimensional nonlinear elliptic boundary value problems. In our initial attempts we have made the logical extension of the one-dimensional ideas. That is, we start on a coarse two-dimensional grid, and add grid pionts according to Eqs. (7), (8) and (9) in both the x and y directions. In two-dimensions the Jacobian of the system is block penta-diagonal and we solve the system by block line SOR iteration. If fronts in the solution align reasonably well with one of the coordinates then the method is efficient. Of course if a front crosses the mesh on a bias then a fine mesh results everywhere and the direct extension of the one-dimensional method is not really useful. In these cases either a coordinate rotation or a local mesh refinement[13] must be employed.

Time-Dependent Problems

The ideas used in solving one-dimensional steady-state problems are readily adapted for time-dependent mixed initial-boundary value problems. In particular, by considering the solution of a time dependent problem as the solution of an inhomogeneous two-point boundary value problem at each time level, the methods developed in the realm of steady-state problems can be applied in a time-dependent setting. Both the implicit and the explicit equidistribution techniques have natural time-dependent analogues. In the case of the implicit methods, the original equations are recast so that in addition to solving the original equations in the transformed variables a set of equations is also solved to describe the movement of the original physical coordinates relative to the new transformed variables. In general, one can expect the difficulty with specifying an initial solution estimate for the dependent solution components and the grid points to be less severe in the time-dependent setting than in the steady-state one since the previous time level solution is often an excellent starting estimate. The explicit equidistribution techniques can be used in a time-dependent setting by explicitly moving the grid based upon solutions at previous time levels.

Coordinate Transformation Methods

Our coordinate transformation technique has been tested extensively by Dwyer, et. al. for one-dimensional problems, and more recently, it has also performed quite well in two dimensions.[14] We note, however, that so far we have not implemented a general two-dimensional adaptation procedure. Instead, we take advantage of some *a priori* physical knowledge of the

solution to extend the one-dimensional method. Although the method is not yet fully adaptive in two-dimensions, we believe that the solution to many important problems can benefit from its use. Moreover, we believe that generalization of the method will follow from current work. The solution technique in the two-dimensional problems is a non-iterative block-tridiagonal ADI method[15,16] in which the Jacobians are evaluated analytically.

In this method the lines of constant η are fixed (forming arcs in space), and the adaptation is done along these arcs. In effect, the method is quasi-one-dimensional, and it relies on the modeler having sufficient qualitative knowledge about the solution to be able to fix a set of coordinates which are roughly normal to any steep fronts in the solution. We typically take the weight function (or transformation) for adaptation along the fixed arcs to be given by:

$$\xi(x,y,t) = \frac{\int_0^S (1 + b|\partial T/\partial S|)dS}{\int_0^{S_{\max}} (1 + b|\partial T/\partial S|)dS} \tag{10}$$

where S is the length along the fixed arc, and b is an adjustable constant used for "optimization" of the grid distribution. For the case $b = 0$ a uniform distribution of points along the fixed arc results. For large values of b, the mesh intervals are determined so that the same change in the dependent variable T occurs between mesh points. A typical value of b is 1/3. The coefficient b can be thought of in terms of the "buffering" concept introduced earlier. That is, b is chosen so that not all the mesh points are concentrated in the front region. Some are in regions of relatively uniform T, and there is a smooth progression of mesh interval sizes in moving away from a front.

The weight function is evaluated explicitly, and the mesh transformation is held fixed throughout the time step. In some cases, however, we have used a prediction of the solution at $t + \Delta t$ instead of the solution at t to form the basis of the transformation. In all cases the integrals in Eq. (10) are evaluated using the trapezoidal rule. If Δt is large enough for the front to move out of the fine-mesh region, then implicit or iterative coordinate generation schemes would have to be considered. However, this was never the case in our problems, since the fast chemical reactions prohibit the taking of large time steps. Also, the buffering effect of the b parameter causes there to be adaquate resolution even if the front does move away from its optimal location.

Linear Algebra Considerations

We take as an assumption that for problems of interest (in combustion) the system of equations is stiff—they are characterized by widely disparate time and length scales. This fact leads us to consider only implicit solution procedures.[17] A salient characteristic of implicit methods is that they require the simultaneous solution of nonlinear equations at each time step (or iteration). For multi-dimensional problems or problems involving many dependent

variables (e.g. species concentrations), these solutions lead to a need to form and solve systems represented by large matrices. Therefore, the way in which this task is accomplished has a major bearing on the structure of the computer codes which solve the systems.

Our approach currently for one-dimensional problems is to employ a modified Newton method. The block tri-diagonal Jacobian matrix is formed numerically using finite differences and its LU decomposition is computed immediately. The LU decomposition factors occupy the same storage locations as the Jacobian did originally. The same decomposed matrix is then used for several iterations (or time steps in transient problems). As long as the iterations are convergent, a sizable cost savings is realized by not re-evaluating and factoring the Jacobian. This approach is commonly used for solving systems of stiff nonlinear ODEs.

In two-dimensional problems we take two approaches. For the fixed-number-of-grids coordinate transformation problems we employ a standard alternating direction implicit (ADI) method. Here the block tri-diagonal Jacobian is formed and LU decomposed, and the linear system is solved along each row and column of the mesh at each time step. No iteration is done. Justification for the approach follows the well-known arguments that the error incurred by the ADI splitting is of the same order as the truncation error already incurred by the discretization of the time derivative.[15,16]

We take a different approach in solving the nonlinear equations in the variable node formulation. Here the full Jacobian, a block five-diagonal matrix, is formed at once. A modified Newton method is used to solve the nonlinear system. At each stage of the Newton iteration an iterative block-line-SOR method is employed to solve the linear system. The LU factors are stored and re-used for successive iterations. However, after the solution is completed on a given mesh and new mesh points are added as needed, a new Jacobian must be computed on the new mesh.

We expect that significant computational gains will result from research on and development of incomplete Jacobian factorizations or matrix splittings. The objective here is to avoid solving the original equations directly, and instead to solve a related, and approximately equivalent, system that is much easier to solve. The best known example of such splitting is the ADI method, which can be thought of as an incomplete factorization of the full Jacobian. Even though the factorization is incomplete, the error which it introduces is of the same order as that introduced by the time discretization. Therefore, the approximation does not degrade the accuracy of the solution but it increases significantly the efficiency of the computation.

The ADI factorization is only one of a large family of related splittings which can take advantage of some particular characteristic of a problem. For example, it is often the case in systems of PDEs that some of the equations are weakly coupled to the others. In such cases, solving the equations sequentially (instead of fully coupled) is known to result in significant savings. Instead of solving systems of block tri-diagonal equations, one is able to solve a sequence of scalar tri-diagonal equations with far fewer operations required. Similarly, in some

combustion problems, considerable savings are realized through operator splitting algorithms in which the chemical rate terms are handled separately from the transport terms. These methods are equivalent to matrix splittings of the system's Jacobian. However, in both cases, application of the procedures has been ad-hoc, i.e. with little theory to help determine the rate of convergence, or whether the process converges or not. By studying pre-conditionings and incomplete factorizations of the Jacobians, rather than ad-hoc splittings of the equations, such methods can be put on a firmer theoretical footing and thus more reliable and effective PDE methods should result.

The cost of evaluating the Jacobian is usually very high in our problems (up to 95% of the computer time in some flame problems). Therefore, it is natural to seek methods which require as few Jacobian evaluations as possible. Based on the success of the modified Newton method,[18] where we have applied it, and its success in the ODE software, we expect that similar approaches will ultimately find wider application in the solution of PDEs. The dilemma is that in order to use a modified Newton method, the full Jacobian must be stored. For multi-dimensional problems this storage requirement is usually too large for the memory of any computer in use today. Therefore, effective use of a modified Newton method requires development of algorithms which quickly move Jacobian information between computer memory and peripheral storage. We note here also that some splitting methods, as discussed above, lead to fewer function evaluations to complete a Jacobian evaluation.

EXAMPLE PROBLEMS

Steady Premixed Flames

The method we have implemented in the calculation of premixed flame structure equidistributes the difference in the components of the solution and its gradient between consecutive grid points. To illustrate the importance of adaptively placing grid points in the flame zone to the accuracy and efficiency of the flame calculation, we have performed several calculations for an acetylene-oxygen flame using equi-spaced and adaptively placed grids. (For these problems a system of 21 species and 72 reactions was used.) Figure 1 shows the molecular hydrogen profiles for a series of calculations using 20, 40, 80, and 160 equi-spaced points. We include the experimental data for reference. We secure not only a much smoother solution but one which agrees better with the experimental data as a finer and finer grid is used.

Figure 2 shows the molecular hydrogen profile for the same flame but solved using adaptive meshing. In this case 41 adaptively placed points are used to obtain three significant figures of accuracy in the solution. As expected, the adaptive calculation secures a highly resolved species profile with far fewer points than are required using the equi-spaced grid.

In the adaptive calciation, 19 of the 41 grid points are located in the "flame zone," or region of fast chemical reaction. Note that a relatively large region of the computation has

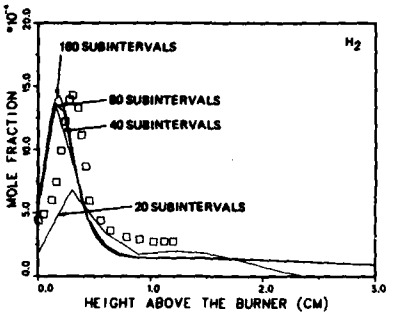


Figure 1. Hydrogen mole fraction distribution profiles in an acetylene-oxygen flame, computed on various equi-spaced grids. Boxes are experimental data of Eberius, Hoyermann and Wagner.

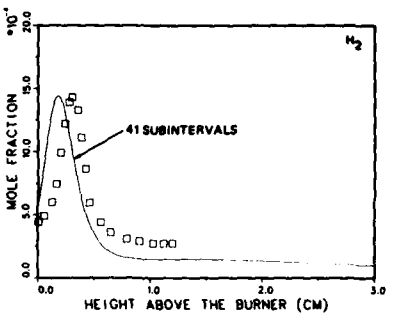


Figure 2. Hydrogen mole fraction distribution profiles in an acetylene-oxygen flame, computed on 41 adaptively placed grids. Boxes are experimental data of Eberius, Hoyermann and Wagner.

relatively little chemical reaction. Either the temperature is too low, or the fuel is almost all consumed. The smallest mesh interval is 640 times smaller than the total interval of the problem. So to obtain the same resolution over 600 equi-spaced grids would be required. The adaptive calculation took 275 seconds of CPU time on a CRAY-1S computer. The equi-spaced calculation with 160 subintervals took 585 seconds of CPU time.

In the next example we compare the effects of adaptive and equi-spaced grids in the prediction of flame speeds in a one-atmosphere, stoichiometric, hydrogen-air flame.[19] The accurate placement of grid points in regions where the solution varies rapidly leads to a significant reduction in the number of subintervals needed to obtain accurate flame speeds. As a result, the overall cost of a flame speed computation can be substantially reduced. In the first set of calculatons we determined flame speeds on grids consistng of 20, 40, 80, 160, 320, and 640 equi-spaced points. The results of the calculatons are listed in Table I.

The second set of calculations was performed using the adaptive grid procedure. In this case we used grids of 20, 30, 40, 50, and 60 adaptively placed points. The results are listed in Table II.

Several points merit further discussion. First, for both the equi-spaced and adaptively placed grids, we see that as the number of mesh intervals increases, the flame speeds decrease. Second, the sequence of flame velocities obtained in the adaptive calculations approach a limiting value with only 40 to 50 grid points, while flame velocities obtained in the equi-spaced calculations are still changing by almost 15 percent as we go from 80 to 160 grid points. In fact, it was not until 640 equi-spaced points were used that the flame speed was within 2 percent of the result calculated on the 50 point adaptive grid. Like the previous example, the ratio of

TABLE I

HYDROGEN-AIR FLAME SPEEDS, EQUI-SPACED GRIDS (cm/sec)

No. of Points	20	40	80	160	320	640
Flame Speed	445	289	244	211	193	184

TABLE II.

HYDROGEN-AIR FLAME SPEEDS, ADAPTIVE GRIDS (cm/sec)

No. of Points	20	30	40	50	60
Flame Speed	248	212	185	181	181

minimum mesh size to the domain of integration was 625. Also, as expected, the adaptive grid computation is less expensive. The 50 point adaptive calclation took 45 seconds of CPU time while the 640 point equi-spaced calculation took 327 seconds. A savings of about a factor of seven resulted in going from equi-spaced to adaptive grids.

Two-Dimensional Elliptic Boundary Value Problem

We demonstrate here our two-dimensional extension of the variable node method. The equation we have chosen is the nonlinear Poisson equation on the unit square:

$$\frac{\partial^2 Z}{\partial x^2} + \frac{\partial^2 Z}{\partial y^2} + Z^2 = f(x,y)$$

$$Z = g(x,y) \text{ on the boundary}$$

We have chosen $f(x,y)$ and $g(x,y)$ so that the solution is $Z = \exp -30(x^2+y^2)$. The initial equi-spaced grid was 2×2. After five mesh refinements the nonuniform 18×18 mesh shown in Fig. 3 evolved. Note the high resolution of the solution in the regions of high slope and curvature.

Unsteady Two-Dimensional Flame Propagation, Coordinate Transformation Method

In this section we demonstrate coordinate transformation adaptive grid techniques by disccussing several examples. First we present solutions for unsteady flame propagation about spherical particles. In these examples the time scales for convection and reaction are small

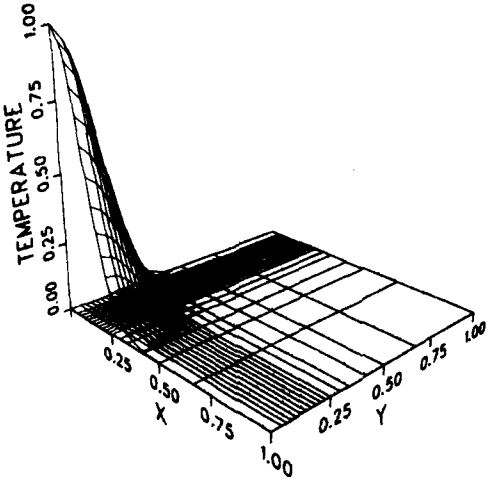


Figure 3. Solution to the elliptic test problem with a two-dimensional variable node adaptive grid.

compared to conduction and diffusion, or

$$\Delta t_U \text{ and } \Delta t_{\dot{\omega}_m} \ll \Delta t_\alpha \text{ and } \Delta t_{D_m}$$

A one-step chemical reaction is used and the vector of dependent variables and rate terms are

$$Z_m = (T, \rho_A)$$

$$\dot{\omega}_m = \left(\rho_A \frac{\Delta t_\alpha}{\Delta t_{\dot{\omega}_A}} \exp(-\theta_A/T), -\rho_A \frac{\Delta t_\alpha}{\Delta t_{\dot{\omega}_A}} \exp(-\theta_A/T)\right)$$

where T, ρ_A and θ_A, the nondimensional temperature, premixed fuel concentration and activation energy, have been normalized by reference values.[14] The calculation is simplified so that the overall density remains constant and thus the flow field is independent of the combustion process. The velocity field is given as a low Reynolds number Stokes flow.

The results of an interesting calculation are shown in Figs. 4 through 9. The following ratios of time scales are used:

$$\frac{\Delta t_\alpha}{\Delta t_\nu} = \frac{\Delta t_{D_A}}{\Delta t_\nu} = Pe \text{ (Peclet Number)} = 200$$

$$\frac{\Delta t_\alpha}{\Delta t_{\dot{\omega}_A}} = 2.2 \times 10^5$$

Figures 4 and 5 illustrate unsteady flame propagation after surface ignition, when using a uniform grid. (These figures are divided into two parts, the top shows the coordinate system

and the bottom plots the isotherms. There are ten normalized isotherms plotted which range in values between 0.2 and 1.2) The figures show that the grid is uniform and the isotherm distribution exhibits significant oscillation. Note that the oscillation in the isotherms becomes larger in amplitude as the flame moves into the large cell regions away from the body. These oscillations are a result of the large cell Peclet number and the central difference approximation for the spatial derivatives.[20] The cell Peclet number is large because of the increasing velocity and cell size as the grid moves away from the body. If we had used windward differences, the numerical viscosity would have increased significantly, and thus introduce significant errors such as an the increase in flame thickness. Use of a refined uniform grid is unreasonable because of the additional computational requirements of time and storage.

Now consider the problem using an adaptive grid as shown in Figs. 6 and 7. These figures show the coordinate and isotherm distributions for the same times as shown in Figs. 4 and 5. Notice that the flame has a new and more accurate velocity and position and that there are no oscillations. By resolving the flame, the cell Peclet number is reduced to values less than one. This guarantees that the solution will be oscillation free. To illustrate our point further, the radial temperature distributions at similar angular positions are shown in Fig. 8 for both the uniform and adaptive grid solution. The oscillations in the uniform grid solution are quite

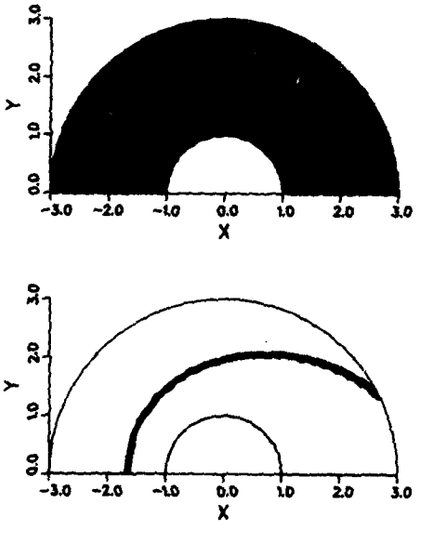


Figure 4. Coordinate system and isotherm distribution about a burning particle with a uniform grid, early time.

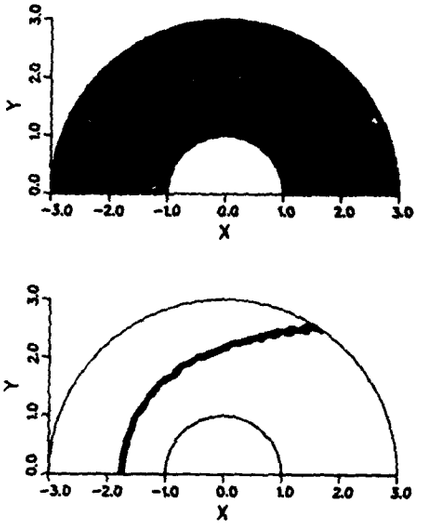
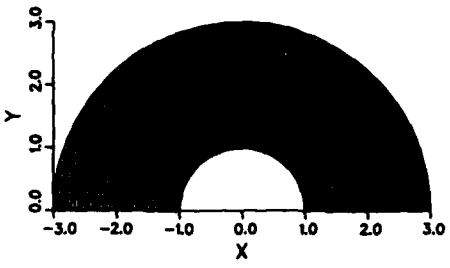


Figure 5. Coordinate system and isotherm distribution about a burning particle with a uniform grid, later time.



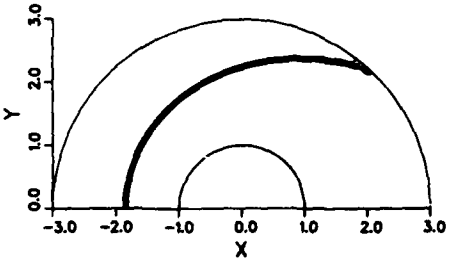
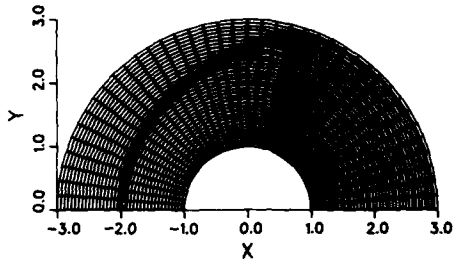


Figure 6. Coordinate system and isotherm distribution about a burning particle with a coordinate transformation adaptive grid, early time.



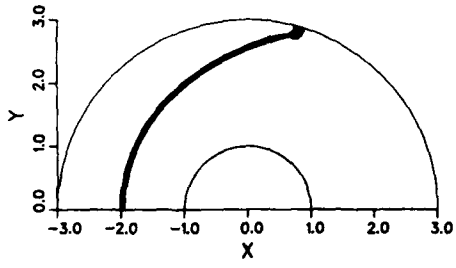


Figure 7. Coordinate system and isotherm distribution about a burning particle with a coordinate transformation adaptive grid, later time.

apparent. Also, it should be mentioned that the uniform grid solution terminated at the next time step because of negative temperatures caused by the oscillations.

With the same number of grid points we have been able to convert an unusable calculation to an efficient and accurate one. However, we have introduced some new, but minor, problems with the remedy. One of these problems is caused when the thin flame passes out of the boundaries of the system and there are no longer any gradients along some of the fixed arcs. The grid then reverts back to a uniform grid over one time step. In the present calculation this does not cause a problem because the dependent variable is uniform and the rapid change in metrics is unimportant because the solution isn't changing. However, if another variable such as velocity was being calculated in this region it would be extremely difficult to obtain an accurate solution for that variable. In this case the other variables (besides temperature) should be considered in the formation of the weight function and the grid transformation. A possible solution to this problem is shown in Fig. 9 where the grid distribution has been frozen at the value it had when the flame left the computational region. With this strategy the metrics are smooth but the mesh is wastefully fine near the outer boundary.

Another potential problem exists when different regions of high gradient exist within the same problem. This is particularly troublesome when the regions have incompatible

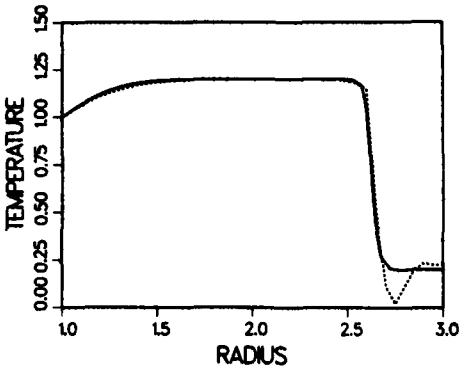


Figure 8. Typical temperature distribution with uniform and adaptive grids. Dashed line is on uniform grid, and shows oscillations due to high cell Peclet number.

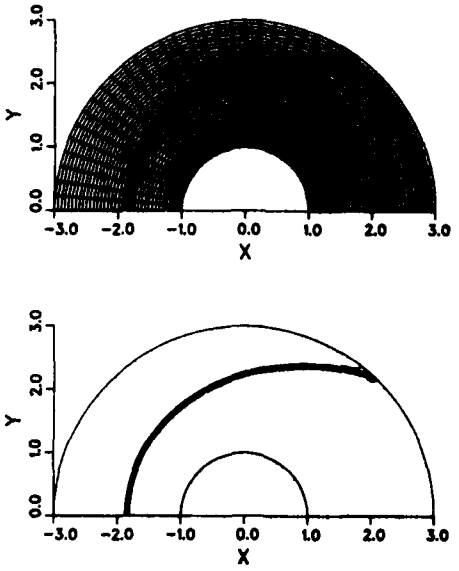


Figure 9. Coordinate system and isotherm distribution about a burning particle with a coordinate transformation adaptive grid. Coordinate system frozen at outer boundary.

geometries for grid stretching, as is the case in our next example. Here a flame surrounds a burning spherical particle over which the flow (Reynolds number of 100) has separated. In this calculation both the flow field in separation region and the temperature gradients in the flame must be resolved, and the boundary conditions must be applied far from the body. The coordinate system used for the flame is not well suited for the flow, and we have taken the approach of using two different coordinate systems and interpolating between them. Figure 10 shows the vorticity pattern together with the grid used to compute the flow field. The temperature distribution and its grid are shown in Fig. 11. Certainly the use of two coordinate systems increases storage and computation time, but the one order of magnitude improvement of grid resolution achieved by the adaptive gridding method, more than makes up for the additional effort. However, it is easily seen that this approach to grid adaptation introduces many new problems, which should prove fertile ground for new solution procedures.

CONCLUSIONS

We believe that we have achieved considerable success in applying adaptive grid methods to solve a variety of problems, but it is also true that the results are not complete. We have

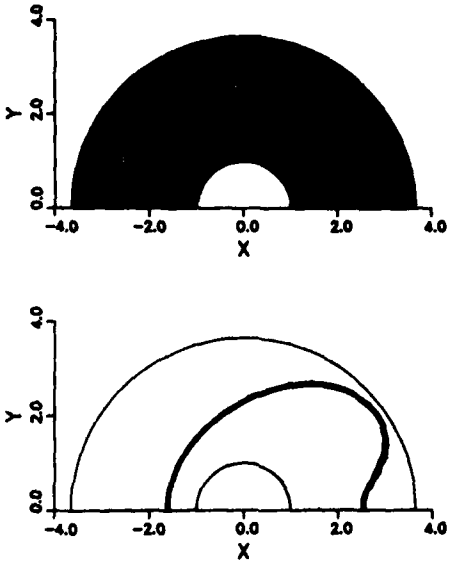


Figure 10. Coordinate system and isotherm distribution about a burning particle with separated flow, using a coordinate transformation adaptive grid.

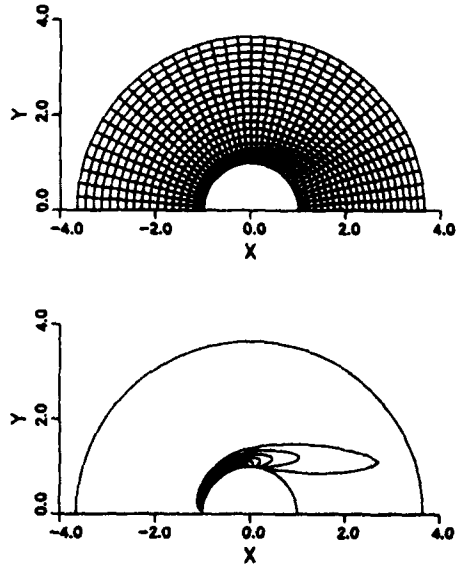


Figure 11. Coordinate system and vorticity distribution about a burning particle with separated flow, using a coordinate transformation adaptive grid.

seen clearly that adaptive gridding is necessary for a wide range of heat and mass transfer applications. Our examples demonstrated a strong dependence of flame shape, flame speed, and numerical stability on the mesh spacing. In regions such as flame fronts, the grid has to be so fine that uniform meshing is completely impractical. However, using the adaptive approach, we have kept cell Reynolds and Peclet numbers less than one with relatively few grid points!

We discussed the concept of equidistribution of a positive weight function and we regard it as a useful framework from which to develop adaptive grid methods. The variable node approach is analogous to ODE initial value problem software, and, because it attempts to bring the weight function within pre-specified bounds, it is potentially the most accurate approach to adaptive meshing. However, due to the number of grid points which may be needed, it is often inefficient in computer time and storage. On the other hand, using generalized coordinates and adaptive gridding through coordinate transformations allows for good resolution of body shapes and flame structure in separated flows at moderate Reynolds Number. In this case, however, the weight function is equidistributed, but not driven below a prespecified bound. We have successfully applied both the coordinate transformation approach and the variable node approach in one- and two-dimensions. Full two-dimensional generalisations are yet to come.

Although adaptive gridding is required for accurate resolution in many problems its use

can introduce new problems. For example, we see problems caused when high gradient regions intersect boundaries or leave the computational zone by convective processes. Also, when more than one physical variable causes scaling problems, such as in flame propagation and flow separation, it may be difficult to use one grid system for the entire problem. Instead, it may be avantageous to use more than one adaptive coordinate system simultaneously. So far the "fix" to many of these problems has been problem dependent. Generalizations are needed.

REFERENCES

1. Thompson, J. F., Thames, F. C. and Mastin C. M. (1974) *J. Comp. Phys.*, **15**, pp. 299-319.
2. Steger, J. L. (1978) *AIAA J.*, **16**, pp. 679-686.
3. Smooke, M. D. (1982) "Solution of Burner-Stabilized Pre-Mixed Laminar Flame Problems by Boundary Value Methods", Sandia National Laboratories Report, SAND81-8040.
4. Kautsky, J. and Nichols, N. K. (1979) "Equidistributing Meshes with Constraints," Stanford University Report, STAN-CS-79-766.
5. White, A. B. (1979) *SIAM J. Numer. Anal.*, **16**, pp. 472-502.
6. Pereyra, V. and Sewell, E. G. (1975) it Num. Math.,**23**, pp. 261-268.
7. Russell, R. D. and Christiansen, J. (1978) it SIAM J. Numer. Anal., **15**, pp. 59-80.
8. Ablow, C. M. and Schecter, S. (1978) *J. Comp. Phys.*,**27**, pp.351-362.
9. deRivas, E. K. (1972) *J. Comp. Phys*,**10**, pp. 202-210.
10. Denny, V. E. and Landis, R. B. (1972) *J. Comp. Phys*,**9**, pp. 120-137.
11. Pearson, C. E. (1968) *J. Math. Phys.*,**47**, pp. 134-154.
12. Smooke, M. D., (1982) "On the Use of a Modified Newton Method for the Solution of Boundary Value Problems", in preparation.
13. Berger, M., Gropp, W. D. and Oliger, J. (1981) "Grid Generation for Time Dependent Problems," NASA Conference Publication 2611.
14. Dwyer, H. A., Kee, R. J. and Sanders, B. R. (1980) *AIAA J.*,**18**, (10), pp 205-212.
15. Beam, R. M. and Warming, R. F. (1978) *AIAA J.*,**16**, pp. 393-402.
16. Briley, W. R. and McDonald, H. (1977) *J. Comp. Phys.*, **24** (4), pp. 372-397.
17. Kee, R. J. and Dwyer, H. A. (1982) *Progress in Astronautics and Aeronautics*,**76**, pp. 485-500, AIAA, New York.
18. Smooke, M. D., Miller, J. A. and Kee, R. J. (1982) "Numerical Solution of Burner-Stabilized Laminar Pre-Mixed Flame Problems, by an Efficient Boundary Value Method," to appear Proceedings of the GAMM-Conference on Pre-mixed Laminar Flames, Aachen, Germany.
19. Smooke, M. D., Miller, J. A. and Kee, R. J. (1982) "On the Use of Adaptive Grids in Numerically Calculating Adiabatic Flame Speeds," to appear, Proceedings of the GAMM-Conference on Pre-mixed Laminar Flames, Aachen, Germany.
20. Roache, P. (1971) "Computational Fluid Dynamics," Hermosa Publishers, Albuquerque, NM.

AD P000980

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

APPLICATION OF CURVILINEAR COORDINATE GENERATION TECHNIQUES TO THE COMPUTATION OF INTERNAL FLOWS

DOYLE D. KNIGHT
Department of Mechanical and Aerospace Engineering
Rutgers University
New Brunswick, New Jersey 08903

INTRODUCTION

In recent years, considerable effort has been focused on the development of techniques for the generation of curvilinear coordinates to facilitate computation of fluid flows in a variety of complex configurations. ~~The present~~ paper focuses on the application of such techniques to the computation of internal flows. ~~The paper~~ It is divided into three major sections. First, a brief review of applications of grid generation techniques in internal flows is presented. Due to the impending publication of a major review article[1] on grid generation covering up to mid-1981, the present discussion focuses on a number of recent publications. Second, a simple method for generating orthogonal or nearly orthogonal curvilinear grids with controlled mesh spacing is presented. Third, a current research problem in generation of three-dimensional internal flow grids is discussed.

Characteristics of Internal Flows

Prior to a discussion of grid generation techniques, it is instructive to broadly characterize internal fluid flow problems. First, internal flows are characterized by lateral physical boundaries which are contained within a finite region. An obvious example is the simple rectangular-to-round diffuser shown in Figure 1, for which the lateral boundaries are solid walls. The cascade in Figure 1 is another example, for which periodic boundary conditions are applied on a portion of the lateral boundaries of a chosen computational domain.[2] Second, the computational domain may be singly- or multiply-connected. The diffuser is an example of a singly-connected flow. The shell-and-tube heat exchanger of Figure 2 illustrates a complex, three-dimensional, multiply-connected domain. Third, internal flows are often characterized by high Reynolds number turbulent flow, thereby requiring refined grid resolution near solid boundaries.

Desirable Characteristics of Curvilinear Coordinates

Although the requirements on curvilinear coordinates are dependent to a certain extent upon the particular physical application, a number of desirable attributes can be enunciated. First, the coordinate transformation should be boundary-oriented, i.e., the boundaries of the physical domain should coincide with a portion (or all) of a curvilinear coordinate line or surface. This concept is illustrated in Figure 3, where upstream and downstream boundaries Γ_1 and Γ_2, respectively, coincide with lines of constant ξ (i.e., segments γ_1 and γ_2) in the transformed plane, and lateral boundaries Γ_3 and Γ_4 coincide with lines of constant η (i.e., segments γ_3 and γ_4) in the transformed plane. This simplifies the coding of the fluid dynamic algorithm and application of the boundary conditions. Although non-boundary conforming coordinate systems (typically cartesian) are currently employed,[3,4] their application requires special treatment of the fluid dynamic algorithm near the boundaries. A slightly different treatment is illustrated in Figure 4, where a "C"-type grid for a cascade flow is shown. A branch cut AB is introduced, and the curve ABCD, which includes the airfoil surface, is mapped into a segment of the ξ axis. In a more complicated geometries, a cut may be introduced into the transformed plane as illustrated in the branching circuit of Figure 5. Further interesting examples are presented in the paper by Kumar et al.[5]

Second, the coordinate transformation should be orthogonal or nearly orthogonal within boundary-layer regions in order to provide accurate resolution of the viscous stresses. Also, the use of a two-layer zero-equation turbulence model such as Cebeci-Smith[6] or Baldwin-Lomax[7] requires the local normal to the boundary to be defined in order to determine, among other items, the point at which the eddy viscosity switches from the "inner" to the "outer" formulation. Clearly, an orthogonal or nearly orthogonal grid within the boundary layers eliminates the need for interpolation among the grid points to determine the inner and outer eddy viscosity profiles along the local normal. It should be emphasized, however, that there is no universal *a priori* reason for requiring the grid to be orthogonal within the inviscid region of the flow. For example, MacCormack and Paullay,[8] using the explicit finite-difference algorithm of MacCormack,[9] demonstrate that a non-orthogonal grid, which aligns one family of coordinate lines with a shock surface, yields a more accurate solution than a grid which has no coordinate lines aligned with the shock.

Third, the coordinate generation technique must provide the capability for some degree of control on grid spacing. For computations utilizing zero-equation turbulent eddy viscosity models and solid wall boundary conditions,

the normal mesh spacing Δn adjacent to the solid boundary should satisfy $\Delta n^+ \equiv \Delta n\, u_*/\nu_w \leq 2$ to 5, where $u_* = \sqrt{\tau_w/\rho_w}$, τ_w is the local wall shear stress, ρ_w is the density at the wall, and ν_w is the kinematic viscosity at the wall.[10,11,12] For computations employing a multi-equation turbulence model, a more stringent mesh spacing requirement is usually required,[10,11,13] typically $\Delta n^+ \leq 0.2$ to 0.5. Since these requirements imply a physical mesh spacing Δn much smaller than the local boundary layer thickness δ (typically much less than 1% of δ), it is clear that grid control is important.

Fourth, the curvilinear coordinates should display smoothly varying metric coefficients. Grid smoothness criteria have not been generally developed, and can be expected to depend on the nature of the flow (e.g., viscous or inviscid) and numerical algorithm. For example, Forester[14] suggests the following rough guidelines for compressible potential flow calculations, namely, a) the stretching factor for successive grid cells should be less than two, and b) the rate of mesh twisting should be less than about one half radian. Also, a discussion of the relationship between coordinate grids and numerical fluid dynamics algorithms is presented in the papers of Hindman[15] and Thomas and Lombard.[16]

The above list of desirable coordinate attributes is certainly incomplete, and experience in particular types of flow computation have added further information. For example, computations of high Reynolds number viscous cascade flows have demonstrated a superiority of the "C"-type grid shown in Figure 4 over the sheared-type and "O"-type grids.[17-19] The "C"-type grid provides good resolution of the stagnation region near the leading edge, an attribute which is lacking in the sheared-type. It also provides a natural set of coordinates for the wake region, which is less easily achieved in the "O"-type.

BRIEF REVIEW OF RECENT APPLICATIONS

Since a major review of grid generation techniques is pending publication,[1] our attention is focused on presenting a representative sample of *recent* applications to internal flow computations. Although in the author's opinion it is too early in the development of the field of grid generation techniques to attempt a comprehensive categorization of methods, it is nonetheless instructive to discuss recent work in an approximate framework. The categorizing parameter has been arbitrarily chosen to be the nature or type of the mesh generation algorithm (e.g., elliptic partial differential equations, conformal transformation, etc.) as opposed to, say, the character of the resultant grid (e.g., orthogonal, non-orthogonal, controllable mesh spacing,

etc.). It is emphasized that this discussion is intended to highlight a selection of recent applications in internal flows, and the reader is referred to the other papers in this Conference and the recent workshop at NASA Langley[20] for additional discussion in the following categories.

Elliptic Partial Differential Equations

Although the idea of curvilinear coordinate generation by means of elliptic partial differential equations has a considerable history (see, for example, Refs. 21 to 23), the popularity of the concept was substantially increased by the contributions of Thompson, Thames and Mastin.[24,25] Several improvements were added by Ghia et al.[26,27] and the technique was applied to generation of curvilinear grids for cascade flows. The technique was utilized by Knight[28-31] in computations of supersonic aircraft inlets, and Kumar[32] for hypersonic scramjet flows. Johnson and Thompson[33] treated the confluence of several waterways in Charleston Harbor, and Kumar et al.[5] computed the flowfield in several multi-channel configurations. The technique was also applied to turbomachinery flows by Camamero and Younis,[34] Camamero and Reggio[35] using the multi-grid technique for the solution of the non-linear partial differential equations. The method was utilized for grid generation in three-dimensional ducts by Roberts and Forester.[36]

A number of modifications of the method of Thompson, Thames and Mastin have been developed in order to improve grid orthogonality, mesh spacing control, and other features. Sorenson and Steger introduced direct control of the grid spacing and grid orthogonality near boundaries.[37-39] Their method was employed by Chausee et al.[40] and Biringen et al.[41,42] for the computation of two-dimensional high speed aircraft inlet flowfields. Visbal and Knight[43,44] developed a general technique to generate orthogonal or nearly orthogonal curvilinear grids with direct control of grid spacing, which is discussed in detail in a later section. The technique was applied to the generation of curvilinear grids for a variety of internal flow configurations including inlets, diffusers and cascades. Mobley and Stewart[45] developed an orthogonal grid generation scheme based on the method of Thompson, Thames and Mastin and the technique of Pope[46] and applied the method to several internal flow geometries. Thomas and Middlecoff[47] developed techniques for approximate control of grid spacing, and applied them to generation of coordinates for three-dimensional internal flows. Roach and Sankar[19] introduced a formulation for the source terms to help improve grid orthogonality.

Conformal Transformations

The method of grid generation by conformal transformation has a rich and varied history (see, for example, Ref. 48), with substantial emphasis on aerodynamic applications. The generalized Schwartz-Christoffel transformation[49] for curvilinear boundaries was applied by Davis[50] to the generation of orthogonal coordinates for a selection of internal and external flow configurations. The technique was extended by Sridhar and Davis[51] to the construction of orthogonal grids for a greater variety of two-dimensional internal flow geometries including ducts, nozzles and cascades. The conformal mapping procedure of Anderson[52] was employed in the computation of the flowfield in a turbofan forced mixer nozzle by Anderson and Hawkins.[53] Aircraft inlet configurations including centerbodies have been treated by Ives and Menor.[54] A double conformal mapping procedure was employed by Sockol[55] for the generation of C-type cascade grids. Transonic potential flow past an airfoil in a wind tunnel was treated by Doria and South[56] using curvilinear coordinates generated by a combination of shearing and Schwartz-Christoffel transformations following the approach of Caughey.[57]

Algebraic Techniques

In recent years, a significant effort has been focused on the application of algebraic methods to the generation of curvilinear grids for internal and external flows. A general algebraic scheme was introduced by Eiseman[58] for two-dimensional flows, and employed for the generation of curvilinear coordinates for airfoil cascades. The technique was later extended to three-dimensional geometries,[59] and additional control of grid spacing and orthogonality was devised.[60] The method was utilized by Shamroth et al.[61] for the computation of viscous cascade flows. Eiseman and his colleagues also developed a general "tube-like" coordinate treatment of three-dimensional duct flows.[62,63] The approach was utilized by Levy et al.[64] in the computation of three-dimensional turbulent subsonic flow in ducts of super-elliptic and elliptic cross-section. The algebraic approach of Eiseman was extended by Smith and Weigel[65], and Smith et al.[66] The technique was applied to turbofan lobed mixers and inlets by Kowalski,[67] and to non-axisymmetric nozzles by Swanson.[68] In addition, Drummond and Weidner[69] utilized the algebraic mapping formulation of Roberts[70] and Holst[71] in the computation of scramjet engine flows.

The inherent flexibility of algebraic methods is an important feature. However, unlike conformal transformations and certain elliptic formulations[24-27]

which possess a maximum principle, the algebraic methods do not necessarily always generate a one-to-one transformation (i.e., there can be crossing of grid lines of the same family[61,67]), and therefore the coordinate transformation must be carefully evaluated *a posteriori* by, for example, interactive graphics.[66]

Other Methods

Although the majority of coordinate transformation techniques applied to internal flows can be classified according to the above three categories, several additional approaches have been employed. McNally[72] developed a simple predictor-corrector method for constructing two-dimensional orthogonal grids between two arbitrary boundaries. The method was utilized by Graves[73] for axisymmetric blunt bodies, and is extendable to certain internal flow configurations as well. Steger and Sorensen[74] developed a hyperbolic grid generation scheme for two- and three-dimensional flows. The technique, however, does not guarantee that a one-to-one mapping will be obtained, and cross-over of grid lines of the same family can occur. Modest success has also been achieved in the development of flow-adaptive grids. For example, the method of Hirt et al.[75] has been employed by Hasen,[76] Perry,[77], and Kowalski et al.[78] for the alignment of one family of grid lines with the local flow direction for nozzle and aftbody flows.

A METHOD FOR GENERATING TWO-DIMENSIONAL ORTHOGONAL OR NEARLY ORTHOGONAL GRIDS WITH DIRECT CONTROL OF MESH SPACING

Introduction

The purpose of this section is to present the method developed by Visbal and Knight[43,44] for the generation of orthogonal or nearly orthogonal curvilinear grids and to demonstrate its application to two-dimensional internal flows. In Figure 3, a flow region ABCD is shown. The region is taken to be bounded by two straight line segments Γ_1 and Γ_2 of arbitrary length and two arbitrary curvilinear boundaries Γ_3 and Γ_4. The boundaries Γ_3 and Γ_4 need not be known analytically nor have a continuous slope. The general flow region is typical of the geometries of two-dimensional diffusers, ducts and inlets. The configuration also incorporates the "C"-type grid utilized for airfoil cascades as shown in Figure 4, where Γ_1, Γ_2, Γ_3 and Γ_4 are identified as curves AF, DE, ABCD and FE, respectively.

The method is characterized by four major features, which are indicated in

Table 1, and can be operated in two modes. In Mode 1, an orthogonal grid is constructed with weak control of grid spacing in the η-direction near Γ_3 and Γ_4.

TABLE 1

OPERATIONAL MODES OF GRID GENERATION TECHNIQUE OF VISBAL AND KNIGHT

	Feature	Mode 1	Mode 2
1.	Orthogonal Grid	Yes	Nearly-Orthogonal
2.	Control of grid spacing in η-direction near Γ_3 and Γ_4	Weak Control	Strong Control
3.	Arbitrary user-specified grid spacing along Γ_3 or Γ_4	Yes	Yes
4.	User manipulation of forcing functions	Not required	Not required

In Mode 2, strong control over grid spacing in the η-direction near Γ_3 and Γ_4 is obtained at the expense of achieving a less orthogonal grid. Both modes permit arbitrary specification of grid points along Γ_3 or Γ_4. Also, the method is fully automatic for both modes, i.e., user manipulation of forcing functions is *not* required.

The method is based upon the use of Poisson's equation, and represents an extension of the technique of Thompson, Thames and Mastin.[24] Although the method of Thompson et al. provides control of grid spacing along both curvilinear coordinate directions by inclusion of a general class of forcing functions (i.e., inhomogeneous terms in the governing elliptic equations), the parameters in the forcing functions are not known *a priori* except in certain simple geometries and must therefore be adjusted repeatedly by the user until the desired mesh spacing is achieved. In addition, the transformation is generally non-orthogonal. Sorenson and Steger[37-39] incorporated a dynamic adjustment of the forcing functions with the method of Thompson et al. to provide orthogonality and control of the grid spacing in the vicinity of selected physical boundaries. Their method, however, does not achieve near-orthogonality within the *entire* flow region and highly skewed grids can result.[41,42] In the techniques of Ghia et al.[26,27] and Thomas and Middlecoff,[47] the forcing functions were evaluated along the boundaries employing simplified forms of the governing equations and the prescribed Dirichlet boundary conditions. Interpolation was used to obtain the corresponding values at the

interior of the region. The resulting forcing functions, however, are not consistent with orthogonality and significant skewness can occur depending upon the curvature of the physical domain and the specified distribution of mesh points along the boundaries.[43] In the work of Mobley and Stewart,[45] some measure of control of the mesh spacing in orthogonal grids was achieved through the use of user-specified stretching functions. However, the stretching function corresponding to an *arbitrary* distribution of mesh points along a particular boundary (for example, Γ_3 in Figure 3) is in general unknown, which constitutes a serious practical disadvantage. Roach[19] proposed a form for one of the forcing functions which is consistent with orthogonality only under certain particular conditions and assuming an appropriate treatment of the boundary conditions is given; consequently, the transformation is in general non-orthogonal.

The method of Visbal and Knight is a two-part procedure consisting of an intermediate and final transformation. In the intermediate transformation, an orthogonal grid is generated with a user-specified distribution of mesh points on Γ_3 (see Figure 3). The final transformation can be operated in two modes. In Mode 1, an *orthogonal* grid is obtained with a user-specified distribution of mesh points along Γ_1 and Γ_3. Grid orthogonality is achieved, however, at the expense of precise control over mesh spacing in the η-direction. For example, the spacing in the η-direction adjacent to Γ_3 and Γ_4 is determined by the governing elliptic equations and cannot be specified by the user (e.g., for a diverging duct, the grid spacing in the η-direction will increase downstream in a manner similar to Figure 7 of Ref. 46). In Mode 2, a *nearly* orthogonal grid is obtained with a user-specified distribution of mesh points on Γ_1, Γ_2 and Γ_3 and with direct control of the mesh spacing in the η-direction adjacent to Γ_3 and Γ_4. This second mode is particularly important in turbulent flow computations where the requirement of resolution of the viscous sublayer is more important than strict orthogonality. Finally, the entire method is fully automatic and does not require iterative adjustment of forcing functions by the user.

Intermediate Transformation

In the first step an orthogonal grid is generated with a user-specified distribution of the mesh points along Γ_3 (see Figure 3). The purpose of this intermediate step will be discussed at the end of this subsection. The intermediate transformation $(\xi(x,y), \chi(x,y))$ satisfies the Poisson equations[43,44]

$$\nabla^2\xi = \phi(\xi,\chi) = \frac{1}{h_\xi h_\chi}\frac{\partial}{\partial\xi}\left(\frac{h_\chi}{h_\xi}\right) \tag{1a}$$

$$\nabla^2\chi = 0 \tag{1b}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. The quantities h_ξ and h_χ are the scale factors

$$h_\xi = (x_\xi^2+y_\xi^2)^{1/2} \tag{2a}$$

$$h_\chi = (x_\chi^2+y_\chi^2)^{1/2} \tag{2b}$$

where $x_\xi = \frac{\partial x}{\partial\xi}$, etc. The boundary conditions on ξ are (see Figure 6)

$$\xi = 0 \quad \text{on} \quad \Gamma_1 \tag{3a}$$

$$\xi = 1 \quad \text{on} \quad \Gamma_2 \tag{3b}$$

$$\xi = F_3(t) \quad \text{on} \quad \Gamma_3 \tag{3c}$$

$$\frac{\partial\xi}{\partial n} = 0 \quad \text{on} \quad \Gamma_4 \tag{3d}$$

where t is the arc length along Γ_3 measured from Γ_1 and n is the normal distance measured from Γ_i. Equation (3c) simply indicates that the user has distributed the mesh points along Γ_3 in an *arbitrary* monotonic fashion as desired and no analytic expression for $F_3(t)$ is required. The boundary conditions on χ are

$$\frac{\partial\chi}{\partial n} = 0 \quad \text{on} \quad \Gamma_1, \Gamma_2 \tag{4a,b}$$

$$\chi = 0 \quad \text{on} \quad \Gamma_3 \tag{4c}$$

$$\chi = 1 \quad \text{on} \quad \Gamma_4. \tag{4d}$$

The forcing function ϕ in Equation (1a) is consistent with the generation of an orthogonal grid through Equation (1) and the boundary conditions (3) and (4). The form of ϕ results from the general expression for the Laplacian in orthogonal curvilinear coordinates

$$\nabla^2 = \frac{1}{h_\xi h_\chi}\left[\frac{\partial}{\partial\xi}\left(\frac{h_\chi}{h_\xi}\frac{\partial}{\partial\xi}\right) + \frac{\partial}{\partial\chi}\left(\frac{h_\xi}{h_\chi}\frac{\partial}{\partial\chi}\right)\right].$$

The intermediate transformation is obtained by inverting Equations (1) to yield

$$\tilde{L}(x) = -\tilde{J}^2 x_\xi \phi \tag{5a}$$

$$\tilde{L}(y) = -\tilde{J}^2 y_\xi \phi \tag{5b}$$

where

$$\tilde{L} = \tilde{\alpha}\frac{\partial^2}{\partial\xi^2} - 2\tilde{\beta}\frac{\partial^2}{\partial\xi\partial\chi} + \tilde{\gamma}\frac{\partial^2}{\partial\chi^2}$$

$$\tilde{\alpha} = x_\chi^2 + y_\chi^2$$

$$\tilde{\beta} = x_\xi x_\chi + y_\xi y_\chi$$

$$\tilde{\gamma} = x_\xi^2 + y_\xi^2$$

$$\tilde{J} = x_\xi y_\chi - x_\chi y_\xi$$

and the forcing function ϕ is

$$\phi = [\tilde{\gamma}(x_\chi x_{\xi\chi} + y_\chi y_{\xi\chi}) - \tilde{\alpha}(x_\xi x_{\xi\xi} + y_\xi y_{\xi\xi})]/\tilde{\alpha}\tilde{\gamma}^2 \tag{5c}$$

The transformed boundary conditions are prescribed on the boundaries γ_1, γ_2, γ_3, γ_4 of the unit square in the (ξ,χ) plane, as indicated in Figure 7. The expressions obtained from Equations (3) and (4) are

$$x_\xi x_\chi + y_\xi y_\chi = 0 \quad \text{and } G_1(x,y) = 0 \text{ on } \gamma_1 \tag{6a}$$

$$x_\xi x_\chi + y_\xi y_\chi = 0 \quad \text{and } G_2(x,y) = 0 \text{ on } \gamma_2 \tag{6b}$$

$$t = \tilde{F}_3(\xi) \text{ and } G_3(x,y) = 0 \text{ on } \gamma_3 \tag{6c}$$

$$x_\xi x_\chi + y_\xi y_\chi = 0 \text{ and } G_4(x,y) = 0 \text{ on } \gamma_4 \tag{6d}$$

As indicated previously, the function $\tilde{F}_3(\xi)$ in Equation (6c) denotes that the mesh points are arbitrarily distributed by the user along Γ_3 and no analytic expression for $\tilde{F}_3(\xi)$ is required. The functions $G_1(x,y) = 0,\ldots,G_4(x,y) = 0$

simply indicate that the shape of the boundaries $\Gamma_1,\ldots,\Gamma_4$ is known either analytically or through specification of a discrete number of points.

The inverted intermediate transformation equations are solved using a two-step iterative procedure. First, the point successive overrelaxation (SOR) method is applied for typically one to five times at all interior points to the finite-difference form of Equation (5) which are obtained using second-order accurate centered difference approximations for the derivatives. Second, the forcing function ϕ is updated by the use of Equation (5c), and the mesh points are redistributed along Γ_1, Γ_2 and Γ_4 according to Equations (6a), (6b) and (6d), which are approximated using second order accurate centered or one-side differences as required. This results in three systems of simultaneous non-linear equations which are solved by Newton's method. In thos cases where the shape of the boundaries Γ_3 and Γ_4 are defined by a suitable discrete set of points, piecewise cubic spline interpolation is used to determine the functions G_3 and G_4. These two steps are repeated until the following requirements are satisfied:

(a) The maximum displacement of the interior mesh points in the physical plane during a given SOR iteration is a small fraction of the local mesh size in the ξ and χ directions. A value of 1% was found to be satisfactory and was used in all cases presented here.

(b) The condition of orthogonality at Γ_1, Γ_2 and Γ_4 is effectively satisfied, i.e.,

$$|\tilde{\beta}|/(\tilde{\alpha}\tilde{\gamma})^{1/2} \leq \epsilon$$

for all points on γ_1, γ_2 and γ_4. This implies that the absolute value of the cosine of the intersection angles formed by the coordinate lines at the boundaries Γ_1, Γ_2 and Γ_4 are less than the specified parameter ϵ. For the results presented here, the value chosen for ϵ is 0.01. This corresponds to a maximum deviation from orthogonality of 0.57 degrees (i.e., the intersection angles are between 89.43 and 90.57 degrees).

It is important to note that SOR was used for simplicity and other numerical techniques could be employed if additional efficiency is desired.

The purpose of the intermediate transformation is to *efficiently* obtain the forcing function ϕ and distribution of mesh points along Γ_4 which are consistent with orthogonality. As indicated in the next section, this distribution of mesh points along Γ_3 and Γ_4 are then prescribed as Dirichlet

boundary conditions in the solution of the final transformation. Also, the intermediate grid is employed in the determination of the forcing functions and the initial guess for the solution of the final transformation equations (the intermediate mesh must be stored for this purpose). Our experience indicates that the intermediate transformation is a very efficient technique particularly when highly refined grids near Γ_3 and Γ_4 are required in the final mesh. In the intermediate transformation a relatively coarse grid in the χ direction is employed. This improves the efficiency of the implementation of the Neumann boundary condition on Γ_4 and also the dynamic adjustment of the forcing function ϕ. Past experience indicates that the use of a Neumann boundary condition on Γ_4 in the presence of a highly refined grid is very time consuming. In addition, updating the forcing functions in a highly clustered mesh may result in numerical instabilities.[38,39] This does not occur in a coarse grid, even when a poor initial guess is prescribed.

Final Transformation

Although the intermediate transformation exhibits the desirable characteristics of orthogonality and arbitrary specification of mesh points along the boundary Γ_3, it does not provide control over the mesh spacing in the χ-direction as desired. The final transformation is then introduced to generate a grid which manifests the features of orthogonality or near-orthogonality, and weak or strong control of the grid spacing in the η-direction near Γ_3 and Γ_4 as desired (features #1 and #2 in Table 1).

The final transformation is motivated by the simple observation that a transformation

$$\chi = \chi(\eta) \tag{7}$$

can be introduced to allow a concentration or stretching of the grid points in the η direction. The governing equation for ξ becomes[43]

$$\nabla^2\xi = P(\xi,\eta) = \frac{1}{h_\xi h_\eta}\frac{\partial}{\partial\xi}\left(\frac{h_\eta}{h_\xi}\right) \tag{8a}$$

and the equation for η is[28,45,47,79]

$$\nabla^2\eta = Q(\xi,\eta) = \sigma(\eta)(\eta_x^2+\eta_y^2) \tag{8b}$$

$$\sigma(\eta) = -\frac{d^2\chi}{d\eta^2}\Big/\frac{d\chi}{d\eta} \tag{8c}$$

and the scale factors h_ξ and h_η are

$$h_\xi = (x_\xi^2+y_\xi^2)^{1/2} \tag{9a}$$

$$h_\eta = (x_\eta^2+y_\eta^2)^{1/2} \tag{9b}$$

The boundary conditions on ξ (see Figure 8) along Γ_1, Γ_2 and Γ_3 are given by Equations (3a), (3b) and (3c), respectively. The boundary condition on Γ_4 is

$$\xi = F_4(t) \tag{10}$$

where $F_4(t)$ denotes the distribution of the ξ-lines along Γ_4 obtained from the intermediate transformation and t is the arc length along Γ_4. The boundary conditions on η for orthogonal grids (Mode 1) are

$$\eta = F_1(s) \text{ on } \Gamma_1 \tag{11a}$$

$$\frac{\partial \eta}{\partial n} = 0 \quad \text{on } \Gamma_2 \tag{11b}$$

$$\eta = 0 \quad \text{on } \Gamma_3 \tag{11c}$$

$$\eta = 1 \quad \text{on } \Gamma_4 \tag{11d}$$

where s is the arc length along Γ_1 measured from Γ_3. Equation (11a) indicates that the η-lines are arbitrarily distributed along Γ_1 as desired by the user, and no analytic expression for $F_1(s)$ is required. For nearly orthogonal grids with direct control of the normal distance of the η = constant lines from Γ_3 and Γ_4 (i.e., Mode 2), the boundary condition on Γ_2 becomes $\eta = F_2(s)$ indicating a user-specified distribution on η-lines on Γ_2.

The final transformation is obtained by inverting Equations (8) to yield

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -J^2(x_\xi P + x_\eta Q) \tag{12a}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2(y_\xi P + y_\eta Q) \tag{12b}$$

where

$$\alpha = x_\eta^2 + y_\eta^2$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta$$

$$\gamma = x_\xi^2 + y_\xi^2$$

$$J = x_\xi y_\eta - x_\eta y_\xi$$

The transformed boundary conditions for the final transformation for orthogonal grids (Mode 1) are given by (see Figure 9)

$$s = \tilde{F}_1(\eta) \text{ and } G_1(x,y) = 0 \text{ on } \gamma_1 \tag{13a}$$

$$x_\xi x_\eta + y_\xi y_\eta = 0 \text{ and } G_2(x,y) = 0 \text{ on } \gamma_2 \tag{13b}$$

$$t = \tilde{F}_3(\xi) \text{ and } G_3(x,y) = 0 \text{ on } \gamma_3 \tag{13c}$$

$$t = \tilde{F}_4(\xi) \text{ and } G_4(x,y) = 0 \text{ on } \gamma_4 \tag{13d}$$

Equation (13a) implies that the user has distributed the mesh points on Γ_1 as desired. Equation (13c) denotes the same user-specified distribution of grid points on Γ_3 as employed in the intermediate transformation. Equation (13d) denotes the grid point distribution on Γ_4 obtained from the intermediate transformation. For operation in Mode 2, Equation (13b) is replaced by

$$s = \tilde{F}_2(\eta) \text{ and } G_2(x,y) = 0 \text{ on } \gamma_2 \tag{13e}$$

which denotes a user-supplied grid distribution on Γ_2.

The forcing function $P(\xi,\eta)$ is determined from the intermediate transformation according to

$$P(\xi,\eta) = \phi(\xi,\chi(\eta)) \tag{14}$$

In general, the function $\chi(\eta)$ is not known a priori. However, since $\phi(\xi,\chi)$ usually varies smoothly with χ, satisfactory results are obtained when the initial guess for x and y (used in the numerical solution of the inverted final transformation equations) and linear interpolation in the physical plane along the ξ-lines (obtained in the intermediate transformation) are employed. For operation in Mode 1, the function $P(\xi,\eta)$ may be updated during the solution although experience with highly curved and clustered grids indicates that such adjustment is unnecessary. For operation in Mode 2, iterative adjustment of P

in the final transformation according to the *orthogonal* expression (8a) is inconsistent and is therefore not performed.

The forcing function Q is determined as follows. Application of the condition of local orthogonality $\beta = 0$ in Equations (12) gives

$$Q = \gamma S/J^2 \text{ where} \tag{15a}$$

$$S = T - \frac{y_\eta}{x_\xi}\frac{R}{\gamma} \text{ if } x_\xi \neq 0 \tag{15b}$$

$$S = T + \frac{x_\eta}{y_\xi}\frac{R}{\gamma} \text{ if } y_\xi \neq 0 \tag{15c}$$

where

$$T = -(x_\eta x_{\eta\eta}+y_\eta y_{\eta\eta})/(x_\eta^2+y_\eta^2) \tag{16}$$

$$R = x_\xi y_{\xi\xi}-y_\xi x_{\xi\xi} \tag{17}$$

It can be simply shown that $S = \sigma$ (see Equation (8b)). For an orthogonal grid (Mode 1), therefore, S is a function of η alone (see Equation (7)), and is evaluated at $\xi = 0$ according to Equations (15)-(17) with the ξ-derivatives approximated using the intermediate solution and linear interpolation. These derivatives can be updated during the solution, although experience indicates that such adjustment is unnecessary.

For a nearly orthogonal grid with controlled spacing of the η-lines (Mode 2), the function R is obtained from the intermediate transformation using linear interpolation. Further iterative adjustment of R is inconsistent with the expressions for Q in Equations (15) obtained using the assumption of orthogonality and is therefore not performed. If the distribution of the η-lines is specified in terms of the distance $s(\xi,\eta)$ measured from Γ_3 along the ξ-lines, then

$$s_\eta^2 = x_\eta^2 + y_\eta^2 \tag{18a}$$

and

$$s_\eta s_{\eta\eta} = x_\eta x_{\eta\eta} + y_\eta y_{\eta\eta} \tag{18b}$$

From Equations (15) and (17), the function T becomes

$$T = -s_{\eta\eta}/s_\eta \tag{19}$$

Therefore, T can be determined for any arbitrary distribution of the η-lines. In this research, the η-lines are distributed exponentially near the boundaries Γ_3 and Γ_4, and uniformly in between. This distribution would be appropriate for numerical simulation of high Reynolds number flows with thin boundary layers along Γ_3 and Γ_4. The resulting expression for T is

$$T = \begin{cases} -C_1/\eta_1 & 0 \le \eta \le \eta_1 \\ 0 & \eta_1 < \eta < \eta_2 \\ C_2/(1-\eta_2) & \eta_2 \le \eta \le 1 \end{cases} \qquad (20)$$

where C_1, C_2, η_1 and η_2 are slowly varying functions of ξ. This formulation permits control of the normal distance of selected η-lines η_1^* and η_2^* ($0 < \eta_1^* < \eta_1$, $\eta_2 < \eta_2^* < 1$) from the respective boundaries Γ_3 and Γ_4. Equation (20) may be integrated along each ξ-line to determine $s(\xi,\eta)$ as a function of C_1, C_2, η_1 and η_2. The specified control of the $\eta = \eta_1^*$ and $\eta = \eta_2^*$ lines provides a pair of nonlinear coupled equations for C_1 and C_2 for each ξ-line which are easily solved by Newton's method.

The inverted final transformation Equations (12) and boundary conditions (13) are solved using point SOR. Convergence was assumed when criteria (a) discussed in the previous subsection was achieved. Again, it is noted that SOR was employed for simplicity and other techniques could be employed to improve computational efficiency.

Results

The capabilities of the method are illustrated for the various flow regions shown in Figures 10, 12 and 13. The results are summarized in Table 2 and discussed in detail below.

Example 1: Diffuser/Nozzle Using Mode 2

In this example, the capability of the method is demonstrated for generating a very nearly-orthogonal mesh with an arbitrary non-uniform distribution of ξ-lines along Γ_3 and with direct control of the spacing of selected η-lines throughout the entire domain (Mode 2). The physical region, shown in Figure 10a is representative of a diffuser or nozzle. The upper boundary is formed by two horizontal segments and a sinusoidal curve. The ratio of the heights of the right and left boundaries is four to one. The mesh points are arbitrarily distributed along Γ_3 and are clustered in the

TABLE 2

CHARACTERISTICS OF COORDINATE TRANSFORMATIONS

	Example No. 1	2	3	4	5
Intermediate Mesh					
No. of ξ-lines	52	31	31	59	59
No. of χ-lines	8	8	11	11	11
Final Mesh					
No. of ξ-lines	52	31	31	59	59
No. of η-lines	31	31	31	31	31
Maximum θ_I	4.6°	4.3°	2.8°	10.9°	2.3°
Maximum θ_L	1.9°	0.7°	0.7°	3.3°	2.3°
Maximum θ_U	1.5°	1.4°	0.5°	1.0°	1.5°
$\bar{\theta}_I$	1.0°	1.0°	0.7°	2.5°	0.6°
$\bar{\theta}_L$	0.6°	0.4°	0.4°	1.0°	0.7°
$\bar{\theta}_U$	0.4°	0.4°	0.3°	0.4°	0.6°
Computer time (sec) using IBM 370/168 Fortran G Compiler)	16.0	19.0	19.0	32.0	36.0

Legend: θ = deviation from orthogonality (i.e., the absolute value of the amount by which the angle of intersection of the coordinate lines differs from 90°)
$\bar{\theta}$ = average deviation from orthogonality

Subscripts: I = Interior mesh points
L = Mesh points on Γ_3
U = Mesh points on Γ_4

vicinity of the center of the flow region, as illustrated in Figure 10a. The ratio of maximum and minimum physical grid spacing on Γ_3 is 2.7. The expression (20) was employed for T with $\eta_1 = 1/3$ and $\eta_2 = 2/3$. The values of C_1 and C_2 at each ξ-line were determined by the requirements that the normal distance of the $\eta = \eta_1^* = 2/15$ line from Γ_3 be constant, and the normal distance of the $\eta = \eta_2^* = 13/15$ line from Γ_4 increase smoothly to twice its value on Γ_1 over the length of the flow region. The minimum and maximum values of C_1 are 0.582 and 2.817, respectively, and the corresponding values for C_2 are 0.579 and 1.916.

The initial guess is shown in Figure 10a, and the intermediate transformation is shown in Figure 10b. The final transformation is shown in Figure 10c, where only the odd numbered η-lines have been plotted. The results in Table 2 indicate that the transformation is very nearly-orthogonal, with an average

deviation from orthogonality of 1.0° in the interior. The average absolute deviation of the η_1^* line from the specified normal distance to Γ_3 is 3.8%, and the maximum absolute deviation is 10.5%. The corresponding figures for the η_2^* line relative to Γ_4 are 9.6% and 35.0%, respectively. The maximum deviations occur in the vicinity of the x-station where the slope of the upper surface is a maximum. In evaluating the success of the control of the spacing of the η_1^* and η_2^* lines, it is worthwhile to note that when the condition of strict orthogonality was imposed (i.e., Mode 1), the normal distance of the η_1^* and η_2^* lines increased by 300% between Γ_1 and Γ_2 as expected from the overall change in shape of the flow region. The results of Example 1 indicate that direct control of the η-lines can be accomplished with a very slight increase in deviation of the transformation from orthogonality.

Example 2: Diffuser/Nozzle with $P(\xi,\eta) = 0$ Using Mode 2

The purpose of this example is to illustrate the capability of generating a nearly-orthogonal transformation (Mode 2) with a specified forcing function $P(\xi,\eta)$. The flow region is the same as in Example 1. For the purposes of simplicity, P was taken to be zero. The ξ-lines were redistributed along both Γ_3 and Γ_4 in the intermediate mesh in order to satisfy orthogonality. The expression (20) was used for T. The values of η_1, η_2 and C_1 and C_2 at Γ_1 are the same as for Example 1. The normal distance of the $\eta = \eta_1^* = 2/15$ and $\eta = \eta_2^* = 13/15$ lines were controlled in the same manner as for Example 1.

The final transformation is shown in Figure 11, where only the odd numbered η-lines are shown. The transformation is very nearly orthogonal, as indicated in Table 2, with an average deviation from orthogonality of 1.0° in the interior. The average absolute deviation of the η_1^* line from the specified normal distance to Γ_3 is 2.9%, and the maximum absolute deviation is 11.3%. The corresponding figures for the η_2^* line referenced to Γ_4 are 6.4% and 34.8%, respectively. The final transformation also demonstrates the mesh spacing along Γ_3 corresponding to $P = 0$ increases monotonically with the height of the flow region as expected.

Example 3: Asymmetric Half Annulus Using Mode 2

In this example, the capability is demonstrated for generating a very nearly orthogonal grid (Mode 2) with controllable mesh spacing (in the ξ and η directions) in a highly curved region. The physical region is shown in Figures 12a,b. The boundaries Γ_3 and Γ_4 are two non-concentric circles of radius 1.0 and 2.5, respectively. The ratio of the lengths of the boundaries Γ_1 and Γ_2 is five to one. The ξ-lines are redistributed along Γ_3 with equal increments in arc length. The expression (20) was employed for T with

$\eta_1 = 7/15$ and $\eta_2 = 2/3$. The values of C_1 and C_2 at each ξ-line were determined by the requirements that the normal distance of the $\eta = \eta_1^* = 1/30$ line from Γ_3 be constant, and the normal distance of $\eta = \eta_2^* = 29/30$ from Γ_4 increase smoothly to four times its value on Γ_1 over the length of the flow region.

The final mesh is shown in Figure 12b. The results in Table 2 indicate that the transformation is very nearly orthogonal, with an average deviation from orthogonality of 0.7° in the interior. The average absolute deviation of the η_1^* line from the specified normal distance to Γ_3 is 2.4%, and the maximum absolute deviation is 5.7%. The corresponding figures for the η_2^* line relative to Γ_4 are 3.4% and 5.7%, respectively. It was found in this example that for highly curved regions, the term R in Equations (15) is of great importance if near-orthogonality is desired. The forms of the forcing function Q suggested by Ghia et al.[26,27] and Thomas and Middlecoff,[47] which assume $R \equiv 0$, produce skewed grids for this type of domains.

Example 4: Airfoil Using Mode 2

In this example a C-grid was generated about an airfoil, as shown in Figure 13, using Mode 2 operation. For the purpose of simplicity the boundary Γ_3 is formed by a symmetric Joukowsky airfoil with a straight cut, and the boundary Γ_4 is elliptical with major and minor semi-axes of 2.87 and 1.5, respectively. The mesh points were arbitrarily distributed along Γ_3 with some clustering near the leading and trailing edges and a geometric-stretching along the cut. The term T was evaluated according to Equation (20) with $\eta_1 = 7/15$ and $\eta_2 = 2/3$. The values of C_1 and C_2 were determined by the requirements that the normal distance of the $\eta = \eta_1^* = 1/30$ line from Γ_3, and of the $\eta = \eta_2^* = 29/30$ line from Γ_4 be constant.

As shown in Table 2, a nearly orthogonal grid was obtained with an average deviation from orthogonality of 2.5° in the interior. The average absolute deviation of the specified normal distance for the $\eta = \eta_1^*$ and $\eta = \eta_2^*$ lines are 3.9% and 2.7%, respectively.

Example 5: Airfoil Using Mode 1

The previous case was also computed using the condition of strict orthogonality (Mode 1). A very nearly orthogonal grid was achieved (see Table 2), with an average deviation from orthogonality of 0.6° in the interior. The maximum deviation from the specified normal distance of the $\eta = \eta_1^*$ line from Γ_3 is 85.8%, which compares very unfavorably with 20.5% for the grid of Example 4. It is therefore clear that if strict orthogonality is imposed, direct control of the spacing of the η-lines cannot be enforced. It is however possible, as the present example shows, to achieve direct control of

the mesh spacing with only a slight deviation from orthogonality away from the boundaries.

A CURRENT RESEARCH PROBLEM IN GRID GENERATION FOR INTERNAL FLOWS

Despite the notable recent successes in developing grid generation techniques for internal flows, a number of important problems remain. The focus of this section is the description of one particular problem, namely the generation of a suitable grid for simply-connected three-dimensional internal flows.

A straightforward approach to grid generation for 3-D internal flows is to develop a sequence of 2-D curvilinear grids in planes normal to some prescribed centerline of the duct.[62] Each plane intersects the duct boundary along some continuous closed curve as shown in Figure 14. The problem of 3-D grid generation thereby reduces to the task of generating a sequence of 2-D body-oriented curvilinear grids.

Two basic approaches have been employed. The first method, which may be denoted the "single-focus" technique, introduces a polar-like curvilinear grid as illustrated in Figure 14. A branch cut is introduced, and the two sides of the cut Γ_1 and Γ_2 are mapped into the left and right boundaries γ_1 and γ_2 in the transformed plane. The focus Γ_3 is a singularity of the transformation (i.e., the Jacobian vanishes at the focus), and is mapped into the segment γ_3, while the entire outer boundary Γ_4 of the duct is mapped into the segment γ_4. The presence of the singularity requires special treatments in the fluid dynamic calculation.[36] The single-focus technique, however, is not well-suited for duct cross-sections whose boundary is convex (relative to the duct interior) over a large extent. This problem is illustrated in Figure 15, which represents, for example, a cross-section of a half-axisymmetric aircraft inlet upstream of the inlet throat. Clearly, the curvilinear grid is highly distorted near the upper and lower corners, and the single-focus approach hinders the achievement of resolution of the boundary layers in the vicinity of the corners. The extension of the focus into a "slit"†, as illustrated in Figure 16, alleviates the aforementioned difficulty albeit at an increase in the complexity of the fluid dynamic algorithm (i.e., the Jacobian is now singular at two points, namely b and c).

The second method, which maybe denoted the "artificial corner" technique,[47] introduces one or more fictitious corner points along the duct boundary. For a duct boundary with continuous tangent, four artificial corners are added as shown in Figure 17. The Jacobian vanishes at the artificial corners, however,

†This technique was also suggested to the author by G. Paynter.

which requires special treatment in the fluid dynamic algorithm. Also, the necessity of resolving the boundary layer on the entire boundary implies that the curvilinear coordinates, although possibly intersecting the boundary at right angles, must rapidly bend towards the adjacent section of the duct boundary (e.g., the ξ-lines emanating from γ_3 near point A must rapidly bend towards γ_1 in order to resolve the boundary layer on γ_1). Thus, the grid lines emanating from the duct boundary do not follow the normal to the boundary, and therefore interpolation is required when utilizing a zero-equation turbulence model as discussed in the first section.

Numerous additional problems in grid generation for internal flows can be cited. It is clear that the field is open to innovative techniques.

ACKNOWLEDGEMENTS

The assistance of M. Visbal is gratefully acknowledged. This research was performed through sponsorship of the Air Force Flight Dynamics Lab and the Air Force Office of Scientific Research under AF Contract F33615-78-C-3008 and AFOSR Grants 80-0072 and 82-0040. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon. Figures 12b and 13 are reprinted with the permission of the American Institute of Aeronautics and Astronautics.

REFERENCES

1. Thompson, J., to appear, J. Comp. Physics.
2. McDonald, H. and Briley, W., "Computational Fluid Dynamic Aspects of Internal Flows", AIAA Paper 79-1445, 1979.
3. Moses, H., and Thomason, S., "Simultaneous Solution of Inviscid Flow and Turbulent Boundary Layers for a Compressor Cascade", AIAA Paper 81-1476, 1981.
4. Carlson, L., "Transonic Airfoil Analysis and Design Using Cartesian Coordinates", AIAA 2nd Computational Fluid Dynamics Conference, 1975, pp. 175-183.
5. Kumar, D., Hester, L., and Thompson, J., "Development of Partial Channel Flow for Arbitrary Input Velocity Distribution Using a Boundary Fitted Coordinate System", Non-Steady Fluid Dynamics, Amer. Soc. Mech. Engr., 1978, D. Crow and J. Miller, editors.
6. Cebeci, T., Smith, A.M.O., and Mosinskis, G., "Calculations of Compressible Adiabatic Turbulent Boundary Layers", AIAA J., 8, 1970, pp. 1974-1982.
7. Baldwin, B., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows", AIAA Paper 78-257, 1978.
8. MacCormack, R., and Paullay, A., "The Influence of the Computational Mesh on Accuracy for Initial Value Problems with Discontinuous or NonUnique Solutions", Computers and Fluids, 2, 1974, pp. 339-362.
9. MacCormack, R., "The Effect of Viscosity in Hypervelocity Impact Cratering", AIAA Paper No. 69-354, 1969.
10. Viegas, J., and Horstman, C., "Comparison of Multiequation Turbulence Models for Several Shock Separated Boundary Layer Interaction Flows", AIAA J., 17, 1979, pp. 811-820.

11. Coakley, T., and Bergman, M., "Effects of Turbulence Model Selection on the Prediction of Complex Aerodynamic Flows", AIAA Paper No. 79-0070, 1979.
12. Horstman, C., Hung, C., Settles, G., Vas, I., and Bogdonoff, S., "Reynolds Number Effects on Shock-Wave Turbulent Boundary Layer Interactions - A Comparison of Numerical and Experimental Results", AIAA J., 15, 1977, pp. 1152-1158.
13. Horstman, C., Settles, G., Bogdonoff, S., and Williams, D., "A Reattaching Free Shear Layer in Compressible Turbulent Flow - A Comparison of Numerical and Experimental Results", AIAA Paper No. 81-0333, 1981.
14. Forester, C., "Body-Fitted 3-D Full-Potential Flow Analysis of Complex Ducts and Inlets", AIAA Paper No. 81-0002, 1981.
15. Hindman, R., "Geometrically Induced Errors and their Relationship to the Form of the Governing Equations and the Treatment of Generalized Mappings", AIAA Paper No. 81-1008, AIAA 5th Computational Fluid Dynamics Conference, 1981, pp. 113-124.
16. Thomas, P., and Lombard, C., "The Geometric Conservation Law - a Link Between Finite-Difference and Finite-Volume Methods of Flow Computation on Moving Grids", AIAA Paper No. 78-1208, 1978.
17. Steger, J., and Bailey, H., "Calculation of Transonic Aileron Buzz", AIAA J., 18, 1980, pp. 249-255.
18. Steger, J., Pulliam, T., and Chima, R., "Implicit Finite-Difference Code for Inviscid and Viscous Cascade Flow", AIAA Paper No. 80-1427, 1980.
19. Roach, R., and Sankar, N., "Strongly Implicit Procedure Applied to the Flow Field of Transonic Turbine Cascades", AIAA Paper No. 81-0211, 1981.
20. Smith, R., ed., "Numerical Grid Generation Techniques", NASA CP 2166, 1980.
21. Thom, A., "The Flow Past Circular Cylinders at Low Speeds", Proc. Royal Soc. Ser. A., 141, 1933, pp. 651-666.
22. Thom, A., and Apelt, C., "Field Computations in Engineering and Physics", Van Nostrand, London, 1961.
23. Barfield, W., "An Optimal Mesh Generator for Langrangian Hydrodynamic Calculations in Two Space Dimensions", J. Comp. Physics, 6, 1970, pp. 417-429.
24. Thompson, J., Thames, F., and Mastin, C., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies", J. Comp. Physics, 15, 1974, p. 299-319.
25. Thompson, J., Thames, F., and Mastin, C., "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies", J. Comp. Physics, 24, 1977, p. 274-302.
26. Ghia, U., and Ghia, K., "Numerical Generation of a System of Curvilinear Coordinates for Turbine Cascade Flow Analysis", Univ. of Cincinnati Report No. AFL 75-4-7, 1975.
27. Ghia, U., Hodge, J., and Hankey, W., "An Optimization Study for Generating Surface-Oriented Coordinates for Arbitrary Bodies in High Reynolds Number Flow", AFFDL TR-77-117, December 1977.
28. Knight, D., "Numerical Simulation of Realistic High Speed Inlets Using the Navier-Stokes Equations", AIAA J., 15, 1977, pp. 1583-1589.
29. Knight, D., "Improved Calculation of High Speed Inlet Flows. Part I: Numerical Algorithm", AIAA J., 19, 1981, pp. 34-41.
30. Knight, D., "Improved Calculation of High Speed Inlet Flows, Part II: Results", AIAA J., 19, 1981, pp. 172-179.
31. Knight, D., "Calculation of High Speed Inlet Flows Using the Navier-Stokes Equations", J. Aircraft, 18, 1981, pp. 748-754.
32. Kumar, A., "Numerical Analysis of Scramjet Inlet Flow Field Using the 2-D Navier-Stokes Equations", AIAA Paper No. 81-0185, 1981.

33. Johnson, B., and Thompson, J., "A Discussion of Boundary-Fitted Coordinate Systems and their Applicability to Numerical Modeling of Hydraulics Problems", Misc. Paper No. H-78-9, U.S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., 1978.
34. Camamero, R., and Youris, M., "Efficient Generation of Body-Fitted Coordinates for Cascades Using Multigrid", AIAA J., 18, 1980, pp. 487-488.
35. Camamero, R., and Reggio, M., "Three-Dimensional Body-Fitted Coordinates for Turbomachine Applications", Computers in Flow Predictions and Experiments, ASME, 1981, pp. 51-57.
36. Roberts, D., and Forester, C., "Parabolic Procedure for Flows in Ducts with Arbitrary Cross Section", AIAA J., 17, 1979, pp. 33-40.
37. Sorenson, R., and Steger, J., "Simplified Clustering of Nonorthogonal Grids Generated by Elliptic Partial Differential Equations", NASA TM 73252, 1977.
38. Steger, J., and Sorenson, R., "Automatic Mesh-Point Clustering Near a Boundary in Grid Generation with Elliptic Partial Differential Equations", J. Comp. Physics, 33, 1979, pp. 405-410.
39. Sorenson, R., "A Computer Program to Generate 2-D Grids About Airfoils and Other Shapes", NASA TM 81198, 1980.
40. Chausee, D., and Pulliam, T., "Two-Dimensional Inlet Simulation Using a Diagonal Implicit Algorithm", AIAA J., 19, 1981, pp. 153-159.
41. Biringen, S., McMillan, O., and Chausee, D., "Calculation of Inlet Flow-fields by An Implicit Technique", AIAA Paper No. 80-0031, 1980.
42. Biringen, S., and McMillan, O., "A Numerical Simulation of 2-D Inlet Flow-Fields", AIAA Paper No. 81-0187, 1981.
43. Visbal, M., "Generation of Nearly-Orthogonal Body-Fitted Coordinate Systems in Two-Dimensional Singley-Connected Regions", M.S. Thesis, Dept. Mech. and Aero. Engr., Rutgers Univ., July 1980.
44. Visbal, M., and Knight, D., "Generation of Orthogonal and Nearly Orthogonal Coordinates with Grid Control Near Boundaries", AIAA J., 20, 1982, pp. 305-306.
45. Mobley, C., and Stewart, R., "On the Numerical Generation of Boundary-Fitted Orthogonal Curvilinear Coordinate Systems", J. Comp. Phys., 34, 1980, pp. 124-135.
46. Pope, S., "The Calculation of Turbulent Recirculating Flows in General Orthogonal Coordinates", J. Comp. Phys., 26, 1978, pp. 197-217.
47. Thomas, P., and Middlecoff, J., "Direct Control of Grid Point Distribution in Meshes Generated by Elliptic Equations", AIAA J., 18, 1980, pp. 652-656.
48. Ives, D., "A Modern Look at Conformal Mapping Including Multiply-Connected Regions", AIAA J., 14, 1976, pp. 1006-1011.
49. Woods, L., The Theory of Subsonic Plane Flow, Cambridge Univ. Press, 1961.
50. Davis, R., "Numerical Methods for Coordinate Generation Based on Schwartz-Christoffel Transformations", AIAA Paper No. 79-1463, 1979.
51. Sridhar, K., and Davis, R., "A Schwartz-Christoffel Method for Generating Internal Flow Grids", Computers in Flow Predictions and Experiments, ASME, 1981, pp. 35-44.
52. Anderson, O., "Calculation of Internal Viscous Flows in Axisymmetric Ducts at Moderate to High Reynolds Numbers", Computers and Fluids, 8, 1980, pp. 391-411.
53. Anderson, O., and Hawkins, G., "Development of a Parabolic Finite Difference Method for 3-D High Reynolds Number Viscous Internal Flows", Computers in Flow Predictions and Experiments, ASME, 1981, pp. 119-128.
54. Ives, D., and Menor, W., "Mesh Generation for Inlet and Inlet Centerbody Configurations Using Conformal Mapping and Stretching", AIAA Paper No. 81-0997, 1981.
55. Sockol, P., "Generation of C-Type Cascade Grids for Viscous Flow Computation", NASA CP 2166, 1980, pp. 437-448.

56. Doria, M., and South, J., "Transonic Potential Flow and Coordinate Generation for Bodies in a Wind Tunnel", AIAA Paper No. 82-0223, 1982.
57. Caughey, D., "A Systematic Procedure for Generating Useful Conformal Mappings: Applications to Transonic Aerodynamics", Int. J. Num. Meth. in Engr., 12, 1978, pp. 1651-1657.
58. Eiseman, P., "A Coordinate System for a Viscous Transonic Cascade Analysis", J. Comp. Phys., 26, 1978, pp. 307-338.
59. Eiseman, P., "Three-Dimensional Coordinates About Wings", AIAA Paper No. 79-1461, 1979.
60. Eiseman, P., "Geometric Methods in Computational Fluid Dynamics", ICASE Report No. 80-11, NASA Langley Research Center, 1980.
61. Shamroth, S., Gribeling, H., and McDonald, H., "A Navier-Stokes Solution for Laminar and Turbulent Flow Through a Cascade of Airfoils", AIAA Paper No. 80-1426, 1980.
62. Eiseman, P., McDonald, H., and Briley, W., "A Method for Computing 3-D Viscous Diffuser Flows", United Aircraft Research Lab Report R75-911737-1, 1975.
63. Eiseman, P., Levy, R., McDonald, H., and Briley, W., "Development of a 3-D Turbulent Duct Flow Analysis", NASA CR 3029, 1978.
64. Levy, R., McDonald, H., Briley, W., and Kreskovsky, J., "A 3-D Turbulent Compressible Subsonic Duct Flow Analysis for Use with Constructed Coordinates", AIAA Paper No. 80-1398, 1980.
65. Smith, R., and Weigel, B., "Analytical and Approximate Boundary-Fitted Coordinate Systems for Fluid Flow Simulation", AIAA Paper 80-0192, 1980.
66. Smith, R., Kudlinski, R., and Everton, E., "A Grid Spacing Control Technique for Algebraic Grid Generation Methods", AIAA Paper No. 82-0226, 1982.
67. Kowalski, E., "Boundary-Fitted Coordinate Systems for Arbitrary Computational Regions", NASA CP 2166, 1980, pp. 331-353.
68. Swanson, R., "Navier-Stokes Solutions for Nonaxisymmetric Nozzle Flows", AIAA Paper No. 81-1217, 1981.
69. Drummond, J., and Weidner, E., "Numerical Study of a Scramjet Engine Flow Field", AIAA Paper No. 81-0186, 1981.
70. Roberts, B., "Computational Meshes for Boundary Layer Problems", in Lecture Notes in Physics, Springer-Verlag, 1971, pp. 171-177.
71. Holst, T., "Numerical Simulation of Axisymmetric Boattail Fields with Plume Separators", AIAA Paper No. 77-224, 1977.
72. McNally, W., "Fortran Program for Generating a 2-D Orthogonal Mesh Between Two Arbitrary Boundaries", NASA TND-6766, 1972.
73. Graves, R., "Application of a Numerical Orthogonal Coordinate Generator to Axisymmetric Blunt Bodies", NASA TM 80131, 1979.
74. Steger, J., and Sorenson, R., "Use of Hyperbolic Partial Differential Equations to Generate Body Fitted Coordinates", NASA CP 2166, 1980, pp. 463-478.
75. Hirt, C., Nichols, B., and Romero, N., "SOLA-A Numerical Solution Algorithm for Transient Fluid Flows", Report LA-5852, Los Alamos Scientific Laboratory, 1975.
76. Hasen, G., "Navier-Stokes Solutions for Axisymmetric Nozzle", AIAA Paper 81-1474, 1981.
77. Perry, K., "Nonaxisymmetric Nozzle/Aftbody Flow Field Analysis", AFWAL TR 81-3046, 1981.
78. Kowalski, E., Perry, K., and Klees, G., "Numerical Simulation for the Design of a Supersonic Cruise Nozzle with Fluid Noise Shield", AIAA Paper No. 81-1218, 1981.
79. Reddy, R., and Thompson, J., "Numerical Solution of Incompressible Navier-Stokes Equations in the Integro-Differential Formulation Using Boundary-Fitted Coordinate Systems", AIAA Paper No. 77-650, AIAA 3rd Computational Fluid Dynamics Conference, 1977, pp. 178-188.

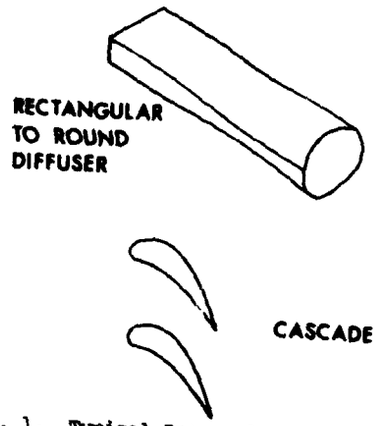


Fig. 1. Typical Internal Flow Configurations: Rectangular-to-Round Diffuser and Cascade

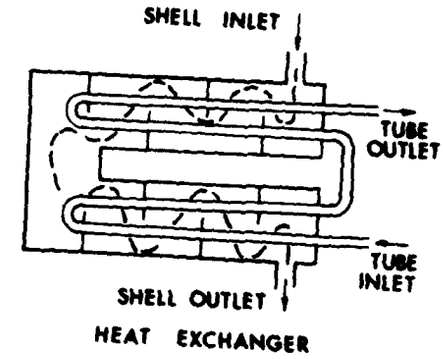


Fig. 2. Typical Internal Flow Configuration: Shell-and-Tube Heat Exchanger

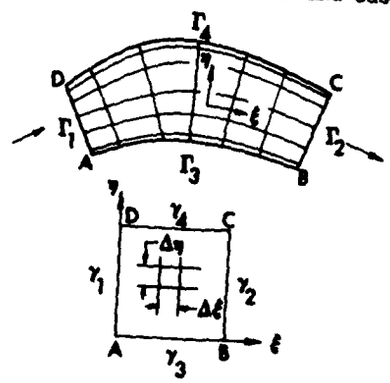


Fig. 3. Curvilinear Coordinate Transformation

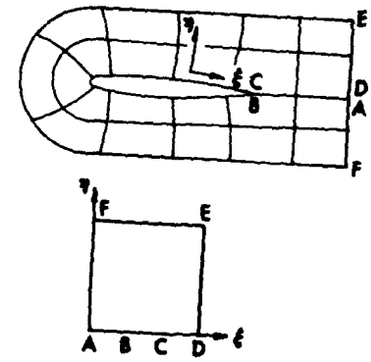


Fig. 4. "C"-type Grid for Cascade Flows

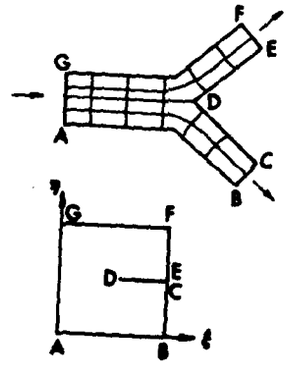


Fig. 5. Fluid Branching Circuit

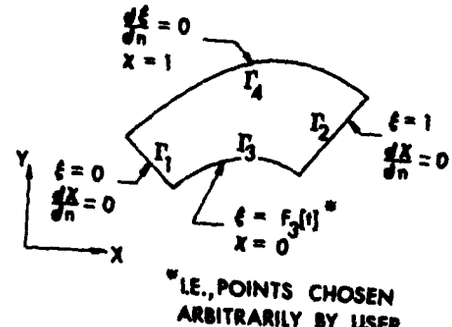


Fig. 6. Boundary Conditions for Intermediate Transformation in (X,Y) Plane

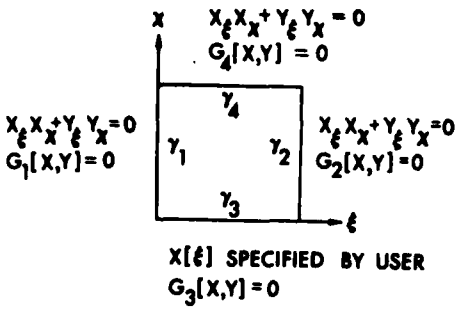


Fig. 7. Boundary Conditions for Intermediate Transformation in (ξ,χ) Plane

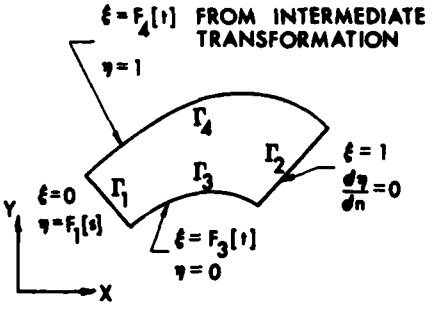


Fig. 8. Boundary Conditions for Final Transformation in (X,Y) Plane for Mode 1 Operation [for Mode 2 Operation, $\partial\eta/\partial n = 0$ on Γ_2 is replaced by $\eta = F_2(s)$]

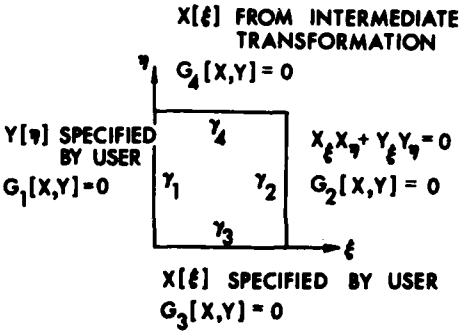


Fig. 9. Boundary Conditions for Final Transformation in (ξ,η) Plane for Mode 1 Operation [for Mode 2 Operation, $x_\xi x_\eta + y_\xi y_\eta = 0$ on γ_2 is replaced by $s = F_2(\eta)$]

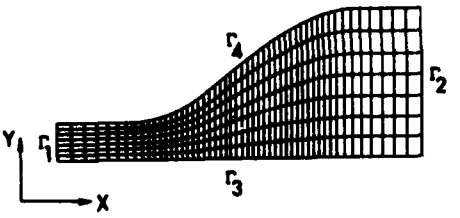


Fig. 10a. Initial Guess for Example 1

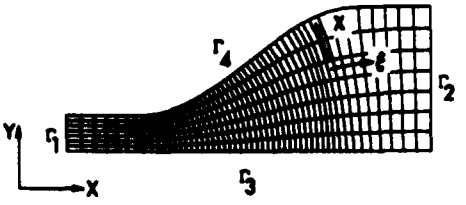


Fig. 10b. Intermediate Transformation for Example 1

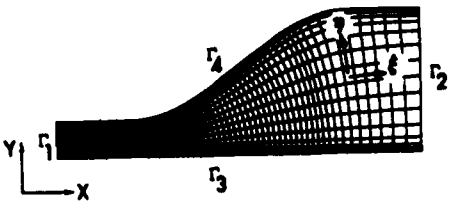


Fig. 10c. Final Transformation for Example 1

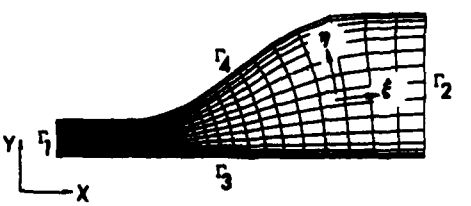


Figure 11. Final Transformation for Example 2

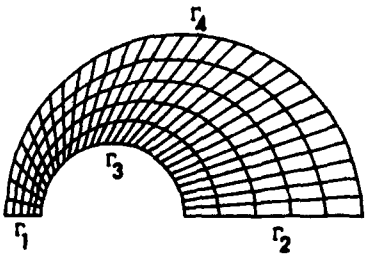


Fig. 12a. Initial Guess for Example 3

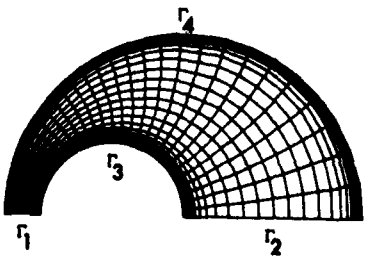


Fig. 12b. Final Transformation for Example 3

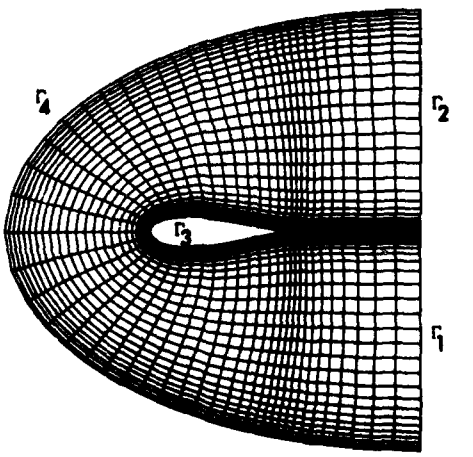


Fig. 13. Final Transformation for Example 4

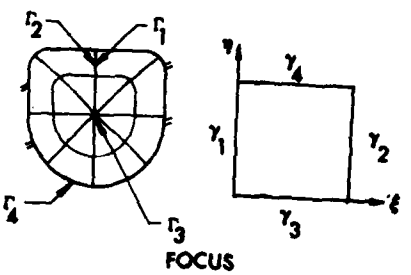


Fig. 14. Single Focus Technique

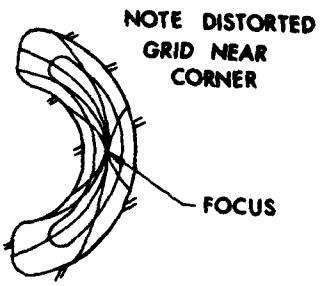


Fig. 15. Typical Problem Using Single Focus Technique

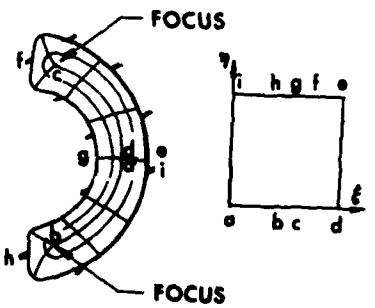


Fig. 16. Slit Technique

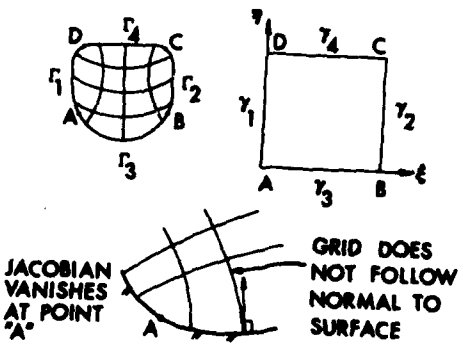


Fig. 17. Artificial Corner Technique

AD P000981

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

SOLUTION OF NONLINEAR WATER WAVE PROBLEMS USING BOUNDARY - FITTED COORDINATE SYSTEMS

Henry J. Haussling
David W. Taylor Naval Ship Research and Development Center,
Bethesda, Maryland 20084

INTRODUCTION

The mathematical study of wave motion in water with a free surface dates back at least as far as Lagrange around 1800. Since then, steady progress has been made in this field by many researchers through the application of many of the tools of mathematical physics. Two important approximate theories have evolved: the linear theory for small amplitude water waves, and the nonlinear theory for waves in shallow water. Some progress has even been made in the solution of the general nonlinear problem by the use of perturbation theory. However, the application of analytical mathematical techniques to these general problems has been hindered by the nonlinear difficulties, not the least of which is the fact that the flow domain is not known a priori but is part of the solution. The challenging nature of the nonlinear problems and even of the more complex of the linearized problems has led to considerable effort to apply the power of modern computers to facilitate their solution. Computers have been of tremendous help in obtaining meaningful numbers from fairly complex closed form solutions for simplified problems. Perhaps more importantly computers are also being used to directly attack the unsimplified equations through the use of numerical techniques which have been and are being developed specifically for this purpose. Among these techniques are finite-difference, finite-element, and panel methods.

Many of the impressive numerical solutions of water wave problems have been obtained with the so-called marker-and-cell finite-difference technique (Ref.[1]). In such an endeavor, physical space is divided into a number of rectangular cells. The cells are divided into three sets the elements of which can change with time; (1) those which are completely filled with fluid, (2) those through which the free surface passes and hence which are only partially filled with fluid, and (3) those which are outside the flow domain. Through appropriate application of the governing equations and boundary conditions the movement of the free surface through such a cell structure can be followed. However, tracking a boundary movement through such a fixed grid

can be quite difficult and error prone. Provision must be made for finite-difference stars of changing arbitrary shape, and boundary-conditions involving derivatives normal to and tangential to the surface can be difficult to apply. Even so, many such solutions have been and are being obtained and they have contributed considerably to the progress of computational fluid dynamics.

The difficulties and deficiencies of the marker-and-cell techniques have motivated many improvements and alternative approaches for the numerical solution of free surface problems. Boundary-fitted curvilinear coordinate systems offer one such alternative. Mapping of the physical region to a fixed computational region enables the calculation to be carried out on a time-independent grid even when the flow is time-dependent. Since the boundaries are always constant-coordinate surfaces, some of the possible difficulties in applying boundary conditions at curved boundaries are eliminated. In recent years these coordinate systems have been applied to water wave problems, and the possibilities and limitations of these applications are still being investigated. the authors

At the David W. Taylor Naval Ship R&D Center we have been investigating the application of numerically generated curvilinear coordinate systems to ship wave problems. These are water wave problems with a body present, and this investigation is part of a broader effort over the past few years to develop and test numerical techniques for use in ship performance prediction. Since no one numerical method is best for all problems, and a wide variety of promising methods is available, the work has involved the investigation of the capabilities of various methods. their

This paper presents an account of our experience in applying numerically-generated coordinate systems to water wave problems and a brief discussion of closely-related efforts underway elsewhere. Application to free-surface potential flow is highlighted since most of our work so far has focussed on such inviscid flows. Similar coordinate generation techniques could be used for viscous free-surface problems.

MATHEMATICAL THEORY OF FREE-SURFACE POTENTIAL FLOW

For the purposes of this review we limit the discussion essentially to flows of water for which the effects of viscosity (friction) and compressibility can be neglected. For detailed descriptions of such inviscid free-

surface flows see Ref. [2] or Ref. [3]. With the neglect of viscosity, we can consider the Euler equations

$$d\vec{v}/dt = -\frac{1}{\rho}\nabla p - g\hat{k} \tag{1}$$

where $\vec{v}$ is the velocity vector describing the flow, t is time, ρ is density (assumed constant), p is pressure, g is the gravitational acceleration, and $\hat{k}$ is a unit vector parallel to the local direction of the Earth's gravitational field. Eq. (1) is merely Newton's law of conservation of momentum, where the left side is the acceleration of a fluid particle and the right side consists of the forces acting on such a particle - the internal pressure force and the external gravitational force. Note that shear stresses are absent here since they are among the neglected frictional effects. To Eq. (1) must be added the equation of continuity

$$\nabla \cdot \vec{v} = 0 \tag{2}$$

to form a system which is sufficient, with initial and boundary conditions, to solve for the velocity field $\vec{v}$ and the pressure p. Eq. (2) represents conservation of mass and can be derived from the fact that for an incompressible, constant density fluid the flux of mass is zero across the boundary surface of any region in which liquid cannot be created or destroyed. Expressing this flux condition in integral form and making use of Gauss' divergence theorem leads to Eq. (2).

A measure of the local rotationality of a fluid is the vorticity $\nabla \times \vec{v}$. A flow for which

$$\nabla \times \vec{v} = 0 \tag{3}$$

is said to be irrotational. It can be shown, using Eq. (1), that a flow of an inviscid fluid which is irrotational at one time remains irrotational for all time. Since many flows are for practical purposes nearly irrotational throughout much of the flow field, and since the assumption of irrotational flow results in major simplifications, such an assumption is often made and it is convenient to do so for the purposes of this paper.

The vanishing of $\nabla \times \vec{v}$ ensures that the function

$$\phi(x,y,z) = \int_{f_o}^{f(x,y,z)} \vec{v} \cdot d\vec{s} \tag{4}$$

is well-defined where the line integral is over any path from a fixed point f_o to an arbitrary point f. Since Eq. (4) implies that

$$\vec{v} = \nabla\phi \tag{5}$$

the assumption of irrotationality implies that the velocity field can be represented as the gradient of a potential.

Eq. (2) and (5) yield

$$\nabla^2\phi = 0 \tag{6}$$

and thus the potential satisfies the well-known Laplace equation. The pressure does not appear in this equation for the velocity potential. It can be computed from the velocity field by using the momentum equations (Eq. (1)) which can be rewritten, using Eq. (5) with some manipulation, as

$$\partial\phi/\partial t = - (\nabla\phi\cdot\nabla\phi) /2 - p/\rho - gy, \tag{7}$$

where y is the vertical coordinate. Eq. (7) is known as Bernoulli's law.

A free surface S is a boundary surface of unprescribed shape which separates the fluid from another medium. For water wave problems the pressure on the surface is the constant atmospheric pressure which can be taken to be zero since only pressure gradients affect the flow field. According to Eq. (7) we then have the dynamical free surface boundary condition

$$\partial\phi/\partial t = - (\nabla\phi\cdot\nabla\phi) /2 - gy \qquad \text{on S} \tag{8}$$

Since the location of the surface is unknown, an additional boundary condition is needed. It is the kinematic condition which results from a basic assumption of continuum mechanics and states merely that any fluid particle on the free surface remains on the free surface. The kinematic condition has more than one mathematical representation that is useful for computations.

One useful form can be derived by considering the surface S as represented by an equation $f(x,y,z;t) = 0$. The fact that the derivative of f following a fluid particle on the surface is zero means that

$$df/dt = \partial f/\partial t + \vec{v}\cdot\nabla f = 0 \tag{9}$$

holds on S. If the surface can be represented by specifying the vertical coordinate as a single-valued function of the horizontal coordinates

$$y = Y(x,z;t) \tag{10}$$

it follows that $f = y - Y(x,z;t)$ and hence from Eq. (9) that

$$Y_t = -\phi_x Y_x - \phi_z Y_z + \phi_y \qquad \text{on S} \tag{11a}$$

Eq. (11a) is a commonly used form of the kinematic free surface boundary condition but is useful only for simple surfaces for which Eq. (10) retains its single-valued property. For more general surfaces it can be helpful to use

$$d\vec{x}/dt = \nabla\phi \qquad \text{on S,} \tag{11b}$$

where $\vec{x}$ is the position vector of a fluid particle on the surface. Eq. (11b) is just the mathematical representation of the fact that the surface moves with the local fluid velocity.

The free surface boundary conditions must be supplemented by appropriate conditions at other boundaries. The most commonly encountered condition on solid boundaries such as ship hulls or rigid walls is

$$\nabla\phi \cdot \hat{n} = V_n, \tag{12}$$

where $\hat{n}$ is a unit vector normal to the boundary and V_n is the normal component of the velocity of the boundary. Eq. (12) states that fluid cannot pass through such a solid boundary. Far away from disturbances the fluid is often

assumed to be at rest. That is

$$\nabla\phi = 0 \tag{13}$$

To summarize, the solution of free surface potential flow problems usually amounts to solving Eq. (6) subject to the free-surface boundary conditions of Eq. (8) and Eq. (11a) or Eq. (11b), along with other appropriate conditions such as those of Eqs. (12) and (13).

COORDINATE SYSTEM TRANSFORMATION

The coordinate system construction techniques that we have used to date at DTNSRDC are based on the Poisson generating equations

$$\begin{aligned} \nabla^2\xi &= P \\ \nabla^2\eta &= Q \\ \nabla^2\zeta &= R \end{aligned} \tag{14}$$

which map the flow region in (x,y,z)-space to a computational region in (ξ,η,ζ)-space according to a transformation of the form

$$\begin{aligned} \xi &= \xi(x,y,z;t) \\ \eta &= \eta(x,y,z;t) \\ \zeta &= \zeta(x,y,z;t) \end{aligned} \tag{15}$$

In Eq. (14) the source terms, P, Q, and R, can be used to control the coordinate system. The time dependence of the transformation is a result of the boundary movement. These methods are extensions of the methods first successfully employed by Thompson (Ref. [4]). As usual the techniques are capable of treating arbitrary flow regions. This capability has been enhanced by the introduction of computational regions which are constructed from an arbitrary number of rectangles (boxes in 3-d) which are fit together so as to yield a suitable set of coordinates. Details on constructing such computational regions are given in another paper at this conference (Ref. [5]) which describes a recently-developed program for generating such arbitrary

transformations in two dimensions.

For computational purposes the generating system of Eq. (15) is transformed to the computational space by interchanging the dependent and independent variables. For the two-dimensional case (which for simplicity is discussed hereafter) the transformed generating equations are

$$\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} + J^2(Px_\xi + Qx_\eta) = 0 \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} + J^2(Py_\xi + Qy_\eta) = 0 \end{aligned} \tag{16}$$

where

$$\begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 & \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2 & J &= x_\xi y_\eta - x_\eta y_\xi \end{aligned} \tag{17}$$

and subscripts indicate differentiation. The transformation can then be determined by solving Eq. (16) subject to appropriate boundary conditions. These conditions usually specify the dependence of x and y on ξ and η along the boundaries. Alternative conditions, such as the requirement that coordinate lines be orthogonal in physical space at a specified boundary, can also be applied.

The governing equations of the fluid flow must also be transformed to the (ξ,η) coordinate system. Eq. (6) is rewritten as

$$\alpha\phi_{\xi\xi} - 2\beta\phi_{\xi\eta} + \gamma\phi_{\eta\eta} + \sigma\phi_\eta + \tau\phi_\xi = 0 \tag{18}$$

where

$$\begin{aligned} \sigma &= J^2 Q \\ \tau &= J^2 P \end{aligned} \tag{19}$$

The 2-d version of Eq. (8) transforms to

$$\begin{aligned} (\phi_t)_{\xi,\eta = \text{constant}} &= x_t(\phi_\xi y_\eta - \phi_\eta y_\xi)/J - y_t(\phi_\xi x_\eta - \phi_\eta x_\xi)/J \\ &\quad - [(y_\eta\phi_\xi - y_\xi\phi_\eta)^2 + (x_\xi\phi_\eta - x_\eta\phi_\xi)^2]/(2J^2) \\ &\quad - gy \qquad \eta = \eta_\ell \end{aligned} \tag{20}$$

where η_ℓ is the coordinate line that coincides with the water surface. The

kinematic free-surface condition of Eq. (11a) can be written in the form

$$(Y_t)_{x = \text{constant}} = (y_\eta \phi_\xi - y_\xi \phi_\eta) Y_\xi / (J x_\xi) + (x_\xi \phi_\eta - x_\eta \phi_\xi)/J \text{ at } \eta = \eta_\ell \tag{21a}$$

while Eq. (11b) takes the form

$$\begin{aligned} dx/dt &= (x_\xi \phi_\eta - x_\eta \phi_\xi)/J \\ dy/dt &= (y_\eta \phi_\xi - y_\xi \phi_\eta)/J \end{aligned} \qquad \text{on } \eta = \eta_\ell \tag{21b}$$

Other boundary conditions such as Eqs. (12) and (13) can be applied by using the expression for the derivative of ϕ normal to any boundary

$$\nabla\phi \cdot \hat{n} = [\phi_\xi (g' y_\eta + x_\eta) - \phi_\eta (g' y_\xi + x_\xi)] / [J(1 + g')^2]^{1/2} \tag{22}$$

where $g' = dy/dx$ is the slope of the boundary.

Since the coordinate system is needed for the solution of the flow problem and yet the coordinates cannot be computed before the free-surface location is known, the determination of the flow and of the coordinate system must be carried out simultaneously. This can be done with a fairly straightforward marching procedure. If the solution is known at any time, the free-surface position can be advanced in time according to Eq. (21a) or Eq. (21b). Similarly, the potential on the surface can be advanced according to Eq. (20). A coordinate system can be computed at the advanced time according to Eq. (16) and the new surface geometry. Finally the complete flow field can be computed by solving Eq. (18) with the aid of the new coordinates and with already-computed surface values of ϕ as a boundary condition. Of course, such a time integration is impossible to carry out exactly, so numerical techniques must be used.

NUMERICAL TECHNIQUES

This section presents the basic details of the numerical aspects of both the coordinate generation and flow problem solution techniques that we have been using at DTNSRDC. The domain of integration in the (ξ,η) - plane is covered by a uniform network of points $(\xi_i = i, \eta_j = j)$, with $i = 1, \ldots, i_\ell$ and

$j = 1,\ldots, j_\ell$. If the computational region is a rectangle, all these points are included in the region. However, if the shape of the region is more complicated, many of the points may not be involved in the computations. The differential equations are replaced by difference equations involving the values of the variables at the grid points of interest.

To compute the coordinates, Eq. (16) is replaced by central difference formulae yielding

$$\begin{Bmatrix} x_{i,j} \\ y_{i,j} \end{Bmatrix} = \left[(\alpha_{i,j}+J_{i,j}^2 P_{i,j}/2) \begin{Bmatrix} x_{i+1,j} \\ y_{i+1,j} \end{Bmatrix} \right.$$
$$+(\alpha_{i,j}-J_{i,j}^2 P_{i,j}/2) \begin{Bmatrix} x_{i-1,j} \\ y_{i-1,j} \end{Bmatrix}$$
$$+(\gamma_{i,j}+J_{i,j}^2 Q_{i,j}/2) \begin{Bmatrix} x_{i,j+1} \\ y_{i,j+1} \end{Bmatrix} \qquad (23)$$
$$+(\gamma_{i,j}-J_{i,j}^2 Q_{i,j}/2) \begin{Bmatrix} x_{i,j-1} \\ y_{i,j-1} \end{Bmatrix}$$
$$-(\beta_{i,j}/2) \begin{Bmatrix} x_{i+1,j+1} + x_{i-1,j-1} - x_{i-1,j+1} \\ y_{i+1,j+1} + y_{i-1,j-1} - y_{i-1,j+1} \end{Bmatrix}$$
$$\left. \begin{matrix} -x_{i+1,j-1} \\ -y_{i+1,j-1} \end{matrix} \right\rbrace]/2(\alpha_{i,j} + \gamma_{i,j})$$

where $\alpha_{i,j}$, $\beta_{i,j}$, $\gamma_{i,j}$ and $J_{i,j}$ are central difference approximations to Eq. (17). Source terms $P_{i,j}$ and $Q_{i,j}$ are specified to yield suitable grids for the particular applications.

For convenience, Eq. (23) is solved by successive overrelaxaton (SOR). Optimum overrelaxation factors are determined through numerical experimentation. It has been found (Ref. [6]) that on vector processing computers mesh generation equations such as these can be solved very efficiently with SOR by sweeping the mesh in the so-called "red-black" or "four-color" manner.

Eq. (18) is replaced by the difference equation

$$\begin{aligned}\phi_{i,j} = [&(\alpha_{i,j}+\tau_{i,j}/2)\phi_{i+1,j} \\ &+(\alpha_{i,j}-\tau_{i,j}/2)\phi_{i-1,j} + (\gamma_{i,j}+\sigma_{i,j}/2)\phi_{i,j+1} \\ &+(\gamma_{i,j}-\sigma_{i,j}/2)\phi_{i,j-1} - (\beta_{i,j}/2)(\phi_{i+1,j+1} \\ &-\phi_{i-1,j-1}-\phi_{i+1,j-1}-\phi_{i-1,j+1})]/2(\alpha_{i,j}+\gamma_{i,j})\end{aligned} \tag{24}$$

which is also solved with SOR.

Euler's modified method of time differencing is used to replace the free-surface boundary conditions of Eq. (20) and Eq. (21a) or Eq.(21b) with difference equations of the form

$$f_i^{n+1} = f_i^n + \Delta t(F_i^{n+1} + F_i^n)/2 \tag{25}$$

where the superscripts refer to time levels, Δt is the time increment, f_i represents a quantity to be advanced in time at the i^{th} free-surface grid point, and F_i is a finite-difference approximation to the right-hand side of the corresponding free-surface condition.

The implicit system of Eq. (25) is solved iteratively for the surface values of ϕ and the surface location at the advanced time level. The iterative solution of these equations is combined with the iterative solution for the velocity potential and the coordinate system. Thus the new coordinate system and flow field are computed simultaneously. The iterative procedure at each time advancement can be started with initial estimates of surface location, potential, and coordinates obtained by extrapolation from previous time levels. The iterations are halted when the precentage change from iteration to iteration of one or more of the dependent variables is less than some specified small number.

Numerical solutions of nonlinear free-surface problems are highly susceptible to numerical instability, leading to oscillation from grid point to grid point at the surface. Such instability seems to be a standard feature of the numerical solution of nonlinear water wave problems, independent of the numerical technique. The standard method for eliminating the instability is to apply a filtering technique described by Shapiro in Ref. [7]. After

each time advancement, new free surface quantities are computed according to smoothing formulae, such as

$$f_i' = [-f_{i+2}-f_{i-2}+4(f_{i+1}+f_{i-1})+10f_i]/16 \qquad (26)$$

Such filtering has been found necessary by many researchers (e.g., Ref. [8] and Ref. [9]). However, in a recent private communication, S. Orszag described some work on his generalized vortex method in which he found that, if numerical errors are carefully minimized, filtering is not necessary.

SELECTED RESULTS

Some of the earliest work on the use of boundary-fitted coordinates for the solution of nonlinear water wave problems was carried out by Thompson and his colleagues at Mississippi State University (e.g. Ref. [10]). Previously they had carried out viscous and potential flow calculations for flows about arbitrary bodies in an unbounded fluid using a transformation to a rectangular computational region. The far-field boundary conditions were applied on one side of the rectangle. To handle the free-surface flows part of this boundary was converted to a free boundary. Impressive results were obtained for viscous free surface flows about hydrofoils, although the coordinate system, which was well-suited for infinite fluids, had obvious deficiencies for the water wave problems. To overcome some of the deficiencies, a T-shaped computational region was also used. Although this yielded even better results, further improvements were necessary.

For the free-surface potential flow problems being studied at the time at DTNSRDC an H-shaped computational region was used as shown in Figs. 1 - 5, taken from Ref. [11]. The resulting coordinates have the good properties that coordinate lines wrap around the body, yet away from the body the lines conform well to the free surface and bottom boundary. In addition, the number of points on the body is independent of the number used to define the water surface. Two problem areas in which the transformation is singular and six-sided cells are generated can be noted in Fig. 3. Source terms P and Q are used in the Poisson generating system of Eq. (16) to improve the resolution in these regions. Grid points are not placed at the singular points. We have seen no indication of fundamental problems from such singularities, although further study seems warranted on whether a numerical

solution will converge to the exact solution in such a situation as the mesh is refined. We now know how to construct a better coordinate system which has less of a resolution problem from these singularities. A suitable mesh can be constructed without any attraction near the singularities. Such a system is currently being used for a submerged hydrofoil and is described in Ref. [5].

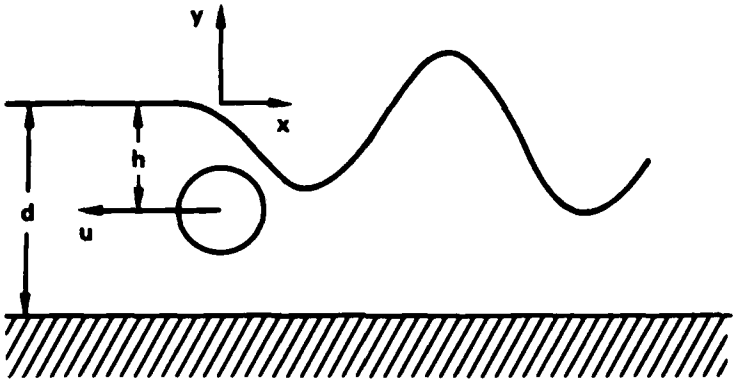


Fig. 1. Translating circular cylinder below a free water surface.

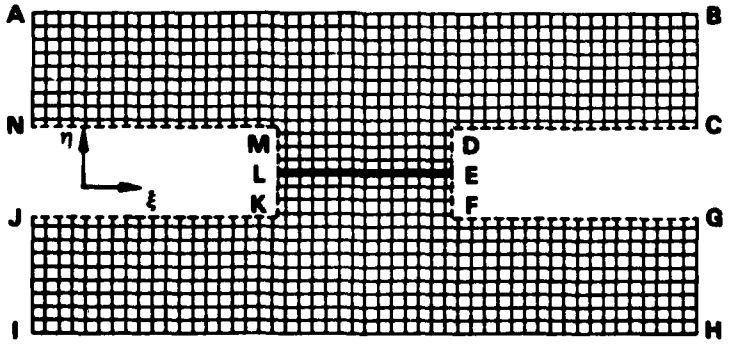


Fig. 2. Computational region for translating circular cylinder.

The coordinate system of Fig. 3 was applied to translating and oscillating circular cylinders. Typical results are presented in Figs. 4 and 5

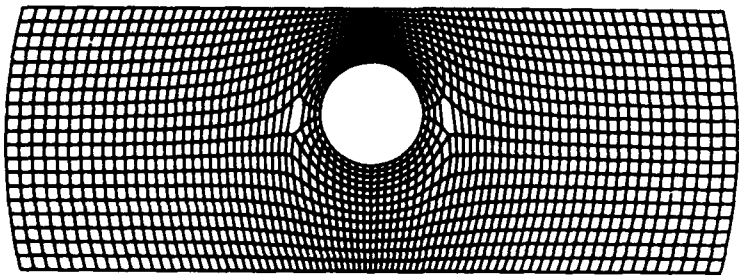


Fig. 3. Coordinate system for submerged cylinder at rest.

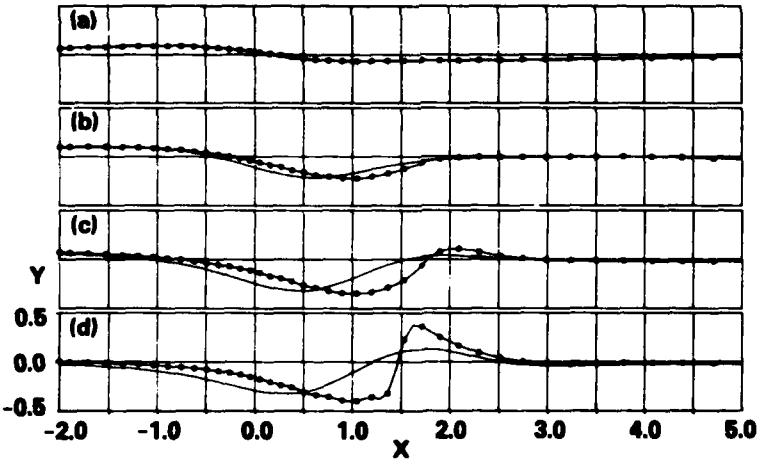


Fig. 4. Computed evolution of the water surface after an abrupt acceleration to the left of a submerged circular cylinder (at x=0).
•••••••••• nonlinear, ———— steady-state linear.

(Ref. [11]). The cylinder is centered at x = o, and at time t = o is accelerated leftward abruptly to a constant speed. Calculations are carried out in a moving reference frame. The nonlinear numerical results are compared with a computed linear solution in Fig. 4. It is apparent that nonlinear effects are significant. A wave grows behind the cylinder to a point at which the computations cease to converge. In reality the wave would break, but the numerics are unable to continue farther into the breaking process. A few shortcomings are apparent in Fig. 5. As the wave grows, the distance

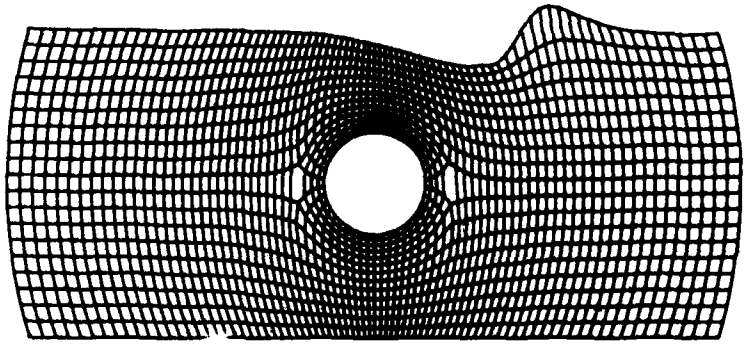


Fig. 5. Coordinate system for submerged translating cylinder showing wave development.

between grid points on the free surface increases, because the x-coordinate of each grid point is held constant while the y-coordinate is computed according to Eq. (21a). Also the grid lines below the surface do not move far enough upward with the surface to maintain adequate resolution. Further problems with the singularities in the transformation can be seen in Fig. 6, where results for the oscillating cylinder are presented. These calculations were carried out as the cylinder oscillated horisontally in a fixed reference frame. The source term attraction which was chosen for the cylinder at rest

is not suitable for maintaining good grid point distribution at all times. Time-dependent attraction such as that used in Ref. [6] probably would have helped.

Many improvements were obviously called for in the coordinate systems that were used up to that time. Fortunately such improvements were regularly appearing at DTNSRDC and elsewhere. About that time the interesting results of Longuet-Higgins and Cokelet (Ref. [8]) came to our attention. Using an

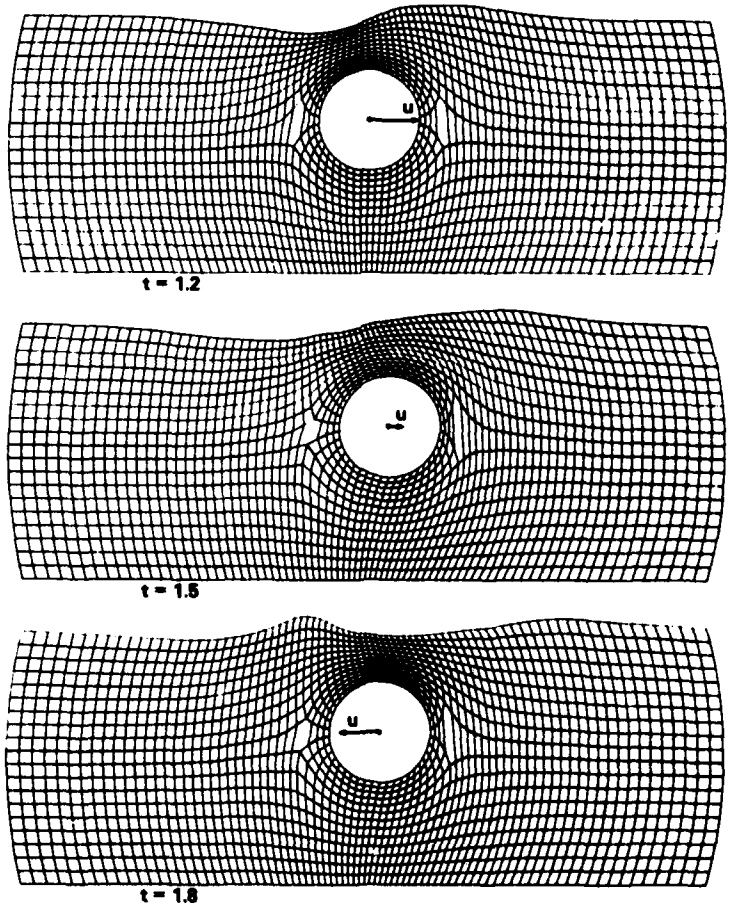


Fig. 6. Time dependent coordinate system for swaying cylinder.

exact mapping they studied the problem of the breaking of a large-amplitude periodic wave. They were eventually able to continue the calculations until a jet of water shooting outward and downward from the breaking wave tip just contacted the water surface. We applied the boundary-fitted coordinate system finite-difference method to the same problem to see how much of their result we could reproduce. A rectangular computational region was used. Previously unpublished results are presented in Fig. 7, which is a superposition of computed wave profiles at several times. The calculations accurately reproduced the previous results until about the time that the water surface became vertical; then the computations broke down. It can be seen that at that time the coordinate system was becoming quite distorted, with large deviations from orthogonality. Even so, these results represent a considerable advancement over the submerged circular cylinder effort and were achieved by applying improved techniques. A fixed frame of reference was used and the

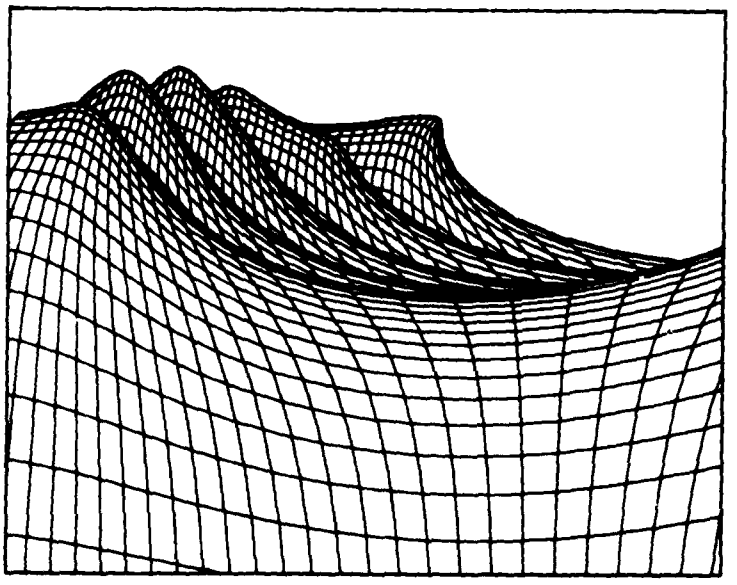


Fig. 7. Computed two-dimensional breaking wave development.

surface grid points moved with the fluid acccording to Eq. (21b). This led naturally to a clustering of grid points near the wave peak. A source term used by Plant (Ref. [13]) and similar to those derived by Thompson (Ref. [14]) and Ghia, et al (Ref. [15]) was used to attract horizontal grid lines toward the free surface. The effect of the source term is to maintain boundary coordinate spacing throughout the fluid and thus overcome the tendency of Laplace-generated systems to yield uniform spacing in the interior independent of boundary distribution. Grid lines are concentrated near the free-surface at the upstream and downstream boundaries, and the source term attraction serves to maintain this concentration beneath the entire free surface. However, even with these improvements, calculations could not be continued beyond the time at which the surface became vertical. In fact, it is not yet clear whether the boundary-fitted coordinates can efficiently handle the curling over of such a wave. A possible technique for carrying the computation farther was proposed by Ghia et al in Ref. [16]. It was suggested that, as the wave grows, the computational region be modified to that pictured in Fig. 8 to yield a coordinate system fitted to the breaking wave as

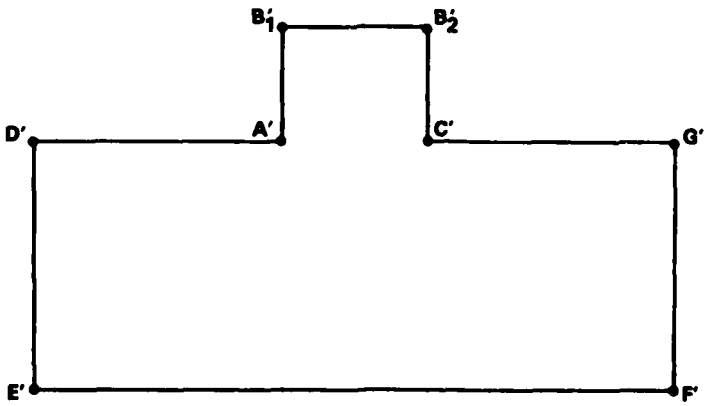


Fig. 8. Computational domain for breaking wave proposed by U. Ghia and K. Ghia.

displayed in Fig. 9.

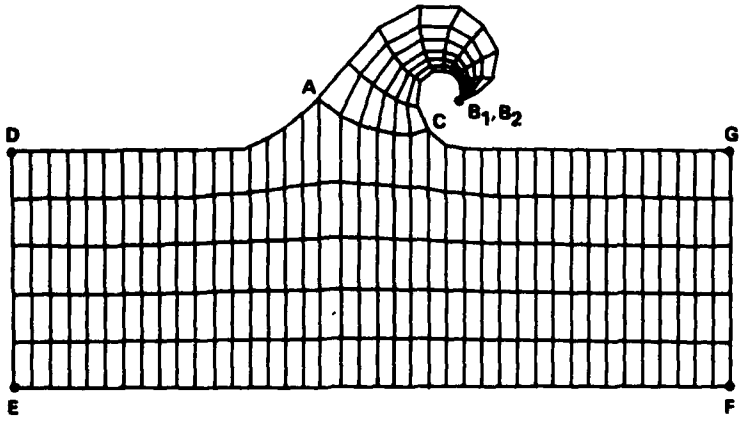


Fig. 9. Coordinate system for breaking wave.

After the breaking periodic wave calculation was completed, we chose to apply the improved techniques to breaking waves associated with a ship-like body in the water surface. One of the simplest such problems is for an infinitely wide (2-d), infinitely long (no bow) transom stern. In Ref. [17] results were reported on the generation of breaking waves behind such a body after acceleration from rest as displayed schematically in Fig. 10. Results

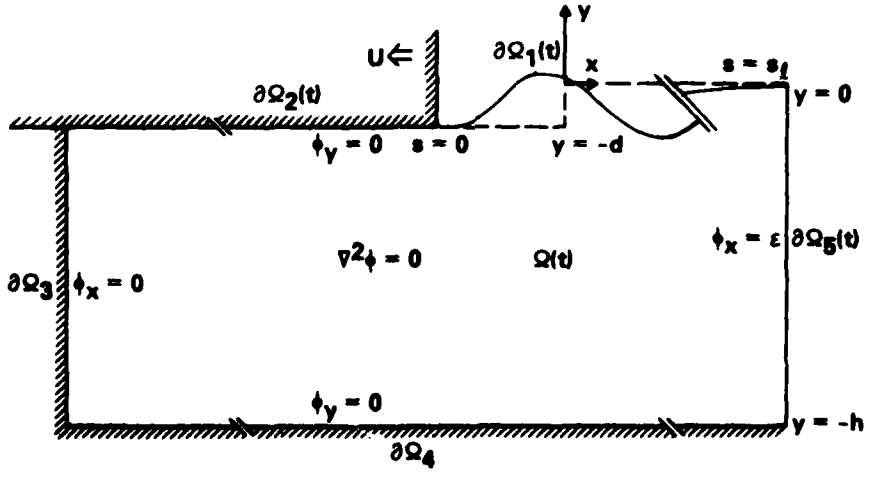


Fig. 10. Two-dimensional transom stern moving over water surface.

are presented in Fig. 11. Important details about transom stern flows were

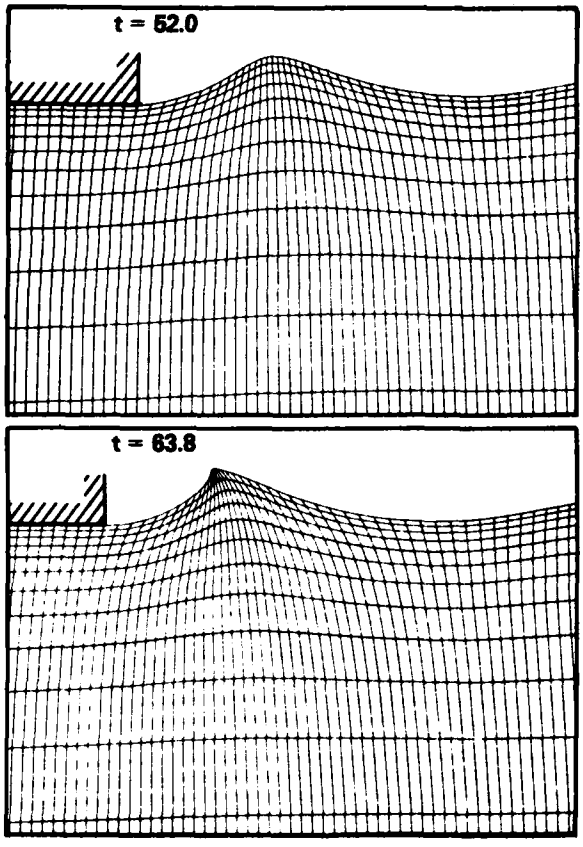


Fig. 11. Computed breaking wave behind transom stern moving to the left.

revealed. Again, a simple rectangular computational region was used. However, the upper boundary was divided into two parts, the free surface and the hull bottom. Since grid points moved with the water, many points underwent a transition from the hull boundary condition to the free-surface conditions. Such a transition posed no problem for the techniques.

Coordinate systems have also been developed for 2-d ship hull cross

sections, and an example is presented in Fig. 12. Preliminary ship motion calculations have been carried out in which these hulls are oscillating

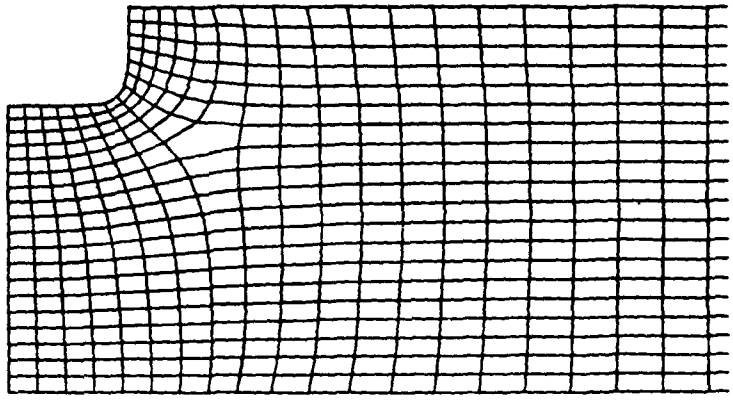


Fig. 12. Coordinate system for ship hull cross section.

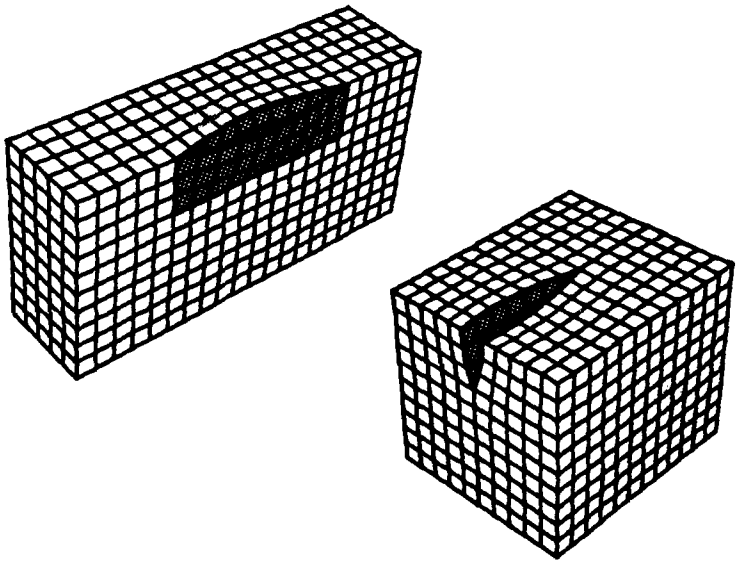


Fig. 13. Coordinate system for a thin ship.

vertically (heaving). But in such an application we run into a difficulty which must be overcome before truly accurate and stable nonlinear numerical calculations can be carried out. The potential flow field contains singularities at the intersections of the water surface with rigid boundaries. These singularities have yet to be understood both mathematically and numerically.

The coordinate generation techniques for free surfaces have also been extended to three dimensions (Ref. [18]). They have been applied to a few simple hull geometries such as those shown in Figs. 13 and 14. How best

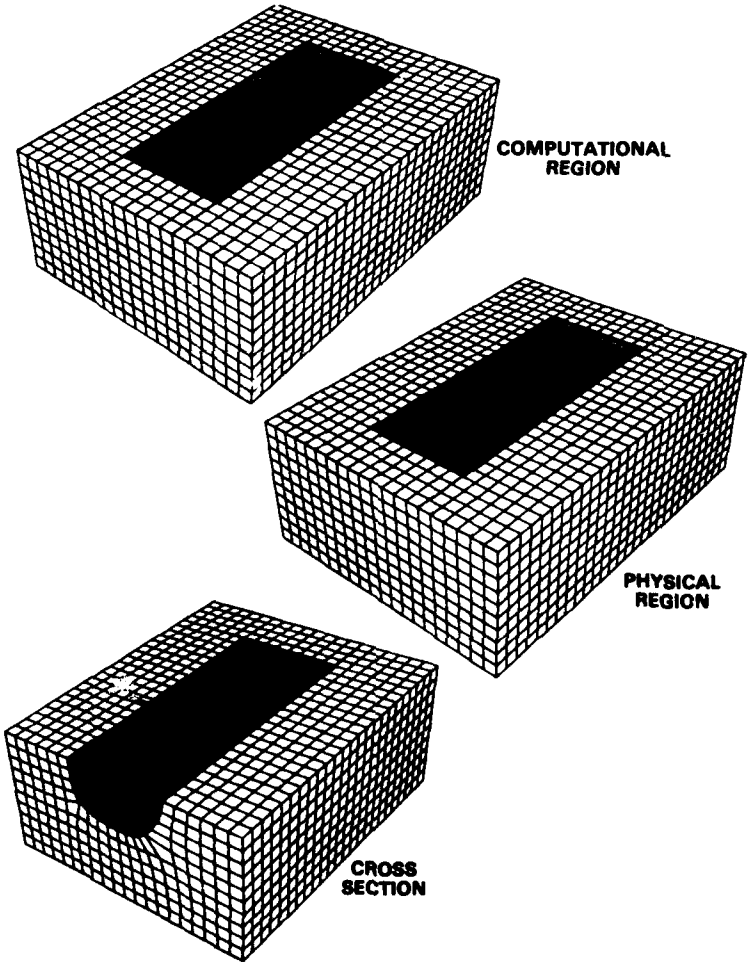


Fig. 14. Coordinate system for a finite circular cylinder.

to apply the methods to 3-d, in particular, the kinds of computational regions to use for complicated ship wave geometries, is not clear. So far we have carried out only linearized 3-d flow calculations for which numerical coordinate system generation is not necessary. However, efforts are currently underway to study the nonlinear 3-d equivalent of the transom stern flow previously described in this section.

CONCLUSION

Over the past few years numerical coordinate system generation technology has developed at a rapid pace. These techniques are a powerful addition to the tools available for solving problems in physics. At the David W. Taylor Naval Ship R&D Center contributions have been made to this development and the methods are being applied to ship wave problems. Arbitrary computational regions facilitate the treatment of arbitrary time dependent physical flow domains. Such an approach resulted in considerable progress in the solution of previously unsolved nonlinear free surface problems. Efforts are continuing on the application to ever more difficult and realistic configurations.

Considerable further research on coordinate system generation is warranted. Automated coordinate control is one important objective. It remains to be seen how these techniques can be carried toward the ultimate goal of numerically modeling the complete flow field about a ship as it moves in the water. It is possible that other developing numerical methods will prove to be more useful for certain problems. Even if this occurs, the boundary-fitted coordinate system approach will still be available as a powerful, easy to apply tool that has made a significant contribution to our understanding of nonlinear free-surface flows and of ways to model them numerically.

REFERENCES

1. B.D. Nichols & C.W. Hirt, "Calculating Three-Dimensional Free Surface Flows in the Vicinity of Submerged and Exposed Structures," J. Comp. Phys. Vol. 12, p. 234, 1973.
2. J.J. Stoker, Water Waves, Interscience Publishers, New York, 1957.
3. J.V. Wehausen & E.V. Laitone, "Surface Waves", Handbuch der Physik, Vol. 9, Springer Verlag, Berlin, 1960.
4. J.F. Thompson, F.C. Thames, and C.W. Mastin, "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies," J. Comp. Phys. Vol. 15, p. 299, 1974.
5. R.M. Coleman, "Generation of Boundary-Fitted Coordinate Systems

Using Segmented Computational Regions" Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Use in the Numerical Solution of Partial Differential Equations, Nashville, Tenn., Apr. 13-16, 1982.

6. H.J. Haussling, "Boundary-Fitted Coordinates for Accurate Numerical Solution of Multi-Body Flow Problems," J. Comp. Phys. Vol. 30, p. 107, 1979.
7. R. Shapiro, "Linear Filtering," Mathematics of Computation, Vol.29, p. 1094, 1975.
8. M.S. Longuet-Higgins & E.D. Cokelet, "The Deformation of Steep Surface Waves. I. A Numerical Method of Computation," Proc. Roy. Soc. Lond. A, Vol. 350, p. 1, 1976.
9. G.R. Baker et al, "Applications of a Vortex Method to Nonlinear Free Surface Flows," Third Int. Conf. on Numerical Ship Hydrodynamics, Paris, June 16-19, 1981.
10. S.P. Shanks & J.F. Thompson, "Numerical Solution of the Navier-Stokes Equations for 2-D Hydrofoils in or Below a Free Surface," Second Int. Conf. on Numerical Ship Hydrodynamics, Berkeley, Calif., Sept. 1977.
11. H.J. Haussling & R.M. Coleman, "Finite-Difference Computations Using Boundary-Fitted Coordinates for Free-Surface Potential Flows Generated by Submerged Bodies," Second Int. Conf. on Numerical Ship Hydrodynamics, Berkeley, Calif, Sept. 1977.
12. H.J. Haussling & R.M. Coleman, "Nonlinear Water Waves Generated by an Accelerated Circular Cylinder," J. Fluid Mech., Vol. 92, p.767, 1979.
13. F.J. Plant, "An Exact Velocity Potential Solution of Steady, Compressible Flow over Arbitrary Two-Dimensional and Axisymmetric Bodies in Simply Connected Fields," Air Force Flight Dynamics Laboratory Report AFFDL-TR-77-116, 1977.
14. J.F. Thompson & S.P. Shanks, "Numerical Solution of the Navier-Stokes Equations for 2-D Surface Hydrofoils," Miss. State Univ. Report MSSU-EIRS-ASE-77-4, 1977.
15. U. Ghia, et al, "An Optimization Study for Generating Surface-Oriented Coordinates for Arbitrary Bodies in High-Re Flow," Air Force Flight Dynamics Laboratory Report AFFDL-TR-77-117, 1977.
16. U. Ghia et al, "Analysis of a Breaking Free-Surface Wave Using Boundary-Fitted Coordinates for Regions Including Reentrant Boundaries," Third Int. Conf. on Numerical Ship Hydrodynamics, Paris, June 16-19, 1981.
17. R.M. Coleman & H.J. Haussling, "Nonlinear Waves Behind an Accelerated Transom Stern," Third Int. Conf. on Numerical Ship Hdrodynamics, Paris, June 16-19, 1981.
18. R.M. Coleman "Numerically Generated Boundary-Fitted Coordinate Systems for Arbitrary Three-Dimensional Regions," David W. Taylor Naval Ship R&D Center Report DTNSRDC-78/085, 1978.

AD P000982

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

NUMERICAL MODELING OF ESTUARINE HYDRODYNAMICS ON A BOUNDARY-FITTED COORDINATE SYSTEM

BILLY H. JOHNSON
U. S. Army Engineers Waterways Experiment Station, P. O. Box 631,
Vicksburg, Mississippi.

INTRODUCTION

Physical models of rivers, estuaries, and bays have been utilized for years to model the hydrodynamics of those bodies of water. However, physical models are more limited in addressing water quality problems which have received more emphasis in recent years. With the advent of digital computers over the past 20-30 years it has become feasible to develop numerical models to determine the chemical and biological changes that occur if the physical character of an existing estuary is changed, e.g. by dredging, diking, or permanently changing the freshwater inflow. Solving such problems requires predictive models to compute the distribution of currents and circulation in the water body. These results are then used to predict the distribution of quality parameters.

Although all hydrodynamic models are similar in nature, estuarine models pose additional problems of scope and complexity. Estuarine hydrodynamics are determined by the interaction of the tides from the ocean and the influx of fresh water from the rivers, modified by the influence of the semienclosed physiography containing islands, embayments, etc. In addition, the hydrodynamics can be further altered by gravitational circulations arising from density gradients.

There have been many numerical estuarine hydrodynamic models developed since the pioneering work of Leendertse[1] in the mid 1960's. The majority of these models employ the method of finite differences to solve the equations of motion on a rectangular grid with fixed-grid spacing. As a result, irregular boundaries are represented in a "stair-stepped" fashion. In addition with a fixed-grid spacing, if the spacing is made small enough to represent islands, embayments and/or channels in sufficient detail for currents induced by these features to be computed, grids that are so large as to make computations uneconomical can result.

Over the past few years attempts have been made at more accurate handling of irregular boundary and/or internal features. The development

PREVIOUS PAGE IS BLANK

of hydrodynamic models employing the method of finite elements is one approach that has been taken.[2] Paralleling the development of finite element hydrodynamic models has been the implementation of techniques within the finite difference framework to address the problem of handling irregular boundaries and/or variable grid spacing. Examples include Wanstrath's[3] conformal mapping of storm surge areas, Waldrop's, et al.,[4] use of an orthogonal curvilinear grid in river studies, Rodenhuis', et al.[5], implementation of embedded rectangular grids, and the use of grid stretching by Waldrop, et al.,[6] and Butler.[7] Over the past 3 years work has been conducted at the Waterways Experiment Station on the development of a two-dimensional vertically averaged estuary model[8] employing Thompson's work on boundary-fitted coordinates.[9] Thompson's method generates curvilinear coordinates as the solution of two elliptic partial differential equations with Dirichlet boundary conditions. No restrictions are placed on the irregularity of the boundaries, and fields containing multiple bodies or branches can be handled as easily as simple geometries.

The above efforts have all been concerned with the more accurate modeling of features in the horizontal plane, Lick's[10] three-dimensional model employs a transformation of the vertical dimension that allows for the water depth to be mapped between the values of 0 and 1. Such a transformation allows for bottom topographies to be more accurately handled.

Although both three-dimensional and laterally averaged hydrodynamic models have been developed, the discussion to follow concerns only the numerical modeling of flow in estuaries that are classified as well mixed, i.e., the vertically averaged approximation is appropriate. The governing equations are first developed in cartesian form before being transformed for solution on a curvilinear grid. An example utilizing the vertically averaged hydrodynamic model (VAHM) serves to demonstrate the practical aspects of generating an estuarine flow field on a boundary-fitted coordinate system.

VERTICALLY AVERAGED ESTUARINE HYDRODYNAMICS

Classification. Many physical variations of natural estuaries exist and as a result various kinds of estuaries are often described in the literature. Terms such as tidal, stratified, well mixed, etc. are often

found. Glenne[11] has attempted to classify estuaries according to the effects on mixing by factors such as, depth, length, cross-sectional area, tides, friction, and advective flow. Figure 1 is a schematic sketch of this classification system. As indicated, the estuaries of concern here are well mixed with large openings to the sea. A vertical averaging of the governing equations is appropriate for such systems.

Governing Equations in Cartesian Coordinates. The Navier Stokes equations express the conservation of mass and momentum of a flow field and are the basic governing equations for the solution of any fluid dynamics problem. Written in tensor notation these equations are

$$\underline{\text{Continuity}}: \quad \frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{1}$$

$$\underline{\text{Momentum}}: \quad \frac{\partial \rho u_i}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = \frac{-\partial P}{\partial x_i} + \rho g_i - 2\varepsilon_{ijk}\Omega\, \rho u_k + \frac{\partial T_{ij}}{\partial x_j} \tag{2}$$

All symbols used in the text are defined in Appendix A.

If the effect of density gradients are considered, a conservation of mass equation must also be written for the salinity.

$$\underline{\text{Salinity}}: \quad \frac{\partial s}{\partial t} + \frac{\partial (s u_i)}{\partial x_i} = \frac{\partial\, D_{ij} \dfrac{\partial s}{\partial x_j}}{\partial x_i} \tag{3}$$

This equation states that the salinity can change as a result of advection by the flow field and molecular diffusion.

Since the salinity is coupled to the flow equations through its influence on the density, one additional equation remains to be written in order to close the system. An equation of state expressing the density as a function of the temperature and salinity must be employed.

$$\underline{\text{Equation of State}}: \quad \rho = \rho(T,s) \tag{4}$$

With the closure of the system, there exists six equations to be solved for the six unknowns; density -ρ , three velocity components -u,v,w , pressure -p , and salinity -s.

Time Averaging for Turbulent Flows. The above equations written with molecular values of viscosity and diffusivity are only applicable in a practical sense to laminar flow fields. However, most fluids in motion exhibit random irregular fluctuations and are referred to as turbulent flows.

Following Reynolds, the approach normally taken to make the equations applicable to turbulent flows is to assume that the dependent variables are composed of an average time-varying component plus a small randomly varying component about the average value. Integration of the equations over a time increment Δt then produces the same form as the previous equations, but now written with the time-averaged components as the dependent variables, plus additional terms involving products of the randomly varying components. Boussinesq's concept of eddy viscosity (diffusivity) is then used to relate these terms to the mean flow field.

Depth Averaging for Nearly Horizontal Flow. A solution of the resulting set of equations for turbulent flow constitutes a fully-time varying, three-dimensional model of the flow and salinity fields. However, when modeling nearly horizontal flow in relatively shallow and well-mixed water bodies the usual approach is to employ a spatial averaging to yield a two-dimensional model.

The basic assumption in the spatial averaging of the time-averaged three-dimensional equations is that the dependent variables can be represented by an average value over the depth plus some small random deviation. An integration over the water depth then yields a set of equations with the time-averaged and depth-averaged components of the flow and salinity as dependent variables plus additional terms involving the random deviations. As in the time-averaged case, these terms are normally approximated through the use of eddy coefficients. These are referred to as eddy dispersion coefficients by Holley[12] to distinguish them from the turbulent eddy diffusion coefficients arising from the time averaging. An excellent discussion of both time and space averaging can be found in Reference 13.

The resulting vertically averaged equations are presented below. It should be noted that the Boussinesq approximation has been made, i.e., the effect of density variations is neglected in all terms except those multiplied by the acceleration of gravity.

$$\underline{\text{Continuity}}: \frac{\partial \phi}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (vh)}{\partial y} = 0 \tag{5}$$

$$\underline{\text{x-momentum}}: \frac{\partial (hu)}{\partial t} + \frac{\partial (hu^2)}{\partial x} + \frac{\partial (huv)}{\partial y} = -\frac{h}{\rho}\frac{\partial P}{\partial x} + \frac{\partial\, hD_{xx}\frac{\partial u}{\partial x}}{\partial x} + \frac{\partial\, hD_{xy}\frac{\partial u}{\partial y}}{\partial y} + \tau_{s_x} - \tau_{B_x} + fhv \tag{6}$$

$$\underline{\text{y-momentum}}: \frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial (hv^2)}{\partial y} = -\frac{h}{\rho}\frac{\partial P}{\partial y} + \frac{\partial\, hD_{yx}\frac{\partial v}{\partial x}}{\partial x} + \frac{\partial\, hD_{yy}\frac{\partial v}{\partial y}}{\partial y} + \tau_{s_y} - \tau_{B_y} - fhu \tag{7}$$

$$\underline{\text{Salinity}}: \frac{\partial (hs)}{\partial t} + \frac{\partial hus}{\partial x} + \frac{\partial (hvs)}{\partial y} = \frac{\partial\, hE_x\frac{\partial s}{\partial x}}{\partial x} + \frac{\partial\, hE_y\frac{\partial s}{\partial y}}{\partial y} \tag{8}$$

The equation of state relating the water density to the salinity and water temperature (assumed constant) has been taken from Leendertse[14] and is given as

$$\rho(s,T) = 1000\ \frac{P0}{AL} + ALO * P0 \tag{9}$$

where

$$AL = 1779.5 + 11.25T - 0.0745T^2 - (3.80 + 0.01T)s$$

$$ALO = 0.6980$$

$$P0 = 5890.0 + 38T - 0.375T^2 + 3s$$

In the above equations the surface wind shear is

$$\tau_{s_x} = \frac{W_c}{\rho_o} \rho_a v_w^2 \cos \alpha \tag{10}$$

$$\tau_{s_y} = \frac{W_c}{\rho_o} \rho_a v_w^2 \sin \alpha \tag{11}$$

and the bottom shear is

$$\tau_{B_x} = gu \sqrt{u^2 + v^2} \Big/ C^2 \tag{12}$$

$$\tau_{B_y} = gv \sqrt{u^2 + v^2} \Big/ C^2 \tag{13}$$

The coriolis parameter, f, is computed from

$$f = 2\omega_e \sin \lambda \tag{14}$$

where ω_e = earth's angular velocity and λ is the angle of latitude of the center of the area being modeled.

In order to finalize the above system of equations it remains to couple the salinity computations with those of the flow field. Assuming that the pressure is hydrostatic, it can be shown (see Reference 8) that the horizontal pressure gradient terms can be written as

$$h \frac{\partial P}{\partial x} = h \frac{\partial P_a}{\partial x} + h\rho g \frac{\partial \phi}{\partial x} + \frac{1}{2} h^2 g \frac{\partial \rho}{\partial x} \tag{15}$$

and

$$h \frac{\partial P}{\partial y} = h \frac{\partial P_a}{\partial y} + h\rho g \frac{\partial \phi}{\partial y} + \frac{1}{2} h^2 g \frac{\partial \rho}{\partial x} \tag{16}$$

Substituting Equations 10-16 into Equations 5-8 would then yield the final form of the governing equations in cartesian coordinates.

Required Input Data. The vertically-averaged equations of mass continuity, momentum, and salt balance, along with the equation of state, represent a set of five equations which can be solved simultaneously for

the five dependent variables u, v, ϕ, s and ρ provided the necessary numerical values of the initial and boundary conditions along with a description of the estuary are available. The required inputs for numerical solution of the equations can be summarized as follows:

(a) The physical dimensions of the estuary;
(b) The temporal and spatial distribution of atmospheric pressure and of surface wind over the computational period;
(c) The fresh water inflow to the estuary as a function of time;
(d) An average water temperature in the estuary;
(e) Values of the Chezy resistance coefficient as a function of position in the estuary;
(f) Values of the various eddy diffusivity and viscosity coefficients as a function of position in the estuary and of time;
(g) Values of the surface elevation ϕ and the vertically-averaged salinity s as a function of time along the seaward boundary;
(h) An initial set of values of the dependent variables at all positions in the estuary.

The primary focus of the remainder of the paper concerns the first, i.e., techniques for accurately handling the estuary geometry in the horizontal directions.

HORIZONTAL GRIDS EMPLOYED IN HYDRODYNAMIC MODELS

The earliest developers of vertically averaged estuarine hydrodynamic models all solved the basic mass continuity and momentum equations on a rectangular grid with uniform grid mesh. The computation of the salinity field and its coupling with the flow was not attempted. Although the majority of estuarine models solve the governing equations on such rectangular grids, there have been attempts at developing models that handle irregular boundaries in a more accurate fashion while retaining the method of finite differences for solution.

Conformal Curvilinear Grids. Wanstrath[3] developed a vertically averaged numerical model for computing storm surges at coastlines through the use of conformal mapping. A spatial region of prototype space is conformally mapped into a rectangle in a mathematical image plane. To provide an evenly spaced computing grid an independent stretching of the coordinates is then performed.

Orthogonal Curvilinear Grids. Although the Waldrop, et al.[4] hydrodynamic model was developed to anlayze hot water discharges from power

plants rather than for application in an estuarine environment, it deserves mentioning due to its use of an orthogonal curvilinear grid which is generated algebraically. As with the grids generated by Wanstrath through conformal mapping, these grids lack generality in that boundary point selection and grid spacing is not arbitrary. In addition, interior bodies are not allowed in either the Wanstrath or the Waldrop models.

Stretched Rectangular Grids. In addition to Wanstrath, others such as Waldrop, et al.[6] and Butler[7] have employed the use of independent stretching of the horizontal coordinates to improve grid spacing. For example, the stretching transformation in Waldrop's buoyant plume computations is

$$X = \frac{1}{c_1} \tan^{-1} \frac{x}{c_2}$$

$$Y = \frac{1}{c_3} \tan^{-1} \frac{y}{c_4} \qquad (17)$$

thus if $c_1 = \pi/2$, then at $x = \infty$, $X = 1.0$. Even increments of ΔX thus produce a close spacing of grid points near the plume entrance, where the best resolution is desired, and yet extend the region of computation far from the origin so that boundary conditions can be more easily specified. Other researchers have employed different stretching functions, e.g., Butler employs an exponential form of stretching.

Boundary-Fitted Coordinate Systems. Through coordinate transformations, irregular boundaries and variable grid spacing can be more accurately handled while still making use of the simplicity of finite differences to obtain solutions. An extremely general coordinate transformation results from a method developed by Thompson[9] which generates curvilinear coordinates as the solution of two elliptic partial differential equations with Dirichlet boundary conditions, one coordinate being specified as constant on the boundaries, and a distribution of the other specified along the boundaries. No restrictions are placed on the irregularity of the boundaries, and fields containing multiple bodies or branches can be handled as easily as simple geometries. Regardless of

the shape and number of bodies and regardless of the spacing of coordinate lines, all numerical computations, both to generate the coordinate system and to subsequently solve the fluid flow equations, are done on a rectangular grid with square mesh.

Since the boundary-fitted coordinate system has a coordinate line coincident with all boundaries, all boundary conditions may be expressed at grid points, and normal derivatives may be represented using only finite differences between grid points on coordinate lines. No interpolation is needed, even though the coordinate system is not orthogonal at the boundary.

Since Thompson has discussed the basic development of boundary-fitted coordinates earlier in these proceedings, only a few high points will be presented here. A logical choice of the elliptic generating system is Poisson's equation. Thus for a domain such as illustrated in Figure 2 the basic problem is to solve

$$\begin{aligned} \xi_{xx} + \xi_{yy} &= P \\ \eta_{xx} + \eta_{yy} &= Q \end{aligned} \tag{18}$$

with boundary conditions

$$\begin{aligned} &\xi = \xi_1(x,y)\ ,\ \eta = \text{const on } \Gamma_1\ ;\ \xi = \xi_3(x,y)\ ,\ \eta = \text{const on } \Gamma_3 \\ &\xi = \xi_5(x,y)\ ,\ \eta = \text{const on } \Gamma_5\ ;\ \xi = \xi_7(x,y)\ ,\ \eta = \text{const on } \Gamma_7 \\ &\eta = \eta_2(x,y)\ ,\ \xi = \text{const on } \Gamma_2\ ;\ \eta = \eta_4(x,y)\ ,\ \xi = \text{const on } \Gamma_4 \\ &\eta = \eta_6(x,y)\ ,\ \xi = \text{const on } \Gamma_6\ ;\ \eta = \eta_8(x,y)\ ,\ \xi = \text{const on } \Gamma_8 \end{aligned} \tag{19}$$

The functions P and Q may be chosen to cause the coordinate lines to concentrate as desired. The form of these functions incorporated by Thompson, based upon much computer experimentation, is that of decaying exponentials.

Since all numerical computations are to be performed in the rectangular transformed plane, it is necessary to interchange the dependent and independent variables in Equation 18. Using the expressions for ξ_{xx}, ξ_{yy}, η_{xx}, and η_{yy} that have previously been presented by Thompson, Equation 18 becomes

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} - \gamma x_{\eta\eta} + J^2(Px_{\xi} + Qx_{\eta}) = 0$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} + J^2(Py_{\xi} + Qy_{\eta}) = 0 \tag{20}$$

where

$$\alpha = x_{\eta}^2 + y_{\eta}^2$$

$$\beta = x_{\xi}x_{\eta} + y_{\xi}y_{\eta} \tag{21}$$

$$\gamma = x_{\xi}^2 + y_{\xi}^2$$

J = Jacobian of the transformation = $x_{\xi}y_{\eta} - x_{\eta}y_{\xi}$

with transformed boundary conditions from Equation 19.

Although the new system of equations is more complex than the original system, the boundary conditions are specified on straight boundaries and the coordinate spacing in the transformed plane is uniform. Computationally, these advantages outweigh disadvantages resulting from the extra complexity of the equations to be solved.

The rectangular transformed grid is set up to be the size desired for a particular problem. Since the values of ξ and η are meaningless in the transformed plane, the η lines are assumed to run from 1 to the number of η lines desired in the physical plane. Likewise, the ξ lines are numbered 1 to the number specified on the boundaries of the physical plane. The grid spacing in both the ξ and η directions of

the transformed plane is taken as unity. Second order central difference expressions are used in Thompson's coordinate generation code, TOMCAT,[15] to approximate all derivatives in Equations 20 and 21. The resulting set of nonlinear difference equations, two for each point, are solved in TOMCAT by Accelerated Gauss-Seidel iteration.

The same procedure may be extended to regions that are more than doubly connected, i.e. have more than two closed boundaries, or equivalently, more than one body within a single outer body. A river reach containing more than one island is an example. As will be illustrated later, the basic data required to generate a boundary-fitted coordinate system are the physical coordinates of points on the boundaries. The computing cost is trivial.

As a final note, both conformal and orthogonal coordinate systems are special cases of boundary-fitted coordinates as generated from elliptic systems. Additional discussion is provided by Thompson, et al.[16]

TRANSFORMATION OF VERTICALLY AVERAGED EQUATIONS

Boundary-fitted coordinate systems generated from an elliptic system provide extremely general grids for computing flows in estuaries. As previously discussed, all computations are to be made in a transformed rectangular plane. Therefore, Equations 5-9 must be transformed such that (ξ , η) are the independent variables. To accomplish the transformation, the following expressions are utilized

$$f_x = \frac{1}{J}\left[\left(fy_\eta\right)_\xi - \left(fy_\xi\right)_\eta\right]$$

$$f_y = \frac{1}{J}\left[-\left(fx_\eta\right)_\xi + \left(fx_\xi\right)_\eta\right] \qquad (22)$$

It should be noted that these expressions are written in a fully conservative form which should result in a more accurate solution in highly irregular coordinate systems.

Applying the above expressions, with the assumption that the coordinate system is time invariant, the transformed set of equations to be

solved for the computation of vertically averaged flows is given below.

<u>Continuity</u>: $$\frac{\partial \phi}{\partial t} + \frac{1}{J}\left[\left(uhy_\eta - vhx_\eta\right)_\xi + \left(vhx_\xi - uhy_\xi\right)_\eta\right] = 0 \qquad (23)$$

<u>x-Momentum</u>: $$\frac{\partial (hu)}{\partial t} + \frac{1}{J}\left[\left(hu^2 y_\eta - huvx_\eta\right)_\xi + \left(huvx_\xi - hu^2 y_\xi\right)_\eta\right]$$

$$= -\frac{h}{J\rho_o}\left[\left(p_a y_\eta\right)_\xi - \left(p_a y_\xi\right)_\eta\right] - \frac{hg\rho}{J\rho_o}\left[\left(\phi y_\eta\right)_\xi - \left(\phi y_\xi\right)_\eta\right]$$

$$- \frac{h^2 g}{2J\rho_o}\left[\left(\rho y_\eta\right)_\xi - \left(\rho y_\xi\right)_\eta\right] + \frac{1}{J}\left\{\left[\frac{D_{xx}h}{J}\left(\left(uy_\eta\right)_\xi\right.\right.\right.$$

$$\left.\left.\left. - \left(uy_\xi\right)_\eta\right)y_\eta\right]_\xi - \left(\frac{D_{xx}h}{J}\left[\left(uy_\eta\right)_\xi - \left(uy_\xi\right)_\eta\right]y_\xi\right)_\eta\right\}$$

$$+ \frac{1}{J}\left\{-\left[\frac{hD_{xy}}{J}\left(-\left(ux_\eta\right)_\xi + \left(ux_\xi\right)_\eta\right)x_\eta\right]_\xi + \left[\frac{hD_{xy}}{J}\left(-\left(ux_\eta\right)_\xi\right.\right.\right.$$

$$\left.\left.\left. + \left(ux_\xi\right)_\eta\right)x_\xi\right]_\eta\right\} + \frac{W_c}{\rho_o}\rho_a v_w^2 \cos\alpha - gu\sqrt{u^2 + v^2}\Big/C^2 + fhv \qquad (24)$$

$$\underline{\text{y-Momentum}}: \quad \frac{\partial(hv)}{\partial t} + \frac{1}{J}\left[\left(hv^2 x_\xi - huvy_\xi\right)_\eta + \left(huvy_\eta - hv^2 x_\eta\right)_\xi\right]$$

$$= -\frac{h}{J\rho_o}\left[\left(p_a x_\xi\right)_\eta - \left(p_a x_\eta\right)_\xi\right] - \frac{hg\rho}{J\rho_o}\left[-\left(\phi x_\eta\right)_\xi\right.$$

$$\left.+\left(\phi x_\xi\right)_\eta\right] - \frac{h^2 g}{2J\rho_o}\left[-\left(\rho x_\eta\right)_\xi + \left(\rho x_\xi\right)_\eta\right]$$

$$+\frac{1}{J}\left\{\left[\frac{D_{yx}h}{J}\left(\left(vy_\eta\right)_\xi - \left(vy_\xi\right)_\eta\right)y_\eta\right]_\xi\right.$$

$$\left.-\left[\frac{D_{yx}h}{J}\left(\left(vy_\eta\right)_\xi - \left(vy_\xi\right)_\eta\right)y_\xi\right]_\eta\right\} + \frac{1}{J}\left\{-\left[\frac{D_{yy}h}{J}\left(-\left(vx_\eta\right)_\xi\right.\right.\right.$$

$$\left.\left.\left.+\left(vx_\xi\right)_\eta\right)x_\eta\right]_\xi + \left[\frac{D_{yy}h}{J}\left(-\left(vx_\eta\right)_\xi + \left(vx_\xi\right)_\eta\right)x_\xi\right]_\eta\right\}$$

$$+\frac{W_c}{\rho_o}\rho_a v_w^2 \sin\alpha - gv\sqrt{u^2+v^2}/C^2 - fhu \qquad (25)$$

$$\underline{\text{Salinity}}: \quad \frac{\partial(hs)}{\partial t} + \frac{1}{J}\left[\left(husy_\eta - hvsx_\eta\right)_\xi + \left(hvsx_\xi - husy_\xi\right)_\eta\right]$$

$$= \frac{1}{J}\left\{\left[\frac{hE_x}{J}\left(\left(sy_\eta\right)_\xi - \left(sy_\xi\right)_\eta\right)y_\eta\right]_\xi\right.$$

$$\left. - \left[\frac{hE_x}{J}\left(\left(sy_\eta\right)_\xi - \left(sy_\xi\right)_\eta\right)y_\xi\right]_\eta\right\} + \frac{1}{J}\left\{-\left[\frac{hE_y}{J}\right.\right.$$

$$\left.\left. - \left(\left(sx_\eta\right)_\xi + \left(sx_\xi\right)_\eta\right)x_\eta\right]_\xi + \left[\frac{hE_y}{J}\left(-\left(sx_\eta\right)_\xi + \left(sx_\xi\right)_\eta\right)x_\xi\right]_\eta\right\} \quad (26)$$

$$\underline{\text{Eq. of State}}: \quad \rho = \rho\left(s(\xi,\eta),T\right) \quad (27)$$

The above set of equations constitutes the set for which a numerical solution is sought on a rectangular grid with square grid spacing (e.g., $\Delta\xi = \Delta\eta = 1.0$). It remains, of course, to specify proper boundary conditions along the sides of the rectangular grid. Depending upon whether a side contains a solid boundary, an ocean, a river, or some combination of the three; a slip or no-slip condition, a tide curve, or a fresh water inflow might be prescribed over the computational period.

VAHM - A VERTICALLY AVERAGED HYDRODYNAMIC MODEL

To obtain a solution of the governing set of Equations 23-27, finite differences are employed. Finite difference schemes range from fully explicit to fully implicit, with a combination of an explicit-implicit

scheme being employed in some cases. Such a scheme has been implemented in the recent development of a vertically averaged model called VAHM.[8] Basically, the computational cycle consists of the following steps:

a. Solve for the water surface from the continuity equation in a fully implicit fashion.
b. Using the most recent values of the water surface elevations, solve for the u and v velocity components from the x and y momentum equations in an explicit fashion.
c. Solve for the salinity from the salt transport equation in an explicit fashion.
d. Compute the density from the equation of state, using the most recently computed salinity field.
e. Step forward in time and repeat the sequence.

Such a scheme has the stability criterion associated with the speed of a free surface gravity wave removed; although, diffusive criteria as well as the Courant condition associated with the speed of a water particle remain. However, these criteria are not normally overly restrictive.

Computational Grid. The grid in VAHM is rectangular with a grid spacing of $\Delta\xi = \Delta\eta = 1$. The u and v velocity components are computed at the corners of each cell with the water surface elevation, salinity, and density computed at the center of a cell. The (x,y) coordinates are specified at the corners, the center, and at the midpoint of each side of a cell. Diffusion coefficients are specified at the velocity points while water depths and Chezy coefficients are located at the cell center.

Differences. The basic difference equations solved in VAHM are developed using forward differences for all time derivatives. Centered differences are used in all spatial derivatives except in the convective terms. One has the option of requesting the use of either centered or a form of upwind differencing in the momentum convective terms while a fourth order flux corrected transport scheme as outlined by Zalesak[17] can be requested in the transport equation for salinity.

As previously noted, the water surface elevations are to be computed using an implicit scheme. Thus, in writing the difference form of the continuity equation all spatial derivatives are taken at the new time level (n+1). Equation 23 becomes

$$\frac{\phi_c^{n+1} - \phi_c^n}{\Delta t} + \frac{1}{J_c}\left[\left(uhy_\eta\right)_E^{n+1} - \left(uhy_\eta\right)_W^{n+1} - \left(vhx_\eta\right)_W^{n+1} + \left(vhx_\eta\right)_E^{n+1} \right.$$

$$\left. + \left(vhx_\xi\right)_N^{n+1} - \left(vhx_\xi\right)_S^{n+1} - \left(uhy_\xi\right)_N^{n+1} + \left(uhy_\xi\right)_S^{n+1} \right] = 0 \quad (28)$$

In the x and y momentum equations, all terms are taken at the old time step except the water surface slope term which is computed at the new time step. Therefore, the difference form of the x and y momentum equations becomes

$$\frac{\left(hu\right)_c^{n+1} - \left(hu\right)_c^n}{\Delta t} = - \left[\frac{hg\rho}{J\rho_o}\right]_c^n \left[\left(\phi y_\eta\right)_E^{n+1} - \left(\phi y_\eta\right)_W^{n+1} - \left(\phi y_\xi\right)_N^{n+1} + \left(\phi y_\xi\right)_S^{n+1} \right]$$

$$+ F_c^n \quad (29)$$

$$\frac{\left(hv\right)_c^{n+1} - \left(hv\right)_c^n}{\Delta t} = - \left[\frac{hg\rho}{J\rho_o}\right]_c^n \left[- \left(\phi x_\eta\right)_E^{n+1} + \left(\phi x_\eta\right)_W^{n+1} + \left(\phi x_\xi\right)_N^{n+1} - \left(\phi x_\xi\right)_S^{n+1} \right]$$

$$+ G_c^n \quad (30)$$

If one substitutes into Equation 28 for the values of $(uh)^{n+1}$ and $(vh)^{n+1}$ on the faces (from Equations 29 and 30 with appropriate

averaging) an equation containing only ϕ at the (n+1) time level results. This equation is then solved for ϕ_c by using the Accelerated Gauss-Seidel solution technique.

After the water surface elevation at the center of each cell is determined at the (n+1) time level, values of u^{n+1} and v^{n+1} at the cell corners are explicitly determined from Equations 29 and 30 using the new ϕ's at the (n+1) time level. It might be noted that the expressions for F and G are only computed once during each time step. These values are then used in first the iteration on the water surface and then in the velocity computations.

In the computations of ϕ, u, and v, the density is taken at the old time level. Its value at the new time level is computed from the equation of state relating the density to the salinity at the new time level. New salinities are computed from an explicit representation of the salt transport equation (Equation 26).

Boundary Conditions. Three types of boundaries are allowed in VAHM; walls, oceans, and rivers. Wall boundaries are characterized by the specification of a no-slip condition, i.e., the velocity components u and v are set to be zero at walls. Slip conditions would be implemented by setting the normal component of the velocity equal to zero with the tangential component computed from the expression for zero vorticity.

Ocean boundaries are characterized by the specification of a time varying water surface elevation at the boundary. Velocities on the ocean boundary are then computed from a simplified form of the momentum equation where the diffusive terms have been neglected. One-sided differences are used to replace derivatives that need points outside the field.

When the flow is directed into the computational field, the boundary condition on the salinity is prescribed as that of the ocean. However, when the flow is moving out of the computational field, the salinity at an ocean boundary is set to be equal to its value at the next point inside.

River boundaries are characterized by the specification of the velocity. The salinity is set to be zero and the water surface elevation at the center of a river boundary cell is computed as in any interior cell.

EXAMPLE PROBLEM

Generation of Boundary-Fitted Coordinates. The first step in the

computation of a flow field is the generation of the boundary-fitted coordinates. This is accomplished through a coordinate generation code, e.g. Thompson's TOMCAT. Output from the coordinate code is then saved on a file for subsequent use by VAHM. The basic input to the coordinate code is the specification of the (x,y) coordinates of the boundary points (see Figure 3). Although various degrees of coordinate control can be exercised, the boundary-fitted coordinates shown in Figure 4a that correspond to the boundary points selected in Figure 3 were computed using no control. Figure 4b illustrates the corresponding computational grid that is used in VAHM, where velocities are computed at the cell corners and salinities and water elevations at the cell center. However, it should be remembered that VAHM requires that the (x,y) coordinates be specified at not only the corners and center of a computational cell but also on the cell faces. With a geometrically conservative transformation, averaged values of the geometry derivatives should not be used. This is the reason for generating the coordinate system illustrated in Figure 4a.

The coordinate system plotted in Figure 4a was the third attempt at generating a useful grid system. Through the movement of boundary points and/or coordinate control one attempts to compute boundary-fitted coordinates such that the grid spacing does not vary rapidly and such that (ξ,η) lines never approach being parallel to each other. As discussed earlier in these proceedings by Thompson, the reason for this is because truncation errors related to the rate of change of the grid spacing and to the nonorthogonality of the grid are present.

Flow Computations. After the boundary-fitted coordinates are determined, the geometric elements in the transformation, e.g. x_ξ , y_η , J , etc., are computed and the flow model is then applied on the transformed rectangular grid, e.g. the grid in Figure 4c corresponds to the physical system in Figure 4b. VAHM has been applied on the example grid with a river at the top and an ocean on the bottom. A constant river velocity of 0.40 m/s was prescribed while the tide curve presented in Figure 5 was specified on the ocean boundary. Other input data are presented in Table 1.

TABLE 1

INPUT DATA TO VAHM

Variable	Units	Value
Δt	sec	600.0
D_{xx}	m^2/sec	10.0
D_{yy}	m^2/sec	10.0
E_x	m^2/sec	0.0
E_y	m^2/sec	0.0
C	$m^{\frac{1}{2}}/sec$	35.0
Initial depth	m	11.0
Initial velocity	m/sec	0.0
Ocean salinity	ppt	30.0

The effect of wind, atmospheric pressure variations and the Coriolis force were all neglected. Although the example problem is an hypothetical one, it is representative of an estuary such as the Delaware estuary.

Figure 6 presents "snap shots" of the computed flow field. With the flow field initialized to zero at a constant depth of 11.0 m, it can be seen that the influence of the incoming tide and the river meet after about 4 hours. After 6 hours, the ebb portion of the tide is experienced and the flow near the ocean boundary begins to reverse. Figure 7 is a plot of the time history of the water surface elevation at ($\xi = 6$, $\eta = 5$) while Figure 8 presents the time history of the salinity at the same point. Obviously if one was interested in using the computed flow field in subsequent water quality computations, the flow model would be run until the results over a tidal cycle had become repetitive.

SUMMARY AND FUTURE RESEARCH RECOMMENDATIONS

Numerical models for computing vertically averaged estuarine flow fields are required to provide input to water quality models. By employing the concept of boundary-fitted coordinates, irregular boundaries can be accurately modeled in either simple- or multiple-connected regions. Even though the numerical grid is a nonorthogonal curvilinear grid in the physical region being modeled, all numerical computations are carried out in a transformed rectangular grid with square grid spacing.

A feature of the particular model discussed is the solution technique employed to numerically solve the governing equations. A combination implicit-explicit finite difference scheme has been implemented to remove the speed of a free surface gravity wave from stability restrictions on the computational time step while still retaining some of the advantages of explicit schemes. With such a scheme, the water surface elevation is computed implicitly using the Accelerated Gauss-Seidel solution technique while the velocities and salinity are computed in an explicit fashion.

The model has been developed for general applications. Any number of river and/or ocean boundaries can be arbitrarily located on the transformed rectangular plane, as can the placement of islands in the interior of the computational field. Even though a great deal of generality does exist, a major restriction is that no flooding is allowed. In most estuaries, tidal flats are alternately flooded and dried on the flood and ebb portions of the tidal cycle. A flooding capability could be incorporated by allowing cells to flood and dry on a fixed curvilinear grid as is done in cartesian grid models. However, a more elegant treatment which deserves investigation would be to compute flows on a time-varying grid that moves with the flooding boundaries. On such a grid, the time derivative in the governing equations would be transformed as follows:

$$\left(\frac{\partial f}{\partial t}\right)_{xy} = \left(\frac{\partial f}{\partial t}\right)_{\xi,\eta} - \frac{1}{J}\left(f_\xi y_\eta - f_\eta y_\xi\right)\left(\frac{\partial x}{\partial t}\right)_{\xi,\eta}$$

$$\frac{1}{J}\left(f_\xi x_\eta - f_\eta x_\xi\right)\left(\frac{\partial x}{\partial t}\right)_{\xi,\eta} \qquad (31)$$

One disadvantage would be that interpolation would be required to generate time series plots at fixed physical locations.

An additional problem deserving of future research concerns the attraction of grid points along physical lines in the domain rather than coordinate lines that have been generated through only the specification of boundary points. In estuarine modeling studies, sinuous channels

within the estuary are often encountered. A means of forcing the computed coordinate system to follow such channels is needed. As discussed earlier by Thompson, some work has been done in this area.

ACKNOWLEDGMENTS

The development of VAHM was funded by Department of the Army Project 4A061101A91D, "In-House Laboratory Independent Research," sponsored by the Assistant Secretary of the Army. Permission to publish this paper was granted by the Chief of Engineers.

REFERENCES

1. Leendertse, J. J. (1967) "Aspects of a Computational Model for Long-Period Water Wave Propagation," RM-5294-PR, the Rand Corporation, Santa Monica, Calif.
2. Norton, W. R. and King, I. P. (Feb 1977) "Operating Instructions for Computer Program RMA-2 - A Two-Dimensional Finite Element Program for Problems in Horizontal Free Surface Hydromechanics," prepared for Association of Bay Area Governments Environmental Management Plan Bay Modeling Project, Resource Management Associates, Lafayette, Calif.
3. Wanstrath, J. J. (1977) "Nearshore Numerical Storm Surge and Tidal Simulation," TR H-77-17, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss.
4. Waldrop, W. R. and Tatom, F. B. (1976) "Analysis of the Thermal Effluent from the Gallatin Steam Plant During Low River Flows," Report No. 33-30, Tennessee Valley Authority.
5. Rodenhuis, G. S., Brink-Kjaer, O., Bertelsen, J. A. (Oct 1978) "A North Sea Model for Detailed Current and Water-Level Predictions," Journal of Petroleum Technology.
6. Waldrop, W. R., and Farmer, R. C. (20 Mar 1974) "Three-Dimensional Computation of Buoyant Plumes," Journal of Geophysical Research, Vol 79, No. 9.
7. Butler, H. L. (1980) "Evolution of a Numerical Model for Simulating Long-Period Wave Behavior in Ocean-Estuarine Systems," in Estuarine and Wetland Processes, Hamilton, P. and Macdonald, K. (eds.), Plenum press, New York, 1980.
8. Johnson, B. H. (1980) "VAHM - A Vertically Averaged Hydrodynamic Model Using Boundary-Fitted Coordinates," MP HL-80-3, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss.
9. See Thompson's paper in these proceedings.
10. Lick, Wilbert. (1976) "Numerical Models of Lake Currents," EPA-600/3-76-020, U. S. Environmental Protection Agency, Office of Research and Development, Environmental Research Laboratory, Duluth, Minn.
11. Glenne, Bard. (Feb 1976) "Classification System for Estuaries," Journal of the Waterways and Harbors Division, Proceedings, ASCE, No. WW1.
12. Holley, E. R. (1969) "Unified View of Diffusion and Dispersion," Journal of the Hydraulics Division, Proceedings, ASCE, Vol 95, No. HY2.
13. Ward, G. H. and Espey, W. H. (1971) "Estuarine Modeling: An Assessment," TRACOR, Inc., Austin, Tex., for the Water Quality Office, Environmental Protection Agency, 1971.

14. Leendertse, Jan J., et al. (Dec 1973) A Three-Dimensional Model for Estuaries and Coastal Seas: Vol 1, Principles of Computation, R-1417-OWRR, Rand Corporation, Santa Monica, California.
15. Thompson, Joe F., et al. (Jul 1977) "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, Vol 24, No. 3.
16. Thompson, J. F., Warsi, Zahir V. A., and Mastin, C. Wayne. "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," Journal of Computational Physics, to appear in mid-1982.
17. Zalesak, Steven, T. (1979) "Fully Multidimensional Flex-Corrected Transport Algorithms for Fluids," Journal of Computational Physics, Vol 31.

APPENDIX A: NOTATION

C	Chezy coefficient
C_1, C_2, C_3, C_4	Constants in stretching transformation
D_{ij}	Molecular diffusivity
D_{xx}, D_{yy}	Diagonal components of eddy viscosity tensor
D_{xy}, D_{yx}	Off diagonal components of eddy viscosity tensor
E_x, E_y	Components of eddy dispersion tensor
$E(t)$	Tidal elevations
F	Expression containing all terms in x-momentum equation except water surface slope
f	Coriolis parameter; general function
G	Expression containing all terms in y-momentum equation except water surface slope
g	Acceleration of gravity
h	Water depth
J	Jacobian of the transformation
P,Q	Coordinate control functions
P	Pressure
p_a	Atmospheric pressure
$Q(t)$	Discharge
s	Salinity
T	Temperature
Δt	Time step
u,v,w	Components of velocity
u_i, u_j, u_k	Tensor notation for velocity
v_w	Wind speed

W_c	Wind drag coefficient
X,Y	Stretching coordinates
x,y,z	Cartesian coordinates
ξ,η	Boundary-fitted coordinates
ρ	Water density
ρ_o	Reference water density
ρ_a	Density of air
ϕ	Water surface elevation
τ_{ij}	Stress tensor
ϵ_{ijk}	Cylic tensor
τ_{s_x},τ_{s_y}	Components of surface wind shear stress/ρ
τ_{B_x},τ_{B_y}	Components of bottom shear stress/ρ
α	Wind direction
λ	Latitude of center of modeled area
ω_e	Earth's angular velocity
Ω	Coriolis term

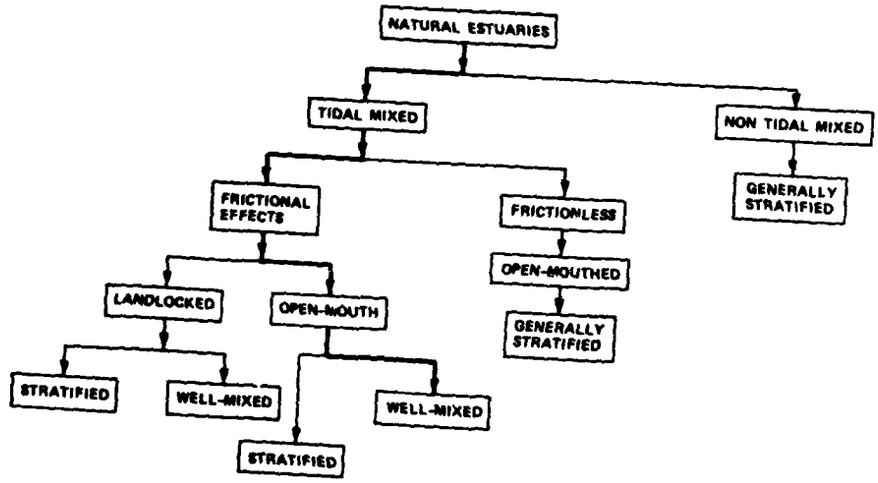


Fig. 1 Estuary classification system

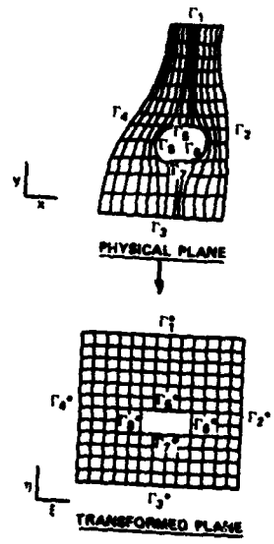


Fig. 2 Transformation of physical estuarine domain to computational domain

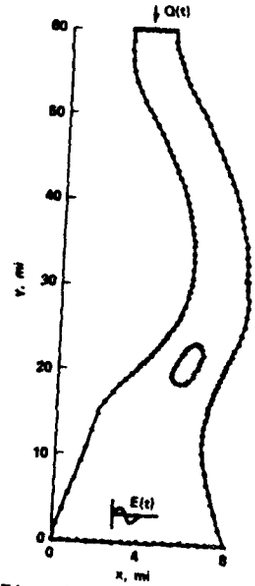


Fig. 3 Boundary points for generation of coordinate system

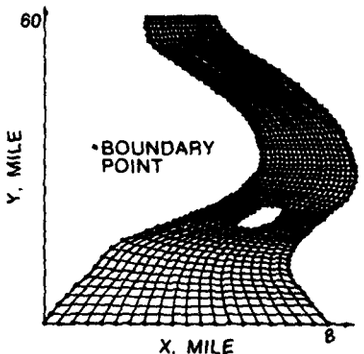


Fig. 4a. Generated coordinate system.

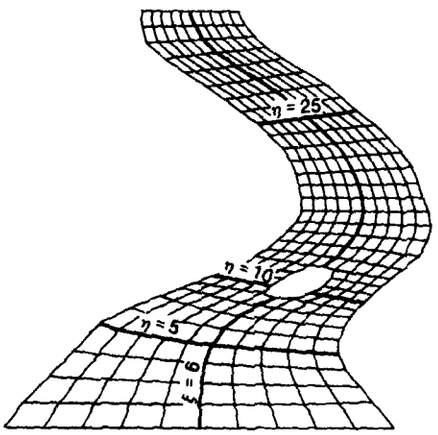


Fig. 4b. Computational grid used in VAHM.

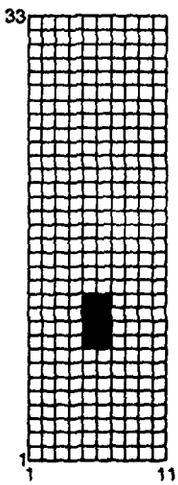


Fig. 4c. Transformed plane.

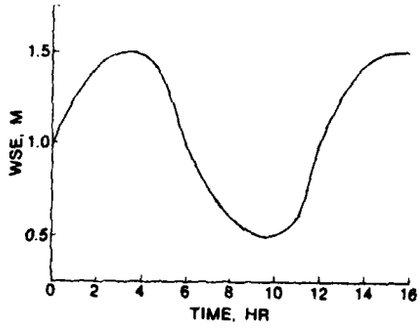


Fig. 5. Water surface elevation at ocean boundary.

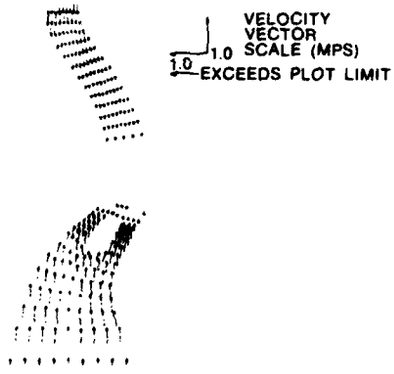


Fig. 6a. Velocity field after 2 hours.

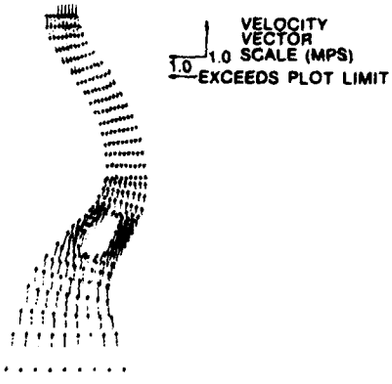


Fig. 6b. Velocity field after 4 hours.

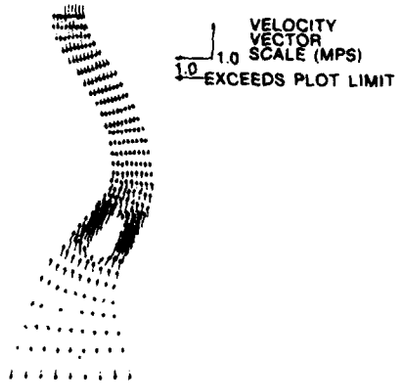


Fig. 6c. Velocity field after 6 hours.

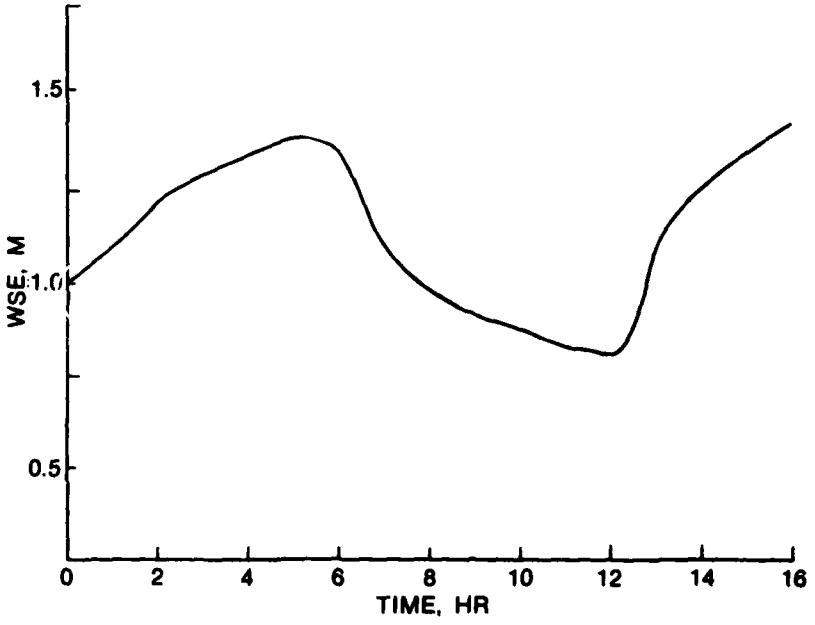


Fig. 7. Water surface elevation at $\xi = 6$, $\eta = 5$.

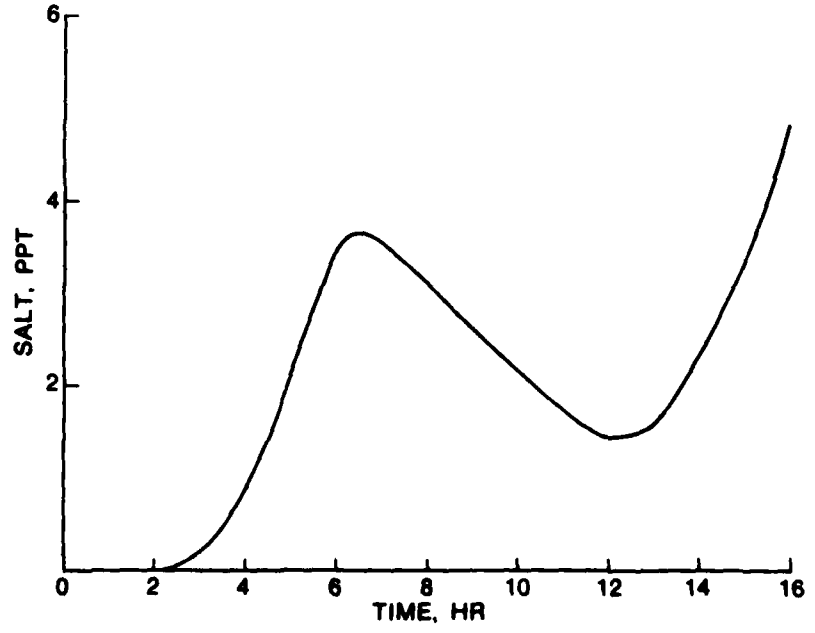


Fig. 8. Salinity at $\xi = 6$, $\eta = 5$.

Numerical Grid Generation

© 1982 by Elsevier Science Publishing Co., Inc.
All rights reserved.

Published by:

Elsevier Science Publishing Co., Inc.
52 Vanderbilt Avenue, New York, New York 10017

Sole distributors outside USA and Canada:

Elsevier Science Publishers B.V.
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

The text of this book appears simultaneously in *Applied Mathematics and Computation*, Volumes 10 and 11 (1982).

Library of Congress Cataloging in Publication Data

Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Use in the Numerical Solution of Partial Differential Equations (1982: Nashville, Tenn.)
Numerical grid generation.

Bibliography: p.
1. Numerical grid generation (Numerical analysis)—Congresses. 2. Differential equations, Partial—Numerical solutions—Congresses. 3. Fluid dynamics—Congresses. I. Thompson, Joe F. II. United States. National Aeronautics and Space Administration. III. United States. Air Force. Office of Scientific Research. IV. Mississippi State University. V. Title.
QA377.S966 1982 515.3'53 82-14244
ISBN 0-444-00757-1

Manufactured in the United States of America

Numerical Grid Generation

Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and their Use in the Numerical Solution of Partial Differential Equations
April 1982, Nashville, Tennessee
Sponsored by NASA and AFOSR
Organized by Mississippi State University

Edited by

Joe F. Thompson
Department of Aerospace Engineering
Mississippi State University, Mississippi State, Mississippi

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

DTIC COPY INSPECTED 3

NORTH-HOLLAND
New York • Amsterdam • Oxford

COMPONENT PART NOTICE

THIS PAPER IS A COMPONENT PART OF THE FOLLOWING COMPILATION REPORT:

(TITLE): Numerical Grid Generation. Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and their Use in the Numerical Solution of Partial Differential Equations, held April 1982, Nashville, Tennessee.

(SOURCE): Mississippi State Univ., Mississippi State. Dept. of Aerospace Engineering.

TO ORDER THE COMPLETE COMPILATION REPORT USE AD-A127 498.

THE COMPONENT PART IS PROVIDED HERE TO ALLOW USERS ACCESS TO INDIVIDUALLY AUTHORED SECTIONS OF PROCEEDINGS, ANNALS, SYMPOSIA, ETC. HOWEVER, THE COMPONENT SHOULD BE CONSIDERED WITHIN THE CONTEXT OF THE OVERALL COMPILATION REPORT AND NOT AS A STAND-ALONE TECHNICAL REPORT.

THE FOLLOWING COMPONENT PART NUMBERS COMPRISE THE COMPILATION REPORT:

AD#:	TITLE:
P000 966	General Curvilinear Coordinate Systems.
P000 967	Error Induced by Coordinate Systems.
P000 968	Basic Differential Models for Coordinate Generation.
P000 969	Elliptic Grid Generation.
P000 970	Conformal Grid Generation.
P000 971	Algebraic Grid Generation.
P000 972	Transfinite Mappings and Their Application to Grid Generation.
P000 973	Orthogonal Grid Generation.
P000 974	Patched Coordinate Systems.
P000 975	Solid Mechanics Applications of Boundary Fitted Coordinate Systems.
P000 976	Coordinate System Control: Adaptive Meshes.
P000 977	On Application of Body Conforming Curvilinear Grids for Finite Difference Solution of External Flow.
P000 978	The Use of Solution Adaptive Grids in Solving Partial Differential Equations.
P000 979	Adaptive Gridding for Finite Difference Solutions to Heat and Mass Transfer Problems.
P000 980	Application of Curvilinear Coordinate Generation Techniques to the Computation of Internal Flows.
P000 981	Solution of Nonlinear Water Wave Problems Using Boundary-Fitted Coordinate Systems.
P000 982	Numerical Modeling of Estuarine Hydrodynamics on a Boundary-Fitted Coordinate System.
P000 983	Automated Three-Dimensional Grid Refinement on a Minicomputer.
P000 984	Automatic Algebraic Coordinate Generation.
P000 985	Automatic Topology Generation and Generalised B Spline Mapping.
P000 986	The Numerical Differentiation of Discrete Functions Using Polynomial Interpolation Methods.
P000 987	Solution of Viscous Internal Flows on Curvilinear Grids Generated by the Schwarz-Christoffel Transformation.

This document has been approved for public release and sale; its distribution is unlimited.

COMPONENT PART NOTICE (CON'T)

AD#:	TITLE:
P000 988	Test Problems, Coordinate Transformations, and Technique for Nonsteady Compressible Flow Analysis.
P000 989	An Experience in Mesh Generation for Three-Dimensional Calculation of Potential Flow Around a Rotating Propeller.
P000 990	Fast Generation of Three-Dimensional Computational Boundary-Conforming Periodic Grids of C-Type.
P000 991	Conformal Grid Generation for Multielement Airfoils.
P000 992	Conformal Mappings onto Multiply Connected Regions with Specified Boundary Shapes.
P000 993	3-D Solution of Flow in an Infinite Square Array of Circular Tubes by Using Boundary-Fitted Coordinate System
P000 994	Generation of Boundary-Fitted Coordinate Systems Using Segmented Computational Regions.
P000 995	Grid Generation by Elliptic Partial Differential Equations for a Tri-Element Augmentor-Wing Airfoil.
P000 996	Numerical Generation of Composite Three Dimensional Grids by Quasilinear Elliptic Systems.
P000 997	Three-Dimensional Grid Generation Using Poisson Equations.
P000 998	Generation of Three-Dimensional Boundary-Fitted Curvilinear Coordinate Systems for Wing/Wing-Tip Geometries Using the Elliptic Solver Method.
P000 999	Numerical Generation of Three-Dimensional Coordinates Between Bodies of Arbitrary Shapes.
P001 000	2-D Elliptic Grid Generation Using a Singularity Method and Its Application to Transonic Interference Flows.
P001 001	Three Dimensional Grid Generation Using Biharmonics.
P001 002	Marching Grid Generation Using Parabolic Partial Differential Equations.
P001 003	Assessing the Quality of Curvilinear Coordinate Meshes by Decomposing the Jacobian Matrix.
P001 004	An implicit Scheme for Water Wave Problems.
P001 005	A vectorized, Finite-Volume, Adaptive-Grid Algorithm for Navier-Stokes.
P001 006	Idealized Dynamic Grid Computation of Physical Systems.
P001 007	Equidistant Mesh for Gas Dynamic Calculations.
P001 008	Applications and Generalizations of Variational Methods for Generating Adaptive Meshes.
P001 009	Orthogonal Coordinate Meshes with Manageable Jacobian.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
Distribution/
Availability Codes
Dist | Avail and/or Special
A

AD P000082

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

AUTOMATED THREE-DIMENSIONAL GRID REFINEMENT ON A MINICOMPUTER

P. D. MANHARDT[+] AND A. J. BAKER[++]
[+]Computational Mechanics Consultants, Inc., 3601A Chapman Highway, Knoxville, TN 37920; [++]Dept. of Engineering Science and Mechanics, Perkins Hall, University of Tennessee, Knoxville, TN 37916.

INTRODUCTION

A persistent requirement in computational fluid dynamics (CFD) applications to practical three-dimensional problem descriptions is a methodology for economically generating three-dimensional solution grids. The basic demands for these grids are smooth progressions of non-uniformity in physical space and a medium of regularity approaching orthogonality in computational space. When accomplished, this permits algorithmic solution of the Navier-Stokes equations in generalized coordinates, taking full advantage of efficient Jacobian factorizations and yielding a significant reduction in computer resource demands. The proceedings[1] of the recent NASA workshop on grid generation techniques summarizes the breadth of procedures available for grid generation, including analytical and algebraic methods as well as numerical solution of hyperbolic and/or elliptic partial differential equations.

Some of these procedures do not readily extend to three-dimensional space in a natural way, especially for multiply-connected solution domains. Others can place excessive demands on human and computer resources just to generate the mesh. A primary requirement, therefore, is an efficient interactive, three-dimensional grid generation capability which is sufficiently flexible to adapt to a broad variety of geometric descriptions without requiring overly extensive data preparation or machine resources useage.

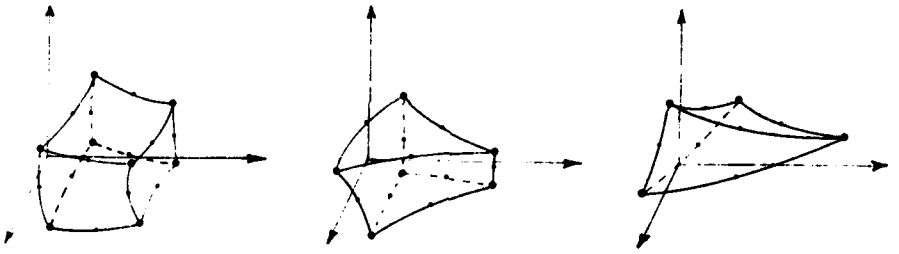
The method described herein, operates on a domain manually subdivided into one or more subdomains called Macro Elements. This subdivision process provides generality for "fitting" the method to geometric boundary shapes of high complexity, including discontinuous surfaces. Each of the Macro Elements is described by its associated vertex and side grid points, thus providing sufficient definition for a bi-quadratic functional interpolation and admitting simply curved boundaries. Generated grids are local to each Macro Element permitting generation of hugh grids on memory limited mini computers and data specification is minimized through use of geometric progressions.

A methodology for piecing the Macro Element data together is described.

The method is table driven and the table is dynamically generated from specified Macro Element connection data. Connectivity is independent of Macro Element orientation and the generated table locates exterior boundaries automatically for boundary condition specification. The method is illustrated with generation of 26,460 grid points for a wind tunnel model support flow-field. Data specification consists of 9 hexahedron shaped Macro Elements and 72 gridpoints.

MACRO ELEMENTS

Generally, grid refinement is accomplished by manually subdividing the solution domain into regular geometric subdomains called Macro Elements. In three-dimensional space, the Macro Elements are six sided (hexahedrons), five sided (pentahedrons) or four sided (tetrahedrons) as illustrated in Figure 1. The Macro Elements are defined by vertex grid points and edge grid points such that three points define any given edge. This provides sufficient information for second degree hermitian polynomial approximation thus permitting Macro Elements to accurately reflect simply curved boundary geometries[2].



a) Hexahedron b) Pentahedron c) Tetrahedron

Fig. 1. Three Dimensional Macro Elements

Each Macro-Element is arbitrarily subdivided three dimensionally to form a partial grid. The partial grids are subsequently combined, eliminating dupiicate generated data at the boundaries, to form a singly connected, undirected loop free mesh.

Partial grids are formed over a Macro Element utilizing a three dimensional-second degree, hermite polynomial interpolation function. The functional transformed space for a hexahedron, for example, is a two unit cube on an orthogonal coordinate system at the cube center (figure 2). Hence, the forward coordinate transformation is

$$X_i = X_i(\eta_i) = \{N(\eta_j)\}^T \{XI\} \qquad (1)$$

In equation (1) the elements of the parametric basis $\{N(\eta_j)\}$ are for the case of figure 2 the bi-quadratic interpolation functions of η_j and $\{XI\}$ are the coordinate triples of the vertex and edge grid points defining the hexahedron boundaries.

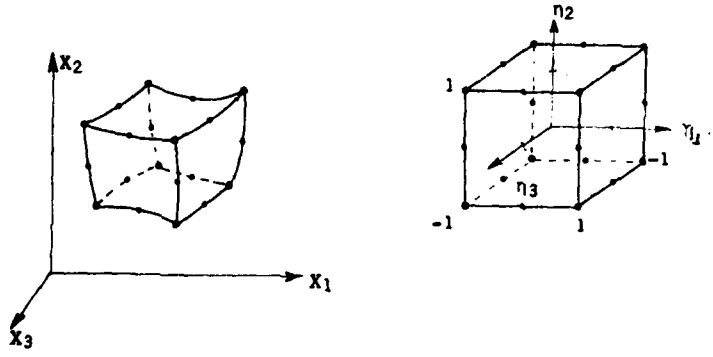


Fig. 2. Hexahedron Transformation

As illustrative of the functional description $(N(\eta_j))$, consider the two dimensional case of a quadrilateral Macro Element[3] (figure 3).

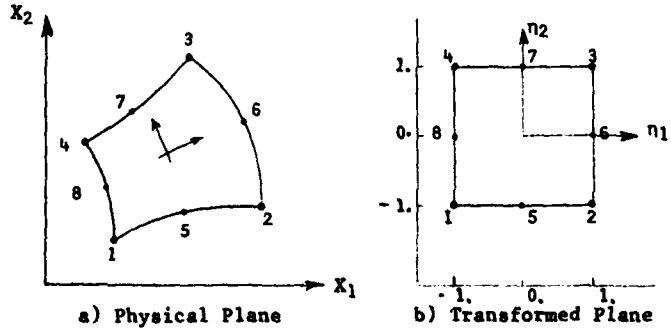


Fig. 3. Natural Coordinate Plane (η_1,η_2) Mapping of a Quadrilateral.

For this case, $\{N(\eta_j)\}$ is evaluated at each of the vertex and side grid points of figure 3 and appears as:

$$\{N(\eta_j)\} = 1/4 \begin{Bmatrix} (1-\varepsilon)(1-\eta)(-\varepsilon-\eta-1) \\ (1+\varepsilon)(1-\varepsilon)(\varepsilon-\eta-1) \\ (1+\varepsilon)(1+\eta)(\varepsilon+\eta-1) \\ (1-\varepsilon)(1+\eta)(-\varepsilon+\eta-1) \\ 2(1-\varepsilon^2)(1-\eta) \\ 2(1+\varepsilon)(1-\eta^2) \\ 2(1-\varepsilon^2)(1+\eta) \\ 2(1-\varepsilon)(1-\eta^2) \end{Bmatrix} \qquad (2)$$

where ε and η at the side grid points are specifically at mid-side and the equations are ordered according to the grid point numbering noted in Figure 3(b). Substitution of equation (2) into (1) for a specific set of $\{x_j\}$ and evaluating the equation over the limits of ε and η (-1 to 1) yields a biquadratic approximation of x over the subdomain. Accuracy of the values of x_i are dependent upon the ability of the shape functions to approximate the physical geometry. A curvature which is exactly biquadratic for instance will be interpolated exactly using equation (2). The side nodes in Figure 1(a) need not be at exactly mid-side since it is not required in the definition of {N}.

A rather unique by-product of the method (serendipity function family) involves placement of the side grid points. The functional interpolation of equation (1) is symmetric when a side grid point is at exactly mid-side in physical space. Hence, mid-side is the obvious location when a quarter circle is to be approximated. Movement of the side point, however, causes a non-symmetry to occur which becomes continuously more accentuated at less centroidal locations. This effect provides a continuous selection of curvatures for approximating boundaries of general shape, thus demonstrating broad applicability of the method as a design tool.

The interpolating nature of equation (1) can be best illustrated by evaluating equation (2) at the specified grid points and noting the results when evaluating (1). For example at η_j = (-1, -1), equation (2) becomes [1, 0, 0, 0, 0, 0, 0, 0] and evaluation of equation (1) for each Xi yields X_1, X_2, X_3 at grid point 1 in figure 3. Likewise, evaluation of equations (1) and (2) for any η_j point pair ($-1 \leq \eta_j \leq 1$) will yield a corresponding point pair in physical space. Extension to 3D is direct.

A primary requirement for grid refinement is smooth variation of grid density for adaptation of the grid to match solution requirements. Evaluation of equations (1) and (2) over Macro-Elements ensures this, since choice of inter-

polation points in η_i is completely arbitrary. Specification must be simple, however, and the suggested methodology is a geometric distribution function ranging ($-1 \leq \eta_j \leq 1$).

$$\eta_{i,j} = \eta_{i,j-1} + \Delta\eta_{i,j-1}^{P_i} \qquad j = 2,n \tag{3}$$

In equation (3), n is the number of generated grid points (nodes) in an η_i direction and P_i is the geometric progression ratio. P_i greater than 1, therefore, causes the grid to grow coarser as j increases and P_i less than 1 causes it to become finer.

Extending the above refinement scheme to two and three dimensions introduces further complexity. Using figure 3(a) for visualization, it is intuitively obvious that P_1 can vary in the η_2 direction and vice versa. Letting P_1 have geometric variation yields an equation for P similar to that for η in equation (3).

$$P_{i,j} = P_{i,j-1} + \Delta P_{i,\,j-1}^{Q_i} \qquad j = 2,\ \eta_j \tag{4}$$

In equation (4) Q_i is the geometric progression ratio causing variation in P_i and for three dimensions, two equations are required (one for each orthogonal direction). Using this method, grid variation over the domain is very general and data specification is minimized. A three dimensional problem for example, requires specification of 6 Q values, 3 p values and 3 n values for each Macro Element. Proper selection of p and Q values for a required grid refinement becomes intuitive with practice since values normally range between .7 and 1.4. As noted later, however, interconnection of Macro Elements imposes the further requirement that grids progress smoothly across coincident boundaries. This is easily accomplished by solving equations (3) and (4) for p and Q from computed or specified end condition grid spacings.

GENERATED GRID ANALYSIS

The quality of a specified grid is measurable in a relative sense. The above mentioned grid generation procedure results in a non-uniform distribution of finite elements over each Macro Element. The isoparametric functional representation for each finite element appears as in equation (1). For linear finite element basis the ($N(\eta_j)$) of equation (1) become the linear approximation set. The estimated error in the energy of a generated grid for each Macro Element can be formed from evaluation of the H^1 Sobolev (and L^2) norms. These

are derived from the inner products of grid and finite element approximation function as

$$\left|\left| X_i^h,\ X_i^h \right|\right|_{H^1} = \sum_{e=1}^{M} \left[\{XI\}_e^T \left([M]_e + [K]_e \right) \{XI\} \right] \qquad (4)$$

where

$$[M_e] = h_e \int_e \{N_j\}\ \{N_j\}^T\ d\vec{x}$$

and

$$[K_e] = h_e \int_e \nabla\{N_j\} \cdot \nabla\{N_i\}^T\ d\vec{x} \qquad (5)$$

In equation (4) the summation is over all generated elements in each Macro Element yielding partial values of scalar energy estimation. These partial scalars are subsequently summed over all Macro Elements to form the total energy estimate. Thus, distributed and overall energy levels are available for comparison and are useful in determination of the quality of a particular specified grid distribution.

GLOBAL CONNECTIVITY

Each of the Macro Elements is refined independently subject to its vertex and edge grid point specifications in global coordinates. The second requirement of the method is to piece the refined grids together to form a global, singly specified grid. The piecing is most efficiently accomplished using Macro Element data information, since the data is sparse, can all fit in main memory and requires minimal searching and comparison. A required data specification therefore, is a Macro Element connection table which specifies the globally defined vertex and side grid point numbers comprising each Macro Element. The connection table is searched to locate adjacent Macro Element sides and a side matching table[3] is formed "on the fly." The side matching table is subsequently used to locate lines and planes of grid points to be eliminated during grid refinement. In addition, the table is useful for locating exterior boundaries since no adjacent sides are found and the table contains zeros[3]. This is useful for locating and mapping boundary conditions on exterior generated grid points.

The memory resident side matching table provides the means for efficiently decoupling Macro Element refinement and, thus permitting huge grid generation

on mini- or mid-sized machines. Macro Element refined data can be generated and written to disk or efficiently paged (virtual systems) by dynamically maximizing array sizes to match rows, planes, or complete Macro Element generated grids. The side matching table is interrogated "on the fly" and duplicate side (2D) or planar (3D) data is never produced. The method in pseudo code is:

```
Loop over the Macro Elements
Loop over the subdivisions
  Generate grid and parameter
  data subject to matching tables
  Store generated data
End loop 1
End loop
```

Thus the method can be made efficient on any machine by maximizing the I/O block or page size.

Another consideration for Macro Element interconnection is orientation of the local Macro Element η_i coordinates and ordering of generated elements to minimize Jacobian Matrix bandwidth. As was illustrated[3], Macro Element orientation is completely arbitrary. Equations 1 and 2 yield identical results regardless of Macro Element orientation, and as noted earlier, the side matching table is useful for elimination of repeated boundary nodes.

Generated node ordering for minimum bandwidth, however, is not guaranteed unless the numbering and orientation of Macro Elements is carefully considered. This is especially true for three dimensional geometries where matrix bandwidth can become quite large. The tendency of the Macro Element grid generator without node reordering treatment is to form a sparse block matrix. The matrix can, however, be globally treated to narrow the bandwidth for efficient envelope method solution[6].

A 3D EXAMPLE

As illustration of the method, an infinite flow-field surrounding a wind tunnel model support was discretized (figure 4). The model support was approximated using two intersecting cones and the flow-field extends approximately 18 feet in all directions.

As illustrated in Figure 4, the flow-field was segmented into 9 hexahedron shaped Macro Elements for grid refinement. Each Macro Element is defined by 20 grid points (8 vertex, 12 side) for a total of 76 grid points in all. A mesh numbering 2744 hexahedron shaped finite elements was generated over each Macro Element domain for a total of 24,696 generated finite elements (26,460 grid points) over the entire flow-field. Figure 5 presents a coarse grid perspective view of a 72 element generated grid for 3D visualization.

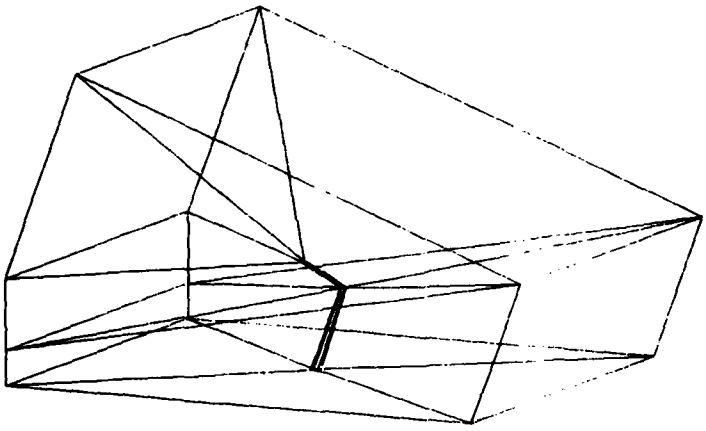


Fig. 4. Model Support, Macro Element Data Description

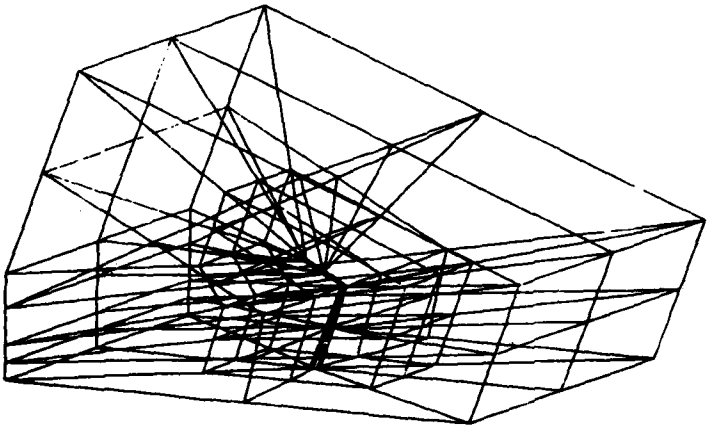


Fig. 5. Model Support Coarse Grid Perspective View

Planar sections of a generated grid on Macro Element boundaries are easily extracted for visualization during generation using the side matching table. Figure 6 presents the full grid generated in the base plane. The circular cone is exactly approximated by the three dimensional equivalent of equations (1) and

(2). Nonuniform grid distribution focuses the grid in the active cone region. Generated grid in the plane of cone intersection is illustrated in Figure 7. The cone radius is about half that of the base plane. Finally, Figure 8 illustrates a coarse grid in the vertical half plane of the model support. Grid refinement is altered by simply modifying the three grid size integers for each Macro Element.

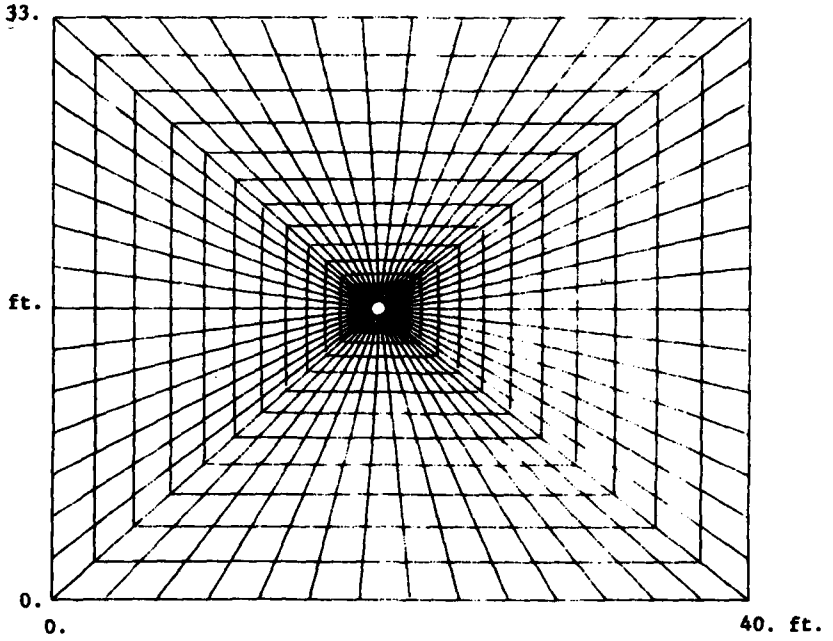


Fig. 6. Model Support Base Plane Discretization

The complete 26,460 grid point generation was performed on a PDP11-34 having 64K of memory and RK05 disk drives in approximately 4 hours. Time for solution increases linearly with grid refinement, thus producing a 50,000 mesh on a relatively slow machine in less than 8 hours. These timings are for a job obviously I/O bound and include additional I/O required for generation of graphical output. Execution actually occurred within 26K of machine memory and due to size limitations (the operating system and libraries require 32K) I/O block size was minimized and the number of I/O interrupts became extremely large. It is anticipated that increasing array storage by doubling the memory capacity of the machine will improve speed by a factor of 5.

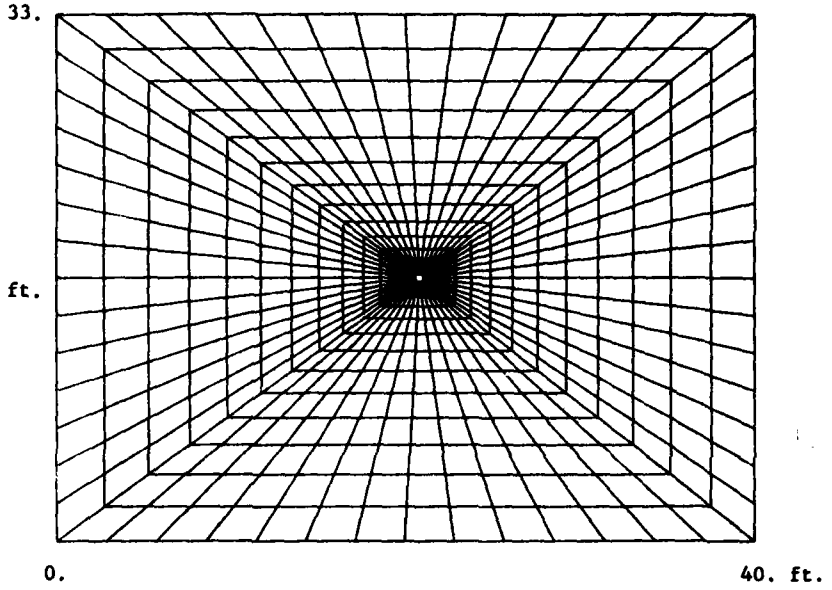


Fig. 7. Model Support, Cone Intersection Plane Discretization

REFERENCES

1. Proceedings of NASA Workshop on Numerical Grid Generation Techniques, Report NASA-CP-2166, 1980.
2. Zienkiewicz, O. C., The Finite Element Method in Engineering Science, McGraw-Hill, London, 1971.
3. Manhardt, P. D., Baker, A. J., "Automatic Discretization Refinement and Graphics for Improved Accessability of Complex Fluid Mechanics Computer Programs," Numerical Laboratory Computer Methods in Fluid Mechanics, A.S.M.E., 1976.
4. Baker, A. J. and Soliman, M. O., "On the Utility of Finite Element Theory for Computational Fluid Dynamics," Technical Paper AIAA-81-1031, 1981.
5. Prenter, P. M., Splines and Variational Methods, John Wiley, New York, 1975.
6. George, Alan and Liu, Joseph W., Computer Solution of Large Sparce Positive Definite Systems, Prentice Hall, Inc., Englewood Cliffs, N. J. 07632, 1981.

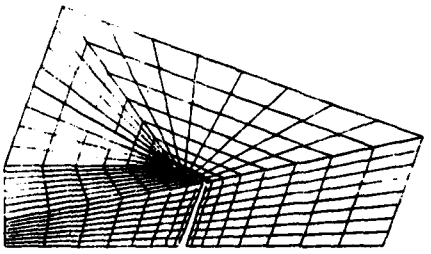


Fig. 8. Model Support, Symmetry Plane Discretization

AD P000984

Published 1982 by [illegible]
NUMERICAL GRID GENERATION
Joe F. Thompson, editor
Not copyrighted. Unrestricted free use is granted by Lockheed Missiles & Space Company, Inc.

AUTOMATIC ALGEBRAIC COORDINATE GENERATION

Peter R. Eiseman
Department of Applied Physics and Nuclear Engineering, Columbia University, New York, New York 10027

INTRODUCTION

A computer software system has been developed to automatically generate two-dimensional coordinates from algebraic transformations. For topologically complex regions, a smooth assembly of the transformations can be used to automatically produce a composite mesh where a general gridded format is retained. To readily obtain a desireable level of global mesh smoothness and to permit a maximum amount of mesh control, the application of local multisurface transformations[1-3] is emphasized. In the formation of the composite mesh, the boundaries for each transformation are either transmissive or the prescribed gemoetry of given physical objects.

In addition to the boundaries, each multisurface transformation is constructed from a sequence of intermediate control surfaces. At each fixed surface parameter, the control is over the corresponding transverse coordinate curve connecting boundaries. In the direction of the surface parameter, the control is longitudinal. Whether longitudinal or transverse, the fundamental properties are the geometry and pointwise distribution along coordinate curves. With the local form of the multisurface transformation, these controls can be precisely given at any location. For basic mesh patching, derivative conditions are prescribed at boundaries. Away from boundaries, the control is applied independently so that desireable mesh structures can be smoothly embedded or boundary slope discontinuities can be kept at the boundaries regardless of pointwise distributions along boundaries.

The algebraic mesh generation system consists of a collection of operator subroutines which are applied to an established data structure and which automatically perform the necessary parts of mesh construction from a sequence of multisurface transformations. The system is split into operational modules for surface definition, for transverse constructions, for discretization, for mesh assembly, and for graphics. Surface definition and graphics modules are applicable to all 2D coordinate generation techniques; the remainder are tailored to but not limited to multisurface transformations. The surfaces here are curves $\vec{\alpha}(t) = (x(t), y(t))$ and their definition consists of both their intrinsic geometry $\vec{\alpha}$ and their parameterization t. The parameterization is equivalent to the coordinte distribution for the one-dimensional surfaces.

Such coordinates are automatically generated with a general curvature clustering mechanism and with the simultaneous capability to prescribe any number of arbitrary clustering locations. When applied to line segments, a general class of useful distribution functions is also obtained.

MULTISURFACE TRANSFORMATIONS

To obtain the two-dimensional multisurface transformation, a sequence of constructive one-dimensional surfaces $\vec{\alpha}_k(t)$ are prescribed with an ordering from boundary $\vec{\alpha}_1(t)$ to boundary $\vec{\alpha}_N(t)$ with the intermediate curves $\vec{\alpha}_2(t)$, $\vec{\alpha}_3(t)$, ..., $\vec{\alpha}_{N-1}(t)$ available for control. At each value of t, a piecewise linear connecting curve is determined by joining the successive points $\vec{\alpha}_k(t)$ with straight line segments. In correspondence with the N-1 straight line segments, a partition $r_1 < r_2 < \ldots < r_{N-1}$ for the assumed transverse coordinate r is prescribed. In further correspondence, a sequence of interpolation functions $\psi_k(r)$ is defined to vanish at all partition points except r_k as k = 1,2, ..., N-1. With the functions, the line segment slopes are interpolated. The general N-surface transformation is obtained by an integration from $\vec{\alpha}_1(t)$ and by a subsequent fit to $\vec{\alpha}_N(t)$ from k normalizations. In terms of the Cartesian locations $\vec{c}$ = (x,y), it is given by

$$\vec{c}(r, t) = \vec{\alpha}_1(t) + \sum_{k=1}^{N-1} \frac{G_k(r)}{G_{N-1}(r_{N-1})}[\vec{\alpha}_{k+1}(t) - \vec{\alpha}_k(t)] \qquad (1)$$

where

$$G_k(r) = \int_{r_1}^{r} \psi_k(z)\, dz$$

The slope interpolation is evident when r-derivatives of both sides are evaluated at r_k. Upon a back substitution, the N-surface transformation can then be expressed in a derivative form. In either form, it can also be expressed as a projector, under which, all coordinates that conform to the given specified properties are projected into the multisurface system. With an interchange of r and t, a similar projector can be defined for the t-direction which in turn can be used as a Boolean summand with the first projector to get an extension of transfinite interpolation methods applicable to function and derivative specifications at all levels. Since the simplest case is with the basic building blocks in one direction and since one direction is sufficient for many applications, the basic multisurface

transformation will be directly applied here with the understanding that the developed automation can also be carried over into the transfinite cases.

SYSTEM OPERATORS

The Data Base

The automation of algebraic transformations is accomplished with a system of operators that are applied with respect to a fixed data base. The data base is split into arrays which either need or need not be understood by someone who wishes to apply the system. Those which do not require understanding typically communicate technical data between the various operator subroutines in an internal automatic fashion. The remaining arrays are of more interest since we are required to either load them for the input or to unload them for the output. For all coordinate systems of the generally composite mesh, the primary input is the data required to define the constructive 1-D surfaces $\vec{\alpha}_k(t)$ for the transformation of Eq. 1. Each surface is represented as a sequence of data points which are distributed well enough to adequately describe its basic geometry and parameterization. Each data point in the sequence consists of two Cartesian components x and y and a surface coordinate t to form an ordered triple (x, y, t) for the parametric representation (x(t), y(t). The basic surface data array is given by

$$\begin{aligned} SD(I, J, 1) &= x \\ SD(I, J, 2) &= y \\ SD(I, J, 3) &= t \end{aligned} \qquad (2)$$

for surface number I and data point number J. For each coordinate system number IC, the data point numbers J vary from 1 to NDATA and the surface numbers I vary from NS(IC, 1) to NS(IC, 2) where the local surface number for the transformation of Eq. 1 is given by K = I-NS(IC,1) + 1. The primary output array is the coordinate mesh array given by

$$\begin{aligned} CM(2*IC-1, J, K) &= x \\ CM(2*IC, J, K) &= y \end{aligned} \qquad (3)$$

for the Jth mesh point in r and the Kth mesh point in t. Specified quantities are the number of r-mesh points MR(IC) and the number of t-mesh points MS(IC). In the current system, the dimensions are given by NS(4,2), SD(40, 100, 3),

and CM (8, 30, 50) which imply that IC $\leq$ 4, NS $\leq$ 40, NDATA $\leq$ 100, MR $\leq$ 30, and MS $\leq$ 50. To save storage, the arrays SD and CM are also equivalenced. The dimensions can be readily changed, for example, to reduce the number of possible coordinate systems and increase the number of mesh points.

The Order of Application

Relative to the established data structure, the system of operators are applied in a sequential fashion to automatically generate a wide variety of grids that conform to the specifications we desire. The entire process can be given a general ordering. Surface parameterization clearly follows surface constructions which must first be in existence; similarly, transverse mesh evaluations must follow the transverse constructions for curve definition and pointwise distribution. The transverse curve definition depends upon the integrals $G_k(r)$ of Eq. 1. With a new independent variable z, we get a function r(z) which yields $G_k(r(z))$ to include both curve definition and pointwise distribution. After both surface and transverse mesh data has been computed, it is assembled to form a mesh from the transformation. Once all of the coordinate meshes are assembled individually, the grand assembly of the global patched together mesh can be viewed either numerically or as a plot.

The system of operators for coordinte generation are written as FORTRAN subroutines and are separated into the categories which are illustrated in Figure 1 along with their sequential interdependence. The categories on the top of the figure are related to surface definition, and consequently, are the primary parts of the mesh generation process. Each category is described separately in the following sections.

Direct Surface Generators

The category of direct surface generators contains the operators on the SD-array which generate surface sections or entire surfaces without parameterization t and without any dependence on existing surfaces. In each case, the 1-D surfaces are generated as a pointwise sequence (x_i, y_i) ordered in an increasing fashion from beginning to end. If the curves are closed, the ordering becomes either clockwise or counterclockwise. As a matter of notation, all operators will be labeled in capital letters to correspond with the associated subroutine names. Basic operator definitions will follow an

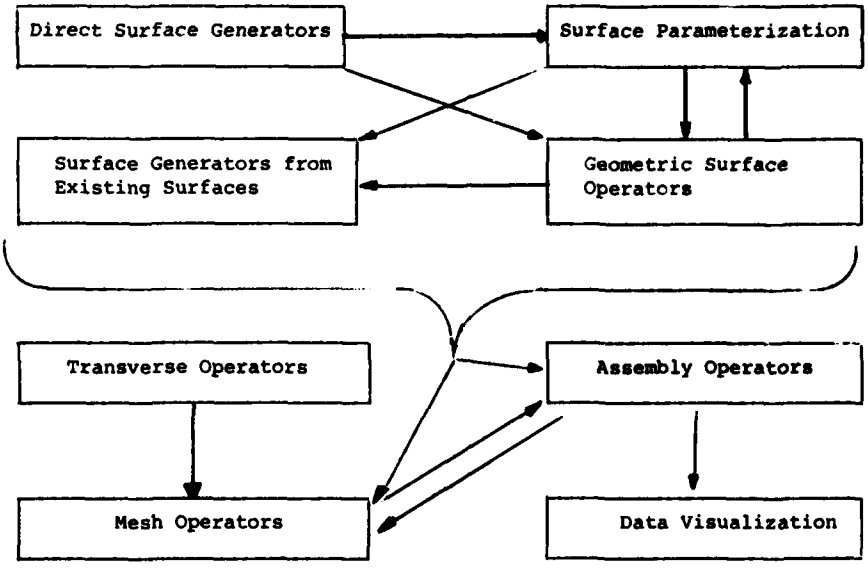


Fig. 1. Operator categories and their interdependence.

arrow that is behind the operator name. In this format, the current operators for direct surface generators are given by

LINE	→	Line segment
CIRCLE	→	Circular segment
BEND	→	Two line segments joined by circular arc
BOX	→	Rectangle with rounded corners (4)
OVAL	→	Ellipse or superellipse
NACA	→	NACA airfoil

Geometric Surface Operators

The category of geometric surface operators contains the operators which depend only on the surface geometry and not on any surface parameterization. When the basic geometry of a surface is known in the SD-array as a sequence of data point locations, the operators of this category can be applied. The current operators are given by

ROTATE	→	Rotate a surface	
SLIDE	→	Slide a surface	
FRAME	→	Construct unit tangent and normal vectors at each point.	(5)

Surface Parameterizatin Operators

Surface parameterization operators are applied to generate the 1-D surface parameterization, or equivalently, surface coordinates t. The operators can be split into those which depend upon an existing parameterization of the given surface and those which do not. The ones which do not are given by

ARCLN	→	Arc length	
PLOAD	→	Load from another surface on a data point basis	(6a)
EXTPAR	→	Project from another surface along a vector field	

The ones which do are given by

PNORM	→	Normalize to an arbitrary interval	
GPAR	→	Reparameterize globally	(6b)
LPAR	→	Reparameterize locally	

With the reparameterization operators, pointwise clustering can be simultaneously done at higher curvature locations and at arbitrarily selected locations. For the arbitrary locations, the selection is accomplished by loading the existing parametric intervals and the degree of desired clustering into the array BUNCH in the form

BUNCH (J,1)	=	Start of cluster	
BUNCH (J,2)	=	Center of cluster	(7)
BUNCH (J,3)	=	End of cluster	
BUNCH (J,4)	=	Degree of cluster	

for cluster number J which, with current dimensions, can be at most 20.

Surface Generators from Existing Surfaces

The basic geometry, the parametric representation, and the data point indexing of existing surfaces can each be utilized to generate new surfaces that are aligned or positioned in such a way that geometric specifications for the mesh are easily given. The specifications include the boundary prescriptions on the behavior of transverse coordinate curves, and more generally, prescriptions of mesh forms in an area sense anywhere in the meshed region. The surface generators from the existing surfaces form a category which extends the direct generators and contains the operators

$$\begin{aligned} &\text{EXPAND} \rightarrow \text{Expand away from a surface to get a new one} \\ &\text{DEFORM} \rightarrow \text{Deform between two surfaces to get a new one} \end{aligned} \tag{8}$$

With EXPAND, the expansion is along directions from a vector field. When the vector field is normal to the existing surface, the data points on the new surface are orthogonally aligned. If subsequently, PLOAD from Eq. 6a is applied, then the orthogonal alignment is used for the new curve. With DEFORM, the deformation is along line segments that join the two existing 1-D surfaces. When applied successively with PLOAD, coordinate line segments can be smoothly embedded into the mesh.

Transverse Operators

The definition of the basic transverse curves for each multisurface transformation (Eq. 1) is accomplished with the transverse operators which are given by

$$\begin{aligned} &\text{PART} \rightarrow \text{Generate partition points } r_k \\ &\text{GKVAR} \rightarrow \text{Generate coefficients of } G_k(r) \end{aligned} \tag{9}$$

Currently, the partition from PART is uniform.

Mesh Operators

After the parameterized surfaces have been constructed and the transverse piecewise polynominals have been defined, each multisurface transformation can be used to generate its corresponding part of the entire two-dimensional mesh. The number of mesh points in both directions (MS and MR for Eq. 3) must be specified and is independent of both the previous discrete parametric surface

constructions and the transverse piecewise polynominal definitions.[1-4] With the specification, the category of mesh operators is applied to obtain mesh point discretizations for each multisurface transformation component ($\vec{\alpha}_k$ and G_k of Eq. 1) and is split into those oriented towards evaluation and those which change the mesh. The evaluation operators are given by

SFIND → Find surface interval containing t
RFIND → Find partition point interval containing r (10)
SMESH → Surface mesh
GK → Evaluate $G_k(r)$,

are usually called by assembly operators, and are consequently used less frequently on an isolated basis. The operators which change the mesh are given by

RDIST → Mesh distribution in r
CSLIDE → Slide a coordinate system
GRID1 → Generate orthogonal trajectories (11)
GRID2 → Generate trajectories with exact central difference orthogonallity

The transverse mesh distribution from RDIST can have specified clustering at each endpoint and simultaneously in the center. The orthogonal trajectories from GRID1 are generated by the method of Graves and McNally[5-6] but with an accuracy increase from automatic refinement.[7] The orthogonal trajectories from GRID2 are generated by the Leap-Frog method presented in Eiseman.[7]

Assembly Operators

To streamline the mesh generation process, assembly operators have been developed to cover the most frequent applications and to reduce the required number of calls to operators. They are given by

TRANS → Assemble multisurface transformations (12)
GRID → Assembles grids without required calls to transverse, mesh, and visualization operators

The operator GRID assumes a uniform r-distribution from RDIST and generates grids from TRANS, GRID1, and GRID2. In contrast to TRANS, only the surface data and the desired number of mesh points are needed.

Data Visualization Operators

To observe the constructive process and to see the mesh in graphical form, the category of data visualization operators has been created. Observations of the surface data array SD and the various other arrays used in the constructions can be viewed through a sequence of printing operators. The printing of the coordinate mesh CM of Eq. 3 is included in the primary visualization operator which is

$$\text{PLOTIT} \rightarrow \text{Plot the entire composite mesh} \qquad (13)$$

This operator is automatically called by the assembly operator GRID which also includes the technical calls for the plotting set up.

APPLICATIONS

To illustrate the mesh generation process with the system of opertors, we first consider a region between an oval contour and a symmetrically surrounding box with rounded corners. Only one coordinate system is needed for this region. The pattern for grid generation is then established on this simple example and is continued further towards other cases. To define the coordinates for the example, the number of constructive surfaces must be selected and is chosen here to be 5. The surfaces are depicted in Fig. 2 where they are numbered from 1 to 5 going from the ellipse to the box. The first data point of each surface is on the x-axis from which the counter-clockwise orientation is indicated by an arrow. The dashed curve, other than the x-axis, is one of the piecewise-linear curves determined by equal parameter values t and is the curve which determines directions for the associated transverse coordinate curve in r. In correspondence with the figure, the surfaces and thus the grid are automatically generated from the short coding sequence:

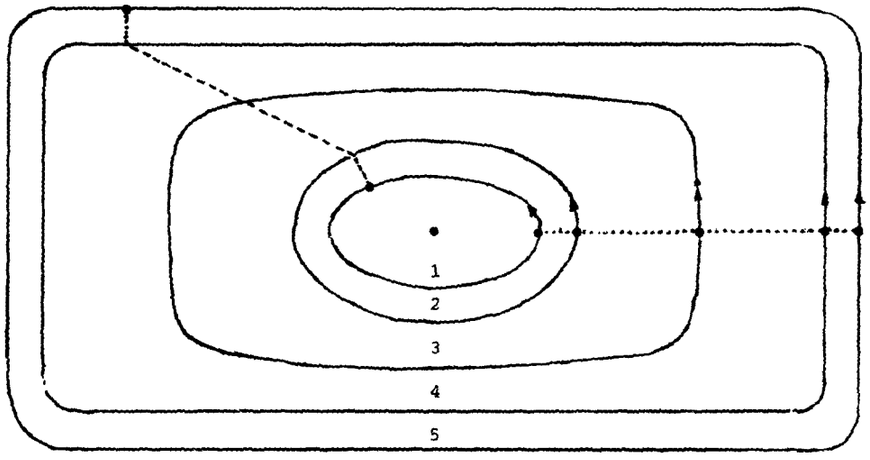


Fig. 2. The constructive surfaces for an oval in a box.

```
MR(1)    = 15   → Number of mesh points in r
MS(1)    = 49   → Number of mesh points in t
NDATA    = 100  → Number of surface data points
NS(1,1)  = 1    → First surface number
NS(1,2)  = 5    → Last surface number
OVAL            → Surface 1
ARCLN           → Arc length t along surface 1
PNORM           → Normalize t to unit interval
EXPAND          → Surface 2 as outward normal expansion from surface 1
PLOAD           → Load t from surface 1 into 2
BOX             → Surface 5
ARCLN           → Arc length t along surface 5
PNORM           → Normalize t to unit interval
EXPAND          → Surface 4 as inward normal expansion from surface 5
PLOAD           → Load t from surface 5 to 4
DEFORM          → Surface 3 as deformation between surfaces 2 and 4
PLOAD           → Load t from surface 2 into 3
GRID            → Assemble and display the grid
END
```

From the successive application of EXPAND and then PLOAD, an orthogonal alignment in t is obtained respectively between surfaces 1 and 2 and surfaces 5 and 4. From the successive application of DEFORM and then PLOAD, a linear alignment in t is obtained between surfaces 2, 3, and 4. Both alignments are illustrated by the dashed piecewise-linear curve in Figure 2. The grid from the coding sequence is displayed in Figure 3 where the effect of the alignments is viewed as orthogonality at each boundary and local linearity in between.

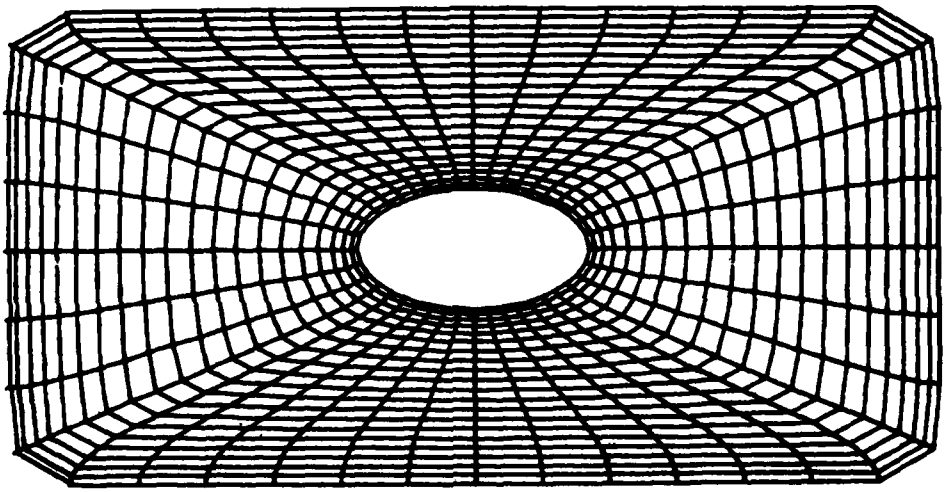


Fig. 3. Grid for an ellipse in a box.

If OVAL is replaced by NACA in the coding sequence, then surface 1 becomes a NACA airfoil contour. With PNORM replaced by GPAR, we get curvature clustering for the leading edge. When the outward normal expansion for surface 2 by EXPAND is adjusted to give the expansion distances illustrated in Figure 4, we get the grid which is displayed in Figure 5. As the trailing edge is approached, the r-coordinates have bends which occur progressively closer to the airfoil surface due to the placement of surface 2.

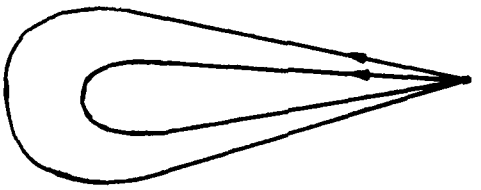


Fig. 4. First two surfaces for an airfoil in a box.

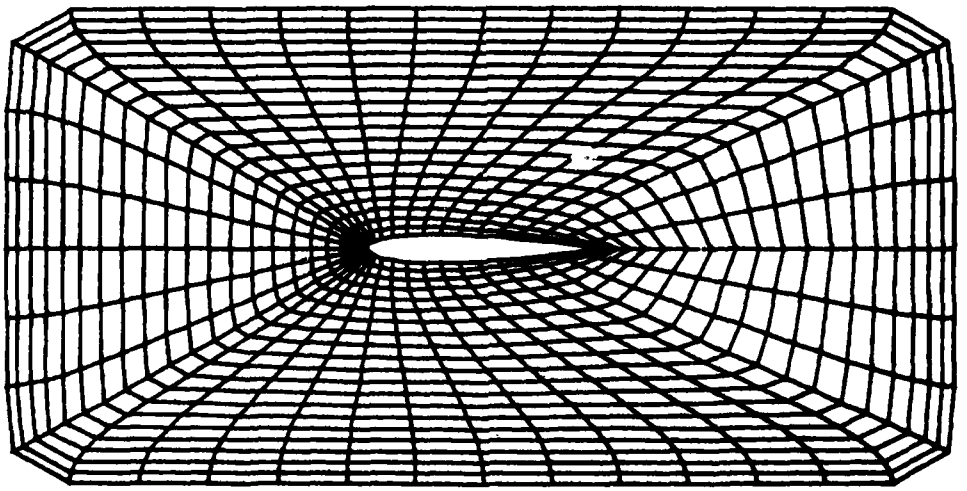


Fig. 5. An airfoil in a box without trailing edge clustering.

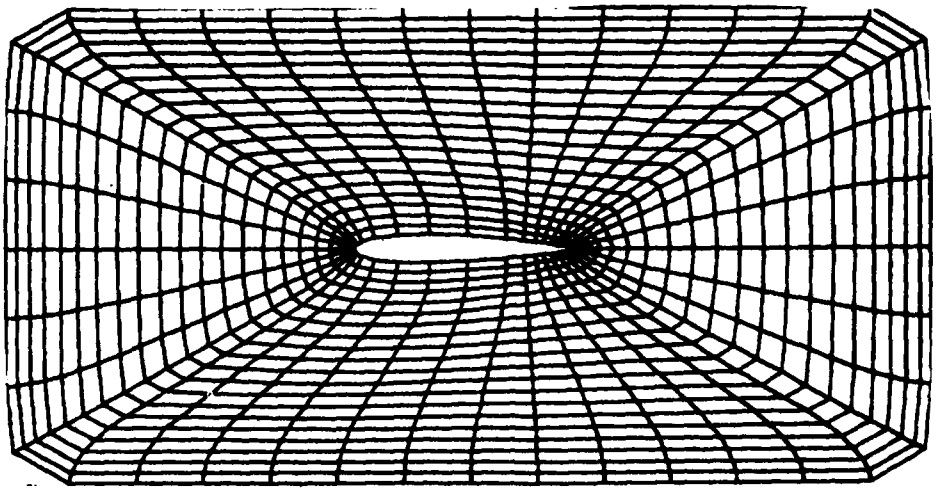


Fig. 6. An airfoil in a box with trailing edge clustering.

This is equivalent to specifying normal derivatives which have progressively decreasing magnitudes. At the training edge, the slope discontinuity is propagated across about two thirds of the field. Because of the local transverse interpolants for Eq. 1, the discontinuity is gone in the last third. It can also be removed from the entire field by clustering. This is accomplished by inserting trailing edge clustering requirements into the BUNCH-array for GPAR to act upon. The result is displayed in Figure 6. An airfoil in a cascade with 45° stagger can be obtained with ROTATE applied to surface 1 and with successive applications of BEND for surface 5. When RDIST is used to cluster points near the airfoil, we get the periodically aligned grid of Figure 7 where Cartesian extensions are given in upstream and downstream directions. A detailed view at 30° stagger is seen in Figure 8 with uniform r-distribution. Returning to the airfoil in a box, a local region of orthogonal coordinates is created about the airfoil by the additon of two more surfaces that are orthogonally aligned with the airfoil. With more mesh points, we get the 7-surface transformation which is displayed in Figure 9.

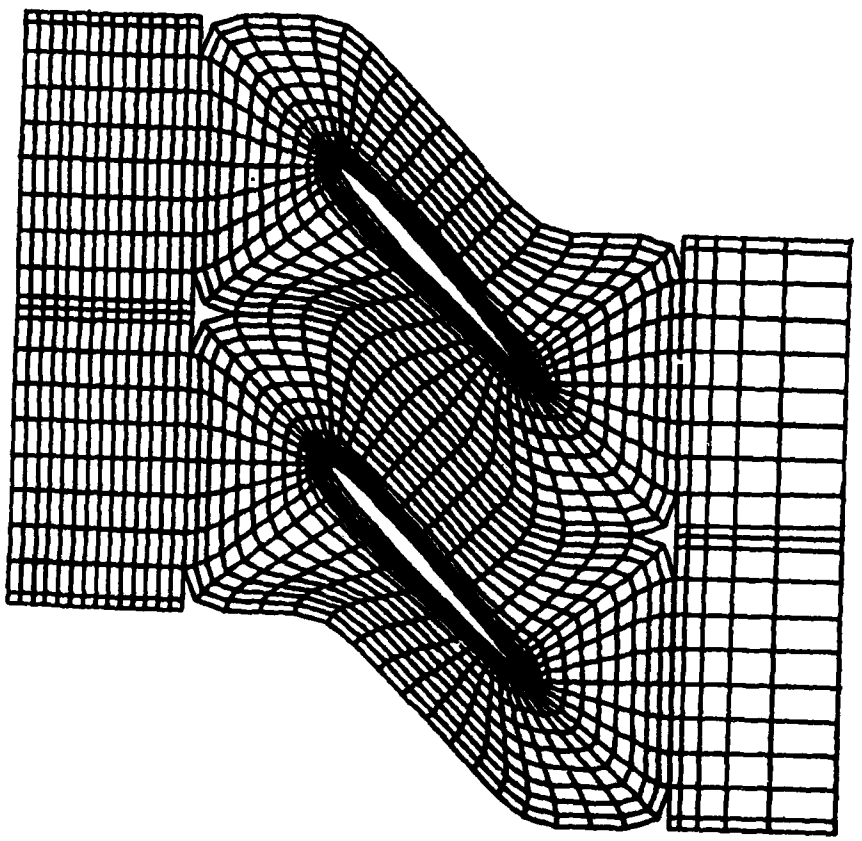


Fig. 7. Airfoil cascade at 45° stagger.

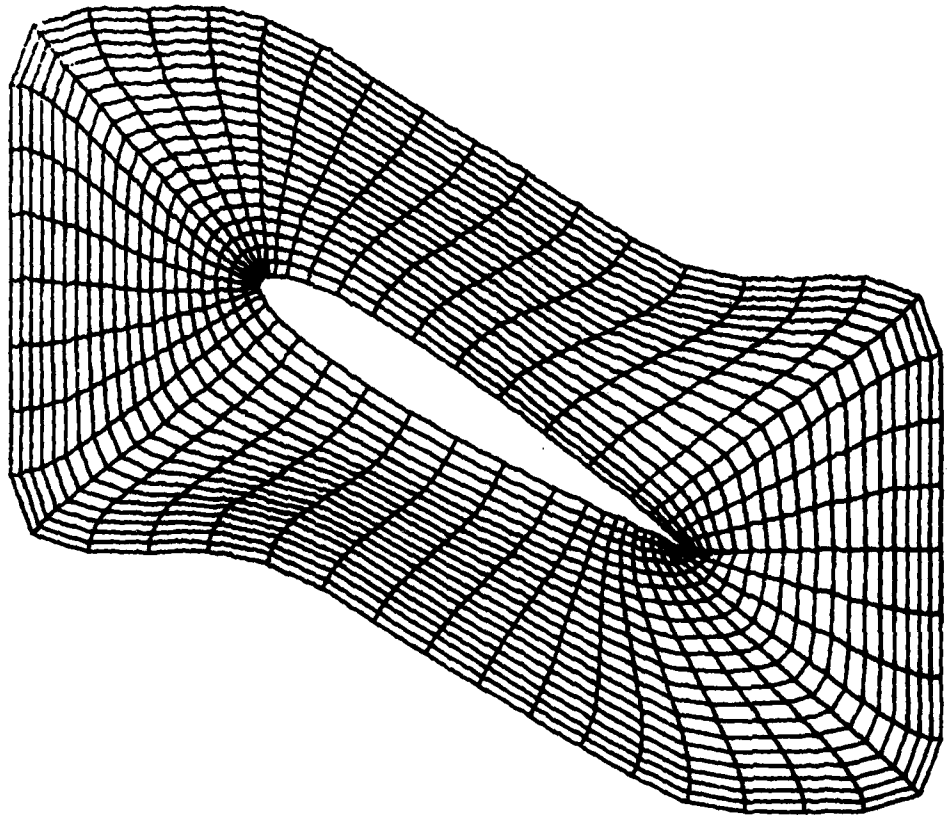


Fig. 8. Basic system for airfoil cascade at 30° stagger.

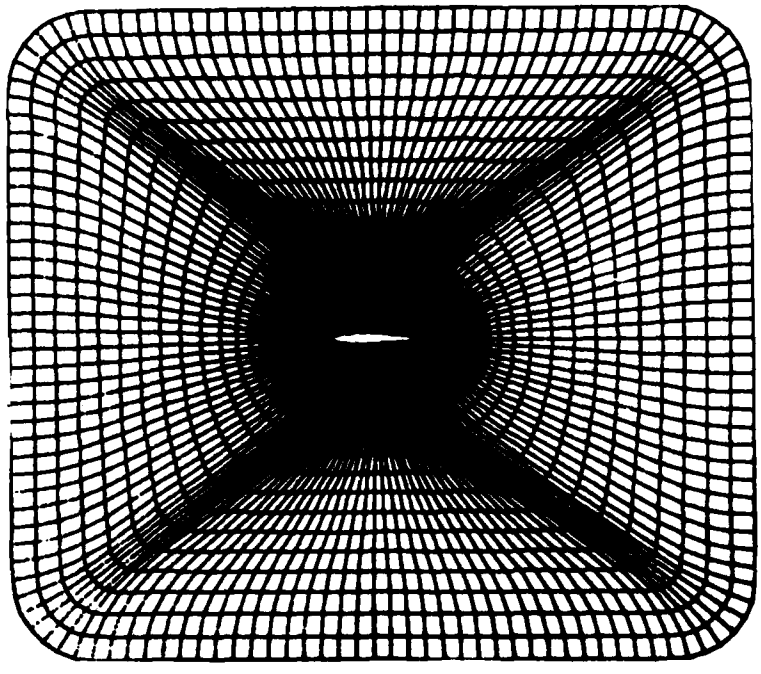


Fig. 9. Airfoil in a box with orthogonal disk around the airfoil.

CONCLUSION

The general format for automatic alegbraic mesh generation has been established with a system based upon multisurface transformations. The system is a collection of operators which automatically perform the various constructive tasks and which are applied in sequence to generate a mesh. The primary operator sequence is for surface construction and for distribution functions. Surface construction and distribution functions are required for boundaries and are basic to all methods of coordinate generation.

ACKNOWLEDGMENTS

Work supported by NASA Lewis Research Center, Contract NAS3-22117 and by NASA Langley Research Center, cooperative agreement NCCI-59.

REFERENCES

1. Eiseman, P. R. (1980) Coordinate Generation with Precise Controls, ICASE Report 80-30, to appear J. Computational Physics.
2. Eiseman, P. R. (1980) Coordinate Generation with Precise Controls Over Mesh Properties. Seventh Int. Conf. on Num. Meth. Fluid Dyn., Lecture Notes in Physics 141, 176.
3. Eiseman, P. R. and R. E. Smith (1980) Grid Generation using Algebraic Techniques, NASA CP2166.
4. Eiseman, P. R. A Mesh Generation Software System. NASA Lewis Research Report, to appear.
5. Graves, R. A. (1980) A Simple Numerical Orthogonal Coordinate Generator for Fluid Dynamic Applications. NASA CP2166.
6. McNally, W. D. (1972) FORTRAN Program for Generating a Two-Dimensional Orthogonal Mesh Between Two Arbitrary Boundaries. NASA TND-6766.
7. Eiseman, P. R. Orthogonal Grid Generations, this proceedings.

AD P000985

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

AUTOMATIC TOPOLOGY GENERATION AND GENERALISED B SPLINE MAPPING

A ROBERTS
British Aerospace

SUMMARY

Completion strategies have been programmed for the removal of topological anomalies in a cube cluster. Synthesis of complicated grid structures can then be defined with very few explicit statements. The realisation of the differential equations uses a generalisation of B spline mapping for modelling through topological singularities in the field. Such B spline schemes provide accelerated convergence and wide band accuracy. Relaxation sequences with B spline can then be more efficient than those for finite difference methods even for topologically regular grids.

INTRODUCTION

The Multi-Volume Data Structure (MVDS) is a general purpose topology system for linking a range of industrial data bases to a range of CFM methods.

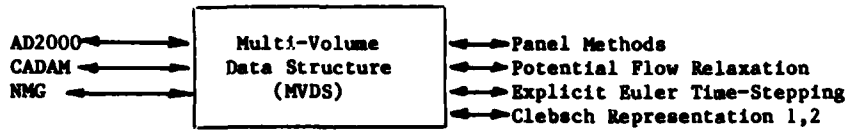


Fig 1. The MVDS System Links Data Bases to Methods

The use of multi-grid techniques for partitioned fields with mapping singularities in the field presents common organisational problems.

The automatic topological units are 'starters' such as a wing-body combination and 'details' such as an engine pod. Each unit refers to both an aircraft component and the adjacent field volumes. The library of such units can be augmented using a cryptic language. To permit display of the grid configuration default coordinates are generated automatically.

The grid is partitioned into a number of cubic blocks connected face to face with 3,4,5 or 6 blocks meeting along junction lines in the field.

PREVIOUS PAGE
IS BLANK

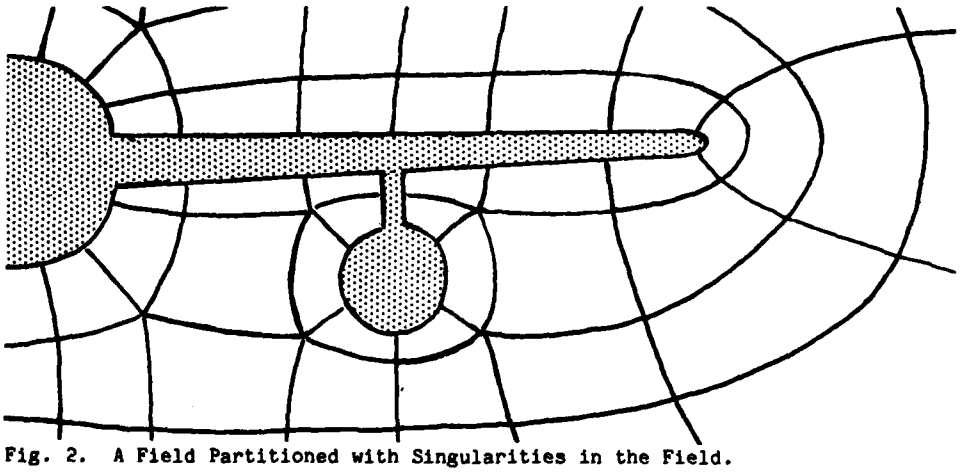


Fig. 2. A Field Partitioned with Singularities in the Field.

Each volume has a different grid structure at each multi-grid resolution level:-

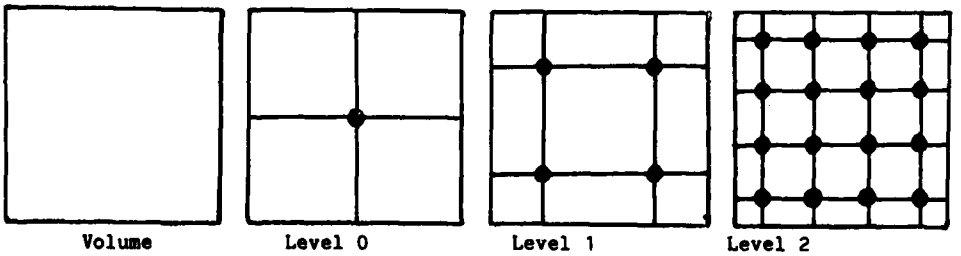


Fig. 3. Multi-grid structures in a volume

At every level the partition boundary is half a parametric interval from the nearest grid surface. Topological singularities then occur between grid points:-

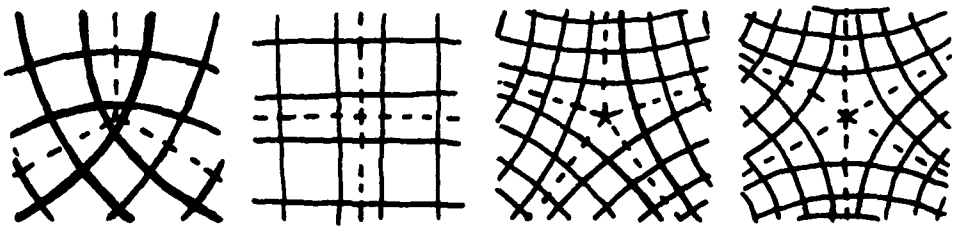


Fig. 4. Mapping Singularities at Node Points.

Power -1 Power 0 Power +1 Power +2

The grid is said to be regular in regions where four volumes meet along every junction line. In such regions the partition boundaries and junction lines have no numerical significance although the orientation of the parametric axes may be discontinuous across partition boundaries. Near topological singularities, shock surfaces and vortex sheets the usual operator expansions are not applicable. B spline techniques can then be used for modelling through such anomalies.

Tesselation of Surfaces

A tesselation of a surface is a decomposition of that surface into four sided cells. Tesselation operations are used here to illustrate strategies suitable for cube clusters. For cube clusters with many singularities in the field, definition by enumeration is not practicable due to the mass of information required combined with conceptual problems. Automatic conversion of a general decomposition of a 3-D field via a tetrahedron cluster is possible in principle but produces unsuitable grid structures. We are then left with progressive synthesis using operations that preserve the cube cluster conventions.

We consider a general polygonal decomposition of a simply connected closed surface. We can select any interior point of a line or a cell to be the median point of that cell. A median line is then a line joining the median point of a cell to the median point of one of the enclosing cell boundary lines. The tesselating operation is then the reclassification of every median line of an old decomposition as an additional boundary line of the new decomposition.

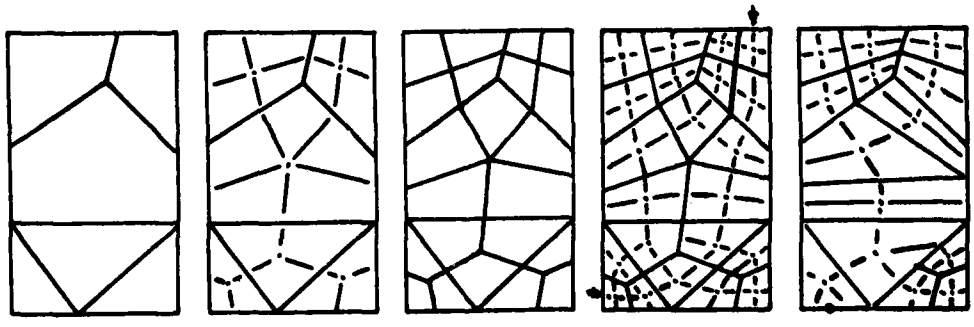


Fig. 5. Tesselation Operations

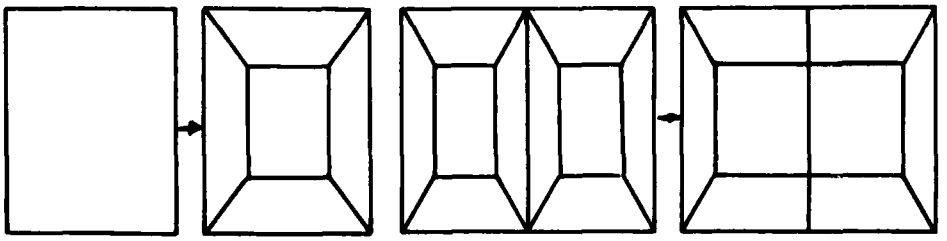
Repeated use of the tesselating operation produces a family of tesselations suitable for a multi-grid technique.

The median lines of the tesselation of a closed surface join up to form endless strings of median lines. A cell string is a set of cells where each cell contains segments of a particular median line string. We may note that tesselation conventions are preserved by the deletion of any complete string for a simply connected closed surface.

For any tesselation of a simply connected closed surface we can delete cell strings, one at a time, until we are left with the minimum tesselation consisting of just two cells. By reversal of this sequence we can synthesise any tesselation of a simply connected closed surface from the minimum tesselation one string at a time.

The general synthesis operation is then to separate any closed string of cell boundary lines into a pair of boundary strings enclosing the new cell string. The boundary line string can be defined by enumeration of the boundary segments forming that string. A more cryptic definition is possible if the boundary line string is always concave with respect to a 'captured zone'. The string then always follows the second left turn except for the nominated bend points where it follows the first left turn. The string is then defined by the bend points or, if there are no bend points, by one segment of the string.

The realisation of the new cell string following this convention can follow a strategy of progressive capture based on just two elementary operations:-



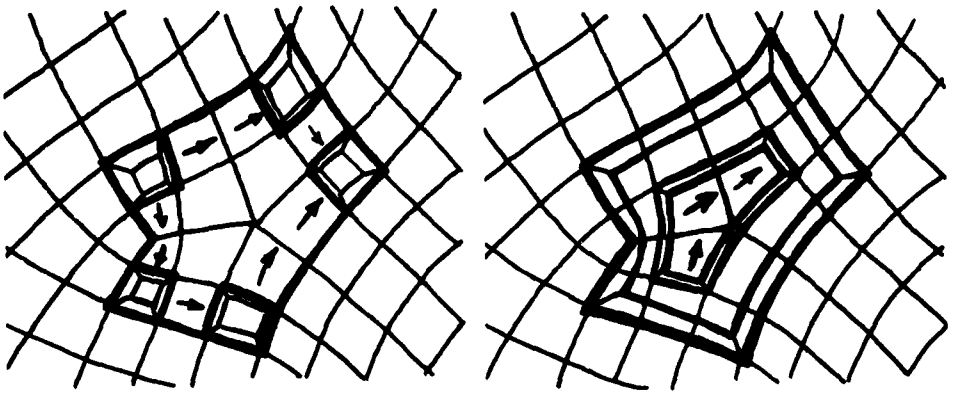
Capture One Cell Delete Two Cells

Fig. 6. Elementary Operations on a Surface

These operations can be controlled by autonomous propagation sequences operating at two priority levels.

First priority propagation is propagation along strings until a loop is completed or a corner encountered.

Second priority propagation is propagation through an area enclosed by new strings.



Bend Points Captured Strings Captured

Fig. 7. Capture Phases

Basic Cube Cluster Operations

The direct analogue of a tesselated closed surface is a cube cluster in which the minimum cube cluster consists of a finite cubic volume enclosed by an 'external cube' consisting of all space not included in the finite cube. We have median surfaces of cubes bounded by a median line of each of four volume interfaces:-

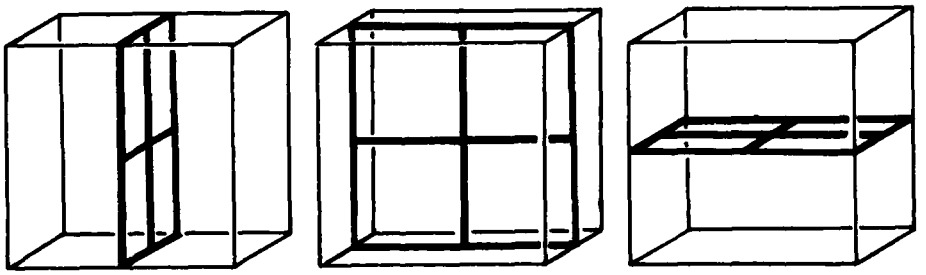
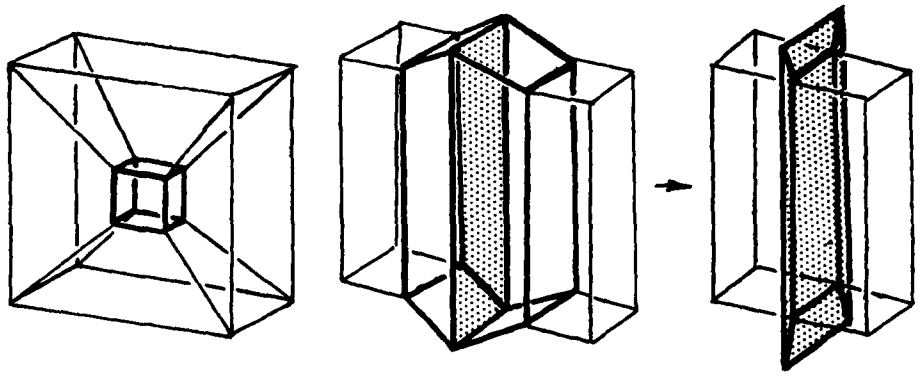


Fig. 8. The Three Median Faces of a Cube.

Median lines of an interface can be regarded as connecting a pair of median faces of one cube to a pair of median faces of an adjacent cube. Median faces then join up to form edgeless sheets. A volume sheet is a set of volumes in which each volume contains facets of a particular median surface sheet. A volume string is the intersection of a pair of volume sheets. Each volume is the triple intersection of volume sheets and of volume strings.

The cube cluster conventions are conserved by the deletion of any complete volume sheet. Any cube cluster can be reduced to the minimum cube cluster by deletion of volume sheets one at a time. Sheets may be deleted in any sequence. By reversal of such sequences any cube cluster can be synthesised one sheet at a time using one of many different sequences.

We have the pair of elementary operations:-



Capture One Volume Delete Two Volumes

Fig. 9. Elementary Operations in a Cube Cluster

We can then have first priority propagation along strings, second priority propagation throught sheet segments bounded by strings and third priority propagation through a captured volume cluster. Corners are points where three bend lines meet. The specification must then enumerate explicitly:-

1. all corners with 1,2,4 or 8 inserted into one old volume

2. for any bend line not terminating at either end by a corner one sample volume referring to 1,2 or 4 bend lines.

3. for any surface not bounded anywhere by bend lines one sample volume referring to 1 or 2 new surfaces.

All other instructions are implicit.

After grid relaxation each median surface will form a surface for which the unit normal is continuous almost everywhere. Such surfaces are useful for displays of the grid and for contours of field functions. Where median surfaces form closed sheets the Euler characteristic[3] can be used to relate the

number of handles[3] h to the sum of the topological powers s where the topological power of a singularity is four less than the number of volumes meeting along a singularity line. For such tesselated surfaces we have

$$s = 8(h-1)$$

This is a useful rule when selecting a suitable topology.

Solid and Null Volumes

At the lowest level of the internal logic the two elementary operations are used and the topology is always a properly connected cube cluster. However, the required grid has sheets that can be bounded by an envelope or aircraft surfaces, regarded as intrusions of the envelope into the field. We can designate volumes in a cube cluster as solid volumes. No propagation is permitted through solid volumes. A null volume is a volume of zero thickness with one great side connected to some old volume or some solid volume and five faces connected to new volumes that may be null volumes.

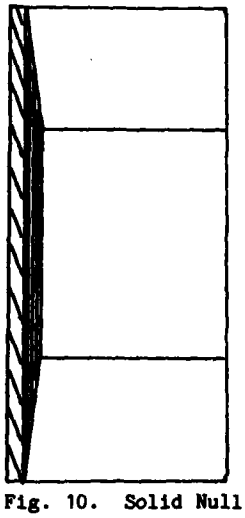


Fig. 10. Solid Null Volume

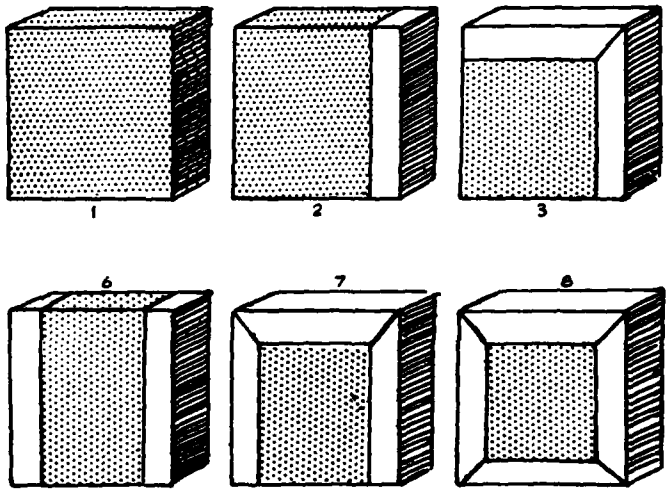


Fig. 11. Apparent Propagation Forms

The null volumes thus bridge apparent anomalies in the cube cluster conventions. There are six propagating forms as shown. Forms 3, 7, 8 propagate 1, 2 and 4 bend lines respectively as first priority propagation along strings. Forms 2 and 6 propagate 1 or 2 sheets through some median sheet segment. Form 1 propagates type distinctions. The shaded zones represent core zones with the same sheet numbers as the original volume. With the qualification SOLID all core zones are designated as solid in the insertion of a sheet. The other volumes are volumes of the new sheet. Forms 5, 6, 9, 10 introduce 1, 2, 4 or 8 corner points of the new sheet.

The control of propagation is based on the examination in turn of each of the six volumes that enclose the core volume. Each volume either is or is not a null volume. The great side can be adjacent to a solid volume, a field null volume, or an old field volume. If it is adjacent to an old field volume, that volume may appear already at the same or higher priority level or at a lower level or not appear in the pending operations lists.

Table 1 NULL VOLUME TESTS AND ACTIONS

TEST	ACTION IF TRUE
VOLUME IS NOT NULL	RETURN
ADJ. VOL IS SOLID	DESIGNATE NULL VOLUME AS SOLID
ADJ. VOL IS NULL VOLUME	DELETE PAIR OF NULL VOLUMES
ADJ. VOL ALREADY IN SAME OR HIGHER LEVEL	RETURN
ADJ. VOL ALREADY IN LOWER LEVEL	DELETE LOWER LEVEL REF. AND RECORD
ADJ. VOL NOT ALREADY RECORDED	RECORD PENDING OPERATION

In practice few sheets require explicit identification of more than two volumes. If all sheets in a 'detail' intersect one sheet then all instructions can refer to that sheet.

These conventions provide a cryptic language for the introduction of additional 'starters' and 'details'.

Topological Displays

In order to define the topology of a library unit in the cryptic language the user must draw a series of sketches. A set of sketches in fig. 12 represent the development of a simple wing-body configuration. In this case all explicit statements refer to the horizontal plane of symmetry. Each one line statement specifies which volume (denoted by #), which operation and, if required, what is the fractional width of the new sheet. If the set of statements is accepted a full set of displays of solid surfaces and field sheets can be requested. With no further data solid surfaces generated from the sketches in fig. 12 are displayed in the form shown in fig. 13. Other examples of display generated from the cryptic topology are shown in fig 14. These show a pitot intake, a side intake and a fan-in wing with adjacent field sheets.

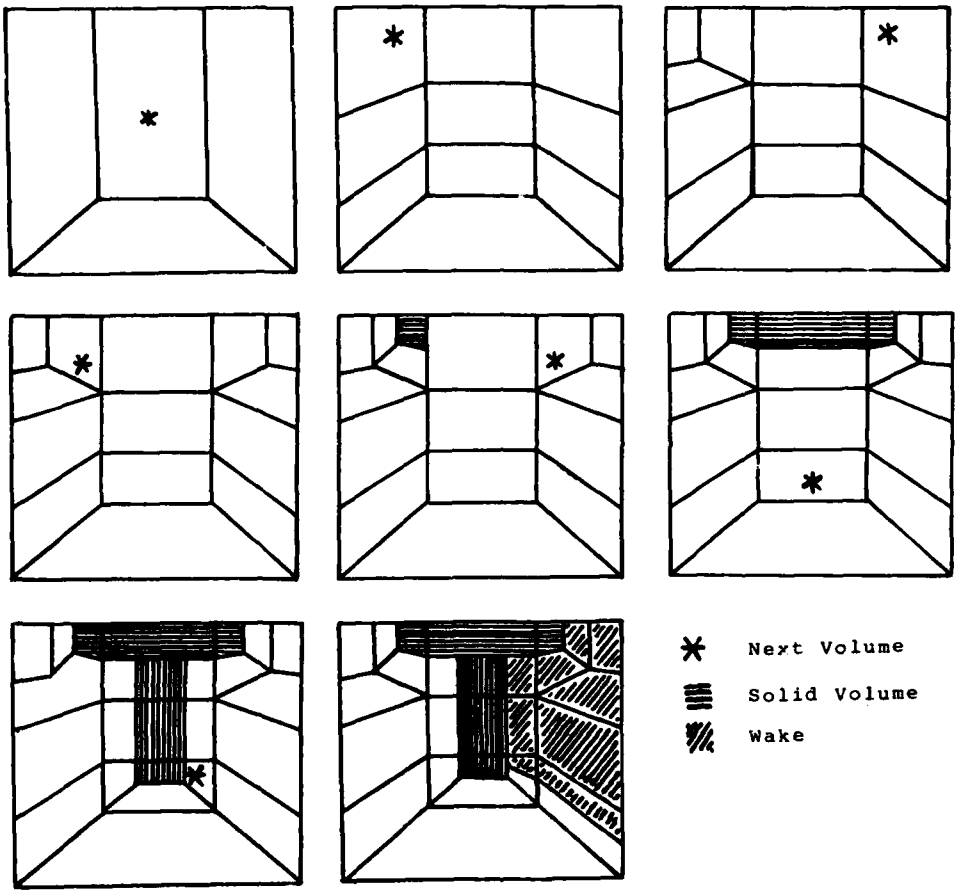


Fig. 12 User's Sketches for a Wing Body Combination.

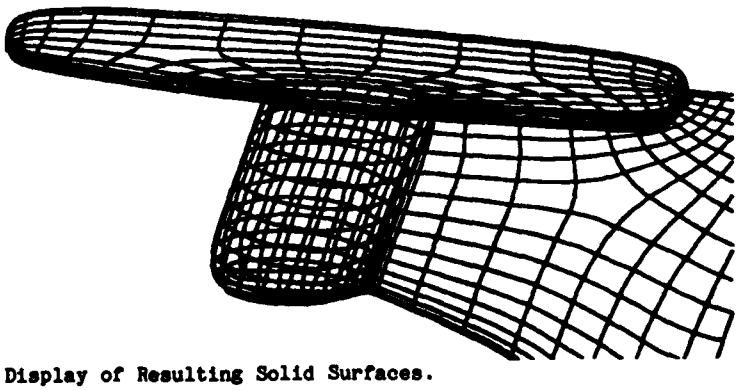
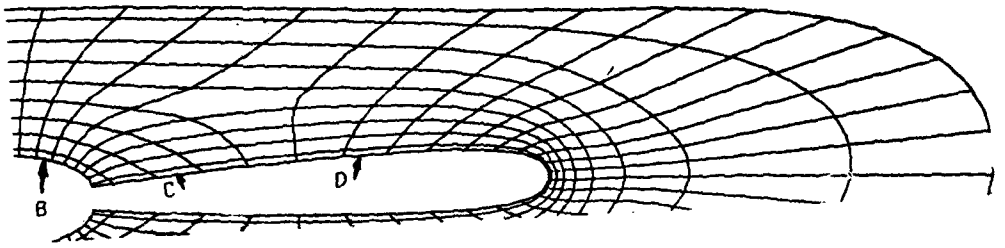
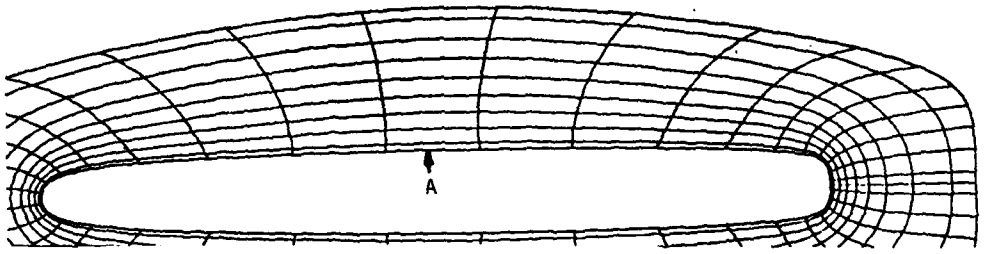


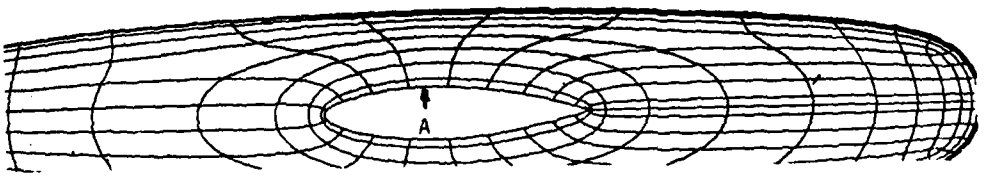
Fig 13a. Display of Resulting Solid Surfaces.



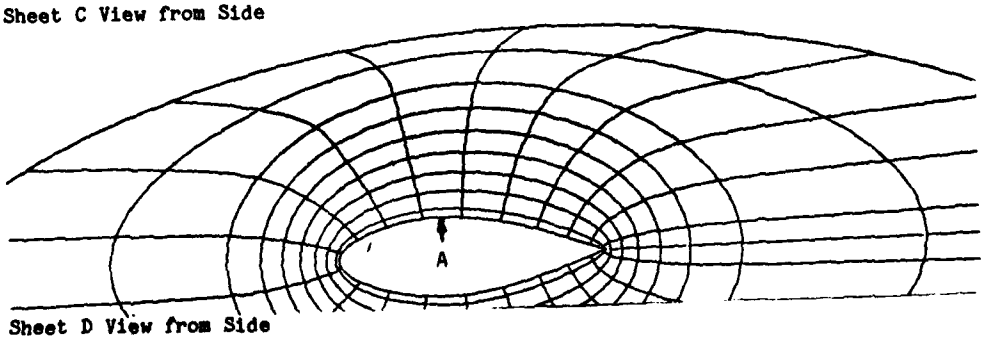
Sheet A View Looking Aft



Sheet B View from Side



Sheet C View from Side



Sheet D View from Side

Fig 13(b) Median Surfaces after Relaxation

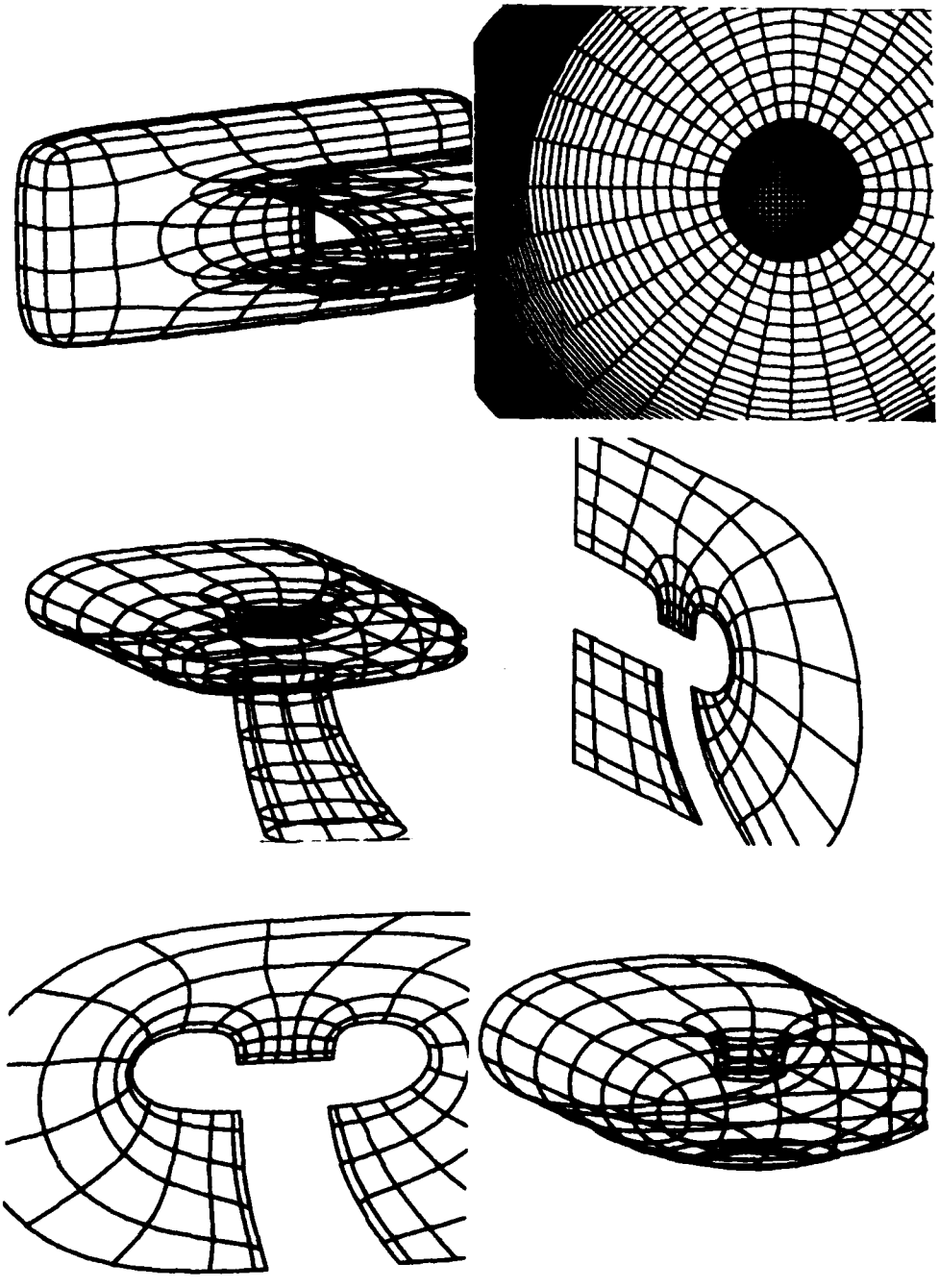


Fig 14. Other Displays Generated from the Topology Definition.

Basis Function Mapping

The B Spline and finite difference schemes will be regarded here as alternative interpretations of the integer spline conventions. The integer spline schemes in turn are realisations of the basis function conventions in topologically regular regions of the grid. The B spline generalisation required to model through singularities remains within basis function format although integer spline conventions are violated. The basis function analysis is relevant near singularities where the operator expansions are not valid. The basis function equations and flat space tensor equations are defined as follows:-

ξ^1,ξ^2,ξ^3	parametric coordinates	x^1,x^2,x^3	Cartesian coordinates
$f_i(\xi^1,\xi^2,\xi^3)$	field functions	$e_j(\xi^1,\xi^2,\xi^3)$	basis functions
λ_{ij}	scalar coefficients	ψ	field function

At all ξ^1,ξ^2,ξ^3 we have

$$\sum_j e_j(\xi^1,\xi^2,\xi^3) = 1 \qquad f_i(\xi^1,\xi^2,\xi^3) = \sum_j e_j(\xi^1,\xi^2,\xi^3,)\lambda_{ij}$$

$$\frac{\partial x^i}{\partial \xi^j}\frac{\partial \xi^j}{\partial x^k} = \delta^i_k \qquad g^{ij} = \frac{\partial \xi^i}{\partial x^k}\frac{\partial \xi^j}{\partial x^k}$$

$$\left\{ {k \atop ij} \right\} = \frac{\partial^2 x^s}{\partial \xi^i \partial \xi^j}\frac{\partial \xi^k}{\partial x^2} \qquad \psi_{,ij} = \frac{\partial^2 \psi}{\partial \xi^i \partial \xi^j} - \left\{ {k \atop ij} \right\}\frac{\partial \psi}{\partial \xi^k}$$

Near a mapping singularity we consider first the general linear field

$$\psi = A + B_p x^p$$

and we want the numerical formulae to satisfy at all points the equations

$$\frac{\partial \psi}{\partial x^p} = B_p \quad , \quad g^{ij}\frac{\partial \psi}{\partial \xi^i}\frac{\partial \psi}{\partial \xi^j} = B_k B_k \quad , \quad \psi_{,ij} = 0$$

These conditions are satisfied at all points near any mapping singularity provided that

$$X^i = f_i \quad \text{for } i = 1,2,3 \qquad \psi = f_4$$

$$\text{and} \quad \lambda_{4j} = A + \sum_{p=1}^{3} B_p \lambda_{pj}$$

This conservation depends on

1 The use of the same basis functions for X^p and ψ.
2 Derivation of the inverse derivations from the explicit derivative by matrix inversion at each point.
3 The use of the 'flat space' formula for $\left\{ \begin{matrix} k \\ ij \end{matrix} \right\}$

The conservation of intrinsic derivatives of linear functions is not valid near mapping singularities using:-

1 A mixture of analytic, B spline and finite difference derivatives
2 Interpolated values of the inverse derivatives
3 The general tensor formula for $\left\{ \begin{matrix} k \\ ij \end{matrix} \right\}$

Subject to such basis function conventions we require to improve the accuracy in the numerical treatment of quadratic functions of the form:-

$$\psi = A + B_p X^p + C_{p2} X^p X^2$$

With the use of a common set of basis functions there is a close connection between the accuracy of computed errors in the field equations and accuracy of the computed unit normal near a mapping singularity on a spherical surface. Using basis function conventions the error of the unit normal should be zero for a plane surface and small for a spherical surface.

Integer Spline Mapping

Let y(u) be the integer spline generating function

W_{ipqr} be the spline weights
Q be the order of the polynomical segments of y
R be the order of derivatives continuous between segments of y
E be the order of operand for which a spline fit is exact
T be the number of non-zero segments of y
g_i be a tripolynomical operand of order Q

The tripolynomical operand is defined by

$$g_i(\xi^1,\xi^2,\xi^3) = \sum_{a=0}^{E}\sum_{b=0}^{E}\sum_{c=0}^{E} A_{iabc}(\xi^1)^a(\xi^2)^b(\xi^3)^c$$

The integer spline system is applicable to a simple Cartesian grid with unit grid interval in parametric space ξ^1,ξ^2,ξ^3.

For such a grid the basis functions and their coefficients are defined by

$$e_j(\xi^1,\xi^2,\xi^3) = y(\xi^1-p)y(\xi^2-q)y(\xi^3-r)$$

$$\lambda_{ij} = W_{ipqr} \qquad j = j(p,q,r)$$

The generating function y consists of T non-zero segments where each segment is a polynomial of order Q extending over a unit interval. The function y is symmetric about the origin and derivatives of order R are continuous between segments.

For an operand of some order E the fit is exact and all derivatives are exact at all points. For a general operand the functions f_i are tripolynomial functions of order Q within each mapping cell where the mapping cell boundaries are

$$\xi^s = i \quad \text{for T even} \qquad \text{for all integer i}$$
$$\xi^s = i+1/2 \quad \text{for T odd}$$

Regarding a function as the derivative of order zero of that function the general explicit derivative within a mapping cell is expanded in the form

$$\frac{\partial^l}{\partial(\xi^1)^l}\frac{\partial^m}{\partial(\xi^2)^m}\frac{\partial^n}{\partial(\xi^3)^n} f_i = \sum_j \frac{d^l y(\xi^1-p)}{d(\xi^1)^l}\frac{d^m y(\xi^2-1)}{d(\xi^2)^m}\frac{d^n y(\xi^3-r)W_{ipqr}}{d(\xi^3)^n}$$

with

$$\frac{\partial^R}{\partial(\xi^1)^R}\frac{\partial^Q}{\partial(\xi^2)^Q}\frac{\partial^Q}{\partial(\xi^3)^Q} f_i$$

continuous across both $\xi^1 = i$ and $\xi^1 = i + 1/2$

Comparison of Generating Functions

Let $Y_1(u)$ = if $-1/2 < u < 1/2$ then 1 else 0

and $Y_{T+1}(u) = \int_{-1/2}^{+1/2} Y_T(u/v)\,dv$

i.e. Y_T is the T^{th} power of convolution of the unit square pulse

Let Z(u) be a function with 4 non-zero segments fitted to the following values at the integer values of u:-

TABLE 2 FUNCTION

u	-2	-1	0	+1	+2
Z(u)	0	-1	+2	-1	0
dZ(u)/du	0	0	0	0	0
$d^2Z(u)/du^2$	0	0	0	0	0

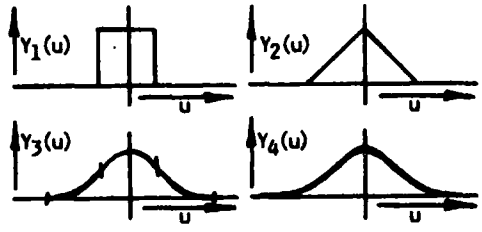


Fig 15. Form of the Y Functions

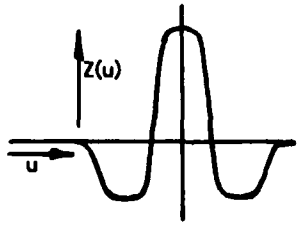


Fig 16. Form of the Z Function

We will compare four generating functions that provide the basis for the three and five point finite difference schemes (3PTFD and 5PTFD) and the cubic and quintic B spline schemes (CBS and QBS). For one dimensional calculations of the point values, first derivatives, and second derivatives at the grid points, the corresponding operators P, F, S, applied to the weights can be expanded in standard finite difference notation for each of the generating functions. We then have:-

TABLE 3 CHARACTERISTICS OF FOUR DISCRETISATION SCHEMES

Scheme	y	Q	R	E	T	P	F	S
3PTFD	Y4 + Z/6	5	2	2	4	1	$\mu\delta$	δ^2
CBS	Y4	3	2	3	4	$1+\delta^2/6$	$\mu\delta$	δ^2
5PTFD		5	2	4	6	1	$(1-\delta^2/6)\mu\delta$	$(1-\delta^2/12)\delta^2$
QBS	Y6	5	4	5	6			

For 3-D applications the operators P, F, S applied parallel to the parametric axis ξ^s are denoted by P_s, F_s, S_s respectively. The formulae for typical derivatives are then expanded:-

TABLE 4
THREE DIMENSIONAL DERIVATIVES

Derivative	Finite Difference	B Spline Form A	B Spline Form B
f_i	f_i	$P_1 P_2 P_3 W_i$	f_i
$\partial f_i/\partial \xi^1$	$F_i f_i$	$F_1 P_2 P_3 W_i$	$F_1 P_1^{-1} f_i$
$\partial^2 f_i/\partial (\xi^1)^2$	$S_1 f_i$	$S_1 P_2 P_3 W_i$	$S_1 P_1^{-1} f_i$
$\partial^2 f_i/\partial \xi^{11} \xi^2$	$F_2 F_3 f_i$	$P_1 F_2 F_3 W_i$	$S_2 P_2^{-1} S_3 P_3^{-1} f_i$

In a relaxation sequence the weights are updated using the residual errors r_i and the relaxation factor ϵ:-

$$Wi_{n+1} = W_{in} + \epsilon r_i$$

For example in the relaxation coverging to the solution of

$$\sum_s S_s P_s^{-1} F_i = 0$$

the relaxation sequence can be expanded as

$$f_{in+1} - f_i^{\infty} + \epsilon \sum_s S_s P_{s+1} P_{s+2} (d_{in} - f_i^{\infty})$$

and this expression can be further expanded using tri-harmonic operands of the form

$$gi = e^{i(\omega_1 \xi^1 + \omega_2 \xi^2 + \omega 3 \xi^3)}$$

For harmonic operands the ratio of the computed values of and $S_1 P_1^{-1} g_i$ and $F_1 P_1^{-1})^2 gi$ to the exact analytic derivative over the ω_1 range as shown in fig 17.

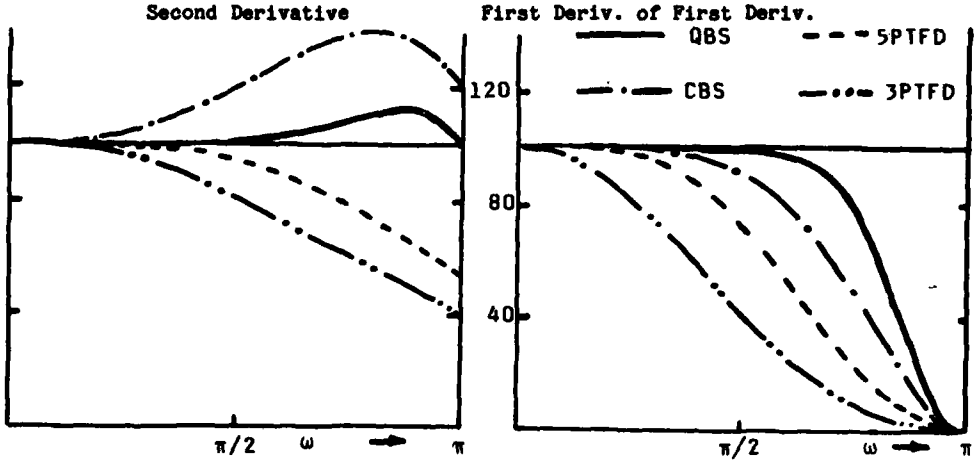


Fig 17 Derivatives of Cos (ωx) as Percentage of the Exact Derivatives

These equations may be interpreted as follows:-

The CBS scheme uses tricubic mapping cells to achieve precise values of all derivatives at all points for a tricubic operand. The 3PTFD system is a fast approximation to CBS: the replacement of $1 + \delta^2/6$ by 1 in the expansion of P can produce computing times for the calculation of derivatives reduced relative to CBS by up to 70%.

Conditions especially favourable to 3PTFD are the solution of transonic potential flow in 2-D using an orthogonal grid. In this case the accuracy of numerical first derivatives is not critical. The change from CBS to 3PTFD is achieved without reduction in the order of accuracy of second derivatives. Errors are dominated by the 'upwinding' term of the order δ^3 so that the reduction of errors of the order δ^4 by the use of 5PTFD is not necessary. Relaxation can be based on line and block relaxation methods.

Since $F_1 F_1 \cos(\omega\xi^1) = S_1 \cos(2\omega\xi^1)$

methods based on first derivatives require eight times as many grid points for the same accuracy as methods based on second derivatives with conformol mapping where the most critical first derivatives are known exactly.

On the other hand, for 3-D potential flow the accuracy of numerical first derivatives is critical for the treatment of non-orthogonality of the grid and for terms with non-zero values of V^j. For first derivatives the CBS scheme uses an operator of the same order as 5PTFD. However, accuracy of 5PTFD has been recovered by the use of remote samples and the ω range over which this accuracy is held is less than that for CBS. A seven point operator would be needed to match CBS accuracy for first derivatives over the harmonic spectrum.

Compared at constant band width the reduction in computing time for the finite difference system can be less than 25%. In a relaxation sequence the critical condition for the selection of ε is $\omega_1 = \omega_2 = \omega_3 = \pi$ for the finite difference system and $\omega_1 = \pi$, $\omega_2 = \omega_3 = 0$ for the B spline system due to the replacement of $\sum_s S_s$ by $\sum_s S_s P_{s+1} P_{s+2}$. This can imply a value of ε for the B spline system three times that for the finite difference system.

For a solution of the Euler equations by time stepping no numerical second differences are required and no upwinding errors are necessary. The correct selection of latent shock surface requires accurate modelling of the wave front with zero rate of propagation. The three point operators of CBS match the ω band width for the second derivatives of five point finite difference scheme and for the first derivatives of a seven point finite difference scheme. The five point operators of the QBS system provide an even wider ω band width. By suitable application of errors to weights the maximum Courant number can be raised by a factor of $\sqrt{3}$.

Thus 2-D potential flow using conformal mapping is especially favourable to the 3PTFD scheme. In most other cases the wide band accuracy of B spline schemes and their effect on rates of convergence are significant. Experience with grid relaxation demonstrates that the predicted relaxation factors can be achieved in the final convergence phase.

Topological Singularities

In a topologically regular grid, numerical interpolation and differentiation operators which produce results within a patch require values within a work zone that extends some distance H beyond the patch boundaries in all directions. The distance H is the halo width given by:-

TABLE 5 THE HALO WIDTH

	grid point operators	interpolation operators
T Even	T/2 -1	T/2
T Odd	(T-1)/2	(T-1)/2

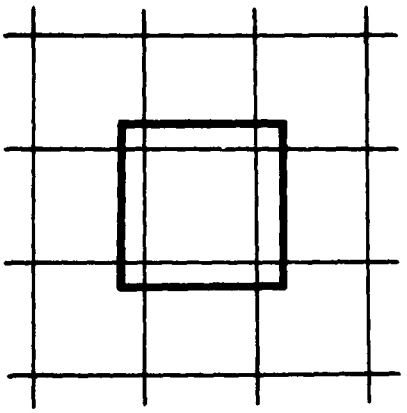


Fig 18. A Work Zone in an Infinite Grid

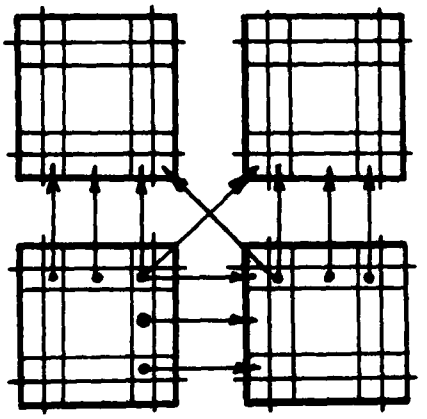
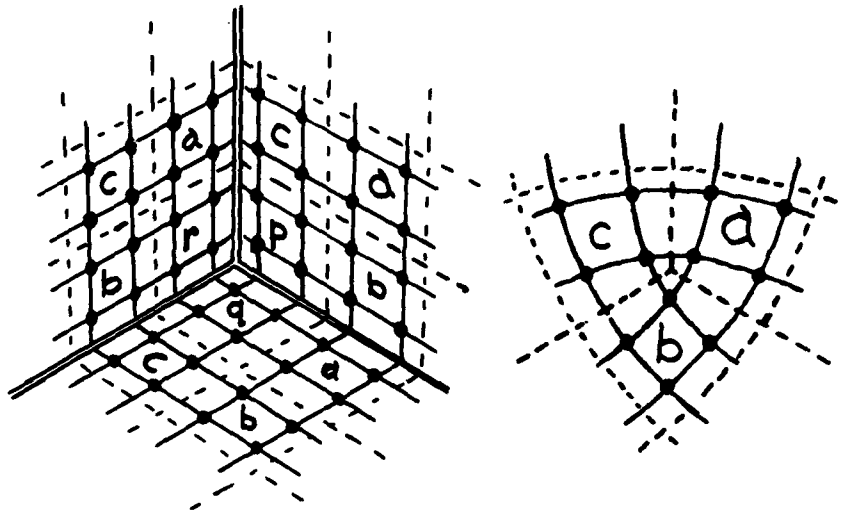


Fig 19. Copy Core Elements into Halo Zones

The numerical operators are always applied to a work zone as if the halo zone represented grid points in adjacent volumes in a topologically regular grid. All grid points in the core zone of the work zone represent grid points within the patch and all values associated with such grid points are regarded as independent variables.

The edges of a core zone consist of overlapping zones of width H within the core zones. The corners of the core zone are the intersections of the edge zones. Edge zones values are always copied into edge zones of the halo of an adjacent carpet. With T even this is sufficient to ensure analytic continuity across a partition line except for parametric distance (T-1)/2 from the node point at each patch corner. At all regular node points the values in a corner of a core zone are copied into a corner of the halo of the diagonally opposite carpet. For T odd this is sufficient to ensure analytic continuity up to the node points.

Where N carpets meet at a node point with $N \neq 4$ the set of N adjacent corner zones of the halo of the adjacent carpets are paired collectively with the set of N adjacent corner zones of the core zones of the adjacent carpets.



EXPLODED VIEW PHYSICAL SPACE

Fig 20. Work Zones near a Singularity

In the exploded view of the N work zones the grid point values in the zones a,b,c -- are each copied into the edge zones of the halos of two adjacent carpets. We then require to determine the values in the N halo corners p,q,r--

The basic function conventions then permit the halo corner values to be defined in matrix format

Then

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \quad B \quad \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

Then

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} A^{-1} \times B \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

The square matrices with dimensions NH^2, NH^2 represent a selected set of continuity conditions modified to ensure that A is not a singular matrix. The column matrices have dimensions NH^2,M where M is the number of f_i functions. The first set of continuity conditions is:-

TABLE 6 CONTINUITY RANGES

Parametric distance from node		Continuity along grid lines
T = ± 4	0	f_i
T = ± 6	1/2	$\partial^2 f_i / \partial(\xi^1)^2$
T = ± 8	3/2	$\partial^4 f_i / \partial(\xi^1)^4$

From consideration of symmetry and antisymmetry for any definition of P,F,S, there are always H redundant equations in this system. To satisfy basis functions conventions we require that when all elements of a,b,c are unity then all elements of p,q,r are also unity. The condition that the sum of the radial tangential vectors at the node point is zero is virtually mandatory. Minor violations of this constraint lead to violent excursions in double curvature very near the node points with little effect elsewhere. There are then H-1 auxiliary solutions and an empirical parameter is used to control tuning for double curvature effects.

An alternative system of contraints drops one order on the continuity at half a parametric interval and introduces the constraint that a common limit of Jn is approached at the node point for every carpet. This system requires one empirical parameter to tune for double curvature and another to control J. This form has the merit that surfaces can be transferred directly to AD2000 and other panel systems.

In field solutions the expansion of ψ about the singularity can be expanded in the form

$$\psi = A + B_i X^i + C_{ij} X^i X^j + D_{ijk} X^i X^j X^k + +$$

For any values of A and B_i modelling throught the singularity should be exact when all the other coefficients are zero. With the CBS scheme it should be possible to neutralise the total source strength for unit length due to the contribution from C_{ij}, for the region near a singularity line independently for N = 3,5,6.

The total doublet, quadrupole and higher order violations of conservation should have effects that subside at least as rapidly as the inverse second power of distance from the singularity line.

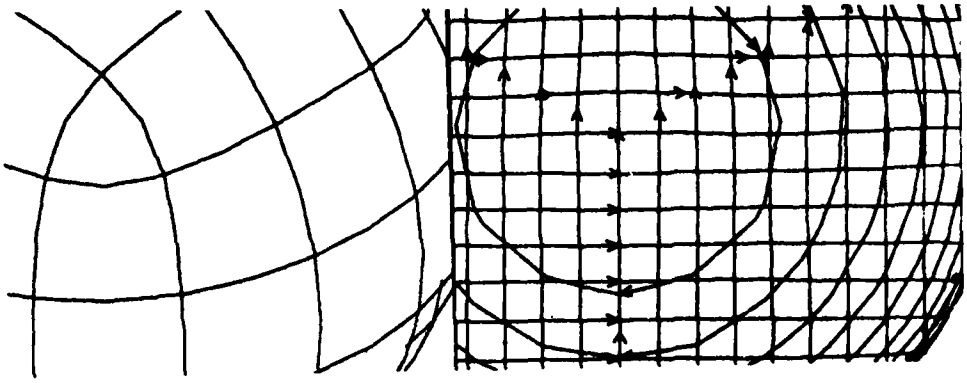


Fig 21 Grid Lines and Direction Cosine Contours.

Concluding Remarks

The MVDS data base is the most versatile convention compatible with generalised B spline mapping. The present cryptic language does not realise the full versatility of the data base but it appears to be adequate for a range of practical problems.

As a grid point technique the B spline system is slow for the standard bench mark cases. However, for 3-D relaxation and Euler solutions the superior harmonic band width, numerical stability and accuracy for first derivatives are relevant. Used with topologically complicated computing grids the alternative is the finite volume technique which is no more accurate than the finite difference system away from mapping singularities and has no special treatment of mapping singularities.

The MVDS system therefore, is suitable as a replacement for the existing facilities which locate mapping singularities at critical regions of the aircraft surfaces.

ACKNOWLEDGEMENTS
The programming in the work described was done by Clive Stops, British Aerospace, Weybridge

List of References

1. Detyna, E. (1981) Variational Principles and Gauge Theory. An Application to continuous Media. Department of Mathematics, University of Reading.

2. Roberts, A. (1980) The treatment of Shocks in Fast Solving Methods. In Numerical Methods in Applied Fluid Dynamics, Hunt, B. ed., Academic Press

3. Lipschutz, M. (1969) Schaum's Outline Series Differential Geometry McGRAW HILL Book Company.

AD P000986

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

THE NUMERICAL DIFFERENTIATION OF DISCRETE FUNCTIONS USING POLYNOMIAL INTERPOLATION METHODS

JAMES M. HYMAN[+] AND BERNARD LARROUTUROU[++]
[+]Center for Nonlinear Studies, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA; [++]Laboratoire Central des Ponts et Chaussees, 58 Bd Lefebvre, 75732 Paris, Cedex 15, France

ABSTRACT

A FORTRAN subroutine package, called DERMOD, has been written for calculating numerical approximations for the spatial derivatives of a function defined only on a discrete set of data points. The routines are designed to complement many existing codes for solving ordinary and partial differential equations and for the interpolation of tabular data. We describe some new numerical differentiation algorithms, discuss a mapping procedure for nonuniform grids, and explain the program methodology used in designing the software.

I. INTRODUCTION

The accurate approximation of spatial derivatives is a crucial element in the numerical solution of partial differential equations (PDEs). The derivation and implementation of accurate differentiation formulas is an error-prone and tedious process. This is especially true in two and three dimensions on nonuniform grids. For these reasons we have written a subroutine package, called DERMOD, to help reduce the effort needed to accurately and reliably differentiate discretely defined functions.

A major advantage of the package, compared with the traditional approach for solving PDEs, is that state-of-the-art numerical methods can be used with a minimum of programming effort to approximate the derivatives in large complicated PDE systems. Furthermore, the resulting programs can easily be modified for a comparison of the relative accuracy and efficiency of the different methods for solving a particular problem. The production runs can then be made using the best available methods.

The numerical analyst will also benefit from this approach. Most of the developmental analysis for numerical methods is for simple linear systems. It is important to understand the behavior of a new method in a complex situation before recommending its use to the uninitiated. New methods can now be quickly tested on any PDE system discretized using the DERMOD package.

All the spatial differentiation methods we describe will follow the same algorithmic flow. A function is defined on a discrete set of mesh points in one, two, or three space dimensions. These discrete data sets are the input to a black box type subroutine in the DERMOD library. In this subroutine the approximation for the described derivative is calculated and returned to the user.

Thus, the spatial differentiation is totally divorced from the nonlinearities of the PDE, the boundary conditions, and the time integration method. This modularity also reduces the redundancy of programming the same approximation to the spatial derivatives each time they appear in an equation. These differentiation routines are designed for no specific PDE and need to be debugged and optimized for a particular machine only once.

The data structures of the grids allowed in the current version of DERMOD, listed in increasing order of complexity and computer cost, include:

- *one-argument grids: (tensor-product grids, Fig. 1a)* $\{x_i\}$, $\{x_i, y_j\}$, *and* $\{x_i, y_j, z_k\}$. *Uniform grids are a special case of one-argument grids,*
- *multiple-argument grids: (logically rectangular, Fig. 1b)* $\{x_i\}$, $\{x_{i,j}, y_{i,j}\}$, *and* $\{x_{i,j,k}, y_{i,j,k}, z_{i,j,k}\}$,
- *neighborhood grids: (Fig. 1c)* $\{x_\ell\}$, $\{x_\ell, y_\ell\}$, $\{x_\ell, y_\ell, z_\ell\}$, *and* $NBRS_\ell$. *These grids are typical of finite element simplex grids.*

The numerical differentiation methods described can be divided into two classes: interpolation methods and mapping methods.

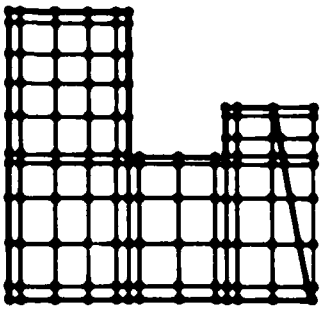
The interpolation method approximates the function with an interpolant (such as splines, or a local Lagrange polynomial), differentiates the interpolant, and evaluates the derivative at the desired location. These formulas are simple on uniform grids but are usually complicated on nonuniform grids. The interpolation method is not as sensitive to rough mesh variations as the mapping method.

In the mapping method, the nonuniform grid is mapped to a uniform reference grid. The derivatives on the nonuniform grid can then be expressed as products and sums of the derivatives of the function on the reference grid and the map. These derivatives on the reference grid are approximated using an interpolation method that is simple and efficient. The accuracy of the mapping method depends upon the smoothness of both the original function and the map. Therefore, the smoothness of the mesh variations in the nonuniform grid can strongly influence the accuracy of the derivative approximations.

One-argument grids

(x_i, y_j, z_k)

2-D Tensor-product grid

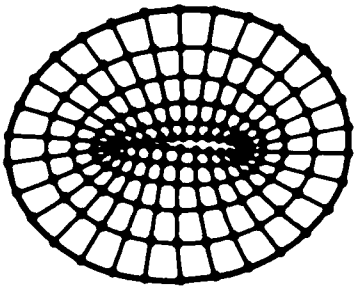


Multiple-argument grids

$(x_{i,j}, y_{i,j})$

$(x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$

2-D logically rectangular grid



Neighborhood grids

$(x,y,z)_\ell$

XLOC(L)
YLOC(L)
ZLOC(L)
NBRS(L,)*

2-D triangular grid

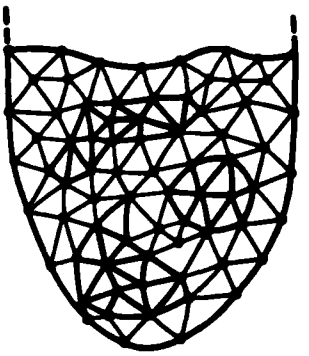


Fig. 1. Different 2-D grid data structures.

After first describing the interpolation method in greater detail we will discuss the mapping method, explain the program methodology used in designing the software, and then provide instructions for using the DERMOD subroutines effectively.

II. INTERPOLATION METHODS

In the interpolation method one can either construct a local interpolation function based only on data points near where the derivative is desired or, by using all the data points, construct a global interpolant. Usually the local interpolants are simpler and more appropriate for handling sharp gradients and nonuniform meshes, but they are less accurate than the global interpolants. In this report we will describe only the local polynomial interpolation methods. In a later report we plan to describe on the implementation of global interpolants based on the Fourier transform or Chebyshev polynomials.

The simplest local interpolation method to approximate derivatives fits a Lagrange or least-squares polynomial through the data at the nearby mesh points, differentiates the polynomial and evaluates it at the desired mesh point.[1-8] This results in an approximation for the k-th derivative of f at x_i, denoted by $f^{[k]}(x_i)$, that is a linear combination of the nearby function values,

$$f^{[k]}(x_i) = \Sigma\, a_j f(x_j)/(\text{constant} \times h^k).$$

On uniform grids these formulas are relatively simple and are called finite difference methods. Table 1 lists some common finite difference formulas used in DERMOD. The centered difference formulas are used whenever possible and the uncentered ones are used only near boundaries when too few function values are available to use a centered scheme.

When the data is smooth, the high order Lagrange interpolation formulas in Table 1a usually provide better accuracy on a given grid than do the lower order formulas.

When the data is rough, the sensitivity of the derivative approximations to noise can be more important than the order of accuracy of the method. For these problems the least-squares formulas[9] in Table 1b are preferred over the Lagrange formulas. These formulas are derived by fitting a least-squares polynomial through the data points of degree two less than the Lagrange polynomial.

TABLE 1a
LAGRANGE APPROXIMATIONS ON EQUALLY-SPACED GRIDS, $ch^k f^{[k]} = \Sigma\, a_j f(x_i)$

Derivative	a_{i-3}	a_{i-2}	a_{i-1}	a_i	a_{i+1}	a_{i+2}	a_{i+3}	a_{i+4}	a_{i+5}	Accuracy
$2hf_x(x_i)$			-1		1					$O(h^2)$
$2hf_x(x_i)$				-3	4	-1				$O(h^2)$
$12hf_x(x_i)$		1	-8		8	-1				$O(h^4)$
$12hf_x(x_i)$			-3	-10	18	-6	1			$O(h^4)$
$12hf_x(x_i)$				-25	48	-36	16	-3		$O(h^4)$
$60hf_x(x_i)$	-1	9	-45		45	-9	1			$O(h^6)$
$60hf_x(x_i)$		2	-24	-35	80	-30	8	-1		$O(h^6)$
$hf_x(x_{i+\frac{1}{2}})$				-1	1					$O(h^2)$
$24hf_x(x_{i+\frac{1}{2}})$			1	-27	27	-1				$O(h^4)$
$24hf_x(x_{i+\frac{1}{2}})$				-23	21	3	-1			$O(h^4)$
$1920hf_x(x_{i+\frac{1}{2}})$		-9	125	-2250	2250	-125	9			$O(h^6)$
$h^2f_{xx}(x_i)$			1	-2	1					$O(h^2)$
$h^2f_{xx}(x_i)$				2	-5	4	-1			$O(h^2)$
$12h^2f_{xx}(x_i)$		-1	16	-30	16	-1				$O(h^4)$
$12h^2f_{xx}(x_i)$			10	-15	-4	14	-6	1		$O(h^4)$
$12h^2f_{xx}(x_i)$				35	-104	114	-56	11		$O(h^3)$
$180h^2f_{xx}(x_i)$	2	-27	270	-490	270	-27	2			$O(h^6)$
$180h^2f_{xx}(x_i)$		-13	228	-420	200	15	-12	2		$O(h^5)$
$2h^3f_{xxx}(x_i)$		-1	2		-2	1				$O(h^2)$
$2h^3f_{xxx}(x_i)$			-3	10	-12	6	-1			$O(h^2)$
$2h^3f_{xxx}(x_i)$				-5	18	-24	14	-3		$O(h^2)$
$h^4f_{xxxx}(x_i)$		1	-4	6	-4	1				$O(h^2)$
$h^4f_{xxxx}(x_i)$			2	-9	16	-14	6	-1		$O(h^2)$
$h^4f_{xxxx}(x_i)$				3	-14	26	-24	11	-2	$O(h^2)$

TABLE 1b
LEAST-SQUARES APPROXIMATIONS ON EQUALLY-SPACED GRIDS, $ch^k f^{[k]} = \Sigma\, a_j f(x_j)$

Approximation	a_{i-3}	a_{i-2}	a_{i-1}	a_i	a_{i+1}	a_{i+2}	a_{i+3}	a_{i+4}	a_{i+5}	a_{i+6}	Accuracy
$35f(x_i)$		-3	12	17	12	-3					$O(h^4)$
$35f(x_i)$			9	13	12	6	-5				$O(h^3)$
$35f(x_i)$				31	9	-3	-5	3			$O(h^3)$
$10hf_x(x_i)$		-2	-1		1	2					$O(h^2)$
$70hf_x(x_i)$			-34	30	20	17	-6				$O(h^2)$
$70hf_x(x_i)$				-54	13	40	27	-26			$O(h^2)$
$7h^2f_{xx}(x_i)$		2	-1	-2	-1	2					$O(h^2)$
$231f(x_i)$	5	-30	75	131	75	-30	5				$O(h^6)$
$462f(x_i)$		-35	155	212	150	25	-65	20			$O(h^5)$
$462f(x_i)$			25	356	155	-60	-65	70	-19		$O(h^5)$
$462f(x_i)$				456	25	-35	10	20	-19	5	$O(h^5)$
$252hf_x(x_i)$	22	-67	-58		58	67	-22				$O(h^4)$
$2772hf_x(x_i)$		158	-1619	-50	1218	764	-607	136			$O(h^4)$
$252hf_x(x_i)$			-104	-25	68	84	16	-59	20		$O(h^4)$
$2772hf_x(x_i)$				-4420	5059	1504	-2394	-1378	2375	-746	$O(h^4)$
$132h^2f_{xx}(x_i)$	-13	67	-19	-70	-19	67	-13				$O(h^4)$
$132h^2f_{xx}(x_i)$		27	3	-35	-34	9	47	-17			$O(h^3)$
$132h^2f_{xx}(x_i)$			103	-145	-39	74	49	-57	15		$O(h^3)$
$132h^2f_{xx}(x_i)$				215	-377	-31	254	101	-245	83	$O(h^3)$

On an equally spaced mesh it is easily seen that the centered first derivative approximations in Table 1 are conservative.[10] On an unequally spaced mesh the interpolation formulas are, in general, not conservative.

To compute more complicated derivatives such as f_{xy} or $(df_x)_x$ the formulas are applied in a two-step process that ensures that the resulting formula will be as compact as possible. For example, to compute $(df_x)_x$, first f_x is computed at the half-points $x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1})$. Then d is defined at these points using the harmonic mean,

$$d_{i+\frac{1}{2}} = \left[\frac{1}{\Delta x_{i+\frac{1}{2}}} \int_{x_i}^{x_{i+1}} d^{-1}(x)\, dx \right]^{-1} \doteq 2d_i d_{i+1}/(d_i + d_{i+1}) \quad .$$

The harmonic rather than the arithmetic mean is used in order to preserve the flux continuity (df_x) accross discontinuities in d.[11] The product df_x is then differentiated and evaluated at the mesh points. On nonuniform grids special care must be taken because the centers of the midpoints are not the mesh points.

The three-point derivative approximations on nonuniform grids using a parabolic interpolant are listed in Table 2. The five-point quintic Lagrange interpolation methods are straightforward[1] and are also available in the

TABLE 2
QUADRATIC APPROXIMATIONS TO f_x AND f_{xx}

$$f_x(x_i) \doteq (\Delta x_{i-1/2}\, S_{i+1/2} + \Delta x_{i+1/2}\, S_{i-1/2})/(\Delta x_{i+1/2} + \Delta x_{i-1/2})$$

$$f_x(x_i) \doteq [(2\Delta x_{i+1/2} + \Delta x_{i+3/2})\, S_{i+1/2} - \Delta x_{i+1/2}\, S_{i+3/2}]/(\Delta x_{i+1/2} + \Delta x_{i+3/2})$$

$$f_{xx}(x_i) \doteq 2(S_{i+1/2} - S_{i-1/2})/(\Delta x_{i+1/2} + \Delta x_{i-1/2})$$

where

$$\Delta x_{i+1/2} = x_{i+1} - x_i$$

$$S_{i+1/2} = \Delta f_{i+1/2}/\Delta x_{i+1/2} \quad .$$

package. The coefficients a_j for the quintic interpolation methods are computed and saved on the first call to the package. By using this information, later derivative calculations on the same nonuniform grid cost little more than the approximations on an equally spaced grid.

On the multiple-argument or neighborhood grids the local Lagrange interpolant is more cumbersome and frequently there is no unique formulation. For example, in two dimensions, the typical Lagrange quadratic interpolant is uniquely defined with six data points, but the (i,j)-th mesh point in two-argument grid has nine data points next to it. A possible approach is illustrated in Fig. 2. First, an orthogonal (x,y) coordinate system is set up and f is interpolated linearly to give values at the on-axis points A, B, C, and D using the function values at the neighboring points. The one-argument grid quadratic interpolation formulas are then used. This procedure, implemented in DERMOD, is not as accurate as it could be, since the interpolated values are only $O(h^2)$. The first derivative approximations are only O(h) accurate and the second derivative approximations may be only O(1), and thus they may be inconsistent.

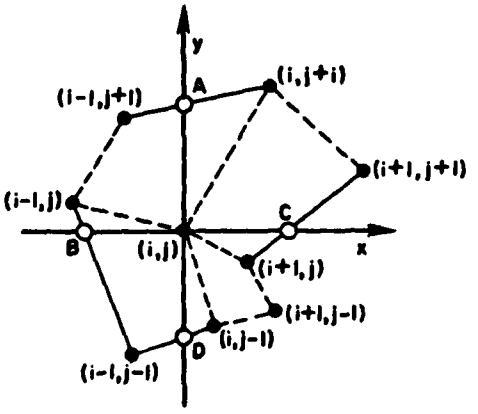


Fig. 2a. Two-argument grid.

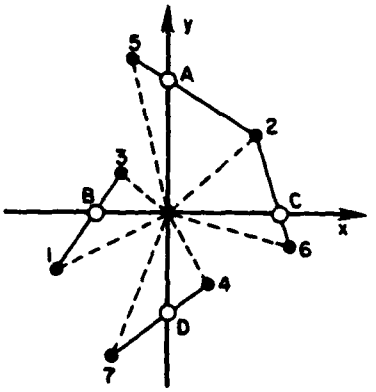


Fig. 2b. Neighborhood grid.

Fig. 2. Interpolation to an underlying orthogonal reference grid.

We have considered two other approaches to overcome this dilemma on multiple-argument and neighborhood grids. The first is to fit a least-squares quadratic polynomial through the nearby data points and differentiate it at the desired location. We expect this method to be accurate and stable. We are currently implementing this approach in DERMOD.

The lumped finite element method is another interpolation method that can also be used to generate local approximations to the derivatives. These formulas work well on uniform grids, but appear to have little advantage over the least squares approach on rough grids. In fact, using a triangulation of the grid in Fig. 2, the approximations thus generated to the second derivatives are pointwise inconsistent. The Minerbo approximation to the Laplacian[12] is an accurate special formula that avoids this inconsistancy.

III. MAPPING METHODS

A simpler approach to numerical differentiation on nonuniform grids is the mapping method. In the mapping method, the physical mesh points (x,y,z) are mapped to reference mesh points (ξ,η,ζ). The derivatives in the physical space are then expressed in terms of the derivatives of the map, called the mesh metrics, and the derivatives of the function on the reference grid.

The mapping method in one space dimension[13] is always nonsingular since the nonuniform mesh $\{x_i\}$ forms a strictly monotonic sequence. That is, there exists a one-to-one map from x_i onto the reference grid ξ_i (for example, $\xi_i = i$). The derivatives of a function defined on $\{x_i\}$ can then be expressed as

$$f_x = f_\xi \xi_x = f_\xi / x_\xi \tag{4.1}$$

and

$$f_{xx} = f_{\xi\xi}\xi_x^2 + f_\xi \xi_{xx} = f_{\xi\xi}(x_\xi)^{-2} + \tfrac{1}{2}f_\xi((x_\xi)^{-2})_\xi \ . \tag{4.2}$$

The derivatives on the right side of these equations are derivatives on the reference mesh. These can be approximated with any of the interpolation methods, all of which have a much simpler formulation on regular reference grids. For example, if fourth-order Lagrange finite differences are used in (4.1), then

$$f_x(x_i) \doteq \left(\frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta\xi}\right)\left(\frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12\Delta\xi}\right)^{-1}$$

$$= \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}$$

Note that the reference grid points need not be evenly spaced. They could, for example, be the Gauss or Chebyshev points depending upon the interpolating functions being used.

Some smoothness in the function being approximated is assumed in deriving all the high order differentiation formulas. For this reason, the order of accuracy is bounded by the smoothness of f, the smoothness of the map, and the order of accuracy of the differentiation formula on the uniform mesh. Therefore the mapping methods are usually less accurate than the interpolation methods on grids with nonsmoothly varying mesh spacing.

Analytically, ξ_x can never vanish. However, on rough grids (where the mapping method is inappropriate) the numerical approximation to ξ_x may vanish or, equally bad, change sign. When this occurs, either an interpolation method or a lower order mapping method should be used.

On two-argument grids the formulas are more complicated,[14] but the derivatives can still be expressed as a function of the derivatives on a regular reference grid and the mesh metrics of the map from the physical (x,y) grid to the reference (ξ,η) grid. For this case we have

$$\begin{bmatrix} f_x \\ f_y \\ f_{xx} \\ f_{xy} \\ f_{yy} \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & 0 & 0 & 0 \\ \xi_y & \eta_y & 0 & 0 & 0 \\ \xi_{xx} & \eta_{xx} & \xi_x^2 & 2\xi_x\eta_x & \eta_x^2 \\ \xi_{xy} & \eta_{xy} & \xi_x\xi_y & \xi_x\eta_y + \xi_y\eta_x & \eta_x\eta_y \\ \xi_{yy} & \eta_{yy} & \xi_y^2 & 2\xi_y\eta_x & \eta_y^2 \end{bmatrix} \begin{bmatrix} f_\xi \\ f_\eta \\ f_{\xi\xi} \\ f_{\xi\eta} \\ f_{\eta\eta} \end{bmatrix} \tag{4.3}$$

Using the Jacobian J of the map and its derivatives,

$$J = x_\xi y_\eta - x_\eta y_\xi \ ,$$

$$J_\xi = x_{\xi\xi} y_\eta + x_\xi y_{\xi\eta} - x_{\xi\eta} y_\xi - x_\eta y_{\xi\xi} \ ,$$

and

$$J_\eta = x_{\xi\eta} y_\eta + x_\xi y_{\eta\eta} - x_{\eta\eta} y_\xi - x_\eta y_{\xi\eta} \ ,$$

the mesh metrics can be easily expressed as derivatives on the (ξ,n) reference grid.

$$\xi_x = y_\eta / J \quad ,$$

$$\xi_y = -x_\eta / J \quad ,$$

$$\eta_x = -y_\xi / J \quad ,$$

$$\eta_y = x_\xi / J \quad ,$$

$$\xi_{xx} = (-J_\xi y_\eta^2 + J y_\eta y_{\xi\eta} + J_\eta y_\xi y_\eta - J y_\xi y_{\eta\eta}) / J^3 \quad ,$$

$$\xi_{xy} = (J_\xi x_\eta y_\eta - J x_{\xi\eta} y_\eta - J_\eta x_\eta y_\xi + J x_{\eta\eta} y_\xi) / J^3 \quad ,$$

$$\xi_{yy} = (-J_\xi x_\eta^2 + J x_\eta x_{\xi\eta} + J_\eta x_\xi x_\eta - J x_\xi x_{\eta\eta}) / J^3 \quad ,$$

$$\eta_{xx} = (-J_\eta y_\xi^2 + J y_\xi y_{\xi\eta} + J_\xi y_\xi y_\eta - J y_{\xi\xi} y_\eta) / J^3 \quad ,$$

$$\eta_{xy} = (J_\eta x_\xi y_\xi - J x_{\xi\eta} y_\xi - J_\xi x_\xi y_\eta + J x_{\xi\xi} y_\eta) / J^3 \quad ,$$

and

$$\eta_{yy} = (-J_\eta x_\xi^2 + J x_\xi x_{\xi\eta} + J_\xi x_\xi x_\eta - J x_{\xi\xi} x_\eta) / J^3 \quad .$$

If the derivatives of many different functions must be calculated on the same mesh, then the mesh metrics need only be calculated once and saved. Usually this means after the initial derivative calculation, additional derivatives on the same nonuniform mesh, using the mapping method, cost only slightly more than derivative approximations on a uniform mesh. The package does this automatically.

The mapping method for three-argument grids is similar to two-argument grids but this has not been implemented in DERMOD, yet.

IV. SOFTWARE DESIGN

In designing DERMOD we placed a priority on making the code reliable, modular and easy to use. All the programs were extensively documented and verified as the code was developed and performance claims were tested and reconfirmed. During execution, the input is consistently checked for reasonability.

The routines are modular and as independent of each other as possible. Minimal internal communication allows sophisticated users to easily experiment and modify a subroutine for special purposes without causing unexpected errors to ripple through the other routines. Modularity also allows the programs using DERMOD to be easily upgraded and to make use of improved methods and implementations as they become available. We anticipate that by using the package the sofware development time and maintenance of codes may be significantly reduced, especially for lengthy 3-D programs.

The machine architecture largely determines the efficiency of many of the numerical differentiation methods. We have opted for the routines to differentiate along one line at a time. This requires only one-dimensional work arrays and allows the code to be easily vectorized on machines such as the CRAY-1 and CDC Cyber-205.

The subroutine names in the package have six characters. These are chosen according to the following convention:

first letter:

A - define all the indicated derivatives
X - sweep in the x direction (first index sweep)
Y - sweep in the y direction (second index sweep)
Z - sweep in the z direction (third index sweep)

second letter:

1 - first derivative
2 - second derivative
3 - third derivative
4 - fourth derivative
X,Y,Z or L - see special cases listed below

third letter:

X - derivative in the x coordinate direction
Y - derivative in the y coordinate direction
Z - derivative in the z coordinate direction
C,D,R,S,Y or Z - see special cases listed below

fourth letter:

P - polynomial approximation (finite differences)
F - Fourier transform (pseudo-spectral method)
C - Chebychev transform (pseudo-spectral method)
R - rational function Padé approximation (implicit method)

fifth letter:

E - equally spaced grid
I - interpolation method (unequally spaced grid)
M - mapping method (unequally spaced grid)
G - Gauss points
C - Chebyshev points

sixth letter:

1 - one-argument (tensor product) grid X(I), Y(I), Z(K)
2 - two-argument grid X(I,J), Y(I,J)
3 - three-argument grid X(I,J,K), Y(I,J,K), Z(I,J,K)
T - triangular neighborhood grid X(L), Y(L) (two dimensions)
P - pyramid neighborhood grid X(L), Y(L), Z(L) (three dimensions)
H - one-argument staggered grid (derivatives at the half points)

special cases for the second and third letter:

XY - mixed xy derivative
XZ - mixed xz derivative
YZ - mixed yz derivative

LR - Laplacian in rectangular geometry ($u_{xx} + u_{yy} + u_{zz}$)

LC - Laplacian in cylindrical geometry ($x^{-1}(xu_x)_x + x^{-2}u_{yy} + u_{zz}$)
LS - Laplacian in spherical goemetry

XD - compute $(df_x)_x$
YD - compute $(df_y)_y$
ZD - compute $(df_z)_z$

For example, subroutine X2YPM2 computes the second derivative of f with respect to y, f_{yy}, at the mesh points along an X coordinate line using a local polynomial interpolant or finite difference method, listed in Table 1, after mapping the unequally spaced two-argument grid to a uniform grid. The mesh metrics are computed using the same order finite difference methods.

At this time the available routines are:
X1XPE1, X2XPE1, X3XPE1, X4XPE1, X1YPE1, X2YPE1, X3YPE1, X4YPE1, X1ZPE1, X2ZPE1, X3ZPE1, X4ZPE1, X1XPI1, X2XPI1, X1YPI1, X2YPI1, X1ZPI1, X2ZPI1, X1XPM1, X2XPM1, X1YPM1, X2YPM1, X1ZPM1, X2ZPM1, XXYPE1, XXZPE1, XXYPI1, XXZPI1, XXYPM1, XXZPM1, X1XPEH, X1YPEH, X1ZPEH, X1XPMH, X1YPMH, X1ZPMH, XXDPE1, XYDPE1, XZDPE1, XXDPI1, XYDPI1, XZDPI1, X1XPI2, X1YPI2, X1XPM2, X2XPM2, X1YPM2, X2YPM2, and XXYPM2.

The nomenclature used by the package will be useful in describing the capabilities of the routines. These variables (used in the above X sweep routines) and their meanings are:

Input Variables:

U - array of the function values to be differentiated

in one space dimension the function u(x) must be defined at U(I) where I is between NXBX and NXEX

in two space dimensions the function u(x,y) must be defined at U(I,J) where
I is between NXBX and NXEX
J is between NYBX and NYEX

in three space dimensions the function u(x,y,z) must be defined at U(I,J,K) where
I is between NXBX and NXEX
J is between NYBX and NYEX
K is between NZBX and NZEX

On neighborhood grids the function u must be defined at U(L) where L is between NLBX and NLEX.

X - the array containing the mesh point locations in the first coordinate diction. The element X(I) in one-argument grids, X(I,J) or X(I,J,K) on multiple-argument grids, or X(L) on neighborhood grids must be defined for the same indices I, J, K, or L as those where U is defined.

Y - the array containing the mesh point locations in the first coordinate diction. The element Y(J) in one-argument grids, Y(I,J) or Y(I,J,K) on multiple-argument grids or Y(L) on neighborhood grids must be defined for the same indices I, J, K, or L as those where U is defined.

Z - the array containing the mesh point locations in the first coordinate diction. The element Z(K) in one-argument grids or Z(I,J,K) on multiple-argument grids or Z(L) on neighborhood grids must be defined for the same indices I, J, K, or L as those where U is defined.

D - array of the diffusion coefficients for the second derivatives. This array must be defined where U is defined and has the same data structure as U.

NXBX - index of the first X point where U is defined.

NXB - index of the first X mesh point where the derivatives of U are to be calculated.

NXE - index of the last X mesh point where the derivatives of U are to be calculated.

NXBX - index of the last X point where U is defined.

NXD - dimension of the first index of U and the mesh arrays.

NYBX - index of the first Y point where U is defined.

NY - index of the Y mesh point where the derivatives of U are to be calculated.

NYEX - index of the last Y point where U is defined.

NYD - dimension of the second index of U and the mesh arrays.

NZBX - index of the first Z point where U is defined.

NZ - index of the Z mesh point where the derivatives of U are to be calculated.

NZEX - index of the last Z point where U is defined.

MORD - method order parameter. The method should be asymptotically MORD-th order

Workspace Variables:

IWS - index to indicate whether the work space array contains information on the grid such as the mesh metrics (IWS = 0 on first call using the grid, IWS = 1 on later calls).

WS - array of workspace used for internal calculations. This array may be input or output.

Output Variables:

U** - array of the derivatives defined at U**(I) for I between NXB and NXE. The second and third letters are the same as these in the subroutine-naming convention.

MORD - The derivative returned is asymptotically MORD-th order. This will be less than or equal to the requested value. MORD returns equal to zero if no calculation was possible.

The variables for the routines that sweep in the Y and Z lines are similarly named.

A sample program to compute the derivative of sin(x) for x between zero and one, using a sixth-order finite difference (polynomial interpolation) method is:

```
      DIMENSION X(11),U(11),U1X(11)
      NXBX=1
      NXEX=11
      DX=1.0/(NXEX-NXBX)
      DO 10 I=NXBX,NXEX
      X(I)=(I-1)*DX
   10 U(I)=SIN(X(I))
```

```
      MORD=6
      NXB=NXBX
      NXE=NXEX
      CALL X1XPE1(U,X,NXBX,NXB,NXE,NXEX,MORD,U1X)
      PRINT 20
   20 FORMAT("     X         U          U1X         ANS")
      DO 30 I=NXB,NXE
      ANS=COS(X(I))
   30 PRINT 40,X(I),U(I),U1X(I),ANS
   40 FORMAT(4F10.6)
      CALL EXIT
      END
```

The output is:

X	U	U1X	ANS
0.000000	0.000000	.999980	1.000000
.100000	.099833	.995009	.995004
.200000	.198669	.980067	.980067
.300000	.295520	.955336	.955336
.400000	.389418	.921061	.921061
.500000	.479426	.877583	.877583
.600000	.564642	.825336	.825336
.700000	.644218	.764842	.764842
.800000	.717356	.696707	.696707
.900000	.783327	.621613	.621610
1.000000	.841471	.540289	.540302

Note that in this example the derivative approximations near the boundaries where the uncentered difference formulas are used are less accurate than where centered differences can be used. These errors could be avoided by defining U on a domain greater than that of the desired derivatives such as: NXBX < NSB - 2 and NXEX > XNE + 2. This is also convenient when using fictitious points to incorporate the effects of the boundary conditions into a discrete approximation[15] and when constructing a local Hermite interpolant and sampling the interior of a table.

The workspace needed by the code is limited to one-dimensional arrays of length NXE. These arrays contain the finite-difference coefficients for a particular mesh line, and can be used to define the coefficient matrix needed in solving the linear systems arising from implicit methods. When the user's program is constrained by computer CPU time and not storage, then by saving workspace arrays of length NXE on one-argument grids or NXE·NYE on two-argument grids, the difference coefficients need only be computed once for the entire problem.

V. USAGE

We expect the package to be used most frequently for calculating derivatives directly for explicit approximations to differential equations[15] and for defect correction improvements of low-order implicit approximations.[16] When used this way a crude estimate of the error can be obtained by comparing the derivatives with those obtained by a different method or on a coarser grid. The structure of the package makes this easy to do.

The direct usage is straightforward; the function to be differentiated is defined at the mesh points and the derivatives are calculated as described in the example in the previous section. We expect this to be the most common usage for explicit integration methods for PDEs and for constructing interpolants.

The indirect defect correction usage occurs most often in the iterative solution of algebraic equations arising in the numerical approximation to differential equations. These equations occur in steady state or time independent problems and on each time step in the implicit integration of time dependent problems.

These systems can be written

$$A(v) - b = 0 \quad , \tag{5.1}$$

where A is a nonlinear discrete operator, b is a known vector, and the discrete solution vector is v. The sparseness of A depends upon the numerical differentiation method used. The high-order methods result in less sparse, more complicated systems than the lower-order methods.

Often the solution of Eq. (5.1) is difficult to obtain directly, but the residual error,

$$r = A(w) - b \tag{5.2}$$

for an approximate solution w, is easy to evaluate. In many complicated PDE problems, one is less likely to introduce errors in evaluating r than in constructing A and solving Eq. (5.1). This is particularly true for high-order approximations of nonlinear systems on irregular domains.

If there is a related system

$$P(w) - b = 0 \tag{5.3}$$

that approximates Eq. (5.1) and is easier to solve, the defect correction

algorithm may be appropriate. The operator P may be a lower order, simpler approximation to the same system.

Given an approximation v^n (where n is the iteration parameter) near a root v^{n+1} of Eq. (5.1), we can expand Eq. (5.1) using the Taylor series to get

$$\begin{aligned} 0 &= A(v^{n+1}) - b \\ &= A(v^{n+1}) - b + P(v^{n+1}) - P(v^{n+1}) \\ &= A(v^n) - b + P(v^{n+1}) - P(v^n) - (J_P - J_A)(v^{n+1} - v^n) + O(\varepsilon^2) \quad , \end{aligned} \tag{5.4}$$

where $\varepsilon = v^{n+1} - v^n$. The defect correction is any $O(\varepsilon)$ approximation to Eq. (5.4); that is, to

$$P(v^{n+1}) = P(v^n) - A(v^n) + b \quad . \tag{5.5}$$

The iteration will converge if v^n and J_P (the Jacobian of P), are near enough to v^{n+1} and J_A, respectively. This will usually be the case if both A and P are different discretizations of the same equation.

The approximate operation P can also be chosen to make Eq. (5.5) even easier to solve using an SOR, ADI, ILU or multigrid approximation.[16] When this is done, the residuals need to be computed with the high-order formula only in the last few iterations when the iteration is almost converged. The cost of the high-order approximations in the residual calculations are often small and more than justified in light of the resulting increase in accuracy.

The defect correction iteration can often be speeded up by using an acceleration technique such as a Chebyshev or conjugate gradient method.

VI. SUMMARY

We have used a modular approach to design a subroutine package calculating numerical approximations to the spatial derivatives of a function defined only at a discrete set of points. The routines are flexible, easy to use, and compatible to further expansions of the package. We hope that the software development and maintenance time of future PDE and interpolation codes using the package will be substantially reduced.

We are extending the package to include some pseudo-spectral methods and some better interpolation methods on two- and three-argument grids. We encourage

others to develop compatible subroutines that could be added to DERMOD. We will gratefully consider including into DERMOD any code sent to us that has been programmed using standard FORTRAN and the same supporting routines as the current package. For further information please contact J. M. Hyman.

ACKNOWLEDGMENTS

We thank Blair Swartz and John Van Rosendale for computing the lumped finite element method formulas on the two-argument grids and pointing out that the second derivative approximations are pointwise inconsistent. This work was supported by the US Department of Energy under contract W-7405-ENG-36.

REFERENCES

1. "Numerical Interpolation, Differentiation, and Integration," Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, M. Abramowitz and I. A. Stegun, Eds. June 1965, No. 55, pp. 877-899.
2. W. G. Bickley, "Numerical Differentiation Formulae for Numerical Differentiation," The Mathematical Gazette, 25, 18-27 (1941).
3. W. G. Bickley, "Finite Difference Formulae for the Square Lattice," Q. J. Mech. Appl. Math., 16, No. 1, 35-42 (1948).
4. G. Birkhoff and S. Gulati, "Optimal Few-Point Discretizations of Linear Source Problems," SIAM J. Numer. Anal., 11, No. 4 (September 1974).
5. Tony F. C. Chan, "Comparison of Numerical Methods for Initial Value Problems," Computer Science Department, Stanford University report STAN-CS-78- 672 (November 1978).
6. L. Collatz, The Numerical Treatment of Differential Equations, 3rd ed. (Springer-Verlag, Berlin, 1960), Appendix.
7. H. B. Keller and V. Pereyra, "Symbolic Generation of Finite Difference Formulas," Mathematics of Computation, 32, No. 144, 955-971 (October 1978).
8. R. E. Lynch and J. R. Rice, "A High-Order Difference Method for Differential Equations," Mathematics of Computation, 34, No. 150, 333-372 (April 1980).
9. J. M. Hyman and T. Manteuffel, "Local Polynomial Smoothing Methods for Noisy Data," Los Alamos National Laboratory informal report LA-UR-82-660 (1982).
10. S. T. Zalesak, "Very High Order and Pseudospectral Flux-Corrected Transport (FCT) Algorithms for Conservation Laws," Proceedings of the Fourth IMACS International Symposium on Computer Methods for Partial Differential Equations, Lehigh University, Bethelehm, PA, June 30-July 2, 1981.
11. J. E. Osborn, "The Numerical Solution of Differential Equations with Rough Coefficients," in Advances in Computer Methods for PDEs-IV, R. Vichnevetsky and R. S. Stapleman, Eds., (IMACS, New Brunswick, NJ, 1981), pp. 9-13.
12. Gerald N. Minerbo, "Another 9-Point Scheme (for Solving the Diffusion Equation on a Lagrangian Hydrodynamic Mesh)," Lecture notes, Los Alamos National Laboratory, 1979.
13. D. Cunsolo and P. Orlandi, "Accuracy in Non-Orthogonal Grid Reference Systems," Numerical Methods in Laminar and Turbulent Flow, (John Wiley & Sons, New York/Toronto, 1978), pp. 899-912.

14. J. F. Thompson and C. W. Mastin, "Grid Generation Using Differential Systems Techniques," Numerical Grid Generation Techniques, Hampton, Virginia, 1980), NASA CP. 2160, pp. 37-72, Institute for Computer Applications in Science and Engineering.
15. J. M. Hyman, "The Method of Lines Solution of Partial Differential Equations," Courant Institute of Mathematical Sciences report COO-3077-139 (1976).
16. J. M. Hyman, "Numerical Methods for Nonlinear Differential Equations," Los Alamos National Laboratory report LA-8927-MS (1981).

AD P000987

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

SOLUTION OF VISCOUS INTERNAL FLOWS ON CURVILINEAR GRIDS GENERATED BY THE SCHWARZ-CHRISTOFFEL TRANSFORMATION

O. L. ANDERSON[+], R. T. DAVIS[++], G. B. HANKINS[+], AND D. E. EDWARDS[+]
[+]United Technologies Research Center, East Hartford, Connecticut
[++]Department of Aerospace Engineering and Applied Mechanics
University of Cincinnati, Cincinnati, Ohio

SUMMARY

A method is presented for combining an accurate orthogonal curvilinear coordinate generation procedure with a fast, accurate, and stable forward marching viscous flow solution technique to solve for the flow field in arbitrary axisymmetric ducts. In this method, the coordinates are generated from the plane potential flow streamlines and potential lines using the Schwarz-Christoffel transformation with a composite finite difference formula which is valid everywhere and which treats the poles exactly by analytic integration. Since the coordinate streamlines approximate the actual streamlines, the equations of motion for viscous compressible flow can be parabolized so as to solve for both the boundary layers and core flow in a single streamwise pass. The versatility of the method is demonstrated by two examples of viscous compressible swirling flow through complex radial gas turbine passages.

INTRODUCTION

Accurate solution of high Reynolds number, viscous, compressible, swirling flows in complex turbomachinery ducts is a continuing concern in fluid mechanics. Two major areas of development are required for the solution of these problems. The first is the development of an efficient and accurate method to generate a coordinate system which should facilitate the formulation and numerical solution of the viscous flow analysis by aligning the coordinates along and normal to the predominant flow direction. The second is the development of a fast and accurate viscous flow solver which makes use of the properties of a properly constructed curvilinear coordinate system to simplify the problem and reduce computing time.

Many authors have addressed the grid generation problem by analytical or

The right to reproduce the copyrighted chapter in whole or in part and distribute copies thereof without charge to employees of United Technologies Corporation (UTC), its divisions, subsidiaries and related entities. The above right may be exercised by the author(s) or delegated to any library or archive within UTC.

numerical means. Sells[1], Jameson[2], and Garabedian and Korn[3] have adapted the conformal mapping procedure to calculate orthogonal curvilinear coordinate systems to solve the steady state transonic flow over an airfoil. Moretti[4] and Ives[5] have adapted conformal mapping methods to generate grids for a variety of other geometries. Anderson[6] has adapted the Schwarz-Christoffel transformation to calculate orthogonal curvilinear grids for a variety of simple internal flow passages and Davis[7] has adapted the same transformation to calculate a variety of external flow curvilinear grids. Recently, Sridhar and Davis[8] have extended the method of Davis to calculate grids for internal flow passages and Anderson, et al.[9] have combined into a single code the Davis method[7] for calculating coordinates and the Anderson method for solving turbulent compressible viscous flows in small radial gas turbine ducts. The present paper is an abbreviated version of the NASA contractor report prepared by Anderson, et al.[9].

The Schwarz-Christoffel transformation transforms the interior of a polygon to the upper half plane which in turn can be transformed to a straight channel. Each corner of the polygon is a pole (singularity) and the transformation is not analytic at that point. The method of Anderson[6] resolved the singularity problem by integrating along a boundary which was just inside the mapping domain. Hence, errors are incurred in mapping the boundary. In addition, the Anderson solution was such that it could not treat complex duct passages which turn 90 degrees or more because of multivalues in the wall coordinates. The Davis method[7], however, uses a composite finite-difference formula which is valid everywhere and which treats the poles exactly by analytic integration. Therefore, the Davis method can integrate along the walls and reduce the errors associated with the Anderson method. Sridhar and Davis[8] have also shown that the new Davis method yields second order accurate coordinates and metric coefficients. In addition, the Davis method has the flexibility to construct orthogonal curvilinear grids for complex internal flow passages which turn up to 180 degrees.

The overall objective of the present paper is to describe the Davis method for the calculation of orthogonal curvilinear grids suitable for internal duct flows and in particular focus on highly converging ducts which turn up to 180 degrees. The viscous flow analysis developed by Anderson[6] is then applied to

this orthogonal curvilinear grid to solve for the turbulent swirling compressible flow through complex small radial gas turbine passages.

ANALYSIS

Curvilinear Coordinate Analysis

The analysis on which the viscous solution technique is based requires that the coordinate system be orthogonal and that it be a first approximation to the viscous flow through the duct since the flow curvature is assumed to be the same as that of the streamwise coordinate lines[6]. A two dimensional orthogonal coordinate system can always be constructed from a potential flow solution by setting the normal coordinate equal to the stream function and the streamwise coordinate equal to the velocity potential. For plane flow, conformal mapping techniques are ideal because it allows solution of the inverse problem by direct means. That is (x(s,n), y(s,n)) rather than (s(x,y), n(x,y)) can be calculated directly where (x,y) is the Cartesian system and (s,n) is the curvilinear system and where s is the velocity potential and n the stream function. For many axisymmetric ducts, this plane flow solution serves as a sufficiently good approximation to the flow curvature of axisymmetric flow. However, for certain cases where this approximation is insufficient, a technique has been developed by Anderson and Edwards[10] to obtain axisymmetric streamline curvatures for use with the coordinate system derived from plane potential flow. Thus, coordinate grids based on conformal mapping have a wide range of applicability to the solution of viscous flow problems.

The mapping of an arbitrary duct in the (z) plane to a straight channel in the (t) plane has a special significance for the formulation of the viscous flow equations. Thus, if t is the complex potential, a Cartesian mesh in the (t) plane maps into an orthogonal curvilinear mesh in the (z) plane in which the coordinates are the potential lines and streamlines for the inviscid incompressible flow through the particular duct being analyzed. The coordinate streamlines thus approximate the viscous flow streamlines at high Reynolds numbers. This property of the coordinate grid is used to simplify the equations of motion.

A two-step transformation, shown in Fig. 1, is adopted. This first step is the Schwarz-Christoffel transformation from the duct (z) plane to the upper

half (ζ) plane which is given by

$$\frac{dz}{d\zeta} = M \prod_{i=1}^{N} (\zeta - b_i)^{-\alpha_i/\pi} \tag{1}$$

This mapping has a constant M which determines the rotation of the duct relative to the real axis. The corner angles are denoted by α_i and are known. The pole locations b_i in the ζ plane, however, are not known.

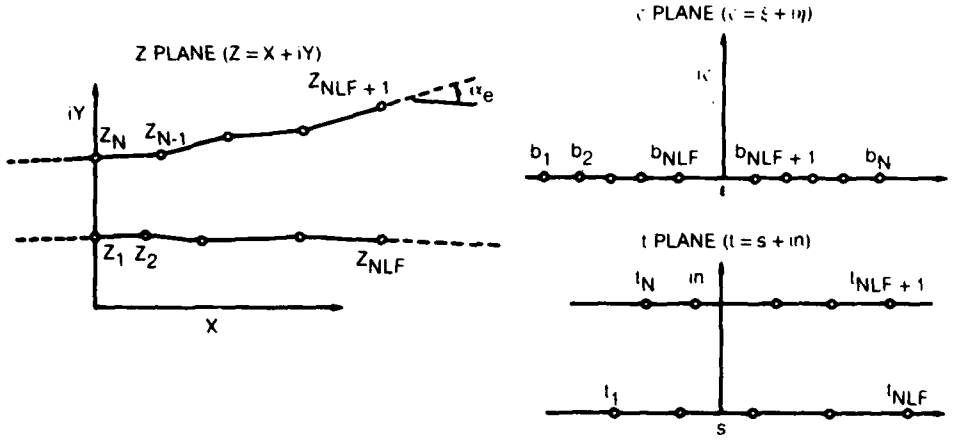


Fig. 1. Mapping of duct to straight channel.

The second step of the transformation is from the upper half (ζ) plane to a straight channel in the t plane is given by

$$t = -\frac{1}{\pi} \ln \zeta + i \tag{2}$$

If t is the complex potential, then

$$t = s + in \tag{3}$$

where s is the velocity potential and n is the stream function, then construction of a Cartesian mesh in the t plane represents a conformal mesh in the z plane composed of the stream function and velocity potential for the plane

potential flow through the duct. The complex conjugate of the potential flow velocity is

$$u - iv = \frac{dt}{dz} \tag{4}$$

Hence, the magnitude of the potential flow velocity is

$$v = \left| \frac{dt}{dz} \right| \tag{5}$$

which is the inverse of the metric coefficient.

Equation (1) can be reduced to a form involving only poles and angles on the duct. This form is given by

$$\frac{dz}{d\zeta} = \frac{M}{\zeta} \zeta^{-\alpha_e/\pi} \prod_{i=1}^{N} (\zeta - b_i)^{-\alpha_i/\pi} \tag{6}$$

The transformation given by Eq. (6) is singular at each pole b_i. Davis[7] has developed a composite finite difference formula by analytically integrating Eq. (6) in the neighborhood of the poles. This formula is given by

$$z_{K+1} = z_K + \frac{M}{\zeta_{K+1/2}} \zeta_{K+1/2}^{-\frac{\alpha_e}{\pi}} \prod_{i=1}^{N} \left\{ \frac{(\zeta_{K+1} - b_i)^{\frac{\alpha_i}{\pi}+1} - (\zeta_K - b_i)^{\frac{\alpha_i}{\pi}+1}}{(\zeta_{K+1} - \zeta_K)\left(\frac{\alpha_i}{\pi}+1\right)} \right\} (\zeta_{K+1} - \zeta_K) \tag{7}$$

From Eq. (2) we have

$$\zeta_{K+1} - \zeta_K = -\pi \zeta_{K+1/2} (t_{K+1} - t_K) \tag{8}$$

which may be combined with Eq. (7) to provide a direct integration to the t plane. These equations, as demonstrated by Sridhar and Davis[8], are second order accurate and contain no singularities. Therefore, the integration may be done along the walls which contain the poles. Equations (7) and (8) may be used to integrate along either streamlines or potentail lines. Thus we have

$$dt = ds + i\,dn \tag{9}$$

By setting dn = 0 the integration is along streamlines and by setting ds = 0 the integration is along potential lines.

The asymptotic solution for far upstream in the straight inlet channel is obtained $\zeta \to \infty$. In this limit, Eq. (1) reduces to

$$\frac{dz}{d\zeta} = \frac{M}{\zeta} \tag{10}$$

Integrating Eq. (10) and substituting Eq. (2), we have

$$z = \pi M(i - t) + z_o \tag{11}$$

Subtracting the lower wall from the upper wall results in

$$z_U - z_L = -\pi M_i \tag{12}$$

Then using the duct height H and angle of rotation θ, as shown on Fig. 2, we may solve for the constant M which results in

$$M = -\frac{H}{\pi} e^{i\theta} \tag{13}$$

Thus, M scales the height and rotation of the duct.

An examination of Eqs. (6) and (2), together with Fig. 3, shows that the α_i's and the line segments $|z_{ci+1} - z_{ci}|$ are known along the walls but the location of the poles b_i or t_i are not known. One constant can be fixed arbitrarily so that we take (z_1, b_1, t_1) as known. For the moment, let us assume (z_N, b_N, t_N)are known. Then new guesses for the poles b_i's are given by comparing the lengths of line segments

$$t_i^{\nu+1} = t_{i-1}^{\nu+1} + \frac{|z_{ci} - z_{ci-1}|}{|z_i^{\nu} - z_{i-1}^{\nu}|}\,(t_i^{\nu} - t_{i-1}^{\nu}) \tag{14}$$

and from Eq. (2)

$$b_i^{\nu+1} = \exp\left[\pi(i - t_i^{\nu+1})\right] \tag{15}$$

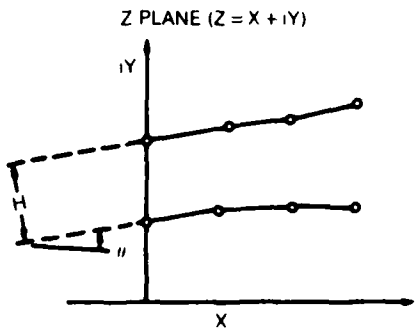


Fig. 2. Asymptotic Solution

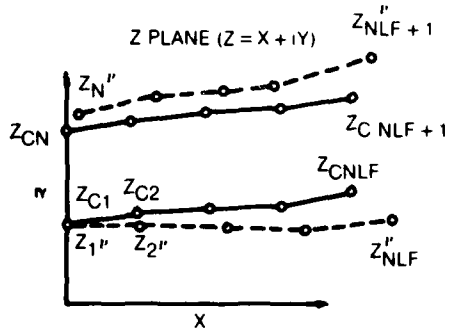


Fig. 3. Integration Update

Absolute and uniform convergence is established when all points satisfy the condition

$$\left| z_{ci} - z_i^{\nu} \right| < \epsilon \tag{16}$$

The iteration formula given by Eq. (14) is valid for all points except $t_N^{\nu+1}$. This point is determined using the asymptotic solution in the following manner. Let us define upstream points t_1' and t_N', shown in Fig. 4, by the following relations

$$\left.\begin{aligned} t_1' &= t_1 - \sigma\left|t_1\right| \\ t_N' &= t_1' + i \end{aligned}\right\} \tag{17}$$

where σ is a parameter chosen to move t_1' sufficiently far upstream to approximate the limiting asymptotic solution as $t \to -\infty$. Referring to Fig. 4, the point z_N^{ν} is determined with known t_i^{ν}'s by integrating along the path (z_{c1} to A to z_N^{ν}). Then the point z_1' is determined by integrating along the path (z_1 to z_1'). The point z_N' is determined using the asymptotic solution, Eq. (12). Hence,

$$z_N' - z_1' = -\pi M i \tag{18}$$

Then the point $t_N^{\nu+1}$ is given by

$$t_N^{\nu+1} = t_N' + \frac{\left|z_{CN} - z_N'\right|}{\left|z_N^{\nu} - z_N'\right|}\left(t_N^{\nu} - t_N'\right) \tag{19}$$

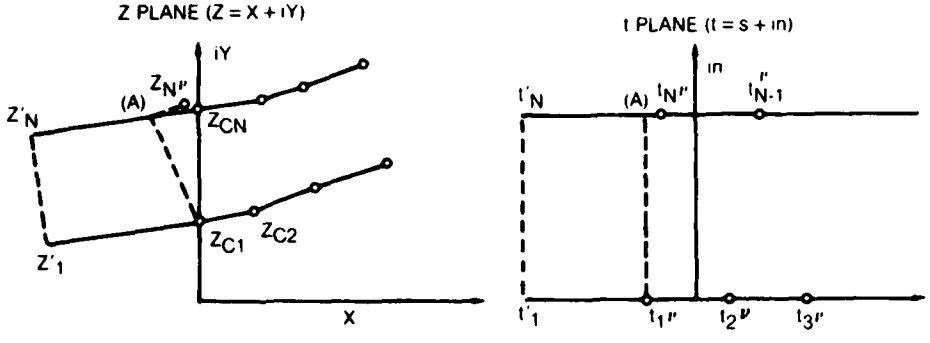


Fig. 4. Update for Corner Point

Viscous Flow Solution

The formulation of the viscous flow equations was presented by Anderson[6]. In this formulation a parabolic system of equations is derived from the Navier Stokes equations by assuming that the velocity component normal to the streamwise coordinate (inviscid streamline) is small compared to the streamwise component of velocity. In addition only the viscous stress component normal to the wall is retained. This method has been shown by Anderson[6] to be numerically stable and capable of simultaneously resolving the inviscid core flow and the boundary layer flow at the walls. It has also been shown by Barber, et al.[11] to yield the same results as viscous-inviscid interaction theory in which a boundary-layer solution was iterated with an inviscid core solution; but in contrast to interaction theory, which can only treat thin boundary layers, this method can treat problems with thin boundary layers up through fully developed channel flow. The method has been extended to treat swirling flows produced by inlet guide vanes

(see Barber, et al.[11]). In this manner, the special properties of the coordinate system are used to simplify the equations of motion so that the flow field can be calculated in a single streamwise pass rather than in multiple passes such as used in a viscous-inviscid interaction approach.

The viscous flow equations, given below, are written in an orthogonal streamline coordinate system where n is the normal coordinate (potential flow stream function) and s is the streamwise coordinate (potential flow velocity potential). The metric scale coefficient is the same in both the n and s directions and is equal to (1/V) where V is the magnitude of the potential flow velocity.

$$\frac{\partial \psi}{\partial n} = \frac{r \rho U_s}{V} \tag{20}$$

$$\frac{\partial \psi}{\partial s} = -\frac{r \rho U_n}{V} \tag{21}$$

$$\frac{V}{r}\frac{\partial \psi}{\partial n}\frac{\partial U_s}{\partial s} - \frac{V}{r}\frac{\partial \psi}{\partial s}\frac{\partial U_s}{\partial n} - \frac{\rho U_\phi^2}{r}\frac{\partial r}{\partial s} + \frac{\partial P}{\partial s} = \frac{V}{r}\frac{\partial}{\partial n}\left(\frac{r\tau_{ns}}{V}\right) - \left(\frac{\tau_{ns}}{V}\right)\frac{\partial V}{\partial n} \tag{22}$$

$$\frac{V}{r}\frac{\partial \psi}{\partial n}\frac{\partial U_\phi}{\partial s} - \frac{V}{r}\frac{\partial \psi}{\partial s}\frac{\partial U_\phi}{\partial n} + \rho\frac{U_s U_\phi}{r}\frac{\partial r}{\partial s} = \frac{V}{r}\frac{\partial}{\partial n}\left(\frac{r\tau_{n\phi}}{V}\right) + \frac{\tau_{n\phi}}{r}\frac{\partial r}{\partial n} \tag{23}$$

$$\rho U_s^2 \frac{\partial V}{\partial n} - V\rho\frac{U_\phi^2}{r}\frac{\partial r}{\partial n} + V\frac{\partial P}{\partial n} = 0 \tag{24}$$

$$T\frac{V}{r}\frac{\partial \psi}{\partial n}\frac{\partial I}{\partial s} - T\frac{V}{r}\frac{\partial \psi}{\partial s}\frac{\partial I}{\partial n} = V\frac{\partial}{\partial n}\left(\frac{rq_n}{V}\right) + \frac{\tau_{ns}^2 + \tau_{n\phi}^2}{\mu_E} \tag{25}$$

$$\tau_{ns} = \mu_E \frac{\partial}{\partial n}(VU_s) \tag{26}$$

$$\tau_{n\phi} = \mu_E \frac{\partial}{\partial n}\left(\frac{U_\phi}{r}\right) \tag{27}$$

$$q_n = C_p \frac{\mu_E}{Pr} V\left(\frac{\partial T}{\partial n}\right) \tag{28}$$

$$P = \rho RT \tag{29}$$

$$I - I_o = C_p \ln(T/T_o) - R \ln(P/P_o) \tag{30}$$

Equations (20) through (30), excluding Eq. (22), form a set of eight first order partial differential equations and two algebraic equations which may be used to solve for ten unknowns. The boundary conditions for this problem are given by

$$\begin{aligned} U_s(0,s) &= 0 \qquad & q_n(0,s) &= 0 \\ U_\phi(0,s) &= 0 \qquad & \psi(0,s) &= 0 \end{aligned} \tag{31}$$

for the inner wall and

$$\begin{aligned} U_s(1,s) &= 0 \qquad & q_n(1,s) &= 0 \\ U_\phi(1,s) &= 0 \qquad & \psi(1,s) &= \psi(1) \end{aligned} \tag{32}$$

for the outer wall.

In previous work, two turbulence models have been incorporated into the viscous flow analysis to provide closure of the problem. The first is an algebraic two layer eddy viscosity model[6] with recent corrections for streamline curvature and swirl developed by Anderson and Edwards[10]. The second is a two equation (k,ε) model developed by Chen[12] with corrections for streamline curvature and swirl developed by Launder, et al.[13]. The implementation of these turbulence models is presented by Anderson and Edwards[10]. The examples given in this paper were calculated using the algebraic turbulence model.

With the relationship between turbulent viscosity and the mean flow specified, Eqs. (20) through (30) can be solved by a forward marching numerical integration scheme. Equations (20) through (30) are first linearized by expanding all dependent variables in a Taylor series expansion in the marching direction (s), and terms of $0(\Delta s^2)$ are dropped. Finite-difference equations

are then obtained using the two point centered difference scheme of Keller[14,15]. The resulting matrix equations are (10 x 10) block tridiagonal and are solved by block factorization using the method of Varah[16]. The numerical solution is second order accurate in the n direction, first order in the s direction, linearly stable, and has no branching solutions[6]. The Δs step size is limited not by linear stability conditions but by the required accuracy in the Taylor series expansion in s.

RESULTS AND DISCUSSION

Comparison of Coordinate Calculations

A comparison of the coordinate grid generation analyses of Anderson and Davis was made by choosing a simple engine exhaust nozzle and calculating the coordinates with both methods. A geometric mesh consisting of 50 equally spaced streamlines and 80 potential lines was calculated using each coordinate analysis. The Davis method obtained uniform and absolute convergence with a tolerance of 10^{-4} in 7 iterations and the Anderson method obtained convergence to 10^{-3} in 8 iterations. The computational (CPU) time on a UNIVAC 1100/81A system was 15 1/2 minutes for the Davis grid generator and 15 minutes for the Anderson grid generator.

The two grid generators produced essentially identical results for the wall boundaries as can be seen in Fig. 5 which is a comparison of the inner wall coordinates for the exhaust nozzle calculated by both methods. This was expected since the two methods are based on the Schwarz-Christoffel transformation. However, as can be observed from Fig. 6 which is a comparison of the metric coefficinets along the inner wall using both methods, the Davis method calculates a smoother distribution of metric coefficients. This improvement can be attributed to the fact that the Davis method treats the poles exactly whereas the Anderson method does not.

AGT101 Gas Turbine

The AGT101 gas turbine, shown in Fig. 7, is a small automotive gas turbine under development by the Department of Energy, NASA Lewis Research Center, and private industry. Although it contains certain design features which are specifically tailored to the automotive gas turbine application, it represents the complexity of flow situations found in a broad class of small gas turbine

engines which can be addressed by computational fluid mechanics. In particular, one notes that the flow is compressible, turbulent, swirling, and passes through flow passages which turn up to 180 degrees. This paper presents two examples taken from this engine to illustrate the calculation procedure.

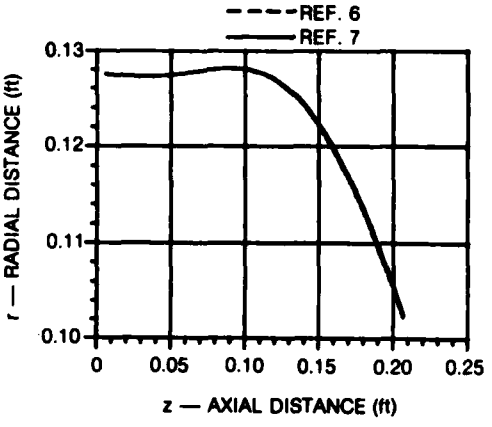


Figure 5. Comparison of Calculated Inner Wall Coordinates

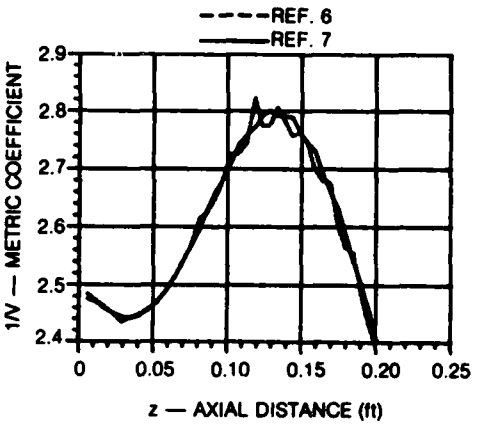


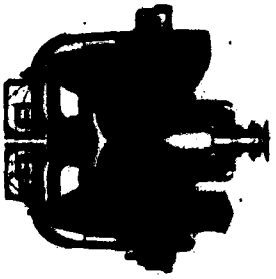
Figure 6. Comparison of the Calculated Inner Wall Metric Coefficients

AGT101 Turbine Inlet Duct

The viscous turbulent flow through the AGT101 turbine inlet duct, with struts, was calculated using inlet flow conditions supplied by NASA-Lewis Research Center[10]. The computational mesh and geometry used to represent the AGT101 turbine inlet duct in the analysis is shown in Fig. 8. This turbine inlet duct is a transition duct from the combustor exit plane to the turbine inlet plane and contains three struts arranged circumferentially around the duct. The plane view of these struts is shown in Fig. 8. The blunt stagnation point at the axis of symmetry is replaced by a faired streamline to bypass the need to solve stagnation point flow since the equations are singular at this point since the metric coefficient, 1/V, is infinite.

Initial flow conditions specified by NASA were uniform total temperature of 794.4 degrees Rankine and uniform total pressure of 3086.78 psf with a corrected weight flow of 0.3369 lbm/sec. These conditions are sufficient to set up all flow variables at the initial station which satisfy the global

continuity equation, normal momentum equation, and equation of state. The analysis was started downstream of the hub stagnation point to bypass the stagnation point solution. Turbulent boundary layers were assumed and a low Reynolds number algebraic turbulence model was used[10]. It was found that the Reynolds number per inch was so low that a turbulent boundary layer start using a momentum thickness estimated from a stagnation point solution was not possible. Since a laminar turbulent transition model is not currently available, the initial station was chosen further downstream and the momentum thickness increased. At this initial station, the Reynolds number based on momentum thickness was 400.



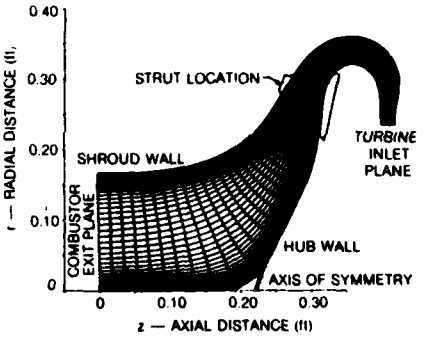


Figure 8. Computational Mesh for AGT101 Turbine Inlet Duct

A geometric mesh was calculated using the Davis method consisting of 50 equally-spaced streamlines and 100 potential lines. Uniform and absolute convergence of the conformal mapping solution was obtained to a tolerance of 1.5×10^{-4} in 13 iterations. The computational CPU time on a UNIVAC 1100/81A operating system was 29.5 minutes. The computational mesh shown on Fig. 8, consisting of 99 unevenly spaced streamlines and 100 potential lines, was obtained by linear interpolation from the 50 x 100 uniform mesh in order to provide adequate resolution of the flow in the wall boundary layers. In addition, the computational mesh was distorted near the wall using the Roberts

transformation[17] to provide grid resolution of the boundary layer.

The results of the flow analysis of the AGT101 turbine inlet duct are shown on Figs. 9 and 10. These figures show a comparison of the calculated static pressure from both the viscous and inviscid solutions along the hub and shroud walls with experimental data. The solid line on Figs. 9 and 10 is the solution for the viscous flow and the dashed line is the solution for the inviscid flow. From Figs. 9 and 10, it is observed that the results agree quite well with the experimental data. The close agreement between the results of the viscous solution and the inviscid solution indicate that in this case the effect of blockage due to the boundary layer is very slight except near the maximum duct height. The viscous solution did not predict separation in the AGT101 turbine inlet duct for the specified flow conditions. The computational CPU time was 11.6 minutes for the complete flow calculations. Of this time, 1.2 minutes was required for the inviscid solution[10] on a 99 x 100 mesh and 10.4 minutes was required for the viscous solution on a 254 x 100 mesh.

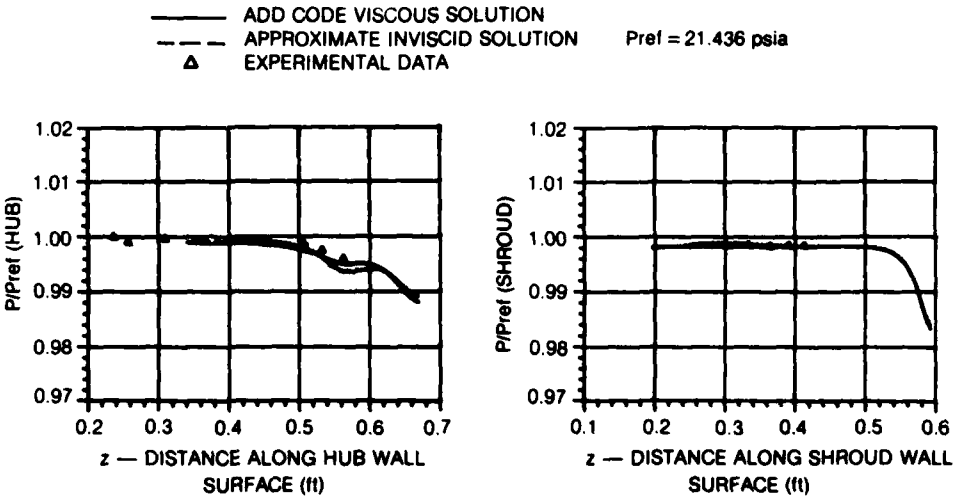


Figure 9. Hub Wall Static Pressure Distribution for AGT101 Turbine Inlet Duct

Figure 10. Shroud Wall Static Pressure Distribution for AGT101 Turbine Inlet Duct

AGT101 Turbine Exhaust Diffuser

The performance of the AGT101 turbine exhaust diffuser (see Fig. 11) was

measured on a test stand in which the turbine exhaust was simulated using inlet guide vanes. These inlet guide vanes (IGV) were a set of 16 blades with a 27 deg circular arc camber which were used to impart swirl to the flow. The projection of these inlet guide vanes onto the (r,z) plane is shown on Fig. 11. Axial flow enters the inlet guide vanes and leaves with a swirl angle of approximately 27 deg. This swirling flow enters the diffuser at the diffuser inlet station and is turned radially outward to exhaust at the diffuser exit plane.

The computational mesh, shown on Fig. 11, consists of 100 streamlines and 100 streamwise stations where the streamlines are concentrated near each wall to provide grid resolution of the boundary layer calculation. This computational mesh was interpolated from a 50 x 100 uniform mesh. Computational time on the Univac 1100/81A computer was approximately 15 min to obtain a convergence level of 10^{-4}.

Inlet conditions provided by NASA consisted of uniform total pressure and temperature at standard atmospheric conditions. A corrected weight flow (1.47 lb/sec) was provided to set the inlet Mach number. However, this weight flow established a Mach number at the inlet guide vane exit plane which was not consistent with the measured wall static to total pressure ratio. A guess for the actual weight flow (1.74 lb/sec) was made in an attempt to establish the correct initial conditions.

An overall view of the solution for the flow through the exhaust diffuser is shown on Fig. 12 for the streamwise velocity distribution across the duct at successive streamwise stations. The boundary layer growth on the hub and shroud walls is vividly illustrated. On the hub wall the boundary layer grows slowly as the flow is decelerated by the initial portion of the turn. Then the boundary layer thickness decreases as it recovers from the turn. Finally, the boundary layer grows slowly as it is decelerated in the radial diffuser. On the shroud wall, the boundary layer is initially accelerated as the flow enters the turn. Then the boundary layer grows rapidly as the flow recovers from the turn and continues to decelerate in the radial diffuser. At the exit, the shroud boundary layer is nearly separated and occupies almost one half the exit flow.

A comparison of the calculated wall pressure distribution with the measured pressured distribution on both the hub and shroud walls is shown on Fig. 13.

The agreement with experimental data is quite good considering the complexity of the flow field and the approximations necessary for estimating weight flow and inlet guide vane performance characteristics.

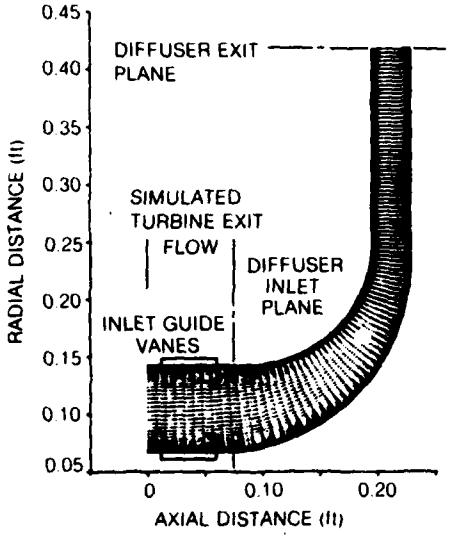


Fig. 11. AGT101 Exhaust Diffuser

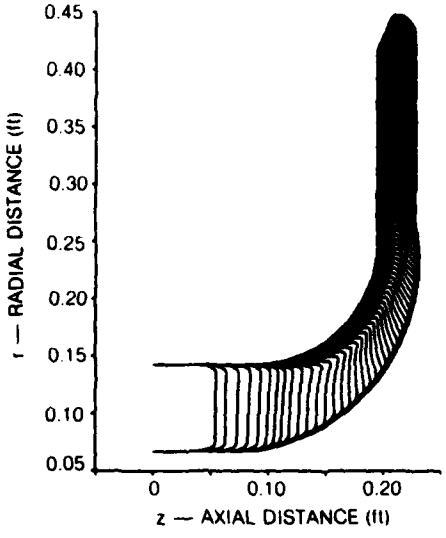
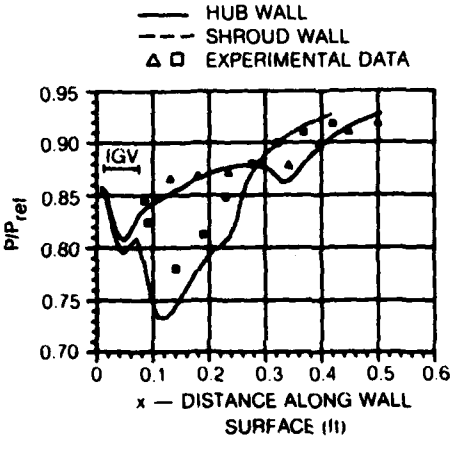


Figure 12. Streamwise Velocity Distributions in AGT101 Exhaust Diffuser

Figure 13. Calculated and Measured Pressure Distribution in AGT101 Exhaust Diffuser



CONCLUDING REMARKS

A method is presented for combining an accurate orthogonal curvilinear coordinate generation procedure with a forward marching viscous flow analysis. In this method properties of the coordinate system are used to formulate the equations of motion and properties of the viscous flow field solution and are used to select the distribution of the computational grid. The method has been successfully applied to calculate the viscous swirling compressible flow through complex radial gas turbine passages which turn up to 180 degrees and the calculated results compare favorably with the limited experimental data which is available.

ACKNOWLEDGEMENT

This work was supported by the Department of Energy under an interagency agreement with NASA-Lewis Research Center under NASA Contract DEN3-235. The technical monitor was K. J. McLallin.

LIST OF SYMBOLS

b_i	Poles	t	Complex potential (s + in)
i	$\sqrt{-1}$	T	Temperature
I	Entropy	U_s, U_n, U_ϕ	Velocity components
M	Complex constant	V	Metric coefficient (1/V)
n	Normal coordinate	Z	Duct plane (x + iy)
P	Pressure	α_i	Wall angle
Pr_T	Prandtl number	μ_T	Turbulent viscosity
q_n	Heat flux	ρ	Density
r	Radius	τ_{ns}, τ_{nt}	Stress components
S	Streamwise coordinate	ψ	Stream function

REFERENCES

1. Sells, C. (1968) Plane Subcritical Flow Past a Lifting Airfoil, Proc. Roy. Soc. (London), Vol. 308A.
2. Jameson, A. (1971) Transonic Flow Calculations for Airfoils and Bodies of Revolution, Grumman Report 390-71-1.
3. Garabedian, P. and Korn, D. (1971) Analysis of Transonic Airfoils, Comm. Pure & Applied Math., Vol. 24.
4. Moretti, G. (1976) Conformal Mappings for Computations of Steady, Three Dimensional Supersonic Flows, Num/Lab. Comp. Methods in Fluid Mechanics, 13 ASME.
5. Ives, D. C. (1975) A Modern Look at Conformal Mappings Including Doubly-Connected Regions, AIAA Paper 75-842.
6. Anderson, O. L. (1980) Calculation of Internal Viscous Flows in Axisymmetric Ducts at Moderate to High Reynolds Numbers, Computers and Fluids, Vol. 8, pp. 391-411.
7. Davis, R. T. (1979) Numerical Methods for Coordinate Generation Based on Schwarz-Christoffel Transformation, AIAA Paper 79-1463, 4th Computational Fluid Dynamics Conference.
8. Sridhar, K. P. and Davis, R. T. (1981) A Schwarz-Christoffel Method for Generating Internal Flow Grids, ASME 1981 Winter Annual Meeting, Symposium on Computers in Flow Predictions and Fluid Dynamics Experiments.
9. Anderson, O. L., Hankins, G. B., and Edwards, D. E. (1982) Extension to an Analysis of Turbulent Swirling Compressible Flow for Application to Axisymmetric Small Gas Turbine Ducts, NASA CR-165597, UTRC 915395-12.
10. Anderson, O. L. and Edwards, D. E. (1981) Extensions to an Analysis of Turbulent Swirling Compressible Flow in Axisymmetric Ducts, UTRC Report R81-914720-18.
11. Barber, T. J., Raghuraman, P., and Anderson, O. (1979) Evaluation of an Analysis for Axisymmetric Internal Flows in Turbomachinery Ducts, ASME Winter Annual Meeting, Flow in Primary Non-Rotating Passages in Turbomachinery.
12. Chen, K-Y. (1980) Predictions of Channel and Boundary Layer Flows with a Low-Reynolds Number Two-Equation Model of Turbulence. AIAA 18th Aerospace Sciences Meeting. AIAA-80-0134.
13. Launder, B. E., Pridden, C. H., and Sharma, B. I. (1977) The Calculation of Turbulent Boundary Layers on Spinning and Curved Surfaces. Trans. ASME J. of Fluids Eng., p. 231.
14. Keller, H. B. A New Difference Scheme for Parab_lic Problems. Numerical Solution of Partial Differential Equations - II SYNSPADE 1970. Academic Press, New York.
15. Keller, H. B. (1968) Accurate Difference Methods for Linear Ordinary Differential Systems Subject to Linear Constraints. SIAM J., Numerical Analysis, Vol. 6, No. 1.
16. Varah, J. M. (1972) On the Solution of Block Tridiagonal Systems Arising from Certain Finite Difference Equations. Mathematics of Computation, Vol. 26, No. 120.

AD P000988

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

TEST PROBLEMS, COORDINATE TRANSFORMATIONS, AND TECHNIQUE FOR NONSTEADY COMPRESSIBLE FLOW ANALYSIS

JON J. YAGLA
Naval Surface Weapons Center, Dahlgren, Virginia 22448

INTRODUCTION

This paper describes a finite difference technique for solving problems in nonsteady, two-dimensional, inviscid flow of an ideal gas. The technique solves the equations of gas dynamics in transformed coordinates obtained by conformal mapping of the physical domain of the problem. Irregular physical domains with curved or piecewise-straight boundaries are transformed onto rectangles to facilitate the application of boundary conditions in the finite difference calculations. The differential equations of fluid flow in conservation law form are solved by means of either the two-step Lax-Wendroff or MacCormack's finite difference method. An expedient means of obtaining coordinate transformations by a numerical integration of a Schwarz-Christoffel type differential equation has been used for problems in transient external aerodynamics. An extensive series of test problems, consisting of one-dimensional traveling shock waves, two-dimensional steady Prandtl-Meyer expansions, and oblique shocks were solved so that the finite difference calculations could be compared with exact mathematical results. Nonsteady Mach reflections were computed and compared with approximate theory and experimental data to test the technique for problems that are both fully two-dimensional and nonsteady.

CONSERVATION LAW FORM

The differential equations of gas dynamics for nonsteady flow of an ideal gas can be written as

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) + \frac{\partial}{\partial y} G(U) + \frac{\partial}{\partial z} H(U) = 0 \; . \tag{1}$$

For two-dimensional flow one has $\partial/\partial z = 0$. Here, U, F, and G are column vectors defined as follows:

$$U = \begin{pmatrix} \rho^* \\ \rho^* \left(\frac{\partial \xi}{\partial x} u + \frac{\partial \xi}{\partial y} v \right) \\ \rho^* \left(\frac{\partial \eta}{\partial x} u + \frac{\partial \eta}{\partial y} v \right) \\ E^* \end{pmatrix} \tag{2}$$

$$F(U) = \begin{pmatrix} \rho^* u \\ \rho^* \left(\frac{\partial \xi}{\partial x} u^2 + \frac{\partial \xi}{\partial y} uv \right) + \frac{\partial x}{\partial \xi} P^* \\ \rho^* \left(\frac{\partial \eta}{\partial x} u^2 + \frac{\partial \eta}{\partial y} uv \right) + \frac{\partial x}{\partial \eta} P^* \\ u\,(E^* + P^*) \end{pmatrix} \tag{3}$$

$$G(U) = \begin{pmatrix} \rho^* v \\ \rho^* \left(\frac{\partial \xi}{\partial y} v^2 + \frac{\partial \xi}{\partial x} uv \right) + \frac{\partial y}{\partial \xi} P^* \\ \rho^* \left(\frac{\partial \eta}{\partial y} v^2 + \frac{\partial \eta}{\partial x} uv \right) + \frac{\partial y}{\partial \eta} P^* \\ v\,(E^* + P^*) \end{pmatrix} \tag{4}$$

$$P^* = (\gamma - 1)\,\rho^* \left[\frac{E^*}{\rho^*} - \frac{1}{2} \sqrt{g}\ (u^2 + v^2) \right] . \tag{5}$$

In the above equations u and v are contravariant velocity components in the (x,y) curvilinear coordinate system. The curvilinear coordinates are related to the (ξ,η) Cartesian coordinates through a conformal transformation. The starred quantities are the usual physical quantities multiplied by $\sqrt{g}$, where $\sqrt{g}$ is the Jacobian of the coordinate transformation. The physical quantities ρ, E, and P are the density, total energy per unit volume, and pressure, respectively. I have called these scalar physical quantities multiplied by $\sqrt{g}$ "tensor densities" of the corresponding physical quantities. Special properties of conformal mappings, e.g. the Cauchy-Riemann conditions and the harmonic property, have been used to simplify the above equations.

Figure 1 shows three representations of a curved duct in the Cartesian and curvilinear coordinate systems. Figure 2 shows the representation of a fluid velocity vector in terms of physical and contravariant components along the curvilinear coordinate lines. The independent variables in the technique are the curvilinear coordinates (x,y). The dependent variables of the technique are the contravariant velocities u and v, and the tensor densities of the fluid density, total energy per unit volume, and pressure. Finite difference equations for solving these equations are presented in the following section.

FINITE DIFFERENCE EQUATIONS

The first attempt at approximating the above differential equations with finite differences failed. The two-step Lax-Wendroff finite difference scheme[1] originally developed for Cartesian coordinates was used as a prototype method for the conservation law form equations in curvilinear coordinates. The approach worked under the trivial coordinate transformation, i.e. w=z,

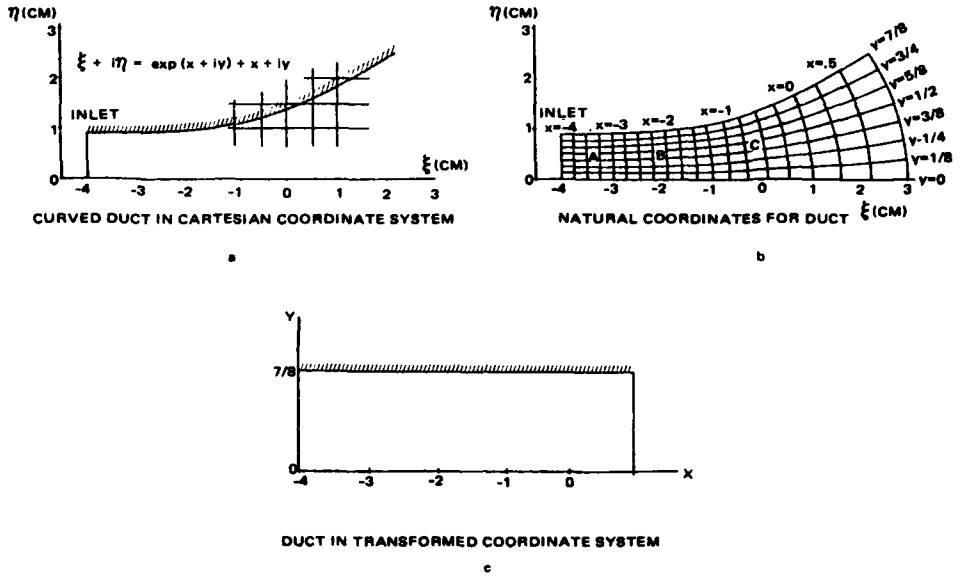


Fig. 1. Representation of duct in different coordinate systems

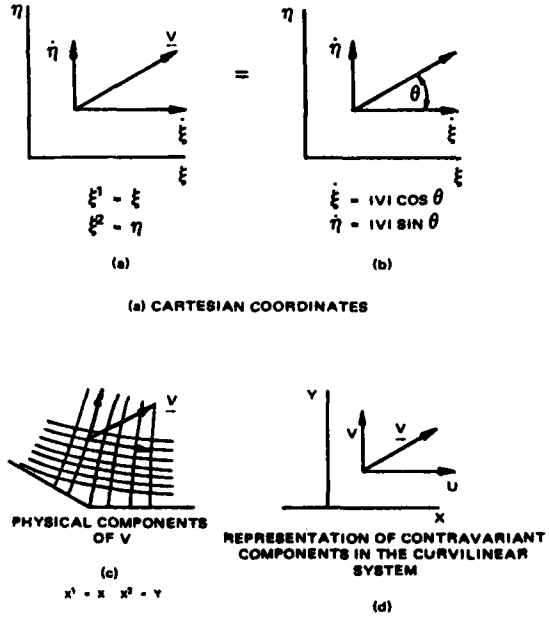


Fig. 2. Representations of the velocity vector

where $w = \xi + i\eta$ and $z = x + iy$. However, under the nontrivial coordinate transformation $w = e^z + z$, for a duct with a curved wall, Figure 1, the method failed. Flows spontaneously arose at interior points of regions of the flow field having an initially uniform quiescent state. The source of the error was found, and could be alleviated through a slight modification of the Cartesian form of the finite difference equations.[2] The modified two-step equations are

$$U^{n+1}_{j,k} = \frac{1}{4} J_{j,k} \left[\frac{U^n_{j+1,k}}{J_{j+1,k}} + \frac{U^n_{j-1,k}}{J_{j-1,k}} + \frac{U^n_{j,k+1}}{J_{j,k+1}} + \frac{U^n_{j,k-1}}{J_{j,k-1}} \right] - \frac{\Delta t}{2\Delta x} (F^n_{j+1,k} - F^n_{j-1,k}) - \frac{\Delta t}{2\Delta y} (G^n_{j,k+1} - G^n_{j,k-1}) , \quad (6)$$

and

$$U^{n+2}_{j,k} = U^n_{j,k} - \frac{\Delta t}{\Delta x} (F^{n+1}_{j+1,k} - F^{n+1}_{j-1,k}) - \frac{\Delta t}{\Delta y} (G^{n+1}_{j,k+1} - G^{n+1}_{j,k-1}) \quad (7)$$

where J is the Jacobian of the transformation. For the trivial coordinate transformation one has J=1, and the usual Cartesian two-step equations are recovered. Several test problems were solved with good results using this "generalized" two-step finite difference technique.

The MacCormack[3] difference equations for Cartesian coordinates are defined by:

$$\tilde{U}^{n+1}_{j,k} = U^n_{j,k} - \frac{\Delta t}{\Delta x} \left(F^n_{j+1,k} - F^n_{j,k} \right) - \frac{\Delta t}{\Delta y} \left(G^n_{j,k+1} - G^n_{j,k} \right), \quad (8)$$

$$U^{n+1}_{j,k} = \frac{1}{2} \left(U^n_{j,k} + \tilde{U}^{n+1}_{j,k} \right) - \frac{\Delta t}{\Delta x} \left(\tilde{F}^{n+1}_{j,k} - \tilde{F}^{n+1}_{j-1,k} \right) - \frac{\Delta t}{\Delta y} \left(\tilde{G}^{n+1}_{j,k} - \tilde{G}^{n+1}_{j,k-1} \right) . \quad (9)$$

These equations replace equations (6) and (7) of the Lax-Wendroff two-step formulation. The first step of the method constructs approximate values for $U^{n+1}_{j,k}$, denoted $\tilde{U}^{n+1}_{j,k}$, for each point using forward differences to approximate the spacial derivatives. The approximate solution is then used to calculate $\tilde{F}^{n+1}_{j,k}$ and $\tilde{G}^{n+1}_{j,k}$. The solution for $U^{n+1}_{j,k}$ is then computed with the second equation which uses backward differences obtained from the approximated functions $\tilde{F}$ and $\tilde{G}$. Since F(U) and G(U) are computed the same way using either Lax-Wendroff or MacCormack differencing, the computer program changes required to change technique are minimal.

COORDINATE TRANSFORMATION

General remarks

Conformal mapping was chosen as the coordinate transformation technique. At risk of repeating arguments of other proponents of the technique, the following list summarizes the basic strengths of conformal mapping method of coordinate generation:

(1) Conformal mapping is a "classical" technique that is highly developed for applications, and has the benefit of an extensively developed mathematical theory.

(2) There exists an ample body of literature with plenty of examples and even a "catalog."[4]

(3) Powerful numerical methods for constructing transformations are available when mapping by elementary functions cannot be accomplished.

(4) The transformed coordinates are orthogonal to the physical boundaries and to each other everywhere in the domain. This facilitates application of boundary conditions and minimizes computations elsewhere.

(5) The transformation has the simplest possible nontrivial structure. This minimizes errors in generating metric data and further errors entering the solutions from finite difference terms involving metric quantities.

(6) When used for two coordinates of a problem with three spacial dimensions, the result is a minimal amount of metric data that must be stored and used in mathematical operations. This may be crucial for solving three-dimensional problems with the presently available computers. For example, Moretti[5] has used rotated conformal mapping to compute the axisymmetric flow in nozzles and around gun barrels. Marconi and Salas[6] used conformal mappings of a succession of cross sectional planes normal to the axis of a fuselage and wing in a three-dimensional, supersonic, steady flow analysis of a fighter aircraft. Moretti[7] used conformal mappings in meridional planes to map an ablated three-dimensional reentry body.

(7) The coordinate lines are well aligned with the local velocity field because *the compressible flow problem is solved in a coordinate system that is the solution to an inviscid incompressible flow problem with the same physical boundaries.* This can only help to reduce numerical errors. (This can be carried even further. Kim, Thareja, and Lewis[8] solved the three-dimensional parabolized Navier-Stokes equations for a blunt cone at large angle of attack in supersonic flow using coordinates obtained from a method of characteristics solution to the corresponding inviscid supersonic flow problem.)

(8) Special properties of the transformation can be used to simplify the transformed equations of gas dynamics, thereby providing economy in the formulation of the finite difference equations and program execution.

A ninth and compelling advantage of conformal mapping for some problems will be presented below.

The drawbacks of using conformal mapping for coordinate transformation are mainly lack of coordinate controls. One cannot specify arbitrary coordinate distribution functions in order to concentrate coordinate lines in regions where flow properties are rapidly changing. A second limitation is that a considerable study of the theory of functions of a complex variable may be required for a user to develop sufficient skill to effectively use the technique.

As previously stated, the physical coordinate system of the gas dynamics problem is taken as the Cartesian (ξ,η) plane. The gas dynamics problem is solved mathematically in the transformed plane (x,y). Points in the physical (ξ,η) plane correspond to points in the (x,y) plane according to the complex valued function $w = f(z)$, where $w = \xi + i\eta$, $z = x + iy$, and $i = \sqrt{-1}$. The function

f(z) must meet the requirement that the complex derivative f'(z) exists at every point of the domain of f(z); then f(z) is said to be analytic. For such functions one has immediately available the Cauchy-Riemann conditions:

$$\partial\xi/\partial x = \partial\eta/\partial y \text{ and } \partial\xi/\partial y = -\partial\eta/\partial x \ ,$$

the implicit function theorem which yields

$$\partial x/\partial\xi = (\partial\xi/\partial x)/J, \quad \partial x/\partial\eta = (\partial\eta/\partial x)/J,$$

$$\partial y/\partial\xi = (\partial\xi/\partial y)/J, \quad \partial y/\partial\eta = (\partial\eta/\partial y)/J,$$

and four Laplace equations $\nabla^2\xi = 0$, $\nabla^2\eta = 0$, $\nabla^2 x = 0$, and $\nabla^2 y = 0$. Here J is the Jacobian determinant $\partial(\xi,\eta)/\partial(x,y)$. Along with the Laplace equations comes almost two hundred years worth of research on potential theory. One also has available powerful numerical techniques that were developed for grid generation by means of Poisson equations, as the Laplace equation is a special case.

The metric tensor for a general two-dimensional transformation, whether or not it is a conformal mapping, is taken from the expression for arc length in the transformed coordinate system:

$$(ds)^2 = (d\xi)^2 + (d\eta)^2$$

$$= \left[\left(\frac{\partial\xi}{\partial x}\right)^2 + \left(\frac{\partial\eta}{\partial x}\right)^2\right](dx)^2 + \left[\left(\frac{\partial\xi}{\partial y}\right)^2 + \left(\frac{\partial\eta}{\partial y}\right)^2\right](dy)^2$$

$$+ \left(\frac{\partial\eta}{\partial x}\frac{\partial\eta}{\partial y} + \frac{\partial\xi}{\partial x}\frac{\partial\xi}{\partial y}\right) dxdy \ .$$

The metric tensor is defined in terms of the Einstein notation

$$(ds)^2 = g_{ij}\, dx^i dx^j$$

where $x^1 = x$, $x^2 = y$, and the off-diagonal components are chosen so that g_{ij} is symmetric. The special properties of the analytic function lead to the immediate result:

$$g_{ij} = \begin{pmatrix} J & 0 \\ 0 & J \end{pmatrix} = J\delta_{ij} \ ,$$

where δ_{ij} is Kronecker's delta. For axisymmetric coordinate systems obtained by rotating a coordinate system obtained by conformal mapping, one obtains

$$g_{ij} = \begin{pmatrix} J & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & r^2 \end{pmatrix} \ ,$$

where r is the radial coordinate. When one examines the tensor form of the conservation equations of gas dynamics as displayed in the "conservation law form," the tremendous potential for simplification of calculations in the transformed coordinate system is readily apparent. One of

the goals of the project was to exploit the potential simplifications of the equations of gas dynamics in tensor form to the fullest possible extent. To this end curvilinear densities, pressures, and energies (the "tensor densities") were defined which made it possible to even remove the explicit appearance of the Jacobian from the transformed conservation equations. This provided a yet simpler system of equations to solve and led to a simple procedure for loading initial and boundary data into the computer program.

The examples and basic test problems of the technique use the principle of Schwarz-Christoffel transformation to obtain transformations of piecewise straight boundaries onto a straight line. In this way piecewise straight physical boundaries become the lower coordinate line ($y = 0$) in the transformed plane. The Schwarz-Christoffel transformation is based on the fact that complex transformations of the form $w = z^n$, where n is a rational exponent, cause the physical plane image of the x-axis to turn through a finite angle at the origin. One can successively apply this principle to obtain any number of finite turns of an axis at discrete points by means of the complex valued differential equation:

$$dw/dz = (z - z_1)^{n_1} (z - z_2)^{n_2} \ldots . (z - z_m)^{n_m} \quad . \tag{10}$$

The function w is called a Schwarz-Christoffel transformation if it maps a closed polygon onto a straight line. One of the early results of the present research was recognizing that this differential equation could be used in *unsolved* form to solve complicated aerodynamics problems. A discussion of this differential equation and its use in unsolved form is presented in the following paragraphs. The transformation in unsolved form was used to establish the coordinates that were used for calculating the "shock on shock" problems that are described in a later section.

Attempting to obtain general solutions to the above differential equation for more than two factors and exponents other than 0, ±1, ±1/2, ±1/3, and ±1/4 is useless. Solutions in terms of elementary functions, if they exist, are unknown. However, Anderson[9] has devised a numerical procedure for solving the Schwarz-Christoffel differential equation for up to 200 turning points. His work is designed to provide coordinate systems for duct flows. Woods[10] has generalized the Schwarz-Christoffel transformation to handle curved as well as piecewise straight boundaries. Davis[11] has devised a means of numerically integrating Woods' transformation for almost arbitrary piecewise smooth boundaries, greatly extending the conformal mapping technique. These recent developments enhance the attractiveness of the conformal mapping approach to coordinate generation, and increase the power and utility of numerical methods that employ conformal mapping as a means of coordinate generation.

Special consideration for external aerodynamics and transient field calculations

The following paragraphs describe a slightly different technique that can be used for grid generation for solving problems in external aerodynamics, heat conduction, and other field problems. In external aerodynamics one is generally interested in computing the flow field over a body. It

is usually the case that one needs to determine the pressure and/or the velocity field along the surface of the body. In this way the lift and drag on the body are computed. In other engineering problems, for example, structural design or determining the response of a vehicle to a transient load, one is required to compute the force on the body. Again the pressure distribution on the surface is required. It is usually the case in external aerodynamics work that one is not interested in the details of the flow field at points that are away from the body. Similar remarks hold for some transient problems in conduction heat transfer. High speed transient heat loads, such as the flow of propellant in a gun barrel or impingement of rocket exhaust on a structure, cause very high, and often damaging temperatures near the heated surface. However, the heat diffuses rapidly, and the in-depth, late time, temperature profiles and back wall temperature increases and boundary conditions are of relatively little interest.

An inspection of the differential equations of fluid dynamics for the transformed coordinates, equations 1 through 5, shows that the coordinate transformation quantities ($\partial\xi/\partial x$, etc., also called metric data) are the only coordinate transformation quantities that appear in the transformed differential equations. The quantity $\sqrt{g}$, the Jacobian determinant of the transformation, which appears in the equation of state, is computed from these same coordinate derivatives. It is especially significant that the coordinate transformation itself does not appear in the differential equations to be solved, only derivatives of the coordinate transformation. This carries over into the finite difference equations, where again only coordinate derivatives are required. One thus realizes that once the initial data and boundary conditions have been specified, then the finite difference calculations of the flow field can be carried out to arbitrarily large times without ever making recourse to the coordinate transformation *per se*, only coordinate derivatives are used.

The only time one requires the coordinate transformation itself is when points of the computational space have to be associated with points of physical space. This is done in establishing the initial data, possibly in application of boundary conditions for special problems, and when transforming the solution back into physical space. For example, in calculating nonsteady external flow, the initial data are usually some initially uniform or quiescent stream. The boundary conditions for inviscid flow over any body are that the flow is parallel to the body, which is true in any coordinate system and can be formulated and implemented in the finite difference calculation independent of the details of the particular coordinate transformation. The no-slip condition for viscous flow is even easier.

SCHWARZ-CHRISTOFFEL TRANSFORMATION

The Schwarz-Christoffel differential equation maps straight lines onto piecewise straight boundaries, or *vice-versa*. The transformation is of the form $w = f(z)$ where w is the solution to the complex valued ordinary differential equation (10). The quantities $\partial\xi/\partial x$ and $\partial\eta/\partial x$ which appear in the transformed differential equations of gas dynamics are the real and imaginary parts of $f'(z)$ respectively equations (1) through (5). The remaining coordinate derivatives $\partial\xi/\partial y$ and $\partial\eta/\partial y$ are

just $-\partial\eta/\partial x$ and $\partial\xi/\partial x$ according to the Cauchy-Riemann conditions. The significance is that once $f'(z)$ is specified, one needs only to take the real and imaginary parts of $f'(z)$ to completely determine the differential and finite difference equations of motion. An explicit representation of $f(z)$ is not required. Numerical values for the coordinate derivatives are readily computed from the expression for $f'(z)$ by FORTRAN statements without ever having to obtain explicit formulae for $\xi(x,y)$ or $\eta(x,y)$. For external aerodynamics calculations one only needs a correspondence between physical boundary points and node points along the image of the boundary in the computational domain. An expeditious means for developing the required correspondence is described below.

A further important property of $f'(z)$ follows from interpreting $f'(z)$ as a magnification and rotation of an infinitesimal line element under the mapping, that is,

$$\frac{\Delta w}{\Delta z} \approx \frac{dw}{dz} = f'(z) = |f'(z)| \exp\left[i \arg f'(z)\right] .$$

The first factor on the right is the magnification, the second factor is the rotation. Taking further absolute values,

$$\mathrm{mod}\,(\Delta w) = |f'(z)|\,\mathrm{mod}\,(\Delta z) .$$

Because the transformation is analytic, the value of f' is independent of the direction of Δz. Thus every infinitesimal line element of length ϵ passing through the point (x,y) is strained by the same factor and rotated through the same angle in passing to the (ξ,η) plane. Also the mapping is conformal, i.e., the angle of intersection of infinitesimal line elements is preserved under the transformation.

By letting $\Delta z = \Delta x$, and then $i\Delta y$, and using the orthogonality property of the transformation, there results

$$\Delta A_w = |f'(z)|^2\,(\mathrm{mod}\,\Delta x)(\mathrm{mod}\,i\Delta y) = |f'(z)|^2\,\Delta A_z ,$$

where ΔA_w is the area in the w plane corresponding to the area element ΔA_z in the z plane. Use of the analyticity of the transformation, or equivalently the Cauchy-Riemann equations, yields

$$\Delta A_w = J\Delta A_z$$

where J denotes the Jacobian determinant $\partial(\xi,\eta)/\partial(x,y)$. Now the physical density of a continuum of unit thickness can be defined by

$$\rho = \lim_{\Delta A_w \to 0} \frac{\Delta M}{\Delta A_w} ,$$

where ΔM is the mass continuously distributed over ΔA_w in the (ξ,η) Cartesian system. Then

$$J\rho = \frac{\Delta M}{\Delta A_z}$$

is the mass per unit area when measured in units of the z (curvilinear) plane. When "tensor densities" are used as dependent variables, the explicit appearance of the Jacobian determinant of the transformation is removed from the flow equations so that it appears only in the equation of state.

NUMERICAL INTEGRATION OF SCHWARZ-CHRISTOFFEL DIFFERENTIAL EQUATION

Because of the difficulty in carrying out the integration of equation (10), a numerical technique is required to obtain the transformation. Equation (10) represents a line integral in the complex z plane. In principle, this integral could be evaluated along lines of constant x and constant y to construct the w plane images of the (x,y) coordinate lines. Alternatively, one can view equation (10) as a first order differential equation for f(z) and use an appropriate numerical procedure. The latter approach, which consists of a generalization of the Runge-Kutta method to the complex plane, is used here.

To construct the equations for transforming a piecewise straight boundary such as shown by Figure 3 onto a straight line, the numerical integration proceeds from a point on the x axis, such as point A on Figure 3, sufficiently far from the first turning point x_1. The Runge-Kutta scheme is then used to integrate up to x_1. As the singular point z_1 is approached, the integral remains finite. This can be seen from the form of equation (10). In a small region near the point z_1, the factors involving z_2, z_3, etc. in equation (10) are nearly constant, while the integrand of equation (10) changes drastically on account of z being near z_1. It is known that the integral of the first factor is well behaved for $k_1 \neq 1$, and is of the form:

$$\int (z - x_1)^{-k_1} \, dz = \frac{1}{1 - k_1} (z - x_1)^{1-k_1} \quad . \tag{11}$$

Therefore, although the differential equation (10) is singular at the points z_1, z_2, and z_3, the solution remains regular at the turning points. After the integration of equation (10) has been accomplished up to z_1, with perhaps equation (11) having been used near z_1, the integration then proceeds from z_1 to z_2 along the x axis in a similar fashion. An alternate procedure is to integrate along a line $z = x + i\epsilon$, where ϵ is a very small number. Then no singular points are encountered. This hueristic procedure is justified on the basis of the accuracy obtained in a variety of sample problems.

Determination of the curvilinear coordinates away from the x axis with the numerical scheme is straightforward, as no singular points are encountered. The construction of the transformation of coordinates for points not on the x axis is carried out as follows. For example, from point A on Figure 3, one integrates equation (10) numerically along a line of constant x ($dx = 0$). The value of f(A) serves as the constant of integration. The integration is carried out from A to B. The point B corresponds to the top of the computational mesh. Values of w along this line then serve as initial data (constants of integration) for computing the image lines of coordinates corresponding to lines of constant y. Starting from initial values taken along the line $\overline{f(A), f(B)}$, the

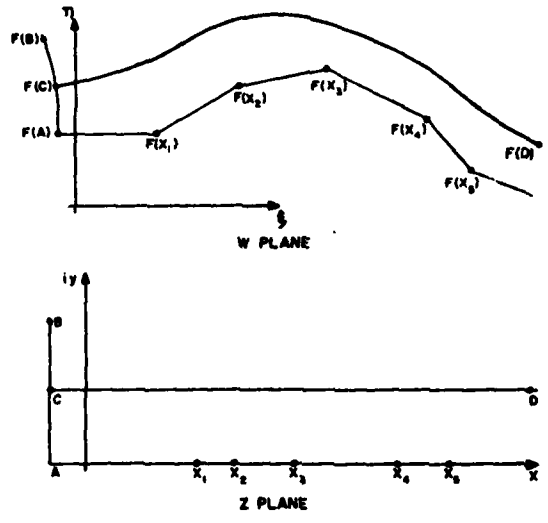


Fig. 3. Numerical integration of Schwarz-Christoffel differential equation

numerical Runge-Kutta scheme is then used to construct curvilinear lines such as $\overbrace{f(C), f(D)}$. For these lines the numerical integration scheme uses dz = dx, i.e., no imaginary component, to construct coordinate lines that are images of the lines of constant y. An analogous procedure is then used for constructing images of lines of constant x to complete the transformation.

The numerical method consists of a fourth order accurate Runge-Kutta scheme that has been generalized to the case of complex variables. For a real valued first order differential equation of the form y = f(x), the Runge-Kutta equations can be reduced to:

$$x = f(x)$$

$$x_1 = x_0 + \Delta x$$

$$y_1 = y_0 + \Delta x \{ f(x_0) + 4 f(x_0 + \Delta x/2) + f(x_1) \}/6 \quad .$$

Generalizing to line integrals in the complex plane, for the differential equation w = f(z),

$$x = f(z)$$

$$z_1 = z_0 + \Delta z$$

$$w_1 = w_0 + \Delta z \{ f(z_0) + 4 f(z_0 + \Delta z/2) + f(z_1) \}/6 \quad .$$

Solutions are obtained by successive applications of the above formula along lines in the complex plane. A simple computer program for the calculations is contained in Figure 4. Several problems have been worked out with the computer program to test accuracy and suitability of the technique.[12]

```
PROGRAM     COMPLEX TRACE                    OPT=0  01/31/74  11.00.29.
      PROGRAM COMPLEX (INPUT,OUTPUT)
C
      COMPLEX POWER,FP
      COMPLEX W,Z,DZ,WP
C
C**** THIS FUNCTION RAISES Z TO THE POWER EXPON
C
      POWER(Z,EXPON)=CABS(Z)**EXPON*CMPLX(COS(EXPON*ATAN2(
     1AIMAG(Z),REAL(Z))),SIN(EXPON*ATAN2(AIMAG(Z),REAL(Z))))
C
C**** THIS FUNCTION IS THE COMPLEX DIFFERENTIAL EQUATION OF THE TRANSFORMATION
C
      FP(Z)=
     1  POWER((Z+5.0,-4)*POWER(Z,2.*A)*POWER(Z-5.0,-A)
      PI=3.14159
      A=(ATAN(2.*PI/5))/3.141593
C
C***** READ INITIAL VALUES OF Z,W, AND INTEGRATION STEP
C
      READ Z,W
      READ Z,W
      READ Z,W
     2 FORMAT(4F10.4)
      WP=FP(Z)
      PRINT 4
     4 FORMAT(1X,1HZ,34X,1HW,28X,2HW'/1HW
     5 FORMAT(1X,2F10.5,4(5X,F10.5))
     6 PRINT5,Z,W,WP
C
C****ADVANCE SOLUTION ONE STEP
C
      DZ=FP(Z)
      W=W+DZ*(1.0+4.*FP(Z+DZ/2.)+FP(Z+DZ))/6.
      Z=Z+DZ
      IF(REAL(Z)-10.) 6,5,999
  999 CALL EXIT
      END
```

Fig. 4. Computer program for fourth order Runge-Kutta integration of Schwarz-Christoffel differential equation

The Runge-Kutta technique provides for fourth order accurate location of boundaries. Since the metric quantities were evaluated exactly from the real and imaginary parts of f'(z), we have arrived at a quasi-numerical method of coordinate generation for piecewise straight boundaries, which is the ninth compelling reason for using conformal mapping which was promised earlier:

(9) The technique provides *exact* metric data and *fourth order* accurate locations of the boundary and coordinate lines away from the boundary.

This is in contrast to most elliptic methods that locate the boundaries exactly but provide coordinate lines and metric data to second order accuracy. Since it is the metric data that appear in almost every term of the governing equations of fluid dynamics, and never the precise location of the boundary or any coordinate line, the present method is superior in this regard to other methods that employ only approximate metric data.

TEST PROBLEMS

The preceding sections presented the differential equations of two-dimensional compressible flow in conservation law form and two finite difference approximations of second order accuracy

to the differential equations. The differential and difference equations are valid for any coordinate system obtained by conformal mapping. A coordinate generation scheme based on the Schwarz-Christoffel differential equation that provides exact metric data and fourth order accurate locations of transformed boundaries has been described. This chapter presents a series of test problems and results for standard compressible flow phenomena. Additional results and a more comprehensive explanation of the test problems can be found in reference 13.

This chapter presents results from calculations of standard compressible flow phenomena, and compares the calculations with exact theory and experimental data. The first calculations are for constant speed shocks and the reflection of a constant speed shock from a solid boundary at normal incidence. The second set of calculations is for oblique shock reflections and Prandtl-Meyer expansions. The constant speed shock calculations provide an elementary test of the technique for computing nonlinear and transient phenomena in one space dimension. The oblique shocks and Prandtl-Meyer expansions provide elementary tests of the technique for computing steady flow in two space dimensions in curvilinear coordinates. Collectively, these problems are the most basic compressible flow phenomena, and therefore provide an essential test of any numerical technique for compressible flow. The first set of test problems demonstrates the ability of the theory to solve transient flow fields; the second set of calculations shows the ability of the technique to solve fully two-dimensional problems with continuous and discontinuous solutions. A method of characteristics for steady two-dimensional flow in coordinate systems generated by conformal mappings was developed as a further means of testing the finite difference technique. As a final test the results of transient Mach reflection calculations are presented to demonstrate the capability for solving more general problems of two-dimensional, nonsteady, compressible flow.

The constant speed shock provides an essential test for nonsteady compressible flow calculations and provides a very basic test for the computer program. For the trivial transformation $w = z$, the (x,y) curvilinear coordinate system degenerates into a Cartesian system identical to (ξ,η). Under this transformation the program for solving curvilinear problems can be used for solving two-dimensional problems in Cartesian coordinates. Calculations with this transformation completely exercise the computer program, although many of the variables and coefficients of the transformation quantities are zero or one. However, for a program to be successful under more complicated transformations, it is necessary that it provide correct results for the trivial transformation. The results of constant speed shock calculations are presented in Figures 5 and 6 and are compared with exact theory and calculations of the same problem using other techniques. A 4 by 100 grid was used. The exact values were obtained from the Rankine-Hugoniot equations.

The most basic two-dimensional, inviscid, steady, compressible flows are the oblique shock and the Prandtl-Meyer expansion. If the flow is turned through an angle such that the flow is compressed, as shown in Figure 7, then an oblique shock is formed. If the flow is turned through an angle such that the flow is expanded, then a fan of expansion characteristics can be used to describe the process, as shown in Figure 7. The flows downstream of the shock and expansion waves

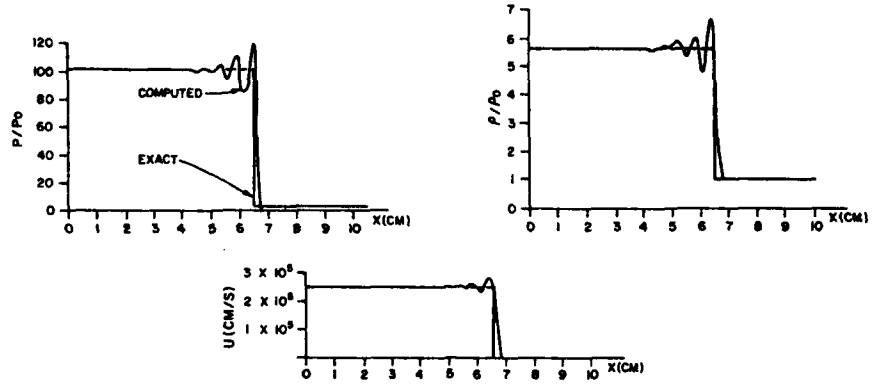


Fig. 5. Computed results for strong shock with Lax-Wendroff finite differences

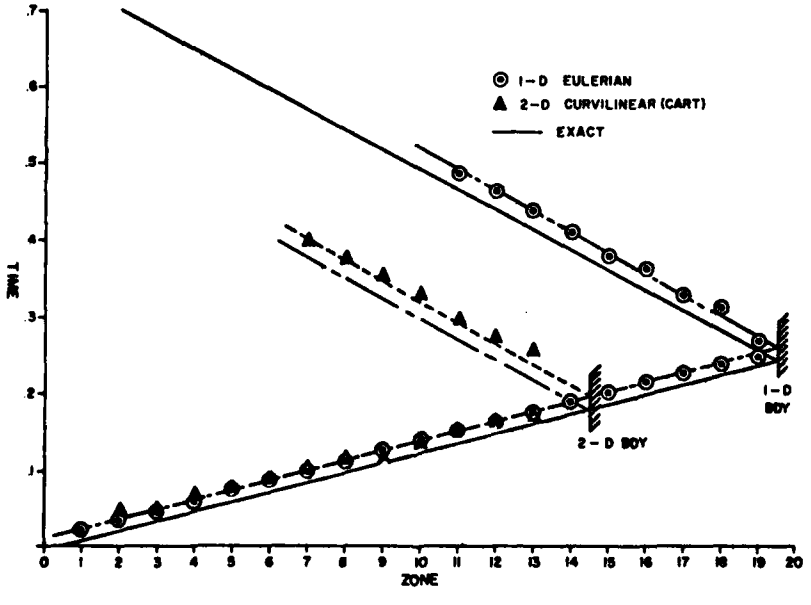


Fig. 6. Wave diagram for shock reflection at solid boundary

can be computed using exact theory from known upstream conditions and the angle of turning, provided that the flow is supersonic on both sides of the shock.

The analytic transformation $dw/dz = nz^{n-1}$, or $w = z^n$, where n is a rational exponent, was used to obtain a natural coordinate system for computing the flow over a wedge. For $n = 17/18$,

5 DEGREE COMPRESSION:

QUANTITY	THEORETICAL (EXACT)	COMPUTED (AVERAGE)	NUMBER OF POINTS	DIFFERENCE	STANDARD DEVIATION
PRESSURE	6.468×10^{6}	6.455×10^{6}	171	0.20%	15.70%
VELOCITY	5.113×10^{4}	5.137×10^{4}	171	0.47%	9.27%
DENSITY	4.580×10^{-3}	4.582×10^{-3}	171	0.04%	11.90%

5 DEGREE EXPANSION:

PRESSURE	3.916×10^{6}	3.850×10^{6}	101	1.80%	6.41%
VELOCITY	6.27×10^{4}	6.29×10^{4}	101	0.32%	1.61%
MACH. NO.	1.517	1.507	101	0.66%	2.35%

Fig. 7. Summary of results for steady supersonic compression and expansion

a ten degree wedge results, as shown in Figure 8. Under this transformation the line $y = 0$, the x axis, is transformed onto the lines

$$w = x^{17/18}, \quad x \geq 0 ,$$

$$w = |x|^{17/18} e^{i17\pi/18}, \quad x < 0 .$$

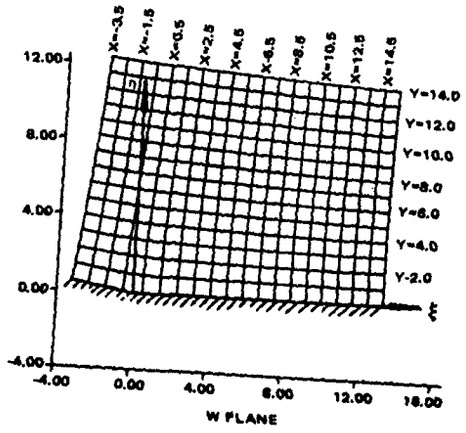


Fig. 8. Ten degree wedge obtained from the transformation $w = z^{17/18}$

The transformed x axis is shown by the heavy line in Figure 8. Other lines of constant x and y are also shown in Figure 8. The physical problem is in the w plane, the line $\eta = 0$, for $x \geq 0$, being the z plane image of the surface of the wedge. The incident flow is a parallel stream, parallel to and above the line $w = |x|^{17/18} \exp(i17\pi/18)$ on Figure 8. Transformations for other turning angles are readily computed. For example, the above formula with $n = 35/36$ produces a five degree compression, and $n = 37/36$ produces a five degree expansion. The solutions shown in Figure 7 were computed with a 20 by 20 grid. The pressure profile on an inclined surface using MacCormack differencing is shown by Figure 9.

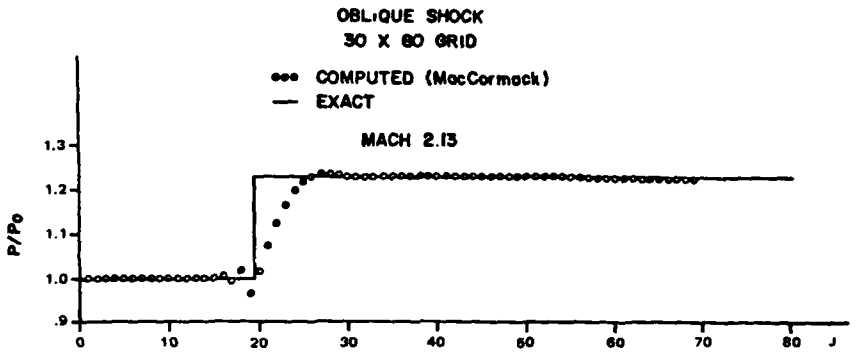


Fig. 9. Surface pressure on a 3.59 degree wedge

High quality experimental data have only recently become available for oblique shock reflections.[14] The experimental data show that real gas effects are of considerable importance to the structure of the shock reflections, particularly for strong shocks and large turning angles. One would not therefore expect to obtain particularly good agreement between the above experiments and calculations unless very general equations of state were employed. Finite difference calculations can, however, provide very good agreement with experimental[15] data, when real gas effects are taken into account.

There are two basic types of transient shock reflections, regular reflection and Mach reflections. Regular shock reflections are somewhat like acoustic or optical reflections, except the angle of incidence does not equal the angle of reflection and the reflected pressure does not vary linearly with the incident pressure. The Mach reflection patterns consist of three distinct shocks.

The following paragraphs describe computer calculations of Mach reflections in curvilinear coordinates. The technique used for the Mach reflections is essentially the same as used for the oblique shock steady flow calculations of the previous section; however, the program was not run until steady state conditions were attained. The Mach reflection calculations presented below were carried out for shocks of various strengths encountering a ten degree wedge. The coordinate system for the calculations was essentially the same as shown by Figure 8. The incident shock was

caused to travel normal to the line $w = x^{17/18} \exp(i17\pi/18)$ by continuously resetting the values along the entire line $x = -3.5$ to the desired upstream values obtained from the Rankine-Hugoniot equations for the desired shock strength.

A very simple approximate theory exists for computing the overpressure behind a traveling Mach stem. Simple reasoning leads one to the conclusion that if the Mach stem is to travel normal to the reflecting surface and still remain attached to the incident wave, then its speed must be approximately the speed of the incident wave divided by the cosine of the wedge angle. Then, if the slip stream attached to the triple point is assumed to travel parallel to the reflecting surface, and all shocks are assumed to be straight, a complete set of formulae can be worked out for calculating the geometry of the Mach stem and the flow field behind each shock of the system.[16] The data points enclosed by squares on Figure 10 were computed with this method. The upper portion of the curve on Figure 10 shows Bertrand's[17] experimental data points for strong shocks enclosed by triangles.

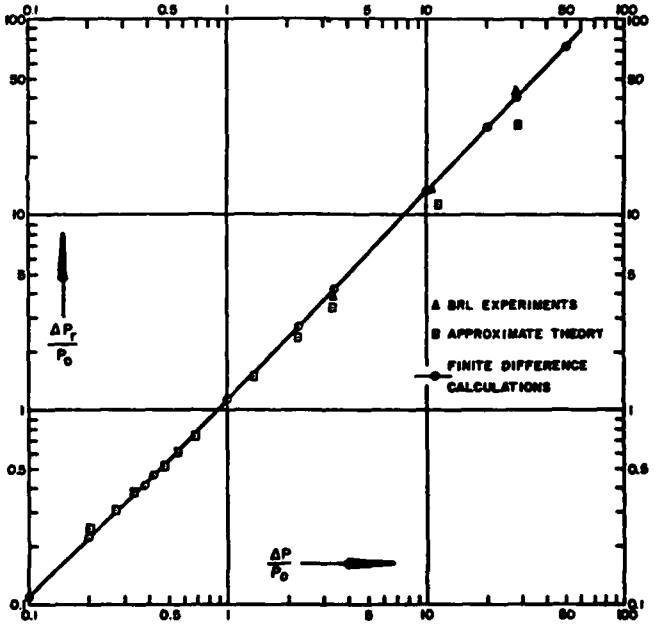


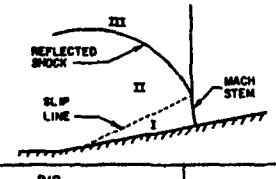
Fig. 10. Finite difference calculations compared with other sources of data

Ben-Dor and Glass[18] have provided density contours and other data for shock Mach numbers in the range of two to eight, and a series of wedge angles of two to sixty degrees. These data would logically form the basis for further comparisons between theory and experiment with real gas effects.

Computer calculations of a nonsteady Mach reflection on a ten-degree wedge were compared with the experimental data obtained by Dewey.[19] The experimental data were obtained from a study of the motion of a number of fluid elements, in the purest Lagrangian sense, using a unique smoke tracer particle technique in a shock tube. Further, the experimenter used the experimental density field behind the Mach reflection to compute the pressure field, and confirmed the pressure field with direct measuring electronic instruments. Although the comparison with Dewey's data should not be considered to be as basic as the comparisons with exact mathematical theory as described above, the experiments provide an interesting test case. The test problem fully exercises both the transient and two-dimensional aspects of the technique. The results are also interesting because the incident shock Mach number is 1.15 and the pressure ratio is 1.378. The shock is therefore neither weak nor strong, and provides a test case for an intermediate strength regime where other experimental data seems to be totally lacking.

A 20 by 20 computing grid was used for the calculations. The results of the computer calculations were output at t = 400 microseconds for comparison with Dewey's results. This time compares to Dewey's time of 240 microseconds. This is because Dewey measured time from the time of arrival of the shock at the wedge. The present calculations have the time equal to zero at the start of the calculations, the shock being upstream of the wedge. The computer calculations required $42{,}300_8$ words of memory and 28 seconds of execution time on the NSWC CDC 6700 computer.

The computed data were compared with the experimental data by calculating the averages of the data points in each of the three regions. The data fields consisted of 154 theoretical points. The averages over the individual regions and the difference between the theoretical and experimental values, expressed in percent, are shown in Figure 11. The pressures agreed to 0.7, 0.0, and 1.4 percent, and the densities to 0.8, 0.0, and 1.5 percent in regions I, II, and III, respectively. Also, the average of the percentage differences was found to be 0.73 percent.



REGION	P/P_0			ρ/ρ_0		
	COMPUTED	EXPERIMENT	PERCENT DIFFERENCE	COMPUTED	EXPERIMENT	PERCENT DIFFERENCE
I	1.42	1.41	.7	1.30	1.27	0.8
II	1.40	1.40	–	1.27	1.27	–
III	1.40	1.38	1.4	1.27	1.25	1.6

AVERAGE DIFFERENCE = 0.7%

Fig. 11. Summary of results of mach reflection calculations

Finally, a method of characteristics for two-dimensional, steady, supersonic flow was developed for the equations of gas dynamics in coordinate systems obtained by conformal mapping.[13] At the expense of slightly more complicated compatibility equations, one has a very simple boundary treatment that applies for all problems, rather than very specific, case dependent boundary treatments as in rectangular coordinates. This provides a natural and aesthetically pleasing test technique for finite difference calculations in coordinates obtained by conformal mapping, as the method of characteristics is known to be very accurate, and the same coordinates can be used for the test problem and baseline problem. The results of a curvilinear coordinates method of characteristics calculation for the flow field in an exponentially divergent duct of Figure 1 is shown in Figure 12.

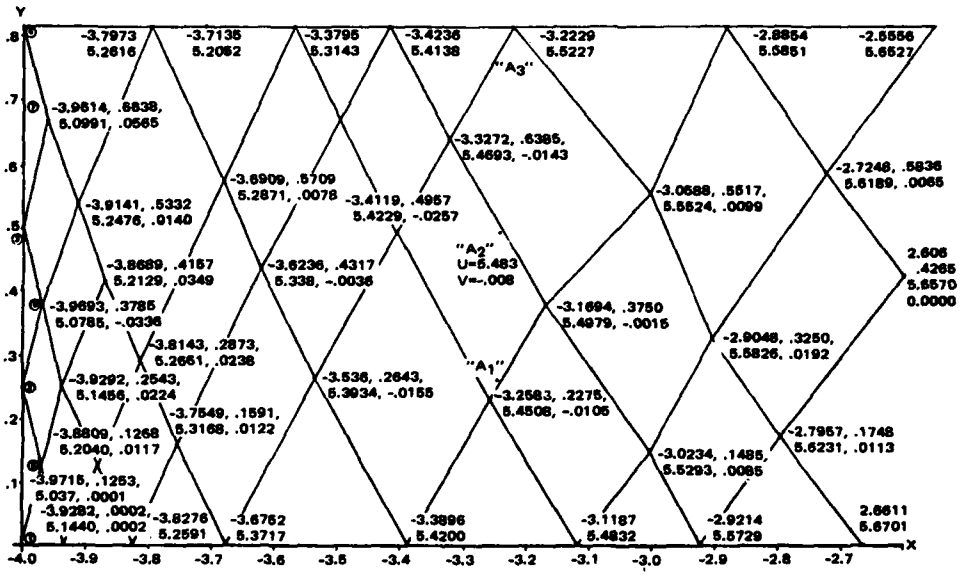
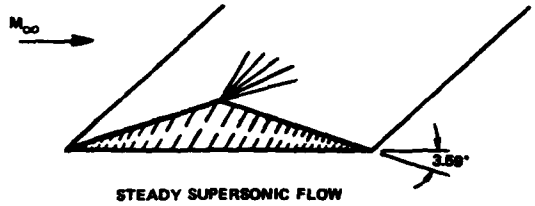


Fig. 12. Wave diagram for characteristics in transformed duct

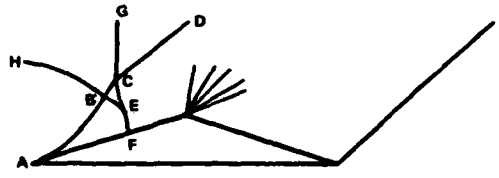
APPLICATION TO SHOCK-ON-SHOCK PROBLEM

A problem of a blast wave interacting with a wing in supersonic flight is sketched in Figure 13 for a symmetrical double wedge airfoil. The initial flow field consists of an oblique shock attached to the leading edge, a Prandtl-Meyer expansion centered at mid-chord, and an oblique shock recompression at the trailing edge as shown in Figure 13 (a). The wave system created when a blast wave intercepts the supersonic airfoil is sketched in Figure 13 (b). The shock ABCD is the oblique



STEADY SUPERSONIC FLOW

(a)



SHOCK INTERACTION PROCESS

(b)

Fig. 13. Steady supersonic flow and nonsteady shock interaction on a triangular airfoil

shock during interaction with the blast wave. The shock GCEF is the incident shock discontinuity, having GC as an undisturbed portion. The segment EF is the Mach stem, with the triple point E. The shock EBH is the reflected portion of the Mach reflection. The entire shock structure, with the exception of GCD, is composed of curved segments. The flow consists of three distinct temporal periods: a steady period prior to intercept; a nonsteady period while the blast wave diffracts over the wing section and the flow adjusts to the new free stream velocity, pressure, and density; and a final steady state that is attained asymptotically after the wing is well into the blast wave. Exact analysis is clearly impossible.

The Schwarz-Christoffel differential equation for the transformation for a particular wing was taken as

$$w' = \frac{z^{2a}}{(z+5)^a\,(z-5)^a} \quad , \tag{12}$$

where a = .039897. The computer program of Figure 4 can be used to generate the image of the wing under the coordinate transformation. The flow field solution only requires that equation (12) be loaded into the compressible flow program.

Figure 14 shows the results of the calculation for the flow over the wing. The coordinates (ξ,η) are Cartesian coordinates traveling at the speed of the wing. The body fixed curvilinear coordinates (x,y) are shown. The approximate space-time trajectory of the incident shock wave

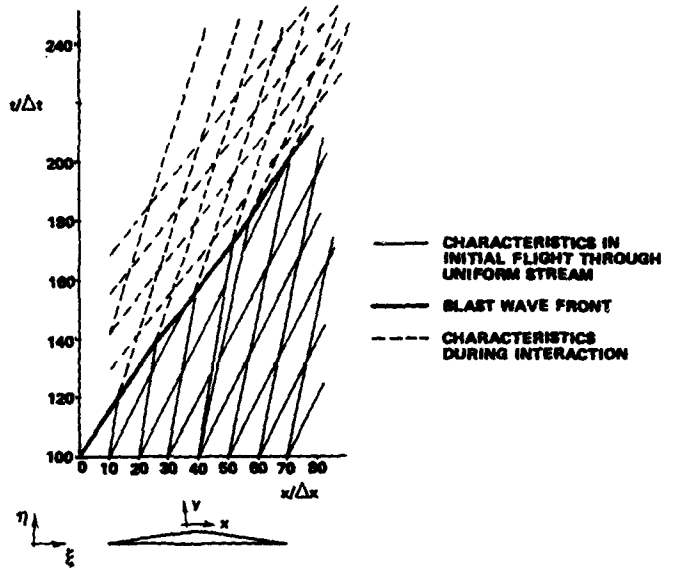


Fig. 14. Stream tube wave diagram for shock interaction on wing

along a small stream tube containing the x (y = 0) coordinate line is shown. The figure is actually a wave diagram in transformed coordinates for an infinitesimal stream tube which travels along the incident stream to the forward stagnation point, then along the wing. Initially, ahead of the wing the flow velocity in the stream tube is U = 8.02 X 10^4 cm/s and the pressure is one bar. The oblique shock compression attached to the leading edge causes the pressure to jump to 1.23 bar and the velocity to drop to 7.49 X 10^4 cm/s. The Prandtl-Meyer expansion causes the pressure to drop and the velocity to increase. The trajectory of the incident blast wave is shown as a heavy line beginning at t/Δt = 100 and x/Δx = 0. Initially the wave travels at speed 7.87 X 10^4 cm/s relative to the incident stream and 15.82 X 10^4 cm/s relative to the wing. Upon arrival at the wing, the blast wave speeds up due to the compression of fluid by the wing. The increase in speed is evident on the wave diagram.

ACKNOWLEDGMENTS

The research described above was performed as a project of the Naval Surface Weapons Center Independent Research Program with discretionary funds provided by the Naval Materiel Command. The work formed a portion of a doctoral research project in the Aerospace Engineering and Engineering Science Department of Arizona State University. Dr. D.L. Evans of the Mechanical and Energy Systems Engineering Department supervised the research.

REFERENCES

1. Richtmyer, R.D. and Morton, K.W. (1967) Difference Methods for Initial Value Problems, Second Edition. New York: Interscience Publishers.
2. Yagla, J.J. (1972) "Calculation of Blast Wave Interactions with Irregular Surfaces." 3. International Symposium on Military Applications of Blast Simulators. Organized by Ernst-Mach-Institut, Freiburg and Akademie für Wehrverwaltung und Wehrtechnik, Mannheim. 19-21 September. Presented as paper D-4 of the proceedings.
3. MacCormack, R.W. (1969) "The Effect of Viscosity in Hypervelocity Impact Cratering." AIAA Paper No. 69-354.
4. Kober, H. (1952) "Dictionary of Conformal Representations." New York: Dover Publications, Inc.
5. Moretti, G. (1981) "A Numerical Analysis of Muzzle Blast Precursor Flow." Polytechnic Institute of New York Aerodynamics Laboratories. Department of Mechanical and Aerospace Engineering. To be published.
6. Marconi, F. and Salas, M. (1973) "Computation of Three-Dimensional Flows about Aircraft Configurations." International Journal of Computers and Fluids, 11; 185-195.
7. Moretti, G. (1980) "Grid Generation using Classical Techniques." Presented as Paper No. 1 at NASA Numerical Grid Generation Techniques Workshop, NASA Conference Publication 2166.
8. Kim, M.D., Thareja, R.R. and Lewis, C.H. (1981) "Three-Dimensional Viscous Flowfield Computations in a Streamline Coordinate System." AIAA Paper No. 81-0401 presented at the AIAA 19th Aerospace Sciences Meeting, St. Louis, Missouri, January 12-15.
9. Anderson, O.L. (1981) "Calculation of Internal Viscous Flows in Axisymmetric Ducts at Mid to High Reynold's Numbers" to appear in International Journal of Computers and Fluids.
10. Woods, L.C. (1961) The Theory of Subsonic Plane Flow. Cambridge: The Cambridge University Press.
11. Davis, R.T. (1979) "Numerical Methods for Coordinate Generation Based on Schwarz-Christoffel Transformations." AIAA Paper 79-1465.
12. Yagla, J.J. (1974a) "A Technique for Constructing Analytic Transformations of Piecewise Straight Boundaries." Naval Weapons Laboratory Technical Note TN-T/1-74, Naval Weapons Laboratory, Dahlgren, Virginia, January.
13. Yagla, J.J. (1981) Appendix E of "A Finite Difference Method for Solving the Equations of Gas Dynamics in Curvilinear Coordinate Systems." Ph.D. Dissertation, Arizona State University. Available from University Microfilms.
14. Ben-Dor, G. and Glass, I.I. (1979) "Domain and Boundaries of Nonstationary Oblique Shockwave Reflections: 1. Diatomic Gas." Journal of Fluid Mechanics, 92; 459.
15. Book, D.L. (1980) "Two-Dimensional FCT Model of Low-Altitude Nuclear Effects." Naval Research Laboratory Memorandum Report 4362.
16. Moore, G.R. (1972) "Calculation of the Reflected Overpressure and Transient Loading on a Deck from Elevated Gun Fire." NWL Technical Report TR-2847, Naval Weapons Laboratory, Dahlgren, Virginia, November.
17. Bertrand, B.P. (1972) "Measurement of Pressure in Mach Reflection of Strong Shock Waves in a Shock Tube." Ballistic Research Laboratories Memorandum Report No. 2196.
18. Ben-Dor, G. and Glass, I.I. (1978) "Nonstationary Oblique Shockwave Reflections: Actual Isopyonics and Numerical Experiments." AIAA Journal 16, No. 11; 1146-1153.
19. Dewey, J.M. (1972) "The Use of Particle Trajectory Analysis for Studying the Blast Loading on Structures," Schwetzingen, Germany: Presented at the Third International Symposium on the Military Applications of Blast Simulators, Sponsored by the Ernst-Mach Institute, September. Published as Paper D-3 of the proceedings.

AD P000989

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

AN EXPERIENCE IN MESH GENERATION FOR THREE-DIMENSIONAL CALCULATION OF POTENTIAL FLOW AROUND A ROTATING PROPELLER

WEN-HUEI JOU†
†Flow Research Company, 21414 68th Avenue South, Kent, Washington 98031

INTRODUCTION

A new generation of propellers with eight to ten blades operating at transonic flight speeds is now under development.[1,2] These propeller blades are typically highly swept and twisted with small aspect ratios and supersonic tip speeds. A three-dimensional finite-volume computational code that accounts for cascade effects, hub-induced flow and nonlinear transonic effects is highly desirable. An effort to develop such a code has been undertaken by the present author; the results will be reported elsewhere.[3]

The present paper discusses the author's experience in mesh generation for this complex configuration. What is reported here is not new methodology but some practical considerations for a specific problem. However, the experience gained here is valuable to the developers of basic methodologies in generalizing their methods to accommodate these practical considerations, particularly for three-dimensional geometries.

In general, most of the three-dimensional mesh generation schemes are the outgrowths of two-dimensional schemes. Hence, with the exception of very few works,[4] one of the dimensions, e.g., the spanwise dimension for a wing, is less emphasized. One would divide the three-dimensional space into two-dimensional sheets on which more exotic two-dimensional body-fitted meshes are generated. The detail of the geometry in the third dimension, e.g., the wing tip, is sacrificed. This bias toward two-dimensionality is somewhat related to the usual requirement of mapping three-dimensional space continuously to a cubical computational domain (I, J, K), a requirement for the ease of formulating a discretized approximation of the governing partial differential equation. If a practical numerical method that does not require continuous mapping could be found, the generation of an unbiased three-dimensional mesh can be made easy. The present work is no exception to the common practice. However, even with this two-dimensionally biased approach, the experience in this work shows that further difficulties arise from assembling seemingly reasonable two-dimensional meshes to form a three-dimensional mesh system. Also, the interdependency between the mesh system, the numerical method and the physics of the flow under consideration is emphasized.

SLICING THREE-DIMENSIONAL SPACE

The flow field around a rotating propeller with an axisymmetric hub is periodic in the azimuthal direction. The period is determined by the number of blades. It is natural to divide the computational space into cylindrical-type surfaces. On these surfaces, two-dimensional meshes which map the domain to a rectangular computational space are generated. The nodal points with the same coordinates in these two-dimensional computational spaces are connected to give the three-dimensional network.

Let (x,r,θ) be the polar cylindrical coordinates with the axis along the axis of rotation of the propeller as has been done in the mesh generation for an airplane.[5] A small-radius semi-infinite cylinder is attached to the nose of the hub. The function of this cylinder is to avoid the mesh singularity that inevitably appears in a mesh system around a finite three-dimensional body. The cylinder is so small that it does not interfere with the flow around the propeller-hub combination in an inviscid flow calculation.

The resulting cylinder-hub combination is then sheared to a constant-radius cylinder by a simple shearing transformation in (x,r) space as shown in Figure 1.

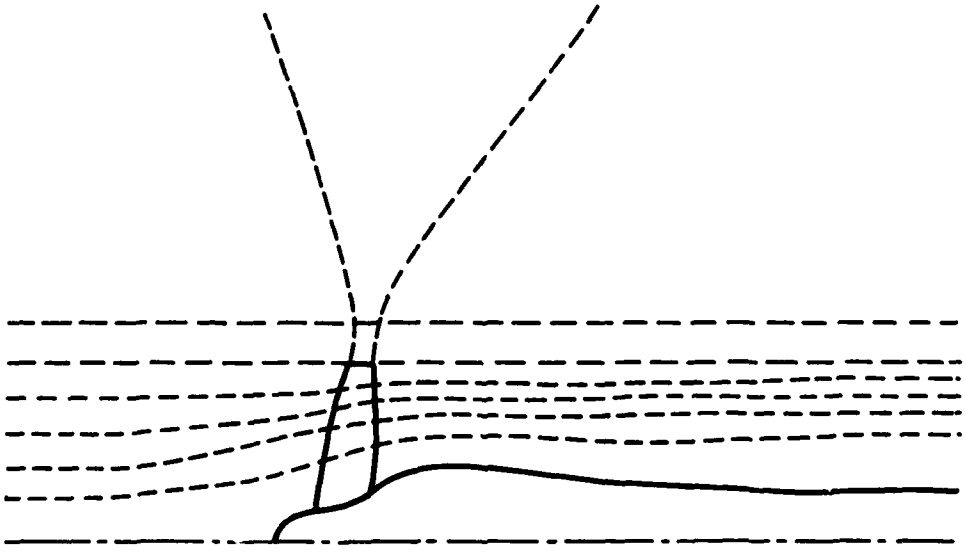


Fig. 1. Cylindrical-type surfaces and augmentation of blades.
SR-1 Propeller, 8 Blades

To ensure that a similar topological structure, i.e., a cascade configuration, can be given on each of the cylindrical surfaces, a fictitious "blade" extending to infinity is attached to the tip of each blade. Two considerations are important in this augmentation of the blade. If one continues the rate of twist of the blade beyond the tip so that the chord of the "blade" is approximately aligned with the undisturbed flow, a cascade of plates facing the flight direction will be formed a large distance from the propeller. It is impossible to generate a cascade mesh compatible to that of the inboard cylinder. Hence, the rate of twist beyond the tip is reduced so that at the outer boundary, the twist angle reaches a reasonable limiting value.

The second consideration is the chord length distribution of the attached fictitious blade. As the radial distance r increases, the distance between the blades increases linearly. Given the same number of mesh lines between blades, the mesh spacing in the blade-to-blade direction increases accordingly.

If the same chord length at the tip is maintained for the fictitious part beyond the tip, the mesh aspect ratio becomes exceedingly large toward the outer boundary. This property of the mesh may cause some numerical difficulty. To avoid this problem, the chord of the "blade" is flared out, approaching linear growth as it approaches the outer cylindrical boundary, as shown in Figure 1.

Having accomplished the augmentation of the blade-hub combination, the exterior of the sheared constant-radius hub is divided into a series of cylinders. The intersection of these cylinders with the sheared blades is obtained through interpolation. A two-dimensional mesh will be generated on these cylinders and sheared back to the physical space.

TWO-DIMENSIONAL CASCADE MESHES

Two-dimensional meshes which map the physical space continuously to the computational space are generally classified into three main categories: O-type, C-type and H-type meshes. The so-called O-type mesh maps the exterior of a body to a ring-shaped computational domain on which a branch cut is defined. For a C-type mesh, a branch cut is defined in the physical space, generally along the wake trajectory. The resulting singly connected physical domain can be mapped to a rectangular strip in the computational domain. These two types of meshes have one family of mesh lines wrapping around the leading edge of the airfoil. No singularity of transformation is presented at the leading edge. The singularity of the transformation at the trailing edge is ignored, since a special treatment of the flow there is often required to satisfy the Kutta condition.

The H-type mesh maps the exterior of an object to an infinite space with the image of the object as an internal branch cut. At the leading edge, a smooth contour is mapped to two sides of the branch cut, giving a square-root singularity of the transformation there. The H-type mesh is therefore considered a grid system not suitable for potential flow calculations. However, recent work has shown that the singularity resulting from the mesh transformation can be removed in the numerical computations if the correct singular behavior can be embedded in the basis function for solution projections.[6]

In terms of computational efficiency, the O-type mesh is rated most efficient, and the C-type and H-type meshes are considered less efficient in that order. This rating is based on the efficiency for a two-dimensional object. Another consideration in selecting one of these meshes is their ability to be assembled to describe a fairly complex geometry in three dimensions. While the efficiency of the meshes can be easily defined as the ratio of the number of points on the solid surface to that of the total number, the latter is hard to quantify. The choice is probably related to various aspect ratios of the dimensions of the object.

In the following, we describe our experience with these types of meshes. This experience may shed some light on the quantification of the flexibility of different types of meshes.

General requirements for cascade meshes

There are a number of requirements for cascade meshes, some of them essential to the cascade configuration and some of them essential for forming a three-dimensional mesh. These requirements are listed below:

(1) For all spanwise configurations, the blade surface must be on mesh lines with the same configuration in the computational space, i.e., one type of mesh must be used consistently at all spanwise stations.

(2) The mesh index at the trailing edge must be the same for all meshes at different spanwise stations, so that when they are connected, a clean trailing edge in the third dimension can be defined.

(3) Because of the periodicity, only flow around one blade can be computed. The periodic boundary condition can be easily applied numerically, if the mesh is periodic.

(4) The finite-volume numerical method for computing potential flow requires that the assumed vortex wake be placed on a mesh surface. For a propeller, the vortex wake is assumed to leave the trailing edge at its bisection angle and to follow asymptotically a helical trajectory along the direction of the local undisturbed flow.

C-type and O-type meshes

The O-type or C-type meshes for a cascade configuration can be generated by various methods, such as solving an elliptic equation[7] or using electrostatic analogy.[8,9] The method of electrostatic analogy for generating a C-type mesh has been experimented with in the present work. These meshes have been used successfully in two-dimensional computations. For three-dimensional calculations, they present some difficulties unforeseeable from the two-dimensional test calculations.

First of all, the upstream distance covered by an orthogonal or near-orthogonal C-type or O-type mesh is of the order of the blade-to-blade distance. For O-type meshes, this is also true for the downstream direction. For an advanced turboprop with eight to ten blades, the blade-to-blade distance near the hub is less than a chord length. The meshes will not be able to cover the entire hub geometry when they are assembled for the three-dimensional configuration. The hub-induced flow cannot be accurately computed and the upstream far field condition is applied too close to the blade. Some stretching of the mesh upstream can be performed, but excessive stretching can result in an extremely skewed mesh that may cause difficulties in numerical solution.

Furthermore, the blade-to-blade arrangement varies substantially along the span because of the high twist. If the trailing edge indices are required to be constants along the span, the resulting meshes will have fairly high densities on the upper surface and low densities on the lower surface for the outboard stations and vice versa for the inboard stations. One can attempt to choose a set of trailing edge indices so that the best compromise can be reached for practical applications. Our experience shows that the results are marginal at best, as shown in Figures 2 and 3.

When these C-type meshes are actually applied to potential flow calculations, another difficulty arises. At the spanwise station off the blade tip, C-type meshes are generated around the fictitious interior boundary (the camber line). The mesh lines approaching this interior boundary from above, having a higher mesh density, do not coincide with those approaching from below. The ratio of the mesh density can be as large as 5, and it is particularly severe right off the blade tip because the fictitious blade does not have a chance to "untwist." To compute the flow across this fictitious boundary, interpolation of the velocity potential is required. For a high-speed propeller, the local flow there is supersonic, and attempting to interpolate the velocity potential between two sets of data points with large differences in density results in

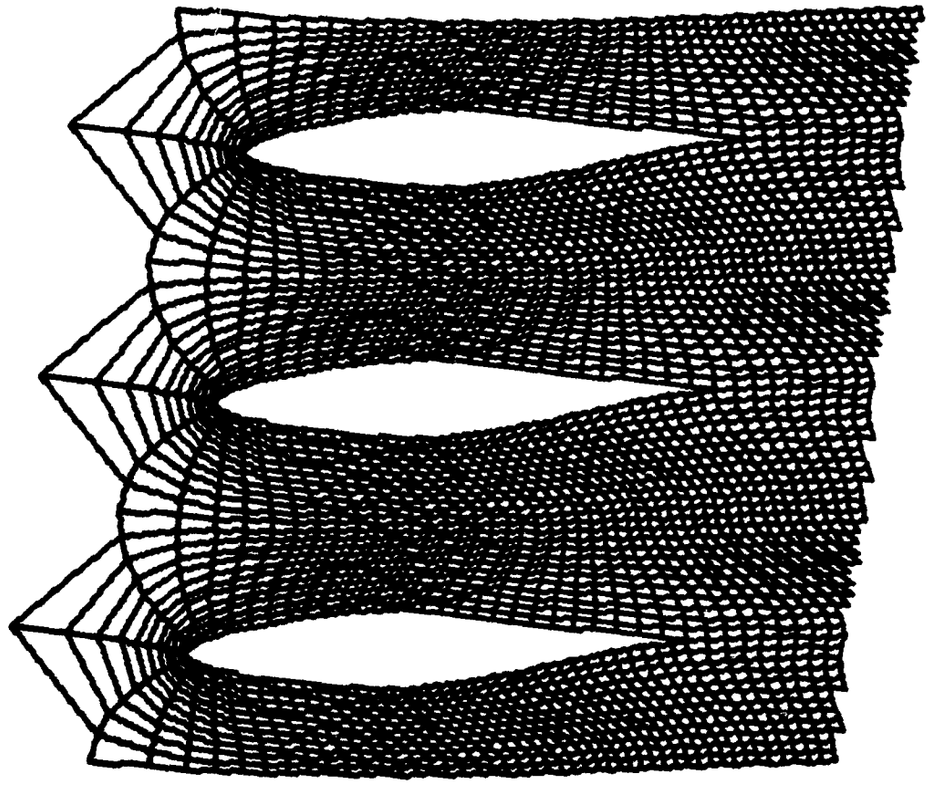


Fig. 2. C-type cascade mesh.
SR-1 Propeller, 8 Blades

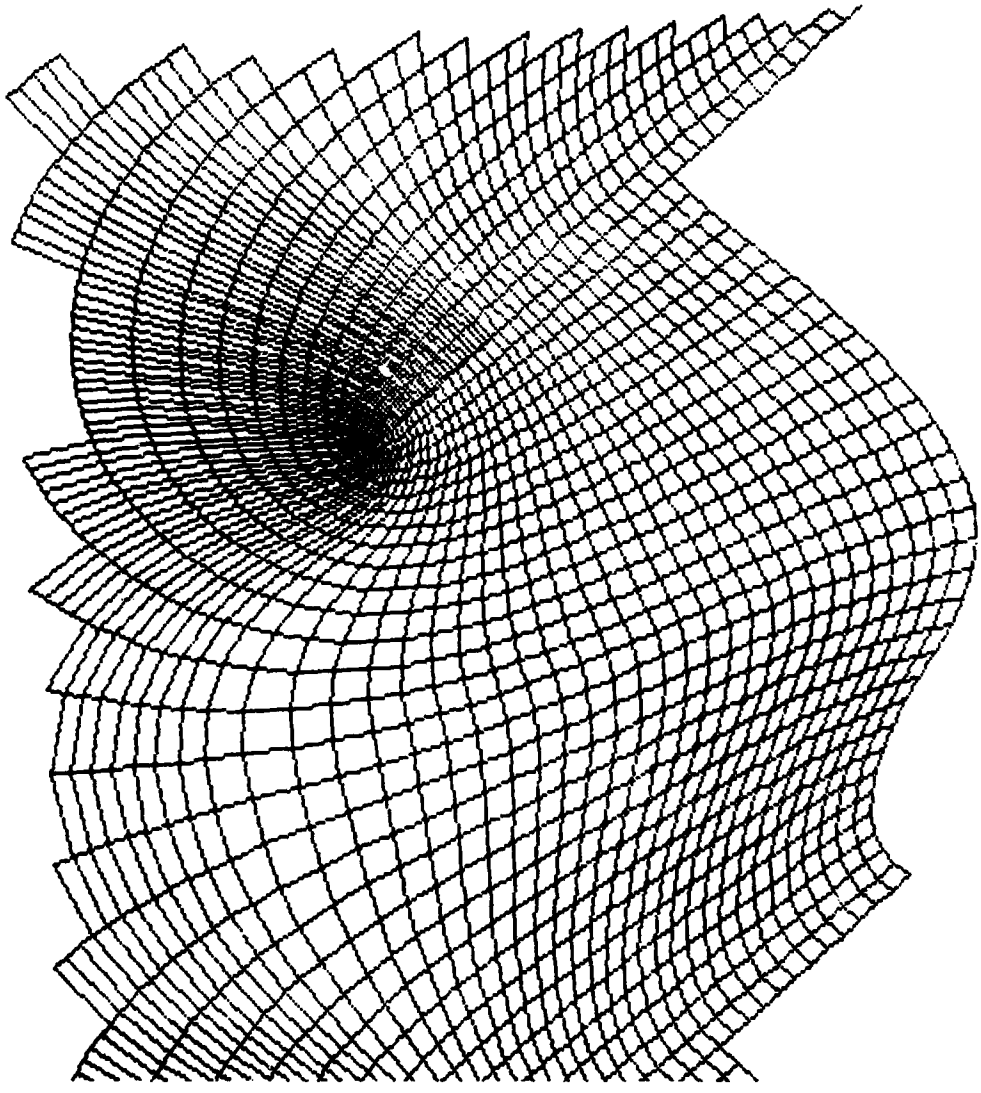


Fig. 3. C-type cascade mesh.
SR-1 Propeller, 8 Blades

undesired disturbances in the supersonic region. The convergence of the iterative scheme is severely affected by this operation.

H-type mesh

The extreme complexity of the geometry, the numerical scheme chosen to solve the specific problem and the physics of the flow force us to look for a better mesh system. The H-type mesh deserves a closer look for this specific configuration for the following reasons:

(1) Because of its inefficiency, the space covered by the mesh can be extended arbitrarily upstream and downstream from the blade providing sufficient distance to cover the hub geometry and to apply the far field boundary conditions.

(2) The blade is mapped to an interior branch cut in the transformed space in which the division into discrete meshes naturally gives continuous mesh lines on the fictitious part of the blade.

(3) A shearing transformation will be required to align the mesh lines on the periodic boundary. For the type of propeller under consideration, it was estimated that the minimum angle between two families of mesh lines will be in the neighborhood of 35 degrees, which, from our experience with the finite-volume calculation, is acceptable.

(4) Because of the ease of generating an H-type mesh by a combination of simple conformal transformations and a series of shearing transformations, the distribution of the mesh density can be fairly easily modified to enhance the numerical solution.

For the range of blade twists considered for the high-speed propeller, the following simple procedure has provided useful results.

(1) Consider the leading edge curvature, κ, of an isolated blade. The following complex transformation, with the origin at the leading edge of the blade, maps the branch cut along the positive axis in ξ space to a parabola in the physical space, z, with a curvature κ at the origin:

$$z = i\xi^{1/2} + \kappa\xi \quad .$$

A set of Cartesian coordinate lines in ξ space is mapped to a streamline/potential-line system in the physical space, and a basic H-type mesh is established around an isolated, semi-infinite parabolic boundary.

(2) To obtain a blade-fitted coordinate system, the parabola is moved to the blade surface by a simple shearing transformation. This shearing also moves the mesh points outside the parabola in a continuous way.

(3) The mesh system around an isolated blade is now manipulated to satisfy the periodic requirements for a cascade configuration. For the range of twist angles and the blade-to-blade distance required, simple shearing transformations produce computationally acceptable results. The periodic boundaries are defined far upstream by choosing two points with the same longitudinal location, x, and with a difference of $2\pi/N$ in the azimuthal angle, where N is the number of blades. The mesh lines passing through these two points and extending to downstream infinity are not straight lines; they are straightened out by the shearing transformations.

(4) The mesh lines in the blade-to-blade direction do not satisfy the periodic conditions. They are now sheared by a cosine shearing function that moves the points on the upper and lower boundaries toward each other. This shearing procedure creates a fairly nonorthogonal mesh near the periodic boundary whi[illegible] preserving near orthogonality on the blade surface.

(5) Further minor shearing transformations are performed, mainly to avoid an extreme mesh aspect ratio in the far field downstream and near the periodic boundaries. These shearing transformations are implemented to enhance the stability of the numerical calculation after the preliminary calculations have been performed using the mesh obtained in steps (1) through (4).

Samples of meshes produced by the procedure described above are shown in Figures 4 through 7.

CONCLUSIONS

The present paper discusses an experience in generating a computational mesh for a rotating propeller. The problem is somewhat unique in that several characteristic length scales are involved, namely the chord of the blade, the length of the blade, the length of the hub and the blade-to-blade distance. These length scales of considerable disparity coupled with other nondimensional quantities, such as the twist angle, create a geometrical complexity beyond the capability of a general mesh generation algorithm. An ad hoc mesh generation scheme must be developed.

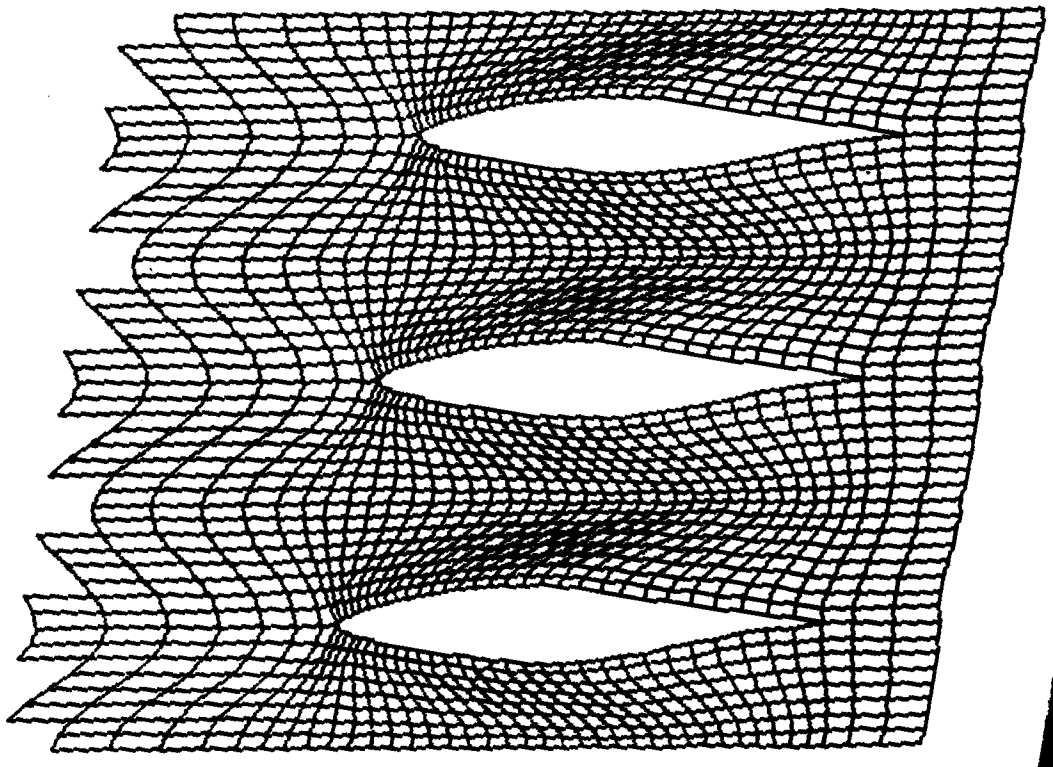


Fig. 4. H-type cascade mesh.
SR-1 Propeller, 8 Blades

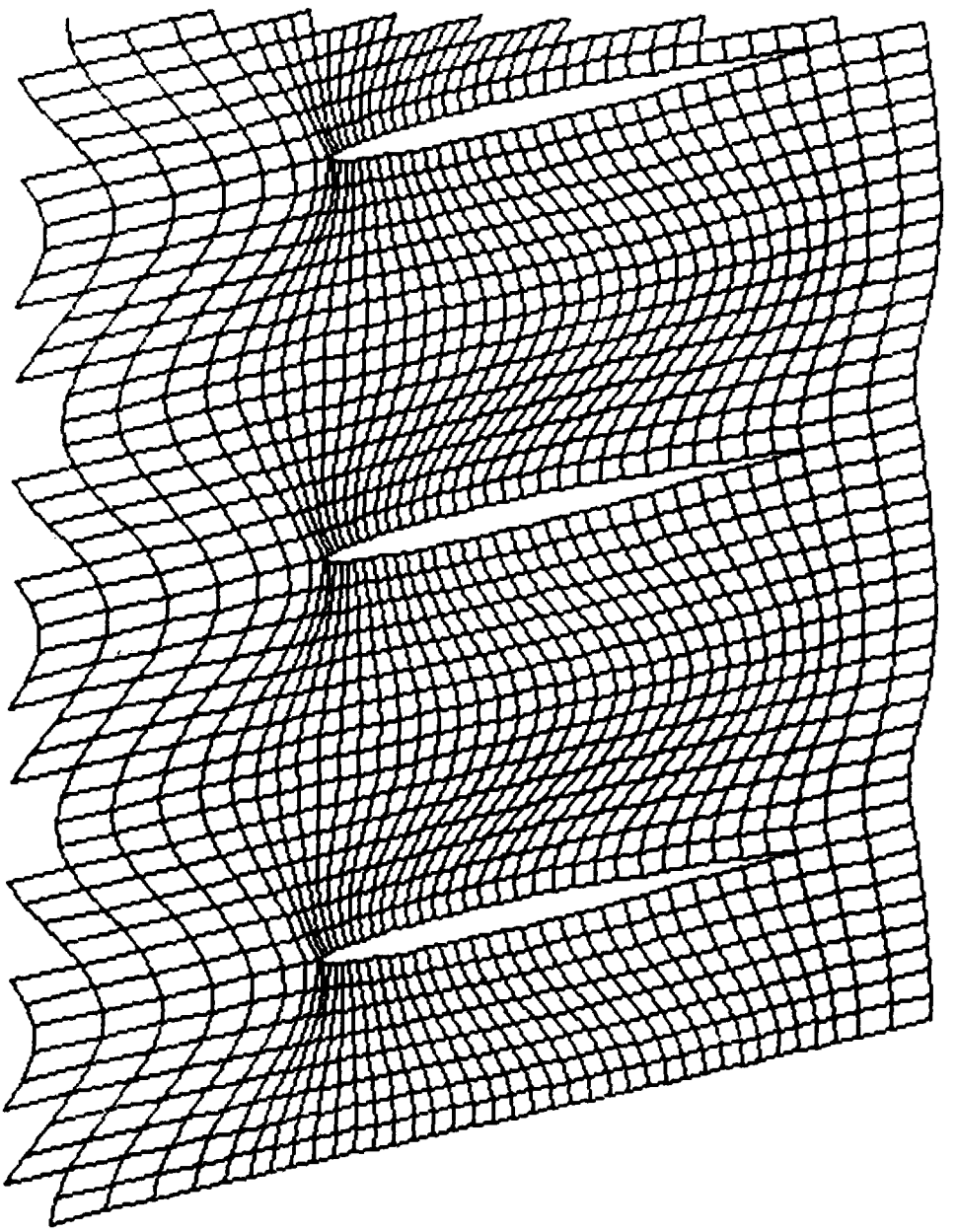


Fig. 5. H-type cascade mesh.
SR-1 Propeller, 8 Blades

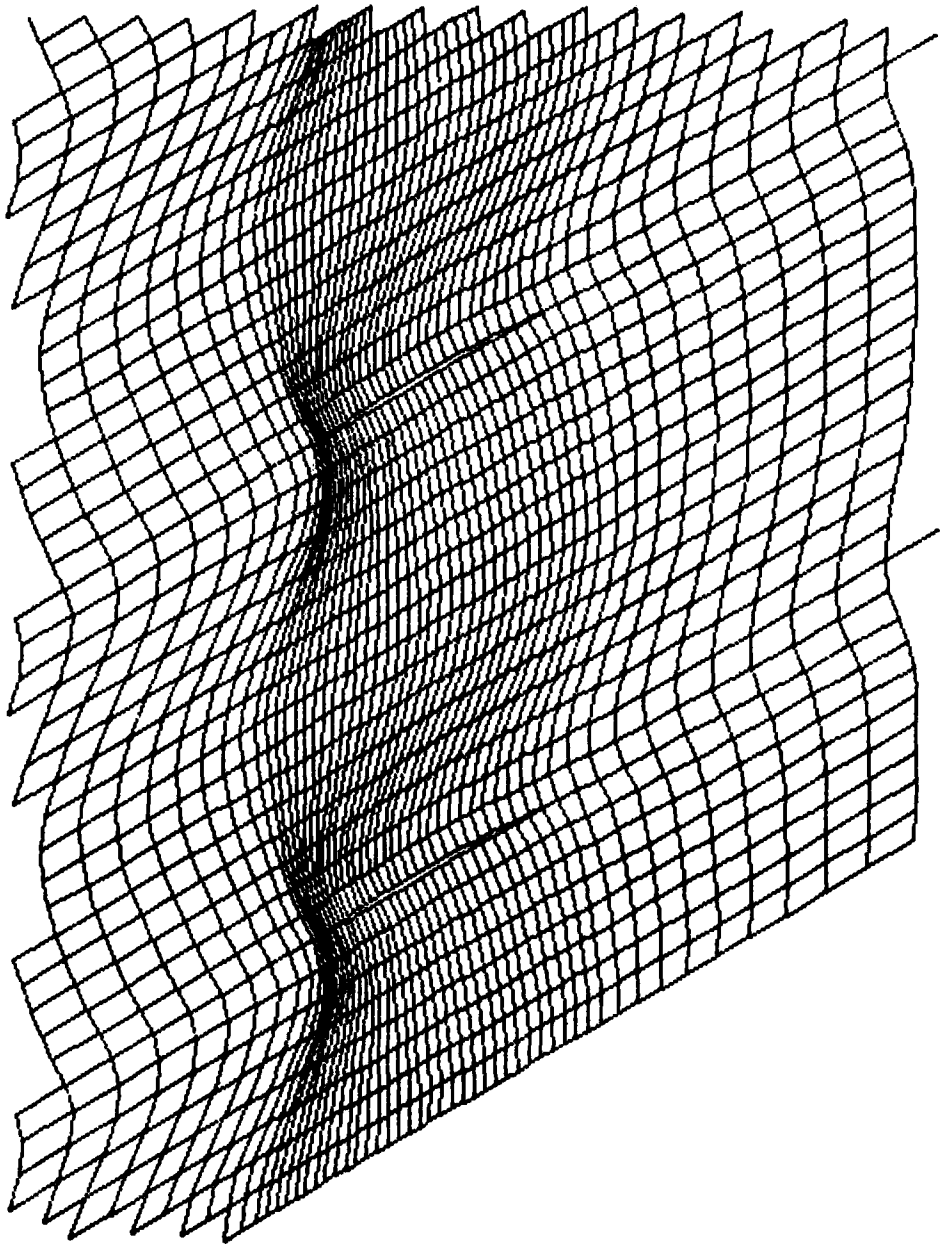


Fig. 6. H-type cascade mesh.
SR-1 Propeller, 8 Blades

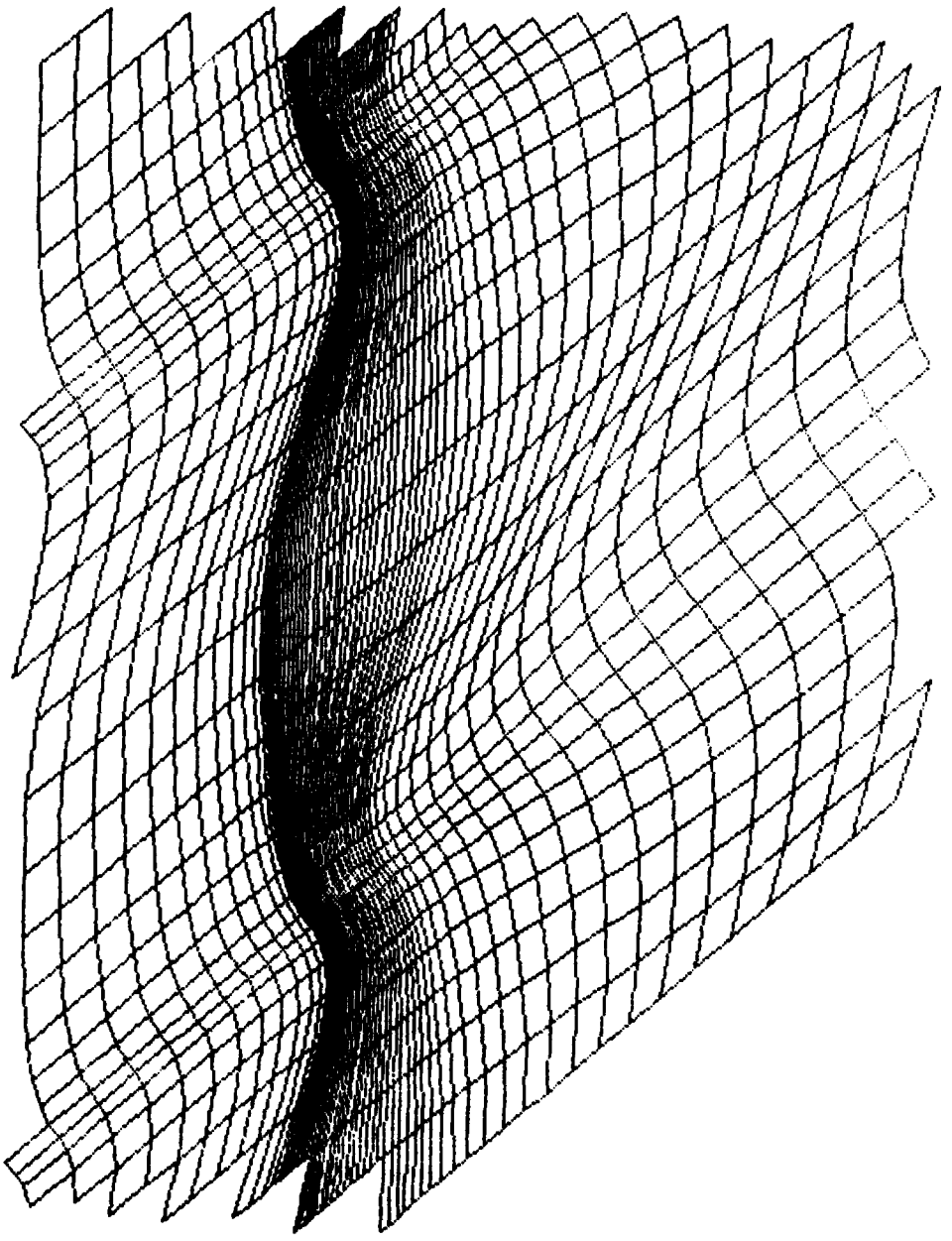


Fig. 7. H-type cascade mesh.
SR-1 Propeller, 8 Blades

The usual practice of dividing the three-dimensional space into two-dimensional surfaces on which two-dimensional meshes will be generated is followed. It is found that the popular O-type and C-type meshes, seemingly attractive from experiences in two-dimensional computations, present some difficulties in the three-dimensional computation. The H-type mesh, commonly considered an inefficient mesh system, is thought to have more flexibility for this particular problem. A nonorthogonal H-type mesh is generated by using a very simple complex-variable transformation followed by a series of shearing transformations. This type of mesh has been used successfully in the numerical computation of transonic flow around a rotating propeller using the finite-volume algorithm.

The properties of the resulting mesh are strongly influenced by the numerical method used, as well as the flow physics involved. The mesh system, the numerical method and the physics involved are integral parts of this complex problem.

ACKNOWLEDGMENTS

This work is supported by NASA Lewis Research Center under Contract No. NAS3-22148. The author acknowledges the useful discussions regarding this work with Lawrence Bober of Lewis Research Center.

REFERENCES

1. Mikkelson, D. C., Blaha, B. J., Mitchell, G. A., and Wikete, J. E. (1977) Design and Performance of Energy Efficient Propellers for Mach 0.8 Cruise, NASA TM X-73612.

2. Bober, L. J., and Mitchell, G. A. (1980) Summary of Advanced Methods for Predicting High Speed Propeller Performance, NASA TM 81409.

3. Jou, W. H. (1982) Finite Volume Calculation of Three-Dimensional Flow Around a Propeller, To be presented at AIAA/ASME 3rd Joint Thermophysics Fluid, Plasma and Heat Transfer Conference, St. Louis, June.

4. Rizzi, A., and Eriksson, L. E. (1981) Transfinite Mesh Generation and Damped Euler Equation Algorithm for Transonic Flow Around Wing-Body Configuration, AIAA Computational Fluid Dynamics Conference, Palo Alto.

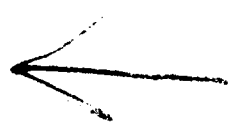
5. Caughey, D. A., and Jameson, A. (1979) Recent Progress in Finite Volume Calculations for Wing-Fuselage Combinations, AIAA Paper 79-1513.

6. Huynh, H., and Jou, W. H. (1982) Singularity Embedding Method in Potential Flow Calculation, To be published in AIAA/ASME 3rd Joint Thermophysics, Fluid, Plasma and Heat Transfer Conference, St. Louis, June.

7. Sorenson, R. L., and Steger, J. L. (1980) Numerical Generation of Two-Dimensional Grids by the Use of Poisson Equations with Grid Control at Boundaries, Numerical Grid Generation Technique NASA Conference Publication 2166, pp. 449-461.

8. Adamczyk, J. J. (1980) An Electrostatic Analog for Generating Cascade Grids, Numerical Grid Generation Technique NASA Conference Publication 2166, pp. 129-142.

9. Jou, W. H. (1981) Calculation of Transonic Potential Flow Over Cascades, Flow Research Report No. 182.



AD P000990

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

FAST GENERATION OF THREE-DIMENSIONAL COMPUTATIONAL BOUNDARY-CONFORMING PERIODIC GRIDS OF C-TYPE

DJORDJE S. DULIKRAVICH*
Universities Space Research Association
Columbia, Maryland 21044

ABSTRACT

A fast computer program, GRID3C, has been developed to generate multilevel three-dimensional, C-type, periodic, boundary conforming grids for the calculation of realistic turbomachinery and propeller flow fields. The technique is based on two analytic functions that conformally map a cascade of semi-infinite slits to a cascade of doubly infinite strips on different Riemann sheets. Up to four consecutively refined three-dimensional grids can be automatically generated and permanently stored on four different computer tapes. Grid nonorthogonality is introduced by a separate coordinate shearing and stretching performed in each of three coordinate directions. The grids can be easily clustered closer to the blade surface, the trailing and leading edges and the hub or shroud regions by changing appropriate input parameters. Hub and duct (or outer free boundary) can have different axisymmetric shapes. A vortex sheet of arbitrary thickness emanating smoothly from the blade trailing edge is generated automatically by GRID3C. Blade cross-sectional shape, chord length, twist angle, sweep angle, and dihedral angle can vary in an arbitrary smooth fashion in the spanwise direction. Input coordinates must be Cartesian, while the output grid coordinates can be Cartesian or cylindrical.

INTRODUCTION

When numerically solving partial differential equations governing the flow of fluid through realistically shaped configurations, exact boundary conditions must be applied at correct locations. This is especially important when calculating internal flows and flows that are governed by nonlinear partial differential equations. Seemingly negligible alterations of geometrical shape or flow conditions at the boundary can drastically change the basic features of the flow field, for example, choking an originally unchoked flow or changing a shock-free flow into a shocked flow[1]. The most economical and accu-

*Visiting Research Scientist with Computational Fluid Mechanics Branch, NASA Lewis Research Center, Mail-Stop 5-9, Cleveland, Ohio 44135.

PREVIOUS PAGE IS BLANK

rate way to numerically apply exact boundary conditions on solid boundaries is to generate and use a computational grid that conforms to these surfaces (fig. 1). Recent numerical techniques do not require orthogonal grids[2] because they use locally isoparametric formulation when numerically determining derivatives of geometric and flow variables. A widely accepted procedure for accelerating an iterative solution process of the flow equations and for resolving or capturing high flow gradients is to perform calculations on a sequence of several successively refined grids. The multigrid technique[3] usually requires four to six such grids. For realistic threedimensional configurations the number of grid points to be generated is prohibitively large even for inviscid flow calculations. Computational grids for such configurations should be easy to regenerate if shock waves and vortex sheets are to be better resolved or if the configuration of the solid boundaries changes with time.

An H-type grid (fig. 1) provides excellent resolution of the flow field at upstream and downstream infinity. It is also the simplest grid to generate. At the same time, H-type grid does not provide for an accurate treatment of rounded leading and trailing edges and wastes points in the flow domains away from the boundaries. An O-type grid represents the other extreme. It gives a very poor resolution at infinities[4], thus creating a problem when Cauchy-type boundary conditions must be enforced at the supersonic inflow boundary (fig. 2). A grid of the O-type also does not provide desirable resolution in the vicinity of the vortex sheet. An open trailing edge simulation of the boundary layer displacement thickness effect cannot be readily incorporated. Nevertheless, an O-type grid provides for accurate discretization of arbitrarily blunt leading and trailing edges and requires a minimum number of grid points. A combination of an O-type grid in the upstream region and an H-type grid in the downstream region creates a C-type grid. This type of grid provides for a good treatment of all boundary and periodicity conditions including wake treatment and supersonic exit flow, although it lacks an adequate resolution at upstream infinity (fig. 1).

In turbomachinery and rotorcraft flow field calculations the flow field is periodic and a geometrically periodic grid provides for a simple and accurate way to enforce the flow periodicity. The simplest and fastest way to generate nonorthogonal periodic grids is to avoid time-consuming techniques based on the numerical solution of sets of partial differential equations whenever possible. Instead, a basic knowledge of complex variables and conformal mapping can be used together with a few additional nonorthogonal coordinate shearings and stretchings. A three-dimensional, periodic, O-type grid generator code was already developed[4] by using this technique, which guarantees that the

grid lines of the same family do not intersect because the basis of the technique is conformal mapping. Another view of a three-dimensional, periodic O-type grid is presented in figure 3.

SHEARING AND STRETCHING IN PHYSICAL SPACE

Conformal mapping can be applied only to two-dimensional plane surface problems. A general procedure for creating such planes can be best described in the case of a rotor mounted on a hub shaped like a doubly infinite circular cylinder and confined inside a doubly infinite circular-cylinder-shaped duct. The intermediate doubly infinite circular-cylinder-shaped surfaces intersecting the blades can be viewed as planes when expressed in terms of $(x,r\theta)$ coordinates. A standard procedure for creating three-dimensional blade shapes is to specify local airfoil shapes on a number of input planes that are orthogonal to a straight radial line. This radial line (z axis in fig. 4) is called a stacking axis, and local blade sweep and dihedral angles are measured from that line (fig. 1). Input planes are defined by z = constant. Intermediate cylindrical surfaces, which we seek for the next step in this grid generation procedure are defined by r = constant. To obtain an intersection contour between the blade surface and r = constant cylindrical surfaces, a spline fitting and interpolation procedure is used along the blade. Input airfoil (x_i,y_i) coordinates on z = constant planes are transformed into cylindrical coordinates

$$x = x_i \tag{1}$$

$$\theta = \arctan(y_i/z_i) \tag{2}$$

$$r = (y_i^2 + z_i^2)^{1/2} \tag{3}$$

Cylindrical coordinates $(x,r\theta)$ are interpolated at r = constant spanwise locations, thus defining blade cross sections on r = constant cylindrical surfaces.

On the other hand, realistically shaped hubs and ducts are not doubly infinite circular cylinders but axisymmetric surfaces. Therefore, the intermediate surfaces are also axisymmetric and not cylindrical. Nevertheless, the same grid generation technique can be used if a simple nonorthogonal shearing (or normalization) and stretching of the radial coordinate (fig. 3) is performed. Nonorthogonal (unidirectional) shearing of the r coordinate converts the axisymmetric surfaces into cylindrical surfaces defined by R = constant. Let subscripts H,T, and D designate R = constant surfaces corresponding to hub,blade tip, and duct (or outer free boundary) location, respectively. Also let the normalized radial coordinate be defined as

$$\bar{R} = \frac{r(x_i) - r_H(x_i)}{r_D(x_i) - r_H(x_i)} \qquad (4)$$

The radial coordinate in the hub-to-tip region is stretched and sheared with the following function

$$R = R_H + (R_T - R_H)((\bar{R}/\bar{R}_T) + A \sin(2\pi \bar{R}/\bar{R}_T)) \qquad (5)$$

The following value was obtained from experience

$$R_H = N/50.0 \qquad (6)$$

The stretching parameter, A, gives best results if

$$0.12 > A > 0.0 \qquad (7)$$

When A = 0, the cylindrical cutting surfaces R = constant are equidistantly spaced from hub to tip. Let the normalized, sheared radial coordinate in the region between the blade tip and the duct (or outer radial boundary) surface be

$$\overset{*}{R} = (\bar{R} - \bar{R}_T)/(\bar{R}_D - \bar{R}_T) \qquad (8)$$

The stretching function for the tip-to-duct domain is chosen to be

$$R = 1.0 + (R_H - q)\overset{*}{R} + q\,\overset{*2}{R} \qquad (9)$$

This function must have the same slope, q, at the location R = 1 as the stretching function in the domain between the hub and the tip (eq. 5).

$$q = (1 + A)(1 - R_H)/R_T \qquad (10)$$

Combining the two stretching functions (eqs. 5 and 9) provides for a smooth and continuous transformation from the physical r coordinate into the sheared R coordinate (fig. 4). For a stator or rotor with no tip clearance, equation 9 is not needed.

Frequently, the input points are not clustered in the same regions on each input plane. Moreover, the number of input points defining the blade cross section on each input plane can vary from one input plane to the next. To accurately determine intersection contours between the blade surface and the axisymmetric surfaces, the corresponding input points must be located at the same percentage of the blade chord length on each input plane. Implicitly, this means that the number of input points must be the same on all input planes. Therefore, these input points must be appropriately redistributed on

each input plane. This redistribution can be performed with respect to the input airfoil contour coordinate defined as

$$s = [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2]^{1/2} \qquad (11)$$

Then the input Cartesian coordinates can be expressed in terms of the input airfoil contour coordinates. Coordinate s is measured clockwise around the input airfoil contour, starting and ending at the trailing edge point. As it was stated earlier, the number of contour points on the pressure surface must be the same as the number of contour points on the suction surface. For non-symmetric airfoils the lengths of these two contour lines are generally not the same. Let ITS denote the trailing edge point on the suction side and ITP denote the trailing edge point on the pressure side of the input airfoil. Also, let LE denote the leading edge, that is, the point that is farthest from the trailing edge. The normalized surface coordinate is defined as

$$\bar{S} = \frac{s - s_{ITP}}{s_{LE} - s_{ITP}} \qquad (12)$$

The redestribution of input points is performed with the following stretching function

$$\bar{\bar{S}} = (1 - \bar{S})\bar{S}^B + \bar{S}[1 - (1 - \bar{S})^B] \qquad (13)$$

where the exponent B should satisfy

$$1.4 > B > 1.0 \qquad (14)$$

When B = 1 the points are equidistantly spaced along the airfoil contour. The points along the pressure surface are redistributed by using the formula

$$S = \bar{\bar{S}}(s_{LE} - s_{ITP}) \qquad (15)$$

and the points along the suction surface are redistributed by using the formula

$$S = \bar{\bar{S}}(s_{ITS} - s_{LE}) + (s_{LE} - s_{ITP}) \qquad (16)$$

This redistribution of input coordinates x and y is performed with a cubic spline fitting applied in the s direction. Interpolation is performed at S locations. Spline fitting and interpolation are also used with respect to the R coordinate in order to find the points on intersection contours between the blade surface and the intermediate axisymmetric surfaces. Locations of those

points in the physical space will not be altered with the subsequent mapping-remapping procedure.

The exact shape of the wake of arbitrary thickness is not known a priori. To eliminate the need for specifying the location of the wake in the preparation of the input, the shape of the wake centerline is automatically generated by using the simple polynomial expression

$$y = a(x - x_{TE})^3 + b(x - x_{TE})^2 + c(x - x_{TE}) + y_{TE} \quad (17)$$

Here the trailing edge point coordinates are

$$x_{TE} = (x_{ITP} + x_{ITS})/2 \quad (18)$$

and

$$y_{TE} = (y_{ITP} + y_{ITS})/2 \quad (19)$$

The point where the wake centerline intersects the downstream-infinity cutoff boundary is defined with the subscript EX. Let c be the average slope of the pressure and suction surfaces of the airfoil at the trailing edge , and let d be the slope of the expected flow angle at the exit boundary. Then the constants a and b in equation 17 are

$$a = [x_w(c + d) - 2\, y_w]/x_w^3 \quad (20)$$

and

$$b = [3y_w - x_w\,(2c + d)]/x_w^2 \quad (21)$$

where

$$x_w = x_{EX} - x_{TE} \quad (22)$$

and

$$y_w = y_{EX} - y_{TE} \quad (23)$$

Wake surface grid points are redistributed (stretched) with the formula

$$x^* = (x - x_{TE})/x_w - n \sin(\pi(x - x_{TE})/x_w) \quad (24)$$

The stretching exponent, n, is determined from the continuity of the slope of the stretching functions at the trailing edge (eqs. 13, 15, 16, 22, and 24)

$$n = 1.05(1.0 - B/2x_w)/\pi \quad (25)$$

If the wake has a finite thickness, that is, if the blade trailing edge is open, coordinates of the points on the upper and lower surfaces of the wake are determined by adding and subtracting the trailing edge half thickness. The axial coordinate of the upper surface of the wake is determined from the formula

$$x^u = x + (x_{ITS} - x_{ITP})/2 \tag{26}$$

and that of the lower surface of the wake by the formula

$$x^l = x - (x_{ITS} - x_{ITP})/2 \tag{27}$$

with similar expressions for the y coordinate. Superscripts u and l designate the upper and lower surfaces of the wake, respectively.

CONFORMAL MAPPING AND REMAPPING

The conformal mapping portion of the present procedure for generating three-dimensional, periodic C-type grids was originally used by Sockol to generate orthogonal, two-dimensional, cascade C-type grids. If the blades are straight, semiinfinite twisted plates of zero thickness, their intersections with circular cylinders generates doubly infinite cascades of semi-infinite straight slits on each of the (x,Rθ) planes (fig. 5). Each of these R = constant planes can be defined in terms of complex variables

$$w = x + iR\theta \tag{28}$$

The goal is to generate a boundary-conforming, periodic C-type grid on each of the planes. This task is accomplished by conformally mapping the w plane via an intermediate "circle" complex plane (fig. 6)

$$v = \xi + i\eta \tag{29}$$

into the interior of a "doubly infinite strip" plane (fig. 7)

$$u = X + iY \tag{30}$$

Uniform grid in the u plane is then conformally remapped into the w plane, thus generating the desired C-type grid. As shown by Sockol[4] a single analytic function

$$w = w_{LE} + e^{i\beta}\frac{R}{N}(2\beta \sin\beta + 2\cos\beta \ln(2\cos\beta))$$

$$+ e^{-i\beta}(\ln v - i\pi) - 2\cos\beta \ln(1 - v) \qquad (31)$$

where N is the number of blades and β is the local stagger angle on the R = constant surface, conformally maps the interior of the unit circle in the v plane to the interior of a periodic strip enveloping a semi-infinite slit in the w plane. The center of the circle ($v = 0$) maps into upstream infinity in the w plane and the point $v = -1$ maps into downstream infinity in the w plane. The zero-thickness slit between the points $v = 0$ and $v = -1$ maps into the upper and lower periodic boundary of a periodic strip in the w plane. The circle in the v plane maps into a semi-infinite straight slit in the w plane. A doubly infinite cascade of semi-infinite straight slits in the w plane is thus created by conformally mapping a doubly infinite cascade of Riemann sheets (v planes) that are interconnected through the slits between the points $v = 0$ and $v = -1$. Sockol[4] used a simple analytic function

$$v = \tanh(u^2/2) \qquad (32)$$

to conformally map the interior of a doubly infinite straight strip in the u plane into the interior of a unit circle in the v plane. The lower strip boundary ($Y = -i\pi/2$) in the u plane maps into the circle in the v plane. The upper strip boundary ($Y = 0$) maps into a zero-thickness slit between the points $v = 0$ and $v = -1$. Axial infinities ($X = \pm\infty$) map into a single point ($v = -1$). The origin ($X = 0; Y = 0$) in the u plane maps into the origin ($v = 0$) in the v plane.

Realistically shaped blade airfoils are not straight semi-infinite lines of zero thickness. A C-type grid generated with the use of equations 31 and 32 alone will not conform to the actual airfoil cascade shapes on R = constant surfaces. To generate a C-type grid that conforms to the shape of the airfoil and wake, several nonorthogonal coordinate shearings and stretchings are used. Airfoil surface points are conformally mapped from the w plane via the v plane into the u plane. As a result, the circle in the v plane becomes deformed (fig. 7), and the corresponding lower wall in the u plane becomes an irregular line (fig. 8). The inverse of equation 31 cannot be analytically obtained for staggered cascades. Therefore, a Newton-Raphson procedure is used to iteratively evaluate on a point-by-point basis the pairs of (ξ,η) coordinates corresponding to the given pairs of $(x,R\theta)$ coordinates. By using an analytic inverse of equation 32, that is,

$$u = [\ln(\frac{1+v}{1-v})]^{1/2} \qquad (33)$$

the deformed circle is conformally mapped from the v plane into the u plane.

SHEARING AND STRETCHING IN COMPUTATIONAL SPACE

It should be pointed out that with the increase in stagger angle in the w plane the image of the leading edge point shifts along the deformed circle in the v plane and along the deformed lower boundary in the u plane. To insure that the corresponding points along the periodic boundaries in the w plane have the common values of x coordinate, their images in the u plane are placed symmetrically along the Y = 0 line (fig. 9). At the same time these periodic points are distributed with a simple stretching function

$$X^U = X^U - e \sin(2\pi X^U/(X^L_{ITS} - X^L_{ITP})) \qquad (34)$$

Superscript U denotes the upper wall (Y = 0) of the u plane and superscript L denotes the lower irregular boundary of the u plane. The stretching coefficient e is determined from experience as

$$e = 0.18 - 0.05 \ln(2R\pi/Nt) \qquad (35)$$

where t is the local blade chord. The periodic grid points located in the wake region are redistributed by using the expression

$$X^U = X^U - f \sin(2\pi(|X|^U - X^U_{ITS})/(X^L_{MAXXP} - X^U_{ITS})) \qquad (36)$$

where MAXXP denotes the last point on the upper surface of the wake. The stretching coefficient, f, is determined also from the experience as

$$0.10 > f > 0.05 \qquad (37)$$

Because only a finite length of the wake is conformally mapped from the w plane into the u plane, the deformed strip in the u plane has a finite length. The shape of the end wall boundaries in the u plane are determined so that they meet the lower boundary of the strip in the u plane almost orthogonally (fig. 8). Consequently, grid orthogonality is well preserved at the wake. Coordinates of the grid points inside the strip in the u plane are determined from

$$Y = Y^L((Y/Y^L) + g \sin(\pi Y/Y^L)) \quad (38)$$

and

$$X = X^U + (X^L - X^U)\ ((Y/Y^L) + C \sin(\pi Y/Y^L)) \quad (39)$$

where

$$0.30 > C > 0.15 \quad (40)$$

$$g = C\ (1.0 - 1.0/\cosh h) \quad (41)$$

$$h = 5\ (X^U/X^U_{MAXXP}) \quad (42)$$

Stretching coefficients C, g, and h are determined from experience and from the condition that C-type grid lines in the w plane closely follow the wake contour. Larger values of C generate grids suitable for viscous flow calculations, because grid layers are positioned closer to the blade and wake surface.

The resulting two-dimensional nonorthogonal periodic grid in the u plane is conformally mapped back into the w plane on a point-by-point basis. Finally, determination of the physical r coordinates of the grid points on the (x,Rθ) planes is obtained by reshearing the R coordinate (eqs. 4, 5, 8, and 9) and fitting it with respect to the x coordinate with a cubic spline.

RESULTS

On the basis of the preceding analysis, a computer program GRID3C was developed and tested[6]. Program GRID3C consists of 1150 card statements and requires approximately 500 K of computer memory. Because of the analytical character of most of the transformations used, GRID3C is very fast. To generate and permanently store x,y,z coordinates of a typical four-level grid sequence consisting of (33*8*6),(63*13*11),(123*23*21),(243*43*41) grid points, respectively, GRID3C requires between three and four minutes of CPU time on an IBM 370/3033 computer. The Newton-Raphson iterative point-by-point mapping procedure of the airfoil and wake contour from the w plane into the v plane consumes most of the computer time. But this procedure needs to be performed only once on each axisymmetric surface.

Input to GRID3C must be provided in the x,y,z coordinate system, while the output grid coordinates can be computed in the x,y,z or x,r,θ coordinate system. GRID3C can automatically generate up to four successively refined three-dimensional grids and store them on four separate tapes. Computational

grids for the blades with closed trailing edge (fig. 10) and for the blades with open trailing edge (fig. 11) can be generated with GRID3C code. For repetitive runs with different numbers of blades or different blade setting angles, only one input parameter needs to be changed in the input deck. Clustering of grid points closer to the leading and trailing edges and closer to the blade and vortex sheet surface (fig. 12) can be easily achieved by varying coordinate stretching parameters A, B, and C. Grid nonorthogonality is almost entirely removed from the airfoil and wake surface. Nevertheless, grid nonorthogonality can become intolerable if this grid generation technique is applied to closely spaced, highly staggered and cambered blades. Nonorthogonality can become excessive in the leading edge region of any blade if the end point of the semi-infinite slit in the w plane is not positioned approximately midway between the leading edge and its center of curvature.

An unsatisfactory grid resolution inherent to the C-type grids can be observed in figure 13. This figure shows a rectangular wing - cylindrical fuselage combination and two computational grid surfaces: one corresponding to the surface of the fuselage and the other being an intermediate surface located between the hub and the wing tip. Note that the wing extension beyond the tip has linearly increasing cord length. The GRI3C code automatically calculates wing (or blade) chord lengths at the off-tip locations based on the constraint that gap-to-chord ratio at the tip should be retained at all outer spanwise locations. Key elements of a three-dimensional C-type grid generated by the GRID3C code for an advanced, eight-blade, transonic, NASA propeller is presented in figures 14 and 15 with intersection contours between a blade and the axisymmetric sur- faces shown. Note the large twist, sweep, and taper variations and the fact that the propeller hub is axisymmetric.

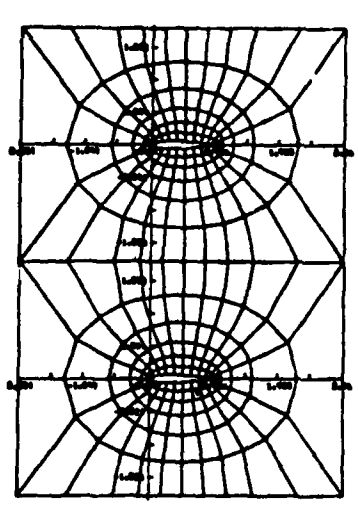
With minor modifications GRID3C can be used for generating computational grids applicable to a midmounted wing-body combination or a finned missile in free air or inside a wind tunnel having axisymmetric walls.

ACKNOWLEDGMENTS

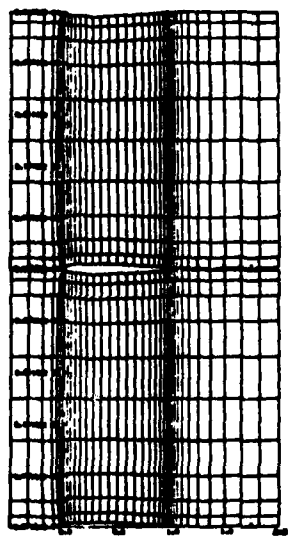
The Computational Fluid Mechanics Branch of the NASA Lewis Research Center provided computational facilities used in this work. The author would also like to express his special gratitude to Dr. Robert Stubbs, Ms. Carol Vidoli, and Ms. Pamelia Caswell of the NASA Lewis who reviewed and edited the manuscript and to Dr. Charles Putt of NASA Lewis Computer Services Division, Dr. Bharat Soni of Sverdrup Technology Inc., and Mr. William Usab of MIT and United Technologies Research Center who exercised the computer codes and provided several grid displays.

REFERENCES

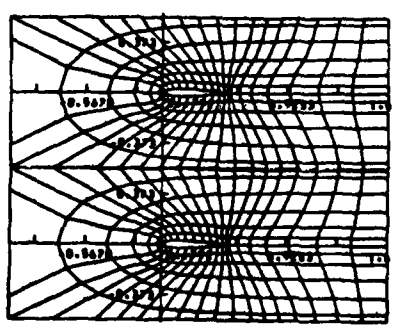
1. Dulikravich, D.S.and Sobieczky, H., "Shockless Design and Analysis of Transonic Blade Shapes," AIAA Paper 81-1237; also NASA TM-82611, 1981.
2. Dulikravich, D.S.," Numerical Calculation of Inviscid Transonic Flow Through Rotors and Fans," Ph.D.Thesis, January 1979.(available through University Microfilms International, 300 N.Zeeb Road, Ann Arbor, MI 48106)
3. Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method," AIAA 4th Computational Fluid Dynamics Conference, July 23-25, 1979, Williamsburg, Va., pp.122-146.
4. Numerical Grid Generation Techniques, NASA CP-2166, 1980.
5. Dulikravich, D.S., "GRID3O-Computer Program for Fast Generation of Multi-level, Three-Dimensional, O-type Boundary Conforming Computational Grids," NASA TP-1920, 1981.
6. Dulikravich, D.S., "GRID3C-Computer Program for Generation of C-Type, Three-Dimensional, Boundary-Conforming, Periodic Grids," NASA CR-167846, 1982.



O-TYPE GRID



H-TYPE GRID



C-TYPE GRID

Figure 1. - Three basic types of two-dimensional, conforming, computational grids.

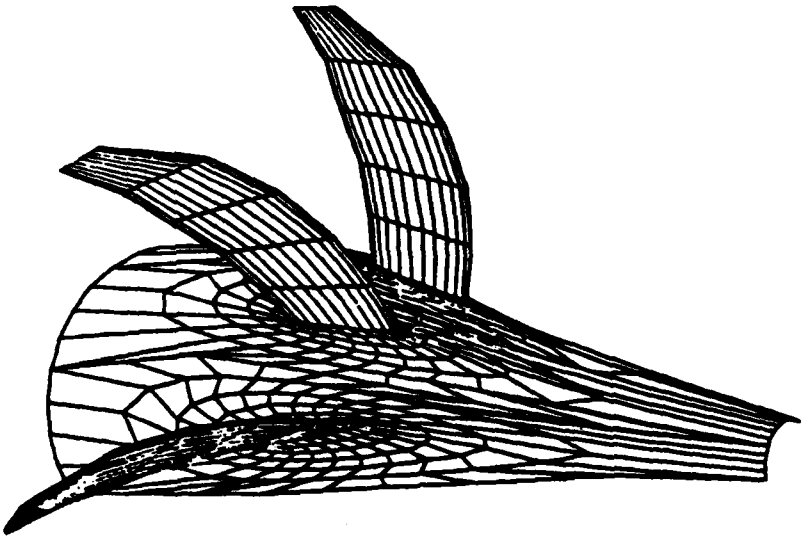


Figure 2. - Axisymmetric view of a three-dimensional, O-type, periodic boundary conforming grid for NASA eight-blade transonic prop fan. Shown are the hub surface grid and three neighboring blades with their surface grids.

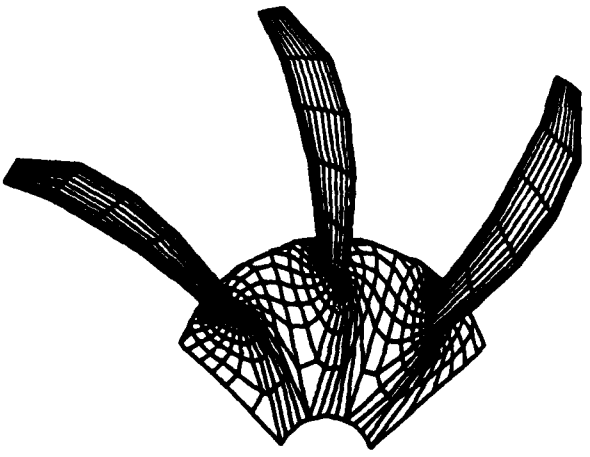


Figure 3. - Frontal view of the O-type grid for NASA eight-blade transonic prop fan.

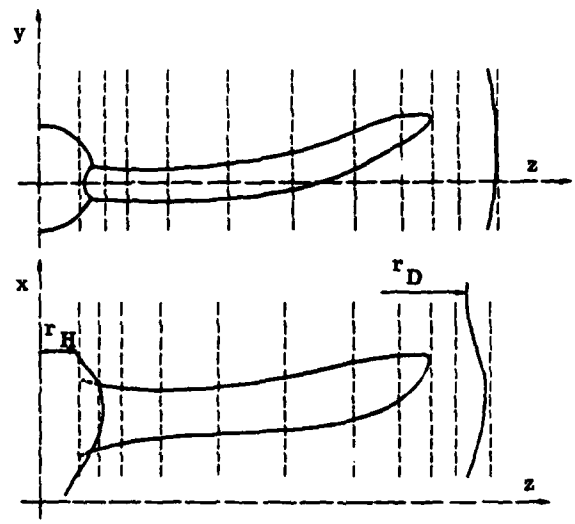


Figure 4. - Spanwise input planes (stations) perpendicular to the stacking (z) axis. Physical x,y,z coordinate system rotates with the blade.

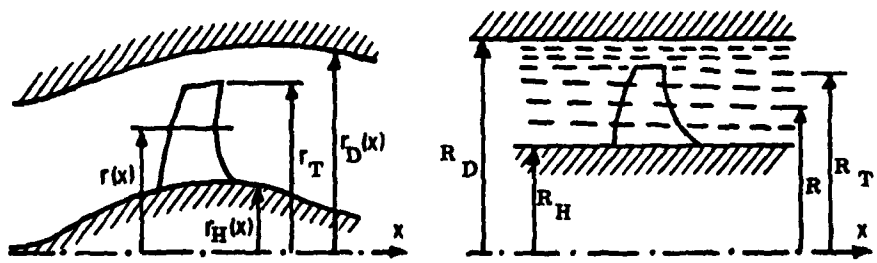


Figure 5. - Radial coordinate nonorthogonal sheering concept.

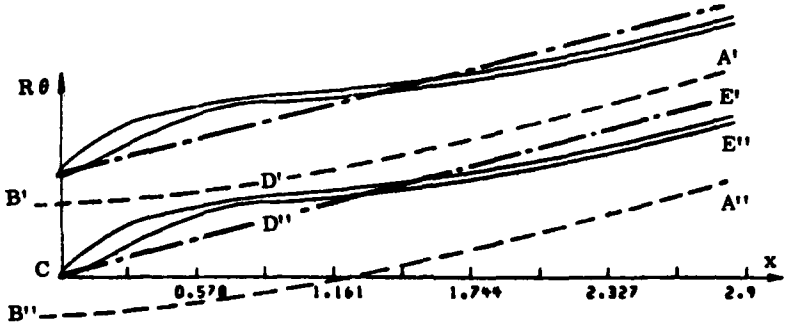


Figure 6. - Two-dimensional cascade of semi-infinite staggered slots of zero thickness with an indication of a cascade of realistically shaped airfoils and their wakes.

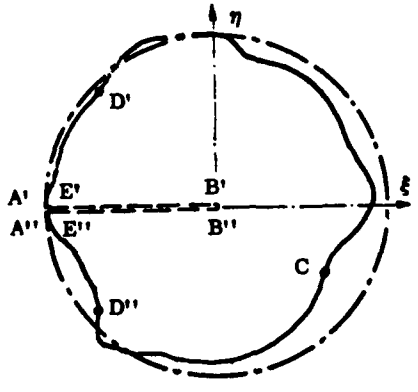


Figure 7. - Intermediate ("circle") plane used in conformal mapping sequence. Deformed circle corresponds to the realistic airfoil shape.

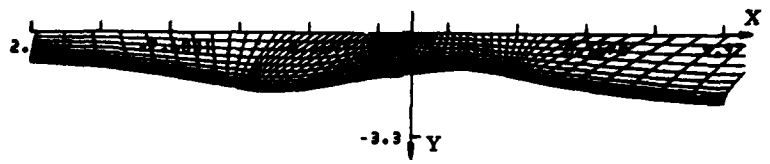


Figure 8. - "Strip" plane obtained by conformally mapping "circle" plane. Upper boundary corresponds to periodic boundaries, and lower boundary to airfoil shape.

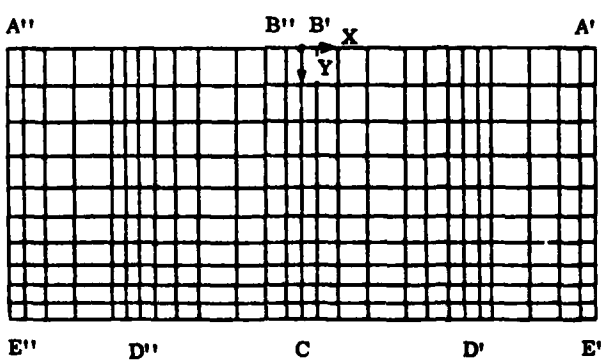


Figure 9. - Nonorthogonal coordinate shearing and stretching concept applied to X (eqs. 34, 36, and 39) and Y (eq. 38) coordinates results in a desired rectangular computational surface.

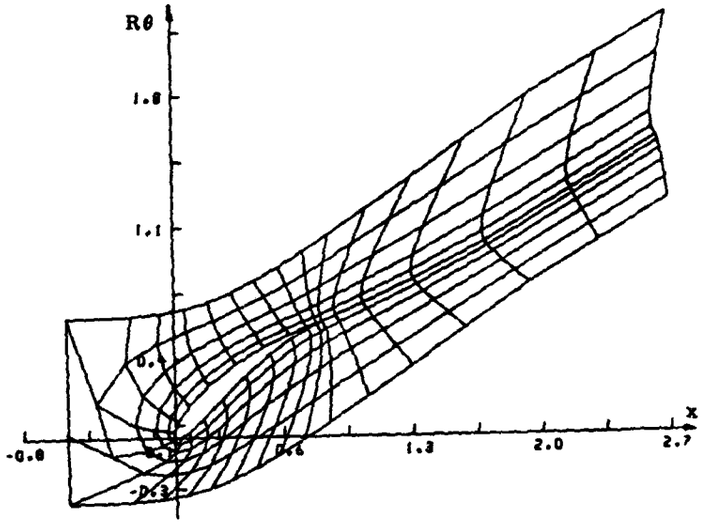


Figure 10. - An example of a two-dimensional (x,Rθ) surface discretized with a coarse, C-type, periodic grid.

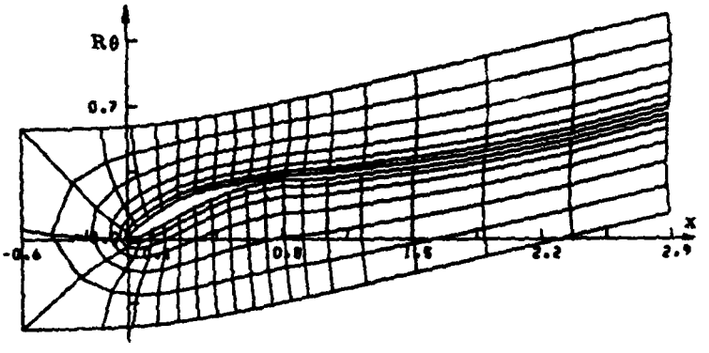


Figure 11. - Two-dimensional (x,Rθ), C-type, periodic, boundary conforming grid for a cascade of blades with open trailing edges.

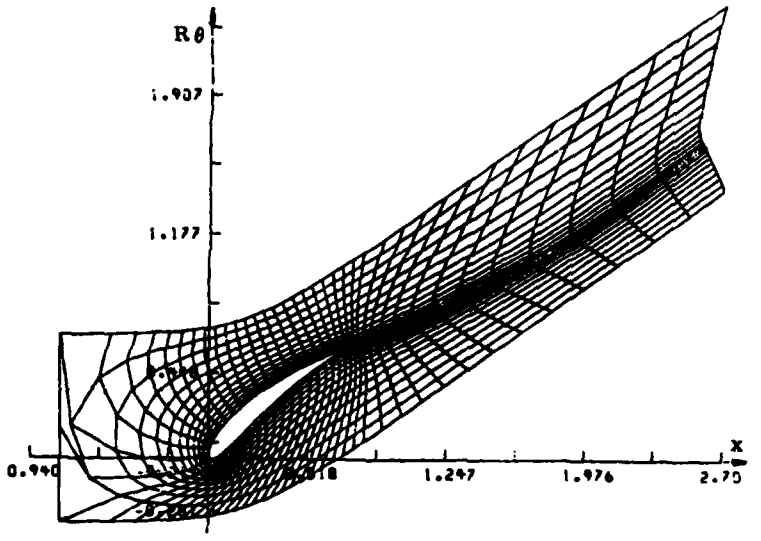


Figure 12. - Effect of controlled grid clustering. Grid points can be easily concentrated in the regions of leading and trailing edges as well as closer to the surface of the airfoil and its wake.

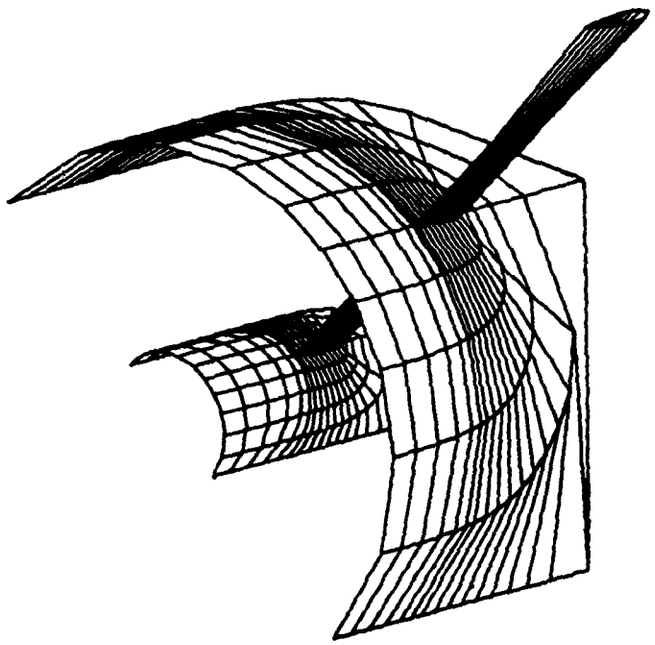


Figure 12. - Elements of a three-dimensional, C-type, periodic grid generated by GRID3C code for a geometry consisting of a rectangular unswept wing attached to a circular cylinder. Note deteriorating grid quality in the far upstream region. Only every fourth cylindrical surface is shown.

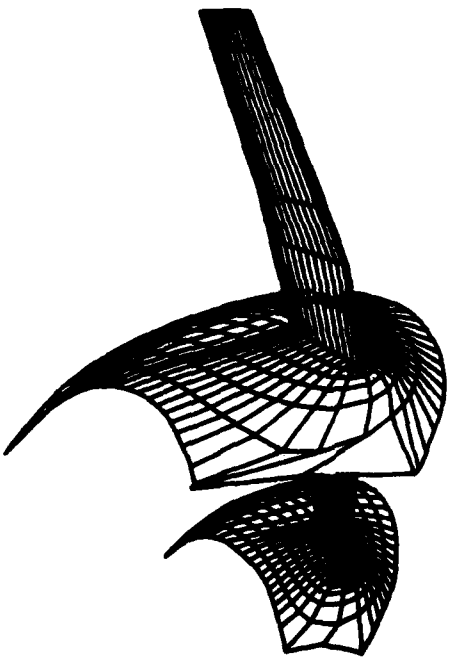


Figure 14. - Blade surface grid and one of the axisymmetric surfaces generated by GRID3C for an advanced, eight-blade NASA prop fan.

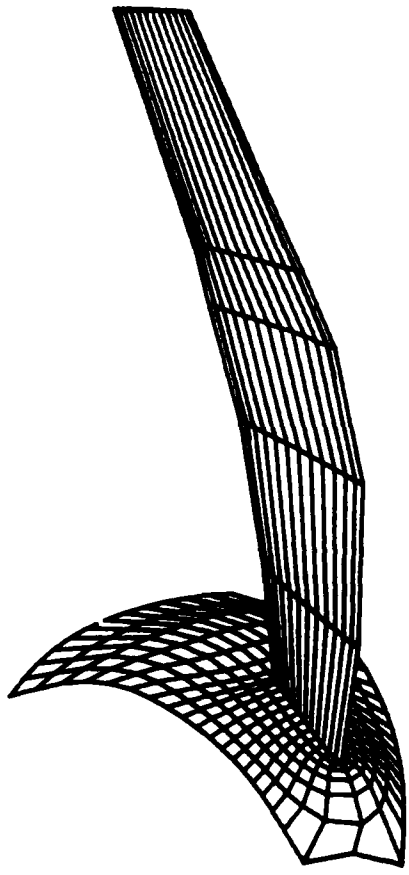


Figure 15. - Another view of the same prop fan grid generated by GRID3C shows more clearly the axisymmetric shape of the propeller hub surface.

AD P000991

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

CONFORMAL GRID GENERATION FOR MULTIELEMENT AIRFOILS*

DOUGLAS HALSEY+
+Aerodynamics Research Department, Douglas Aircraft Company, 3855 Lakewood Blvd., Long Beach, California, 90846 USA

INTRODUCTION

Conformal mapping provides an effective means of generating suitable grids for use in the numerical solution of many two-dimensional flow problems. There are numerous examples of its use for problems involving single-element airfoils, including the well-known finite-difference transonic flow codes of Garabedian and Korn[1] and Jameson[2]. The present author[3] has found conformal mapping to be especially useful in computing compressible potential flow using an integral-equation (or field-panel) approach similar to that used by Wu and Thompson[4], Luu and Coulmy[5], and others. In this approach, a body is analyzed in an equivalent inviscid, incompressible flow with distributed singularities in the external field. The distribution of the singularities is determined in an iterative manner, using the fully nonlinear field equation, the computed flow field, and the appropriate solid-body boundary conditions. Application of a conformal mapping to this problem simply modifies the magnitudes of the singularity strengths and the boundary conditions, without changing the general form of either. The regular spacing in the transformed plane allows the application of very efficient numerical procedures which make effective use of the fast Fourier transform algorithm. For example, using the grid transformations shown in figure 1, accurate subsonic compressible flow solutions for single-element airfoil cases have been obtained in less than two seconds of CPU time on an IBM 370 computing system.

Conformal mapping has also been used for problems involving two-element airfoils. For example, the finite-difference transonic flow code developed by Grossman and Volpe[6] makes use of a mapping, developed by Ives[7], in which the region outside two airfoil elements is transformed to the annular region between two concentric circles. In that case, however, it was necessary to apply nonconformal shearing transformations to the resulting grids, in order

*This research was sponsored by the Independent Research and Development Program of the McDonnell-Douglas Corporation.

to obtain suitable point spacing distributions. This illustrates the unfortunate fact that conformal mapping methods do not allow the degree of control over grid point spacing offered by some other methods, such as the differential equation methods of Thompson[8].

Conformal mapping has not yet been used to generate grids for flow problems involving general multielement airfoils with more than two elements (at least, not to the author's knowledge). This development has been hindered by the absence of any suitable conformal mapping methods for general multielement airfoils. However, the recent development of such methods by the present author[9] and by Harrington[10] should result in the increased use of conformal mapping for multielement airfoil problems.

This paper describes recently-developed techniques applicable to cases involving general multielement airfoils having any number of airfoil elements. Each technique can be considered as a purely geometric construction or, equivalently, as a network of streamlines and potential-lines of an auxiliary potential-flow solution. The nonuniqueness of such solutions ensures the existence of a wide variety of conformal grid types, each having different point-spacing characteristics. A chronicle is given of the search for the type of grid most suitable for solving the inviscid compressible flow equations using a distributed-source field-panel approach. Examples are shown for grids involving up to four airfoil elements.

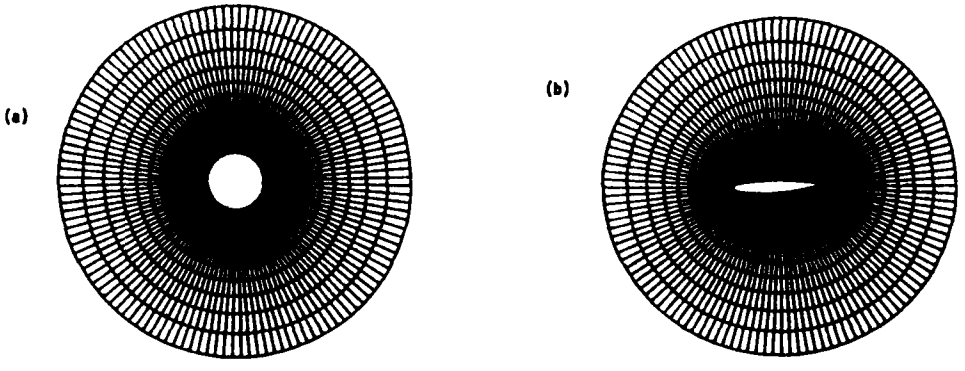


Fig. 1. Typical conformal grid for a single-element airfoil.
(a) Circle plane.
(b) Physical plane.

CONFORMAL MAPPING OF MULTIELEMENT AIRFOILS

A prerequisite for the development of conformal grid generation techniques for multielement airfoils is the existence of a method for transforming the multiple airfoil elements to a system of bodies having much simpler geometry. Such a method has been developed by the present author[9]. This method makes use of a succession of comparatively simple single-body transformations to solve the more difficult multiple-body problem. In the first step, a succession of inverse Karman-Trefftz mappings is applied. Each of these mappings removes a single corner (or a pair of corners in some cases) from a single body, and also causes smaller perturbations to the shapes of the other bodies. At the end of this step, there are no corners and, in most cases, all bodies are quasi-circular in shape. The next step is an iterated sequence of mappings, each of which maps a single body to a perfect circle, and also causes small perturbations to the other bodies. At the end of each iteration of this sequence, the final body is perfectly circular and the other bodies are more nearly circular than at the end of the previous iteration. Iterations proceed until all bodies are sufficiently close to perfect circles and the derivative of the mapping function converges to within a sufficiently small tolerance. Because of the rapid decay with distance of the effects of

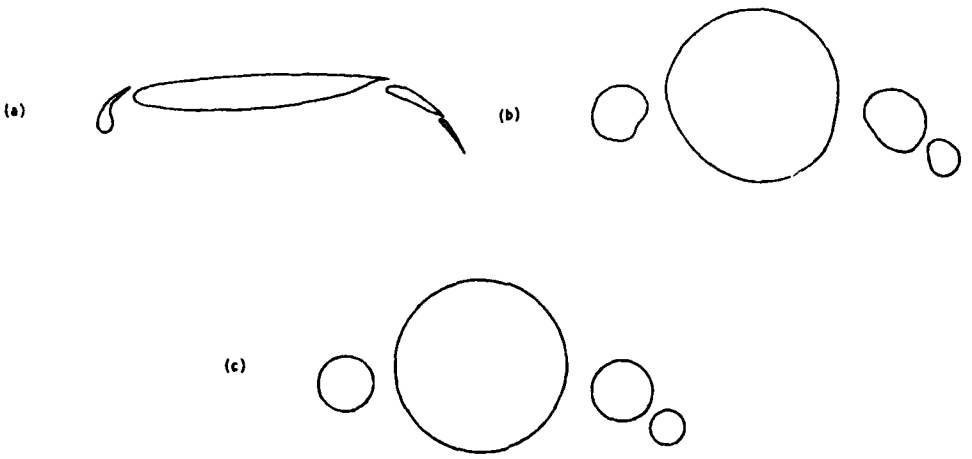


Fig. 2. Transformation of a four-element airfoil into four circles.
(a) Physical geometry.
(b) Geometry after four corner-removing mappings.
(c) Geometry after four circle mappings.

these mappings, the entire procedure converges extremely quickly. Three to five iterations are usually sufficient to give four-digit accuracy. The entire procedure usually requires less than three CPU seconds on an IBM 370 computer. This process is illustrated in figure 2, which shows a four-element airfoil, the geometry after the removal of all corners, and the geometry after only one iteration of the circle mappings. An extension of this procedure to allow airfoils with open trailing edges to be included is described in reference 11.

GRIDS USING A STRING MAPPING

One straightforward grid generation procedure for multielement airfoils involves stringing the airfoil elements together into a single effective body, in a manner reminiscent of Thompson's treatment of multielement airfoils[8]. In this approach, it is not necessary to use a conformal mapping method for multielement airfoils, although the geometric problems are simplified somewhat if the bodies have been previously transformed to circles. This grid generation procedure requires the following steps: 1) String the bodies together into a single body and order the points in a continuous array around the perimeter of the effective body. 2) Apply the Karman-Trefftz transformation successively to remove the resulting corners in the effective body. 3) Transform the resulting body into a perfect circle. 4) Set up the grid in the circle plane. 5) Perform the transformations in reverse order, bringing the grid points back to the multiple-circle plane and finally back to the physical plane. The steps in this process are illustrated in figure 3 for a three-element airfoil case. For step 3, a very robust circle mapping method is necessary, since the shapes to be transformed are too complicated for more limited methods. A comparison of two alternative circle mapping methods is given in reference 12.

Grids produced by this technique for two- and three-element airfoil cases are illustrated in figure 4. These grids are very similar to single-element grids, such as the one illustrated in figure 1 and the flow calculation technique of reference 3 can be directly applied. Point density in these grids is suitably high near the leading edge of the forward element and the trailing edge of the aft element, but there are undesirable sparse areas between the airfoil elements. Although these sparse areas may not cause serious errors in some calculations, they clearly limit the general applicability of this grid generation technique.

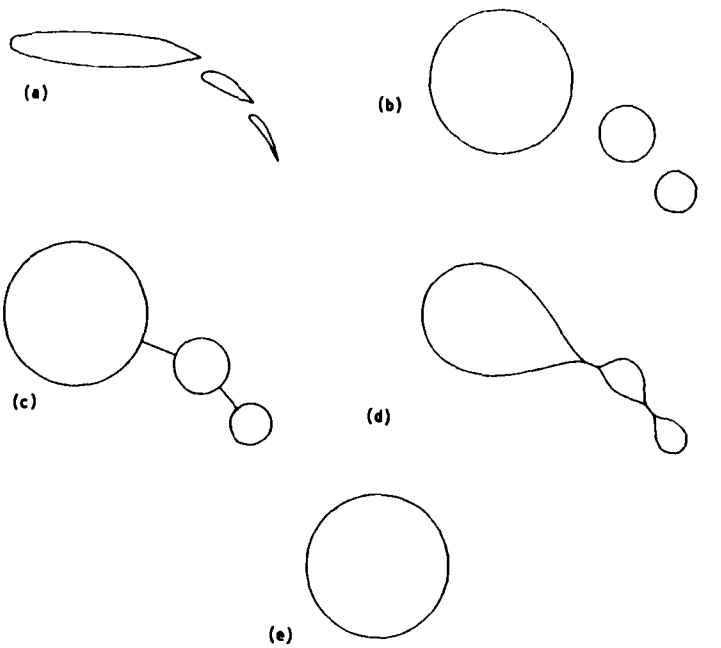


Fig. 3. "String mapping" for a three-element airfoil.
(a) Physical geometry.
(b) Geometry in multiple-circle plane.
(c) Single body produced by stringing circles together.
(d) Geometry after corner-removing mappings.
(e) Unit circle.

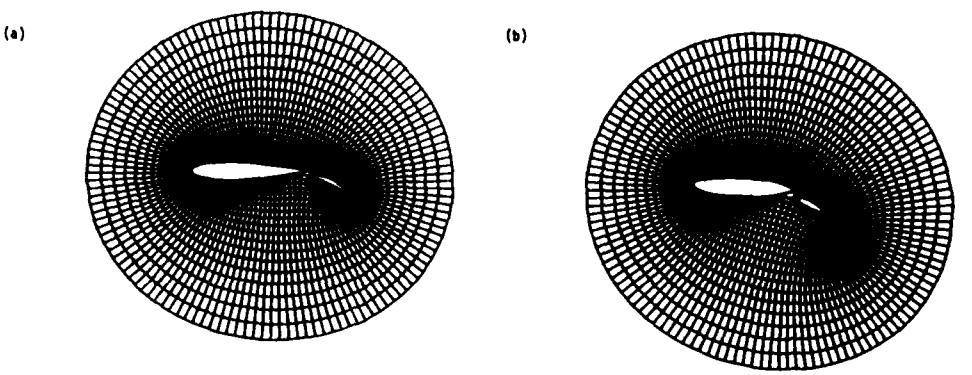


Fig. 4. Grids generated using the string mapping.
(a) Two-element airfoil.
(b) Three-element airfoil.

GRIDS USING STREAMLINE/POTENTIAL-LINE NETWORKS

The grids described above were derived using purely geometric conformal constructions. Other types of grids can be derived from networks of streamlines and potential-lines from auxiliary potential-flow solutions. These grids are conformal as a result of the fact that the complex potential ($\Phi = \phi + i\psi$, where ϕ and ψ are the scalar potential and stream function) and the complex velocity are analytic functions of each other. In fact, any conformal grid can be considered to be a potential/stream-function network (ϕ-ψ grid for short) for some potential flow. In this context, the grids described in the above section can be derived from a flow with a point vortex at the center of the circle in the transformed plane. The nonuniqueness of the potential-flow problem for given geometry ensures that a wide variety of types of conformal grid can be constructed.

The development of a ϕ-ψ grid generation capability for multielement airfoils (using the present author's conformal mapping procedures) requires a method for computing the flow around the multiple circles, with constant stream function on each circle. Such a method is described in reference 9 and briefly below.

Any incompressible potential flow solution can be represented by a linear combination of simpler fundamental solutions. In the present method, there are two noncirculatory fundamental solutions and a number of circulatory solutions equal to the number of circles. Each noncirculatory solution has unit freestream and zero circulation about each circle. The two solutions have different angles of attack of the freestream flow. Each circulatory solution has zero freestream, unit circulation about one circle, and zero circulation about all other circles. The two noncirculatory solutions and one of the circulatory solutions for a three-circle case are illustrated in figure 5.

The calculation of each noncirculatory flow solution involves finding an infinite sequence of reflected point doublet singularities within the circles. Each circulatory flow solution involves finding a similar infinite sequence of point vortex singularities. The result of each flow solution is a series expansion for the complex velocity as a function of complex coordinate. This is easily converted to a series for the complex potential having the following form:

$$(\phi+i\psi) = \sum_{NB=1}^{NBDS} \left\{ a_{0_{NB}} (\zeta-\zeta_{NB}) + a_{1_{NB}} \log(\zeta-\zeta_{NB}) + a_{2_{NB}} (\zeta-\zeta_{NB})^{-1} + a_{3_{NB}} (\zeta-\zeta_{NB})^{-2} + \ldots \right\} \tag{1}$$

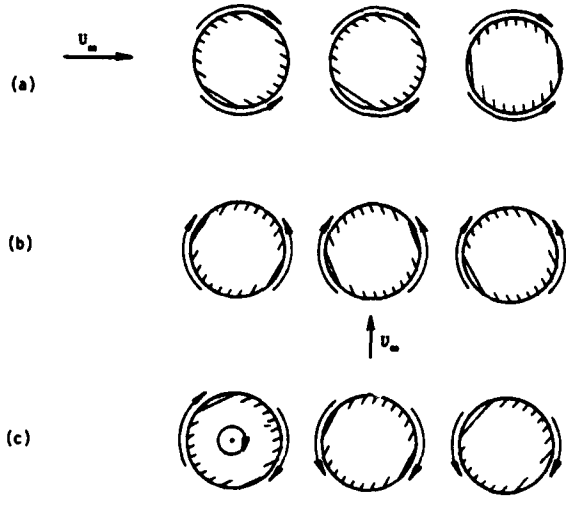


Fig. 5. Nonuniqueness of potential-flow solutions.
(a) Noncirculatory flow with freestream at $\alpha = 0°$.
(b) Noncirculatory flow with freestream at $\alpha = 90°$.
(c) Circulatory flow with stagnant freestream.

where ζ is the complex coordinate of the point at which the flow is to be computed, ζ_{NB} is the complex coordinate of the center of the circle having index NB, NBDS is the total number of circles, and the series coefficients (a_{ij}) are generally complex.

The calculation procedure for each point in a ϕ-ψ grid consists of solving equation (1) for the complex coordinate corresponding to the specified value of the complex potential in the multiple-circle plane, followed by a transformation to determine the complex coordinate in the physical plane. The solution of equation (1) is accomplished by a Newton iteration procedure for nonlinear complex equations. Having solved equation (1) on the boundaries of a region of the flow, the solution in the interior can often be obtained more efficiently using a fast Laplace solver.

ϕ-ψ grids for streaming flows

The most common flow solutions used for producing ϕ-ψ grids are probably the standard streaming flows, with uniform freestream and smooth flow off the trailing edge of each airfoil element. These can be obtained by combining all the fundamental flows described in the previous section. The combination constants for the noncirculatory fundamental flows depend only on the flow

angle of attack. The combination constants for the circulatory solutions are found by imposing the Kutta condition at the trailing edge of each airfoil element. In the multiple-circle plane, this requires specifying zero tangential velocity component at the images of the trailing-edge points and solving the resulting set of linear equations.

The point spacing in the physical plane of a ϕ-ψ grid is inversely proportional to the local flow speed. Consequently, a grid around a body which causes only a small perturbation to a uniform flow should have nearly uniform spacing. This is illustrated in figure 6(a), which shows a grid for a single-element airfoil at a small angle of attack. Flow solutions with extensive low-speed regions have extensive sparse areas. This is illustrated in figure 6(b), which shows a grid around two circles, with large sparse areas near the leading- and trailing-edge stagnation points.

These grids are divided into a number of segments, separated by the stagnation streamlines. Within each segment, increments of stream function and potential are constant, resulting in a rectangular grid in the ϕ-ψ plane. A logarithmic mapping transforms the rectangular region into an annular one similar to figure 1(a). The efficient flow calculation techniques of reference 3 can then be used to find the influence of each region at points in all the regions.

For the present application, these grids have several drawbacks. First, the sparse areas near the leading edges would give inadequate definition of the rapidly-varying field-source density. Second, the uniformly-spaced areas far from the bodies would reduce the solution efficiency by adding unnecessary points. Third, the flow calculation procedure of reference 3 is most efficient if O-type grids can be used.

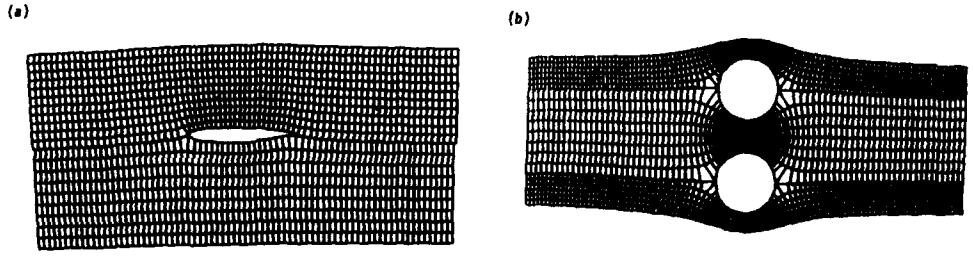


Fig. 6. Grids derived from potential-flow solutions for streaming flows.
(a) Single-element airfoil.
(b) Two circles.

ϕ-ψ grids for circulatory fundamental flows

O-type grids can be obtained from ϕ-ψ networks if flow solutions having circulation but no freestream are used. The point vortex solution for single-element cases, mentioned earlier, is an example of such an application. The simplest multielement flow solutions having circulation but no freestream are the circulatory fundamental flow solutions, having unit circulation about one body and zero circulation about all others. Like the ϕ-ψ grids for streaming flows, these grids are divided into a number of segments by the stagnation streamlines. In each segment, the increments in stream function and potential are constant and the flow calculation procedure is identical to that for a ϕ-ψ grid for a streaming flow.

Examples of circulatory ϕ-ψ grids are shown in figure 7 for two- and three-element airfoil cases. In general, the point distribution around the circulatory body is very desirable, with high point density near the leading and trailing edges and lower point density near mid-chord. The noncirculatory bodies have high point density near the leading and trailing edges, but they have far too low point density near the stagnation points on the upper and lower surfaces. Grids of this type would probably only be suitable for cases in which the expected flow solution is rapidly varying on just one of the airfoil elements.

ϕ-ψ grids for more general flows

The most undesirable features of the ϕ-ψ grids discussed above are the sparse areas associated with stagnation points on the bodies. In many cases,

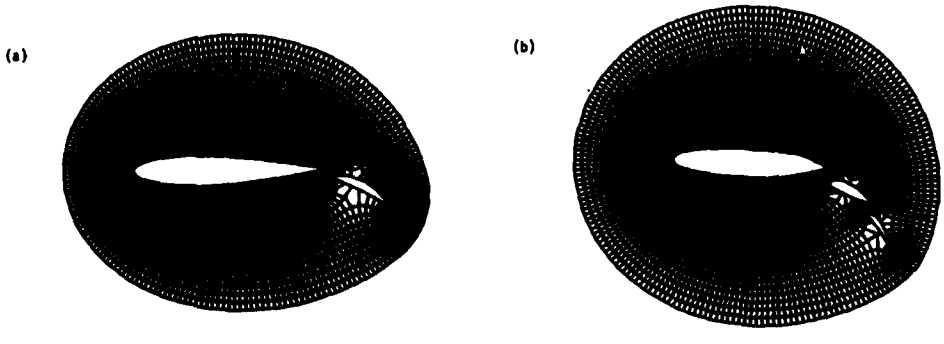


Fig. 7. Grids derived from potential-flow solution for circulatory fundamental flows.
(a) Two-element airfoil.
(b) Three-element airfoil.

however, it is possible to eliminate stagnation points entirely or move them so far from the bodies as to be inconsequential. One strategy for accomplishing this is to combine circulatory fundamental flows, alternate the sign of the circulation on adjacent bodies, and adjust the magnitudes to make the total circulation equal zero. Examples of portions of grids of this type are illustrated in figure 8. The most obvious feature of these grids is their extremely high point density in the areas between the bodies which, for a given total number of points causes sparse areas elsewhere. Another feature is that each grid is divided into a number of segments, within each of which the streamlines circulate around a single body. The dividing streamlines between the segments extend to infinity in both directions. Both of these features are undesirable for the present flow computation procedure, prompting the search for still further types of conformal grids.

One way of eliminating the infinite extent of the grid and also changing the point distribution is to allow the total circulation to be nonzero. At some distance from the bodies, the streamlines will then circulate around all the bodies and the grid can be truncated at any one of these streamlines. Spacing problems still remain, however, and can even become more serious as new stagnation points arise in the flow.

More control of the spacing can be obtained by introducing fictitious bodies or singularities into the flow (out of the range covered by the grid). If a new body surrounds all the other bodies and contains the entire flow in its interior, control is also achieved over the extent of the grid. This has

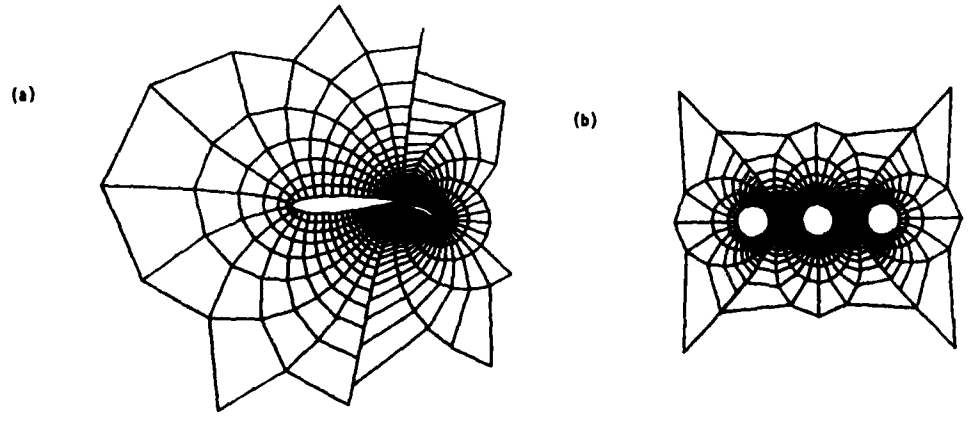


Fig. 8. Grids derived from potential-flow solutions for more general flows (zero total circulation).

been implemented by adding a larger circle around the original bodies in the multiple-circle plane. Calculation procedures for the auxiliary potential flow are only slightly modified by this addition, requiring the series for the complex potential to include positive as well as negative powers of the complex coordinate. An example of a portion of a grid generated in this manner is shown in figure 9. This is a big improvement over the previous grids; the point spacing is more appropriate and the grid extent is now finite.

Flow calculations using any of the grids discussed in this section encounter difficulties not found when any of the previous grids are used. The efficient flow calculation procedures of reference 3 can still be used to find the influence of the singularities within any given grid segment at points within that same segment, but they can no longer be used directly to find the influence at points outside the given segment. This is because the segment boundaries now represent folds in a Riemann surface, rather than just discontinuities in point spacing. Another way of expressing this is to note that the stream function is not a monotonic function of distance along any line crossing the dividing streamlines and, as a result, two or more points in different segments of a grid can have identical values of the complex potential.

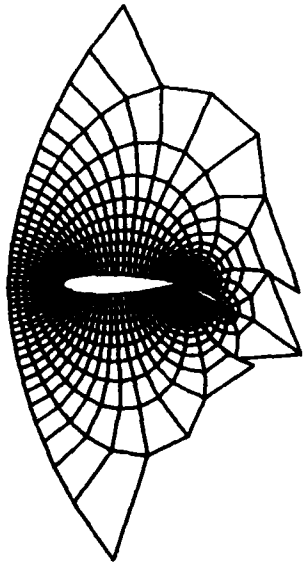


Fig. 9. Grid derived from a potential-flow solution for a more general flow (nonzero circulation and fictitious body).

These grids can still be very useful, especially in a field-panel method, but means of communication between the segments must be developed. These techniques could be very similar to the flow segmentation techniques described by Wu[13], et al. Given the influence of a segment on its boundaries, the influence at exterior points can be computed using either boundary singularities or a fast Laplace solver (perhaps with the aid of additional transformations). The details of these segment communication techniques have yet to be worked out.

SEGMENTED GRIDS WITH SPECIFIED BOUNDARIES

A greater degree of grid control can be achieved by directly specifying the shapes of the region boundaries, rather than using whatever shapes the dividing streamlines of a flow solution may form. In order to force the boundaries to be streamlines of the flow, it is necessary to distribute vortex singularities on the boundaries. The distribution of these singularities could be computed using a boundary-integral-equation technique similar to the panel methods for aerodynamic analysis developed by Hess[14] and others. A more efficient computational approach makes further use of conformal mapping. In this approach, each segment of the grid is dealt with independently of the other segments. The region between a single element of the multielement airfoil system and the boundary surrounding it is transformed to the annular region between two concentric circles. A polar grid in each annular segment is constructed and transformed back to the physical plane. The resulting grid is equivalent to the ϕ-ψ grid which would be computed using the vortex singularity approach.

The process of transforming a given region to an annulus is very similar to the method for transforming a multielement airfoil to a system of multiple circles. The first step is to apply a sequence of inverse Karman-Trefftz mappings, each of which removes a single corner from one of the boundaries. (If the boundary specification is performed in the multiple-circle plane, only the outer boundary will have any corners.) The next step is to apply an iterated sequence of mappings, each of which maps either the inner or the outer boundary to a perfect circle. In order to avoid the necessity of applying an interior mapping to the outer boundary and an exterior mapping to the inner boundary, an inversion mapping is performed after each circle mapping. At the end of a small number of iterations (typically three or four), both inner and outer boundaries are sufficiently close to circular and the derivative of the mapping function converges to within a small tolerance. Since these circles

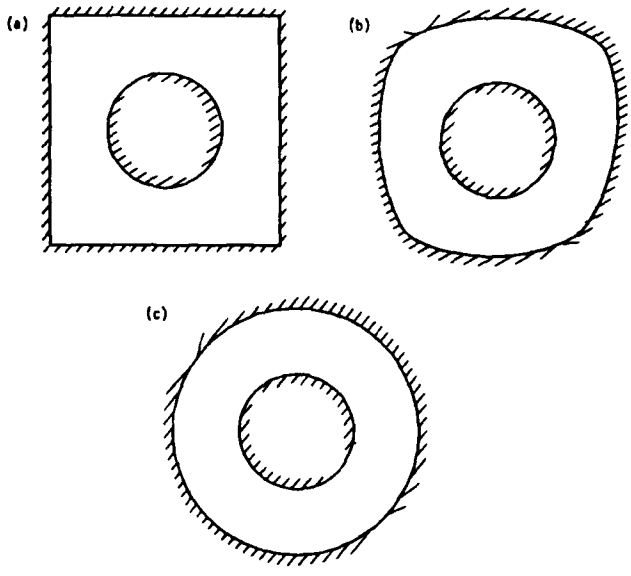


Fig. 10. Transformation to an annular region.
(a) Original geometry.
(b) Geometry after four corner-removing mappings.
(c) Geometry after two circle mappings and two inversions.

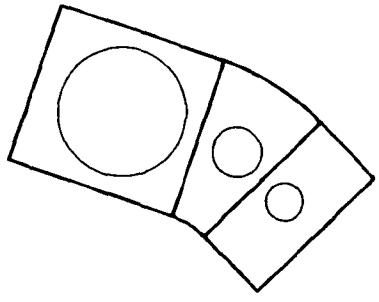


Fig. 11. Boundary construction for a segmented grid.

may not be concentric, it is necessary to perform a final linear fractional mapping. The steps of this transformation procedure are illustrated in figure 10 for one of the segments of the three-circle case shown in figure 11. An alternative approach to the annular mapping problem has been described by Ives[7]. Since his method is noniterative (except for the single-body mappings) it is perhaps more efficient. However, the present method can use simpler functions and the overall procedure is only slightly more expensive than computing two independent single-body mappings.

Grids produced by this technique for two- and three-element cases are shown in figure 12. In each case, a simple construction of straight lines and/or circular arcs in the multiple-circle plane was used to define the region boundaries. Point spacing around each airfoil element is similar to the spacing around the single-element airfoil of figure 1, with high point density at leading and trailing edges and no glaring sparse areas on the airfoil surfaces. Possible drawbacks of these grids include the presence of sparse areas near the corners of the outer boundaries and the necessity to locate boundaries too near the airfoil surfaces. (Moving the boundaries too far away produces sparse spacing on the airfoil surfaces.)

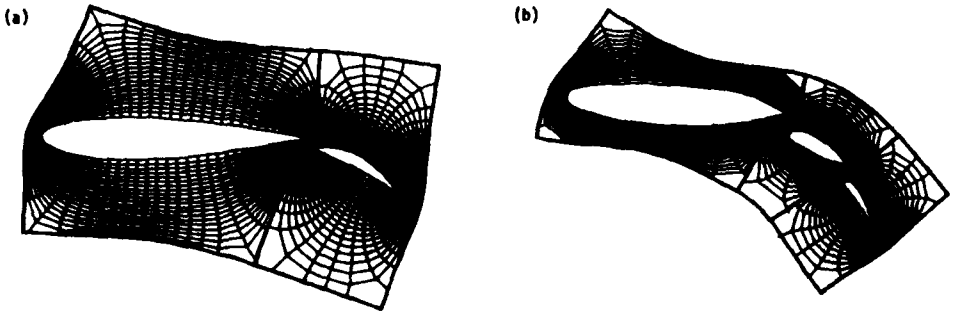


Fig. 12. Grids generated using the annular mapping.
(a) Two-element airfoil.
(b) Three-element airfoil.

HYBRID GRIDS

Improved grids can be obtained by combining the method described above with the string mapping illustrated earlier (figures 3 and 4). Instead of specifying the region boundaries arbitrarily, use can be made of curves generated using the string mapping. Region boundaries can be constructed using any of the curves surrounding all airfoil elements, together with sets of curves which run between the airfoil elements. In this way, two of the four corners (and their corresponding sparse areas) on the outer boundaries of the grid segments associated with the forward and aft airfoil elements are eliminated. It is also possible to extend the grid as far from the airfoil system as desired, by using a portion of the string grid directly in this region.

Grids produced by this technique for two-, three- and four-element cases are illustrated in figure 13. These grids retain the desirable features of the grids of figure 12, while eliminating most of their drawbacks.

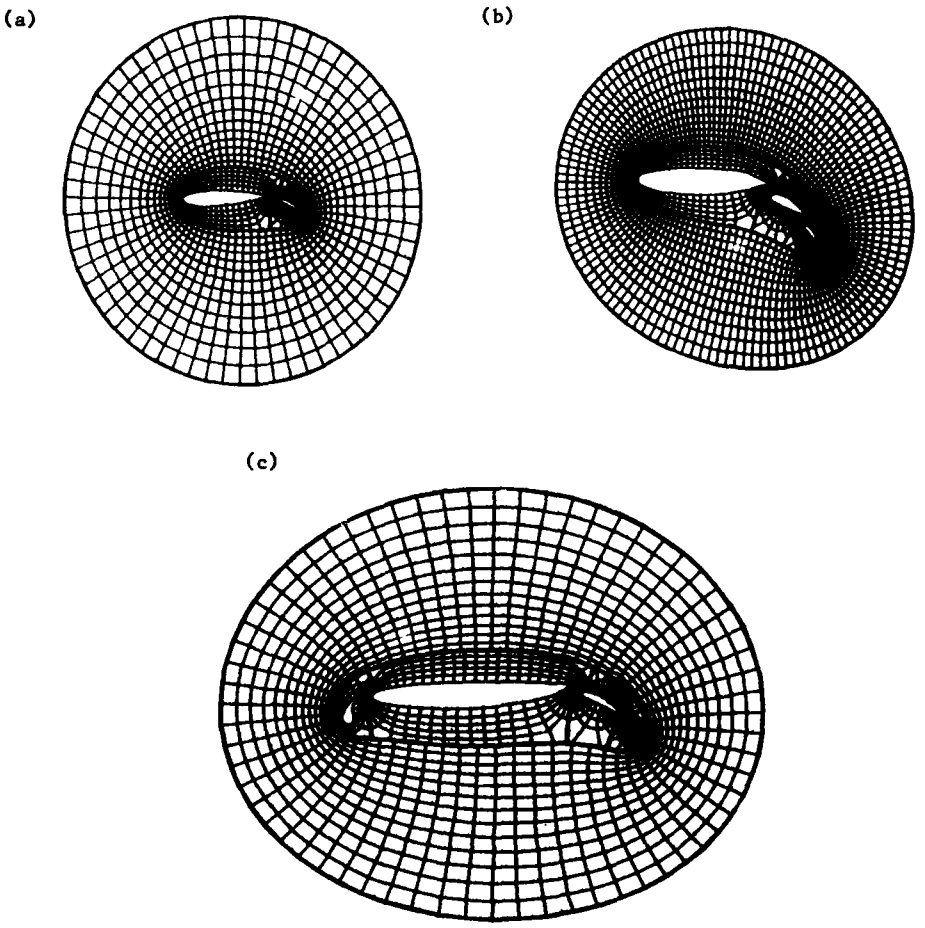


Fig. 13. Hybrid grids generated using both annular and string mappings.
(a) Two-element airfoil.
(b) Three-element airfoil.
(c) Four-element airfoil.

CONCLUSIONS

A chronicle has been given of the search for the type of conformal grid most suitable for use in computing the inviscid compressible flow around multielement airfoils using a distributed-source field-panel approach. Although many of the grids were deemed not suitable for this application, they were included in order to illustrate the great diversity of types of conformal grid which can be constructed. The final grids, for the most part, have

desirable point distributions around each airfoil element, of a form to which the efficient flow analysis techniques developed earlier can be readily applied. They are possibly close to the best which can be derived without sacrificing the conformality properties.

ACKNOWLEDGMENT

The author has benefitted considerably from numerous discussions of conformal mapping and grid-generation techniques with his colleague at Douglas Aircraft Company, Dr. R. W. Clark.

REFERENCES

1. Garabedian, P. and Korn, D. (1971) Comm. P. & Appl. Math, XXIV.
2. Jameson, A. (1971) Grumman Report 390-71-1.
3. Halsey, D. (1981) Proceed. of the Symposium on Numerical Boundary Condition Procedures, NASA CP-2201, pp. 61-71.
4. Wu, J.C. and Thompson, J.F. (1973) Computers and Fluids, 1, 2, pp. 197-215.
5. Luu, T.S. and Coulmy, G. (1977) Computers and Fluids, 5, 4, pp. 261-275.
6. Grossman, B. and Volpe, G. (1977) Office of Naval Research Report ONR-CR215-241-1.
7. Ives, D.C. (1975) AIAA Paper 75-842.
8. Thompson, J.F. (1978) Lecture Series in Computational Fluid Dynamics, Von Karman Inst. for Fluid Dynamics, Belgium.
9. Halsey, N.D. (1979) AIAA Paper No. 79-0271, also AIAA J., 17, 12.
10. Harrington, A. To appear in Journal d'Analyse Mathematique.
11. Halsey, N.D. (1980) AIAA Paper No. 80-0069.
12. Halsey, N.D. To appear in AIAA J.
13. Wu, J.C., Spring, A.H., and Sankar, N.L. (1975) Lecture Notes in Physics - Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics, 35, pp. 452-457.
14. Hess, J.L. (1975) Computer Methods in Applied Mechanics and Engineering, 5, pp. 145-196.

AD P000992

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

CONFORMAL MAPPINGS ONTO MULTIPLY CONNECTED REGIONS WITH SPECIFIED BOUNDARY SHAPES

A PRELIMINARY DISCUSSION OF COMPUTER IMPLEMENTATION

ANDREW HARRINGTON
School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332

INTRODUCTION

We describe a method of calculating conformal mappings of any given finitely connected region onto a region with arbitrarily specified boundary shapes. If the specified shapes are rectangles, then this method can be used to generate conformal grids which should be useful for numerical solution of many partial differential equations, for example in calculating the airflow past an airflow with flaps or the flow of cooling water past fuel pins in a nuclear reactor.

This author has proved that there exists a conformal mapping of any given finitely connected region onto a region with arbitrarily specified boundary shapes. The construction in the proof has been adapted for computer implementation. Some examples have been worked to determine the region bounded by circles which is the image of a given region in the extended complex plane under a conformal mapping taking ∞ to ∞. Mappings have also been calculated of the form indicated by figure 1 below. A region whose outer boundary is a

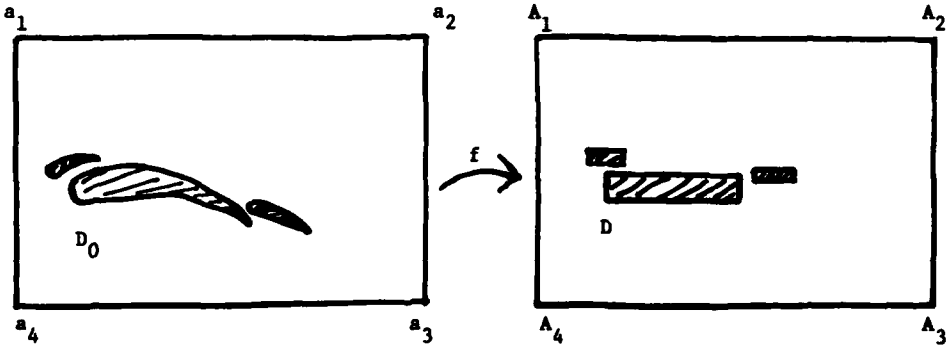


Fig. 1. Conformal mapping onto a region bounded by rectangles.
$f(a_j) = A_j$, $j = 1,2,3,4$.

rectangle is mapped conformally to a region with all rectangular boundaries, and the vertices of the outer boundaries correspond. Mappings onto regions bounded by rectangles should be of considerable use in grid generation for numerical solution of partial differential equations. Grids have been calculated for one simple example.

Conformal transformations are particularly useful when calculating fluid flow from the Navier-Stokes equations. In many formulations of these equations there is a system of linked partial differential equations in which the highest order part of each differential operator is the Laplacian. If a grid is generated based on a nonconformal transformation to a computational domain, then the transformed equations become more complicated. The Laplacian, however, is invariant under conformal transformation, so the second order part of the equations remains simple, saving considerable calculation. Thus conformal transformations have long been used for fluid flow problems in simply or doubly connected regions, where a variety of known conformal mapping techniques are applicable. Multiply connected regions also arise in flow problems. Examples are airflow past an airfoil with flaps and flow of cooling water past fuel pins in a nuclear reactor. Conformal grids have not been used generally for such problems because of a lack of appropriate mapping techniques. The method we describe is applicable to these problems.

The paper is organized in the following manner: We discuss the theoretical formulation for the mappings between regions containing ∞ in §1, and then our present numerical formulations and some possible improvements in §2. In §3 we discuss the differences in these formulations when we consider mappings between regions with outer rectangular boundaries. In §4 we discuss further possible improvements, and end in §5 with data about some specific examples worked out on the computer.

§1. THEORETICAL FORMULATION

The method involves matching potentials. Suppose D is an n-tuply connected region containing ∞ in the extended complex plane. We shall call a multiple valved function ψ a (complex) potential on D if

1) ψ is analytic in D and continuous in the closure of D, except $\psi(z) = -\log(z) + O(1/z)$ near ∞.

2) $\mathrm{Re}(\psi)$ is constant on each component of the boundary of D. We call these constants the boundary potentials.

Associated with each such potential is a charge distribution $\rho(\phi)$ on the boundary B of D such that

$$\psi(z) \quad - \int_B \rho(\phi)\, \log(\phi - z)\, |d\phi| .$$

Suppose ψ_0 is a potential on the region D_0. Let $b_1^*, b_2^*, \cdots b_n^*$ be the boundary potentials and let $q_1, q_2, \cdots q_n$ be the total charges on the components of the boundary.

Now suppose we are given n simple closed curves $\Gamma_1, \Gamma_2, \cdots \Gamma_n$. Let $M = (M_1, M_2, \cdots M_n) \in \mathbb{C}^n$ and $R = (R_1, R_2, \cdots R_n) \in \mathbb{R}^n$, with each $R_j > 0$. Let $B_j(M,R) = \{R_j z + M_j : z \in \Gamma_j\}$, a shape similar to Γ_j. For an open set of valves in $\mathbb{C}^n \times \mathbb{R}^n$ the sets $B_j(M,R)$, $j = 1,2,\cdots n$, are the boundary components of an n-tuply connected domain containing ∞, $D(M,R)$. Let $\psi(z) = \psi(z;M,R)$ be the complex potential for the domain $D(M,R)$ with total charges $q_1, q_2, \cdots q_n$ on the boundaries.

For some (M,R) with $M_n = 0$ and $R_n = 1$ suppose there is a conformal mapping f of D_0 onto $D(M,R)$ with $f(z) = cz + O(1)$ near ∞, $c > 0$. Then

$$(2) \quad \psi_0(z) = \psi(f(z)) + \log c.$$

Suppose $\psi_0'(z) = 0$ for $z = z_j \in D_0$, $j = 1,2,\cdots n-1$. The points z_j are the critical points of the potential ψ_0. Now $f'(z_j) \neq 0$, and $\psi_0'(z_j) = \psi(f(z_j))f'(z_j)$ so ψ must have critical points at $f(z_j)$, $j = 1,2,\cdots n-1$. The maximum number of critical points that any complex potential can have off of its boundaries is n-1, so the points $w_j = f(z_j)$, $j = 1,2,\cdots n-1$, are all the critical points of ψ in $D(M,R)$.

Let us collect the conditions we have on $\psi(z;M,R)$. Let b_j be the j^{th} boundary potential for $\psi(\cdot;M,R)$. Then from (2) $b_j^* = b_j + \log c$ for $j = 1,2,\cdots n$. We take $b_n^* = b_n + \log c$ to define c. Then we have, for some ordering of the critical points w_j of ψ:

$$(3) \quad \psi(w_j) = \psi_0(z_j) - \log c$$

$$b_j = b_j^* - \log c \qquad j = 1,2,\cdots n-1$$

This 3(n-1) real equations. We have 3(n-1) real parameters free in (M,R) since we have fixed M_n and R_n. This author has proved with a homotopy argument that, even if we do not start off knowing a conformal mapping of D_0 onto some $D(M,R)$, there is an (M,R) solving these equations such that

$$(4) \quad \psi(w) = \psi_0(z) - \log c$$

implicitly defines a conformal mapping $w = f(z)$.

§2. NUMERICAL FORMULATION

Our initial objective was to rapidly write programs that demonstrate the feasibility of implementing the theory just described. Since accomplishing that, numerous improvements and extensions suggested themselves. In a subsequent paper we shall report on the implementation of the latter ideas. In this paper we will describe the ideas behind our first approach, aome of the further ideas generated, and some of the numerical results of our initial approach.

Given a domain D_0 and shapes Γ_j we can break the problem into six steps: (i) choose charges q_j, (ii) approximate ψ_0, (iii) calculate critical points z_j, potentials $\psi_0(z_j)$ and boundary potentials b_j^*, (iv) devise a subroutine to calculate the corresponding quantities for D(M,R), (v) use this subroutine with a nonlinear equation solver to find a domain, D(M,R), conformally equivalent to D_0, and (vi) solve $\psi(w) = \psi_0(z) - \log c$ to find the conformal mapping and/or its inverse.

So far we have bypassed step (ii) by first choosing ψ_0 as a discrete sum $\psi_0(z) = -\sum a_j \log(z-z_j)$ and choosing a_j's and b_j's so ψ_0 is the potential for an n-tuply connected domain. For example with $\psi_0(z) = -\frac{2}{3}\log(z+2) - \frac{1}{3}\log(z-1)$ and boundary potentials $-\frac{1}{3}\log 2$ and $-\frac{1}{3}\log 25/8$ on the left and right boundary components we get a domain D_0 illustrated in Figure 2. For a more general domain D_0, ψ_0 may be calculated in the same manner that we calculate $\psi(z;M,R)$, described next.

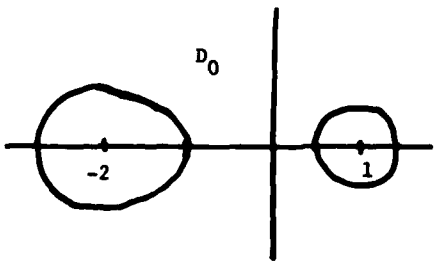


Figure 2. Domain with potential $-\frac{2}{3}\log(z+2) - \frac{1}{3}\log(z-1)$

We approximate ψ by a variant of Symm's method. We approximate a boundary charge distribution $\rho(\phi)$ and then approximate the integral $\psi(z) = -\int_B \rho(\phi)\ \log(\phi - z)|d\phi|$. For simplicity we took all boundaries to have the same shape Γ and use related N dimensional approximation spaces to approximate on each boundary component $B_1, B_2, \cdots B_n$. Suppose Γ has arc length parameterization $\gamma(\sigma)$ and γ has period N, the length of Γ. Let $\gamma_m = \gamma(m)$ for each integer m. First we construct an approximation space of functions on Γ with basis $\{e_m\}_{m=1}^N$ where each basis element has support in a neighborhood of γ_m and has integral 1. Initially we take N=4. Later we shall increase it. Next define the corresponding approximation space of functions with basis $\{e_{mj}\}_{m=1,j=1}^{N\quad n}$, where $e_{mj}(R_j(\gamma(\sigma)) + M_j) = \frac{1}{R_j}\, e_m(\gamma(\sigma))$, σ real, and $e_{mj}(\phi) = 0$ for $\phi \notin B_k$. We take the function approximating ρ to be $\tilde{\rho}(\phi) = \sum_{m=1}^{N} \sum_{j=1}^{n} a_{mj}\, e_{mj}(\phi)$, where the constants a_{mj} will be chosen shortly to satisfy a linear system of equations. Let $\tilde{\psi}(z) = -\int_B \tilde{\rho}(\phi)\ \log(\phi - z)|d\phi|$ be the corresponding approximation to the potential.

Let $\phi_{mj} = R_j\gamma_m + M_j$, $m = 1,2,\cdots N$, $j = 1,2,\cdots n$, be the points on the boundaries B_j corresponding to the points γ_m on Γ. The conditions determining the approximate charge density $\tilde{\rho}$ are that there are constants $\tilde{b}_j$, $j = 1,2,\cdots n$ such that

$$(5) \qquad \sum_{m=1}^{N} a_{mj} = q_j \qquad j = 1,2,\cdots n$$

$$\operatorname{Re}\, \tilde{\psi}(\phi_{mj}) = \tilde{b}_j \qquad m = 1,2,\cdots N,\ j = 1,2,\cdots n.$$

The first equation says that the total charge on B_j is q_j, as with ρ. We cannot ensure that $\operatorname{Re}\, \tilde{\psi}$ is constant on all of B_j, but only that it has the same valve at N points ϕ_{mj}, $m = 1,2,\cdots N$. This gives $(N+1)n$ equations in the $(N+1)n$ unknowns $q_{mj}, \tilde{b}_j$. The equations and variables can easily be reduced in number to $(N-1)n$ by replacing the second set of equations by $\operatorname{Re}(\tilde{\psi}(\phi_{mj}) - \tilde{\psi}(\phi_{Nj})) = 0$, $m = 1,2,\cdots N-1$, $j = 1,2,\cdots n$ and eliminating the variables a_{Nj} by solving the first equations for them and substituting into the second set of equations. The linear system becomes

$$(6)\quad \sum_{m=1}^{N-1}\sum_{j=1}^{n}\left[(C_{mjst}-C_{mjNt})-(C_{Njst}-C_{NjNt})\right]a_{mj} = \sum_{j=1}^{N}(C_{Njst}-C_{NjNt})q_j \qquad \begin{matrix} s = 1,2,\cdots N-1, \\ t = 1,2,\cdots n \end{matrix}$$

where

$$(7)\quad C_{mjst} = -\int_{B_j} e_{mj}(\phi)\log|\phi_{st}-\phi||d\phi|,$$

the real part of the potential at ϕ_{st} due to the charge density e_{mj}. In our calculations of the numbers C_{mjst} we consider two cases, where the point and the charge density lie 1) on the same boundary component and 2) on different components.

Case 1). Here $j = t$. We may map back to Γ and use the fact that e_m has integral 1 to calculate $C_{mjsj} = -\int_{\Gamma} e_m(\phi)\log|\gamma_s-\gamma||d\gamma| - \log R_j$. Thus the difference $C_{mjsj}-C_{mjNj}$ is independent of M,R, and j. We put the unknowns a_{mj} in an (N-1)n dimensional vector whose $(m+(N-1)j-1))^{th}$ entry is a_{mj} and similarly order the equations. The matrix of the linear system has (N-1) × (N-1) blocks on the main diagonal which are identical and are independent of M and R. We can evaluate this (N-1) × (N-1) block once accurately without worrying much about cost.

Case 2). Now $j \neq t$. In this case C_{mjst} is strongly dependent on M and R, so it must be repeatedly calculated, so we would like to do it cheaply. We are helped by the fact that the charge density e_{mj} has support just on a neighborhood of ϕ_{mj} on B_j, and ϕ_{st} lies on a different boundary component, so the interval of integration is short and the derivatives of $\log(\phi-\phi_{st})$ will be much smaller in general than when ϕ_{mj} and ϕ_{st} lie on the same boundary component. We should be able to use a relatively low order approximation. In practice thus far we have used the lowest order approximation, $C_{mjst} \simeq -\log|\phi_{mj}-\phi_{st}|$. We will have a discussion of errors and possible improvements later in §3.

The linear system is solved directly with a canned subroutine to calculate the a_{mj}'s. The boundary potentials are determined using the calculated values of the a_{mj}'s and C_{mjst}'s. We also need to find the critical points and potentials at the critical points. The critical points all lie away from the

boundaries so we use a low order approximation of $\psi(z)$: $\psi(z) \simeq -\sum \sum a_{mj} \log(\phi_{mj}-z)$ and $\psi'(z) \simeq \sum\sum a_{mj}/(\phi_{mj}-z)$. We use Newton's method to determine the critical points. Thus for a given domain D(M,R) we could approximate the boundary potentials and potentials at critical points. We used a canned nonlinear equation solver to find the right valve of (M,R) so the potentials match those associated with D_0. The nonlinear equation solver efficiently employed a secant method so that after some initial calculations only one time-consuming function evaluation was needed at each step. If the initial guess was bad, however, the secant solver had a hard time, so a nonlinear equation solver like the one used by Trefethen[2] to find Schwarz-Christoffel parameters would probably be better. He used a solver which started with the method of steepest descent before the secant method became effective.

The closeness of the valve of (M_*, R_*) calculated by the nonlinear equation solver to some (M*,R*) such that D_0 is conformally equivalent to D(M*,R*) depends on the accuracy of our approximation of ψ and improves as N increases. The cost of calculating the approximation to ψ also increases dramatically as N increases. We have first used a value of 4 for N to cheaply get a fair approximation to (M*,R*) and used this approximation to (M*,R*) as an initial guess when calculating with a doubled valve of N(and halved mesh size on B). This doubling procedure can be repeated to further improve the accuracy of $\hat{\psi}$. As we halve the mesh size on B the quadrature formulas we have used also improve in accuracy.

§3. REGIONS WHOSE OUTER BOUNDARIES ARE RECTANGLES

The final step of calculating the conformal mapping has been carried out so far only for a problem with a slightly different formulation. A given domain D_0 bounded by an outer rectangle and n inner boundaries is to be mapped onto a region D bounded by an outer rectangle and n squares on the inside. We require that the vertices of the outer rectangles correspond. The potentials, parameters and conditions in this situation are slightly different from before.

We can avoid calculating a charge density on the outer boundary by expressing our potentials in terms of the elliptic functions sn(z;k) which are meromorphic and doubly periodic in z for each valve of the parameter k, $0 < k < 1$, and are defined by

$$z = \int_0^{sn(z;k)} \frac{dt}{\sqrt{1-k^2t^2}\,\sqrt{1-t^2}}$$

We denote the smallest real and imaginary periods of $sn(z;k)$ by $4K(k)$ and $2iK'(k)$. For simplicity we consider only domains D whose outer rectangle has vertices 0, K, K'i, and K+K'i for some k, $0 < k < 1$. This means we allow outer rectangles of all proportions, since K'/K goes through all positive real values for $0 < k < 1$.

We take our complex potentials ψ in this situation to be multiple valved analytic functions on D, continuous on the closure of D, with constant real part on each boundary compontent, and with $\psi(iK') = 0$. With this definition ψ may be represented in terms of a charge distribution on just the inner boundary components,

$$(8)\quad \psi(z) = -\int \rho(\phi)\, \log(\Delta(z,\phi,k))\,|d\phi|$$

and

$$(9)\quad \Delta(z,\phi,k) = \frac{sn^2(z;k)-sn^2(\phi;k)}{sn^2(z;k)-sn^2(\phi;k)}$$

We could have used an integral representation for $\psi(z)$ involving a charge distribution on all the boundaries, but the elliptic functions are easy to calculate and avoiding approximating a charge distribution on the outer boundary considerably reduces the size of the linear system to be solved to approximate the charge density.

The potential was formulated by considering a doubly periodic domain related to D, as illustrated in figure 3. The figure shows a period module

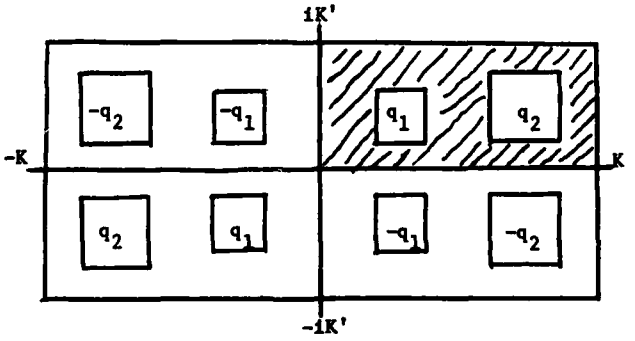


Fig. 3. Period module for the doubly periodic region derived from the shaded region D.

for the periodic domain derived from the shaded region D. There is also a doubly periodic distribution of charges to generate the potential. The

boundary components are labeled with total charges showing how opposite charges are placed in the reflections of D across the real and imaginary axes. The charge distribution with periods 2K and 2K' determines a potential with the same periods, which are the periods of $sn^2(z;k)$. The symmetries of the charge distribution ensure that $Re\psi(z)$ is constant on the outer rectangular boundary of D. The pole of sn(z;k) at iK' ensures that $\psi(iK') = 0$, (using the principal branch of the logarithm).

The possible image domains are now denoted D(M,R,k) where k is the elliptic parameter determining the outer boundary and M and R give the positions and magnifications of the n inner boundary components in the same way we did before. Thus we have 3n+1 real parameters. We will also have 3n+1 real conditions relating potentials on D_0 and some conformally equivalent D(M,R,k). We choose positive total charges q_k for the inner boundary components of D_0 so the corresponding potential ψ_0 on D_0 has n-1 distinct critical points in D_0. We set $\psi(w) = \psi(w;M,R,k)$ to be the potential on D(M,R,k) with the same total boundary charges. All the potentials are normalized to be 0 in the upper left hand corner of their domains, and thus the outer boundary potential is 0. In order for the equation $\psi_0(z) = \psi(w)$ to implicitly determine a conformal mapping of D_0 onto D(M,R,k) with outer vertices corresponding, we must have that the imaginary part of the potentials at the other three outer vertices, the real part of the potential on the n inner boundaries, and the complex potentials at the n-1 critical points all correspond for the two potentials. This gives us the 3n+1 real conditions for a nonlinear equation solver.

The numerical formulation of this problem parallels that of the first problem. At one point we need to do a little extra work. Analogous to the quantity C_{mjst} we need to calculate

$$(10)\qquad D_{mjst} = -\int_{B_j} e_{mj}(\phi)\,\log|\Delta(\phi,\phi_{st},k)|\,|d\phi|$$

Again we consider the two cases j=t and j≠t.

1) j=t. We rewrite

$$D_{mjsj} = -\int_{B_j} e_{mj}(\phi)\log\left|\frac{\Delta(\phi,\phi_{sj},j)}{\phi-\phi_{sj}}\right|\,|d\phi| - C_{mjsj}.$$

Generally the only singularity of $\log|\Delta(\phi,\phi_{sj},k)|$ for ϕ near the support of e_{mj} is at ϕ_{sj}. We have cancelled out this singularity in the integrand and introduced the quantity C_{mjsj} from (7), which as we have said in §2, need only be calculated once. The removal of the nearby singularity in the inte-

grand allows us to make a simpler approximation of the integral. Again we have used the simplest approximation,

$$D_{mjsj} \tilde{=} -\log \left| \frac{\Delta(\phi_{mj},\phi_{sj},k)}{\phi_{mj}-\phi_{sj}} \right| - C_{mjsj}$$

for $m \neq s$, and the quotient is replaced by the analytically calculated limit of $\Delta(\phi,\phi_{sj},k)/(\phi-\phi_{sj})$ as $\phi \to \phi_{sj}$ if $m=s$.

2) $j \neq t$. Generally in this case all of the singularities of $\log \Delta(\phi,\phi_{st},k)$ are far from the small interval of support of e_{jk}, so a fairly simple quadrature formula can be used. Again we used the simplest approximation, $D_{mjst} \tilde{=} -\log|\Delta(\phi_{mj},\phi_{st},k)|$. This was probably a place we introduced a large part of our error. In future versions we should analyze the errors more completely and, at least when we have located M,R, and k quite closely, we should improve the approximation in our quadrature.

Another possible major source of error is our approximation space for the potential. We shall now discuss some possible spaces and their relation to possible quadrature formulas. We start with only a few widely spaced nodes ϕ_{mj} on each boundary component so approximations with a high degree of smoothness do not seem appropriate. Away from any corners on the boundaries, piecewise linear functions of arc length seem sufficient to approximate the charge density. Thus the basis element e_m will have support from γ_{m-1} to γ_{m+1} and

$$e_m(\gamma(\sigma)) = 1 - |m-\sigma|, \quad m-1 < \sigma < m+1.$$

When the number of nodes increases it may be appropriate to switch to a higher degree spline approximation, say with cubic splines. In the examples we worked out we used a compromise, piecewise quadratic function of arc length.

So far we have used the same approximation at the corners of the squares, also. In fact the charge densities will be singular there. Using appropriate singular basis elements in the approximations should give a considerable improvement. For instance suppose we translate a corner on an inner square to the origin. We can remove the corner by the conformal transformation $w = z^{2/3}$. If the corresponding charge densities in the z-plane and w-plane are $\rho(z)$ and $\rho^*(w)$, then $\rho(z) = \rho^*(w)\ |\frac{dw}{dz}| = \rho^*(z^{2/3}) \cdot 2/3|z^{-1/3}|$.

The new density ρ^* will not have a singularity at the origin. We can approximate ρ^* with the same kinds of functions discussed above for smooth boundaries. For instance, if we are using piecewise linear functions and

γ_0 is a corner, then $e_0(\gamma(\sigma)) = \frac{2}{3}(1-\sigma^{2/3})|\sigma^{-1/3}|$, $-1 \le \sigma \le 1$. In this case the integrals C_{mjsj} can be evaluated exactly. We still need quadrature formulas for the integrals D_{mjst}, however. Since all the integrals involve weight functions e_{mj} of known form, quadrature formulas of Gauss type suggest themselves. The proper weights need only be calculated once.

Once the proper values of M,R, and k are determined so that D_0 in the z-plane and $D = D(M,R,k)$ in the w-plane are conformally equivalent, the last step is to calculate the conformal mapping. We can easily generate a grid of points w_{ij} in the domain D bounded by rectangles. We would like to calculate the corresponding points z_{ij} in D_0 which satisfy $\psi_0(z_{ij}) = \psi(w_{ij})$. So far we have only worked out one simple example with one inner boundary where the potential ψ_0 had the form $\psi_0(z) = \log(\Delta(z,(K+iK')/2,k))$. We simply solved the implicit equation for z_{ij} by Newton's method. If ψ_0 were more complicated this would be very time consuming.

Another common method of calculating grids in related situations takes advantage of the speed of elliptic equation solvers for rectangular regions. First we divide the grid points w_{ij} in D into rectangular blocks. We can calculate the few grid points z_{ij} corresponding to points w_{ij} on the boundaries of the rectangular blocks by solving the implicit equation. The remaining points z_{ij} corresponding to points w_{ij} in the interior of any block can be calculated with a fast elliptic solver since Re(z) and Im(z) are harmonic functions of w. If we want more accuracy than is provided by the elliptic solver, then its answers will make excellent starting values for finally solving $\psi_0(z_{ij}) = \psi(w_{ij})$ by Newton's method.

Another advantage of the equation $\psi_0(z) = \psi(f(z))$ determining the conformal mapping $f:D_0 \to D$ is that we obtain an analytic expression for the derivatives of the mapping,

$$f'(z_{ij}) = \psi_0'(z_{ij})/\psi'(w_{ij}).$$

This derivative is needed for the solution of most PDE's using the grid. Nonconformal grids are routinely calculated as the solution of a system of finite difference equations. The grid points in the physical domain D_0 correspond to the grid points in a computational domain D under some nonconformal mapping that is not completely specified. The partial derivatives of this mapping function at the grid points must be approximated by finite differences. This introduces an error which we can avoid because of our analytic expression.

§4. FURTHER POSSIBLE MODIFICATIONS

We shall describe two further possible modifications of our procedure. The first will be to change the net boundary charges. Initially we chose all the charges to be positive. They were chosen, as they always can be for a given domain, so that the critical points all lay away from the boundaries. For the theoretical paper[1], this was convenient. In practice, however, we must then locate each critical point with some effort. A more practical choice is to have a charge -1 on one inner boundary, and a zero net charge on each other inner boundary. Suppose ψ_0 and ψ are potentials with these net charges for D_0 and D. Such potentials will have no critical points in the interiors of their domains. Let $F_0(z) = \exp(\psi_0(z))$, $F(w) = \exp(\psi(w))$. Then F_0 and F map D_0 and D conformally onto regions bounded by circles and circular arcs with center zero. See Nehari[3]. The boundaries with nonzero net charge map to whole circles. The boundaries with zero net charge map to circular arcs. See Figure 4. We want to choose our parameters so $F_0(D_0)$ and $F(D)$ are the same region. Then $f(z) = F^{-1} \circ F_0(z)$ is a conformal mapping of D_0 onto D.

In place of worrying about potentials at critical points we must make sure that the endpoints of the circular arcs in $F_0(D_0)$ and $F(D)$ coincide. These endpoints may be easily located. See Figure 5 showing a neighborhood of one boundary in D with zero net charge. The boundary will always be split into two sections so the charge density is positive on one section, negative on the other, and zero only at the two points in between, labeled a_1 and a_2. The endpoints of the image of the boundary under F are $F(a_1)$ and $F(a_2)$. There are only three real parameters associated with $F(a_1)$ and $F(a_2)$ since $|F(a_1)| = |F(a_2)| = \exp(\text{the boundary potential})$. The other two parameters are the arguments of $F(a_1)$ and $F(a_2)$, that is $\mathrm{Im}(\psi(a_1))$ and $\mathrm{Im}(\psi(a_2))$. The points a_1 and a_2, where the boundary charge density changes sign, may be easily calculated. We avoid finding critical points of ψ in the interior of D.

The last modification we shall discuss may be useful when the dimension of the linear system solved to calculate the approximate charge density on the boundary of $D(M,R,k)$ is large. Repeated direct solution of the system with different parameter values would be very time consuming. The linear system has special features making it seem well suited to iterative solution. Suppose, as we discussed in §2, $D(M,R,k)$ has n inner boundaries of similar shape with corresponding N dimensional approximation spaces for the charge density. Let $Lx = b$ be the $n(N-1)$ dimensional linear system to solve. First we will restructure the linear system. Let $A^{(j)}$ be the $(N-1) \times (N-1)$ matrix

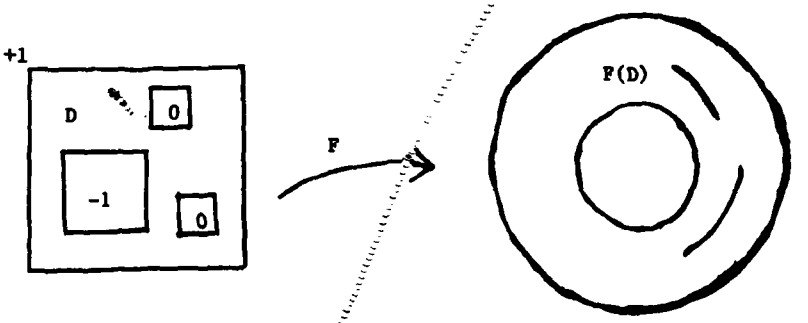


Fig. 4. Alternate boundary charges are shown for a potential associated with the intermediate conformal mapping F onto a slit ring region.

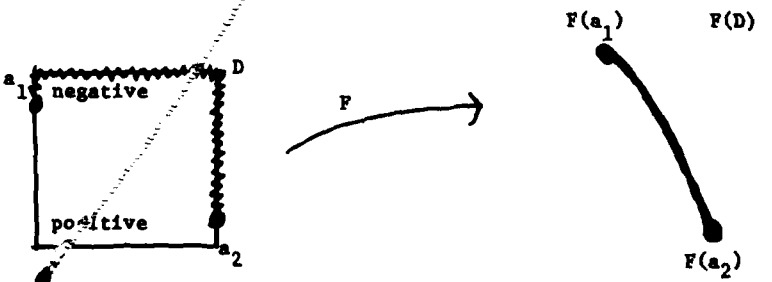


Fig. 5. Charge distribution on one boundary component in Figure 4 and the intermediate conformal mapping F in the vicinity.

with entries

$$(11) \qquad A_{ms}^{(j)} = (C_{mjsj}-C_{mjNj}) - (C_{Njsj}-C_{NjNj}).$$

The elements of $A_{ms}^{(j)}$ appeared in (6). Let A be the matrix with submatrices $A^{(j)}$, $j = 1,2,\cdots n$ on the main diagonal and zeros elsewhere. Let $B = L-A$. We may rewrite $Lx=b$ as

$$(12) \qquad (I+A^{-1}B)x = A^{-1}b$$

The matrices A and B measure the effect on potential differences along boundaries coming from varying the charge distribution on the same boundary and on different boundaries respectively. Particularly if the boundaries are well separated, the norm of $A^{-1}B$ should be small.

Because we are using the same sort of approximation space on each boundary,

independent of M,R, and k, the matrices $A^{(j)}$ are identical and independent of M,R, and k. Thus A^{-1} need only be calculated once, and the only work is in inverting one submatrix $A^{(j)}$. The considerations make it seem that (12) can can be solved iteratively quite easily.

We will need to solve (12) repeatedly as we look for the right values of M,R, and k. The values should vary little from one step to the next as we home in on the right values, so the linear system should vary little. If $\tilde{x}$ is the solution at the previous step, we need only solve for $y = x-\tilde{x}$ in

(13) $\quad (I+A^{-1}B)y = c \quad$ where $\quad c = b-(I+A^{-1}B)x.$

When the magnitude of y is small relative to x, we can solve (13) in fewer iterations than we can in (12) with comparable absolute errors.

§5. SOME EXPERIMENTAL RESULTS

Two main steps in the problems we have discussed are 1) picking values of (M,R) or (M,R,k) so ψ_0 and ψ match in the right places and 2) calculating grids. Tables 1 and 2 describe searches for (M,R). Table 3 describes a search for (M,R,k). Plot 1 and Plot 2 are corresponding grids.

We looked for domains bounded by three circles with specified boundary potentials, b_1, b_2, and b_3, and with specified complex potential at its critical points, z_1 and z_2. We assume the total charge on each boundary is one. We tried to guess the right values for the centers, M_1 and M_2, and radii, R_1 and R_2, of the first two circles if $M_3 = 0$ and $R_3 = 1$. Tables 1 and 2 show the number of subdivisions N of each boundary used to approximate ψ, the number of iterations taken, and the values of M and R calculated at the end of the iterations leaving us with a maximum error in the potentials as specified. In successive columns with higher N we use the previous values of M and R as initial guesses.

In both tables 1 and 2 the parameters are for a domain symmetric across the real axis. Our potential approximations did not preserve this symmetry automatically. We can see how the number of digits agreeing in M_1 and $\overline{M_2}$ and in R_1 and R_2 correspond to the maximum errors in the potentials.

TABLE 1

SOLVING FOR (M,R) SO $b_1 = b_2 = .111$, $b_3 = 0$, $\psi(z) = -1.3$, $\psi(z) = -1.39$, WITH INITIAL GUESS $M_1 = 20+10i$, $M_2 = 20-10i$, $R_1 = R_2 = 1$

N	4	8	10
iterations	7	4	5
M_1	19.9911938+9.83435871	20.0103733+9.83483381	20.0108152+9.83507531
M_2	19.9911935-9.83435811	20.0103735-9.83483411	20.0108152-9.83507531
R_1	1.0136412	1.0137188	1.013715623
R_2	1.0136429	1.0137185	1.013715621
maximum error	2.7×10^{-6}	8.5×10^{-7}	5×10^{-9}

TABLE 2

SOLVING FOR (M,R) SO $b_1 = b_2 = -1.03$, $b_3 = 0$, $\psi(z_1) = -1.4$, $\psi(z_2) = -1.41$, WITH INITIAL GUESS $M_1 = 19.5 + 10i$, $M_2 = 19.5 - 10i$, $R_1 = R_2 = 3$

N	4	10
iterations	3	4
M_1	18.2536288+9.73733041	18.424888+9.7952151
M_2	18.2585680-9.74638261	18.424872-9.7952301
R_1	3.053838	3.059700
R_2	3.052190	3.059702
Maximum error	10^{-4}	10^{-6}

Table 3 has a similar form, except now we are looking for (M,R,k) parameterizing a domain whose outer boundary is a rectangle and whose two inner boundaries are squares. Here we are trying to match up with specified values Im(ψ) at three vertices of the outside rectangle, Re(ψ) on the two inner boundaries, and ψ at the critical point.

TABLE 3

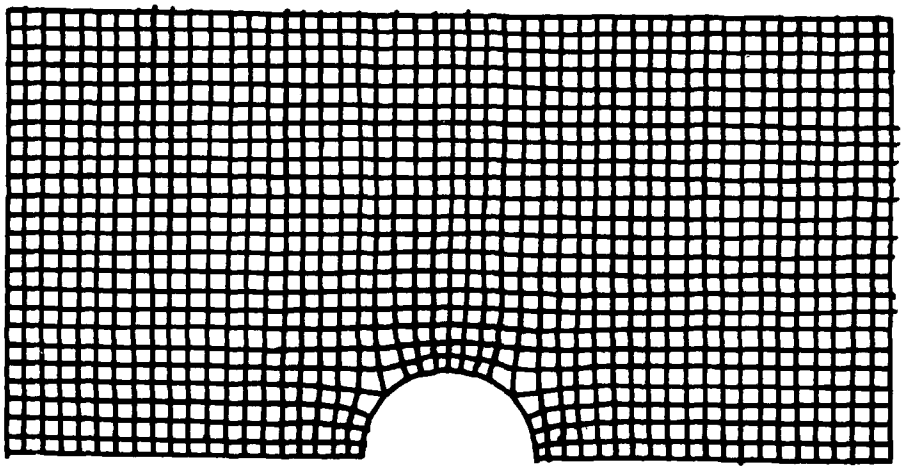
SOLVING FOR (M,R,k)

INITIAL GUESS: k = .31622, M_1 = .6+.8i, M_2 = 1+1.5i, R_1 = .1, R_2 = .06

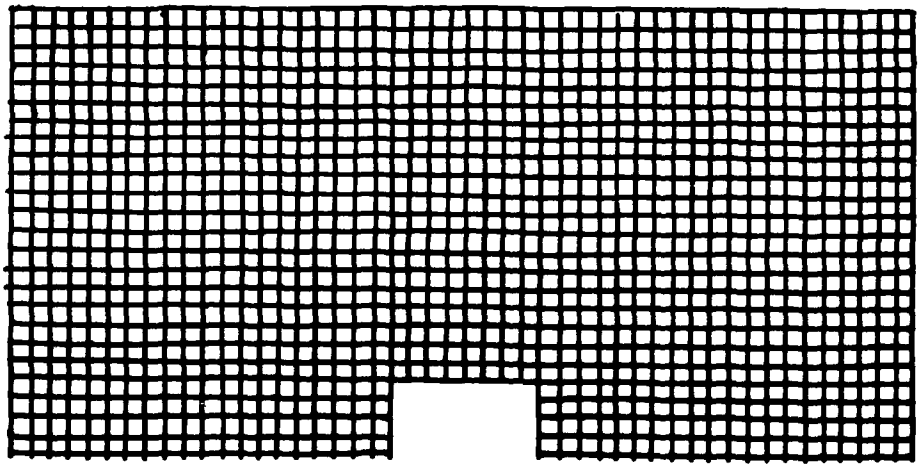
N	4	8	16
iterations	4	2	2
k	.3164766	.3164395	.3164345
M_1	.6110767+81218811	.6111020+.81218511	.6111145+.81219241
M_2	.9943007+1.4938876i	.9942437+1.49399121	.9942284+1.49400091
R_1	.1253212	.1271056	.1279121
R_2	.0914252	.0927318	.0933205
maximum error	4.4×10^{-9}	9.4×10^{-9}	2.5×10^{-10}

We did not use singular elements to approximate the charge density at the corners of the squares. That probably explains why the error increased from column one to column two. When the number of subdivisions was increased the potential approximation changed enough that even after two iterations the error was larger than before.

In plots 1 and 2 we show corresponding grids in a domain D_0 and in a conformally equivalent domain D. The domain D_0 is bounded by a square on the outside and a near circular inner boundary centered at the center of the square. D_0 was chosen to have a particularly simple potential associated with it, $\psi_0(z) = -\log \Delta(z,\frac{1}{2}(K+iK'), 1/\sqrt{2})$. In the same manner as with the more complicated domain associated with Table 3, we calculated the proper parameters for a conformally equivalent domain D with a square inner boundary. The symmetry of the regions provides a partial check of our procedure. We started with initial guesses where the outer boundary was not a square and M_1 was not in the center, but the paremeters converged to the correct ones which provided symmetry. At the outer corners of D_0 where our program specifically arranged to make a match with corners of the computational domain, we see that there is little distortion in the grid. The computational domain does have extra corners introduced on inner boundaries. The grid in D_0 shows the extent of the local distortion near the points corresponding to these corners in D. The distortions at corners would be similar if there were more interior boundaries. In our example we used a doubly-connected region D_0 merely to simplify the logic in the grid generation. In actual practice, however, a doubly-connected domain would probably be mapped onto an annulus in order to calculate a grid. Our method becomes useful particularly



Plot 1. Top half of physical domain D_0 with conformal grid.



Plot 2. Top half of computational domain D corresponding to D_0 in Plot 1.

for domains of higher connectivity.

Much more can be done to develop this conformal mapping method. We have indicated some of the directions to follow, and we have demonstrated the promise of this approach to a previously intractible conformal mapping problem.

REFERENCES

1. Harrington, A.N. Conformal Mappings onto Domains with Arbitrarily Specified Boundary Shapes, to appear in Journal d'Analyse Mathematique.
2. Trefethen, L.N. (1980) Numerical Computation of the Schwarz-Christoffel Transformation, SIAM J. Sci. Stat. Comput., 1, 82-102
3. Nehari, Z. (1952) Conformal Mapping, McGraw-Hill, New York

AD P000993

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

3-D SOLUTION OF FLOW IN AN INFINITE SQUARE ARRAY OF CIRCULAR TUBES BY USING BOUNDARY-FITTED COORDINATE SYSTEM

B. C-J. Chen,* T. H. Chien,* W. T. Sha*, and J. H. Kim**
*Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439; **Electric Power Research Institute, 3412 Hillview Avenue, Palo Alto, California 94303

INTRODUCTION

Heat transfer and fluid flow over circular tubes have wide applications in the design of heat exchangers and nuclear reactors. However, it is often difficult to accurately calculate the detailed velocity and temperature distributions of the flow because of the complex geometry involved in the analysis, and a lack of an appropriate coordinate system for the analysis. Boundary conditions on the surfaces of the tubes are often interpolated. This interpolation process introduces inaccuracy. To overcome this difficulty, the present study used the technique of the boundary-fitted coordinate system . In this technique, all the physical boundaries are transformed into constant coordinate lines in the transformed coordinates. Therefore, the boundary conditions can be specified on the grid points without interpolation.

COORDINATE TRANSFORMATION

The coordinate transformation technique used for the present analysis is based on the numerical solution of a set of elliptic partial differential equations (PDE).[1-3] The transformed coordinates (ξ, η, ζ) are independent variables; the physical coordinates (x, y, z) are dependent variables. Constant values of one of the curvilinear coordinates (ξ, η, ζ) are specified as Dirichlet boundary conditions on each boundary. Values of the other curvilinear coordinates are either specified by a monotonic variation over a boundary as Dirichlet boundary conditions, or determined by Neumann boundary conditions. In the latter case, the curvilinear coordinate lines can be made to intersect the boundary according to some specified conditions, such as being normal or parallel to some given directions. Also, the spacings of the curvilinear coordinate lines can be controlled. The PDE used for coordinate generation in the present analysis is of the general form,

$$\xi_{xx} + \xi_{yy} = P(\xi, \eta)$$

$$\eta_{xx} + \eta_{yy} = Q(\xi, \eta), \tag{1}$$

subject to boundary conditions,

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} \xi_1(x, y) \\ \eta_1 \end{bmatrix}, \quad (x, y) \in \tau \tag{2}$$

where ξ_1 is a specified monotonic function of x and y, η_1 is a specified constant, P and Q are functions for controlling spacings and τ is the physical boundary. P and Q were set equal to zero for the analysis presented here. For ease in specifying the boundary condition, the dependent and independent variables in Eq. (1) are interchanged as

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0, \tag{3}$$

where

$$\alpha = x_\eta^2 + y_\eta^2 ,$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta ,$$

$$\gamma = x_\xi^2 + y_\xi^2 . \tag{4}$$

The transformed boundary conditions for the present case of an infinite square array of circular tubes are given as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} R \sin \frac{\xi \pi}{12} \\ R \cos \frac{\xi \pi}{12} \end{bmatrix} \qquad \text{for } -3 < \xi < 3 \text{ and } \eta = 3$$

$$= \begin{bmatrix} R \sin \frac{\xi \pi}{12} \\ -R \cos \frac{\xi \pi}{12} \end{bmatrix} \qquad \text{for } -3 < \xi < 3 \text{ and } \eta = -3$$

$$= \begin{bmatrix} R \cos \frac{\eta \pi}{12} \\ R \sin \frac{\eta \pi}{12} \end{bmatrix} \quad \text{for } \xi = 3 \text{ and } -3 < \eta < 3$$

$$= \begin{bmatrix} -R \cos \frac{\eta \pi}{12} \\ R \sin \frac{\eta \pi}{12} \end{bmatrix} \quad \text{for } \xi = -3 \text{ and } -3 < \eta < 3, \qquad (5)$$

where R is the radius of the tubes. The grid spacings on the surface of the tube are uniform. Symmetry condition is imposed on the lines of symmetry as shown in Fig. 1. This symmetry condition causes the grid lines to be normal to the symmetry lines. The computational meshes so generated are shown in Fig. 1 for the case of pitch-to-diameter ratio (S/R) of 1.05. The inlet conditions of the flow and the geometry are also given in Fig. 1. All the physical properties are assumed to be constant. Equations (3) and (4), subject to the boundary conditions, Eq. (5), were expressed in finite-difference form and solved by successive-over-relaxation technique. Once the curvilinear coordinates are generated, the conservation of mass, momentum, and energy equations in terms of the transformed coordinates are solved.

A computer code (BODYFIT-1FE)[3] based on this procedure was developed at Argonne National Laboratory. BODYFIT-1FE is a three-dimensional, steady-state/transient single-phase thermal-hydraulic code for rod-bundle applications. It solves the complete Navier-Stokes and energy equations by a cell-by-cell numerical procedure. It uses a modified staggered-cell arrangement where velocity, energy, and mass-balance cells are all staggered at different locations. Detailed descriptions of the code are given in Ref. 3. Brief descriptions of the methods used in the code are given as follows.

CONTROL VOLUMES

In the conventional staggered mesh arrangement, the horizontal velocity, u, is stored at the middle of the vertical grid lines, and the vertical velocity, v, is stored at the middle of the horizontal grid lines. Pressures are stored at the centers of the cells formed by grid lines. In this arrangement, all velocities are sandwiched between pressures. Furthermore, the x-momentum equation for the u velocity depends on only the x-derivative of pressure, while the y-momentum equation for the v velocity depends on only the y-

derivative of pressure. Due to this characteristic in the Cartesian coordinate system, specification of pressures at physical boundaries is not required. This avoidance of specifying pressures at the boundaries is a great advantage of the conventional staggered mesh arrangement. However, in the boundary-fitted coordinate system, the x-momentum equation for the u velocity depends on both the ξ and the η derivatives of pressures and likewise for the y momentum equation for the v velocity. Because of this unique characteristic for the boundary-fitted coordinate system, either the pressure at the boundary has to be specified or the one-sided differencing scheme has to be used to evaluate the gradient of the pressure at the boundary. In order to avoid this difficulty, the conventional staggered mesh arrangemnt has been modified.

In the present scheme, both the horizontal and vertical velocities are stored at the intersections of grid lines as shown in Fig. 2. Pressure, temperature, enthalpy, and density variables are stored at the centers of the cells formed by the grid lines. In this arrangement, one can again avoid the specification of pressures at the boundaries. The control volumes used for momentum calculations center around the grid intersections as staggered cells shown in Fig. 2. The control volumes used for energy calculations are formed by the grid lines as basic cells shown in Fig. 2. However, the control volumes for mass-residue calculations are shifted half a cell in the ζ direction from the basic cells. This shifting of the control volume in the ζ direction is necessary to eliminate the numerical oscillation of cross flows, as to be explained next.

The present scheme of computing the pressure at a given cell is based on the mass residue of the cell. If too much mass is accumulated at a cell, the pressure of the cell is increased to push the mass out of the cell. If too little mass is accumulated in the cell, the pressure of the cell is decreased to pull in more mass. However, it is possible that the flows are in opposite directions at adjacent points of the computational cell such that the mass is nearly balanced and the mass residue is very small, and hence the pressure correction is negligible. In this condition, the above pressure correction scheme fails to correctly adjust the pressure and the oscillation of cross flow results. Furthermore, the rate of calculational convergence is extremely slow. The remedy to the above difficulty is to shift the control volumes for computing the mass residue half a cell up in the ζ direction such that the cross flow velocities are in the middle of the surfaces of the control volumes. This new arrangement entirely eliminates the numerical oscillation of cross flows and enhances the rate of calculational convergence significantly.

NUMERICAL METHODS

The transformed Navier-Stokes and energy equations were expressed in the finite-difference forms in the control volumes previously described. Donor-cell differencing was used for all the convective terms in the governing equations; central differencing for all the other terms. All variables were expressed in the current time step except the principle variable which had values at the current and previous time steps. Since the finite-difference equations were solved by using cell-to-cell calculational procedure, only the principle variable was expressed in the current iteration step such as

$$w_i^{t+\Delta t,n+1} = a_1 w_i^{t,n} + a_2 w_{i+1}^{t+\Delta t,n} + a_3 w_{i-1}^{t+\Delta t,n} + s, \tag{6}$$

where superscript t refers to the previous time step; t+Δt, the current time step; superscript n, the previous iteration step; superscript n+1, the current iteration step; subscript i, the principle cell to be solved; subscripts i+1 and i-1 the neighboring cells; a's, the coefficients; s, the source term; and w, velocity or enthalpy variables to be solved. All the terms involving w_i were factored to the left hand side of Eq. (6) to form the diagonal term to enhance the rate of calculational convergence. Equation (6) was first solved for velocities and then for enthalpy from cell to cell until all the computational cells within a given plane of the fluid domain were exhausted. The mass residues of all the cells were computed and the pressures were adjusted proportionally to the mass residue. This proportional constant used in adjusting the pressure has a sensitive effect on the rate of convergence. This constant was determined by substituting the finite-differenced momentum equation into the finite-differenced continuity equation and taking the partial derivative of the continuity equation with respect to pressure to obtain

$$\left(\frac{\partial S_m}{\partial P}\right)_{i,j,k}$$

where S_m was the mass residue of the continuity equation, and P was the pressure of the cell i,j,k. The reason for using the finite-differenced momentum equations instead of using the differential equations was that the rate change and the convective and viscous terms in the momentum equation can all be included to derive a relation between the velocity, $v_{i,j,k}$ and the pressure, $P_{i,j,k}$ of the cell as

$$v_{i,j,k} = \sum_{i'j'k'} (aP)_{i'j'k'} + s_{ijk}, \tag{7}$$

where i'j'k' are summed over the neighboring points of the ijk cell and s_{ijk} is the source term including $v_{i'j'k'}$ of the neighboring cells. Equation (7) was substituted into the finite difference continuity equation

$$(S_m)_{ijk} = \sum_{ijk} (av)_{ijk} + s_{ijk}, \tag{8}$$

to compute the partial derivative of S_m with respect to P_{ijk} as the factor $(\partial S_m/\partial P)_{ijk}$. The inverse of this factor gave the required change of pressure proportional to the mass residue of the cell as

$$(\Delta P)_{ijk} = (\Delta S_m)_{ijk} \Big/ \left(\frac{\partial S_m}{\partial P}\right)_{ijk}. \tag{9}$$

The use of the complete momentum equation instead of the conventional truncated momentum equation gave more accurate proportional factor for Eq. (9) and hence a faster rate of convergence for velocity and pressure calculations.

In most reactor applications, there exists a predominant direction of flow. A planar-mass-balance technique was used to speed up the convergence of the pressure calculation in these cases. The planar mass residue was computed by adding up all the cell mass residues, Eq. (8), for a given plane. Based on these planar mass residues, all the pressures across and downstream of the given plane were changed uniformly according to the planar mass residue. This technique provided a very effective way for speeding up the rate of convergence. Therefore, in the case of flow having a predominant direction, there were two procedures for pressure corrections, one in the cross-flow direction, and one in the axial direction of predominant flow.

The velocity, enthalpy, and pressure calculations were performed from cell to cell for a given plane, and then in a plane by plane sweep down the entire flow domain. This procedure was repeated until a converged solution was obtained.

APPLICATIONS

Several cases of infinite square array of tubes of different S/R ratios with the same hydraulic diameter and inlet conditions were studied. Water with the uniform velocity of 1 cm/s is flowing parallel to the axis of the cylindrical tubes arranged in a square pitch. Uniform heat flux was used to simulate the reactor fuel rod-bundle. Both the velocity and the temperature profiles were developing along the tubes. Since the analytical solution in the developing region for this configuration was not available, only the velocity and the temperature in the fully developed region were compared. For the axial velocity, Figs. 3(a) and 3(b) give the comparison between the BODYFIT results and the analytic solutions by Sparrow and Loeffler.[4] The agreements are in general very good. The total number of computational grid lines between tubes is fixed to be nine for all cases of different S/R ratios. In the case of large S/R ratio where tubes are far apart, the number of grid lines used in the present analysis may not be fine enough to resolve the detailed velocity profile. This slight inaccuracy can be seen in Fig. 3(a) for the case of S/R = 4.

For the comparison of temperature distributions, Table 1 gives the BODYFIT-calculated Nusselt number as a function of the dimensionless $\overline{Z} = (Z/D_e)/(RePr)$ for various S/R ratios. The Nusselt number at $Z = \infty$ is given by the analytic solution[5] for the case of constant heat flux. Reference 6 gives the similar analytic solution for the case of constant peripheral temperature. In the case of large S/R ratios, the two cases are very similar. In the case of small S/R ratios, the two cases differ quite a bit. However, the constant heat flux case is closer to the condition in reactor application than the constant peripheral temperatures case. The same information in Table 1 is plotted in Fig. 4. It is observed that the temperature profile reaches fully developed profile more slowly as the S/R ratio gets smaller. For the case of S/R = 1.05, the temperature profile did not fully develop at the length of 156 times of hydraulic diameter, D_e. This phenomenon also affects the comparisons shown in Fig. 3.

From the study, it is concluded that BODYFIT-1FE can provide detailed velocity and temperature distributions with good accuracy. This information is valuable for designing a mechanical heat transfer component. Furthermore, the code is very flexible and can provide analysis of the complicated geometries in most nuclear reactor applications.

REFERENCES

1. J. F. Thompson, F. C. Thames, and C. W. Mastin, "Boundary-Fitted Curvilinear Coordinate System for Solution of Partial Differential Equations on Fields Containing any Number of Arbitrary Two-Dimensional Bodies," NASA-CR-2729 (1977).

2. W. T. Sha and J. F. Thompson, "Rod Bundle Thermal-Hydraulic Analysis using Boundary-Fitted Coordinate System," NUREG/CR-0001, ANL-78-1 (Jan 1979).

3. B. C-J. Chen, W. T. Sha, M. L. Doria, R. C. Schmitt, and J. F. Thompson, "BODYFIT-1FE: A Computer Code for Three-dimensional Steady-State/Transient Single-Phase Rod-Bundle Thermal Hydraulic Analysis," NUREG/CR-1874, ANL-80-127 (Nov 1980).

4. E. M. Sparrow and A. L. Loeffler Jr., "Longitudinal Laminar Flow between Cylinders Arranged in Regulary Array," AIChE Journal Vol. 5, No. 3, pp. 325-330 (Sept 1959).

5. J. H. Kim, B. C-J. Chen, T. H. Chien, and W. T. Sha, "Heat Transfer in Longitudinal Laminar Flow between Cylinders Arranged in Regular Array," to be published.

6. J. H. Kim, "Heat Transfer in Longitudinal Laminar Flow along Circular Cylinders in Square Array," *"Fluid Flow and Heat Transfer over Rod or Tube Bundles,"* ASME, pp. 155-161, New York, New York (Dec 1979).

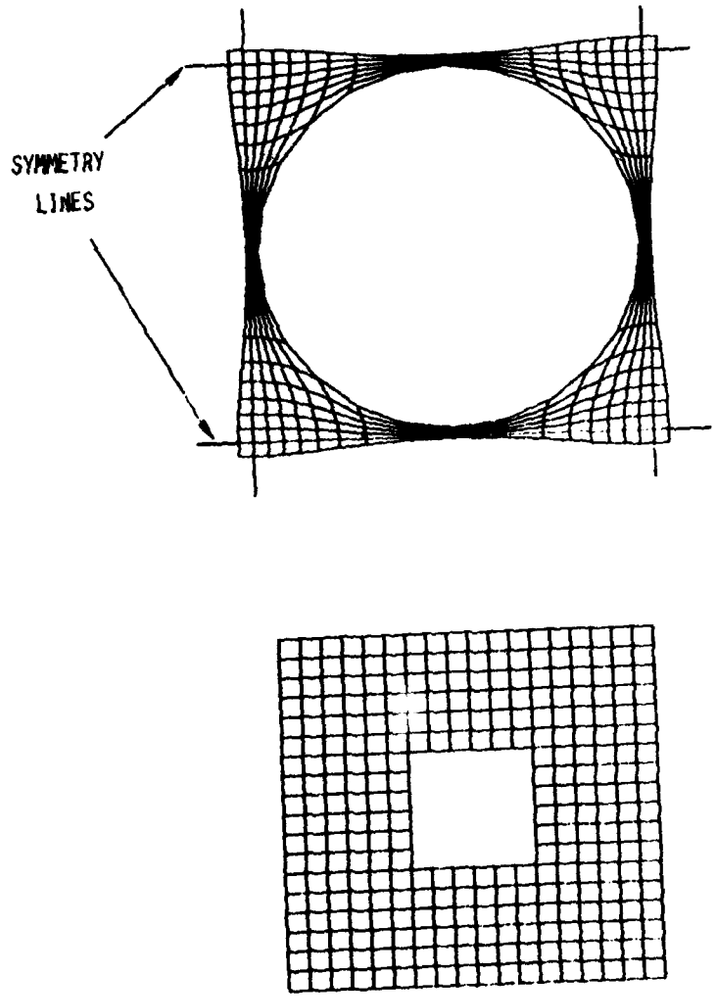
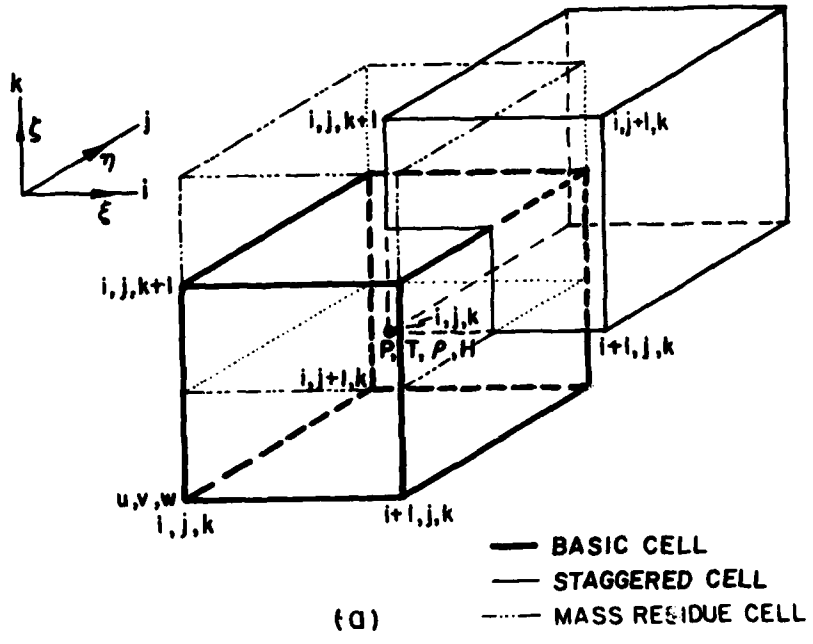
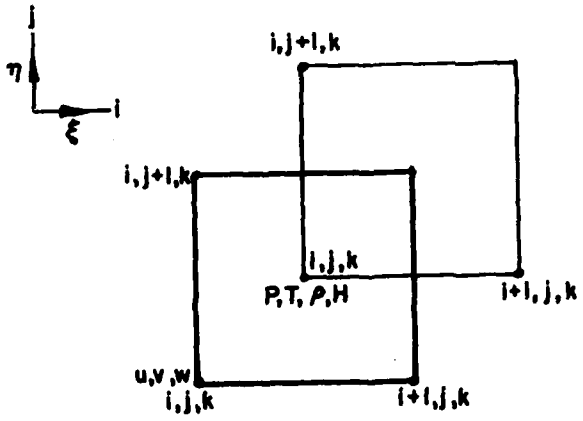


Fig. 1. Computational Mesh for a Unit Cylindrical Tube Arranged in a Square Pitch



(a)



(b)

Fig. 2. Control Volumes for Various Variables

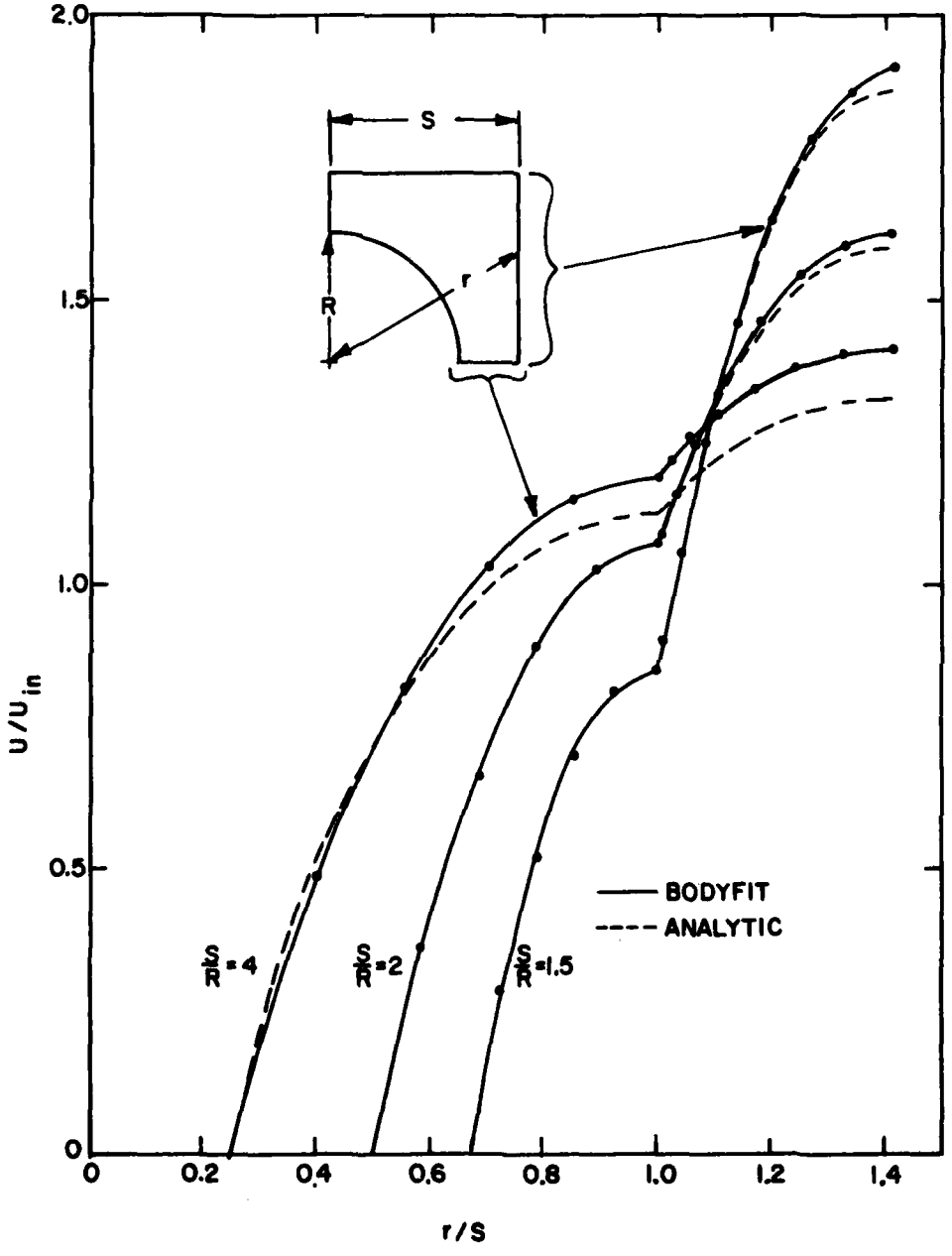


Fig. 3(a). Comparison of Axial Velocities between BODYFIT and Analytic Calculations for S/R = 4, 2, and 1.5

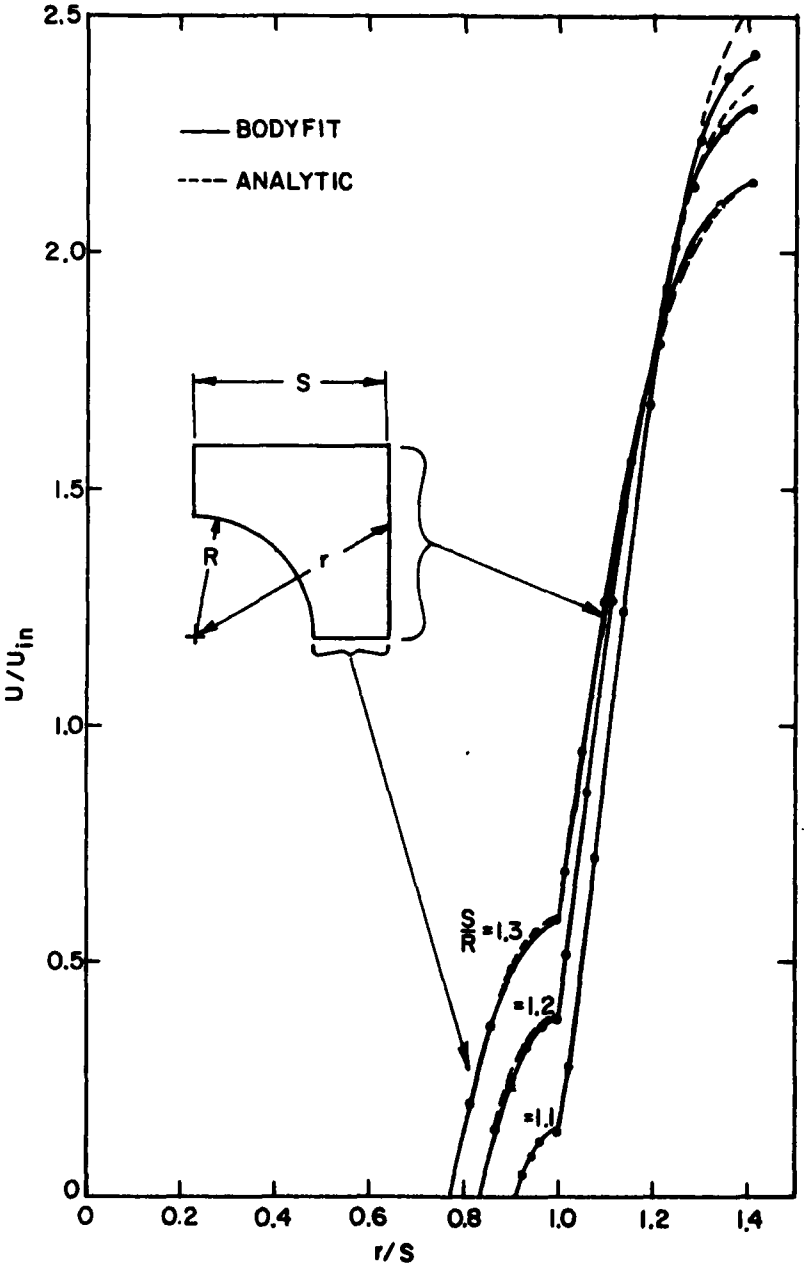


Fig. 3(b). Comparison of Axial Velocities between BODYFIT and Analytic Calculations for S/R = 1.3,1.2, and 1.1

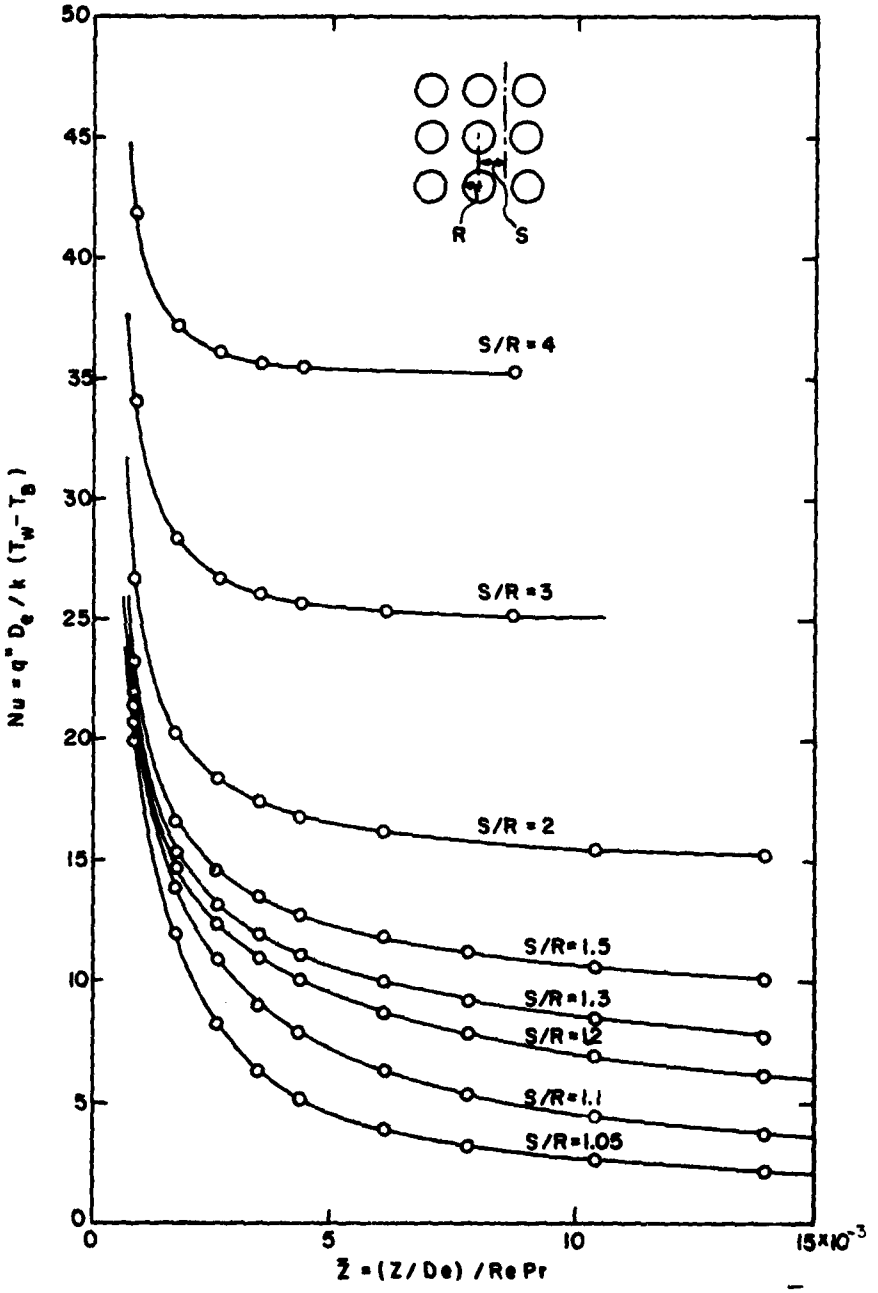


Fig. 4. BODYFIT Calculated Nusselt Number as a Function of $\bar{Z}$ for Various S/R Ratios

Table 1. (BODYFIT Calculated) Nussult Number

As a Function of $\bar{Z}$ for Various S/R Ratios

$\bar{Z}x10^3$	S/R=1.05	S/R=1.1	S/R=1.2	S/R=1.3	S/R=1.5	S/R=2	S/R=3	S/R=4
0.873	19.95	20.74	21.47	22.10	23.31	26.69	34.07	41.92
1.75	11.89	13.84	14.72	15.35	16.65	20.33	28.34	37.21
2.62	8.23	10.89	12.36	13.16	14.59	18.43	26.78	36.12
3.49	6.31	9.11	11.00	11.94	13.49	17.49	26.10	35.70
4.36	5.15	7.89	10.05	11.12	12.77	16.92	25.73	35.52
6.11	3.88	6.30	8.75	10.01	11.86	16.24	25.39	---
7.85	3.20	5.33	7.86	9.27	11.27	---	---	---
10.47	2.62	4.44	6.95	8.49	10.68	15.50	---	---
13.96	2.19	3.74	6.16	7.79	10.16	15.27	25.17	35.34
27.92	1.58	2.61	4.73	6.47	9.26	---	---	---
55.84	1.19	2.02	3.89	5.72	---	---	---	---
-*	0.92*	1.68*	3.68*	5.82*	9.29*	15.05*	25.23*	36.64*

*Analytic Solution (Ref. 5)

AD P000994

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

GENERATION OF BOUNDARY-FITTED COORDINATE SYSTEMS USING SEGMENTED COMPUTATIONAL REGIONS

Roderick M. Coleman
David W. Taylor Naval Ship Research and Development Center,
Bethesda, Maryland 20084

INTRODUCTION

INMESH is a computer program for the generation of two-dimensional boundary-fitted coordinate systems which allows the creation of an unlimited number of different types of meshes. This general program enables the user to design a mapping configuration suitable for even the most complicated geometric domains including those with multiple-connectedness. This versatility is achieved through the use of segmented computational regions tailored to handle the topological complexities of each individual problem. The proper choice of mapping configuration is particularly important for problems which contain two or more boundaries of different shapes, especially if these boundaries vary with time. Several computer programs have been written such as TOMCAT [1] and GRAPE [2] which allow the user to automatically generate a boundary-fitted coordinate system for a given geometry with some control over the distribution of the coordinate lines. These programs, however, are limited to certain predetermined mapping configurations. INMESH is not restricted in this manner and thus some of the previously-experienced need for coordinate system control has been eliminated. Its modular construction makes INMESH a convenient and efficient tool not only for the generation of curvilinear coordinate systems, but also for the numerical solution of partial differential equations using those coordinate systems.

DESCRIPTION OF THE METHOD

INMESH is based on the method introduced by Thompson [3] in 1974 which uses either the Laplace or Poisson equation to produce a mesh which adapts to the physical region of interest. The physical region under consideration in (x,y)-space is mapped to a computational region in (ξ,η)-space composed of the union of rectangular grid sections. The desired transformation represented by

$$\xi = \xi(x,y) \quad \text{and} \quad \eta = \eta(x,y) \tag{1}$$

is computed using numerical methods employed by Haussling & Coleman [4,5] to

generate similar mappings. The transformations are required to be solutions of the equations

$$\xi_{xx} + \xi_{yy} = P(\xi,\eta)$$
$$\text{and} \tag{2}$$
$$\eta_{xx} + \eta_{yy} = Q(\xi,\eta)$$

subject to appropriate boundary conditions. The source functions P and Q may be zero (a Laplace generating system) or nonzero (a Poisson system) in order to influence the spacing of coordinate lines in the physical region. At present, INMESH does not have the capability to automatically supply the proper values of P and Q needed to attract lines into or repel lines from specified regions of the physical plane. This feature may be added in the future, but the proper choice of mapping configuration should provide sufficient coordinate system control for most geometries.

For computational purposes, Equations (2) are transformed to (ξ,η)-space by interchanging dependent and independent variables to yield

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} + J^2(Px_\xi + Qx_\eta) = 0$$
$$\text{and} \tag{3}$$
$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} + J^2(Py_\xi + Qy_\eta) = 0$$

where

$$\alpha = x_\eta^2 + y_\eta^2 \qquad \beta = x_\xi x_\eta + y_\xi y_\eta$$
$$\gamma = x_\xi^2 + y_\xi^2 \qquad J = x_\xi y_\eta - x_\eta y_\xi \tag{4}$$

Each derivative in Equations (3) and (4) is replaced by the appropriate central difference approximation, and the resulting system is solved using successive over-relaxation. A more complete discussion of this solution technique can be found in previous work [5].

The computational region comprises rectangular grid segments which are mapped onto sections of grid in the physical region. The sides of the grid segments in computational space are joined together in such a way that the desired distribution of corresponding grid lines in physical space is

achieved. The sides of each rectangle are designated as one of three fundamental types: Fixed (F-type), Matching (M-type), or Re-entrant (R-type). For the F-type side, the x- and y-coordinates of the grid points are fixed and hence do not change during the computation of the mesh. An F-type side would be found in a grid section adjacent to a boundary in physical space such as a wall or far-field boundary. M- and R-type designations are used to connect two sides of different segments or different sides of the same segment. The two sides that are joined must be of the same type and have the same number of points. An M-type connection has an overlap of two grid lines while R-type sides are joined with an overlap of three grid lines. An R-type connection is often needed where a coordinate line in physical space lies partly along a boundary and partly in the interior of the region such as occurs when a body is mapped onto a slit. Unlike F-type sides, the physical coordinates of M- and R-type sides are not determined in advance but must be computed.

USE OF THE PROGRAM

To create a coordinate system with INMESH, the user must first decide on the mapping configuration that best suits the geometry of his particular problem. This step often involves making a sketch of the coordinate system as it will appear in the physical plane. The grid in the physical region is then broken up into sections which ensure that each side of each segment in the computational region is either fixed to a physical boundary or is joined to another entire side. The sides are defined as one of the three types (F,M, or R), with connections specified where appropriate. Finally, the coordinates of the fixed sides are set and the program is ready to be run. The coordinate system generated may be returned in a form suitable for graphical display or in a form to be used in the solution of a partial differential equation arising from a physical problem.

All user-supplied input to the program is given in the form of 80-column card images with variables in free format. The input consists of values of parameters which define the mesh segments, designate the type of each side, and provide the linkage between these sides. This input is simple with two parameters required for each segment to define its dimensions and at most three parameters per side to establish connections among the segments. The physical coordinates of the fixed sides must be specified where necessary. Also included in the input are values of parameters needed for the initial guess and iterative solution of the mesh. Generation of the initial guess can

be a problem in cases where some segments have no fixed side. This difficulty is currently being studied in an effort to improve the initialization algorithm.

GENERAL REMARKS

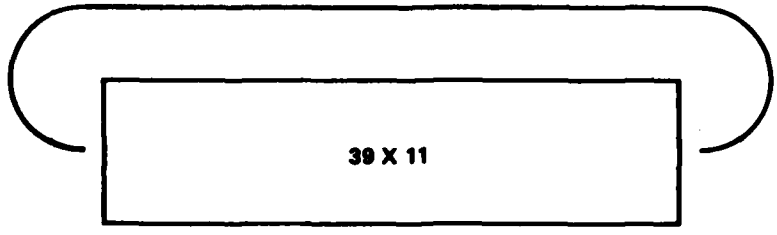
Almost any type of boundary-fitted coordinate system can be created with INMESH. The coordinate system that is generated may be easily changed by altering the number of and connections between grid segments in the computational region. Grid lines are added to or removed from specific areas simply by changing the number of points in the appropriate segments. This capability, along with the fact that the coordinate lines are smooth throughout the entire physical region, is used to control distribution of the lines so that the functions P and Q may be set to zero in the generating equations (a Laplace system). The flexibility provided by INMESH is especially useful for multi-body problems in which different mesh configurations are needed in different parts of the physical region.

Until better methods are devised and perfected, the choice of a "best" mesh for a particular problem will be made by examining different coordinate systems as they appear in the physical region. INMESH can aid in this decision by enabling the researcher to efficiently produce a wide variety of grids for the same geometry. This versatility will be enhanced when an interactive version of the program becomes available so that in a single terminal session a user can quickly generate and display a variety of meshes for the same region.

EXAMPLES AND APPLICATIONS

As a first example of meshes that can be created with INMESH, consider the simple computational region of Figure 1a and the associated physical region of Figure 1b. This figure illustrates a polar or "O-type" coordinate system about a circular body. In this and succeeding examples, R-type connections in the computational region are labeled as such. Unlabeled connections are thus M-type. Sides with no connections given are fixed. Although not to scale, sketches of the computational regions show how linkage of grid segments sides determines the nature of the coordinate system as it appears in the physical region. The number of grid lines in each direction is also indicated for the segments since sides that are joined must have an equal number of points. In Figure 1a the horizontal sides of the single computational segment

a)



b)

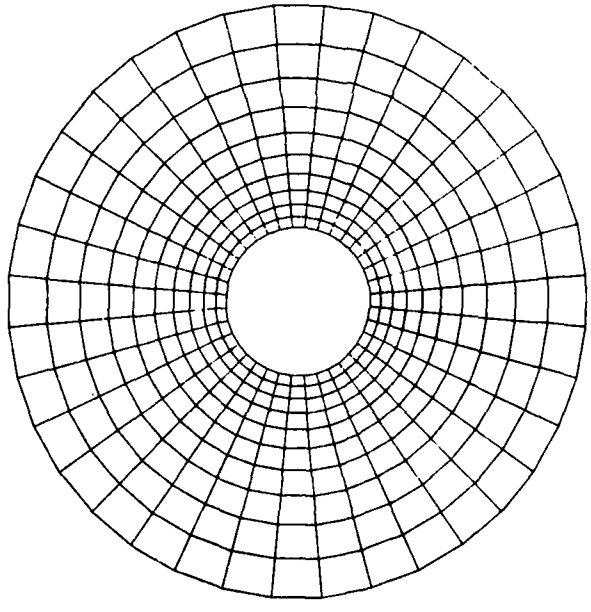


FIGURE 1. O-type mapping: single body
a) Computational region; b) Physical region

correspond to the outer boundary and the body in physical space and are thus F-type sides. The two vertical sides represent a cut between boundaries in the physical region and are joined here by an M-type connection.

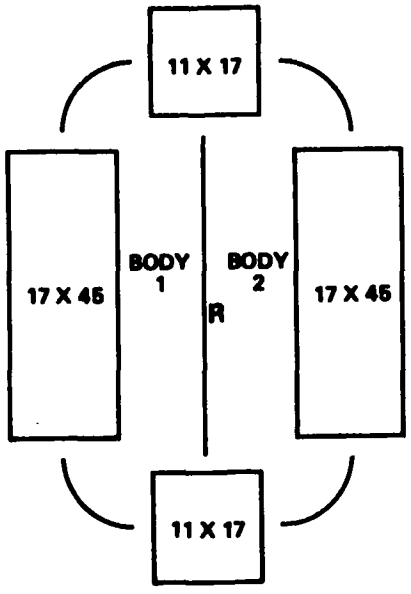
The second example demonstrates how the same type of coordinate system may be generated for a region with more than one body contained within a single outer boundary. Figure 2a shows the computational region for an O-type coordinate system about two foils; the physical region is shown in Figure 2b. M-type connections again form cuts between the outer boundary and the bodies while an R-type connection generates the coordinate line running from Body 1 to Body 2. Four separate computational segments are needed so that each side is either fixed or joined to another entire side. A coordinate system having a line both within the physical region and on the boundary usually requires a more complex computational space than a configuration without this property. Coordinate lines which change character in this manner are easily handled by INMESH but do necessitate a computational region with multiple segments.

Although the O-type grid seems to be a good choice for the circular body of the first example, this type of coordinate system may not be best suited for bodies shaped like those of the second example. Perhaps a more natural grid for a foil-like body is the "C-type" grid seen in the next example. Figure 3 shows the computational and physical regions for a C-type mesh about a foil. Note that three computational segments are required because the coordinate line that lies along the body also lies within the physical region. This is an example of a simple mesh that has many applications.

Both the inner and outer boundaries must often be considered in choosing a grid system for a particular geometry. As the next example, consider a rectangular region containing a foil-like body. Figure 4a shows the segmented computational space needed to map the body to a slit and Figure 4b shows the resulting coordinate system in physical space. Again the R-type connection is used to generate a coordinate line which "splits" to form the upper and lower portions of the body. Also note that different portions of the body are mapped to different grid segments for this configuration. A close-up view of this "slit-type" mesh near the body is given in Figure 5.

The slit transformation is easily extended to a multi-body situation. Figure 6 shows computational and physical regions for a grid system surrounding a cascade of foils. If the vertical arrangement of the foils is slightly changed, the mapping configuration is somewhat more complicated as illustrated by Figure 7. The increase in complexity of the computational region arises

a)



b)

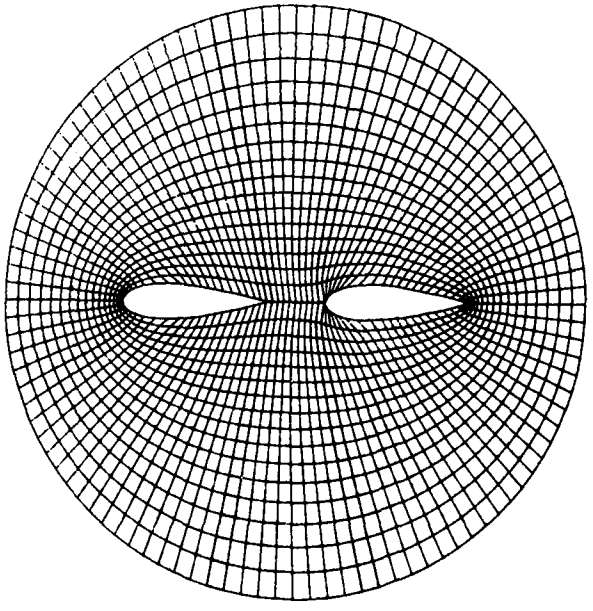
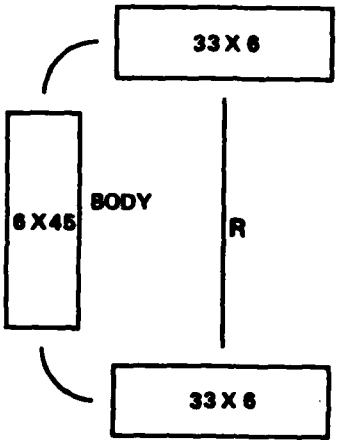


FIGURE 2. O-type mapping: multiple bodies
a) Computational region; b) Physical region

a)



b)

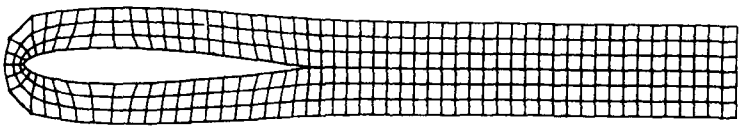
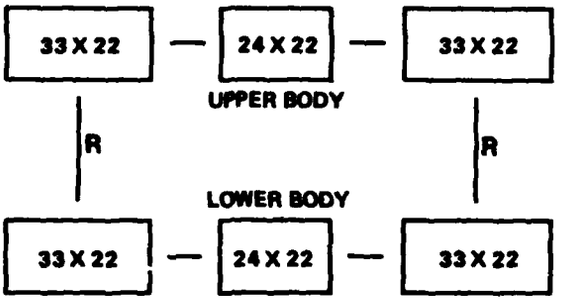


FIGURE 3. C-type mapping: single body
a) Computational region; b) Physical region

a)



b)

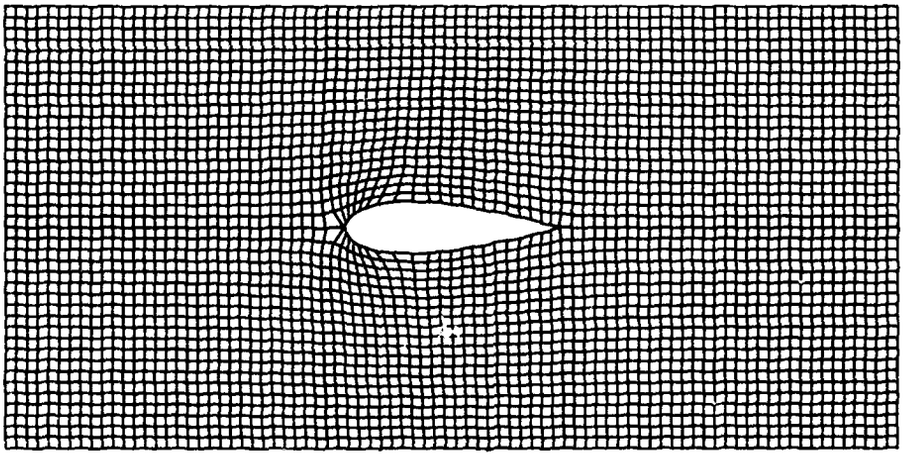


FIGURE 4. Slit mapping: single body
a) Computational region; b) Physical region

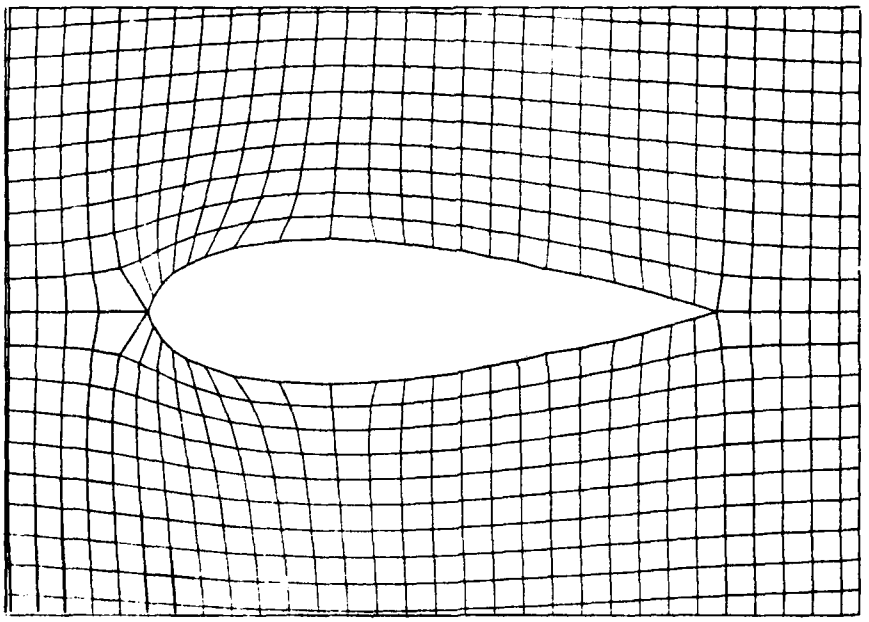
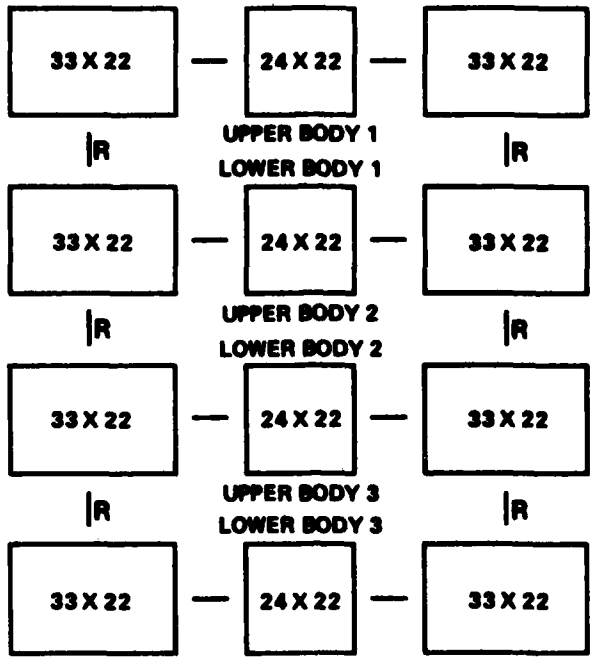


FIGURE 5. Detail of physical region for slit mapping

a)



b)

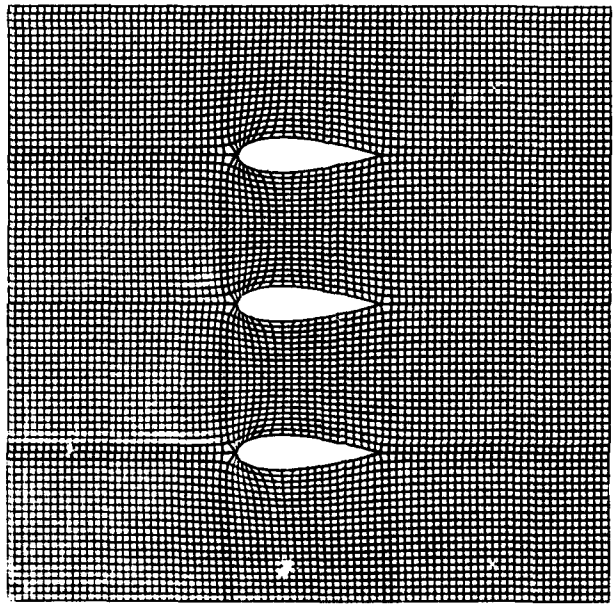
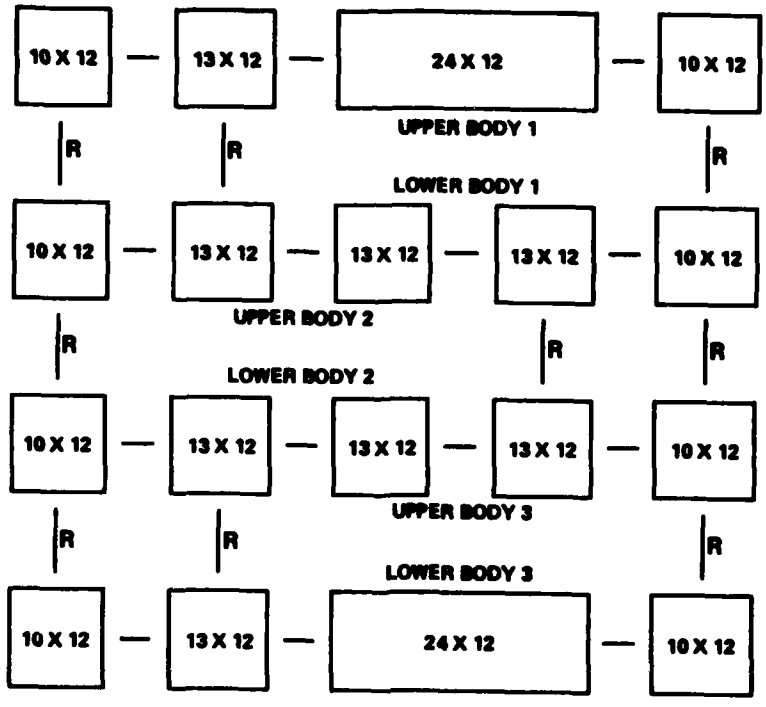


FIGURE 6. Slit mapping: cascade of foils
a) Computational region; b) Physical region

a)



b)

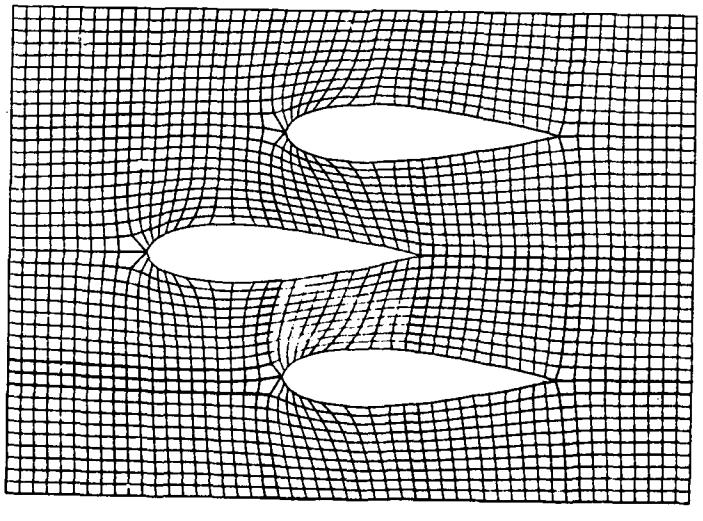


FIGURE 7. Slit mapping: staggered cascade
a) Computational region; b) Physical region

from the increase in the number of discontinuous coordinate lines in the physical region.

Another type of transformation well suited to a region with a rectangular outer boundary is the "box-type" mapping. Figure 8 shows the computational and physical planes for this system as applied to square region containing a circular body. In this configuration, portions of the body are mapped to four separate grid segments. If the body is shaped like a foil, a useful coordinate system might be one derived from a combination of the box and slit transformations, the "modified box" mapping. The computational and physical regions for the modified box are given in Figure 9. This arrangement, where the body is mapped to three different grid segments, accommodates both the blunt and sharp characteristics of the foil. Figure 10 provides detail of the grid near the body. The modified box mapping can be used in a straightforward manner for a staggered cascade of foils, as indicated in Figure 11.

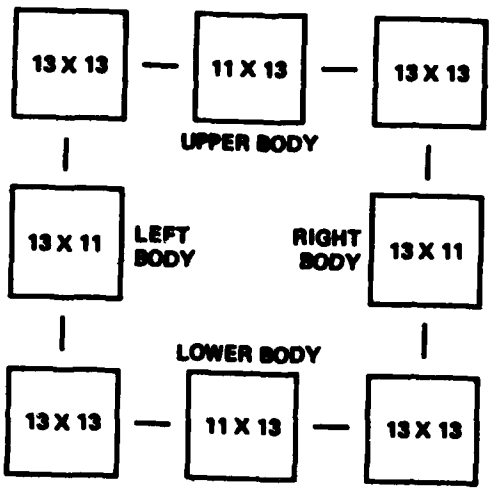
As a final example, consider again a rectangular region containing a foil-like body. In order to obtain more grid lines around the body than are provided by the modified box mapping alone, we choose a combination of the C-type and modified box transformations as shown by Figure 12. This configuration has coordinate lines which wrap the body yet become more rectangular near the outer boundary. In Figure 12b, note the irregular five-sided cells which appear both above and below the foil where the grid system changes character.

The coordinate system illustrated in Figure 12 has been used as the initial mesh for a preliminary calculation of the potential flow resulting from a hydrofoil moving beneath a free-surface. A finte-difference technique was used along with a marching scheme for the advancement of the time-dependent free-surface, requiring that a new mesh be generated at each time step as the free-surface evolves. Details can be found in a previous paper by Haussling & Coleman [4]. Figure 13 shows the development of the free-surface for various times t as the foil is abruptly put in motion. This computation was easily performed on the segmented grid since the same general subroutines used to generate the mesh were also used for the flow calculations.

CONCLUSION

A computer program (INMESH) for the generation of boundary-fitted coordinate systems has been written which makes use of segmented computational regions. This program is not restricted to certain predetermined mapping configurations and therefore can be used to produce an unlimited number of dif-

a)



b)

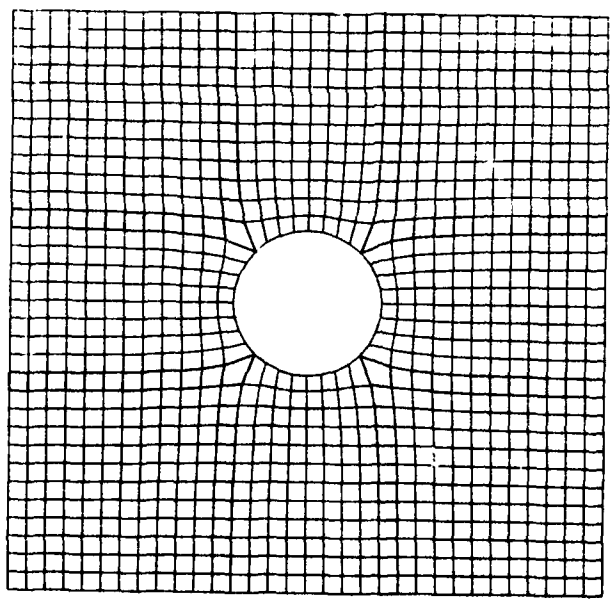
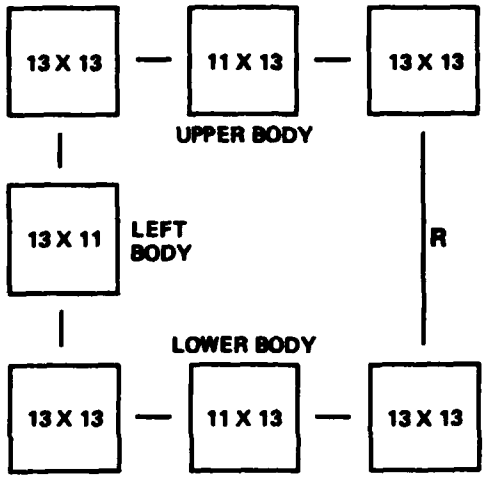


FIGURE 8. Box mapping: single body
a) Computational region; b) Physical region

a)



b)

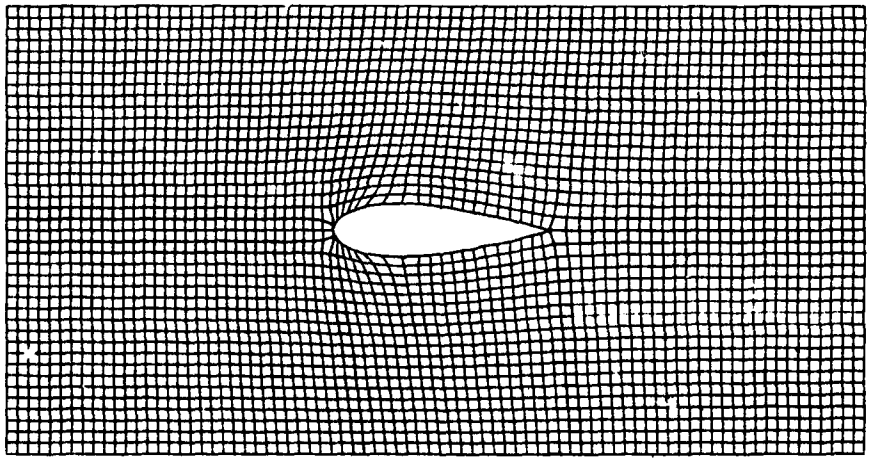


FIGURE 9. Modified box mapping: single body
a) Computational region; b) Physical region

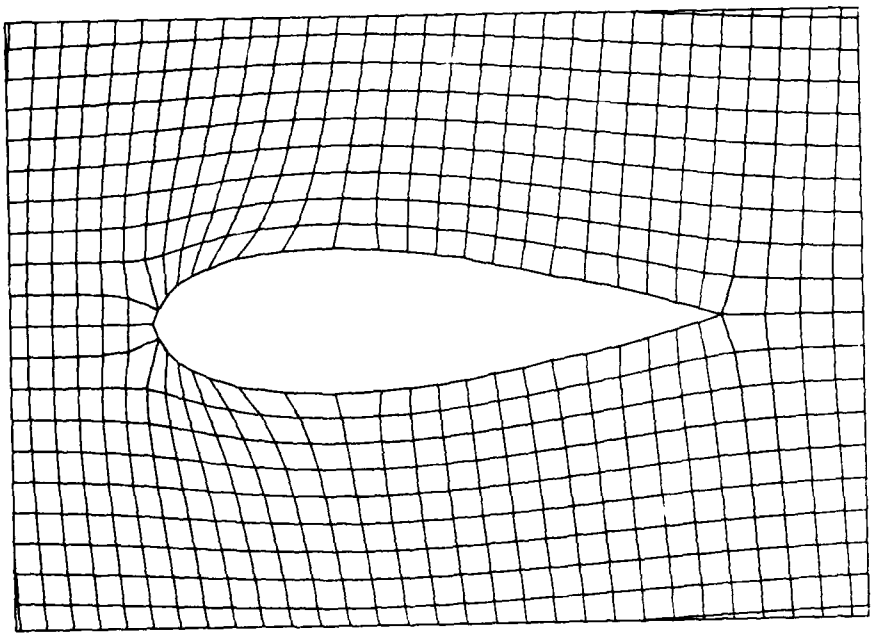
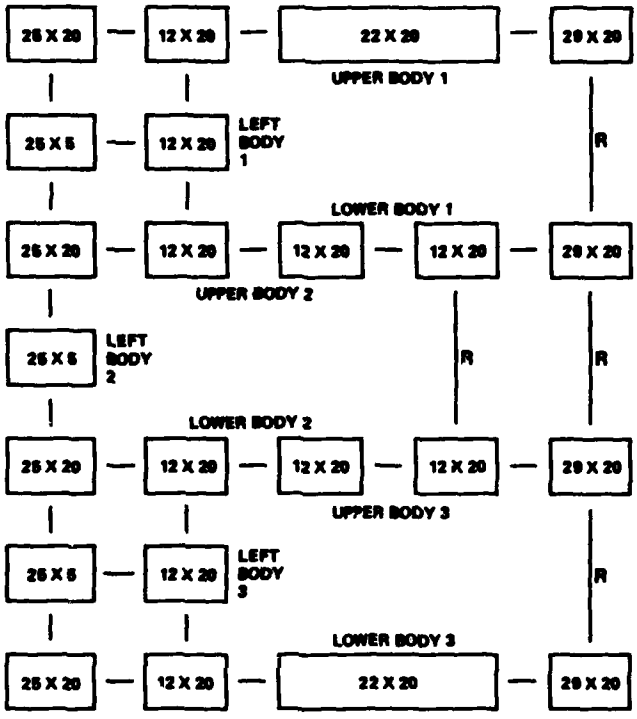


FIGURE 10. Detail of physical region for modified box mapping

a)



b)

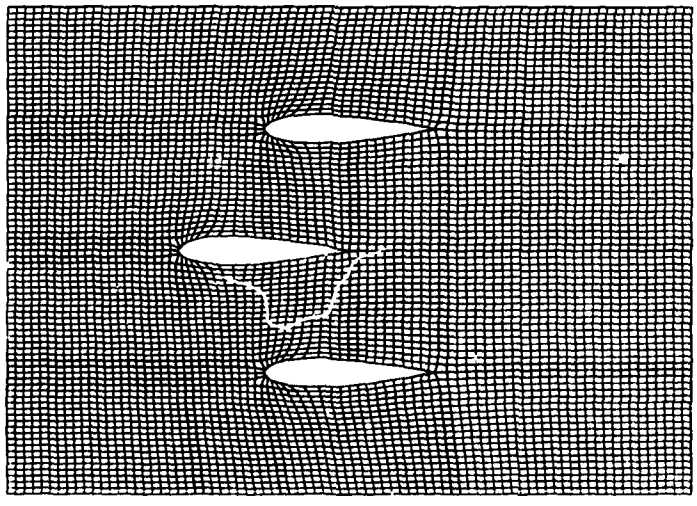
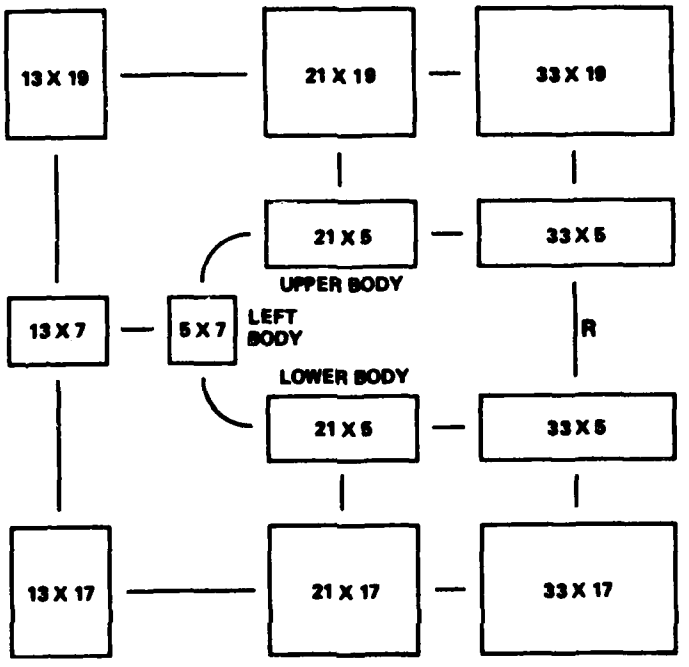


FIGURE 11. Modified box mapping: staggered cascade
a) Computational region; b) Physical region

a)



b)

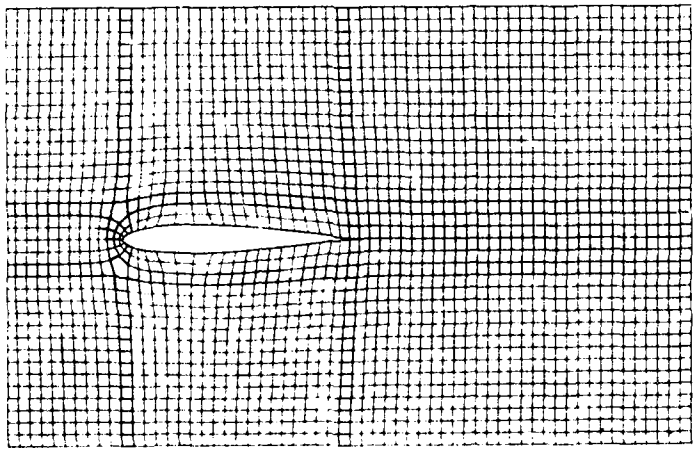


FIGURE 12. C-type/modified box mapping: single body
a) Computational region; b) Physical region

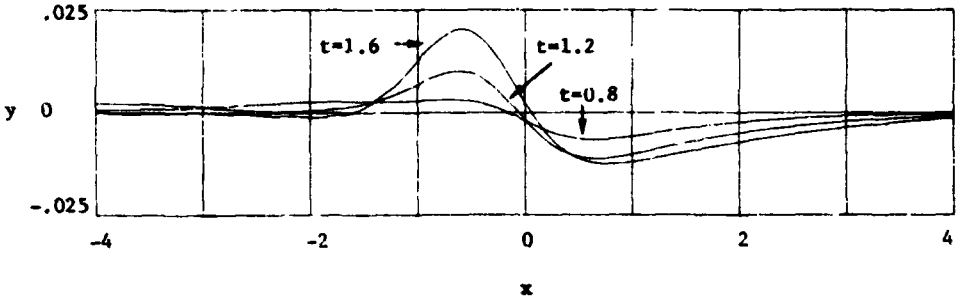


FIGURE 13. Free-surface development for translating hydrofoil

ferent grid systems. The fact that the user may design a computational region suited to a particular geometry makes INMESH a useful tool in the creation of curvilinear coordinate systems. The use of INMESH was illustrated by a series of sample grids ranging from simple coordinate systems to complicated meshes that would have been difficult and time-consuming to generate without INMESH. An application also demonstrated that the segmented computational region may be used successfully in the numerical solution of a fluid dynamics problem.

REFERENCES

1. J.F. Thompson, et al, "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems of Fields Containing Any Number of Arbitrary Two Dimensional Bodies," J. Comp. Phys., 24, pp. 274-302 (1977).
2. R.L. Sorenson, "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poisson's Equations," NASA Technical Memorandum 81198 (1980).
3. J.F. Thompson, et al, "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," J. Comp. Phys., 15, pp. 299-319 (1974).
4. H.J. Haussling and R.M. Coleman, "Nonlinear Water Waves Generated by an Accelerated Circular Cylinder," J. Fluid Mech., Vol. 92, Part 4, pp. 767-781 (1979).
5. H.J. Haussling and R.M. Coleman, "Finite-Difference Computations Using Boundary-Fitted Coordinates for Free-Surface Potential Flows Generated by Submerged Bodies," Proc. 2nd Int. Conf. Numerical Ship Hydrodyn., Univ. of California at Berkeley, pp. 221-233 (1977).

AD P000995

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

GRID GENERATION BY ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS FOR A TRI-ELEMENT AUGMENTOR-WING AIRFOIL

REESE L. SORENSON
Ames Research Center, NASA
Moffett Field, California, U.S.A.

ABSTRACT

The Augmentor-Wing is a promising new development in airfoils for transport aircraft. It consists of a main airfoil with a slotted trailing-edge for blowing, and two smaller airfoils shrouding the blowing jet. Impressive short-takeoff-and-landing (STOL) results have been observed in flight, and both wind-tunnel tests and computational studies predict remarkably low drag in the cruise configuration.

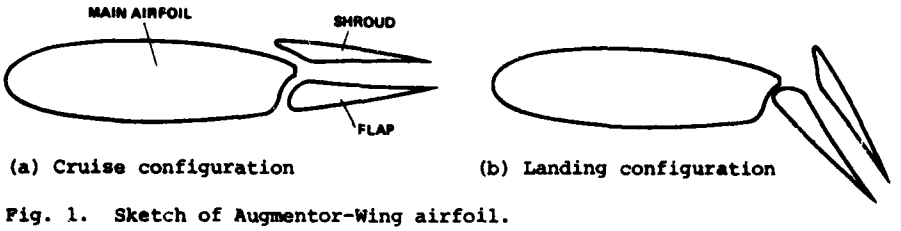
Two separate efforts to numerically simulate the flow about this airfoil in the cruise configuration are being pursued. One uses an unsteady viscous flow-solver, and the other a transonic full-potential approach. The inviscid full-potential approach will be used as a preliminary design tool to test a variety of configurations because of its relatively high speed; the viscous flow-solver will be used as a design verification tool to supplement experimental testing. The grid-generation program GRAPE has been adapted to satisfy the significantly different needs of these two studies. The flow field is divided into zones, and continuity of grid lines and their slopes across zonal boundaries is assured. In addition, grid cell size and skewness at body boundaries is controlled. This new application of grid generation by elliptic partial differential equations for this "real-world," multielement problem is described.

THE AUGMENTOR-WING

The Augmentor-Wing,[1] designed and built by De Havilland Aircraft of Canada, Ltd., consists of a thick nonsymmetric main airfoil having a slotted trailing-edge for blowing, a large flap, and an additional airfoil element, called the shroud (see Fig. 1a). The shroud is located above the blowing jet and the flap below it, forming a channel and acting together as an ejector, or thrust augmentor. Air for blowing is ducted from the bypass flow of the turbofan engines.

The flap and shroud are fixed with respect to each other and rotate downward together for the landing configuration, directing the blown air and entrained

PREVIOUS PAGE IS BLANK



(a) Cruise configuration (b) Landing configuration

Fig. 1. Sketch of Augmentor-Wing airfoil.

air downward and providing powered-lift (see Fig. 1b). Blown air is cross-ducted, eliminating the danger of asymmetric lift in the engine-out case. As a result, it is not necessary to use four engines--as opposed to the potentially more economical twin-engine arrangement--for safety considerations, as it is in some other powered-lift designs. To test the concept, a De Havilland Buffalo aircraft was modified by adding Rolls-Royce Spey jet engines and an Augmentor-Wing fixed in the landing configuration. Five years of research flying with this aircraft showed good short takeoff and landing (STOL) performance.[2,3]

Theoretical studies and wind-tunnel tests of scale models[4-6] indicate that this airfoil in the cruise configuration will be competitive with, if not superior to, a conventional supercritical airfoil section of the same thickness-to-chord ratio. Advantages claimed are increased drag-rise Mach number, improved buffet boundaries, and reduction of "effective" drag because of full-time blowing. It is expected that a large part of the main-airfoil boundary layer can be ingested into the augmentor and reenergized, resulting in increased propulsive efficiency. Further, boundary layers formed on the flap and shroud do not tend to separate; this gives a "flatter" pressure distribution over the entire airfoil and reduces peak suction pressures, which lead to high values of critical Mach number. In addition, the 18% overall thickness-to-chord ratio (including flap and shroud) allows the main-airfoil element to be extremely thick, at 26%. This facilitates the placement of large low-loss blowing ducts, gives more room for fuel, and yields structural advantages such as reduced weight and increased torsional stiffness, thus permitting higher aspect ratios.

THE "GRAPE" GRID-GENERATION ALGORITHM

The grid for discretizing the flow about the Augmentor-Wing is constructed using a component adaptive grid interfacing (CAGI) technique which employs

Sorenson's grid-generation program GRAPE.[7,8] The GRAPE algorithm generates grids by solving a set of two Poisson differential equations after the manner of Thompson et al.:[9]

$$\xi_{xx} + \xi_{yy} = P \tag{1a}$$

$$\eta_{xx} + \eta_{yy} = Q \tag{1b}$$

These equations are solved by iteration in the transformed plane:

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -J^2(Px_{\xi} + Qx_{\eta}) \tag{2a}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2(Py_{\xi} + Qy_{\eta}) \tag{2b}$$

where

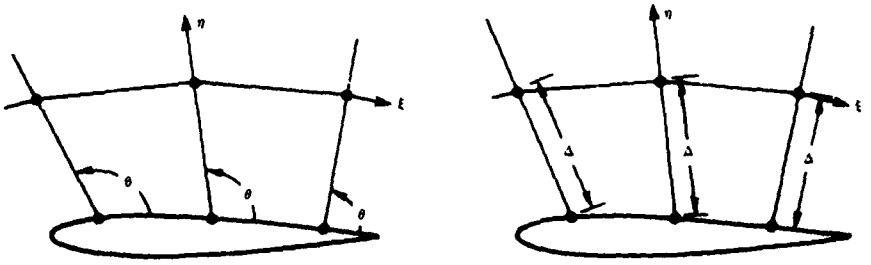
$$\alpha = x_{\eta}^2 + y_{\eta}^2 \tag{2c}$$

$$\beta = x_{\xi}x_{\eta} + y_{\xi}y_{\eta} \tag{2d}$$

$$\gamma = x_{\xi}^2 + y_{\xi}^2 \tag{2e}$$

$$J = x_{\xi}y_{\eta} - x_{\eta}y_{\xi} \tag{2f}$$

The GRAPE method differs from that of Thompson et al.[9] in that it uses inhomogeneous terms P and Q, which though simpler, yield the ability to arbitrarily impose two types of control on the grid at boundaries. The computational coordinate ξ is directed around the airfoil; thus, lines of constant ξ are those which intersect the airfoil boundary. The first type of control which may be imposed is of the angle of inclination, denoted θ in Figure 2a, with which the lines of constant ξ intersect the boundaries. Grid lines directed around the airfoil are of constant η. The second type of control is of the distance between the airfoil boundary and the next constant-η line, denoted Δ in Figure 2b, and measured along constant-ξ lines. Both θ and Δ may be constants, or any reasonable and smooth function of ξ. Thus, for example, near-orthogonality may be achieved at the boundary by setting θ to a constant value of 90°; the "normal distance" off of the body to the next grid line, Δ, may be set to any fraction of the chord length, such as



(a) Control of angles (b) Control of spacing

Fig. 2. Two types of grid control at boundaries

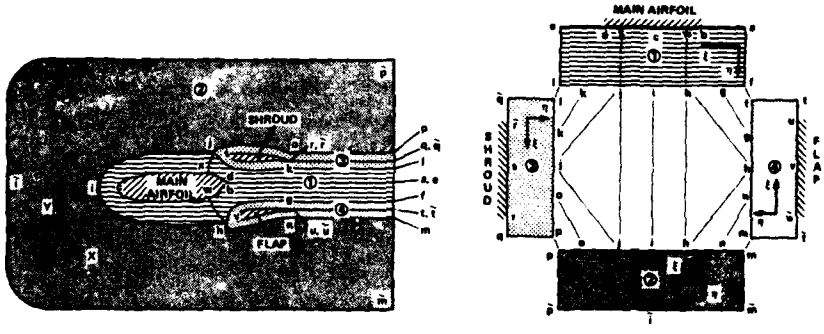
0.01 times the chord length. This control of θ and Δ may be exercised at the inner or outer boundary, or both.

The constraints on θ and Δ are embedded implicitly within the equations governing the grid mapping, and thus control of θ and Δ is dependent on the accuracy with which the equations are solved. Some compromise of this control must be accepted, for example, near any point where the slope of the boundary is discontinuous. But in most places on most grids the control is quite effective.

APPLICATION OF GRAPE TO THE TWO FLOW SIMULATION STUDIES

The two methods discussed here for numerical simulation of the flow about the Augmentor-Wing--the inviscid full-potential and viscous thin-layer Navier-Stokes methods--both require surface-conforming grids. Each method has its own particular grid requirements. The CAGI approach satisfies these requirements by partitioning the flow region into computational zones, each in the shape of a distorted "C."

In the viscous flow-simulation effort, Lasinski et al.[10] use a steady, thin-layer Navier-Stokes solver based on the work of Steger,[11] with the main airfoil modified to have a sharp trailing edge for the no-blowing case. This method requires a body-oriented C-type grid about each element, with the lines of constant η clustered to the body surface and lines of constant ξ nearly normal to the body. Thus, in the CAGI approach four zones are required: a thin zone conforming to the flap, another thin zone conforming to the shroud, a zone conforming to the main airfoil and passing between the flap and shroud grids, and a fourth zone filling the outer, or far-field region. Figure 3a illustrates the four zones (not to scale). The GRAPE method is then used to make grids for each of the zones in turn. Function and slope continuity of



(a) Physical space (b) Computational space

Fig. 3. Grid mapping.

coordinate lines across zonal boundaries is achieved by using GRAPE's ability to control angles and spacings at boundaries.

The first step in this method is to locate the zonal boundaries. This is done in an automatic and arbitrary fashion. Short line-segments are projected normally outward from the body points. The outward end-points of these line-segments are connected to form the C-shaped lines denoted f-g-h-n-m (about the flap), p-o-j-k-l (about the shroud), and f-g-h-i-j-k-l (about the main airfoil), as shown in Figure 3a. The line-segments normal to the bodies are then discarded, leaving the zonal boundaries.

A grid for each zone is then made in turn, using the program GRAPE as though it were a subroutine. Because the computational variable ξ is proportional to the distance around each airfoil element, lines of constant ξ are those which extend outward from each airfoil element in a radial-like fashion. It is ensured that lines of constant ξ are continuous across zonal boundaries by using the same boundary points for adjacent zones. Further, it is ensured that lines of constant ξ have continuous slopes and continuous point-spacing in the η-direction across zonal boundaries by either (1) imposing a fixed point-spacing and the condition of local near-orthogonality at boundaries where zones are adjacent, or (2) generating the grid for one zone, then imposing the angles and point-spacing of its constant-ξ lines at the boundary upon the adjacent grid as an a priori condition.

Technique (2) above is best understood by an example. The grid in the zone conforming to the shroud is made first, with θ at the inner boundary set to 90° and Δ at the inner boundary set to 10^{-5} times the chord length

of the main airfoil. No control is exercised at the outer boundary of this zonal grid. The angles and spacing that *result* at the outer boundary of this grid, however, are used as constraints in subsequently generating the two adjacent zonal grids. To illustrate this, note the line j-o-p in Figure 3a. This line is part of the outer boundary of the grid conforming to the shroud, and is also part of the inner boundary of the far-field grid. Thus the angles and spacings along this line j-o-p *resulting* from the generation of the grid about the shroud are imposed as θ and Δ along that part of the inner boundary of the far-field grid. The result is that angles and spacing vary smoothly across zonal boundaries.

The four-zone physical domain for the viscous case is mapped into a computational domain consisting of four distinct blocks, as shown in Figure 3b. These blocks are adjacent along parts of their boundaries, as shown by the connecting lines between the blocks. The flow solver must take careful note of how and where the blocks join. Of special note are points j and h; three zones are contiguous at each of those points.

Work is proceeding on the viscous case with blowing.[12] The upper and lower surfaces of the main-airfoil element approaching the trailing edge are faired to be parallel with the x-axis, and are truncated at the point which gives the proper blowing jet width (see Fig. 4). Zonal boundaries are extended rearward from the top and bottom of the truncation, as extensions of the coordinate boundary conforming to the surface of the main airfoil. Thus a fifth zone is created and a grid for it is made. Within this new zone, lines of constant η are directed generally fore and aft and are clustered to the two new zonal boundaries. The lines of constant ξ are roughly vertical, and have end points that line up with those of grid lines of constant ξ in the adjacent

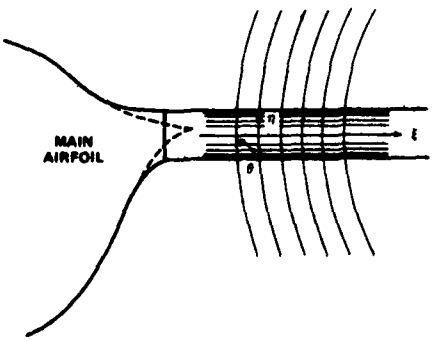


Fig. 4. Creation of fifth zone for blowing case.

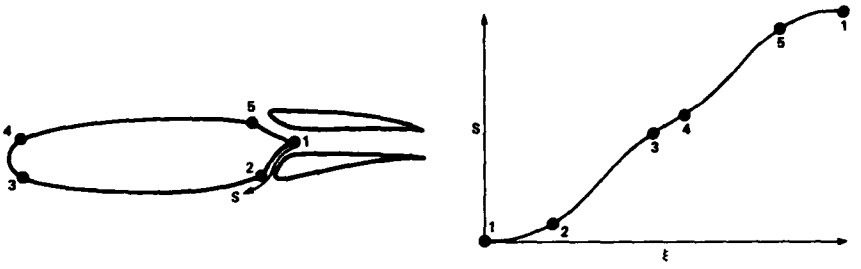
zone. Because lines of constant ξ approaching the new zonal boundaries from both sides are forced to do so with the same θ (90°) and with the same Δ, function and slope continuity of coordinate lines across zonal boundaries is maintained. The Navier-Stokes flow solver is being adapted to utilize the fifth zone with appropriate boundary conditions. An example grid for the blowing case is presented below in the results section.

In the inviscid flow-simulation effort, Flores[13] is adapting the TAIR full-potential solver of Holst[14] first to use a C-type grid, and then to accept the flap and shroud as residing in slits within the grid. Thus the grid about the flap and shroud locally resembles an H-type mesh. GRAPE has been applied to the generation of this grid, using a two-zone approach, with one zone conforming to the main airfoil and to the inboard (toward the jet) surfaces of the flap and shroud, and with the second zone covering the outer region and conforming to the outboard sides of the flap and shroud. The point-spacings along lines of constant ξ at the body surfaces are much larger in the inviscid case, being chosen to give cells with aspect ratios near 1. An example is discussed below in the results section.

BODY-SURFACE DISTRIBUTION

It was realized at the outset of this effort that the distribution of grid points on the surfaces of the three airfoil elements would be a problem requiring serious attention. In all computational aerodynamic applications, the distribution of body-surface points is important. The points must be dense enough to resolve changes in the gradients of the flow variables along the surfaces. In addition, point distribution around the body must be smooth. Yet this must be done with a minimal number of points, for the sake of computational efficiency. However, in this multielement airfoil problem the body-surface-point distribution is even more critical, since the points on one element must "line up" with points on another element; the number of points on the two sides of a channel, such as between the flap and shroud of the Augmentor-Wing, must agree.

A method for distributing body-surface grid points that is simple and straightforward yet effective has been developed. The procedure is one of relating the surface distance around the airfoil, s, measured clockwise from the trailing edge, and illustrated in Figure 5a, to the computational variable ξ, with integer values of ξ corresponding to actual grid points. The function $s(\xi)$, described by the curve in Figure 5b, shows an actual surface-point distribution. This function must be smooth and monotonically increasing.



(a) Arclength s and control points (b) Sample s vs ξ distribution

Fig. 5. Surface distribution algorithm

Where the slope is small, the surface points are close together; where the slope is large, the surface points are far apart.

Plots of $s(\xi)$ such as that in Figure 5b are commonly used to examine and evaluate a point distribution, but in this method the function $s(\xi)$ is used to *create* the distribution. Because index values in the ξ direction, hence values of ξ, are assigned a priori to physical locations on the body, some small number of points on the $s(\xi)$ curve are predetermined. The remainder of the $s(\xi)$ curve is found by fitting the given points with a spline function. As a result, the entire $s(\xi)$ function is found, and a distribution is created.

The term "control point" is used here to mean a point on the airfoil surface to which we wish to assign a particular value of j, the index in the ξ direction. By assigning index values to physical locations in this way, clustering or declustering in the regions between control points can be enforced. For example, Figure 5a shows five control points on the main-airfoil element. Suppose that it is desired to put 100 points over the entire surface of this airfoil. In this case a uniform distribution of grid points [which would make $s(\xi)$ linear] would result in eight grid points between control point No. 1 at the trailing edge and control point No. 2 at the entrance to the lower channel, since this distance is 8% of the circumferential length of the airfoil. Then suppose the body-surface spacing in this region is halved. This is done by requiring 16 points in the region, necessitating the removal of 8 points from other regions. A correspondence between the 16th ξ-value ($j = 16$) and the s-value of that control point is therefore required, and the function $s(\xi)$ is no longer linear. Other correspondences between s-values locating control points and ξ-values specifying clustering are established around the leading edge and the upper channel. These control

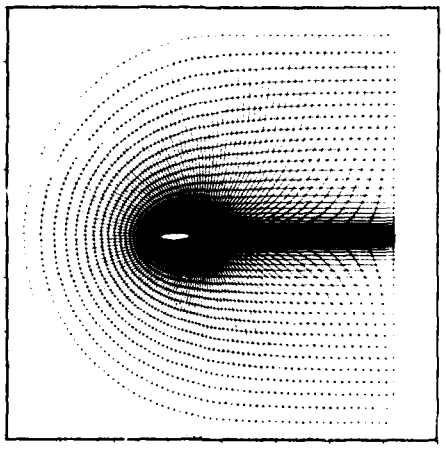
points and their ξ-values could be plotted on an $s(\xi)$ graph such as the points on Figure 5b. Any reasonably smooth and monotonic curve passing through the control points would serve as the desired surface-point distribution. In the present method this curve is found by fitting the control points with a cubic spline. The actual values of s, denoted s_j, which define the distribution of body points, are found by evaluating the spline fit at integer values of ξ. The body shape is represented parametrically by the spline fits $x(s)$ and $y(s)$. The actual coordinates of the body surface points are found by evaluating the $x(s)$ and $y(s)$ spline fits at the parameter values s_j.

A similar procedure is followed for all three airfoils. It can thus be ensured that there is an equal number of points on both sides of the channel by specifying the same number of points between opposing control points. In practice more control points than are shown in Figure 5a are required to control the second derivative $s''(\xi)$. Some trial and error is required, but the method is effective.

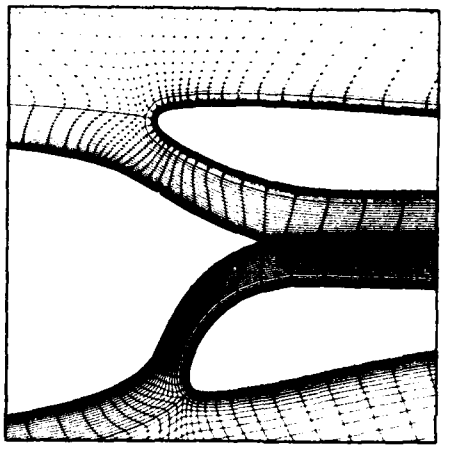
A solution-adaptive approach to surface-point distribution based on the work of Johnson[15] and Nakamura and Holst[16] is also being investigated. A partial differential equation involving the second derivative of the fluid density is solved to give a reclustering. Although this method appears promising for simpler single-element airfoils, problems relating to the multielement aspect of this airfoil have been encountered. The method tends to give surface-point distributions in channel regions wherein points on one side are aligned poorly with the points on the opposing side, causing lines of constant ξ to be highly skewed.

RESULTS

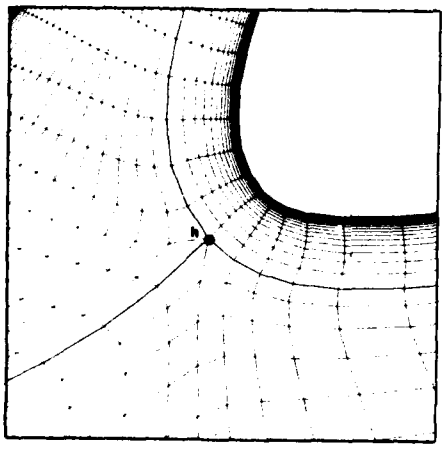
Figure 6 shows a grid generated by the present method and used in calculating some of the results for the no-blowing viscous case reported by Lasinski et al.[10] Of particular interest is Figure 6c; it shows a close view of the region near the leading edge of the flap, including the point labeled h in Figure 3a, at which three computational zones come together. Where "chord" is taken to mean the chord-length of the main airfoil element, the standoff distance to the first point, Δ, is set to 10^{-5} chords for all three airfoil elements. The total thickness of the grids conforming to the flap and shroud, that is, the length of the line-segments used in creating the zonal boundaries, is 0.01 chords. The total thickness of the grid conforming to the main airfoil element varies but is about 0.03 chords. The far-field grid extends 4 chords upstream, 6 chords downstream, and 6 chords vertically upward and downward.



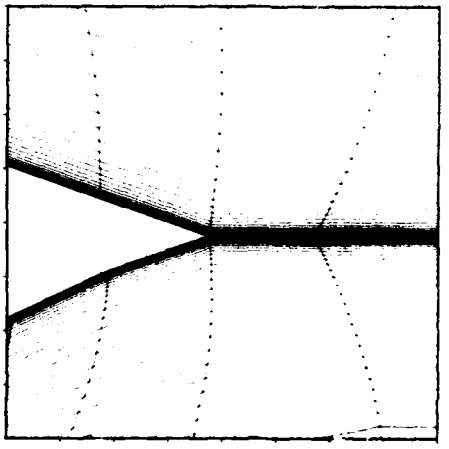
(a) Entire grid



(b) Confluence region



(c) Leading edge of lower flap



(d) Trailing edge of main airfoil

Fig. 6. Grid for viscous treatment of Augmentor-Wing, no blowing.

There are 25 points in the direction "normal" to the surface in the grids conforming to the flap and shroud, and 96 points in the tangential direction, including the wake. Similarly, the grid conforming to the main airfoil element is 37 by 150 points. The far-field grid is 37 by 142 points, summing to a total of 15,604 points for the entire grid.

This grid was used in calculating the results presented by Lasinski et al.[10] for Re = 12.5 × 10^6 (based on the main airfoil as unit chord), M_∞ = 0.7, and an angle of attack of 1.0°. This solution required approximately 16 hr of CPU time on a CDC 7600 computer. Generating the grid required 93 sec of 7600 CPU time. GRAPE has a feature that will accelerate its convergence by more than an order of magnitude, if for each zone the maximum number of points in both directions is of the form 3n + 1 for n an integer. However, because of the need for points to line up across zonal boundaries in this problem, that condition is not met in three of the four zones.

A five-zone grid (a region of which is shown in Fig. 7) for the viscous study with blowing has been generated and is presently in use on a CRAY-1S computer. A grid for the two-zone full-potential case has been generated, and is illustrated in Figure 8.

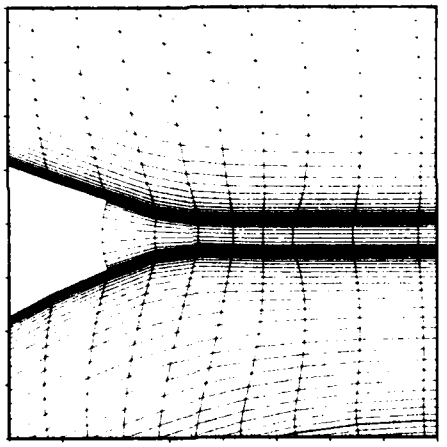
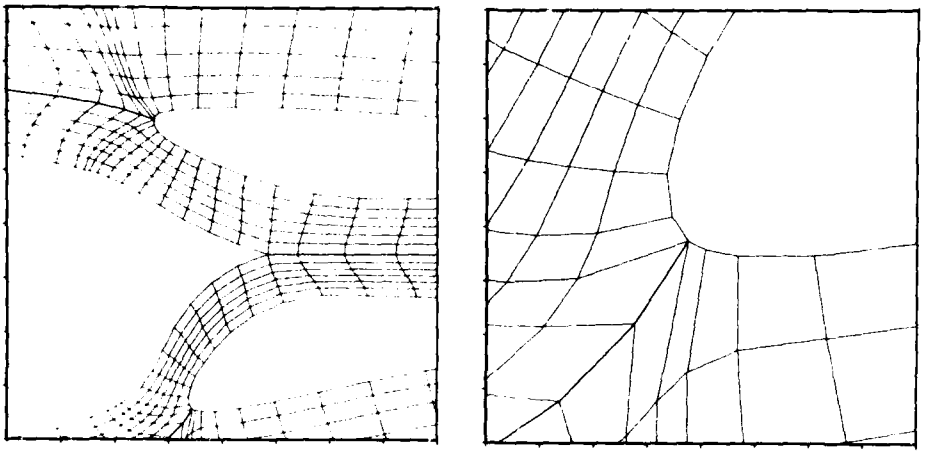


Fig. 7. Trailing edge of main airfoil, with blowing.

CONCLUSIONS

Computational grids for a complicated, multielement, airfoil shape have been successfully generated using the GRAPE elliptic partial differential equation grid-generator program. It has been demonstrated that the approach of dividing a complicated flow region into an arbitrary number of zones and ensuring continuity of grid lines, as well as their slopes and point distributions, across the zonal boundaries is practical and effective.



(a) Confluence region (b) Leading edge of lower flap

Fig. 8. Grid for full-potential treatment of Augmentor-Wing.

REFERENCES

1. Whittley, D.C. (1975) Augmentor-Wing Technology for STOL Transport Aircraft. U. of Tennessee Space Institute, Tullahoma, Tenn., Oct. 27-31.
2. Quigley, H.C., Innis, R.C., and Grossmith, S. (1974) A Flight Investigation of the STOL Characteristics of an Augmented Jet Flap STOL Research Aircraft. NASA TM-X 62334.
3. Whittley, D.C., and Cook, W.L. (1975) Comparison of Model and Flight Test Data for an Augmentor-Wing STOL Research Aircraft. AGARD Flight Mechanics Symposium, Valloire, France, June 9-12.
4. Farbridge, J.E. (1977) The Transonic Multi-Foil Augmentor-Wing. AIAA Paper 77-606, AIAA/NASA AMES V/STOL Conference, Palo Alto, Calif., June 6-8.
5. The De Havilland Aircraft of Canada, Limited (1975) Analysis of Results From Test of an Asymmetric High Speed Augmentor-Wing Model (WTCB) in the NAE Two-Dimensional High Reynolds Number Blowdown Tunnel. DHC-DIR 75-2.
6. Elfstrom, G.M. (1977) Tests on a 15" Chord Cruise Augmentor Model (WTCC) in the NAE 15" × 60" Test Facility. National Research Council Canada Laboratory Technical Report LTR-HA-5X5/0100, Mar.
7. Steger, J.L. and Sorenson, R.L. (1979) Automatic Mesh-Point Clustering Near a Boundary in Grid Generation with Elliptic Partial Differential Equations. J. Comp. Phys., 33, 3, Dec.
8. Sorenson, R.L. (1980) A Computer Program to Generate Two-Dimensional Grids about Airfoils and Other Shapes by the Use of Poisson's Equation. NASA TM-81198.
9. Thompson, J.F., Thames, F.C., and Mastin, C.W. (1974) Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies. J. Comp. Phys., 15, 3, July, pp. 299-319.
10. Lasinski, T.A., Andrews, A.E., Sorenson, R.L., Chaussee, D.S., Pulliam, T.H., and Kutler, P. (1982) Computation of the Steady Viscous Flow Over a Tri-Element "Augmentor-Wing" Airfoil. AIAA Paper 82-0021, Orlando, Fla., Jan. 11-14.

11. Steger, J.L. (1978) Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries. AIAA J., 16, 7, July, pp. 679-686.
12. Andrews, A.E., Lasinski, T.A., Sorenson, R.L., and Chaussee, D.S. (1983) Computation of the Steady Viscous Flow Over a Tri-Element "Augmentor-Wing" Airfoil with Blowing. Paper in preparation for AIAA 21st Aerospace Science Conference, Reno, Nev., Jan. 10-12.
13. Flores, J. (1982) Private Communication. Ames Research Center, NASA, Moffett Field, Calif., Feb.
14. Holst, T.L. (1979) Implicit Algorithm for the Conservative Transonic Full-Potential Equation Using an Arbitrary Mesh. AIAA J., 17, 10, Oct., pp. 1038-1045.
15. Johnson, M.S. (1981) Augmentor-Wing Grid Adaption Program (ADAPT). Unpublished report, Sept.
16. Nakamura, S.I. and Holst, T.L. (1981) A New Solution-Adaptive Grid Generation Method for Transonic Airfoil Flow Calculations. NASA TM-81330.

AD P000996

Published 1982 by Elsevier
NUMERICAL GRID GENERATION
Joe F. Thompson, editor
Not copyrighted. Unrestricted free use is granted by Lockheed Missiles & Space Company, Inc.

NUMERICAL GENERATION OF COMPOSITE THREE DIMENSIONAL GRIDS BY QUASILINEAR ELLIPTIC SYSTEMS*

P. D. THOMAS†
†Staff Scientist, Applied Mechanics Laboratory, Lockheed Palo Alto Research Laboratory, Lockheed Missiles and Space Co., Inc., 3251 Hanover St., Palo Alto, California

ABSTRACT

A technique is presented for constructing a boundary-conforming grid throughout a general three-dimensional flow region as a composite of subregion grids. Each subregion grid is generated numerically by solving a quasilinear system of elliptic equations. The boundary values represent nodal points in a quasi-two-dimensional grid that covers the curved surface bounding the subregion, and are generated numerically by a modified elliptic system. The boundary values are used to compute grid control parameters that are contained in the elliptic systems. This provides flexible control over the distribution of grid points in the interior of the region, in that the interior grid distribution is governed by the distribution of points on the boundary as well as by the boundary's geometric shape. A primary feature of the technique is that the composite three-dimensional grid remains both continuous and smooth across the surface of juncture between any two adjoining subregions. The present paper elucidates the details of the method and of its implementation, and displays numerical results for both surface grids and space grids. Comprehensive results are displayed for a three-dimensional grid about a simple wing-body combination. A numerical example is presented of a surface grid on a NACA 0012 airfoil, with high resolution of the wingtip region.

I. INTRODUCTION

To simulate complex three-dimensional fluid dynamic phenomena by a direct numerical solution of the partial differential equations that govern the flow, one first must have a spatial grid that covers the flow region. The grid defines each of the nodal points or spatial cells for which the equations are to be represented by a discrete approximation.

The three-dimensional grid generation technique described below is designed to construct numerically a boundary-conforming grid within a three-dimensional region R bounded by a closed surface. An example is shown in Figure 1 of a quasi-rectangular region bounded by six surface segments represented in a Cartesian coordinate system. A boundary-conforming grid is one in which grid points are distributed smoothly throughout the interior of the region R and along its bounding surface. For a region such as that in Fig. 1, this is equivalent to defining a new curvilinear coordinate system ξ,η,ζ such that each

* Work sponsored by the NASA Ames Research Center and the Lockheed Independent Research Program

PREVIOUS PAGE IS BLANK

of the six boundary surface segments is a coordinate surface ξ=const., η=const., or ζ=constant. When plotted in such a coordinate system, the region R takes the form of a simple rectangular solid that may be scaled so that each edge is of unit length. This forms a cube as illustrated in Fig. 2.

The problem of constructing a boundary-conforming grid thus may be viewed as the search for a transformation

$$\begin{aligned} x,y,z \quad &\text{-----}> \quad \xi,\ \eta,\ \zeta \\ \xi &= \xi(x,y,z) \\ \eta &= \eta(x,y,z) \qquad (1) \\ \zeta &= \zeta(x,y,z) \end{aligned}$$

that maps a general region R onto the unit cube in ξ, η, ζ coordinates. If the mapping functions (1) were known, then it would be easy to construct a grid in R by first constructing a grid in the cube of Fig. 2. The image of this grid in R then could be computed directly by solving Eq's (1) for x,y,z; that is, by inverting the mapping transformation. It is easy to construct a uniform grid in the cube as

$$\xi_j = (j-1)\Delta\xi \quad , \quad 1 \leq j \leq J \qquad (2a)$$

$$\eta_k = (k-1)\Delta\eta \quad , \quad 1 \leq k \leq K \qquad (2b)$$

$$\zeta_l = (l-1)\Delta r \quad , \quad 1 \leq l \leq L \qquad (2c)$$

where we have chosen

$$\Delta\xi = 1/(J-1),\ \Delta\eta = 1/(K-1),\ \Delta\zeta = 1/(L-1)$$

A general class of grid generation techniques has evolved in which the transformation functions in (1) are taken to be the solutions of an elliptic system of partial differential equations[1,2,3]. The grid in the physical region R that is the image of the uniform grid (2) is generated numerically by solving an elliptic boundary value problem for the cube. For a simply-connected region R such as that in Fig. 1, the boundary values represent the Cartesian coordinates of nodal points in a quasi-two-dimensional grid that covers the boundary surface of R. These points may be distributed non-uniformly in any fashion one may desire to resolve local features of the surface shape, such as regions of high curvature, etc. The main problem with these elliptic grid generation techniques in general is that the form of the elliptic equations dictates the positions of the grid points in the interior of R and the spacing between points. This makes it difficult to control the interior grid to achieve the flexibility of spatial resolution that is needed to solve practical flow problems by finite-difference or finite-volume methods.

The present technique overcomes this deficiency by using a special system of elliptic partial differential equations that contains information about the boundary values[1]. The distribution of grid points throughout the region R then is controlled by the distribution of points assigned on the boundary surface as well as by the geometric shape of that surface. This is in contrast to methods which rely on ad hoc "grid control functions" that are introduced into the elliptic equation system, and that usually must be individually tailored to the peculiarities of each problem[2,3].

II. THREE-DIMENSIONAL GRIDS FOR SIMPLY-CONNECTED REGIONS

In the present technique, the mapping functions (1) are taken to be solutions to the following system of quasilinear elliptic equations

$$\begin{aligned} \nabla^2\xi &= \phi(\xi,\eta,\zeta)|\nabla\xi|^2 \\ \nabla^2\eta &= \psi(\xi,\eta,\zeta)|\nabla\eta|^2 \\ \nabla^2\zeta &= \omega(\xi,\eta,\zeta)|\nabla\zeta|^2 \end{aligned} \tag{3}$$

where ϕ, ψ, and ω are universal grid control parameters whose use will be explained shortly. The equations that govern the inverse transformation are obtained from (3) by interchanging the roles of dependent and independent variables[2]. This yields an elliptic system of quasilinear equations that can be written in the vector form

$$\alpha_1(\vec{r}_{\xi\xi}+\phi\vec{r}_\xi) + \alpha_2(\vec{r}_{\eta\eta}+\psi\vec{r}_\eta) + \alpha_3(\vec{r}_{\zeta\zeta}+\omega\vec{r}_\zeta)+2(\beta_1\vec{r}_{\xi\eta} + \beta_2\vec{r}_{\eta\zeta} + \beta_3\vec{r}_{\zeta\xi}) = 0 \tag{4a}$$

$$\vec{r} = (x,y,z) \tag{4b}$$

$$\begin{aligned} \alpha_1 &= J_3^2\,(\nabla\xi\cdot\nabla\xi) & \alpha_2 &= J_3^2\,(\nabla\eta\cdot\nabla\eta) & \alpha_3 &= J_3^2\,(\nabla\zeta\cdot\nabla\zeta) \\ \beta_1 &= J_3^2\,(\nabla\xi\cdot\nabla\eta) & \beta_2 &= J_3^2\,(\nabla\eta\cdot\nabla\zeta) & \beta_3 &= J_3^2\,(\nabla\zeta\cdot\nabla\xi) \end{aligned} \tag{4c}$$

The gradients of the transformation functions in Eq's (4) are given by[4]

$$\begin{aligned} \nabla\xi &= J_3^{-1}(\vec{r}_\eta\times\vec{r}_\zeta) \\ \nabla\eta &= J_3^{-1}(\vec{r}_\zeta\times\vec{r}_\xi) \\ \nabla\zeta &= J_3^{-1}(\vec{r}_\xi\times\vec{r}_\eta) \end{aligned} \tag{5}$$

where J_3 denotes the Jacobian determinant of the inverse transformation

$$J_3 = \partial(x,y,z)/\partial(\xi,\eta,\zeta) \tag{6}$$

The coefficients α_i, β_i are scalar of functions $\vec{r}$ and it's first partial derivatives with respect to ξ, η, and ζ, and can be written in terms of the metric coefficients of the transformation simply by inserting Eq's (5) into the definitions (4c) and expanding the repeated vector and scalar products to obtain

$$\begin{aligned}
\alpha_1 &= (|\vec{r}_\eta||\vec{r}_\zeta|)^2 - (\vec{r}_\eta \cdot \vec{r}_\zeta)^2 \\
\alpha_2 &= (|\vec{r}_\zeta||\vec{r}_\xi|)^2 - (\vec{r}_\zeta \cdot \vec{r}_\xi)^2 \\
\alpha_3 &= (|\vec{r}_\xi||\vec{r}_\eta|)^2 - (\vec{r}_\xi \cdot \vec{r}_\eta)^2 \\
\beta_1 &= (\vec{r}_\eta \cdot \vec{r}_\zeta)(\vec{r}_\zeta \cdot \vec{r}_\xi) - (\vec{r}_\xi \cdot \vec{r}_\eta)|\vec{r}_\zeta|^2 \\
\beta_2 &= (\vec{r}_\zeta \cdot \vec{r}_\xi)(\vec{r}_\xi \cdot \vec{r}_\eta) - (\vec{r}_\eta \cdot \vec{r}_\zeta)|\vec{r}_\xi|^2 \\
\beta_3 &= (\vec{r}_\xi \cdot \vec{r}_\eta)(\vec{r}_\eta \cdot \vec{r}_\zeta) - (\vec{r}_\zeta \cdot \vec{r}_\xi)|\vec{r}_\eta|^2
\end{aligned} \tag{7}$$

The Cartesian coordinates (x,y,z) of grid points in a physical region R such as that of Fig. 1 are computed numerically by solving Eq's (4) on the uniform grid in the computational cube of Fig. 2, subject to Dirichlet boundary values specified on the six faces of the cube. For each such face, the boundary values are the Cartesian coordinates of nodal points in a quasi-two-dimensional grid covering that surface segment of the physical region R which maps onto the face in question. Thus, to obtain the boundary values for the three-dimensional problem, we must first construct a set of six two-dimensional grids, one for each surface segment that bounds the physical region R. We shall see later that each of these two-dimensional grids can be generated by using a special two-dimensional elliptic equation system that takes account of the shape, slope, and curvature of the surface. Once the boundary values have been obtained in this fashion for all faces of the computational cube, the control parameters ϕ, ψ, ω must be defined so that the three-dimensional grid can be generated numerically by solving the elliptic system (4).

In the present technique, the grid control parameters ϕ, ψ, and ω are evaluated in terms of the boundary values. This is accomplished as follows. First, a limiting form of the elliptic system that is valid at the boundaries is used to compute local values of the control parameters at each point on the boundaries. The parameters then are interpolated linearly into the interior to obtain a continuous representation of $\phi(\xi,\eta,\zeta)$, $\psi(\xi,\eta,\zeta)$, and $\omega(\xi,\eta,\zeta)$ throughout the computational cube. Solved numerically, the resulting elliptic system creates an interior grid that mimics not only the spatial distribution

of grid points on the boundaries but the geometric shapes of the boundaries, inasmuch as the locations of the grid points on each boundary surface reflect the shape of that surface. In essence, this technique makes partial use of ideas from both algebraic grid generation techniques and other elliptic grid generation techniques. The latter usually are based on Poisson equations[2,3] whose source terms are selected to affect the geometric behavior of the internal grid lines, whereas algebraic techniques generally use elaborate direct interpolation procedures[5] to project the surface grids into the interior of the region. In the present technique, the surface grids are used to compute local values of the grid control parameters ϕ, ψ, ω which then are projected into the interior of the computational cube by simple linear interpolation. Solution of the elliptic system containing these parameters then merely constitutes an elaborate indirect interpolation mechanism for projecting the surface grids into the interior of the region.

The first step is to evaluate the grid control parameters at the faces of the computational cube (Fig. 2). Under the mapping, each face is the image of one of the surface segments that bound the physical region (Fig. 1). Consider, for example, the upper surface $\zeta=1$. The grid control parameters can be evaluated locally in terms of boundary values, provided that constraints are imposed on the slope and curvature of the family of ζ-directed grid lines transverse to the boundary surface segment[1]. The simplest case results if this family is taken orthogonal to the boundary. The orthogonality constraint may be stated as

$$\vec{r}_\zeta \cdot \vec{r}_\xi = 0, \quad \vec{r}_\zeta \cdot \vec{r}_\eta = 0, \text{ on } \zeta=1 \tag{8}$$

which follow from the fact that the vectors $\vec{r}_\xi$, $\vec{r}_\eta$ are locally tangent to the boundary surface, whereas the vector $\vec{r}_\zeta$ is locally tangent to the transverse ζ-directed coordinate lines. The coefficients β_2, β_3 in Eq. (4) then vanish, and α_2, α_3 are correspondingly simplified [see Eq. (7)].

A further simplification results if the boundary values come from a ξ,η surface grid which itself is orthogonal, for then we have

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \tag{9}$$

Two independent uncoupled equations for evaluating the parameters ϕ,ψ at the boundary $\zeta=1$ then can be obtained by taking the scalar product of Eq. (4) with the vectors $\vec{r}_\xi, \vec{r}_\eta$, respectively, which lie in the tangent plane. That is, we take the local projection of Eq. (4) onto the ξ-directed and η-directed coordinate lines (grid lines) that comprise the known grid on the boundary

surface ζ=1. The resulting equation for ϕ is

$$\phi = -s^{(\xi)} - |\vec{r}_\xi| \left[c^{(\eta)} + c^{(\zeta)}\right] \tag{10a}$$

$$s^{(\xi)} = \vec{r}_\xi \cdot \vec{r}_{\xi\xi} / |\vec{r}_\xi|^2 \tag{10b}$$

where

$$c^{(\eta)} = (\vec{r}_\xi \cdot \vec{r}_{\eta\eta})/|\vec{r}_\xi||\vec{r}_\eta|^2 \qquad c^{(\zeta)} = (\vec{r}_\xi \cdot \vec{r}_{\zeta\zeta})/|\vec{r}_\xi||\vec{r}_\zeta|^2 \tag{10c}$$

The quantities that involve ξ and η derivatives can be evaluated directly from boundary values (i.e., from the given surface grid) using standard difference operators to replace the differential operators. However, the term $c^{(\zeta)}$ contains derivatives in the transverse ζ direction and cannot be evaluated from boundary data alone.

One can show easily that each of the terms in Eqs. (10a) has a simple physical interpretation (see Appendix). For each ξ-directed coordinate line such as PQ in Fig. 1, $s^{(\xi)}$ represents the logarithmic derivative of arc length between the known grid points along the line, whereas $c^{(\eta)}$ is proportional to the curvature of the η-directed coordinate lines in the surface grid. The term $c^{(\zeta)}$ similarly is proportional to the curvature of the ζ-directed coordinate lines transverse to the boundary surface, and may be specified arbitrarily. However, note that the available boundary data for the surface grids in the transverse boundary surface segments ξ=0 and ξ=1 can be used to evaluate $c^{(\zeta)}$ at the end points P,Q of each ξ-line in the surface ζ=1. This transverse boundary curvature information is used to control the curvature of interior ζ-lines by interpolating $c^{(\zeta)}$ between its values $c_P^{(\zeta)}$ and $c_Q^{(\zeta)}$ at the end-points P and Q as follows. If $c_P^{(\zeta)}$ and $c_Q^{(\zeta)}$ are both non-zero and have the same algebraic sign, then $c^{(\zeta)}(\xi)$ is found by interpolating the signed radius of curvature $R = 1/C$ linearly versus arc length $s = \int |\vec{r}_\xi| d\xi$. The curvature itself is linearly interpolated if the algebraic signs of $c_P^{(\zeta)}$ and $c_Q^{(\zeta)}$ are different.

An equation for ψ that is similar to Eq. (10) is obtained by taking the scalar product of Eq. (4) with $\vec{r}_\eta$. The described procedure thus allows one to evalute the grid control parameters ϕ,ψ at each point of a boundary surface segment ζ=const. when the surface grid on that segment is orthogonal.

In the more general case where the ξ,η grid on a boundary surface ζ=const. is not orthogonal, Eq's (9) and (10) no longer hold, but the same approach can be employed to evaluate the parameters ϕ,ψ. Upon taking the scalar product of Eq (4) with $\vec{r}_\xi$ and with $\vec{r}_\eta$, one obtains a pair of linear equations that can be

solved easily to obtain unique expressions for evaluating ϕ and ψ in terms of the given boundary values at faces ζ=const. of the computational cube. A similar procedure is used to evaluate all three grid control parameters ϕ,ψ,ω at all faces of the computational cube. In general, this defines each parameter at four of the six faces. One can infer from the structure of Eq. (4) that the parameter ϕ is associated primarily with the spacing of grid points along ξ-directed coordinate lines. This inference is consistent with the fact that the described procedure for evaluating ϕ from boundary values at the faces of the cube defines ϕ at only those four faces in which ξ is an interior coordinate parameter. This excludes the two faces ξ=constant. Similarly, ψ is defined at all save the two faces η=const., and ω at all save the faces ζ=constant.

Once the grid control parameters ϕ,ψ,ω have been determined at the faces of the computational cube, a continuous representation of each throughout the interior of the cube is obtained by linear interpolation. Consider, for example, the parameter ϕ which is known at the four faces ζ=0,1 and η=0,1. In each plane ξ=const., $0\leq\xi\leq1$, of the computational cube, we have a square $0\leq\eta,\zeta\leq1$ on whose perimeter ϕ is known. A continuous representation $\phi(\xi,\eta,\zeta)$ in the interior of the square is obtained from a generalized linear interpolation based on the equation

$$\phi_{\eta\eta\zeta\zeta}=0$$

which can be integrated analytically subject to the known boundary values of ϕ at the perimeter of the square. The elliptic system (4) then can be solved numerically to generate the three-dimensional grid once the parameters ϕ,ψ,ω have been defined throughout the computational cube of Fig. 2.

III. TWO-DIMENSIONAL GRIDS ON CURVED SURFACES

The boundary values for generating the three-dimensional grid are obtained by constructing a two-dimensional grid on each of the six surface segments that bound the physical region R depicted in Fig. 1. Each surface grid is generated by solving a two-dimensional Dirichlet problem governed by a two-dimensional elliptic system that is similar in form to the three-dimensional system (4), but that contains terms describing the slope and curvature of the surface. The two-dimensional elliptic system is deduced from the three-dimensional system (4) as follows[1]. The surface on which the grid is to be generated is taken as a coordinate surface ζ=constant. By requiring that the superfluous ζ-directed coordinate lines both have zero principal curvature and

be orthogonal to this surface, one obtains the simplified 3D system

$$\alpha(\vec{r}_{\xi\xi} + \phi\vec{r}_{\xi}) - 2\beta\vec{r}_{\xi\eta} + \gamma(\vec{r}_{\eta\eta} + \psi\vec{r}_{\eta}) + (\alpha_3/\delta)[\omega + (\ell n/\delta)_{\zeta}]\vec{r}_{\zeta}=0 \qquad (11a)$$

$$\alpha = |\vec{r}_{\eta}|^2 \ , \ \beta = \vec{r}_{\xi}\cdot\vec{r}_{\eta} \ , \ \gamma = |\vec{r}_{\xi}|^2 \ , \ \delta = |\vec{r}_{\zeta}|^2 \qquad (11b)$$

where

$$\vec{r}_{\zeta}\cdot\vec{r}_{\xi} = \vec{r}_{\zeta}\cdot\vec{r}_{\eta} = 0 \qquad (11c)$$

Let the surface on which a grid is desired be defined in Cartesian coordinates by the function $z=f(x,y)$, where f is single-valued and twice-differentiable. Then Eq.(11) can be manipulated to yield the following set of equations for the projection of the grid onto the x-y plane[1]

$$\alpha(x_{\xi\xi} + \phi x_{\xi}) - 2\beta x_{\xi\eta} + \gamma(x_{\eta\eta} + \psi x_{\eta}) + f_x G = 0 \qquad (12a)$$

$$\alpha(y_{\xi\xi} + \phi y_{\xi}) - 2\beta y_{\xi\eta} + \gamma(y_{\eta\eta} + \psi y_{\eta}) + f_y G = 0 \qquad (12b)$$

$$\vec{r} = (x,y,z) \quad , \quad z = f(x,y) \qquad (12c)$$

$$G = J_2^2[(1+f_y^2)f_{xx} - 2f_x f_y f_{xy} + (1+f_x^2)f_{yy}]/(1+f_x^2+f_y^2) \qquad (12d)$$

$$J_2 = \partial(x,y)/\partial(\xi,\eta) = x_{\xi}y_{\eta} - x_{\eta}y_{\xi} \qquad (12e)$$

where J_2 denotes the two-dimensional Jacobian determinant. These equations can be used to generate a boundary-conforming grid within any closed, simply-connected region that lies on a given surface $z=f(x,y)$. Consider, for example, the top surface segment ζ=const. in Fig. 1.

The surface grid is generated by solving Eq's (12) on a uniform ξ,η grid over the unit square on the face $\zeta=1$ of Fig. 2, subject to Dirichlet boundary values $\vec{r}_{jk}$ assigned along the perimeter of the square. These boundary values represent the coordinates (x,y,z) of grid points along the perimeter of the surface segment $\zeta=1$ in Fig. 1, and may be distributed along that perimeter in any fashion desired. In analogy with the three-dimensional case discussed earlier, an expression for ϕ or ψ along each segment of the boundary curve is obtained by taking the projection of Eq. (11a) onto the vector $\vec{r}_{\xi}$ or $\vec{r}_{\eta}$ tangent to that segment[1]. The parameters ϕ,ψ are then interpolated linearly into the computational square from the boundaries, and the resulting elliptic system (12) is solved numerically to generate the surface grid[1]. This requires that

boundary values (i.e., grid point locations) be assigned a priori along each of the four segments of the boundary curve. Since the spacing of grid points between the endpoints of each boundary segment can be parametrized easily in terms of a single quantity, the arc length, one need not be burdened with the task of manipulating individual grid points, but may distribute grid points along each boundary segment with the aid of a general one-dimensional stretching function such as that devised by Vinokur[6].

IV. COMPOSITE GRIDS AND MULTIPLY-CONNECTED REGIONS

The technique outlined in the preceding section provides the Dirichlet boundary values for solving Eq's (4) to generate a 3D space grid within an arbitrary spatial region such as that depicted in Figure 1. One proceeds as follows. First, the technique of Section III is used to generate a quasi-two-dimensional grid over each surface segment that bounds the region. This requires the a priori selection of a very small subset of nodal points in the three-dimensional grid, namely, those points which lie along the curves of intersection between the surface segments which bound the physical region. These are the curves that map onto the edges of the computational cube. The bounding surfaces, and hence, their curves of intersection are known. The locations of grid points along each such curve can be parametrized easily in terms of the arc length, and the points may be distributed along the curve with a general one-dimensional coordinate-stretching function[6]. Once the surface grids have been constructed, the surface grid point coordinates are used as the boundary values for the three dimensional Dirichlet problem on the cube. The grid control parameters that enter into the elliptic system are evaluated at the faces of the computational cube and are interpolated linearly into the interior. The Dirichlet problem then is solved numerically by some iterative method.

This direct approach is likely to be inadequate for geometrically complicated regions, and is invalid for multiply-connected regions. These situations always can be treated by subdividing the region into a collection of contiguous simply-connected subregions, each of which has a more or less uniform geometric character. Each subregion grid is generated independently, and then is joined with the others to form a composite grid for the original region.

An important feature inherent in the present method is that the composite grid automatically remains both continuous and smooth across the surface of juncture between any two adjacent subregions as long as the same boundary

values are used at that common surface when generating the grid for either subregion. This feature is a direct consequence of using the boundary values to evaluate the grid control parameters ϕ,ψ,ω that enter into the generating elliptic system. An example of such a composite grid is presented in the next section.

V. NUMERICAL RESULTS

Three-Dimensional Composite Grid for a Wing-Body Combination

The technique outlined in the preceding section has been applied to construct a three-dimensional grid about the simple wing-body combination depicted in Fig. 3. The configuration consists of a cylindrical fuselage with spherical end caps and a straight wing that has a super-elliptical planform. The axis of the cylinder coincides with the Cartesian y axis, the wing mid-chord line coincides with the x axis, and the wing cross-section is symmetric about the plane z=0. In the positive half-space $z \geq 0$, the wing surface is generated by the equation

$$(\varepsilon x)^n + y^n = [(\tau - z)(\tau^{-1} + z)]^{n/2}$$
$$\varepsilon = 0.2, \quad \tau = 0.25, \quad n=8$$

where τ is the thickness-to-chord ratio in the plane x=0 and ε is the chord-to-span ratio. The configuration is symmetric about each of the three coordinate planes x=0, y=0 and z=0.

The surface grids shown in Fig. 3 on the cylinder, the sphere, and the wing were generated by the method described in Section III, treating each of the three parts as an independent subregion.

To form a bounded region in which to construct a three-dimensional grid, the configuration is enclosed by an outer "freestream" boundary surface consisting of an elliptical cylinder with an end cap formed by the matching ellipsoid of revolution. Since the region is symmetric about the three coordinate planes, we consider only the positive octant $x,y,z \geq 0$, and include the planes x=0 and z=0 as boundaries. This region was subdivided into two subregions separated by the plane normal to the y axis that passes through the sphere-cylinder juncture of the body. The composite surface grids on the body, wing, and symmetry planes are displayed in Fig. 4. A three-dimensional view of each subregion is given in Figs. 5 and 6, and shows the grid on each visible surface. Figure 7 shows the composite grid formed by joining the two subregions.

In Figure 4, the surface grids on the nosecap and on the symmetry planes for the forward subregion are joined to those for the aft subregion to show that

grid lines remain smooth between adjoining subregions. This smoothness also is evident in Figs. 8 and 9, which display several interior coordinate surfaces of the composite grid.

The number of grid points in each of the two subregions depicted in Figs. 5 and 6 is 26 x 11 x 21 = 6000. For each subregion, solution of the elliptic system by successive line over-relaxation (SLOR) required approximately 150 iterations and 15 min. of CPU time on a VAX 11/780 minicomputer.

Surface Grids

The surface grid generation technique presented in Section III has been applied to a standard NACA 0012 symmetric airfoil with a wingtip formed by rotating the profile about its central symmetry line. Figs. 10-12 show plan, end, and perspective views of the grid on the upper half of the wing surface. Note the relatively high resolution of the critical region near the wingtip that is obtained with the 21 x 20 grid.

Experimentation recently has been performed with wing-body surface grids having a topological structure different from that used in Fig. 3. An example is displayed in Fig. 13 for the same spherically-capped cylindrical body joined to the blunt NACA 0012 wing instead of the sharp-edged elliptical wing. The body grid has no polar singularity at the nosetip, unlike Fig. 3, and has a topology similar to that employed in Reference 7.

ACKNOWLEDGEMENT

The author is indebted to Mrs. K. L. Meier, who programmed the equations, carried out their solution, and created the three-dimensional plots to display the results.

REFERENCES

[1]Thomas, P. D., (1981) Paper 81-0996-CP, Proc. AIAA 5th Computational Fluid Dynamics Conf., pp. 24-32.
[2]Mastin, C.W. and Thompson, J.F., (1978) Numer. Math., Vol. 29, No. 4, pp. 397-407.
[3]Lee, K. D., (1981) Paper 81-0998, Proc. AIAA 5th Computational Fluid Dynamics Conf.
[4]Thomas, P.D. and Lombard, C.K., (1979) AIAA Journal, Vol. 17, No. 10, pp. 1030-1037.
[5]Eiseman, P.R. and Smith, R.E., (1980) NASA Conference Publication 2166, pp. 73-120.
[6]Vinokur, M., (1980) NASA CR 3133
[7]Caughey, D.A. and Jameson, A., (1980) AIAA Jour., Vol. 18, No.11, pp.1281-1288.
[8]Franklin, P. (1944) Methods of Advanced Calculus, McGraw-Hill, New York, pp. 107-109.

APPENDIX

Geometric Interpretation of Terms in Equation (10)

In Equation (10a) that is used to evaluate the grid control parameter ϕ at a boundary surface, each of the terms has a simple geometric interpretation. It follows from the identity

$$\vec{r}_\xi \cdot \vec{r}_{\xi\xi} = \frac{1}{2}(|\vec{r}_\xi|^2)_\xi$$

and from the definition

$$ds = |\vec{r}_\xi| d\xi$$

of arc length s along a ξ-directed coordinate line that the term $S^{(\xi)}$ defined by Eq. (10b) can be rewritten as the logarithmic derivative of arc length

$$S^{(\xi)} = (\ell n\, s_\xi)_\xi$$

Hence, this term depends only on the spacing between grid points along the line.

The term $C^{(\zeta)}$ defined in Eq. (10c) can be interpreted in terms of the curvature of ζ-directed grid lines. The principal curvature κ of a space curve such as a ζ-directed coordinate line is defined classically so that the rate of turning of the unit tangent vector

$$\vec{t} = \vec{r}_\zeta / |\vec{r}_\zeta|$$

with respect to arc length $ds = |\vec{r}_\zeta| d_\zeta$ is given by[8]

$$\frac{d\vec{t}}{ds} = \kappa \vec{p}$$

where the unit vector $\vec{p}$ is orthogonal to $\vec{t}$. Upon performing the indicated differentiation, the result can be cast in the form

$$\vec{r}_{\zeta\zeta} = |\vec{r}_\zeta|^2 \kappa \vec{p} + (\ell n\, s_\zeta)_\zeta \vec{r}_\zeta$$

If one takes the scalar product of this equation with the unit vector tangent to the ξ-directed coordinate lines, the second term drops out because of the orthogonality condition (8), and the term $C^{(\zeta)}$ is found to be proportional to the curvature, κ.

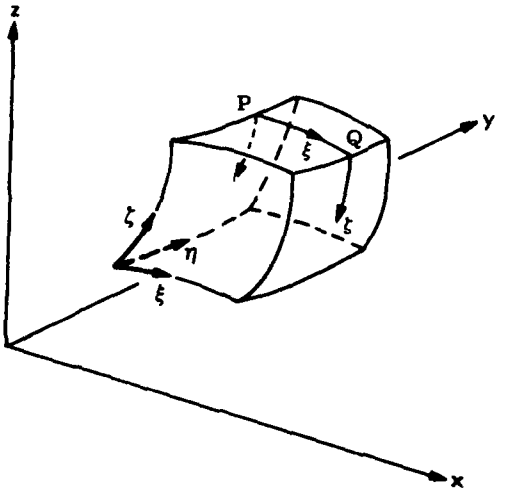
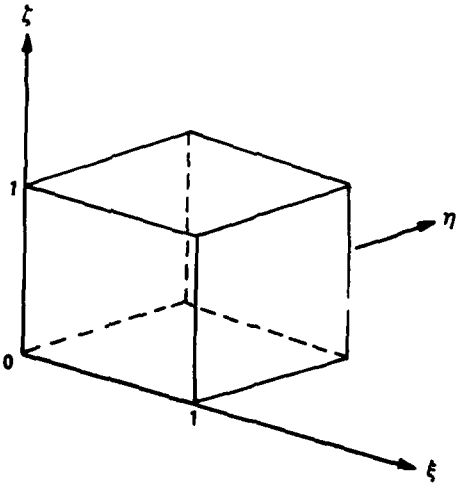


Fig. 1. Physical region R.



$\xi = \xi(x,y,z)$
$\eta = \eta(x,y,z)$
$\zeta = \zeta(x,y,z)$

Fig. 2. Physical region R mapped onto a cube.

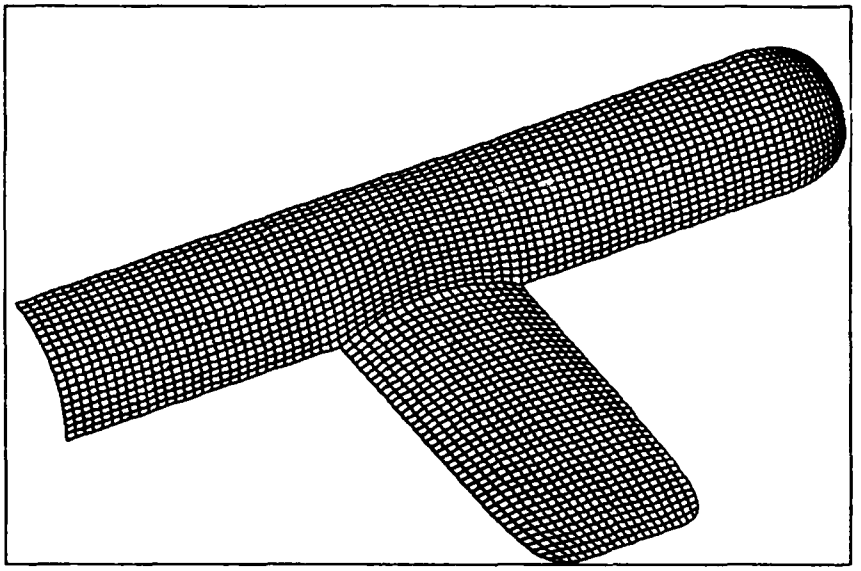


Fig. 3. View of wing body combination showing surface grids.

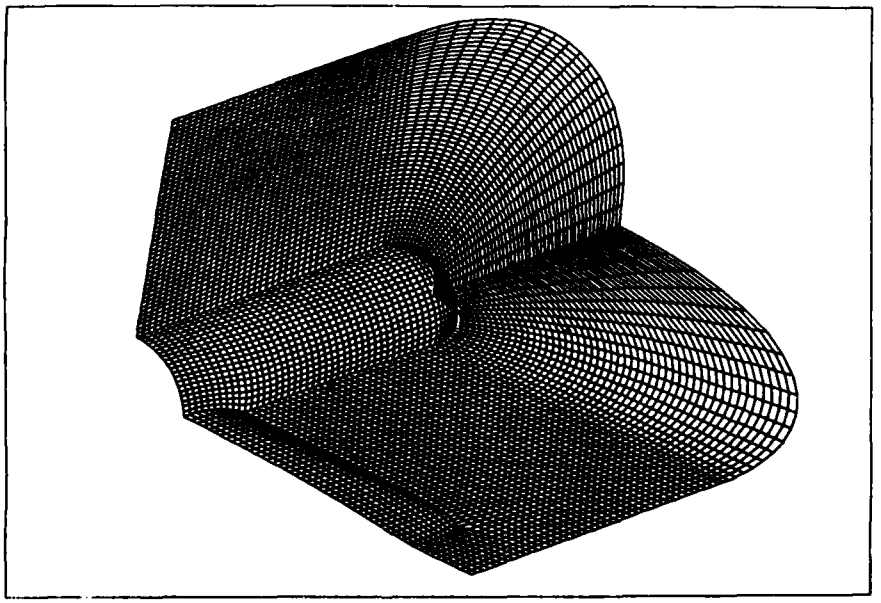


Fig. 4. Composite surface grids on wing, body, and symmetry planes.

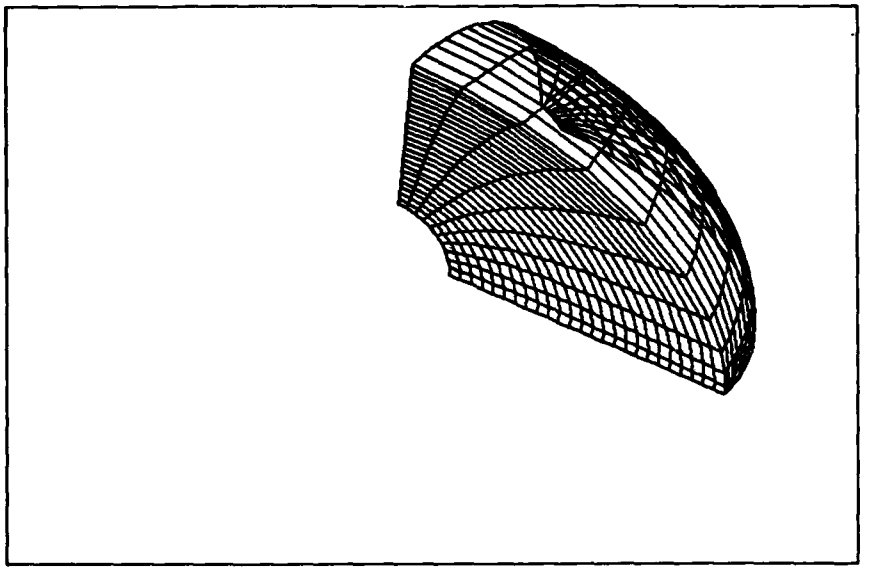


Fig. 5. Outer surfaces of forward subregion grid.

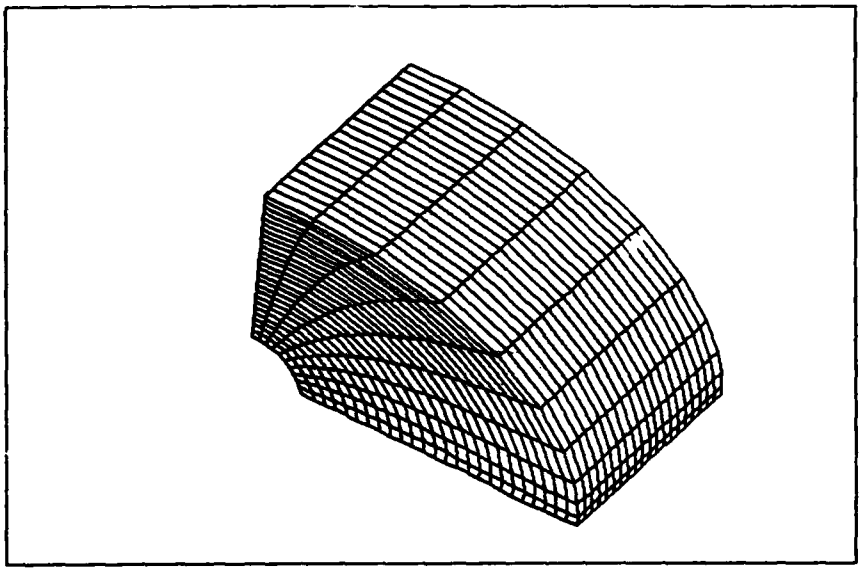


Fig. 6. Outer surfaces of aft subregion grid.

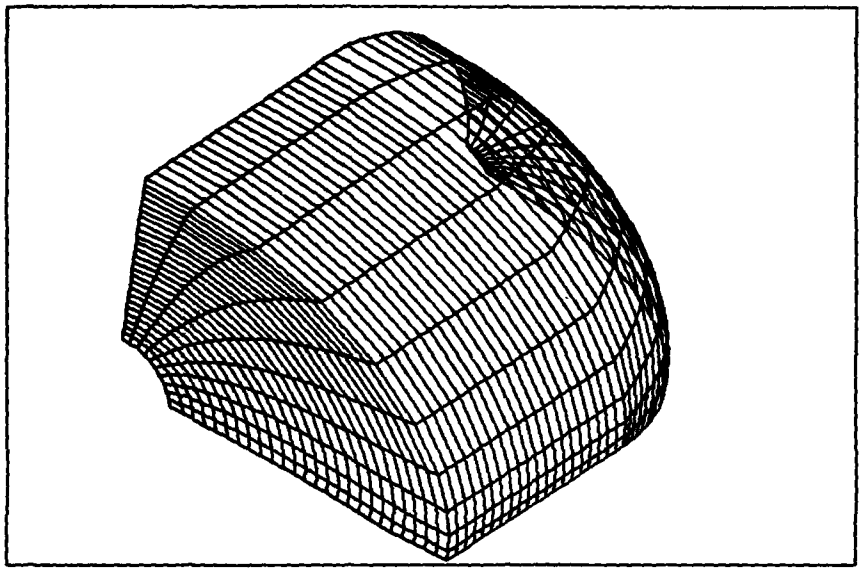


Fig. 7. Outer surfaces of composite grid.

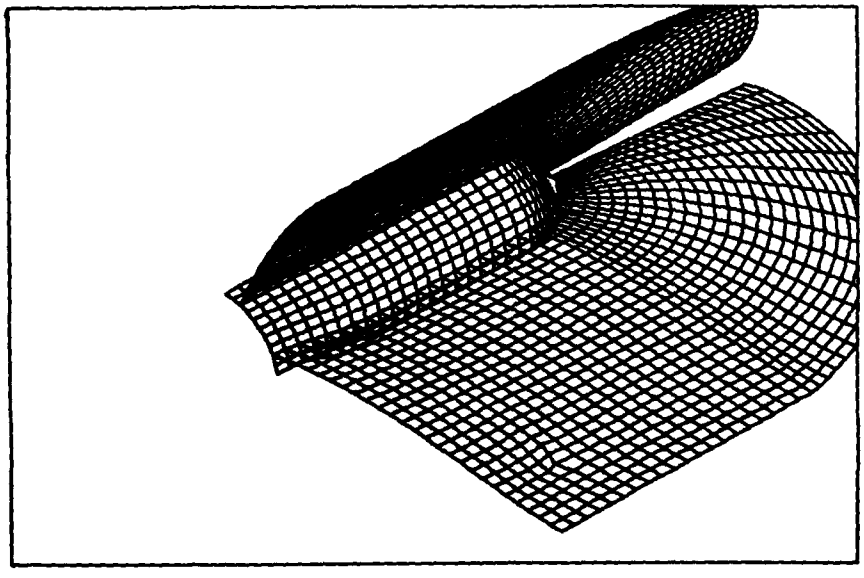


Fig. 8. View of composite 3D grid showing grid lines on body surface and on two interior "body-normal" coordinate surfaces.

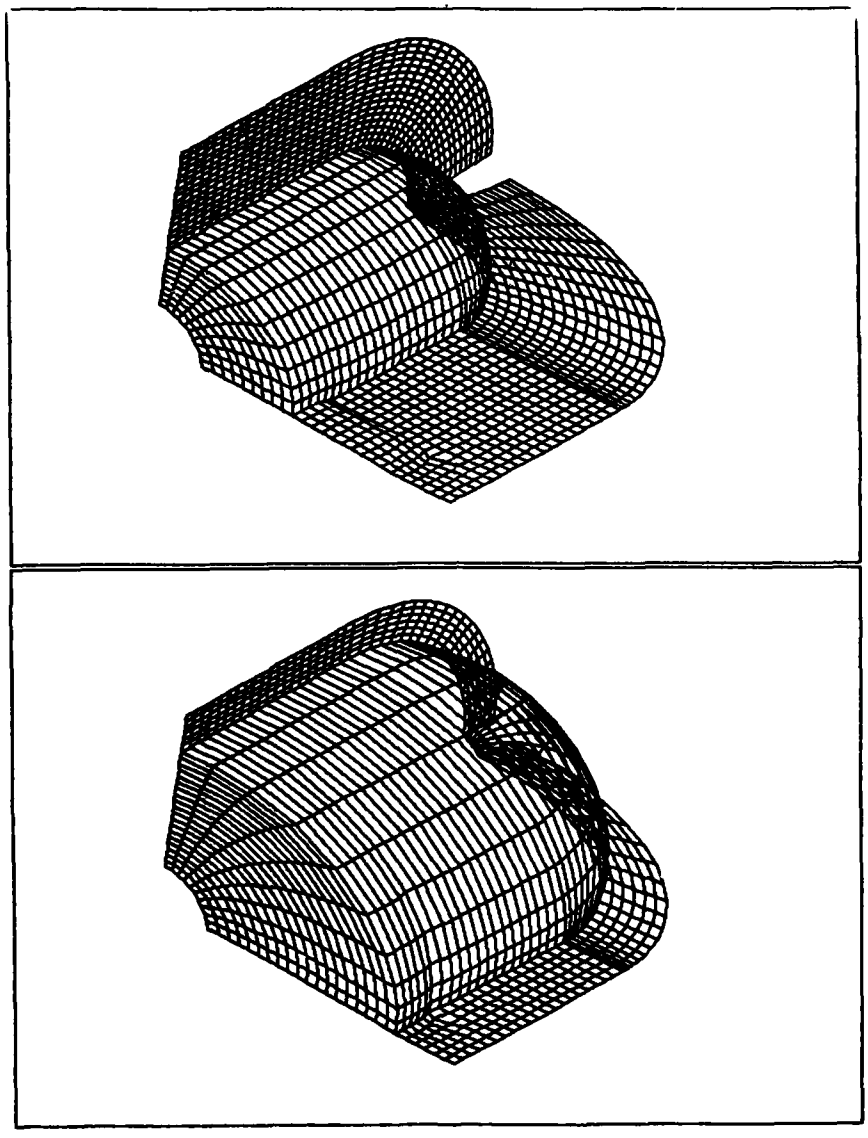


Fig. 9. Views of composite 3D grid showing grid lines on interior "body-like" coordinate surfaces.

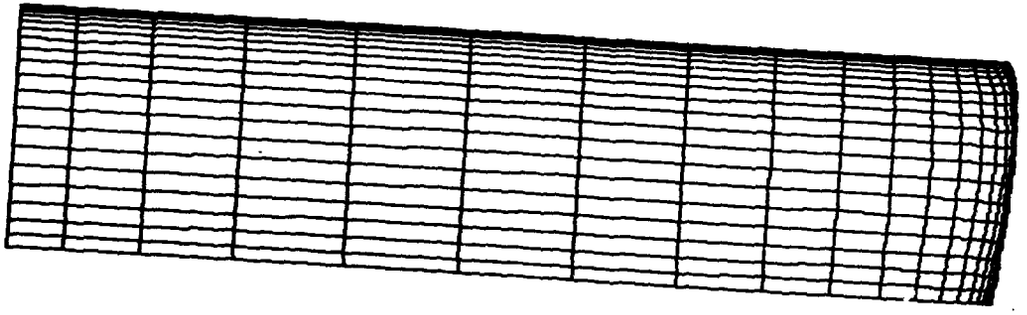


Fig. 10. Plan view of NACA 0012 airfoil surface grid.

Fig. 11. End view looking toward wingtip of NACA 0012 airfoil surface grid.

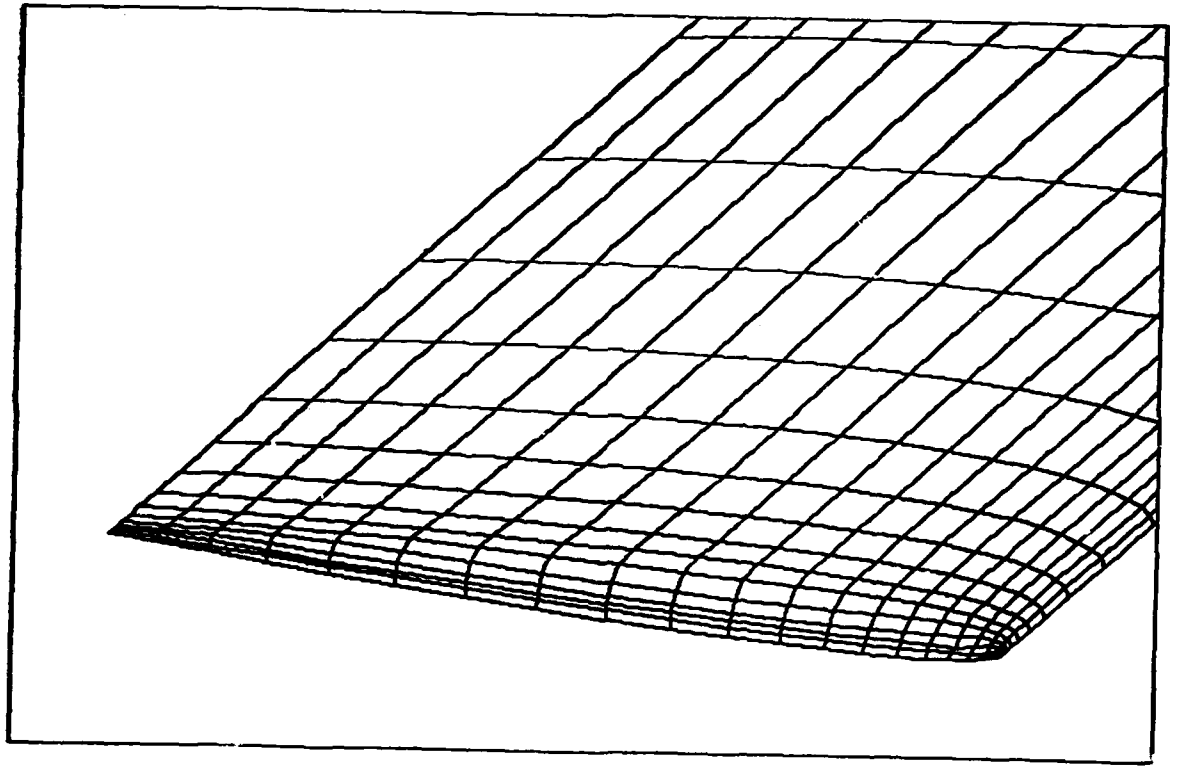


Fig. 12. Perspective view of grid on NACA 0012 airfoil upper surface near wingtip.

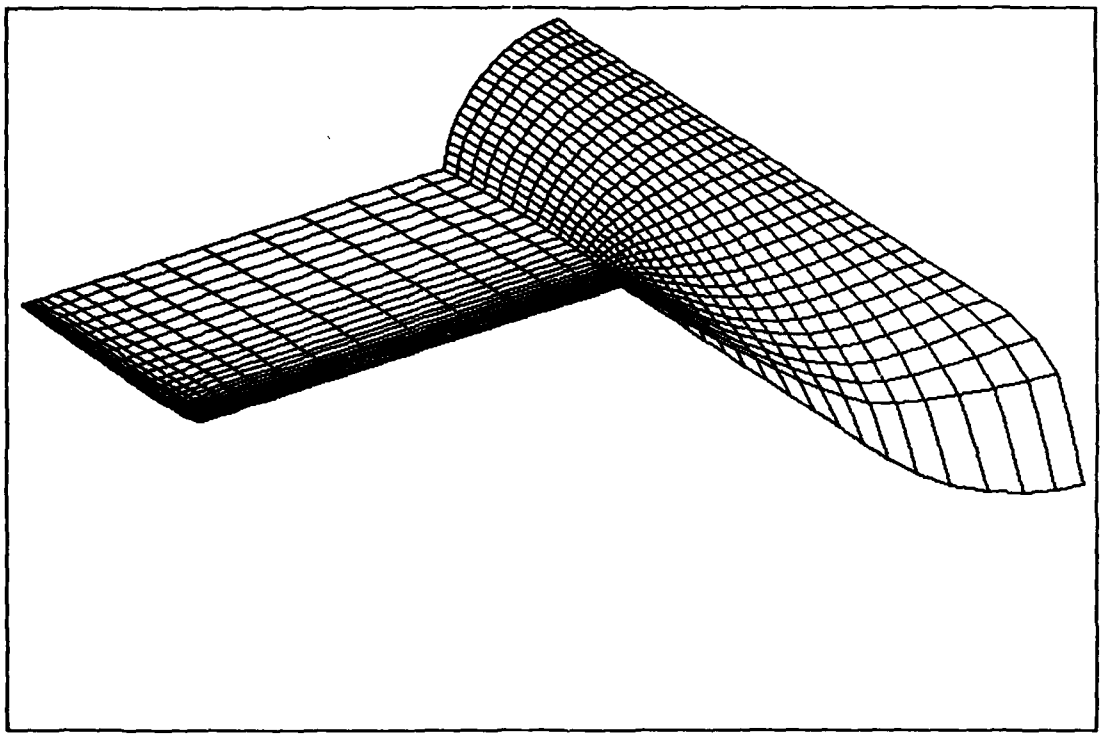


Fig. 13. Alternative surface grid topology for a body with a NACA 0012 wing.

AD P000997

Published 1982 by Elsevier
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

THREE-DIMENSIONAL GRID GENERATION USING POISSON EQUATIONS*

C. F. SHIEH
Calspan Field Services, Inc., AEDC Division
Arnold Air Force Station, Tennessee 37389

SUMMARY

A method for constructing a boundary-fitted grid system for arbitrary three-dimensional (3-D) configurations is presented. The grid generation procedure is formulated through the Poisson equations. A feature of the scheme is that grid lines in the grid system are smoothly interconnected between the boundary coordinates. The control of the grid distributions in the interior of the system has been implemented by the forcing functions appearing in the Poisson equations and by the boundary coordinates. Reasonable cell sizes and shapes are obtained even for extreme boundary shapes which normally tend to cause poorly controlled grid distributions.

INTRODUCTION

For the numerical study of fluid dynamics problems, the generation of computational grids is generally required in the finite difference or the finite element solutions of the governing flow equations. A poorly constructed grid system may cause erroneous results and/or bring about slow numerical convergence. The grid generation technique is, therefore, a critical element for supporting numerical flow simulation, especially for three-dimensional flow-field computations. The purpose of this paper is to present a method for constructing a boundary-fitted grid system suitable for the solution of fluid dynamics problems associated with complex 3-D configurations.

In recent years, the creation of a grid system for arbitrary flow regions has been approached by several ways including conformal mapping, algebraic construction, and partial differential equations solutions.[1] Among these methods, an extensively used scheme for grid generation uses the Poisson equations and has been applied successfully to various two-dimensional (2-D) geometries.[2-4] In this method, the Cartesian coordinates of the grid points in the physical plane

---

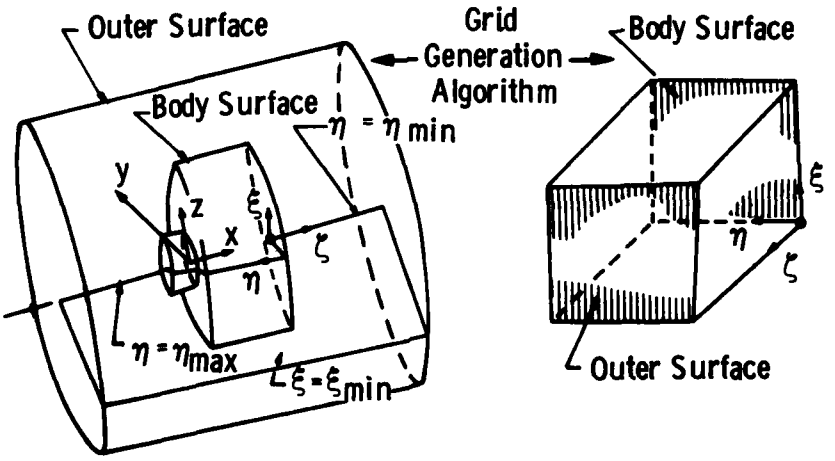
*The research reported herein was performed by the Arnold Engineering Development Center, Air Force Systems Command. Work and analysis for this research were done by personnel of Calspan Field Services, Inc., AEDC Division, operating contractor of the flight dynamics test facilities at AEDC. Further reproduction is authorized to satisfy needs of the U. S. Government.

are computed as the solutions to the grid generation system formulated by the Poisson equations. Because of its potential for creating smoothly connected grid lines between the specified boundaries, this grid generation scheme has been adapted and extended in the present study for 3-D applications.

In previous studies,[5,6] a similar grid generation technique is employed for wing/body configurations using multiple block structures in the transformed plane. In the present approach, however, a single block structure is utilized and control of the interior grid distributions is implemented by using both the specified forcing functions incorporated in the elliptic grid generation scheme and the specified boundary coordinates. The forcing functions are used for the stretching of coordinate lines toward other coordinate lines. The grid spacing distribution, however, is specified by the boundary coordinates. Reasonable cell sizes and shapes are obtained even for extreme boundary shapes which normally tend to cause poorly controlled grid-point distributions.

GRID GENERATION PROCEDURE

In the physical x,y,z-space, the region under consideration for grid generation is bounded by a body surface (inner surface) and a surface surrounding the body (outer surface) (Fig. 1a). In transformed ξ,η,ζ-space, the region appears as a single block as shown in Fig. 1b.



a. x,y,z-space b. ξ,η,ζ-space

Fig. 1. Illustration of the region wherein the grid points are generated.

The system utilized for generating boundary-fitted grids is based on the approach of Thompson, et al.[2] The transformation of the grid coordinates from the Cartesian space, i.e., x,y,z-space, satisfies the following Poisson equations:

$$\begin{aligned} \xi_{xx} + \xi_{yy} + \xi_{zz} &= P\ (\xi,\eta,\zeta) \\ \eta_{xx} + \eta_{yy} + \eta_{zz} &= Q\ (\xi,\eta,\zeta) \\ \zeta_{xx} + \zeta_{yy} + \zeta_{zz} &= R\ (\xi,\eta,\zeta) \end{aligned} \tag{1}$$

By interchanging the dependent (ξ,η,ζ) and independent (x,y,z) variables in Eq. (1), the following nonlinear equation is formed:

$$\begin{aligned} &a_{11}\ (x_{\xi\xi} + \phi x_{\xi}) + a_{22}\ (x_{\eta\eta} + \psi x_{\eta}) + a_{33}\ (x_{\zeta\zeta} + \Lambda x_{\zeta}) \\ &+ 2\ (a_{12}\ x_{\xi\eta} + a_{13}\ x_{\xi\zeta} + a_{23}\ x_{\eta\zeta}) = 0 \end{aligned} \tag{2}$$

where

$$a_{ij} = \sum_{m=1}^{3} A_{mi} A_{mj}$$

and A_{mi} *is the cofactor of* the (m,i) element in the following matrix

$$M = \begin{bmatrix} x_{\xi} & x_{\eta} & x_{\zeta} \\ y_{\xi} & y_{\eta} & y_{\zeta} \\ z_{\xi} & z_{\eta} & z_{\zeta} \end{bmatrix}$$

The equations for y and z are the same as Eq. (2) with the variable x replaced by y and z, respectively.

The variables ϕ, ψ, and Λ in Eq. (2) provide a means for controlling the interior grid distribution.[3]

$$\begin{aligned} \phi &= \frac{J^2 P}{a_{11}} \\ \psi &= \frac{J^2 Q}{a_{22}} \\ \Lambda &= \frac{J^2 R}{a_{33}} \end{aligned} \tag{3}$$

where

$$J = \frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)} = \begin{vmatrix} x_{\xi} & x_{\eta} & x_{\zeta} \\ y_{\xi} & y_{\eta} & y_{\zeta} \\ z_{\xi} & z_{\eta} & z_{\zeta} \end{vmatrix}$$

In the present study, a central-difference numerical scheme is applied to Eq. (2). The finite difference form of Eq. (2) is then solved by an alternate direction implicit (ADI) method with the coordinates at the boundaries and the functions ϕ, ψ, and Λ being specified. In the iterative procedure, the functions ϕ, ψ, and Λ are treated as local constants. The method for selecting these functions is discussed in the next section.

GRID SPACING CONTROL

To solve Eq. (2), the physical coordinates of the boundary-grid points are specified with user-desired grid spacing. However, the control of the interior grid spacing, which is governed primarily by Eq. (2), depends on the choice of the forcing functions, ϕ, ψ, and Λ. One can verify that the following exponential form is a solution of Eq. (2) with the parameters ϕ, ψ, and Λ constant:

$$\vec{x} = c_1 e^{-\phi\xi} + c_2 e^{-\psi\eta} + c_3 e^{-\Lambda\zeta}$$

where c's are constants and

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

For positive values of ϕ, ψ, and Λ, the magnitude of the gradients of $\vec{x}$ with respect to ξ, η, or ζ are reduced as ξ, η, or ζ approach $\xi = \xi_{max}$, $\eta = \eta_{max}$, or $\zeta = \zeta_{max}$, respectively. Therefore, the grid spacing is reduced accordingly. However, the grid spacing becomes larger as ξ, η, and Λ approach their maximum values for negative constants ϕ, ψ, and Λ. In order to illustrate these results, the grid networks generated for a multicylinder case (Fig. 1a) are shown in Fig. 2. Comparing Fig. 2a, i. e., $\Lambda = 0$, with Fig. 2b, i. e., $\Lambda = \sin\left(\frac{\zeta_N - \zeta}{\zeta_N - \zeta_o} 2\pi\right)$, where $\zeta_N = \zeta_{max}$ and $\zeta_o = \zeta_{min}$, it is seen that positive values of Λ tend to stretch the ζ = constant plane toward the outer boundary. For negative Λ, however, the grid lines are attracted toward the inner boundary (i. e., the body surface). Based on these observations, the following exponential form for the forcing functions was selected:

$$\phi = a_1 \frac{\xi_c - \xi}{\xi_N - \xi_o} \exp\left[- b_1 \frac{(\xi - \xi_c)^2}{(\xi_N - \xi)(\xi - \xi_o)} \right] \tag{3a}$$

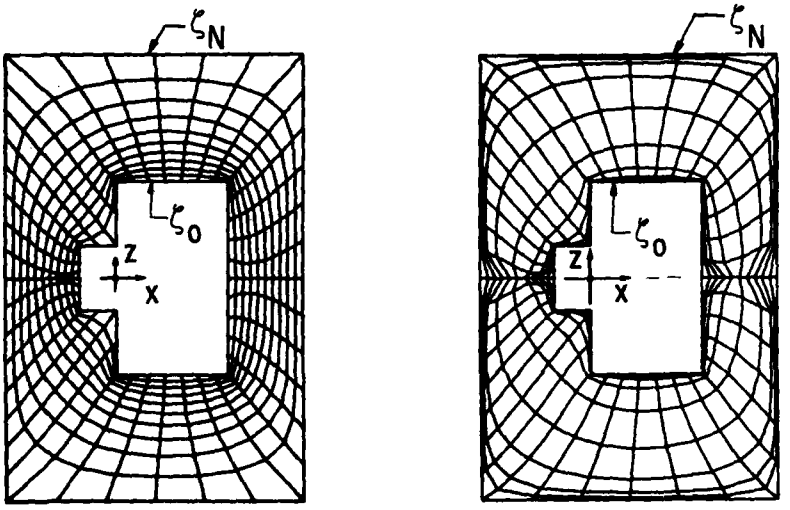
$$\psi = a_2 f \frac{\eta_c - \eta}{\eta_N - \eta_o} \exp\left[- b_2 \frac{(\eta - \eta_c)^2}{(\eta_N - \eta)(\eta - \eta_o)} \right] \tag{3b}$$

$$\Lambda = a_3 \frac{\zeta_c - \zeta}{\zeta_N - \zeta_o} \exp\left[- b_3 \frac{(\zeta - \zeta_c)^2}{(\zeta_N - \zeta)\ (\zeta - \zeta_o)} \right] \tag{3c}$$

where the subscripts "N" and "o" denote the "max" and "min", respectively. The subscript "c" denotes a control station between "N" and "o". In Eq. (3), the value of the adjustable parameters b, which controls the decay rate of the exponential function, is always positive. The adjustable amplitude factor, a's, however, may be positive or negative. Positive a's tend to stretch the grid lines toward the control station (i.e., $\xi = \xi_c$, $\eta = \eta_c$, or $\zeta = \zeta_c$). Negative a's, however, tend to stretch the grid lines toward either inner or outer boundaries. The function f of Eq. (3b) provides an additional control of the decay rate of ψ in the ζ direction.

$$f = \frac{\eta_N - \eta_o}{|\eta - \eta_c|} \exp\left[- b_4 \frac{(\zeta - \zeta_c)^2}{(\zeta_N - \zeta)\ (\zeta - \zeta_o)} \right] \tag{4}$$

where b_4 is a positive number. The value of f is set to zero for $\eta = \eta_c$. Application of Eq. (4) causes $\psi = 0$ at $\zeta = \zeta_N$ or $\zeta = \zeta_o$.

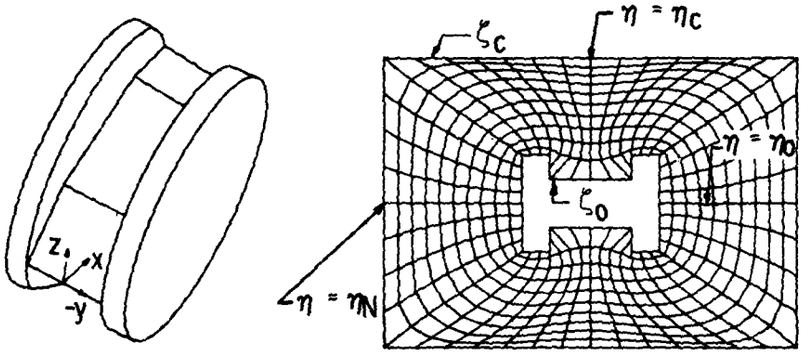


a. $\Lambda = 0$ b. $\Lambda = \sin\left(\frac{\zeta_N - \zeta}{\zeta_N - \zeta_o} 2\pi\right)$

Fig. 2. Grid network generated with $\phi = \psi = 0$ for a multicylinder as shown in Fig. 1a.

Figure 3 shows an example of the application of Eq. (3) to grid generation for a body with an H-shaped cross section. The grid network shown in Fig. 3

was obtained for the case $a_1 = 0$, $a_2 = a_3 = 0.8$, $b_2 = b_3 = 1.0$, and $f = 1.0$. The grid network generated for the same body with $\phi = \psi = \Lambda = 0$ is shown in Fig. 4. The effective grid spacing control using Eq. (3) is clearly seen if one compares Figs. 3 and 4. However, refinement of the grid spacing and cell shapes of the corners and in the neighborhood of the inner boundary is necessary.



a. H-shaped body geometry

b. Grid network generated by specifying $a_1 = 0$, $a_2 = a_3 = 0.8$, $b_2 = b_3 = 1.0$, and $f = 1.0$

Fig. 3. Grid network for an H-shaped body.

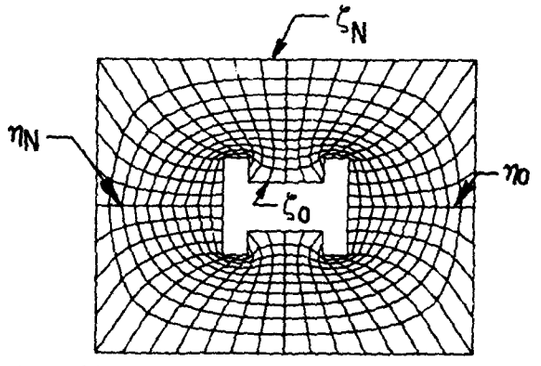


Fig. 4. Grid network generated with $\phi = \psi = \Lambda = 0$ for an H-shaped body (Fig. 3a).

A method for controlling grid spacing in the ζ-direction was also developed and is accomplished by specified boundary coordinates at the $\xi = \xi_o$ plane shown in Fig. 5. Control of grid spacing is implemented by requiring that the ratio of arc length, S, to the total length, S_t, along the grid lines in the

ζ-direction equals the ratio at the $\xi = \xi_o$ plane. By definition,

$$\varepsilon \equiv \frac{s}{s_t} - \frac{s_o}{s_{t_o}} \to 0 \tag{5}$$

This condition is satisfied by an iterative method.

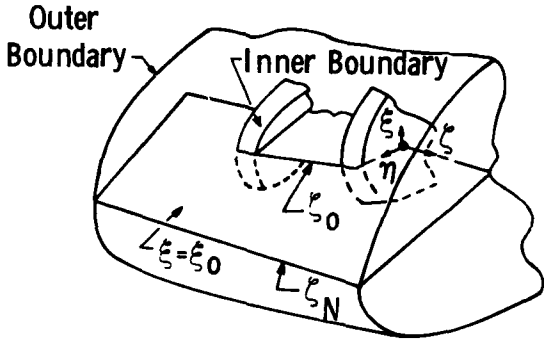


Fig. 5. Definition of the coordinate system ξ, η, and ζ for an H-configuration (Fig. 3a).

As the grid generation proceeds, the values of ϕ and ψ, which are specified in an exponential form such as in Eq. (3), remain unchanged on each iteration level. The values of Λ, however, are adjusted by

$$\Delta\Lambda = -\frac{\varepsilon}{\frac{\partial \varepsilon}{\partial \Lambda}} \tag{6}$$

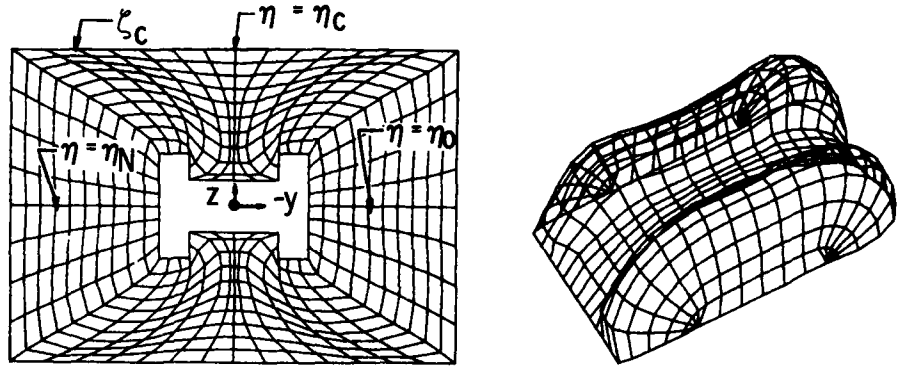
Grid computation is conducted in two iterative steps. First, the coordinates are calculated with specified forcing functions. Then the Λ function is updated using

$$\Lambda^{n+1} = \Lambda^n + \Delta\Lambda \tag{7}$$

where the superscript "n" denotes the number of iterations and the value of $\Delta\Lambda$ is approximated by

$$\Delta\Lambda = \frac{-\varepsilon^n}{|\varepsilon^n|}\left(\frac{\varepsilon^n}{\varepsilon^n - \varepsilon^{n-1}}\right)\left(\Lambda^n - \Lambda^{n-1}\right) \tag{8}$$

The effectiveness of this grid control procedure for the H-shaped body as shown in Fig. 4a is demonstrated in Fig. 6. The grid network shown in Fig. 6 is generated by specifying $a_1 = 0$, $a_2 = a_3 = 0.8$, $b_2 = b_3 = 1.0$, and $f = 1.0$. The value of Λ is obtained from Eq. (7) at each iteration level.



a. An example of the generated grid network

b. Grid network at the plane next to the H-shaped body (only the upper half surface is shown)

Fig. 6. Grid network generated by the present iterative method for an H-configuration (Fig. 3a).

CONCLUSION

A 3-D grid generation technique based on solving the Poisson equations has been presented. A satisfactory grid spacing distribution is obtained by adjusting three parameters in the forcing functions. The present method can be used to generate a 3-D grid network for advanced flow calculations about complex geometries of practical interest.

REFERENCES

1. NASA Scientific and Technical Information Office (1980) Numerical Grid Generation Techniques, NASA Conference Publication 2166.
2. Thompson, J. F., Thames, F. C., and Mastin, C. W. (1977) "Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies," NASA CR-2729.
3. Thomas, P. D. (1979) "Numerical Method for Predicting Flow Characteristics and Performance of Nonaxisymmetric Nozzles - Theory," NASA CR-3147.
4. Sorrenson, R. L. and Steger, J. L. (1980) "Numerical Generation of Two-Dimensional Grids by the Use of Poisson Equations with Grid Control at Boundaries," in Numerical Grid Generation Techniques, NASA Conference Publication 2166, 449-460.
5. Lee, K. D., Huang, M., Yu, N. J., and Rubbert, P. E. (1980) "Grid Generation for General Three-Dimensional Configurations," in Numerical Grid Generation Techniques, NASA Conference Publication 2166, 355-366.
6. Yu, H. Y. (1981) "Transonic Flow Simulations for Complex Configurations with Surface Fitted Grids," AIAA Paper 81-1258, AIAA 14th Fluid and Plasma Dynamics Conference.

AD P000998

Published 1982 by Elsevier Science Publishers
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

GENERATION OF THREE-DIMENSIONAL BOUNDARY-FITTED CURVILINEAR COORDINATE SYSTEMS FOR WING/WING-TIP GEOMETRIES USING THE ELLIPTIC SOLVER METHOD

FRANK C. THAMES
NASA Langley Research Center, Hampton, Virginia

ABSTRACT

A three-dimensional elliptic solver technique is utilized to generate surface-fitted coordinates about wing/wing-tip configurations. The method is applicable to wings of arbitrary section profile and camber, leading-edge sweep, taper ratio, and spanwise thickness variation. The basic theory of three-dimensional elliptic mappings is developed along with a method to compute interior coordinate control functions. Examples of grids generated about several wing/wing-tip geometries are given. A 49 × 33 × 17 grid requires about 3 minutes of CPU time on a CYBER 203 computer.

SYMBOLS

D	physical or computational domain
f_i	coordinate control functions (see eqs. (1a), (2a) and (4)), $i = 1,2,3$
J	determinant of Jacobian matrix, M
M	Jacobian matrix defined in equation (2d)
P_i	boundary condition functions for the ξ_i-coordinates in physical space (see eq. (1b)), $i = 1,2,3$
q_{ij}	boundary condition functions for the x_i-coordinates in computational space (see eqs. (2b) and (3)), $i = 1,2,3$; $j = 1,\ldots,6$
S,T	transformations from physical to computational domain (fig. 1)
x_1,x_2,x_3	Cartesian coordinates
X	coordinate triple (x_1,x_2,x_3)
α_{jk}	metric coefficients defined in equation (2c), $j,k = 1,2,3$
β_{jk}	cofactor of Jacobian matrix, M; $j,k = 1,2,3$
$\Lambda_{i,j.k}$	sub-plane j.k of the ξ_i = constant computational domain plane (see fig. 2(c))

ξ_1,ξ_2,ξ_3	computational domain coordinates
Ξ	coordinate triple, (ξ_1,ξ_2,ξ_3)
ϕ_k	coordinate control function (see eqs. (4)-(10))
∇^2	three-dimensional Laplacian operator in Cartesian coordinates

Subscripts

1,2,3	Cartesian or computational plane direction indicator or simple counter
i,j,k	coordinate direction counters
min	minimum value
max	maximum value

Superscripts

*	pertaining to computational domain

INTRODUCTION

The development of methods to generate surface-fitted curvilinear coordinates about relatively complex three-dimensional geometries has lagged considerably behind similar work in two dimensions. There are two principal reasons for the lack of progress in this area: (1) the work is, at best, quite difficult and (2) even if methods were available, they would be of limited usefulness because current computers simply are not large and fast enough to solve complex sets of three-dimensional partial-differential equations. There have, however, been some developments and several of these are summarized briefly below.

Caughey and Jameson[1] patch together grids suitable for transonic potential solutions for transport-type wing-body configurations utilizing a sequence of sheared conformal mappings. Holst[2] solves similar flow problems by stringing together a series of two-dimensional grids generated using the two-dimensional elliptic solver approach.[3] Middlecoff and Thomas[4] utilized a similar approach to develop grids for nozzle-afterbody geometries. Eiseman has developed a rather general algebraic mapping method applicable to both two and three dimensions and has generated a surface-fit coordinate system for an isolated wing.[5] The first applications of the three-dimensional elliptic solver methods were performed by Mastin and Thompson for spherical-shaped bodies[6] and isolated wings.[7] Yu[8] has used the same methods to develop surface-fit curvilinear systems for general transport aircraft wing-body configurations.

The remainder of this paper is divided into four sections. The first two cover the form of the transformation and its generation, including discussions of the governing elliptic system and choice of grid control functions. Computed results and conclusions are presented in the last two sections.

FORM OF THE TRANSFORMATION

In choosing the form of a transformation, one must keep in mind its ultimate use--namely, what partial-differential equations are to be solved on the resulting coordinate system. The present work concerns the development of boundary-fitted coordinates for wing-tips suitable for Navier-Stokes equations computations. There are many ways to transform wing-tip geometries. Two of the more common approaches are illustrated schematically in figure 1. The transformation S depicts the so-called slit-plane approach which, although adaptable to a wide variety of complex three-dimensional geometries, does a rather poor job on surfaces with small radii of curvature. Another approach, denoted T in figure 1, "slices" the wing along the leading and trailing edges and then "unfolds" the wing and tip and maps them onto a portion of the lower surface of the computational plane. The T-form was chosen for the current work for two reasons: (1) The wing-tip geometry is represented with a higher fidelity and (2) the coding of the Navier-Stokes algorithms of the factored, block type[9,10], which are to be used for the flow solutions, is considerably simpler if the body geometry lies on a boundary of the computational domain.

Figure 2 presents a more detailed look at the T-transformation. Recall that the computational plane is created by slicing through the physical domain along the dotted line (fig. 2(a)) and unfolding the domain such that form shown in figures 2(b) and 2(c) result. These figures illustrate several points. First, the transformation has the form of a "sideways" three-dimensional C-grid as can easily be noted from parts (a) and (d) of figure 2. Second, and perhaps unfortunately, the transformation has an axis singularity similar to that occurring in standard cylindrical coordinates. Note that the line a_3-c_3 in figure 2(a) maps onto the cross-hatched plane forward of the wing-tip area in figure 2(b) (plane $A_{3,1.2}$ in fig. 2(c)) and the line e_3-g_3 maps onto a similar cross-hatched plane aft of the wing-tip area in figure 2(b) (plane $A_{3,1.8}$ in fig. 2(c)). This singularity presents no difficulty in the coordinate generation process; however, it will affect the form of the fluid flow differential equations whose numerical solution is carried out on the curvilinear system. Special, limiting forms of the

equations must be developed for the singular regions of the coordinate system. Note also that in certain instances, the transformation may not be singular. For example, if the forward ξ_2 = constant boundary plane were taken aft of the wing leading edge and the aft ξ_2 = constant boundary plane were placed forward of the wing trailing edge, then the singular regions would be avoided. Examples of this form of the transformation were developed and are presented below. Third, note that the planes on either side of the singular surfaces both forward and aft of the wing are periodic boundaries created by the "unfolding" operation described above. That is, planes $\Lambda_{3,1.1}$ and $\Lambda_{3,1.3}$ are the same physical surface as are planes $\Lambda_{3,1.7}$ and $\Lambda_{3,1.9}$. As a final point, we require that the computational domain be regularly-spaced. This requirement allows us to assume a uniform spatial step size (usually, unity) for the difference approximations made in the computational domain.

GENERATION OF THE TRANSFORMATION

Having established the desired structural form of the mapping, the transformation must now be generated. There are many methods of accomplishing this--we choose the elliptic solver approach.

Governing Equations

In reference 6, Mastin and Thompson set forth the mathematical foundation for the extension of two-dimensional elliptic solver techniques[3] to three dimensions. Their results showed that if the curvilinear coordinates, (ξ_1,ξ_2,ξ_3), were required to satisfy

$$\nabla^2 \xi_i = f_i, \qquad X \in D, \qquad i = 1,2,3 \tag{1a}$$

with boundary conditions

$$\xi_i = p_i(x_1,x_2,x_3), \qquad X \in \partial D, \qquad i = 1,2,3 \tag{1b}$$

then the Cartesian coordinates, (x_1,x_2,x_3), must satisfy the coupled quasi-linear set

$$\sum_{j=1}^{3}\sum_{k=1}^{3} \alpha_{jk} \frac{\partial^2 x_i}{\partial \xi_j \partial \xi_k} + J^2 \sum_{k=1}^{3} f_k \frac{\partial x_i}{\partial \xi_k} = 0, \qquad \Xi \in D^*, \qquad i = 1,2,3 \tag{2a}$$

with boundary conditions

$$x_i = q_{ij}(\xi_1,\xi_2,\xi_3), \qquad \Xi \in \partial D^*, \qquad \begin{matrix} i = 1,2,3 \\ j = 1,\ldots,6 \end{matrix} \tag{2b}$$

where:

$$\alpha_{jk} \equiv \sum_{m=1}^{3} \beta_{mj}\beta_{mk}, \qquad j,k = 1,2,3 \tag{2c}$$

β_{jk} is the cofactor of the (j,k)-element of the Jacobian matrix

$$M = \frac{\partial(x_1,x_2,x_3)}{\partial(\xi_1,\xi_2,\xi_3)} \tag{2d}$$

and J is the determinant of M. Here D is some suitably bounded domain and D^* is the image of D under the transformation and, as mentioned above, is usually required to be uniformly spaced. This requirement simplifies both the numerical solution of equations (2) (to generate the transformation) and the numerical solution of the physical problem of interest. The exact nature of the transformation (as evidenced, for example, by its appearance in physical space) is governed by both the differential form of equations (1) (in this instance, three-dimensional Laplacians) and the properties of the f-functions. The f-functions are covered in detail in the next section.

Equations (1b) and (2b) tend to imply that all boundary conditions are of the Dirichlet type. Actually, mixed Dirichlet and Neumann conditions can be used. However, it is important to keep in mind that both boundary function and normal directional derivative values cannot be specified everywhere on ∂D or ∂D^* as this would over-specify the differential equations. In the current work, only Dirichlet conditions are used.

Since equation (1a) is never actually solved, we will not elaborate on the boundary conditions for this equation. The boundary conditions for equation (2a) are

$$x_i = \left\{ \begin{matrix} q_{i1}\left[\xi_1 = (\xi_1)_{min}, \xi_2, \xi_3\right], & \Xi \in A_{1,1} \\ q_{i2}\left[\xi_1 = (\xi_1)_{max}, \xi_2, \xi_3\right], & \Xi \in A_{1,2} \end{matrix} \right\} i = 1,2,3 \tag{3a}$$

$$x_i = \begin{Bmatrix} q_{i3}\left[\xi_1, \xi_2 = (\xi_2)_{min}, \xi_3\right], \ \Xi \in \Lambda_{2,1} \\ q_{i4}\left[\xi_1, \xi_2 = (\xi_2)_{max}, \xi_3\right], \ \Xi \in \Lambda_{2,2} \end{Bmatrix} \quad i = 1,2,3 \tag{3b}$$

$$x_i = \begin{Bmatrix} q_{i5}\left[\xi_1, \xi_2, \xi_3 = (\xi_3)_{min}\right], \ \Xi \in \Lambda_{3,1} \\ q_{i6}\left[\xi_1, \xi_2, \xi_3 = (\xi_3)_{max}\right], \ \Xi \in \Lambda_{3,2} \end{Bmatrix} \quad i = 1,2,3 \tag{3c}$$

One aspect of these boundary conditions deserves comment. It was mentioned in the previous section that the $\xi_3 = (\xi_3)_{min}$ plane (i.e., $\Lambda_{3,1}$) contained periodic sub-planes (namely, $\Lambda_{3,1.1}$ with $\Lambda_{3,1.3}$ and $\Lambda_{3,1.7}$ with $\Lambda_{3,1.9}$), yet the first of equations (3c) implies that Dirichlet boundary data are specified for all $\Xi \in \Lambda_{3,1}$. In fact, Dirichlet data are specified on all of $\Lambda_{3,1}$ for the solutions given in this paper. This is done to allow the user to control the x_1-coordinate spacing in these regions. (If this is not done, the elliptic equations tend to "smooth-out" the x_1-distributions in these planes in an unsatisfactory manner.) Unfortunately, ξ_3-derivatives of the x_i-coordinates will, in general, be discontinuous across these planes although specification of the Dirichlet data maintains continuity of the x_i-coordinates. However, these discontinuities can be eliminated by smoothing the x_i-coordinate distributions in the periodic planes after the elliptic equations have been solved.

Control Functions

The functions f_k appearing in equations (1a) and (2a) are used to modify the distribution of the ξ_i-coordinates in the domain D. The basic mathematical theory which describes why these functions produce various effects is covered in reference 3 and need not be repeated here. For the current work, these functions will be computed using the boundary distribution compatibility method first developed for two dimensions by Ghia, Hodge, and Hankey[11] and rediscovered later by Middlecoff and Thomas.[4] The method has been used in three dimensions by Yu.[8]

To implement the approach, the f_k-functions are redefined as

$$f_k \equiv a_{kk}\phi_k/J^2 \tag{4}$$

which, when substituted into equation (2a) yields

$$\sum_{j=1}^{3}\sum_{k=1}^{3} \alpha_{jk} \frac{\partial^2 x_i}{\partial \xi_j \partial \xi_k} + \sum_{k=1}^{3} \alpha_{kk}\phi_k \frac{\partial x_i}{\partial \xi_k} = 0, \quad i = 1,2,3 \tag{5}$$

The ϕ_k are user-specified functions. The effect of this substitution is two-fold. First, it removes the numerical problems associated with the Jacobian determinant, J. To see this, note that in equation (1a) f_k is multiplied by J^2 which, for viscous grids, is very small near a solid boundary. Thus, to have any effect at all, the magnitude of f_k must be very large in these regions. (Values of $0(10^8)$ are not unusual.) Therefore, if the form given in equation (1a) is implemented, the computer is constantly multiplying a very large number by a very small number--an undesirable situation. The substitution given by equation (4) eliminates this problem by dividing J^2 out of the equation. The second advantage of the equation (4) substitution is that the controlling functions, ϕ_k, are multiplied by the α_{kk} coefficients which are direct measures of the arc length distribution along the ξ_k-coordinates. Thus, by multiplying α_{kk} by ϕ_k, we are in effect altering or, to some extent, controlling the rate of arc length change along the ξ_k-coordinates, which is exactly what we wish to accomplish.

Consider the sketch of the three intersecting physical domain coordinates given in figure 3. If we assume that the $\xi_2 = \bar{\xi}_2$ and $\xi_3 = \bar{\xi}_3$ coordinate planes intersect the $\xi_1 = (\xi_1)_{min}$ boundary plane almost normally and with small curvature, then we can neglect the first and second ξ_1- and ξ_2-derivatives of x_3 along the dotted vertical line. The $i = 3$ component of equation (5) then reduces to

$$\frac{\partial^2 x_3}{\partial \xi_3^2} + \phi_3 \frac{\partial x_3}{\partial \xi_3} = 0 \tag{6}$$

or,

$$\phi_3\left[(\xi_1)_{min}, \bar{\xi}_2, \xi_3\right] = -\left[\frac{d^2 x_3}{d\xi_3^2} \Big/ \frac{dx_3}{d\xi_3}\right]_{(\xi_1)_{min}, \bar{\xi}_2} \tag{7a}$$

A similar analysis holds along the dashed vertical line so that

$$\phi_3\left[(\xi_1)_{max}, \bar{\xi}_2, \xi_3\right] = -\left[\frac{d^2x_3}{d\xi_3^2} \Big/ \frac{dx_3}{d\xi_3}\right]_{(\xi_1)_{max}, \bar{\xi}_2} \tag{7b}$$

If the boundary distributions of x_3 are given along the two lines, then equations (7) can be solved for the two ϕ_3-distributions. (It is most practical to perform these solutions numerically for $(\xi_3)_{min} < \xi_3 < (\xi_3)_{max}$ using central difference approximations for the derivatives. The values of ϕ_3 at $(\xi_3)_{min}$ and $(\xi_3)_{max}$ are irrelevant.) The ϕ_3-distribution at the remaining interior points can then be computed by linearly interpolating between these two boundary distributions. That is

$$\phi_3\left[\xi_1, \bar{\xi}_2, \xi_3\right] = \left[\xi_1 - (\xi_1)_{min}\right] \frac{\Delta\phi_3}{\Delta\xi_1} + \phi_3\left[(\xi_1)_{min}, \bar{\xi}_2, \xi_3\right] \tag{8a}$$

where

$$\Delta\xi_1 \equiv (\xi_1)_{max} - (\xi_1)_{min} \tag{8b}$$

$$\Delta\phi_3 \equiv \phi_3\left[(\xi_1)_{max}, \bar{\xi}_2, \xi_3\right] - \phi_3\left[(\xi_1)_{min}, \bar{\xi}_2, \xi_3\right] \tag{8c}$$

Applying equations (7) and (8) for all $(\xi_2)_{min} < \xi_2 < (\xi_2)_{max}$ completes the ϕ_3-function computation. Note that ϕ_3 is computed from a user-specified distribution of physical domain coordinates--an approach that is superior to the hit-or-miss methods used in the early stages of elliptic solver grid-generation work.[12] Furthermore, since orthogonality was assumed in developing equation (6), the ϕ_3-distribution obtained by solving this equation will tend to force orthogonality at the boundary. Equation (6) comes from evaluating one component of the field differential equation (eq. (5)) on the boundary--hence, this method of computing control functions is termed the boundary compatibility method.

A similar analysis for the ϕ_1 and ϕ_2 functions yields the formulas:

$$\phi_1\left[\bar{\xi}_1, \bar{\xi}_2, (\xi_3)_{min}\right] = -\left[\frac{d^2x_2}{d\xi_1^2} \Big/ \frac{dx_2}{d\xi_1}\right]_{\bar{\xi}_2, (\xi_3)_{min}} \tag{9a}$$

$$\phi_1\left[\xi_1, \bar{\xi}_2, (\xi_3)_{max}\right] = -\left[\frac{d^2x_2}{d\xi_1^2}\Big/\frac{dx_2}{d\xi_1}\right]_{\bar{\xi}_2, (\xi_3)_{max}} \tag{9b}$$

and

$$\phi_2\left[\bar{\xi}_1, \xi_2, (\xi_3)_{min}\right] = -\left[\frac{d^2x_1}{d\xi_2^2}\Big/\frac{dx_1}{d\xi_2}\right]_{\bar{\xi}_1, (\xi_3)_{min}} \tag{10a}$$

$$\phi_2\left[\bar{\xi}_1, \xi_2, (\xi_3)_{max}\right] = -\left[\frac{d^2x_1}{d\xi_2^2}\Big/\frac{dx_1}{d\xi_2}\right]_{\bar{\xi}_1, (\xi_3)_{max}} \tag{10b}$$

Interpolation formulas such as equations (8) can then be used with equations (9) and (10) to compute ϕ_1 and ϕ_2 over the entire computational domain, D^*.

Numerical Solution Technique

Since the system of three equations to be solved (eqs. (5)) is elliptic with Dirichlet boundary conditions (eqs. (3)), the system can be easily solved by any number of straightforward numerical schemes. The one chosen for use here is successive line over-relaxation (SLOR). The ξ_3-direction was taken as the implicit direction. Standard, second-order accurate central differences were used to approximate all derivatives appearing in equations (5). The four-corner-point formula was used for the cross-derivative approximations. Performance of the scheme is documented in the following section.

DISCUSSION OF RESULTS

Cases Computed and Boundary Geometry

Table 1 summarizes the wing/wing-tip configurations for which computational grids were generated.

TABLE 1

WING CHARACTERISTICS

Wing	Root Airfoil Section*	Tip Airfoil Section*	Leading Edge Sweep	Taper Ratio	Root Chord+	Semispan
1	0012	0012	0	1	1.0089	1
2	9312	9312	0	1	1.0089	1
3	0012	0012	22.5	0.5	1.0089	1
4	0012	0012	30	1	1.0089	1
5	0020	0010	0	1	1.0089	1

*NACA 4-digit airfoil designation.

+This value allows an analytic trailing-edge closure.

Isometric views of these five configurations are given in figure 4. NACA four-digit airfoil sections were used for convenience in generating the boundary data--the method is not restricted to these shapes. The wing-tip surfaces were created by rotating the wing-tip airfoil section thickness distribution about the section mean line. The tip cross-sections normal to the tip-section mean line are, thus, semi-circles with a diameter equal to the local tip airfoil section thickness ratio. Again, this tip generation procedure was used for computational convenience and is not a restriction on the overall approach.

A fifth-order polynomial was used to stretch the x_1-distribution (i.e., the streamwise coordinate) for these wings rather than the more conventional cosine distribution. For those cases in which the physical region extends forward and/or aft of the wing itself, use of the cosine x_1-distribution on the wing leads to jump discontinuities in the second derivatives of x_1 at the leading and trailing edges of the wing. The polynomial distribution avoids this problem by requiring the first and second derivatives of x_1 with respect to ξ_2 to be zero at these two locations. It is relatively easy then to add-on the x_1-distributions fore and aft of the wing in an independent, yet smooth, manner. The spanwise distribution of coordinates pictured on figure 4 are somewhat arbitrary. These can, however, be altered to suit the user's needs.

The outer boundary for all cases studied in this paper consists of a semi-cylindrical tube illustrated in figure 2. Boundary points on curves such as

$b_1 - b_2 - b_3 - b_4 - b_5$ were distributed on an equal arc-length basis. Distribution of boundary points on the "left" boundary shown in figure 2 (i.e., along lines such as $a_1 - b_1$, $a_5 - b_5$, etc.) was specified using a stretch of the form x_3^n where n is some real number. Distribution of the longitudinal (i.e., x_1) coordinate was chosen to be uniform or to match the x_1-distribution for the inner boundary. As noted above, any of these distributions can be altered to suit the user. For example, one interesting alteration would be to "square-off" the outer boundary; the resulting configuration would be representative of a semi-span wing in a wind tunnel.

Sample Results

Sample coordinate systems generated for three of the wing configurations listed in table 1 are shown here to illustrate the mappings. Figure 5 presents an isometric view of the boundary data for a Wing-1 configuration and a similar view with three ξ_2 = constant planes superposed on the boundary data. A similar set of results for the Wing-3 geometry is given in figure 6. All features of these grids appear to be desirable except the rather severe distortion of the ξ_2 = constant planes fore and aft of the wing along the periodic boundary planes. This distortion implies failure of the ϕ_2-function. A modified ϕ_2-function interpolation is suggested below which should help alleviate this problem. The grid size for the results shown in figures 5 and 6 is 33 × 49 × 17. Figure 7 shows the boundary data and several views of a Wing-5 configuration solution. Note that for these results, the regions fore and aft of wing have been eliminated. Thus, this coordinate system has no axis singularity and is quite well behaved. Figure 7 also illustrates a point made in a previous section concerning the form of the transformation--namely, the structure of the grid is quite good in the tip-region, but rather poor near the wing leading edge. The grid size for Wing-5 region shown is 33 × 27 × 17.

Effects of the Control Functions

The influence of the ϕ_1-function on the interior grid distribution is illustrated in figure 8. Note that the principal effect of ϕ_1 is a "pinching" of radial grid lines toward the ξ_1-coordinate plane connecting the midline of the tip with the outer boundary. This behavior is caused by the change in sign of ϕ_1 at the ξ_1-coordinate plane described above. In most cases the effects produced by ϕ_1 are not particularly desirable. If the "pinching" effect is wanted, a redistribution of points on both the tip

surface and outer boundary will produce a more uniform result. The ϕ_1-function was set to zero in all computations reported here save the one given in figure 8.

The results given in figure 9 document the influence of the ϕ_2-function on the grid structure. Unfortunately, ϕ_2 seems to have very little effect as it seems unable to eliminate the "kink" in the ξ_3-grid lines. These "kinks" will lead to problems in the finite-difference solutions of application equations carried out on this grid since some of the grid metric coefficients will be discontinuous. A fix that may improve the performance of ϕ_2 is to replace the linear interpolation scheme currently used to distribute this function in the interior field with a weighted nonlinear method that "concentrates" the interior boundary values of ϕ_2 (see eq. (10)) nearer the inner boundary. This approach has not been tried as yet.

Alterations in the interior grid structure induced by the ϕ_3 control-function are shown in figure 10. Note that ϕ_3 induces a very regular radial stretching replicating that specified on the boundary, which was the desired effect.

Performance of the Numerical Method

To access the performance of the SLOR iteration scheme, computer runs for three different sized grids for three of the wing configurations were made. The average of the timing and iteration count information from these runs is shown in figure 11. As anticipated, both the time and iteration count increase at exponential type rates with grid size. For example, a 50,000-point grid would require about 25 minutes of CPU time. For this and larger grids, a more efficient solution algorithm such as the multi-grid algorithm should be used.

CONCLUSIONS

An approach has been developed to generate surface-fitted coordinates about arbitrary wing/wing-tip geometries using the three-dimensional elliptic solver method. Grids having $O(30,000)$ points were generated about five different wing geometries. Configuration variables included wing-section shape, thickness, and camber; wing leading-edge sweep; taper ratio; and spanwise thickness distribution. Computation time required to develop each of these systems averaged about 3 minutes on a CYBER 203 computer using a simple SLOR solution algorithm. A set of grid control-functions was developed to provide some measure of control over the interior grid-point distribution.

Although the method presented in this paper was able to develop curvilinear systems suitable for accurate finite-difference calculations on quite general wing/wing-tip geometries, the need for further work in two areas was clearly identified. These are:

- o coordinate control function development
- o faster solution algorithm development

REFERENCES

1. Caughey, D. A.; and Jameson, A.: Recent Progress in Finite Volume Calculations for Wing-Fuselage Combinations. AIAA Paper No. 79-1513, July 1979.
2. Holst, T. L.: A Fast Conservative Algorithm for Solving the Transonic Full-Potential Equation. AIAA Paper No. 79-1456-CP, in A Collection of Technical Papers, AIAA 4th Computational Fluid Dynamics Conference, July 1979, pp. 109-121.
3. Thompson, J. F.; Thames, F. C.; and Mastin, C. W.: Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies. J. Comp. Phys., vol. 15, no. 3, July 1974, pp. 299-319.
4. Middlecoff, J. F.; and Thomas, P. D.: Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations. AIAA Paper No. 79-1462-CP, in A Collection of Technical Papers, AIAA 4th Computational Fluid Dynamics Conference, July 1979, pp. 175-179.
5. Eiseman, P. R.: Three-Dimensional Coordinates About Wings. AIAA Paper No. 79-1461-CP, in A Collection of Technical Papers, AIAA 4th Computational Fluid Dynamics Conference, July 1979, pp. 166-174.
6. Mastin, C. W.; and Thompson, J. F.: Transformation of Three-Dimensional Regions Onto Rectangular Regions by Elliptic Systems. Numerische Mathematik, vol. 29, 1978, pp. 397-407.
7. Mastin, C. W.; and Thompson, J. F.: Three-Dimensional Body-Fitted Coordinate Systems for Numerical Solution of the Navier-Stokes Equations. AIAA Paper No. 78-1147, July 1978.
8. Yu, N. J.: Grid Generation and Transonic Flow Calculations for Three-Dimensional Configurations. AIAA Paper No. 80-1391, July 1980.
9. Briley, W. R.; and McDonald, H.: Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique. Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics. Lecture Notes in Physics, vol. 35 (Robert D. Richtmyer, ed.), Springer-Verlag, 1975, pp. 105-110.
10. Beam, Richard M.; and Warming, R. F.: An Implicit Factored Scheme for the Compressible Navier-Stokes Equations. AIAA J., vol. 16, no. 4, April 1978, pp. 393-402.
11. Ghia, U.; Hodge, J. K.; and Hankey, W. L.: An Optimization Study for Generating Surface-Oriented Coordinates for Arbitrary Bodies in High Re-Flow. AFFDL-TR-77-117, Dec. 1977.
12. Thompson, J. F.; Thames, F. C.; and Mastin, C. W.: TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies. J. Comp. Phys., vol. 24, no. 3, July 1977, pp. 274-302.

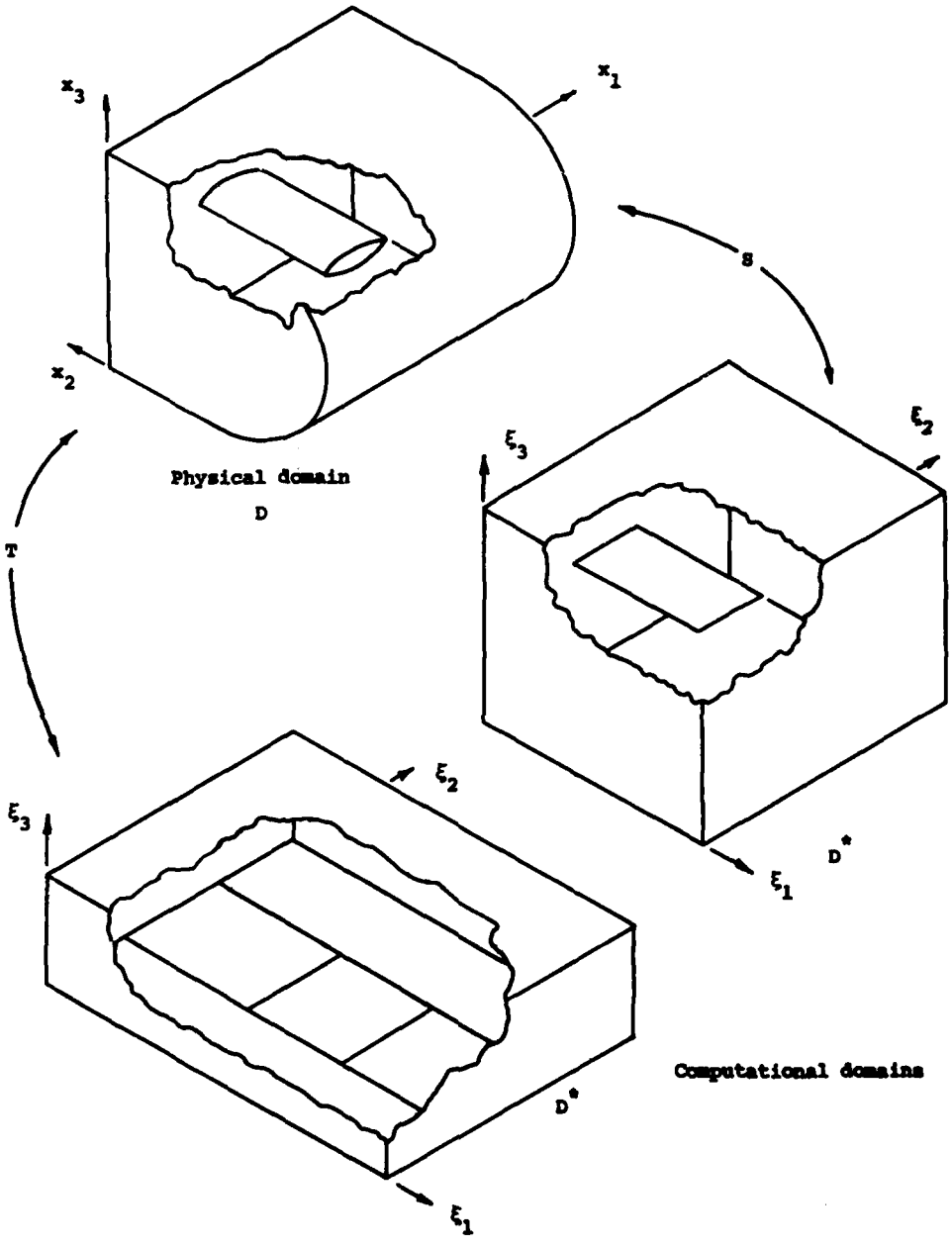
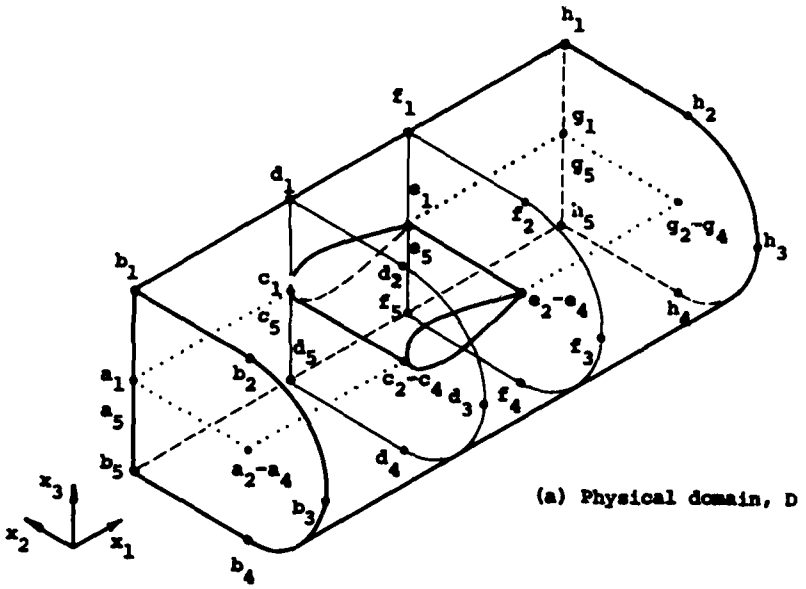
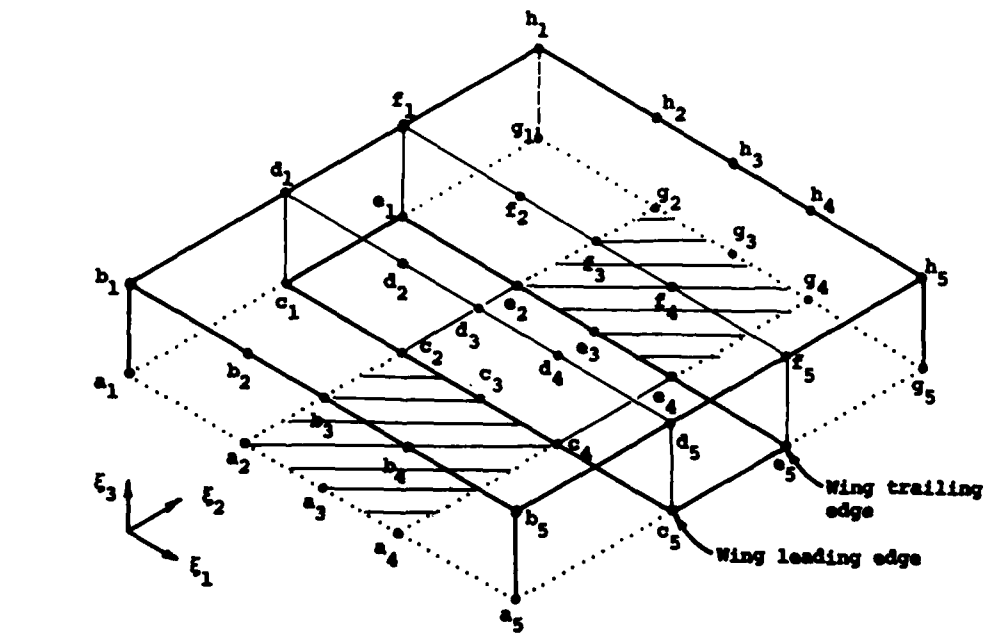


Fig. 1. Schematics of two candidate transformations.

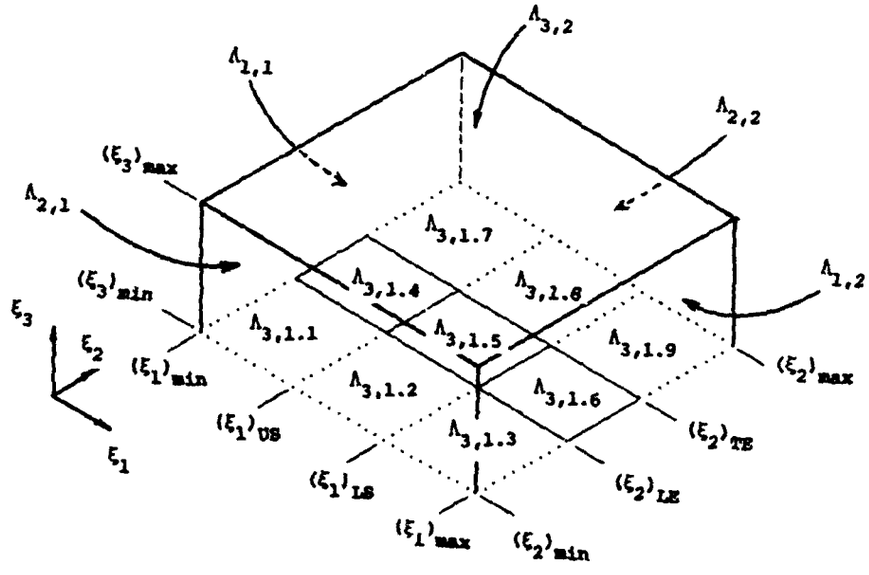


(a) Physical domain, D

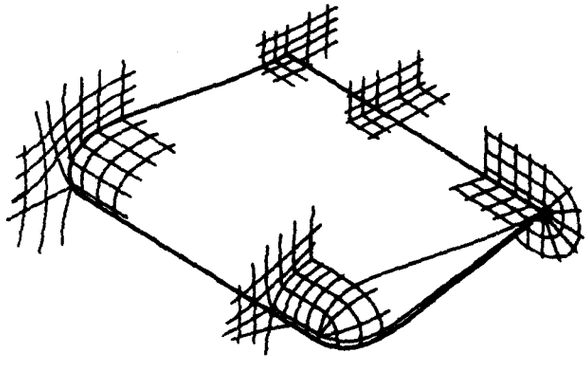


(b) Computational domain, D*

Fig. 2. Details of the T-transformation.



(c) Computational domain planes



(d) Grid structure in the physical domain

Fig. 2. Concluded.

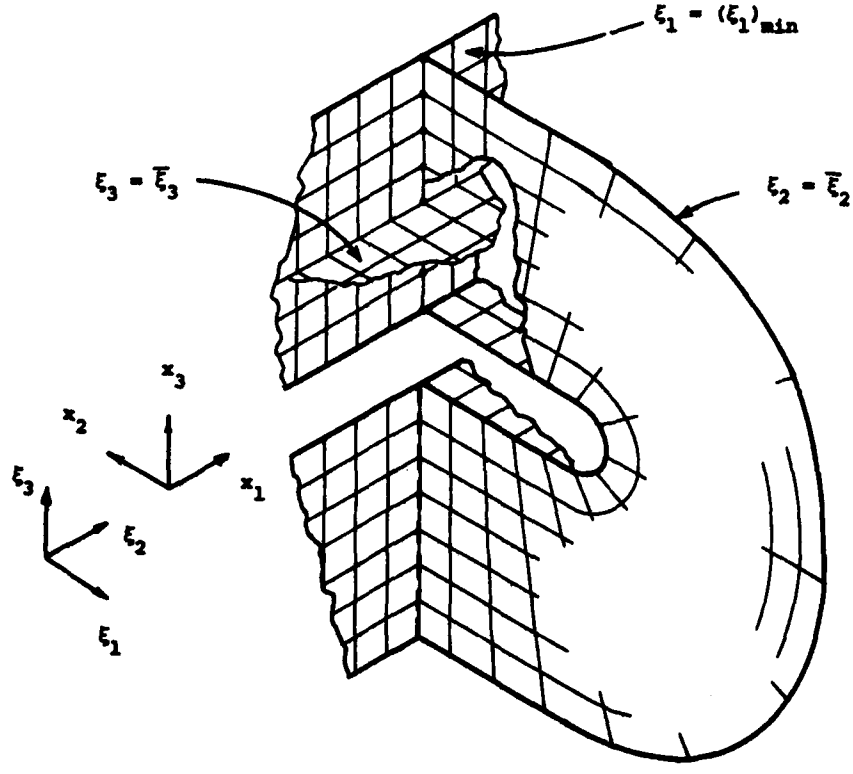
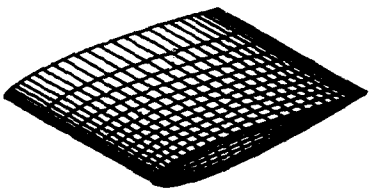
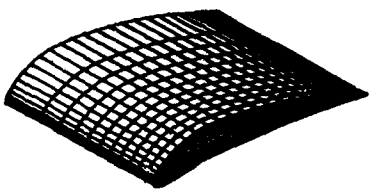


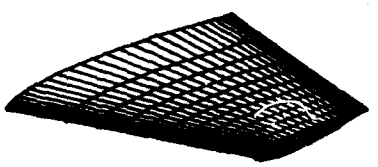
Fig. 3. Cutaway view of the intersection of three ξ-planes.



(a) Wing-1

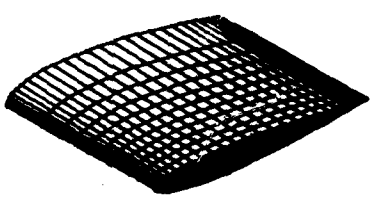


(b) Wing-2



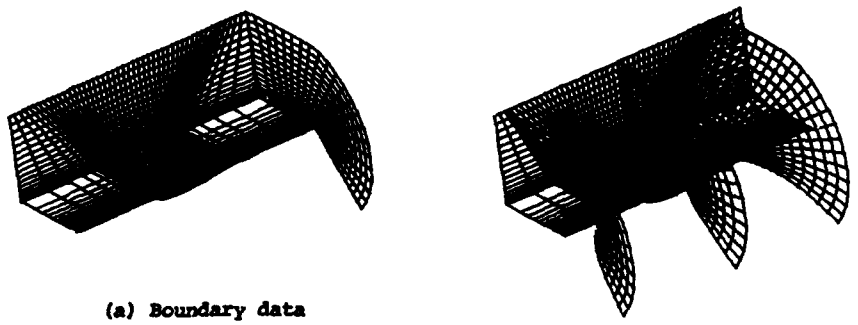
(c) Wing-3

(d) Wing-4



(e) Wing-5

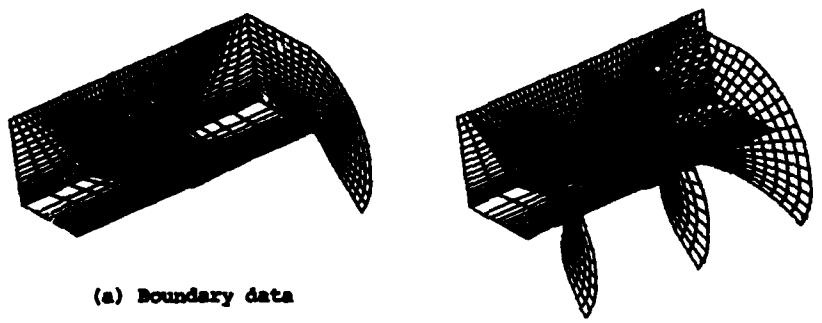
Fig. 4. Isometric views of five wing/wing-tip configurations



(a) Boundary data

(b) Boundary data and three computed ξ_2-coordinate planes

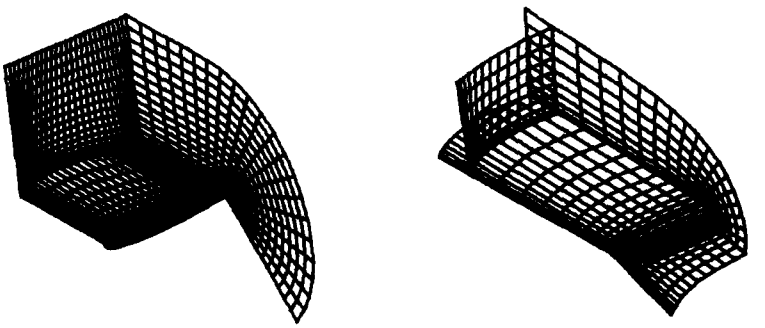
Fig. 5. Coordinate system generation: Wing-1 configuration



(a) Boundary data

(b) Boundary data and three computed ξ_2-coordinate planes

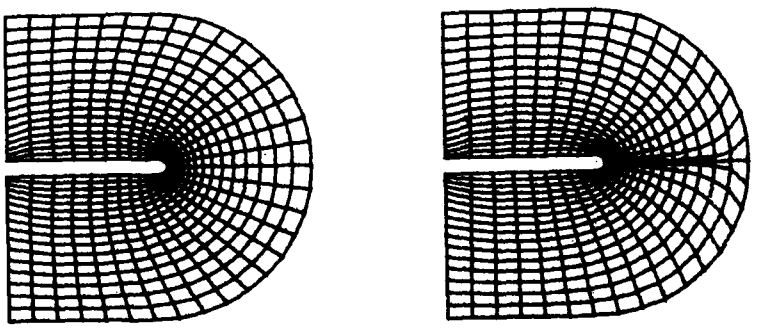
Fig. 6. Coordinate system generation: Wing-3 configuration



(a) Boundary data

(b) Wing upper surface boundary and computed ξ_1 and ξ_2 coordinate planes

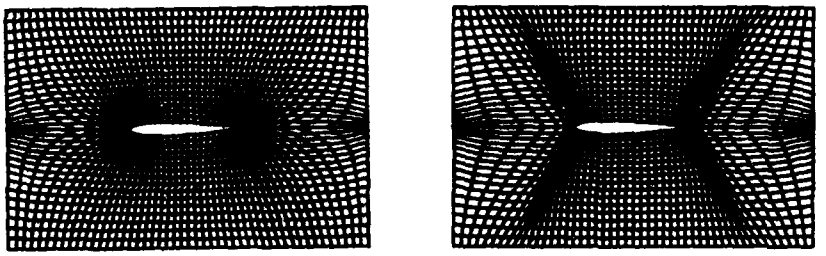
Fig. 7. Coordinate system generation: Wing-5 configuration



(a) $\phi_1 = 0$

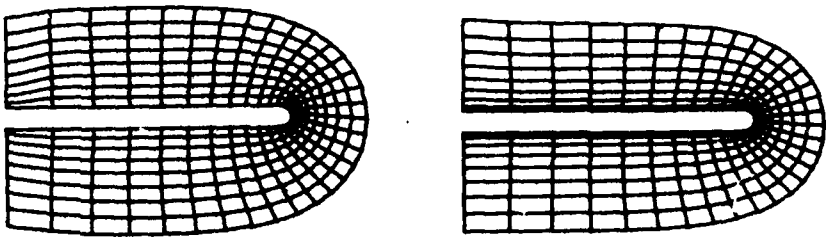
(b) ϕ_1 computed from equations (9)

Fig. 8. Control function effects: ϕ_1-function



(a) $\phi_2 = 0$ (b) ϕ_2 computed from equations (10)

Fig. 9. Control function effects: ϕ_2-function



(a) $\phi_3 = 0$ (b) ϕ_3 computed from equations (7)

Fig. 10. Control function effects: ϕ_3-function

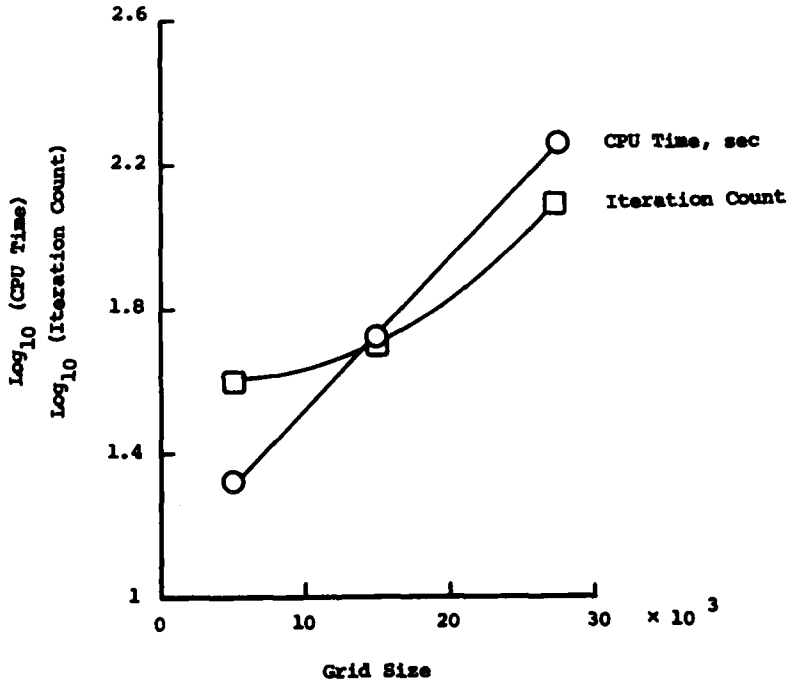


Fig. 11. SLOR algorithm performance.

AD P000999

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

NUMERICAL GENERATION OF THREE-DIMENSIONAL COORDINATES BETWEEN BODIES OF ARBITRARY SHAPES

Z. U. A. WARSI† AND J. P. ZIEBARTH‡
Department of Aerospace Engineering, Mississippi State University,
Mississippi State, Mississippi 39762, USA

INTRODUCTION

This paper is devoted to the numerical solution of a set of second order elliptic partial differential equations for the generation of three-dimensional curvilinear coordinates between two arbitrary shaped bodies. The central idea of the method is to generate a series of surfaces between the given inner and the outer boundary surfaces and then to connect these surfaces in such a manner so as to have a sufficiently differentiable three-dimensional coordinate net in the enclosed region.

The basic analytical foundation of the present method has already been laid out by Warsi in §2 of Ref. 1. However, it is important to state here that the proposed equations for the numerical solution form a consistent set of second order elliptic equations which are a consequence of the equations of Gauss[2] for a surface. Additional constraints are then imposed which, besides yielding the simplest form of equations for numerical purposes, also preserve the essential geometric properties of the generated surfaces.

Formulation of the mathematical model

To fix ideas, let it be desired to generate the coordinates between the two surfaces designated as $\eta = \eta_B$ (the inner surface) and $\eta = \eta_\infty$ (the outer surface) respectively as is shown in Fig. 1 . The two coordinates which vary in these two surfaces are then labeled as ξ or I and ζ or K. The surfaces $\eta = \eta_B$ and $\eta = \eta_\infty$ are the known surfaces in which the Cartesian coordinates $\underset{\sim}{r} = (x,y,z)$ are given as functions of ξ and ζ, that is,

$$\underset{\sim}{r} = \underset{\sim}{r}_B(\xi,\zeta) \ , \ \underset{\sim}{r} = \underset{\sim}{r}_\infty(\xi,\zeta)$$

are known either numerically or analytically. The method to be discussed

---

†Professor

‡Graduate Research Assistant

generates a surface for each fixed value of ζ or K starting from a curve on B and ending at the corresponding value of ζ or K on the outer boundary surface. Refer to Fig. 1b.

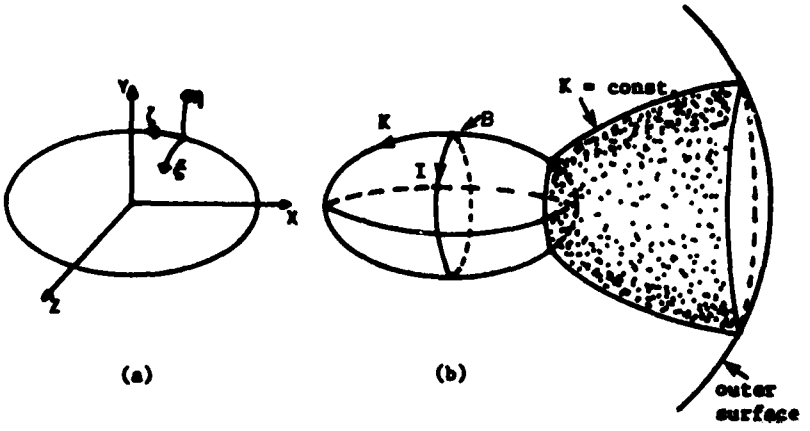


Figure 1.(a). Coordinates ξ and ζ on the given surfaces. (b) the generated surface ζ = const.

Referring to Eq. (18) in Warsi[1], we now impose the restrictions

$$\Delta_2\xi = \frac{1}{G_3}(2g_{12}\Gamma^1_{12} - g_{22}\Gamma^1_{11} - g_{11}\Gamma^1_{22}) = 0 , \tag{1}$$

$$\Delta_2\eta = \frac{1}{G_3}(2g_{12}\Gamma^2_{12} - g_{22}\Gamma^2_{11} - g_{11}\Gamma^2_{22}) = 0 , \tag{2}$$

for ζ = const. In Eqs. (1) and (2) Δ_2 is the Beltrami's second order differential operator[1,2], and

$$g_{11} = x_\xi^2 + y_\xi^2 + z_\xi^2 , \tag{3a}$$

$$g_{12} = x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta , \tag{3b}$$

$$g_{22} = x_\eta^2 + y_\eta^2 + z_\eta^2 , \tag{3c}$$

$$G_3 = g_{11}g_{22} - (g_{12})^2 , \tag{3d}$$

$$\Gamma^1_{11} = \frac{1}{2G_3}\left[g_{22}\frac{\partial g_{11}}{\partial \xi} + g_{12}\left(\frac{\partial g_{11}}{\partial \eta} - 2\frac{\partial g_{12}}{\partial \xi}\right)\right] , \tag{4a}$$

$$\Gamma^2_{22} = \frac{1}{2G_3}\left[g_{11}\frac{\partial g_{22}}{\partial \eta} + g_{12}\left(\frac{\partial g_{22}}{\partial \xi} - 2\frac{\partial g_{12}}{\partial \eta}\right)\right], \tag{4b}$$

$$\Gamma^1_{22} = \frac{1}{2G_3}\left[g_{22}\left(2\frac{\partial g_{12}}{\partial \eta} - \frac{\partial g_{22}}{\partial \xi}\right) - g_{12}\frac{\partial g_{22}}{\partial \eta}\right], \tag{4c}$$

$$\Gamma^2_{11} = \frac{1}{2G_3}\left[g_{11}\left(2\frac{\partial g_{12}}{\partial \xi} - \frac{\partial g_{11}}{\partial \eta}\right) - g_{12}\frac{\partial g_{11}}{\partial \xi}\right], \tag{4d}$$

$$\Gamma^1_{12} = \frac{1}{2G_3}\left(g_{22}\frac{\partial g_{11}}{\partial \eta} - g_{12}\frac{\partial g_{22}}{\partial \xi}\right), \tag{4e}$$

$$\Gamma^2_{12} = \frac{1}{2G_3}\left(g_{11}\frac{\partial g_{22}}{\partial \xi} - g_{12}\frac{\partial g_{11}}{\partial \eta}\right). \tag{4f}$$

Based on the structure of the Christoffel symbols $\Gamma^{\alpha}_{\beta\gamma}$ in Eqs. (4) we conclude that the constraining equations (1) and (2) are essentially a set of differential constraints on the variations of the metric coefficients $g_{\alpha\beta}$. Thus under the constraining equations (1) and (2), the three equations for the generation of the Cartesian coordinates x, y, z can be obtained. Below we write the equations when it is desired to have a concentration or expansion in the coordinates ξ and η, (refer to Eqs. (38) in Warsi[1]). For brevity of notation we use the same coordinates (ξ,η,ζ) either with or without coordinate redistributions. The equations are

$$\mathcal{L}x = XR, \tag{5a}$$

$$\mathcal{L}y = YR, \tag{5b}$$

$$\mathcal{L}z = ZR, \tag{5c}$$

where

$$\mathcal{L} = g_{22}\partial_{\xi\xi} - 2g_{12}\partial_{\xi\eta} + g_{11}\partial_{\eta\eta} + P\partial_{\xi} + Q\partial_{\eta}, \tag{6}$$

$$X = (y_{\xi}z_{\eta} - y_{\eta}z_{\xi})/\sqrt{G_3}, \tag{7a}$$

$$Y = (x_{\eta}z_{\xi} - x_{\xi}z_{\eta})/\sqrt{G_3}, \tag{7b}$$

$$Z = (x_{\xi}y_{\eta} - x_{\eta}y_{\xi})/\sqrt{G_3}, \tag{7c}$$

$$R = (Xx_\zeta + Yy_\zeta + Zz_\zeta)(g_{11}\Gamma^3_{22} - 2g_{12}\Gamma^3_{12} + g_{22}\Gamma^3_{11}) \ , \tag{8}$$

$$\Gamma^3_{11} = \frac{1}{2g}[G_5 \frac{\partial g_{11}}{\partial \xi} + G_6(2 \frac{\partial g_{12}}{\partial \xi} - \frac{\partial g_{11}}{\partial \eta}) + G_3(2 \frac{\partial g_{13}}{\partial \xi} - \frac{\partial g_{11}}{\partial \zeta})] \ , \tag{9a}$$

$$\Gamma^3_{12} = \frac{1}{2g}[G_5 \frac{\partial g_{11}}{\partial \eta} + G_6 \frac{\partial g_{22}}{\partial \xi} + G_3(\frac{\partial g_{13}}{\partial \eta} + \frac{\partial g_{23}}{\partial \xi} - \frac{\partial g_{12}}{\partial \zeta})] \ , \tag{9b}$$

$$\Gamma^3_{22} = \frac{1}{2g}[G_5(2 \frac{\partial g_{12}}{\partial \eta} - \frac{\partial g_{22}}{\partial \xi}) + G_6 \frac{\partial g_{22}}{\partial \eta} + G_3(2 \frac{\partial g_{23}}{\partial \eta} - \frac{\partial g_{22}}{\partial \zeta})] \ , \tag{9c}$$

$$G_5 = g_{12}g_{23} - g_{13}g_{22} \ , \tag{10a}$$

$$G_6 = g_{12}g_{13} - g_{11}g_{23} \ , \tag{10b}$$

$$g = g_{33}G_3 + g_{13}G_5 + g_{23}G_6 \ , \tag{10c}$$

and G_3 has already been defined earlier in (3d).

A successful program of calculations based on the set of Eqs. (5) - (10) now rests on how effectively one can devise a calculation method for the first partial derivatives $\underset{\sim}{r}_\zeta = (x_\zeta, y_\zeta, z_\zeta)$ in the field. In this connection we first note that based on the prescribed values $\underset{\sim}{r}_B(\xi,\zeta)$, $\underset{\sim}{r}_\infty(\xi,\zeta)$ the partial derivatives with respect to ξ and ζ of any order can be evaluated on the given bodies. Thus we must somehow connect the evaluation of $\underset{\sim}{r}_\zeta$ in the field with the partial derivatives in the surface. To maintain the intrinsic geometrical properties of the ζ-lines in the field with the ζ-lines of the inner and the outer boundaries, we consider the differential equations for the surfaces η = const. Following the method in Warsi[1,3] we find that the coordinates ξ, ζ in any surface (including the given boundaries) must satisfy the equations

$$g_{33}\underset{\sim}{r}_{\xi\xi} - 2g_{13}\underset{\sim}{r}_{\xi\zeta} + g_{11}\underset{\sim}{r}_{\zeta\zeta} + (G_2\Delta^{(2)}_2\zeta)\underset{\sim}{r}_\zeta = G_2(k^{(2)}_1 + k^{(2)}_2)\underset{\sim}{n}^{(2)} \ , \tag{11}$$

where the enclosed superscript (2) in Eq. (11) means that all the quantities have been evaluated on the surface η = const. Also

$$G_2 = g_{11}g_{33} - (g_{13})^2 \ , \tag{12a}$$

$$G_2(k_1+k_2) = g_{33}U^{(2)} - 2g_{13}T^{(2)} + g_{11}S^{(2)} , \tag{12b}$$

$$U^{(2)} = \underset{\sim}{n}^{(2)}\cdot\underset{\sim}{r}_{\xi\xi} , \quad T^{(2)} = \underset{\sim}{n}^{(2)}\cdot\underset{\sim}{r}_{\xi\zeta} , \quad S^{(2)} = \underset{\sim}{n}^{(2)}\cdot\underset{\sim}{r}_{\zeta\zeta} \tag{13}$$

and

$$\underset{\sim}{n}^{(2)} = (X^{(2)}, Y^{(2)}, Z^{(2)}) ,$$

where

$$X^{(2)} = (y_\zeta z_\xi - y_\xi z_\zeta)/\sqrt{G_2} , \tag{14a}$$

$$Y^{(2)} = (x_\xi z_\zeta - x_\zeta z_\xi)/\sqrt{G_2} , \tag{14b}$$

$$Z^{(2)} = (x_\zeta y_\xi - x_\xi y_\zeta)/\sqrt{G_2} . \tag{14c}$$

It may be noted that $(k_1+k_2)/2$ is the mean curvature and S, T, U are the coefficients of the second fundamental form of the surface η = const.

Based on Eq. (11), we now formulate the following weighted integral formula for the evaluation of $\underset{\sim}{r}_\zeta$ in the field.

$$\underset{\sim}{r}_\zeta = \int [f_1(\eta)(\underset{\sim}{r}_{\zeta\zeta})_B + f_2(\eta)(\underset{\sim}{r}_{\zeta\zeta})_\infty]\, d\zeta , \tag{15a}$$

where

$$(\underset{\sim}{r}_{\zeta\zeta})_{B,\infty} = [\frac{G_2}{g_{11}}(k_1^{(2)}+k_2^{(2)})\underset{\sim}{n}^{(2)} + \frac{2g_{13}}{g_{11}}\underset{\sim}{r}_{\xi\zeta} - \frac{g_{33}}{g_{11}}\underset{\sim}{r}_{\xi\xi}$$

$$-(\frac{G_2}{g_{11}}\Delta_2^{(2)}\zeta)\underset{\sim}{r}_\zeta]_{B,\infty} , \tag{15b}$$

and

$$f_1(\eta_B) = 1 , \; f_1(\eta_\infty) = 0 , \; f_2(\eta_B) = 0 , \; f_2(\eta_\infty) = 1 . \tag{15c}$$

The functions $f_1(\eta)$ and $f_2(\eta)$ must satisfy the conditions (15c) and should be chosen to reflect the effect of the coordinate redistribution function Q appearing in Eq. (6). It is also to be noted that the coordinate ζ need not in general satisfy the Beltrami equation. That is, in general $\Delta_2^{(2)}\zeta \neq 0$.

Numerical solution of the equations

The numerical method used in this research for solving the system of Eqs. (5) is the method of finite difference using the point-SOR. First the coordinates ξ and ζ for both the inner surface ($\eta = \eta_B$) and the outer surface ($\eta = \eta_\infty$) are to be generated using the available x, y, z values for these surfaces either analytically or by a computer program developed by Craidon.[4] In this research we have used both the analytical methods where possible, and also the subroutine in Ref. 4 to generate the given body surface coordinates, with equal success. Three practical problems have to be resolved before an effective solution algorithm for Eqs. (5) can be developed. They are: (i) a specification of the functions $f_1(\eta)$ and $f_2(\eta)$ appearing in Eqs. (15), (ii) specification of the redistribution functions (concentration or expansion functions) P and Q, and (iii) a method to obtain the same coordinates on the inner and outer boundaries. We now discuss each problem in succession.

(i) Before discussing the specifications of $f_1(\eta)$ and $f_2(\eta)$ we may state that each value like $\eta = \eta_B$ and $\eta = \eta_\infty$ is a parameter to start with rather than an integer. The difference $\eta_\infty - \eta_B$ is the most important difference and is known as the "modulus of the domain." The determination of $\eta_\infty - \eta_B$ is a formidable problem in three dimensions but fortunately there is no need for it in the case of numerical coordinate generation. Writing

$$\bar{Z} = \frac{\eta_\infty - \eta}{\eta_\infty - \eta_B} , \tag{16a}$$

we find that the function f_1 defined in (15c) should be a function of $\bar{Z}$ only, so that

$$f_1(1) = 1 , \quad f_1(0) = 0 , \tag{16b}$$

and

$$f_2(\bar{Z}) = 1 - f_1(\bar{Z}) . \tag{16c}$$

In the present computations we have taken f_1 and f_2 as linear functions of $\bar{Z}$, that is

$$f_1(\bar{Z}) = \bar{Z}. \tag{17a}$$

Other simple possibilities which have been tried are

$$f_1(\bar{Z}) = 0.5(\bar{Z}+\bar{Z}^2) \ , \tag{17b}$$

$$f_1(\bar{Z}) = (1-e^{-\bar{Z}})/(1-e^{-1}) \ . \tag{17c}$$

and these produce the same overall results as Eq. (17a). Many other higher order polynomials in $\bar{Z}$ can also be tried for $f_1(\bar{Z})$.

(ii) In this research we have experimented only with a contraction of η-lines near the inner body surface. The details on this aspect have already been given in earlier papers[1,3] (refer to §3.4 of Ref. 1). In the new coordinates* $(\bar{\xi},\bar{\eta})$, the function $\eta(\bar{\eta})$ is taken as follows.

$$\eta(\bar{\eta})-\eta_B = \frac{(\eta_\infty-\eta_B)(\bar{\eta}-\bar{\eta}_B)}{\bar{\eta}_\infty-\bar{\eta}_B} \ \kappa^{(\bar{\eta}-\bar{\eta}_\infty)} \tag{18}$$

where $\kappa = 1$ corresponds to no contraction while $\kappa > 1$ produces sufficient contraction. From (18) the function Q appearing in the operator (6) is obtained as

$$Q = \bar{g}_{11}\frac{d\eta}{d\bar{\eta}} \Big/ \frac{d^2\eta}{d\bar{\eta}^2} = \frac{\bar{g}_{11}[2+(\bar{\eta}-\bar{\eta}_B)\ell n\kappa]\ell n\kappa}{1+(\bar{\eta}-\bar{\eta}_B)\ell n\kappa} \ . \tag{19}$$

As is seen from (19), the function Q does not depend on the original parametric difference $\eta_\infty-\eta_B$ and therefore we can now treat $\bar{\eta}$ (and so also $\bar{\xi}$) as integers for the purpose of enumeration from one node point to the next.

(iii) Before the start of any numerical or analytic solution program it is important to establish a unique correspondence between the points of the inner and the outer boundary surfaces. That is, the Cartesian coordinates $\underset{\sim}{r}_B$ and $\underset{\sim}{r}_\infty$ of the inner and the outer boundaries respectively must be expressed in the <u>same</u> coordinates ξ, ζ (for η = const.). In symbolic form

$$\underset{\sim}{r}_B = \underset{\sim}{\phi}(\xi,\zeta) \ ,$$

$$\underset{\sim}{r}_\infty = \underset{\sim}{\psi}(\xi,\zeta) \ .$$

*The Eqs. (5) are now considered to be in $\bar{\xi},\bar{\eta}$ coordinates.

Though for convex surfaces the method of spherical projection seems to be most desirable, we have for the present investigations, used the geometrical method of first surrounding the inner body by a sphere of diameter equal to the major length of the inner body. Next, each point (x,y_B, z_B) on the inner body is projected to a point (x,y_s, z_s) on the sphere surrounding the inner body. The correspondence between the inner and outer body is then made by extending a straight line from the center through (x,y_s, z_s) to a point $(x_\infty, y_\infty, z_\infty)$ on the outer sphere.

A number of program runs have been made for prolate ellipsoids of various thicknesses surrounded by sphere of large radii. Also a thin body of revolution with circular sections, resembling the fuselage of an airplane, surrounded by a sphere has been considered. These numerical results with and without coordinate concentration are shown in Figs. 2-7.

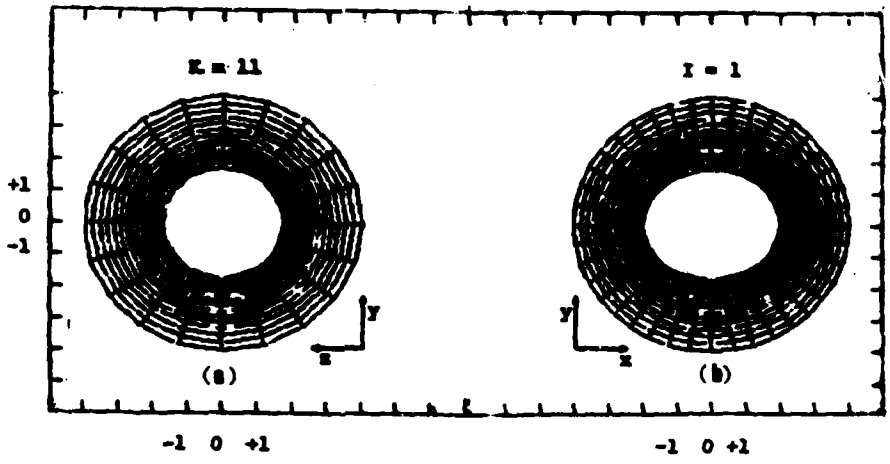


Figure 2. Inner body a thick prolate ellipsoid with major axis 2 and minor axis $\sqrt{3}$ surrounded by a sphere of radius 4. (a) Coordinate contours for a section ζ = const. (K = 11) for all (ξ,η) or (I,J) values, (b) for a section ξ = const. (I = 1) for all (η,ζ) or (J,K) values. In both cases no contraction in η, $\kappa = 1$.

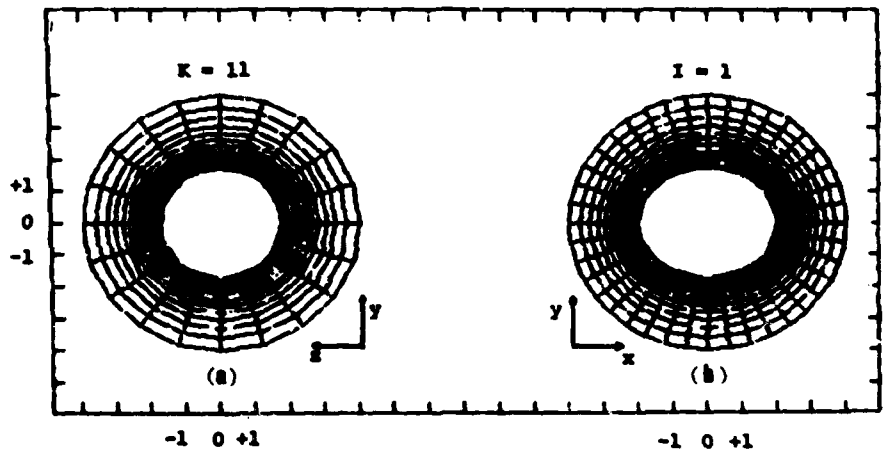


Figure 3. Cases (a) and (b) of Fig. 2, with contraction in η, $\kappa = 1.05$.

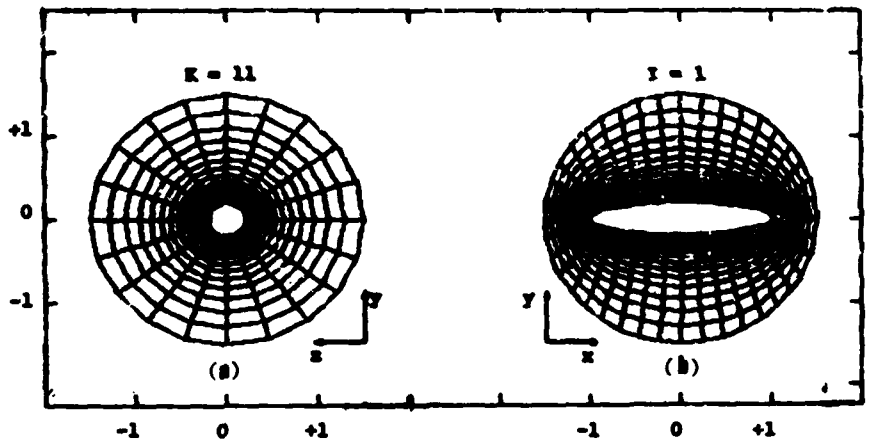


Figure 4. Inner body a thin prolate ellipsoid with major axis 1 02, minor axis 0.201 surrounded by a sphere of radius 1.5. (a) Coordinate contours for a section ζ = const. (K = 11) for all (ξ,η) or (I,J) values, (b) for a section ξ = const. (I = 1) for all (η,ζ) or (J,K) values. In both cases no contraction in η, $\kappa = 1$.

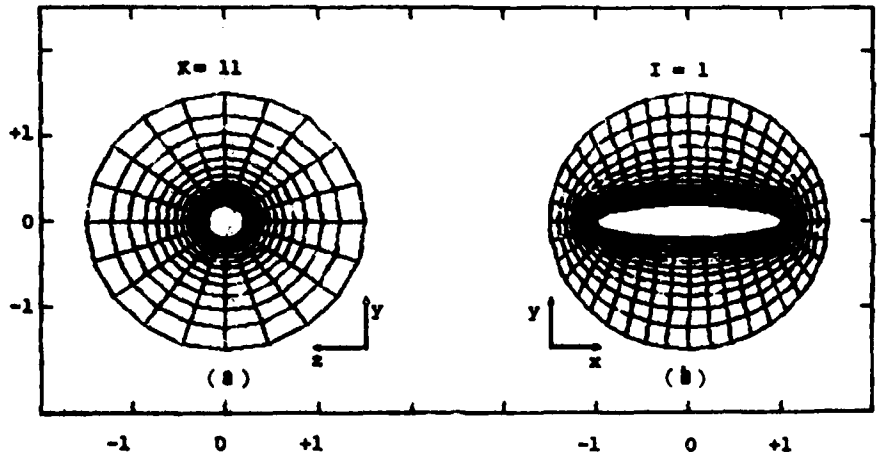


Figure 5. Cases (a) and (b) of Fig. 4, with contraction in η, $\kappa = 1.02$.

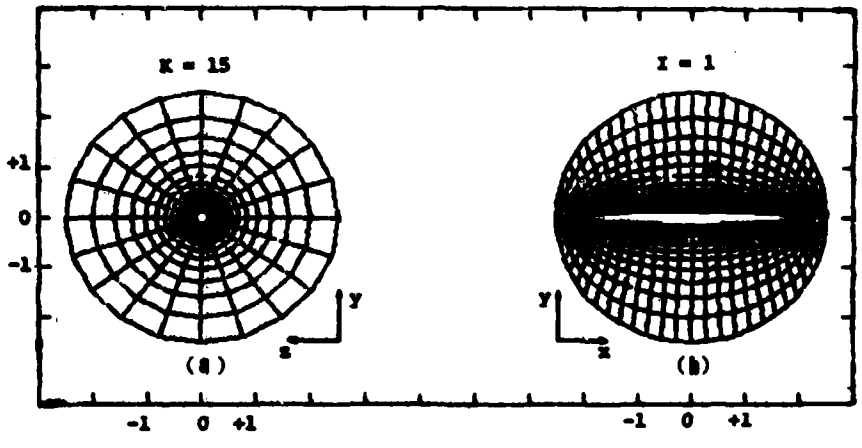


Figure 6. Inner body a thin body of revolution with circular sections having major axis 2 and minor axis 0.1313 surrounded by a sphere of radius 2.5. (a) Coordinate contours for a section ζ = const. (K = 15) for all (ξ,η) or (I,J) values, (b) for a section ξ = const. (I = 1) for all (η,ζ) or (J,K) values. In both cases no contraction in η, $\kappa = 1$.

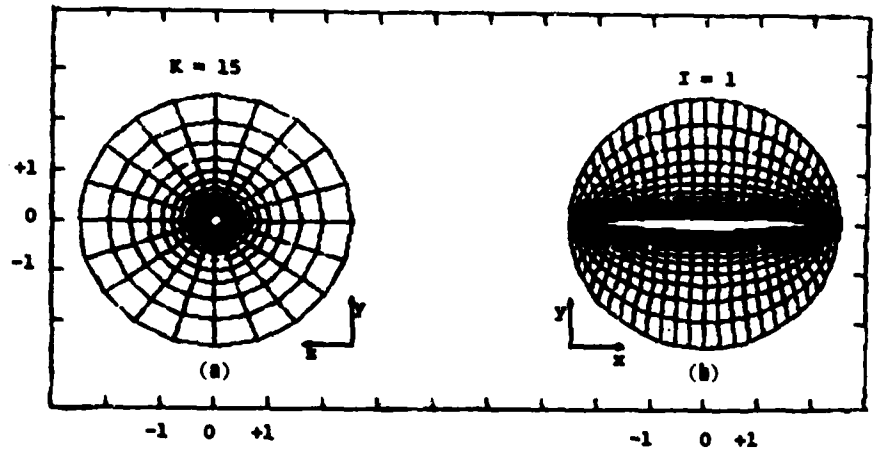


Figure 7. Cases (a) and (b) of Fig. 6, with contraction in η, $K = 1.005$.

CONCLUSIONS

This paper has been devoted to the numerical solution of a set of elliptic equations for the purpose of numerically evolving a series of surfaces and the intersecting surfaces in arbitrary three-dimensional regions in space. The most difficult part of such a program is the generation of surfaces between any two given surfaces. This has been considered here for thick and thin prolate ellipsoids and a body of revolution forming the inner bodies and a sphere forming the outer boundary. Many successful numerical algorithms can be developed using the proposed equations as the core equations for providing the coordinates around a complete aircraft and other aerodynamical shapes.

ACKNOWLEDGMENTS

This research has been supported by the Air Force Office of Scientific Research under the Grant No. AFOSR 80-0185, which the authors gratefully acknowledge.

REFERENCES

1. Warsi, Z. U. A. (1982) These proceedings.
2. Eisenhart, L. P. (1947) An Introduction to Differential Geometry with Use of the Tensor Calculus. Princeton University Press, Princeton, N. J.
3. Warsi, Z. U. A. (1981) Tensors and Differential Geometry Applied to Analytic and Numerical Coordinate Generation. Report MSSU-EIRS-81-1, Mississippi State University Engineering and Industrial Research Station.
4. Craidon, C. B. (1975) NASA TMX-3206.

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

SEMIDIRECT/MARCHING SOLUTIONS AND ELLIPTIC GRID GENERATION

PATRICK J. ROACHE[+]

[+]Ecodynamics Research Associates, Inc., P.O. Box 8172, Albuquerque, New Mexico, USA

INTRODUCTION

This paper describes the use of semidirect/marching methods both for the generation of two-dimensional grids using the elliptic generating equation approach, and for the solution of electric field problems in those coordinate systems. The efficiency of the semidirect/marching methods makes possible interactive design of the electrodes for electron beam lasers using a modest computer. Also described are the applications to the elliptic grid generation problem of computer Symbolic Manipulation.

SEMIDIRECT/MARCHING METHODS: THE GEM CODES

Semidirect methods are rapid finite difference methods for solving various steady-state and slowly varying time-dependent nonlinear problems. Fast elliptic solvers are used to solve linearized equations directly, which are then iterated to solve the nonlinearity. Applications of semidirect methods to problems, many in fluid dynamics, are given by Roache[1]. For the nonseparable partial differential equations of interest here, the fast elliptic solver used is some variation of marching methods for elliptic equations. The algorithms involved have been described in detail[2] and timing and accuracy tests of a particular software realization of the marching methods, called the GEM codes, have been reported[3].

Although stabilizing schemes exist[2,3], as a practical matter the marching methods depend on a favorable cell aspect ratio $\Delta\xi/\Delta\eta$ to stabilize the inherently unstable spatial marching procedure. They are thus well suited to problems with a grid refinement in one coordinate in the transformed plane. *The marching methods are not suitable for problems in which there is a significant grid refinement in both coordinate directions in the transformed plane.* However, for many practical problems, they are very well suited.

The advantages of the marching methods are their generality and speed. Unlike "fast Poisson solver" algorithms such as FFT or odd-even reduction, the marching methods (1) do not depend on separability of the coefficients, and (2) can treat the 9-point operator directly, even for nonseparable stencils.

Both these advantages are pertinent to non-orthogonal grid problems, not only in the solution of the grid by elliptic pde's, but also in the solution of the "hosted equations" [4](in the present case, the electric field equations) no matter how the non-orthogonal grid is generated. As for speed, the marching methods will initialize in order (M^3) operations for an M×M cell problem, and will solve repeat solutions in the optimal order (M^2) operations. For large two-dimensional problems using a 5-point operator, repeat solutions by actual timing tests are obtained[3] in the equivalent of 2 point SOR iterations.

APPLICATION TO ELLIPTIC GRID GENERATION EQUATIONS

The semidirect/marching methods are particularly well suited to the solution of the elliptic grid generating equations pioneered by Thompson et al.[5], provided that the cell aspect ratios are favorable. In this system, two coupled nonlinear equations are solved in the transformed plane (ξ,η) for the physical coordinates (x,y).

$$L(x) = 0,\ L(y) = 0$$

where, for e = x or y,

$$L(e) \equiv \alpha e_{\xi\xi} - 2\beta e_{\xi\eta} + \gamma e_{\eta\eta}$$

The coefficients are nonlinear functions of x and y. See Thompson, et al.[5] for details, and for the use of additional terms P and Q for coordinate adjustments. In the semidirect approach, the equations ate first linearized about some initial guess for the grid, giving values of α^o,β^o, etc. We then solve a sequence of linear problems, indicated by

$$L^o(e^k) = S(e^{k-1})$$

where L^o is based on the initial guess α^o, β^o, etc. and S is defined by

$$S(e) \equiv -\{(\alpha-\alpha^o)e_{\xi\xi} - 2(\beta-\beta^o)e_{\xi\eta} + (\gamma-\gamma^o)e_{\eta\eta}\}$$

If immediate updating of the coefficients were used (true Picard method), the coefficients in L^o would be re-evaluated at each iteration and the GEM solution would be reinitialized, requiring order (M^3) operations for each iteration. Instead, we attempt a single initialization of the GEM code using a quasi-Picard method. Depending on the adequacy of the initial guess, this single initialization may be adequate, or we may require reinitialization during the solution process. The decision to reinitialize is automated and is based on the requirement for at least an 80% reduction in the maximum change in x and y at each iteration. Also, in the interactive design process, the initialization from a previous design (i.e. a previous laser electrode geometry) can be used for the next grid generation.

The semidirect/marching methods are particularly well suited to this problem

of elliptic grid generation for two reasons. First, although two coupled nonlinear equations are used, there is only one matrix for the two equations. Thus, only one matrix initialization is used, and only one set of coefficients must be stored. Second, although the equations are nonlinear and coupled, they are not coupled in the boundary conditions. This adds to the speed of the iterative convergence process. (For the Navier-Stokes equations, the coupling of the boundary conditions leads to time-like iterative behavior, which is comparatively slow[1,6].)

ACCURACY AND TIMING TESTS

For moderate geometries, the semidirect/marching methods give solutions for the grid in typically 8 to 10 iterations, requiring less than 4 seconds on a CDC 6600 for a 31×31 grid with poor initial guesses. We use an unusually tight convergence criterion of $\delta x, \delta y < 10^{-5}$, because we are interested in using Richardson extrapolation to fourth order accuracy for the solutions of the hosted equations. This requires no oscillations in the solution for either the coordinate system or the hosted equations[7]. The number of iterations required is not a strong function of grid size, and the marching error is tolerable for most problems encountered so far (of the order 5×10^{-6} for a 31×61 grid). As yet, we have had no experience with coordinate system control using the P and Q terms[5]. Fortunately, many geometries of practical interest to the electrode design area do not require additional coordinate control, and the present code is being used for interactive computer design of several laser systems.

The electric field solutions are also obtained with the semidirect/marching methods once the coordinate system has been generated. The equation solved is

$$\nabla\cdot\sigma\nabla\phi = 0, \quad \sigma = \sigma(E), \quad E = \nabla\phi$$

where ϕ is the electric potential, σ is the conductivity, and E is the electric field strength. For linear field equations (σ not a function of E) with 1-point or 2-point derivative boundary conditions, the equations are solved directly. For the nonlinear field equations and for 3-point derivative boundary conditions, iteration is required. A representative problem is solved in the order of 10 iterations, requiring less than 5 seconds on a CDC 6600. However, we have encountered nonlinearities in σ which required 50 iterations.

The linear problem is of practical interest, and has been used as an accuracy test by comparison of the computed results with those of the Rogowski electrodes, obtained by conformal transformation methods. With boundary points equidistributed in arc length, we predict the E-field to plotting accuracy in

a 25×25 grid. Using a distribution of boundary points weighted by surface curvature, we have obtained plotting accuracy in a 13×13 grid.

It appears that a good multigrid code using nonlinear grid interpolation (FAS) can achieve the same level of efficiency as the semidirect/marching methods for the nonlinear problems[8]. For the linear problem, the marching methods as embodied in the GEM codes are *the* fastest. However, they are limited in resolution to about a 100×100 grid with favorable cell aspect ratios. More importantly, they are attractive in 3 dimensions only for problems which are separable in the third coordinate so that a FFT can be used.[2] The marching methods appear to vectorize well, especially for repeat solutions, for the 5-point operator. On a vector machine, 9-point operators would be best treated iteratively by lagging, as is customarily done with linear iterative methods. The vectorizing of multigrid codes is an open question at this time. The comparison of marching methods, multigrid methods and the simpler fully vectorizable iterative methods (such as hopscotch SOR) on vector machines will be a complicated job, dependent on the particular machine architecture, the problem size, and the coding details.

CONTINUATION METHODS FOR DIFFICULT GEOMETRIES

Good initial conditions for the grid can be a problem, whether the grid generating equations are solved by semidirect/marching methods or by more conventional iterative methods. Particularly, for slit-like geometries, initial conditions obtained by simple interpolation in the transformed plane can give crossed coordinate lines and negative Jacobians, which can prevent iterative convergence of the nonlinear problem.

We have developed two continuation methods for this problem. Both attain the final solution in N continuation steps (where N is selected by the code user). The weighting function W varies from 0 to 1 for the sequence of problems,

$$W = 0,\ 1/N,\ 2/N,\ \ldots.(N-1)/N,\ 1.$$

The first continuation method builds up to the true boundary conditions. With B = x and y boundary conditions, the continuation method is

$$B^k = (1-W)B^o + W \cdot B^{true}$$

where B^o is some trivial initial geometry, such as a rectangle.

The second method builds up to the true generating equations, and was suggested by Maliska's work[9] using point SOR for the solution. The coefficients α, β, and γ are built up from

$$\beta^k = W \cdot \beta^{true},\quad A^k = (1-W) + W \cdot A^{true} \text{ where } A = \alpha \text{ and } \gamma.$$

This starts from the linear, decoupled problem

$$x_{\xi\xi} + x_{\eta\eta} = 0$$
$$y_{\xi\xi} + y_{\eta\eta} = 0$$

We have had success with both methods, but the second is preferable. It is more systematic, and avoids some clumsy scaling problems of the first. For a rather severe slit-like laser geometry, only two continuation steps were required to solve the grid.

SOLUTION OSCILLATIONS NEAR GRADIENT BOUNDARIES

An illuminating behavior arose in the application of symmetry boundary conditions to the hosted electric field equations. For symmetry at $\xi = 0$, the transformed equation requires

$$\frac{\partial\phi}{\partial n} = (\alpha\phi_{\xi} - \beta\phi_{\eta})/J\alpha^{\frac{1}{2}} = 0, \text{ where } J = \text{Jacobian.}$$

The marching code GEM requires one-sided differences for ϕ_{η} because the boundary conditions must be separable in the march direction. Depending on the curvature at the boundary (the sign of β) and the march direction, this can be analogous to *downwind* differencing along the boundary, and can produce oscillations in the solution of the hosted equations. In analogy with the well known fluid dynamics problems, we would anticipate that other workers may have encountered this behavior using centered differences for ϕ_{η}.

The cure, which almost certainly has been applied in practice elsewhere although not reported (nor perhaps recognized) is to have a nearly orthogonal grid near symmetry and other gradient boundaries, giving $\beta \simeq 0$. (One could also set $\beta = 0$ by reflection[9] but this gives a discontinuity in the grid which will slow the truncation error convergence.)

In the GEM solutions, true second-order accuracy is obtained by a deferred correction approach, lagging the difference between the one-sided and centered forms for ϕ_{η}. It is even more robust, for geometries in which β might change sign along the boundary, to lag the entire ϕ_{η}, along with the deferred correction for the 3-point ϕ_{ξ} and any nonlinearities, and this is now our standard procedure. Note, however, that the GEM code now cannot be considered a direct method for gradient boundary conditions in a non-orthogonal grid.

SENSITIVITY TO CROSS DERIVATIVES

We have generally been impressed with the difficulty of code verification for general non-orthogonal coordinate problems. In particular, the experience related here violated out intuition on the sensitivity of the solutions to

the cross derivative terms like $x_{\xi\eta}$, $\phi_{\xi\eta}$ etc. The experience arose from a coding error in which the cross derivative terms were all calculated a factor of 2 larger than correct. The error was not detected early because the solutions looked good for mild but non-trivial geometries. For electrodes in a quadrant where the lower electrode was described by a $\cos^{\frac{1}{2}}$ curve and the upper electrode by $\cos^{\frac{1}{4}}$, the grid generated and the solution for the E-field were quite accurate. Likewise, the solution for the Rogowski electrode differed by only 0.4% from the exact ϕ, using only a 13×13 grid. However, in systematic convergence testing (performed by H. Happ of Tetra Corporation), the error did not reduce as the grid was refined. The coding error was detected and corrected, and the previous cases were re-calculated. The factor of 2 error in the cross derivatives proved to affect the coordinate generation by less than 0.01% in the location of any x and y of the grid nodes, and to affect the E-field (derivative of the ϕ solution) by 0.016%. The conclusion might seem obvious, that the solutions are very insensitive to the cross derivatives. However, this is actually quite problem dependent. For a slit-like geometry, the coding error seriously affected the grid generation. Iterative convergence was obtained only with the extreme of 20 continuation steps plus the use of extensive under-relaxation of boundary and interior points. The resulting "mesh" was a mess, with coordinate lines that crossed and extended outside of the physical domain, violating the maximum principle. When the coding error was corrected, the method converged to a perfectly good grid in 2 continuation steps. For this class of problems, we conclude that the grid generation process is highly sensitive to the cross-derivatives. Aside from coding errors, this experience also seems to bear on the robustness of alternate elliptic generating systems which use simpler equations in the transformed plane.

SYMBOLIC MANIPULATION AND GRID GENERATION

Coding errors such as the one described above plague all computational work, and the chance for error increases as the complexity of the problems increase. As noted above, we have been impressed with the difficulty of code verification for the transformed grid problems. We have also been impressed with the complexity of the 3-dimensional equations for general non-orthogonal grids.

In association with Prof. Stanly Steinberg of the University of New Mexico, we are addressing this and related problems using computer Symbolic Manipulation. These are not floating-point calculations, but symbolic operations, e.g. the chain rule differentiation, performed by computer logic. The gathering of coefficients is likewise done symbolically, as is the actual

writing of the Fortran subroutines to define the problem. The symbolic code used is a VAX computer version of the code MACSYMA developed over many years at the MIT Lincoln Laboratories.

To recapitulate: we are using MACSYMA to (1) analytically generate the transformation equations, and (2) to actually write a Fortran subroutine to produce the 9-point stencil defining the matrix problem.

Once the computer has written the subroutine defining the problem, the coefficient matrices defining the stencil are passed to some canned solver, in this case the GEM codes. Both the grid generation problem and the hosted equation are solved the same way. Except for input/output and processing of the results, as well as the passing of the matrix problem to the canned solver, the user obtains the answer without writing Fortran or similar code.

The general second-order two-dimensional equation has been solved in this manner, and the results verified by comparison to the hand-coded coefficient matrices. The analytic generation of the transformation equations and the writing of the Fortran subroutine require about 10 minutes on a VAX 780. The three-dimensional problem has also been solved, but the computer time increases dramatically due to the computational complexity of the chain rule operations, similar to the classic "sorting" problem. We are currently involved in the code verification. Rather than generate a hand-coded version, we will obtain three-dimensional solutions of the algebraic equations (using a hopscotch SOR "canned" solver) and verify the code by convergence testing to the exact solution of highly stretched coordinate problems.

In the near future, we intend to work on the relatively straight-forward problems of multiple equations, higher order equations, perturbation terms in the source term formulated so as to give deferred corrections to higher order accuracy and/or nonlinear terms, and validation of all these.

More difficult problems are conservation forms, upwinding (or other conditional differencing), complicated boundary conditions (currently we have used only Dirichlet conditions), and optimization. It is likely that the Fortran code generated will always be less efficient than what could be obtained with expert hand coding. This situation is viewed as analogous to the situation of efficiency attainable from high-level languages like Fortran vs. assembly language. The "efficiency" sought is not that measured by CPU seconds for code execution, but by calendar years for code development.

Human errors are still possible in this process, but they are a different level of error. Grand mistakes will occur, but not the petty ones of writing S(I+1,J) when the term should have been S(I-1,J), etc.

The following areas of application for Symbolic Manipulation appear most promising.

(1) Combination of perturbation methods and numerical methods. These "semianalytic" approaches have already been used with some success, and are not difficult for regular perturbation problems. With insight, they can be used for singular perturbation problems, and could be used in general grid problems to remove grid-introduced singularities.

(2) Coordinate transformations, especially in conjunction with (3).

(3) Constitutive equation testing, in areas like turbulence modeling, non-newtonian fluids, soil mechanics, gravitational theory.

(4) Generation and analysis of new discrete forms via finite difference, finite element, least squares, etc. methodologies.

The prospect of virtually error-free testing of constitutive equations and difference forms is most attractive. I predict that the use of Symbolic Manipulation in these and other problems will shortly be recognized as the way of the future, and that the practice or disciplines like computational fluid dynamics will be revolutionized in the next decade as the power of Symbolic Manipulation becomes widely recognized.

FUTURE WORK

Besides the use of Symbolic Manipulation described above, we expect to extend the work described herein in the near future to include the following: unsteady equations, 3 dimensional problems, magnetic effects (which give rise to a tensor conductivity), dielectric interior boundaries (which require the precise control of the grid at interior points), solution adaptive methods to better resolve the maxima in the E-fields, and semi-automated optimization of the electrode design procedure.

ACKNOWLEDGEMENTS

H. Happ and D. Harrison of Tetra Corporation have provided programming support and code validation efforts. Prof. Stanly Steinberg is primarily responsible for the Symbolic Manipulation work. The work described herein has been partially supported by the U.S. Army Research Office, The U.S. Air Force Weapons Laboratory, and the U.S. Air Force Office of Scientific Research.

REFERENCES

1. Roache, P.J. (1982), Semidirect/Marching Methods for Partial Differential Equations, to appear.
2. Roache, P.J. (1978), Numerical Heat Transfer. Part 1;1, 1. Part 2;1, 163. Part 3;1, 183.
3. Roache, P.J. (1981), Numerical Heat Transfer, 4, 395.
4. Rubbert, P.E. and Lee, K.D., these Proceedings.
5. Thompson, J.F., Thames, F.C. and Mastin, C.W. (1974), Journal of Computational Physics, 15, 299.
6. Roache, P.J. (1978), Computers and Fluids, 3, 179.
7. Roache, P.J. (1981), Proc. Symposium on Numerical and Physical Aspects of Aerodynamic Flows, California State University at Long Beach, 19-21 January 1981.
8. Ghia, U. and Thames, F.C. (personal communications).
9. Maliska, C.R. (1981), A Solution Method for Three-Dimensional Fluid Flow Problems in Nonorthogonal Coordinates, Ph.D. Dissertation, Univ. of Waterloo.

AD P001000

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

2-D ELLIPTIC GRID GENERATION USING A SINGULARITY METHOD AND ITS APPLICATION TO TRANSONIC INTERFERENCE FLOWS.

KARL D. KLEVENHUSEN
Department of Theoretical Aerodynamics VFW Bremen, W.-Germany

NOMENCLATURE

A	square of incompressible velocities
B	square of compressible velocities
c	chord length
C	local speed of sound
C_L	lift coefficient
C_P	pressure coefficient
D	diameter of an inlet
m	doublet strength
M_∞	Mach number at infinity
$\dot{m}/\dot{m}^*$	mass flow ratio
S	boundary
u_∞	velocity at infinity
u, v	incompressible velocity components
u', v'	components of incompressible disturbance velocity
x, y	cartesian coordinates
x', y'	"disturbance coordinates"
α	angle of attack
φ, ψ	streamline coordinates
Φ	compressible potential function
$\Delta\varphi$	potential jump at the trailing edge

PREVIOUS PAGE IS BLANK

INTRODUCTION

It is well known that the solution of many mathematical problems can be simplified by the use of a carefully selected coordinate system. One example is the numerical treatment of boundary-value problems in partial differential equations. An optimal computational grid for a finite element or a finite difference method should be carefully adapted to the mathematical, numerical, physical and geometrical aspects of the problem. In such a case, one can expect a reduction of numerical errors, a reduction of computing time and, most important, an excellent physical presentation of the results. The present paper describes a grid generation procedure to be applied to transonic flows with interferences. The well known full transonic potential equation in cartesian coordiantes (x,y) is:

$$2\left(\phi_{xx}+\phi_{yy}\right)\cdot C^2-\phi_x B_x-\phi_y B_y=0 \qquad (1)$$

with $$B=\phi_x^2+\phi_y^2$$

Then streamline coordinates (φ,ψ) are introduced with the following properties:

$$\varphi_{xx}+\varphi_{yy}=0$$
$$\psi_{xx}+\psi_{yy}=0 \qquad (2)$$

$$\varphi_x=\psi_y$$
$$\varphi_y=-\psi_x \qquad (3)$$

where φ and ψ are conjuyated harmonic functions. It is known from complex analysis that every pair of conjugated harmonic functions in the x-y-plane (physical plane) are conjugated harmonic functions in the φ -ψ-plane (streamline plane). This feature of harmonic functions plays an important role in the development of the present grid generation procedure.

Equation (1) transformed to (φ,ψ) coordinates gives

$$2\left(\phi_{\varphi\varphi}+\phi_{\psi\psi}\right)\cdot C^2-\phi_\varphi B_\varphi-\phi_\psi B_\psi=0 \qquad (4)$$

with $$B=\left(\phi_\varphi^2+\phi_\psi^2\right)\left(\varphi_x^2+\varphi_y^2\right)$$

This equation contains the additional factor

$$\left(\varphi_x^2+\varphi_y^2\right)=A \qquad (5)$$

which can be interpreted as the square of an incompressible velocity.

To simplify equation (4) for practical application purposes the incompressible streamlines are chosen as a good approximation of the compressible streamlines. Then equation (4) can be reduced to

$$2\cdot\left(\phi_{\varphi\varphi}+\phi_{\psi\psi}\right)\cdot C^2-\phi_\varphi\cdot B_\varphi=0 \qquad (6)$$

with $$B=\phi_\varphi^2\cdot\left(\varphi_x^2+\varphi_y^2\right)$$

assuming all first derivatives normal to the streamlines are incremental and terms of second order can be neglected.

The new coordinates are orthogonal and body-fitting. The Neumann boundary condition

$$\phi_n=0 \qquad (7)$$

is transformed to

$$\phi_\psi=0 \qquad (8)$$

Equation (4) and (6) imply singular points at the stagnation points of the coordinates (A = 0) if the compressible stagnation points do not have the same location. A solution of that problem will be discussed later.

The outlined transformation of the transonic potential equation demonstrates: Choosing to use a coordinate system adapted to the special physical problem may lead to simplifications of the governing equations and consequently to computing time reduction. All transonic calculations for this paper using the present method are based upon equation (6).

GRID GENERATION BY SINGULARITY METHODS

Selection of a suitable singularity method

Singularity methods are classical tools in computational fluid dynamics. Since the basic work of Martensen [1] and Hess [2] a lot of singularity methods have been developed for calculating inviscid incompressible plane or spatial flows (panels methods). Most of these methods in two dimensions approximate the curved boundaries of an airfoil section in the simplest way by a polygon.

On each straight line of the polygon a constant or a linear or a higher order singularity distribution of sources and / or vortices and/or doublets are assumed. The boundary conditions can be satisfied as Neumann- or Dirichlet- conditions at the internal or external side of the boundaries.

Hence it appears that a lot of different singularity methods can be developed for the same flux problem. The user's problem is to find the most suitable method for his special case of application. Fig. 1 shows the selection procedure to find an appropriate singularity method for calculating streamline coordinates. In contrast to common flow computation, where the values of the velocities are of interest, the potential and the stream function must be determined as accurately as possible.

Consider a line segment of length l with an assumed constant source or constant vortex distribution. It is known, that at a great distance r the potential function of the source element and the streamfunction of the vortex element are proportional to ln r. This logarithmic behaviour may cause an increase of unavoidable numerical errors with increasing distance. Thus the method of source or vortex singularities is not suitable for calculating streamline coordinates. A more accurate calculation offers the method of doublet singularities with normal or tangential doublet axes. In constrast to the above method the values of the potential and the stream function decrease with increasing distance. Furthermore the doublet equations are obviously simpler than the equations of sources and vortices. This is important if short computing times have to be considered.

At the midpoint of a doublet line of length l the following relations are valid.

In the case of normal doublet axes the normal velocity φ_n is proportional to m/l and the streamfunction ψ is proportional to m'/l where m is the local doublet strength and m' is the local slope of m. In the case of tangential doublet axes φ_n is proportional to m' and the streamfunction ψ is proportional to m.

Hence one may infer that there is only one suitable singularity method using a doublet distribution with tangential doublet axes satisfying the Dirichlet condition of constant streamfunction ψ along the boundaries.

The singularity method in the physical plane

The incompressible flow around an airfoil of a piecewise smooth boundary can be described by a modified Dirichlet problem:

$$\begin{aligned} &\psi_{xx} + \psi_{yy} = 0 && \text{in the flow field} \\ &\psi = \text{const} && \text{at the boundaries of the airfoil} \\ &\psi = \psi_\infty && \text{at infinity} \end{aligned} \qquad (9)$$

Before solving the problem by a singularity method a perturbation stream function is introduced by

$$\psi' = \psi - \psi_\infty$$

and the Dirichlet problem is formulated as follows:

$$\begin{aligned} &\psi'_{xx} + \psi'_{yy} = 0 && \text{in the flow field} \\ &\psi' = \text{const} - \psi_\infty && \text{at the boundaries S} \\ &\psi' = 0 && \text{at infinity} \end{aligned} \qquad (10)$$

The solution of this problem is

$$\psi' = -\frac{1}{2\pi} \int_S m \cdot \frac{\partial}{\partial t} \ln r \, ds \qquad (11)$$

where t denotes the tangent of the boundary . The doublet strength m must be calculated from the integral equation

$$\psi_\infty - \text{const} = \frac{1}{2\pi} \int_S m \cdot \frac{\partial}{\partial t} \ln r \, ds \qquad (12)$$

along S. The solution is not unique and m can be chosen arbitrarily, at one arbitrary point of S. Appropriately zero doublet strenght is assumed at the lower side points of the trailing edges (see fig. 2).

Equation (12) can easily solved when assuming a linear distribution over each straight line of the polygon (fig. 2) and satisfying the condition at each midpoint. For the case of j straight lines j doublet strengths can be calculated at j corners of the polygon.

In the lifting case a slit with constant doublet distribution is necessary to satisfy the Kutta condition by appropriately assuming the same constant doublet distribution on the rearmost line segment on the upper side of the trailing edge (see fig. 2). This means a peacewise linear but continuous doublet strength can be calculated beginning at the lower point of the trailing edge going clockwise around the airfoil and then along a slit up to infinity. The direction of the slit can be chosen arbitrarily but it should not intersect the airfoil or any element in the case of a multi-element configuration. The conjugated harmonic function of ψ' is

$$\varphi' = -\frac{1}{2\pi}\int_S m \cdot \frac{\partial}{\partial n} \ln r \, ds \qquad (13)$$

with tne same doublet strength of formula (11) but with normal doublet axes. When the doublet distribution has been determined it is possible to calculate φ , ψ , φ_x , φ_y at any point of the physical plane.

In this way one can determine the boundary conditions for a boundary value problem in the streamline plane based on Laplace's equation. This is the first step of the present grid generation procedure.

The singularity method in the streamline plane

The transonic potential equation (6) requires the knowledge of the squares of the incompressible velocities φ_x and φ_y at every point(φ , ψ) of the streamline plane. Consider perturbation velocities

$$u' = \varphi_x - U_\infty \cdot \cos\alpha$$
$$v' = \varphi_y - U_\infty \cdot \sin\alpha \qquad (14)$$

where U_∞ is the velocity at infinity and α is the angle of attack. u' and v' are conjugated harmonic functions in the φ , ψ plane and vanish at infinity. Furthermore u' and v' are known along the upper and lower surface of an airfoil from the calculation in the physical plane (see fig. 3). The plane behind the airfoil is slitted along the body's streamline ψ_0 up to infinity.

The velocities at both sides of the slit are equal:

$$\begin{aligned} u'(\varphi, \psi_0 + 0) &= u'(\varphi - \Delta\varphi, \psi_0 - 0) \\ v'(\varphi, \psi_0 + 0) &= v'(\varphi - \Delta\varphi, \psi_0 - 0) \end{aligned} \tag{15}$$

where $\Delta\varphi$ is the potential jump in the lifting case. Now consider the airfoil in the streamline plane. The airfoil is mapped to a slit with well known boundary condition at the upper and lower side. The velocities at both sides are not identical in the lifting case due to equation (15). Two different types of boundary values problems for each velocity u' and v' can be established:

1. The velocities are known along the whole slit up to infinity by integration of the body's streamline in the physical plane. Then a Dirichlet problem can be formulated as follows:

$$u'_{\varphi\varphi} + u'_{\psi\psi} = 0 \quad \text{in the } \varphi, \psi \text{ -plane}$$

with the boundary conditions

$$\begin{aligned} &u' = u'_u \quad \text{at the upper side of the slit} \\ &u' = u'_l \quad \text{at the lower side of the slit} \\ &u' \longrightarrow 0 \quad \text{for} \quad (\varphi, \psi) \longrightarrow \infty \end{aligned} \tag{16}$$

2. The velocities are known only along the upper and lower surface of the airfoil. Then a modified Dirichlet problem can be formulated:

$$u'_{\varphi\varphi} + u'_{\psi\psi} = 0 \quad \text{in the plane}$$

with the boundary conditions

$$\begin{aligned} &u' = u'_u \quad \text{at the upper side of the airfoil} \\ &u' = u'_l \quad \text{at the lower side of the airfoil} \\ &\text{condition (15) along the slit behind the airfoil} \\ &u' \longrightarrow 0 \quad \text{for} \quad (\varphi, \psi) \longrightarrow \infty \end{aligned} \tag{17}$$

Analogous problems can be formulated for v' .

The solution of both problems (16) and (17) is:

$$u' = -\frac{1}{2\pi}\int_S m_\perp^{(u)} \frac{\partial}{\partial n} \ln r \, ds - \frac{1}{2\pi}\int_S m_\parallel^{(u)} \frac{\partial}{\partial t} \ln r \, ds \tag{18}$$

where S denotes the slit up to infinity. Presently, this solution was proved numerically only. It describes a double singularity distribution with doublets having normal and tangential axes, which is the same as a single doublet distribution with oblique doublet axes.

The normal doublet strength with respect to u' is $m_\perp^{(u)}$ and the tangential strength is $m_\parallel^{(u)}$. For v' one gets the result in the same way

$$v' = -\frac{1}{2\pi}\int_S m_\perp^{(v)} \frac{\partial}{\partial n} \ln r \, ds - \frac{1}{2\pi}\int_S m_\parallel^{(v)} \frac{\partial}{\partial t} \ln r \, ds \tag{19}$$

Since u' and v' are conjugated harmonic functions the following relations are valid

$$\begin{aligned} m_\perp^{(u)} &= m_\parallel^{(v)} \\ m_\parallel^{(u)} &= m_\perp^{(v)} \end{aligned} \tag{20}$$

It easily can be shown that $m_\perp^{(u)}$ and $m_\perp^{(v)}$ respectively indicate the potential jump along the slit S :

$$\begin{aligned} m_\perp^{(u)} &= u'(\varphi, \psi_0 - 0) - u'(\varphi, \psi_0 + 0) \\ m_\perp^{(v)} &= v'(\varphi, \psi_0 - 0) - v'(\varphi, \psi_0 + 0) \end{aligned} \tag{21}$$

Taking equations (21) and (20) the solution of boundary value problem 1 can be determined at every point(φ , ψ) of the streamline plane by solving the integrals of equation (18) and (19).

In the case of boundary value problem 2 the doublet strength along the slit behind of the airfoil can be determined from condition (15). For solving the problem numerically, the slit is divided into a finite set of line segments. A linear doublet distribution is assumed on each segment. But in contrast to the

singularity model of the physical plane the doublet strength is determined by satisfying the boundary conditions at the endpoints of each line segment. The slit behind the airfoil can be assumed of finite length, i.e. 10 chord lengths. The conditions (15) and (21) lead to a well behaved system of linear equations which determine the doublet strength along the slit.

This procedure replaces the integration of the body's streamline behind the airfoil in the physical plane and was used for all present calculations.

In concluding this chapter it seems clear that every pair of conjugated harmonic function of the physical plane can be treated in the same way as the perturbation velocities u' and v' if they are bounded at infinity.

In the case of the physical coordinates x, y it is necessary to define "disturbance coordinates" by

$$\begin{aligned} x' &= x - \varphi \cos\alpha + \psi \cdot \sin\alpha \\ y' &= y - \varphi \sin\alpha - \psi \cdot \cos\alpha \end{aligned} \tag{22}$$

At infinity x' makes a jump of $-\Delta\varphi \cos\alpha$ and y' a jump of $-\Delta\varphi \sin\alpha$ across the slit and the according doublet strengths does not vanish.

GRID GENERATION FOR TRANSONIC FLOW COMPUTATION

Grid spacing

An optimal computational grid should be carefully adapted to the mathematical, numerical, physical and geometrical aspects of the problem. The spacing of coordinate lines, for example is of paramount importance to resolve large gradients. A similar problem arises when approximating an airfoil shape by a polygon as indicated in fig. 2. At regions of greater curvature, i.e. at the leading edge, the spacing of the line segments of the polygon should be finer. This is well known from the application of singularity methods to subsonic flow computation problems. Taking the midpoints of the line segments of the polygon as mesh points was found to be an appropriate grid spacing method in φ-direction.

For the case of the far field the spacing was done by an exponential law ensuring an increasing mesh size. The far field boundary was assumed to be 3 to 5 times the chord length from the airfoil.

The stagnation point problem

When using the present streamline-type coordinate system the stagnation point is fixed numerically. This seems to be a great restriction of application and some critics of the method believe that such a coordinate system is not usable.

The physical transonic flow demonstrates the contrary.

Fig. 5 shows, the calculations around the modern supercritical airfoil Va2 were done by the method of Bauer-Garabedian-Korn-Jameson (BGKJ) [3] using an O-type coordinate system for determination of the stagnation point shift. The maximum lift coefficient $C_{L\ max}$ indicated by the dashed line, was found experimentally [4]. This indicates that lift coefficients greater than $C_{L\ max}$ are physically non existent. From the graphic presentation at the right side of the figure it can be seen, that for all realistic lift coefficients the stagnation point shift is smaller than indicated by the dashed line. The shift itself is not detectable by the BGKJ-method when using the standard number of 160 mesh points around an airfoil which gives a minimum mesh size of 0,4 % of chord length.

In fig. 6 the computational results of the present method agree quite well with the results of the BGKJ-method.

In addition, a stagnation point shift may occur due to boundary layer effects. This effect could be taken into account by using the singularity method in the physical plane. In this case an additional singularity distribution of sources would be necessary. But again it was found, that there is no need of a grid correction [5].

On the other side, when the method of a plane streamline grid is applied to rotational transonic flow calculations, there may be a need of corrections.

In the case of a flow around a plane inlet and a rotational inlet with the same boundary conditions the stagnation points of both flows do not match. But if the internal velocity v_p of the

plane inlet is assumed as

$$v_p = \sqrt{v_r} \tag{23}$$

where v_r is the internal velocity of the rotational inlet then the plane and the rotational stagnation points have approximately the same location.

Fig. 7 shows the computational grid around an inlet. The results of transonic flow calculations agree quite well with experimental data (fig. 8).

The numerical treatment of the compressible stagnation points does not cause significant difficulties when using a finite difference method. A numerical singularity at this point can be avoided when solving Laplace's equation instead of the full transonic potential equation. But in some cases the residue at the stagnation point converges to a nonvanishing value. This phenomenon was not analysed in more details because it was found that there is no significant effect to the expected solution.

Application to transonic interference flows

The present grid generation procedure was developed for application to transonic interference flows such as a flow around a multi-element airfoil. Fig. 9 shows the grids around an airfoil with a slit in the physical plane and in the computational plane. In fig. 10 present results are compared with results of Arlinger's method [6].

An other important range of application is the calculation of windtunnel interferences. Fig. 11 shows the grid around a NACA 0012 airfoil between walls. In Fig. 12 calculated pressure distributions are compared with each other. In the case of vanishing free stream Mach number wall interferences are neglectable. However for the case of higher Mach number .75 the windtunnel walls cause a shift of the shock position of about 10 % rearwards.

REFERENCES

1. Martensen, E.
Die Berechnung der Druckverteilung an dicken Gitterprofilen mit Hilfe von Fredholmschen Integralgleichungen zweiter Art

Mitteilungen aus dem Max-Planck-Institut für Strömungsforschung und der Aerodynamischen Versuchsanstalt Nr. 23 Göttingen 1959

2. Hess, J.L.
Numerical solution of inviscid subsonic flows.
Von Karman Institute for Fluid Dynamics Lecture Series 34 (1971)

3. Bauer, F., Garabedian, P., Korn, D., Jameson, A.
Supercritical Wing Sections II
Springer 1975

4. Hilbig, R.
Transonischer Profilentwurf Va2

Bundesministerium für Forschung und Technologie.
Forschungsbericht W 80-023 (1980)

5. Rosch, H., Klevenhusen, K.D.
Flow Computation Around Multi-Element Airfoils in Viscous Transonic Flow.
ICAS - 80-11.3 (1980)

6. Arlinger, B.C.
Analysis of Two-element High Lift Systems in Transonic Flow
ICAS - 76-13 (1976)

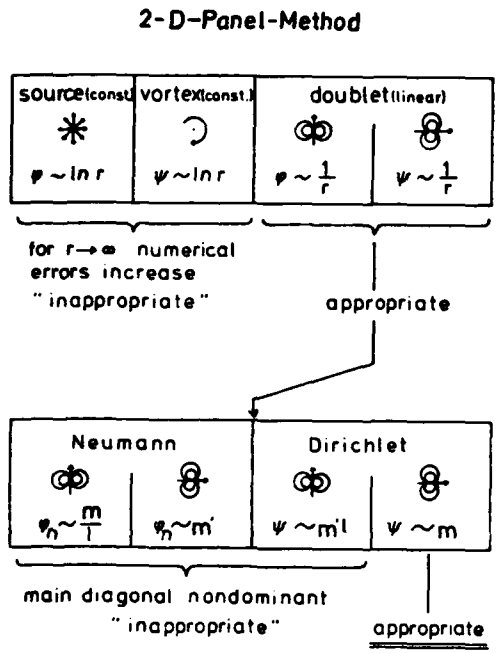


FIG. 1

Modified Dirichlet problem for the streamfunction:

$\psi_{xx} + \psi_{yy} = 0$ in the flow field

ψ = const. at the boundaries

$\psi = \psi_\infty$ at infinity

solution: $\psi' = \psi - \psi_\infty = -\frac{1}{2\pi}\int_s m \frac{\partial}{\partial t} \ln r \, ds$

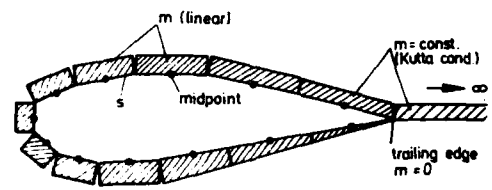


FIG. 2: SINGULARITY METHOD IN THE PHYSICAL PLANE

$u' = \varphi_x'$ x-component of the perturbation velocity
$v' = \varphi_y$ y- " " " " "

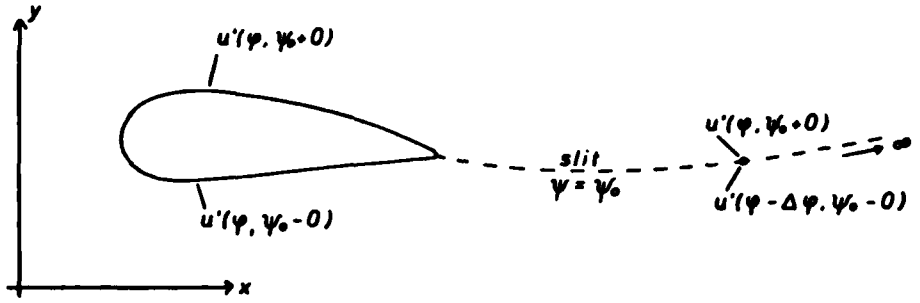


FIG. 3: AIRFOIL IN THE PHYSICAL PLANE

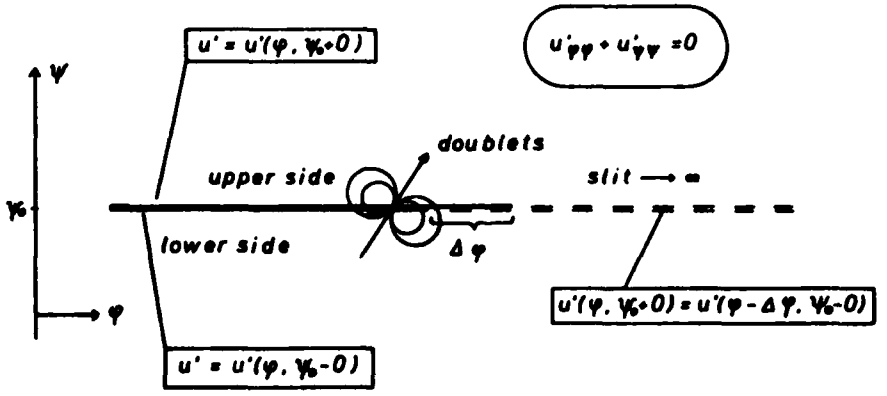
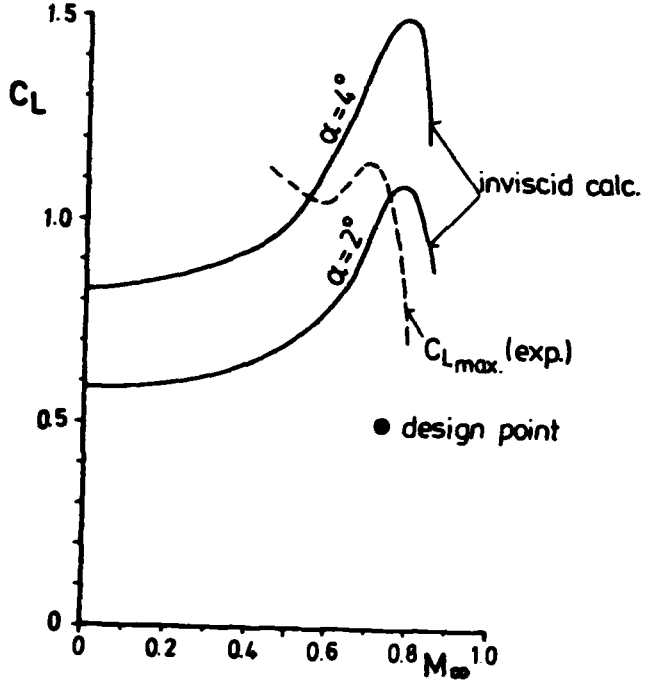


FIG. 4: MODIFIED DIRICHLET PROBLEM IN THE STREAMLINE PLANE



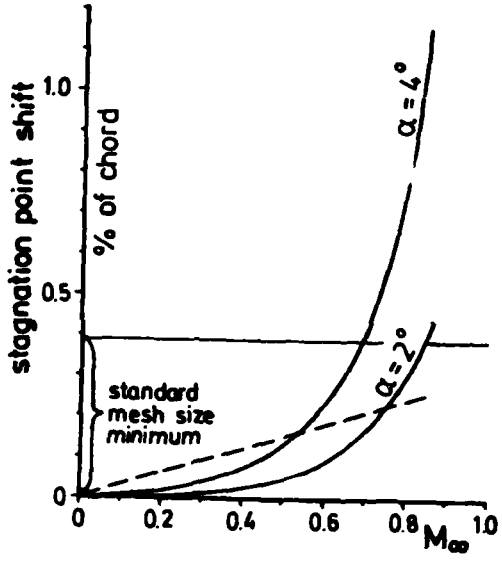


FIG. 5: SUPERCRITICAL AIRFOIL VA 2

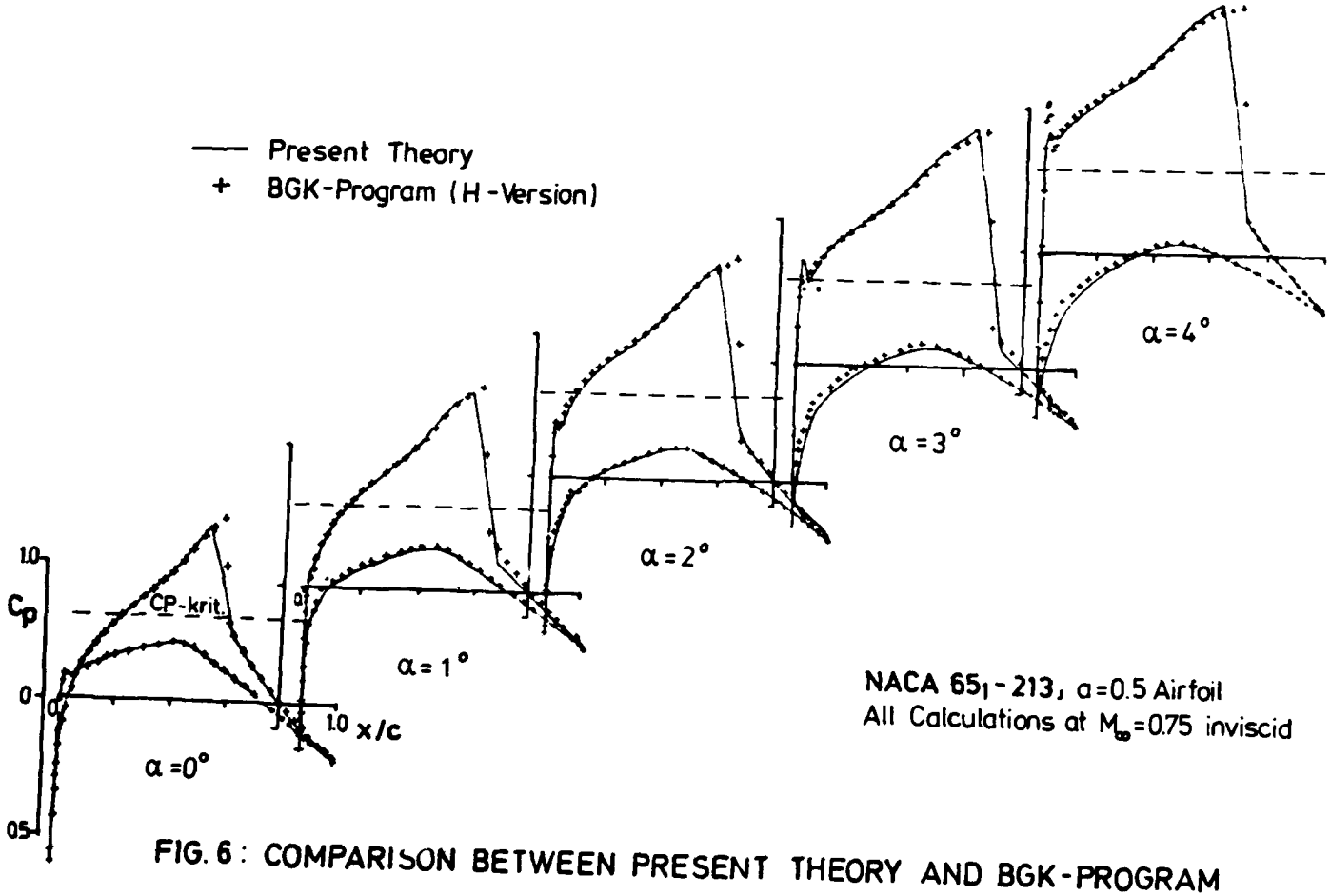


FIG. 6 : COMPARISON BETWEEN PRESENT THEORY AND BGK-PROGRAM

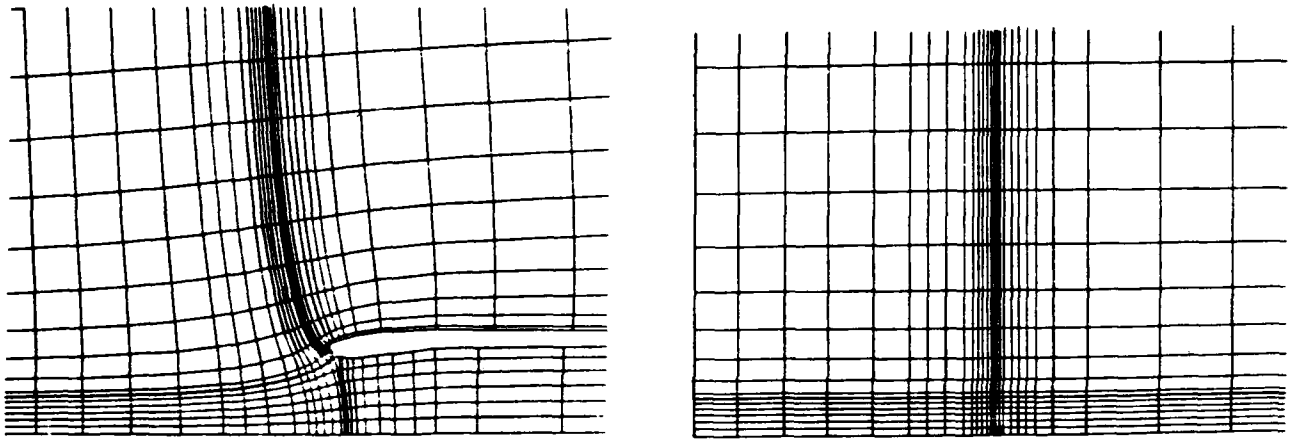


FIG. 7 : GRID AROUND AN INLET

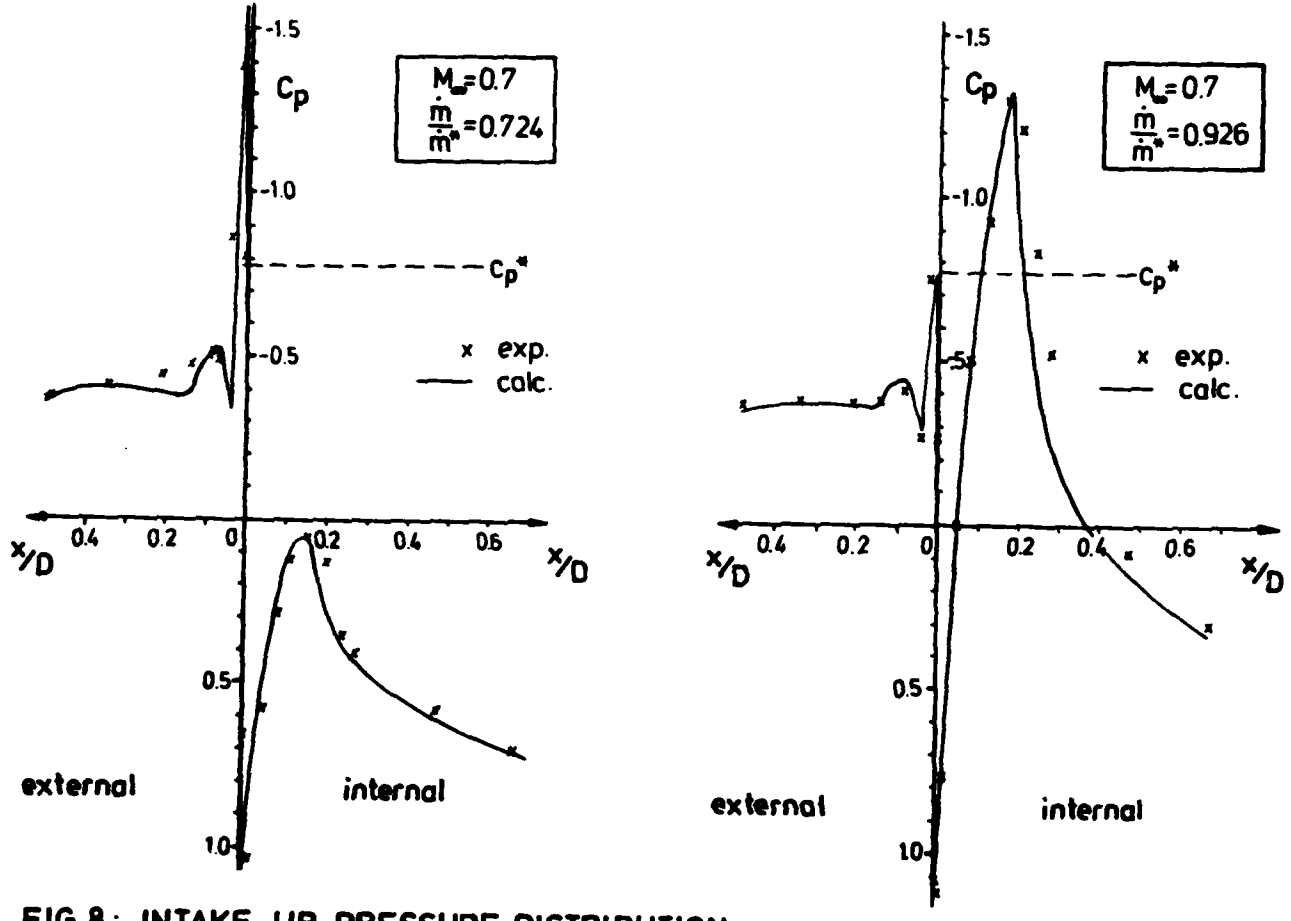


FIG 8: INTAKE LIP PRESSURE DISTRIBUTION

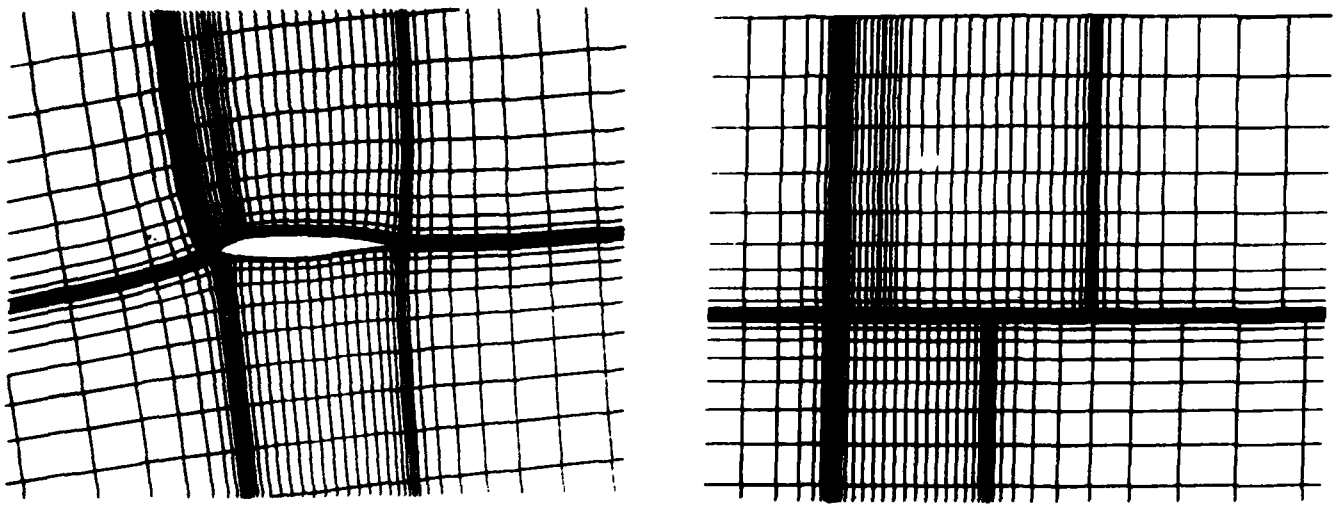


FIG.9 : AIRFOIL WITH SLAT

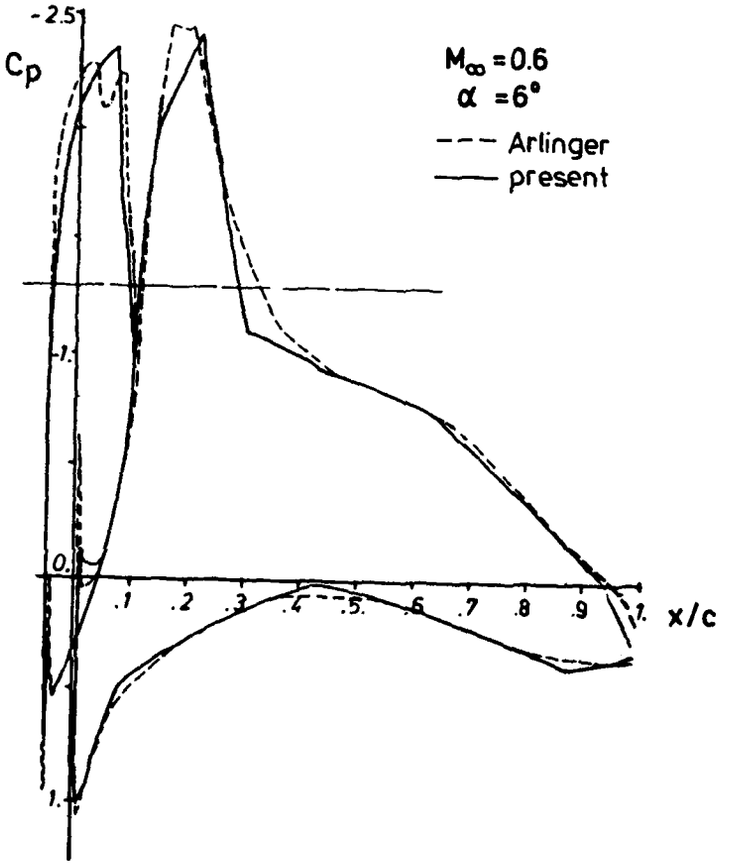


FIG. 10: TWO - ELEMENT AIRFOIL

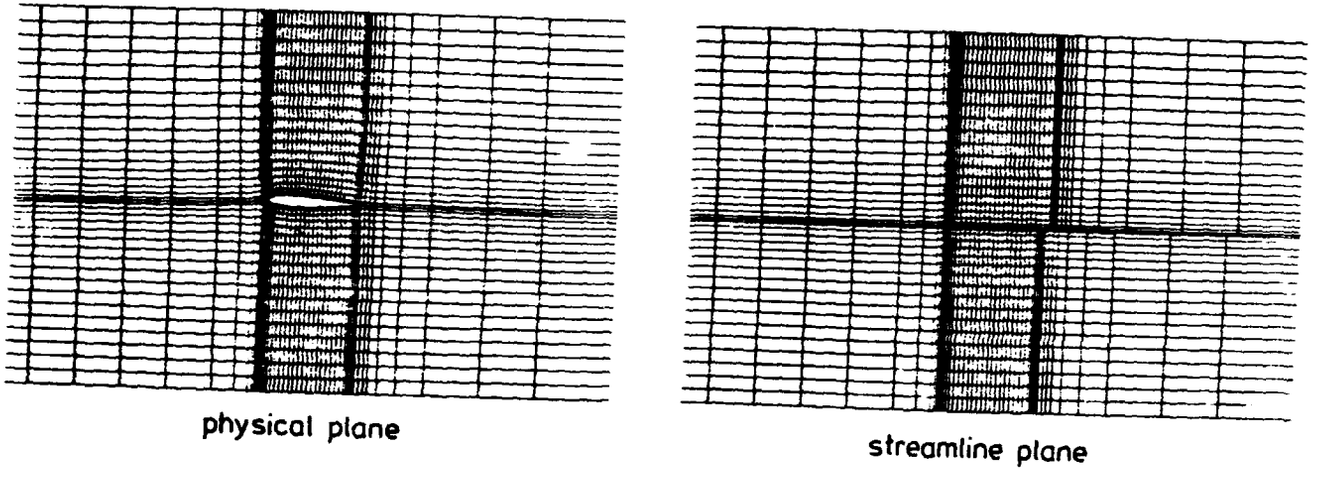
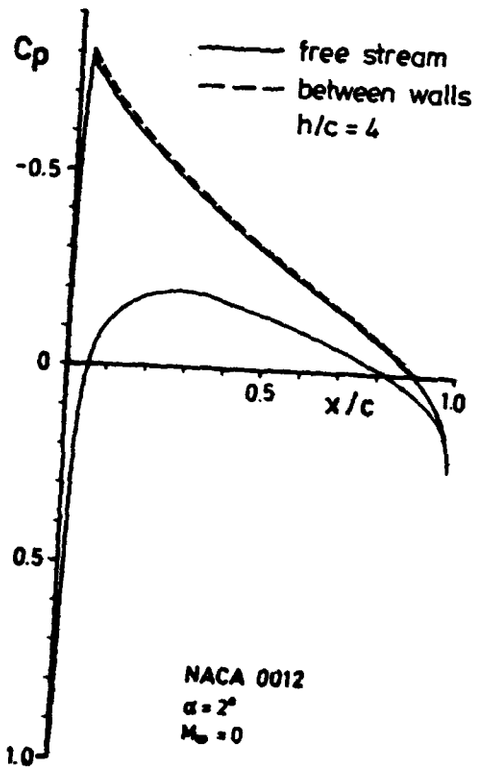


FIG.11: NACA 0012 BETWEEN WINDTUNNEL WALLS $\alpha = 2$, $h/c = 4$



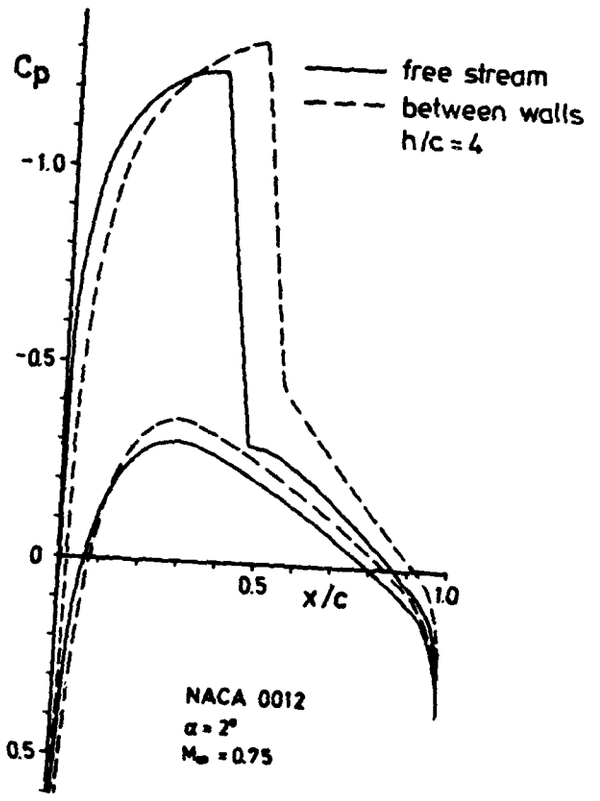


FIG. 12 : WINDTUNNEL WALL INTERFERENCE

AD P001001

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

THREE DIMENSIONAL GRID GENERATION USING BIHARMONICS

G.R. SHUBIN*+, A.B. STEPHENS*, AND J.B. BELL*+
*Naval Surface Weapons Center, Silver Spring, Maryland 20910
+Current address: Exxon Production Research Company, Houston, Texas 77001

INTRODUCTION

We investigate the numerical generation of three dimensional finite difference grids using a segmentation approach and biharmonics. This work is an extension of the two dimensional grid generation technique presented in Reference 1. The present approach is a significant variation on the method of Thompson et al.[2,3].

Our approach is to construct a grid system in a region Ω as a union of subgrids which are determined individually but which join smoothly at the boundaries between subregions. To compute the grid on a subregion Ω_s we determine a transformation from a computational cube R to Ω_s by solving a linear fourth order system of elliptic equations. Grid point locations on Ω_s can then be defined as the image in Ω_s of a uniform grid on R. The fact that the system is fourth order allows enough boundary conditions to be specified so that smoothness across subgrid boundaries is assured. The elliptic nature of the system guarantees smoothness in the subgrid interiors. The governing equations are not only linear but also decouple, so that the x, y, and z components of the transformation can be independently determined. These components turn out to satisfy first boundary value problems for the biharmonic. Discrete solutions to these boundary value problems are obtained by an iterative technique which combines the conjugate-gradient method and Gauss-Seidel iteration. The fact that the subgrids can be small relative to the composite grid permits us to solve several small linear systems instead of one large one. The ability to prescribe (discretely) the transformation on subgrid boundaries gives a simple method for controlling the locations of grid points which are interior with respect to the composite grid.

For the remainder of the paper we will adhere to the convention that the uniform discretization on R used to approximate the transformation to Ω_s has the same number of grid points as we wish to generate in Ω_s. While this is not generally required, when it is true we have the particularly simple situation that each desired grid point in Ω_s is the image of a grid point in R.

ANALYTIC FORMULATION

We now consider the construction of a single subgrid. Each subgrid is defined as the solution to a certain fourth order linear elliptic system where the boundary conditions are used to specify the grid point locations, the orientation of the grid line that intersects the boundary, and the local grid spacing. These conditions provide the control needed at the boundary of each subgrid in order to smoothly fit together the subgrids.

Formulating the problem involves two steps. Following Thompson et al.[2,3], we first formulate equations describing the transformation from physical space to computational space; then, we reverse the roles of dependent and independent variables resulting in equations for the transformation from computational space to physical space. We will consider an ξ-η-ζ computational cube R corresponding to a subgrid region Ω_s in an x-y-z space as shown in Fig. 1.

We now formulate a boundary value problem to determine the transformation $\xi(x,y,z)$, $\eta(x,y,z)$, $\zeta(x,y,z)$ from Ω_s to R. The specified locations of the grid points on the boundary $\partial\Omega_s$ give the boundary values for ξ, η, and ζ. To specify the slope and spacing conditions, we consider an ξ= constant plane (Fig. 2); the η and ζ planes are treated amalogously and the governing equations are obtained by cyclic permutation of (ξ,η,ζ). The tangent vectors t_η and t_ζ at a specified point P can be computed (later by finite differences) from the known boundary data. A normal vector $n=t_\eta \times t_\zeta$ can then be computed, and the three vectors t_η, t_ζ, n define a non-orthogonal local coordinate system at P. Let the vector v give the specified orientation of the grid line intersecting the ξ= constant boundary plane. This orientation could be specified by two angles θ and ϕ giving v relative to the fixed x, y, z coordinate directions; equivalently, we choose to describe v in the local coordinate system at P by

$v = n + \alpha t_\eta + \beta t_\zeta$ where α and β are the two specified quantities. This representation is convenient since α and β are frequently zero, but it is straightforward to relate α and β to θ and ϕ when desired. Since v lies along $\nabla\eta \times \nabla\zeta$, we have $v\cdot\nabla\eta = v\cdot\nabla\zeta = 0$ which constitute the two angle specification boundary conditions. The final boundary condition controls the local grid spacing and is given by specifying $|\nabla\xi|$.

We want our transformation functions to satisfy a system of elliptic partial differential equations in the interior of Ω_s. The differential operator will be denoted L and will be determined later. To summarize, the transformation from the subgrid Ω_s to the computational cube R satisfies

$$L\xi = 0 \qquad \text{in } \Omega_s \tag{2.1}$$

$$L\eta = 0 \qquad \text{in } \Omega_s \tag{2.2}$$

$$L\zeta = 0 \qquad \text{in } \Omega_s \tag{2.3}$$

$$\xi,\eta,\zeta \quad \text{given on } \partial\Omega_s \tag{2.4}$$

$$\left.\begin{aligned} &v(\eta,\zeta)\cdot\nabla\eta = 0 && (2.5)\\ &v(\eta,\zeta)\cdot\nabla\zeta = 0 && (2.6)\\ &|\nabla\xi| \ \text{ given} && (2.7)\end{aligned}\right\} \text{ on } \xi = 0,1$$

$$\left.\begin{aligned} &v(\zeta,\xi)\cdot\nabla\zeta = 0 && (2.8)\\ &v(\zeta,\xi)\cdot\nabla\xi = 0 && (2.9)\\ &|\nabla\eta| \ \text{ given} && (2.10)\end{aligned}\right\} \text{ on } \eta = 0,1$$

$$\left.\begin{aligned} &v(\xi,\eta)\cdot\nabla\xi = 0 && (2.11)\\ &v(\xi,\eta)\cdot\nabla\eta = 0 && (2.12)\\ &|\nabla\zeta| \ \text{ given} && (2.13)\end{aligned}\right\} \text{ on } \zeta = 0,1$$

where the functions v are given.

Since we are interested in the transformation $x(\xi,\eta,\zeta)$, $y(\xi,\eta,\zeta)$, $z(\xi,\eta,\zeta)$ from computational to physical space, we use the assumed invertibility of the transformation to recast the equations by reversing the roles of dependent and independent variables. This process is based on the fact that

$$\frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)} = \left[\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}\right]^{-1}$$

i.e.

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = \frac{1}{J} \begin{bmatrix} y_\eta z_\zeta - y_\zeta z_\eta & z_\eta x_\zeta - z_\zeta x_\eta & x_\eta y_\zeta - x_\zeta y_\eta \\ y_\zeta z_\xi - y_\xi z_\zeta & z_\zeta x_\xi - z_\xi x_\zeta & x_\zeta y_\xi - x_\xi y_\zeta \\ y_\xi z_\eta - y_\eta z_\xi & z_\xi x_\eta - z_\eta x_\xi & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \quad (2.14)$$

where $J = x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)$.

Again we consider an ξ= constant boundary plane (ξ=0 or ξ=1) and examine conditions (2.5)-(2.7). We note that x_ξ, y_ξ, and z_ξ are unknowns but that all other derivatives (x_η, y_η, z_η, x_ζ, y_ζ, z_ζ) can be computed from the specified boundary data. We first reexpress $v = n + \alpha t_\eta + \beta t_\zeta = (a,b,c)$ where a, b, and c are known coordinates with respect to the x, y, z coordinate axes. Then the angle conditions (2.5) and (2.6) become

$$a\eta_x + b\eta_y + c\eta_z = 0 \quad (2.15)$$

$$a\zeta_x + b\zeta_y + c\zeta_z = 0 \quad (2.16)$$

which, using (2.14), are linear in the unknowns x_ξ, y_ξ, z_ξ. The spacing condition (2.7) $|\xi| = g$ (g = given) becomes

$$|\nabla\xi| = (\nabla\xi\cdot\nabla\xi)^{1/2} = (\gamma\cdot\gamma)^{1/2}/J = g \quad (2.17)$$

where $\gamma = (y_\eta z_\zeta - y_\zeta z_\eta,\ z_\eta x_\zeta - z_\zeta x_\eta,\ x_\eta y_\zeta - x_\zeta y_\eta)$ is known. Eq. (2.17) is also linear in x_ξ, y_ξ, z_ξ. Hence (2.15)-(2.17) comprise a 3x3 linear system in the unknowns x_ξ, y_ξ, z_ξ at each point P on an ξ= constant boundary. Specifically, this system is

$$\begin{bmatrix} bz_\zeta - cy_\zeta & cx_\zeta - az_\zeta & ay_\zeta - bx_\zeta \\ bz_\eta - cy_\eta & cx_\eta - az_\eta & ay_\eta - bx_\eta \\ y_\eta z_\zeta - y_\zeta z_\eta & z_\eta x_\zeta - z_\zeta x_\eta & x_\eta y_\zeta - x_\zeta y_\eta \end{bmatrix} \begin{bmatrix} x_\xi \\ y_\xi \\ z_\xi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ (\gamma\cdot\gamma)^{1/2}/g \end{bmatrix} \quad (2.18)$$

We may solve this system at each boundary point, so that the angle and spacing conditions (2.5)-(2.7) are equivalent to specifying the normal derivatives x_ξ, y_ξ, and z_ξ. Clearly, on η= constant and ζ= constant planes we may solve (2.8)-(2.10) and (2.11)-(2.13) to get x_η, y_η, z_η and x_ζ, y_ζ, z_ζ respectively.

Finally, we choose the previously unspecified operator L to be such that, when the roles of dependent and independent variables are reversed, (2.1)-(2.3) become

$$\begin{aligned} \Delta^2 x &= 0 \\ \Delta^2 y &= 0 \\ \Delta^2 z &= 0 \end{aligned} \tag{2.19}$$

Now we note that not only are the governing equations linear, but they also decouple, so that x, y, and z may be determined independently. (It should be noted that this decoupling precludes any special differencing of the boundary conditions as used in Reference 1, resulting in some loss of angle control. This has not been found to cause any serious problems.) In fact, letting w represent either x, y, or z, in order to generate one subgrid Ω_s we must solve three first boundary value problems for the biharmonic, namely

$$\begin{aligned} &\Delta^2 w = 0 \qquad \text{in } R \\ &\left.\begin{matrix} w \\ \dfrac{\partial w}{\partial n} \end{matrix}\right\} \text{specified on } \partial R. \end{aligned} \tag{2.20}$$

Here, the specified values of w are the boundary grid point locations and the specified normal derivatives are obtained by solving the aforementioned 3x3 linear systems. We note that there is no maximum principle for this system of equations; consequently, it is possible to lose invertibility of the transformation so that grid lines of the same family in Ω_s may cross. Our computational experience with this method indicates that problems arise only in cases where the user asks for unreasonable conditions on spacing and slopes of grid lines near boundaries; for example, large spacing and orthogonality near an acute corner.

DISCRETIZATION AND NUMERICAL SOLUTION

In this section we discuss the discretization and numerical solution of (2.20). We first introduce a uniform finite difference grid on R ($\equiv[0,1]^3$) and define for a function w the corresponding mesh function

$$w_{ijk} = w(i\Delta\xi, j\Delta\eta, k\Delta\zeta) \qquad \begin{matrix} i = 0,1,\ldots, I \\ j = 0,1,\ldots, J \\ k = 0,1,\ldots, K \end{matrix}$$

with $I\Delta\xi = J\Delta\eta = K\Delta\zeta = 1$. We also define the usual second divided difference operators δ_ξ^2, δ_η^2, δ_ζ^2 by, e.g.,

$$\delta_\xi^2 w_{ijk} = (w_{i+1\ jk} - 2\, w_{ijk} + w_{i-1\ jk})/(\Delta\xi)^2.$$

Then

$$\Delta w(i\Delta\xi, j\Delta\eta, k\Delta\zeta) = (\delta_\xi^2+\delta_\eta^2+\delta_\zeta^2)w_{ijk} + O(\Delta_\xi^2, \Delta_\eta^2, \Delta_\zeta^2).$$

Denoting $\delta_\xi^2 + \delta_\eta^2 + \delta_\zeta^2$ by Δ_d, we then approximate

$$\Delta^2 w \simeq \Delta_d^2 w_{ijk}.$$

This discretization, which is the "standard" 25-point formula for the biharmonic, introduces additional points outside the computational domain. The unknown values of w at these "fictitious" points are specified using the normal derivative boundary conditions. For example,

$\Delta_d^2 w_{1jk}$ involves w_{-1jk}; however,

$$w_{-1jk} = w_{1jk} - 2\Delta\xi \frac{\partial w}{\partial \xi}(0, j\Delta\eta, k\Delta\zeta) + O(\Delta\xi^2).$$

This relationship and similar ones for the other boundaries allows us to eliminate all of the fictitious points introduced by the discrete representation of the biharmonic. The problem (2.20) is thus reduced to the linear system

$$Aw = f \tag{3.1}$$

where w is a vector of unknowns corresponding to the interior w_{ijk} (i.e. either all of x_{ijk}, y_{ijk}, or z_{ijk}), A is the discrete biharmonic and f contains contributions due to the boundary values of w and $\partial w/\partial n$.

For a variety of reasons, the linear system (3.1) seems amenable to the use of iterative methods. Although reasonably sparse, even if I, J, and K are relatively small, A is a large matrix with large bandwidth. Furthermore, A is symmetric and (as shown by a discrete form of integration by parts) positive definite. Finally, it is important to remember that the solution of (3.1) is used to locate grid points; consequently a highly accurate solution is unnecessary.

Unfortunately, the matrix A is poorly conditioned and point relaxation and ADI methods converge poorly[4,5,6]. For this reason we use a hybrid iteration scheme based on alternation between the conjugate gradient (CG) method and Gauss-Seidel (GS) iteration. The combination of these two iterative methods is quite natural. The conjugate gradient method works well in resolving low frequency error waves so that after several CG steps much of the remaining error is high frequency in nature. These high frequency waves are damped effectively by Gauss-Seidel so that for several iterations GS is very effective.

The CG algorithm (see, e.g. Reference 7) is initialized by defining for an initial guess w_o

$$p_o = r_o = f - Aw_o .$$

The iteration is then given by

$$w_{k+1} = w_k + \alpha_k p_k$$

where

$$\alpha_k = \frac{r_k^t p_k}{p_k^t A p_k}$$

$$r_{k+1} = r_k - Ap_k$$

$$p_{k+1} = r_{k+1} - \beta_k p_k$$

$$\beta_k = \frac{r_{k+1}^t A p_k}{p_k^t A p_k} .$$

Writing the algorithm in this way requires only one multiplication of A per step, thereby minimizing the work per step. However, we note that in general the equation

$$r_{k+1} = r_k - Ap_k \tag{3.2}$$

should not be used because it leads to severe roundoff problems (instead use $r_{k+1} = f - Aw_k$). Here, since we take only a fixed (fairly small) number of CG steps before switching to GS, the extra work in computing Aw_k is not justified.

For the sake of completeness we note that the Gauss-Seidel iteration is given by

$$w_{k+1} = A_L^{-1}(f - A_U w_k)$$

where A_L is the part of A on or below the diagonal and A_U is the part of A strictly above the diagonal.

For the model problems considered in the next section the initial data for w_o was determined by interpolation and we alternately used 25 steps of conjugate gradient and 10 steps of Gauss-Seidel. After 25 steps we begin to see serious roundoff in (3.2) and after 10 steps the GS iterations begin to lose their effectiveness. For the present problem this hybrid method worked faster than either GS alone or CG with the alternate formula for r_{k+1}; CG with (3.2) did not work at all. The usefulness of this hybrid method for more general elliptic problems will be explored more fully and documented in a later report.

RESULTS

We display some simple 3-D grids generated by the present method. In each case the gradient (spacing) conditions (2.7), (2.10), and (2.13) were specified by computing the gradients along the edges of the region (this can be done from the specified boundary data) and linearly interpolating along the "faces" (i.e. the boundary planes).

In Figs. 3A-3D we show a single subgrid in which normality ($\alpha=\beta=0$) was enforced. Fig. 3A shows some of the specified boundary data, and Figs. 3B-3D show the grid lines generated along selected internal ξ, η, and ζ = constant planes. Of course, due to the discretization error in approximating the solutions of (2.20), exact normality of the grid lines intersecting the boundary plane is not obtained.

In Figs. 4A-4D we grid a more complex region by breaking it up into two subgrids. On planes 1 and 3 we specify a uniform grid. On plane 2, the subgrid boundary, we specify a clustered grid. On the remaining boundary planes the grid point locations are linearly interpolated from those on the edges of planes 1, 2, and 3. Since the grid lines proceeding essentially in the y-direction must "turn a corner" at plane 2, some care is used in giving the angle boundary conditions to help the composite grid turn the corner. We thus specify that the interior grid lines intersect plane 2 at angles whose values are interpolated between the angles that the edges AI, IQ, CK, KS, EM, MU, GO, OW make with plane 2. On all other boundaries we specify that the grid lines intersect normally ($\alpha=\beta=0$). In Figs. 4B-4D we examine the smoothness of the grid generated on some internal planes. Fig. 4B shows a plane near the top (where the most turning is required) and the grid is quite smooth. In Fig. 4C we look at a "top view" of another such internal plane and see that indeed the composite grid is smooth at the subgrid boundary PL and that normality is approximately enforced elsewhere. Similarly, in Fig. 4D we look at a "side view" and observe smooth results.

In closing, we note that a time-consuming and tedious aspect of 3-D grid generation appears to be the specification of the data required on the two-dimensional boundary surfaces. Perhaps existing 2-D grid generation techniques can be adapted to manifolds in order to circumvent this problem.

REFERENCES

1. Bell, J.B., Shubin, G.R., and Stephens, A.B., "A Segmentation Approach to Grid Generation Using Biharmonics," submitted for publication.
2. Thompson, J.F., Thames, F.C., and Mastin, C.W., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," J. Comp. Physics, v. 15, 1974, pp. 299-319.
3. Thompson, J.F., Thames, F.C., Mastin, C.W., and Shanks, S.P., "Use of Numerically Generated Body-Fitted Coordinate Systems for Solution of the Navier-Stokes Equations," AIAA Second Computational Fluid Dynamics Conference, Hartford, Connecticut, June 1975, pp. 68-80.
4. Conte, S.D., and Dames, R.T., "An Alternating Direction Method for Solving the Biharmonic Equation," Math. Tables Aid. Comp. v. 12, 1958, pp. 198-205.
5. Fairweather, G., Gourlay, A.R., and Mitchell, A.R., "Some High Accuracy Difference Schemes With a Splitting Operator For Equations of Parabolic and Elliptic Type," Numer. Math., v. 10, 1967, pp. 56-66.

6. Hadjidimos, A., "The Numerical Solution of a Model Problem Biharmonic Equation Using Extrapolated Alternating Direction Implicit Methods," Numer. Math., v. 17, 1971, pp. 301-317.
7. Leunberger, D., Optimization by Vector Space Methods, Wiley.

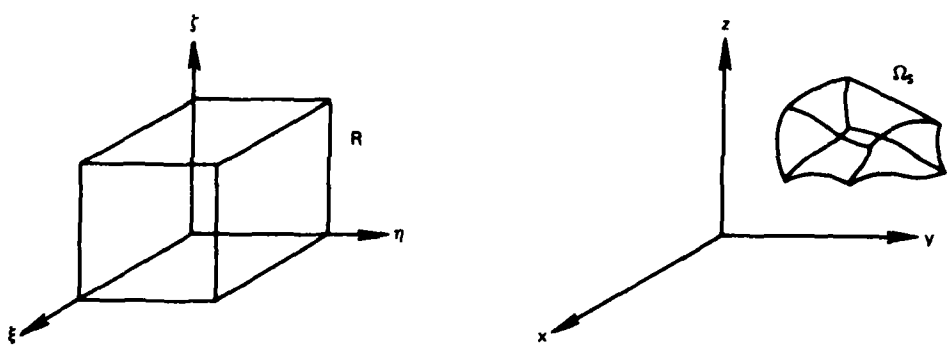


Fig. 1. Computational cube R and subgrid region Ω_s.

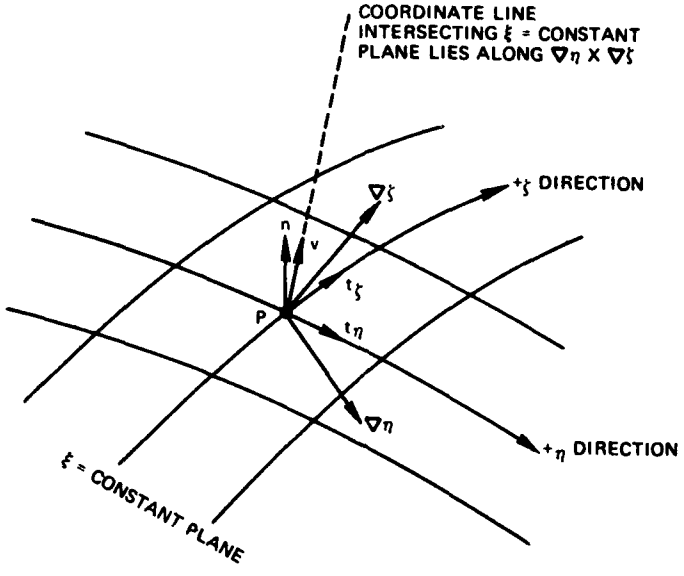


Fig. 2 Boundary condition formulation at grid point P on the image of an ξ = constant plane in x-y-z space.

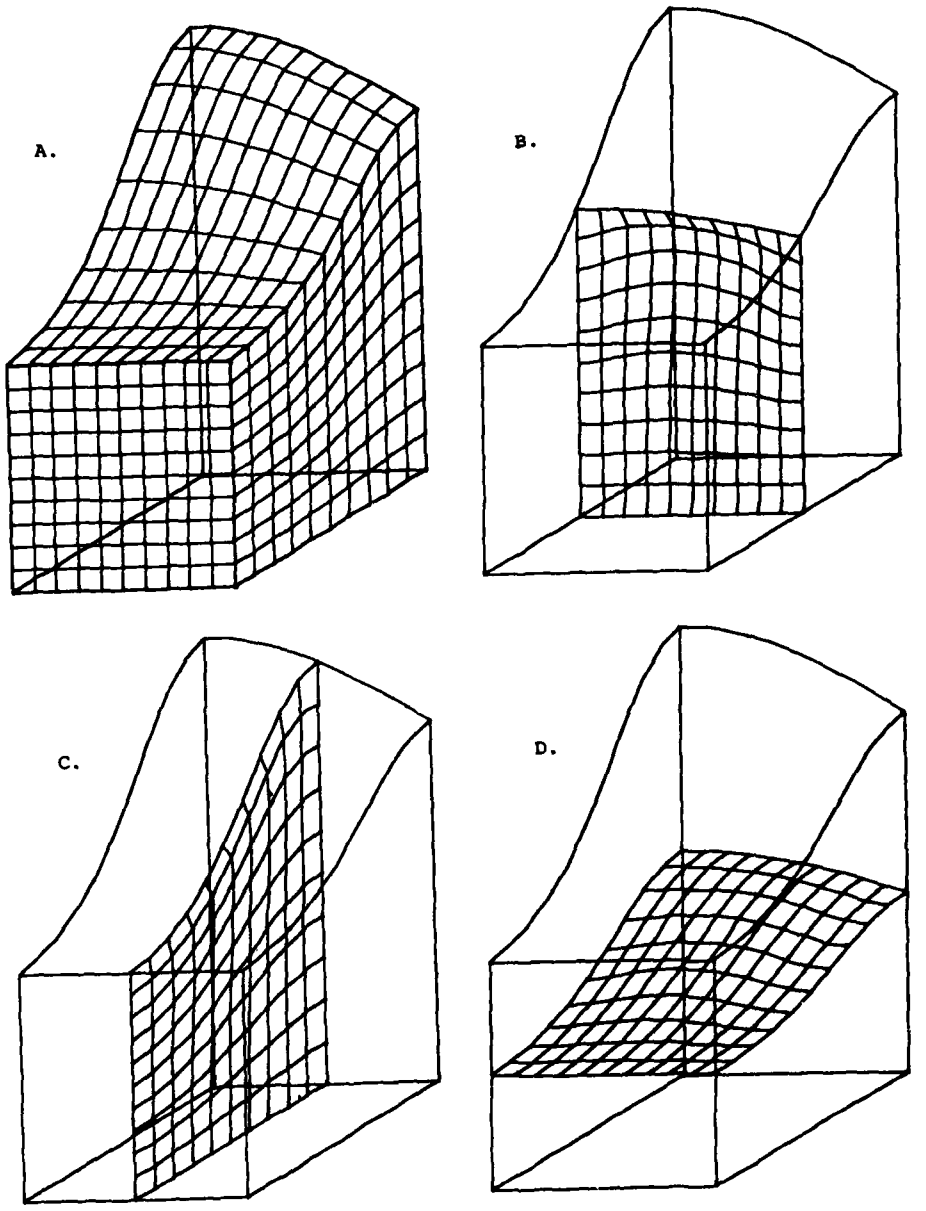


Fig. 3.

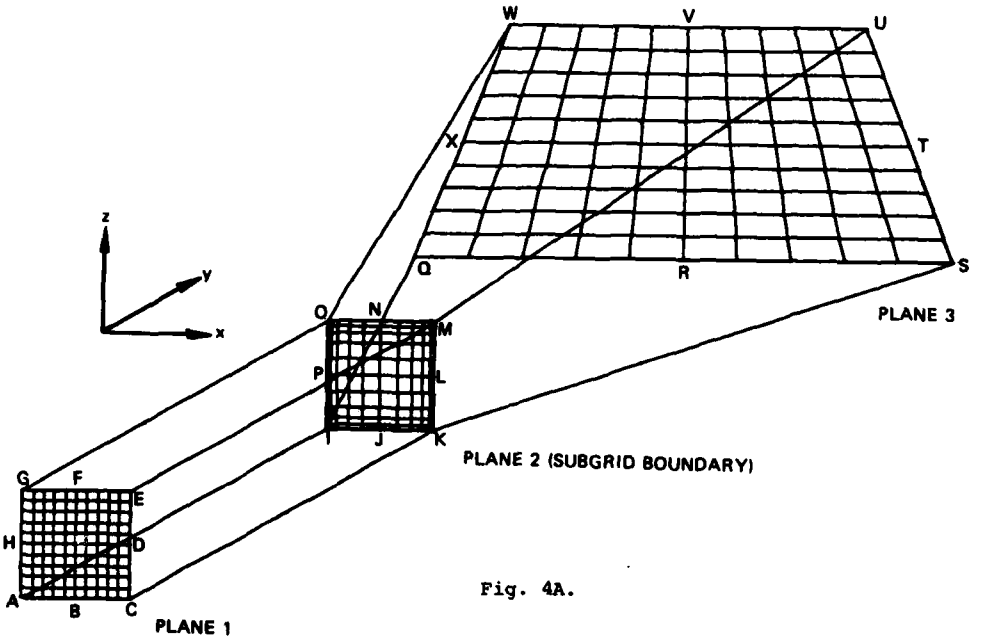

W
V
U
X
T
Q
R
S
PLANE 3
z
y
x
O
N
M
P
L
I
J
K
PLANE 2 (SUBGRID BOUNDARY)
G
F
E
H
D
A
B
C
PLANE 1

Fig. 4A.

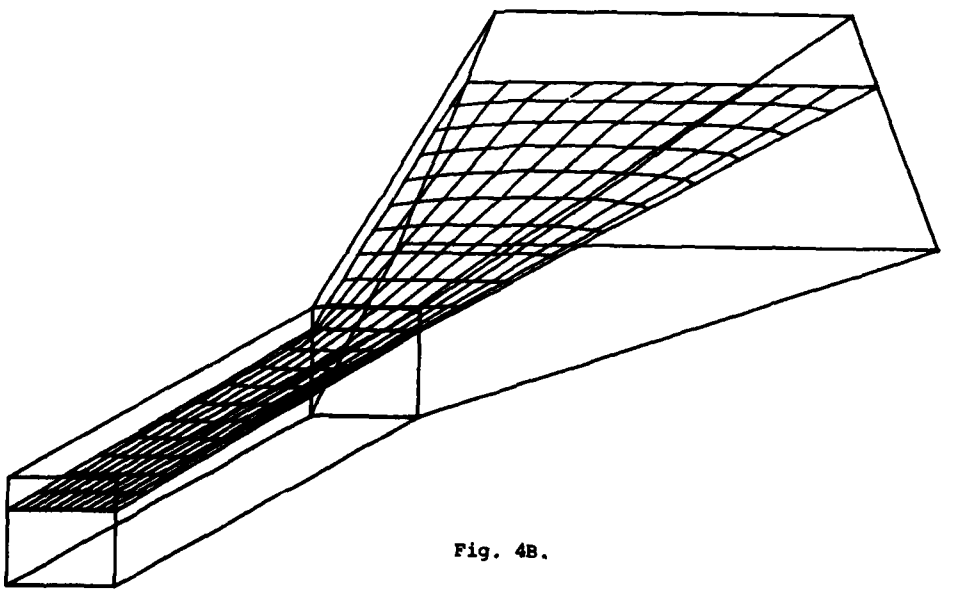


Fig. 4B.

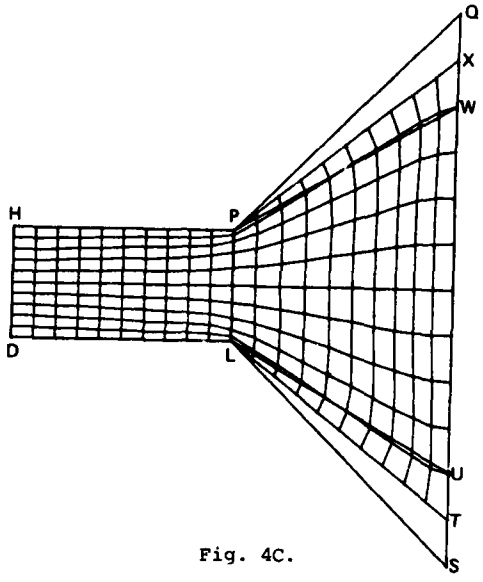


Fig. 4C.

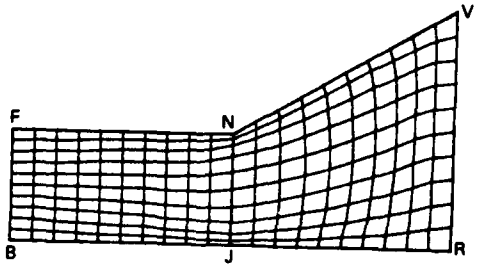


Fig. 4D.

AD P001002

Copyright 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

MARCHING GRID GENERATION USING PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS

S. Nakamura
The Ohio State University, Mechanical Engineering Department
206 West 18th Avenue, Columbus, Ohio, 43210

INTRODUCTION

The unique aspect of grid generation in computational fluid dynamics is that the grid generation equations have no physical meanings, so any equation may be used for this purpose if the grids generated are useful. This aspect provides a vast freedom in developing new methods of grid generation. Although the grid generation method originally proposed by Thompson, et. al.[1] has been most frequently used in computational fluid dynamics, faster and less expensive methods are constantly searched. The hyperbolic grid generation method proposed later by Steger and Chaussee[2] generates two-dimensional grids for airfoil calculations much faster than the elliptic grid generation method. It does not require iterative solution but rather the solution marches from the inner boundary (airfoil surface) toward the outer field generating loops of grids one by one, so the computational time is almost equal to that of one iteration in solving the elliptic grid generation equations by an iterative scheme. The problems with the hyperbolic grid generation method are, however, that (1) often singularities in the boundary condition propagate as the solution marches outward, (2) solution may become unstable unless an "artificial viscosity" term is adequately added to the equations, and (3) outer boundary conditions cannot be specified.

This paper explores feasibility of using parabolic partial differential equations for grid generation. The advantages of using parabolic partial differential equations are as follows: (1) parabolic equations are initial value problems, so grids are generated by a marching algorithm like the hyperbolic grid generation method, (2) the parabolic partial differential equations have most properties of the elliptic equations, particularly the diffusion effect which smooths out any singularity of the inner boundary condition if any, and (3) the prescribed outer boundary conditions may be satisfied. The importance of the marching algorithm stated above is twofold: first, computational time required is only a very small fraction of that for the elliptic grid generation

Financial support for this work was provided by NASA Ames Research Center under NCA2-OR565-101

equations; second, the fast-memory space required during grid generation can be substantially reduced from that required by the elliptic grid generation method. The authors consider the grid generation for a two-dimensional airfoil flow calculations for simplicity of discussions, although the method described in this paper is not restricted to two dimensions or airfoil problems.

PRELIMINARY OBSERVATIONS

Since very little is known about the use of parabolic partial differential equations for grid generation, we examine first the feasibility by considering the following set of equations:

$$\partial x(\xi,\eta)/\partial\eta = A\partial^2 x(\xi,\eta)/\partial\xi^2 + S_x$$
$$\partial y(\xi,\eta)/\partial\eta = A\partial^2 y(\xi,\eta)/\partial\xi^2 + S_y \qquad (1)$$

where

$$S_x = Bx + C$$
$$S_y = By + C \qquad (2)$$

and where A, B and C are constants, (x,y) is the physical domain, and (ξ,η) is the computational domain.

Equation (1) can be discretized on both ξ and η coordinates using the backward differencing scheme on the η coordinate and the central differencing scheme on the ξ coordinate. Once the initial values of x and y are specified at $\eta=0$, the difference equations can be solved with the tridiagonal solution scheme for each increment of η.

We set the initial values as

$$x(\xi,0) = x_o(\xi)$$
$$y(\xi,0) = y_o(\xi) \qquad (3)$$

where $x_o(\xi)$ and $y_o(\xi)$ are the coordinates of the inner boundary (airfoil surface) on the physical domain.

The behavior of the solution of Eq.(1) on the x-y plane for different combinations of selected values of the coefficients ar summarized next:

Case 1 A>0, B=C=0

Each of Eq.(1) is similar to the heat conduction equation for an insulated loop of wire. The value of $x(\xi,\eta)$ and $y(\xi,\eta)$ approach the average of the x_o and y_o, respectively, as η increases.

Case 2: A>0, B<0 and C=0

The values of $x(\xi,\eta)$ and $y(\xi,\eta)$ assymptotically

approach x=y=0 as η increases.

Case 3: A>0, B>0 and C=0

Provided that the origin is inside the inner boundary, the contour of $(x(\xi,\eta), y(\xi,\eta))$ will expand outward as η increases.

Case 4: A>0, B=0 and C>0

The contour of $(x(\xi,\eta), y(\xi,\eta))$ will shrink to a moving point on a line, x=Cs and y=Cs, where s is a parameter.

Among the above four cases, the result of Case 3 is the most encouraging, because the grid lines generated expand as η increases. The above analysis suggests that a more favorable solution for grid generation may be obtained if the source terms are improven.

In order to study the effect of the source terms more clearly, we set A=0 in Eq.(1) and approximate it by

$$\Delta x(\xi,\eta) = S_x(\xi,\eta)\Delta\eta$$
$$\Delta y(\xi,\eta) = S_y(\xi,\eta)\Delta\eta \qquad (4)$$

This form suggests that, as η increases, the change of x and y is determined by S_x and S_y. This implies that S_x and S_y should be specified in such a way that x and y change in the desired direction and amount. The role of $x_{\xi\xi}$ and $y_{\xi\xi}$ in Eq.(1) may be considered as smoothing the grid intervals in the ξ-direction. As a method, S_x and S_y may be determined by using polynomial interpolations[3] between the inner and outer boundaries.

PRACTICAL APPLICATIONS

In the remainder of this paper, we use the source terms in the form of a linear interpolation between the current grid and the outer boundary. The basic form of mesh generation equations adopted here is written in a semi-discrete form as

$$x_j(\xi) - x_{j-1}(\xi) = [Ax_{\xi\xi} + Bx_{\xi\eta} + S_{xj}(\xi)]/C$$
$$y_j(\xi) - y_{j-1}(\xi) = [Ay_{\xi\xi} + By_{\xi\eta} + S_{yj}(\xi)]/C \qquad (5)$$

where j denotes a discretized value of η or equivalently the j-th grid line counted from the inner boundary; A, B and C are constants: S_x and S_y are given by

$$S_{xj}(\xi) = (X(\xi) - x_j(\xi))/(J-j)$$
$$S_{yj}(\xi) = (Y(\xi) - y_j(\xi))/(J-j) \qquad (6)$$

In the above equations, J is the maximum value of j that corresponds to the outer boundary; $X(\xi)$ and $Y(\xi)$ are the outer boundary conditions. The orthogonality of grid lines in the vicinity of inner and outer boundaries may be controlled by changing $X(\xi)$ and $Y(\xi)$ in Eq.(6) as j increases. The grid spacing in both ξ and η directions can be easily controlled by modifying the difference equations for Eq.(5).

Notice that Eq.(5) reduces to the semi-discrete form of the elliptic grid generation equation if S_x and S_y are replaced by

$$S_{xj}(\xi) = x_{j+1}(\xi) - x_j(\xi)$$
$$S_{yj}(\xi) = y_{j+1}(\xi) - y_j(\xi) \qquad (7)$$

This relation to the elliptic grid generation equation is used in the next section to derive the spacing control algorithm in the marching grid generation equations.

GRID SPACING CONTROL

In most coordinate transformations used for computational fluid analyses, the grid spacing in both ξ and η directions on the computational domain are set to unity. However, this restriction is not necessary at least until the grids are used for actual flow calculations. When deriving grid generation equations, we can assume that grid spacings are locally non-uniform on the computational domain. Once the grids are generated, they may be used for flow calculations as if generated on a uniformly spaced grids on the computational domain.

In order to explain the proposed method of controlling grid spacing, we first focus our attention on the grid generation equation of the elliptic type:

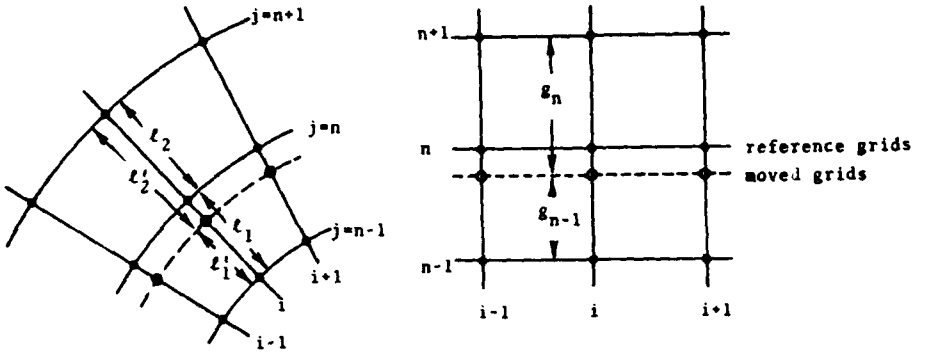
$$Ax_{\xi\xi} + Bx_{\xi\eta} + Cx_{\eta\eta} = 0$$
$$Ay_{\xi\xi} + By_{\xi\eta} + Cy_{\eta\eta} = 0 \qquad (8)$$

where

$$A = x_\eta^2 + y_\eta^2, \quad B = -2(x_\xi x_\eta + y_\xi y_\eta), \quad C = x_\xi^2 + y_\xi^2$$

With a uniform grid spacing on the computational domain, the difference equations for Eq.(8) may be written in the form:

$$A[r_{i-1,j} - 2r_{i,j} + r_{i+1,j}]$$
$$+ B[r_{i-1,j-1} - r_{i-1,j+1} - r_{i+1,j-1} + r_{i+1,j+1}]/4$$
$$+ C[r_{i,j-1} - 2r_{i,j} + r_{i,j+1}] = 0 \qquad (9)$$



(a) The x-y domain (b) The ξ-η domain

Figure 1 Relation between the reference grids and the moved grids

Suppose that a system of grids, referred here as the refence grids, has been generated by Eq.(9), but it is desired to generate another grid system that is almost identical to the reference grids except that one particular grid line, say j=n, is moved toward the adjacent grid line j=n-1. Figure 1 shows the relation between the two grid systems, where the reference grids are plotted by solid lines, while the new desired line is plotted by a dotted line. The coordinates of the reference grids are denoted by $(x_{i,j}, y_{i,j})$, while those of the moved grids in the desired system are denoted by $(x'_{i,j}, y'_{i,j})$. The distance between the reference grids (i,n) and (i,n-1) is denoted by ℓ_1, the distance between (i,n) and (i,n+1) by ℓ_2, while that between (i,n) and (i,n-1) of the desired grids by ℓ'_1 and that between (i,n) and (i,n+1) by ℓ'_2. We assume that the ratios, ℓ'_1/ℓ_1, ℓ'_2/ℓ_2, and ℓ_1/ℓ_2, are constant for all the grids on the grid line j=n, although this assumption may be relaxed in practice. Since the grids (i,n-1) and (i,n+1) are not moved, the following linear relations hold with a good accuracy:

$$r_{i,n} - r_{i,n-1} = (r'_{i,n} - r_{i,n-1})/g_{n-1}$$
$$r_{i,n+1} - r_{i,n} = (r_{i,n+1} - r'_{i,n})/g_n \qquad (10)$$

where

$$g_{n-1} = \ell'_1/\ell_1, \quad g_n = \ell'_2/\ell_2$$

$r_{i,j} = x_{i,j}$ or $y_{i,j}$ for the reference grids, and $r'_{i,j} = x'_{i,j}$ or $y'_{i,j}$ for the moved grids.

Therefore, the difference equations that yield the desired grids are obtained by eliminating $x_{i,n}$ and $y_{i,n}$ in Eq.(9) by using Eq.(10). The difference

equations for the desired grids thus obtained are all identical to Eq.(9) except for the following three equations that involve $r'_{i,n}$:

$$A[\, r_{i-1,n-1} - 2r_{i,n-1} + r_{i+1,n-1}]$$
$$+ B[\, r_{i-1,n-2} - r_{i+1,n-2} - r'_{i-1,n} + r'_{i+1,n}]/[2(1 + g_{n-1})]$$
$$+ C[\, r_{i,n-2} - r_{i,n-1} + (-r_{i,n-1} + r'_{i,n})/g_{n-1}] = 0 \qquad (11)$$

$$A[\, r'_{i-1,n} - 2r'_{i,n} + r'_{i+1,n}]$$
$$+ B[\, r_{i-1,n-1} - r_{i+1,n-1} - r_{i-1,n+1} + r_{i+1,n+1}]/4$$
$$+ C[\, (r_{i,n-1} - r'_{i,n})/g_{n-1} + (-r'_{i,n} + r_{i,n+1})/g_n] = 0 \qquad (12)$$

$$A[\, r_{i-1,n+1} - 2r_{i,n+1} + r_{i+1,n+1}]$$
$$+ B[\, r'_{i-1,n} - r'_{i+1,n} - r_{i-1,n+2} + r_{i+1,n+2}]/[2(1 + g_n)]$$
$$+ C[\, (r'_{i,n} - r_{i,n+1})/g_n - r_{i,n+1} + r_{i,n+2}] = 0 \qquad (13)$$

where the terms with the coefficient B are slightly modified for simplicity of equations. Notice that his procedure is essentially equivalent to deriving the difference equation on the non-uniform grids of the computational domain in which the n-th grid line in the ξ-direction on the uniform grids is moved toward the (n-1)th grid line as shown in Fig. 1b. The ratio g_j does not have to be necessarily a constant for all the grids along the grid line but rather, by changing this ratio gradually, the distance of the movement may be changed as i changes.

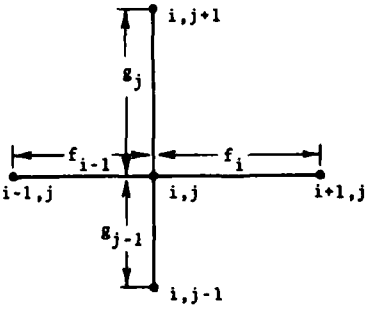


Figure 2 Notations for grid spacing parameters

Although deriving the equations that moves only one grid line has been mentioned, any number of grid lines may be moved in both ξ and η directions. By denoting the distance ratios by f_i and g_j in the ξ and η directions, respectively, as shown in Fig. 2, the difference equations that yield the desired grid spacing can be written as

$$A[(r_{i-1,j} - r_{i,j})/f_{i-1} + (r_{i+1,j} - r_{i,j})/f_i]$$
$$+ B(r_{i+1,j+1} - r_{i+1,j-1} - r_{i-1,j+1} + r_{i-1,j-1})/(f_{i-1} + f_i)/(g_{j-1} + g_j)$$
$$+ C[(r_{i,j-1} - r_{i,j})/g_{j-1} + (r_{i,j+1} - r_{i,j})/g_j] = 0 \qquad (14)$$

The non-uniform grid spacing terms in the above mentioned scheme has the similar effects as the spacing control terms in the form of exponential functions introcuced by Thompson, et al.[1] The major advantage of the present approach when applied to the elliptic grid generation equations is, however, that the value of f_i or g_j and the distance between the adjacent grids maintain approximately a linear relationship. Therefore, if a grid system is generated by using a known set of f_i and g_j for all the grid intervals, and if different spacing distribution is desired in the next grid generation, the values of f and g that fulfill the desired grid spacings can be easily found by the linear relationship between the f or g and the grid spacing in the previous calculation.

The marching grid generation equation in the difference form with spacing control is obtained from the elliptic grid generation equation, Eq.(9), by replacing the coordinates $x_{i,j+1}$ and $y_{i,j+1}$ by known values $X_{i,j+1}$ and $Y_{i,j+1}$, as briefly described in the previous section. Here, $X_{i,J}$ and $Y_{i,J}$ are the coordinates along the prescribed outer boundary, which are represented by $X(\xi)$ and $Y(\xi)$ in the previous section, while $X_{i,j}$ and $Y_{i,j}$ are functions of j, reflecting the earlier statement that $X(\xi)$ and $Y(\xi)$ in Eq.(6) may be changed as j increases. In the simplest case they are set to $X_{i,J}$ and $Y_{i,J}$ respectively for all j. An algorithm to prescribe $X_{i,j}$ and $Y_{i,j}$ is explained in the next section in conjunction with the local orthogonality control. Thus, the marching grid generation equations for both x and y coordinates are written in the form:

$$A[(r_{i-1,j} - r_{i,j})/f_{i-1} + (r_{i+1,j} - r_{i,j})/f_i]$$
$$+ C[(-r_{i,j}/g_{j-1} - r_{i,j}/G_j)$$
$$= - B(R_{i+1,j+1} - R_{i-1,j+1} - r_{i+1,j-1} + r_{i-1,j-1})/(f_{i-1,j} + f_i)/(g_{j-1}+G_j)$$
$$- C[\, r_{i,j-1}/g_{j-1} + R_{i,j+1}/G_j \,] \qquad (15)$$

where G_j is the distance between the grid (i,j) and (i,J) on the computational plane and $R_{i,j}=X_{i,j}$ or $Y_{i,j}$.

Equation (15) is solved simultaneously for all the grids on the j-th grid line by using the tridiagonal solution for each of r=x and r=y. The solution starts with j=2 that is next to the airfoil surface and marches until j=J-1 that is next to the prescribed outer boundary grid line is reached. The coefficients A, B and C are calculated using the coodinates of adjacent grids that are already generated.

LOCAL ORTHOGONALITY OF GRID LINES

In the flow computations using a coordinate transformation, often an approximate orthogonality of grid lines in the vicinity of a boundary, particularly along the airfoil surface, is desired to increase the accuracy of the finite difference approximation for the flow boundary conditions. The approximate orthogonality of grid lines around the airfoil may be obtained by changing $X_{i,j}$ and $Y_{i,j}$ in Eq.(15) as j increases as follows.

First, we consider the coordinates $(X_{i,3}, Y_{i,3})$, which are used when the grid line next to the airfoil surface is generated. Referring to Fig. 3, a straight line AB is extended outward normal to the airfoil surface from the grid (i,1). On the other hand, a circular arc CD that passes the point $(X_{i,J}, Y_{i,J})$ on the outer boundary and has its center at (i,1) is drawn. The coordinates $(X_{i,3}, Y_{i,3})$ are set to those of the intersction E of the two lines.

The values of $X_{i,j}$ and $Y_{i,j}$ for j>3 are obtained by gradually moving the point E toward $(X_{i,J}, Y_{i,J})$ as j increases.

If the grid spacing along the inner boundary changes rapidly, specifying the ratio f_{i-1}/f_i according to the criterion

$$f_{i-1}/f_i = d_{i-1}/d_i \qquad (16)$$

is necessary to enhance the orthogonality, where d_i is the distance between the grids (i,1) and (i+1,1) along the inner boundary. The ratio f_{i-1}/f_i may be gradually changed as j increases.

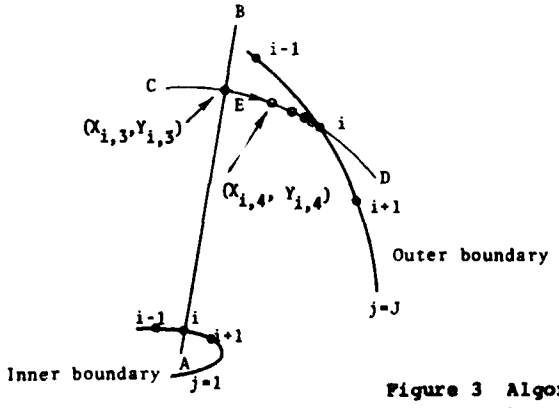


Figure 3 Algorithm to determine $X_{i,j}$ and $Y_{i,j}$

ILLUSTRATION OF THE GRIDS GENERATED

Figure 4 shows an O-mesh generated by the method described in this paper. The grids on the outer boundary are equally spaced along a circle with the radius equal to five chord lengths. The orthogonality control described earlier is allpied in the vicinity of the airfoil.

Figure 5 shows an H-mesh generated by the same method. The horizontal line that splits into the upper and lower surfaces of the airfoil is used as initial conditions. The grids above the airfoil were first generated starting from the airfoil surface to the top boundary, and then the grids below the airfloil were generated by the same procedure.

Figures 6, 7 and 8 illustrate the grids generated by the present method for three different airfoils. Only the grids from j=1 to j=13 are plotted to show the detail. The strong clustering of grids toward an airfoil is obtained by the spacing control algorithm.

CONCLUSIONS

The present feasibility study shows that high quality grids for computational fluid dynamics can be generated by the marching solution of parabolic partial differential equations. The computing cost of the present method is a very small fraction of that of the elliptic grid generation equations and comparable to that of the hyperbolic grid generation equations. However, the present method is free from propagation of singularity and difficulty of satisfying the desired boundary conditions. The extension to three-dimensional geometries, particularly the fuselage-wing flow problems, is being studied.[4]

REFERENCES

1. Thompson, J. F., Thames, F. C., and Mastin, C. M., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," J. of Comp. Physics, Vol 15, pp.299-319, 1974.
2. Steger, J. S. and Chaussee, D. S., "Generation of Body Fitted Coordinates Using Hyperbolic Partial Differential Equations," FSI Report 80-1, Flow Simulation, Inc., Sunnyvale, Ca., January, 1980.
3. Eiseman, P. R., "A Multi-Surface Method of Coordinate Generation," J. Comp. Physics, Vol. 33, pp.118-150, 1979.
4. Nakamura, S. "Non-iterative Grid Generation Using Parabolic Difference Equations for Fuselage-Wing Flow Calculations," to be presented at the Eighth International Conference on Numerical Methods in Fluid Dynamics, 28 June-2 July 1982, Aachen, West Germany.

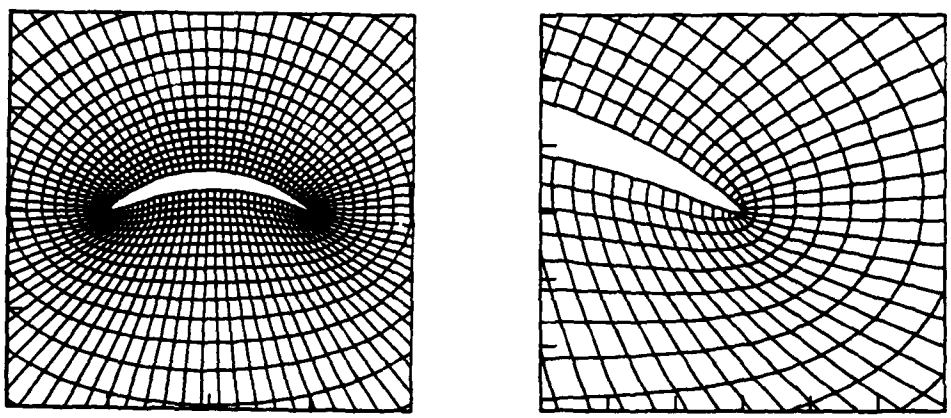


Figure 4 O-mesh generated by the marching algorithm

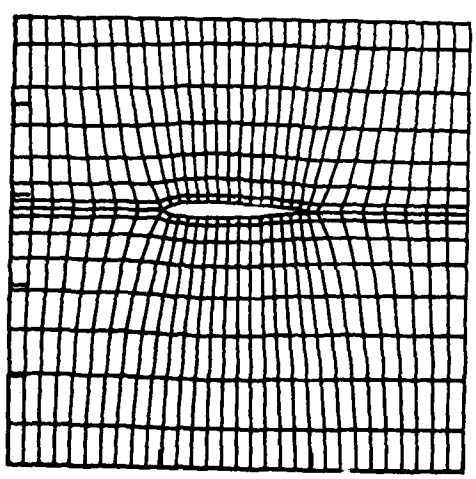
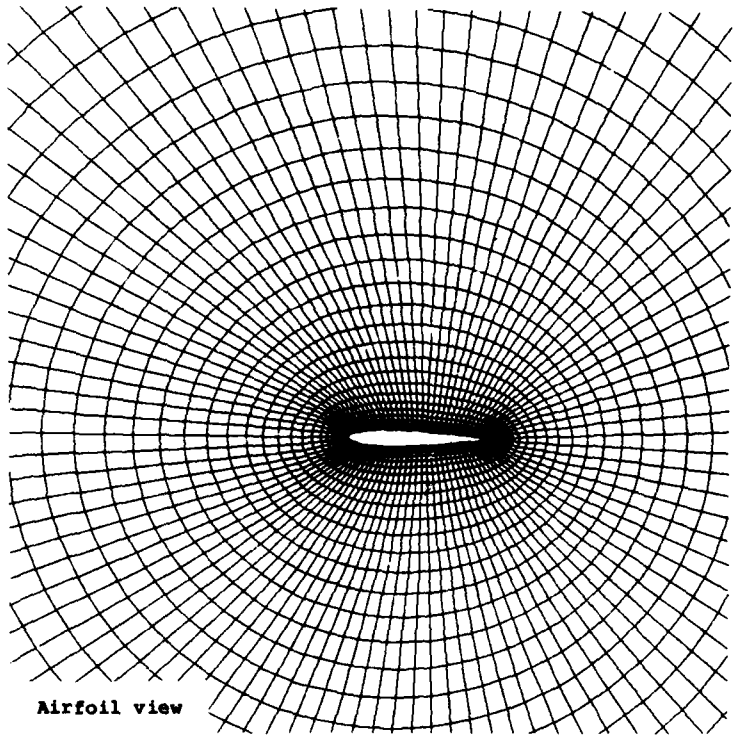
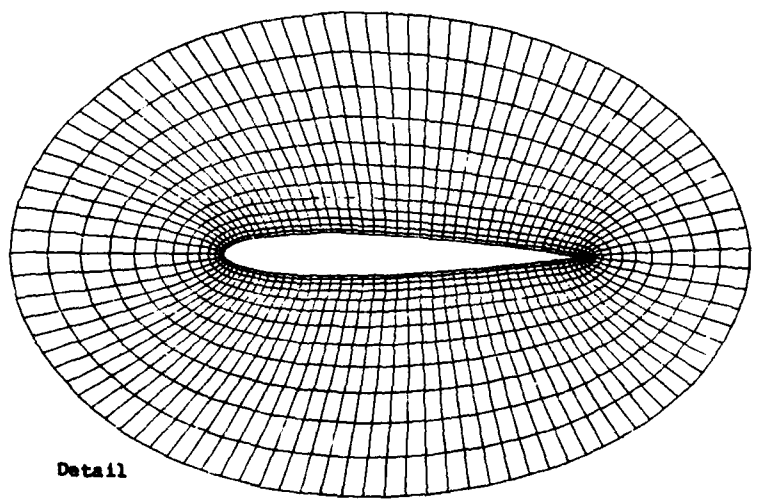


Figure 5 H-mesh generated by the marching algorithm



Airfoil view



Detail

Figure 6 Grids generated for the NACA-0012 airfoil

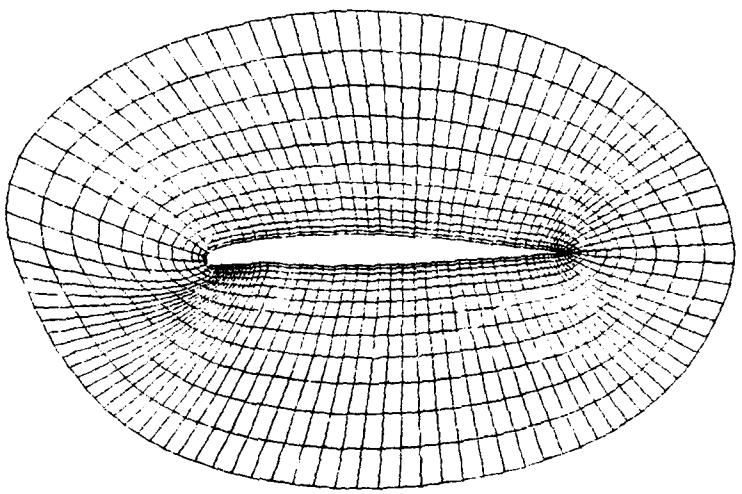


Figure 7 Grids generated for a Gates-Leajet airfoil (detail).

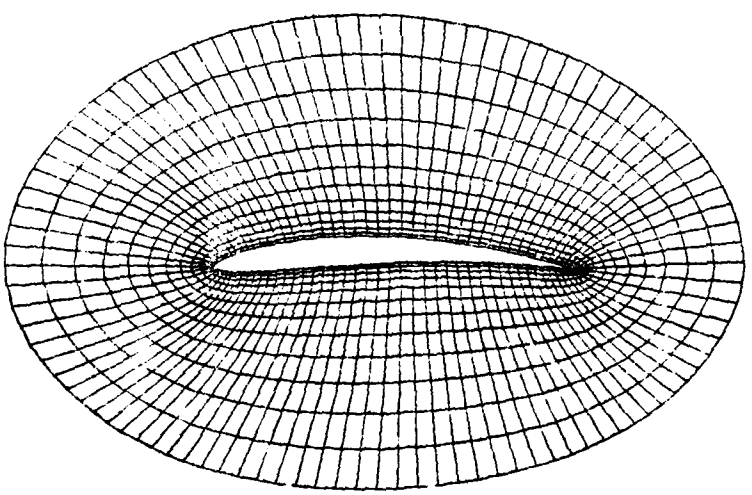


Figure 8 Grids for an arbitrarily cambered airfoil (detail)

AD P001003

Copyright 1982 by Elsevier Science Publishing Co...
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ASSESSING THE QUALITY OF CURVILINEAR COORDINATE MESHES BY DECOMPOSING THE JACOBIAN MATRIX

G. David Kerlick[+] and Goetz H. Klopfer[+,++]
[+]Nielsen Engineering & Research, Inc., 510 Clyde Avenue, Mountain View, California 94043, USA

ABSTRACT

An algebraic decomposition of the Jacobian matrix $[J] = \partial x^i/\partial \xi^j$ which relates physical and computational variables is presented. This invertible decomposition parameterizes the mesh by the physically intuitive qualities of cell orientation, cell orthogonality, cell volume, and cell aspect ratio.

This decomposition can be used to analyze numerically generated curvilinear coordinate meshes and to assess the contribution of the mesh to the truncation error for any specific differential operator and algorithm. This is worked out in detail for Laplace's equation in nonconservative and conservative forms. An analysis (by G. H. K.)[1] of the mesh contribution to truncation error for the full potential code TAIR is given in abbreviated form. The variables introduced here, and their derivatives are also natural Lagrange multipliers for adaptive mesh algorithms based on a variational principle.

1. INTRODUCTION

Local truncation error (TE) can be considered the local measure of accuracy for any numerical scheme which approximates the solution of a partial differential equation. In finite difference schemes, this truncation error will depend upon the discretization of the coordinate system (mesh) upon which the finite difference approximation of the differential equation will be solved. It will also depend upon the form of the equation, the algorithm, and the (unknown) solution. Thus, it is important to know how various properties of the mesh contribute to the local error so that these contributions can be reduced where it is possible to do so.

This paper presents a physically meaningful reparameterization of the Jacobian matrix (and of the metric tensor) which should be of value in analyzing meshes and assessing their contribution to the truncation error. This will be illustrated for two dimensions here, but can be extended to n dimensions.

[++]Work supported by NASA/Ames Research Center. Applied Computational Aerodynamics Branch. Contract NAS2-11063.

Let the physical domain to be modeled be represented by the Cartesian coordinates $x^i = (x,y,\ldots)$ $i = 1,n$ in physical space. The computational space will be reparameterized by local computational coordinates $\xi^i = (\xi,\eta,\ldots)$ in a computational space. Thus $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$ in two dimensions. Without loss of generality, we may assume that the discretization in computational space is even, i.e., $\xi^i = I\Delta\xi$, $I = 1,N$. For the present we will work in terms of infinitesimals.

2. DECOMPOSITION OF THE JACOBIAN MATRIX

The Jacobian matrix $[J] := \partial x^i/\partial\xi^j$ is the matrix of the partial derivatives of the physical coordinates with respect to the computational coordinates, which for two dimensions is

$$[J] = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} \tag{1}$$

We begin the decomposition of [J] by splitting off that part of the Jacobian matrix which gives the relative orientation of the computational coordinate axes (the tangents to the ξ^i) with respect to the physical axes. For example, the direction cosine of the ξ-axis with respect to the x-axis is given by

$$\cos\alpha = \frac{x_\xi}{\sqrt{x_\xi^2 + y_\xi^2}} \tag{2}$$

In two dimensions this is the only independent angle (see also ref. 2). In three dimensions there are three independent angles which can be related to the three Euler angles. The angle $\alpha(\xi,\eta)$ measures the rotation of the (ξ,η) coordinates with respect to the (x,y) coordinates at every point. Note that this rotation is an "extrinsic" property of the coordinate because it depends upon the embedding of (ξ,η) in (x,y) space. This rotation is only defined for flat spaces. For a curved two-dimensional manifold, one could not define such an angle, since the integral of the rotation about a closed contour would be proportional to the integral of the curvature.[3]

The remaining three components of [J] can be expressed in terms of the metric tensor g_{ij} given by

$$g_{ij} = \delta_{k\ell}\frac{dx^k}{d\xi^i}\frac{dx^\ell}{d\xi^j} \tag{3}$$

which is the square of the Jacobian matrix $[J]^T[J]$. In two dimensions

$$\begin{aligned} g_{11} &= x_\xi^2 + y_\xi^2 \\ g_{12} &= x_\xi x_\eta + y_\xi y_\eta \\ g_{22} &= x_\eta^2 + y_\eta^2 \end{aligned} \tag{4}$$

The metric tensor is an intrinsic property of the computational space, independent of its embedding. It may be decomposed into its determinant and into factors which represent the aspect ratio of coordinate cells and factors which measure the deviation of the coordinate system from orthogonality.

The determinant of the metric is given by

$$\det|g_{ij}| = J^2 \tag{5}$$

where

$$J = \det \left| \frac{\partial x^i}{\partial \xi^j} \right| \tag{6}$$

is the usual Jacobian determinant. In two dimensions

$$J = x_\xi y_\eta - x_\eta y_\xi \tag{7}$$

The Jacobian determinant represents the volume in physical space of a mesh cell in computational space.

The n^{th} root of the metric determinant in n dimensions may be factored out, leaving the conformal metric[4] $\tilde{g}_{ij}$ which has unit determinant. In n dimensions

$$\tilde{g}_{ij} = J^{-2/n} g_{ij} \tag{8}$$

The nonorthogonality of the mesh cell is characterized by the angles θ_{ij} between the tangents to the ξ^i and ξ^j coordinate lines, thus

$$\cos\theta_{ij} = \frac{g_{ij}}{\sqrt{g_{ii}\, g_{jj}}} = \frac{\tilde{g}_{ij}}{\sqrt{\tilde{g}_{ii}\tilde{g}_{jj}}} \quad , \; i \neq j, \text{ no sum on } i,j \tag{9}$$

For orthogonal coordinates $\theta_{ij} = \pi/2$. This angle depends only on the conformal metric, since the determinant factors out of the definition.

Finally, the aspect ratios of the cell are defined as the ratio of the magnitude of the tangent vectors

$$A_k = \sqrt{\frac{g_{kk}}{g_{11}}} = \sqrt{\frac{\tilde{g}_{kk}}{\tilde{g}_{11}}}\ , \quad k = 2,n, \text{ no sum on } k \tag{10}$$

Here again, this ratio depends only on the conformal metric.

The decomposition is summarized in Figure 1. In n dimensions there are n(n-1)/2 orientation angles, 1 volume parameter, n(n-1)/2 orthogonality parameters and (n-1) aspect ratios.

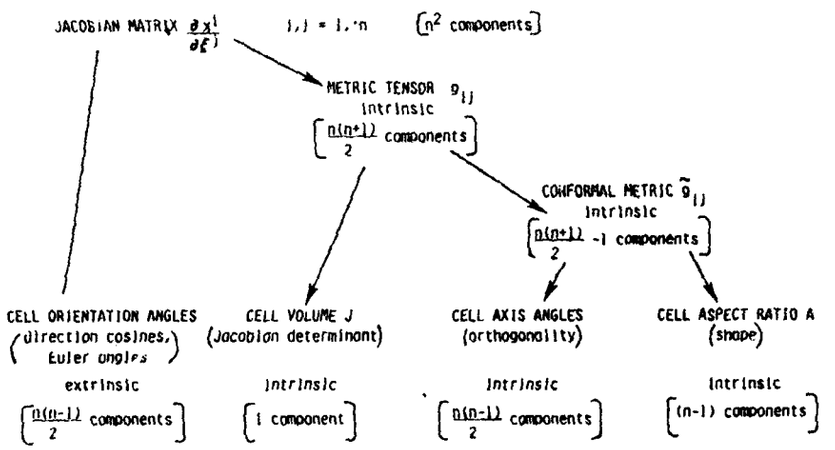


Fig. 1. Decomposition of the Jacobian Matrix.

The complete reparameterization in two dimensions is thus

$$J = x_\xi y_\eta - x_\eta y_\xi = \sqrt{\det|g_{ij}|}$$

$$A = \left[\frac{x_\eta^2 + y_\eta^2}{x_\xi^2 + y_\xi^2}\right]^{1/2} = \sqrt{\frac{g_{22}}{g_{11}}}$$

$$\cos\theta = \frac{x_\xi x_\eta + y_\xi y_\eta}{(x_\xi^2 + y_\xi^2)^{1/2}(x_\eta^2 + y_\eta^2)^{1/2}} = \frac{g_{12}}{\sqrt{g_{11}g_{22}}}$$

$$\cos\alpha = \frac{x_\xi}{(x_\xi^2 + y_\xi^2)^{1/2}} = \frac{x_\xi}{\sqrt{g_{11}}} \tag{11}$$

The inverse of this parameterization in two dimensions is

$$\begin{aligned} x_\xi &= \left(\frac{J}{A\sin\theta}\right)^{1/2} \cos\alpha \\ y_\xi &= -\left(\frac{J}{A\sin\theta}\right)^{1/2} \sin\alpha \\ x_\eta &= \left(\frac{AJ}{\sin\theta}\right)^{1/2} \cos(\theta - \alpha) \\ y_\eta &= \left(\frac{AJ}{\sin\theta}\right)^{1/2} \sin(\theta - \alpha) \end{aligned} \tag{12}$$

Equations (12) are unique up to a replacement of α by $(\alpha - \delta)$ which corresponds to taking a different choice of axis for defining the orientation angle.

The formulas for the covariant metrics in two and three dimensions are, respectively

$$g_{ij} = J\begin{pmatrix} A^{-1}\csc\theta & \cot\theta \\ \cot\theta & A\csc\theta \end{pmatrix} \tag{13}$$

$$g_{ij} = J^{2/3}\Lambda^{-1/3}\begin{pmatrix} 1 & A_2\cos\theta_{12} & A_3\cos\theta_{13} \\ A_2\cos\theta_{12} & A_2 & A_2A_3\cos\theta_{23} \\ A_3\cos\theta_{13} & A_2A_3\cos\theta_{23} & A_3 \end{pmatrix} \tag{14}$$

where the normalization factor Λ is

$$\Lambda = A_2^2A_3^2[1 + 2\cos\theta_{12}\cos\theta_{13}\cos\theta_{23} - \cos^2\theta_{12} - \cos^2\theta_{13} - \cos^2\theta_{23}] \tag{15}$$

3. LAPLACE'S EQUATION

The nonconservative (de Rham) Laplacian operator may be written in general curvilinear coordinates as

$$\nabla^2 f = g^{ij}\nabla_i\partial_j f \tag{16}$$

where g^{ij}, the contravariant metric, is the matrix inverse of g_{ij}, ∂_j is partial differentiation and ∇_i is covariant differentiation. We may also write the conservative form of this equation in terms of the vector density $F^i = \sqrt{g}\, g^{ij}\, \partial_j f$, so that Laplace's equation becomes

$$\sqrt{g}\, \nabla^2 f = \partial_i F^i = \partial_i (\sqrt{g}\, g^{ij} \partial_j f) = 0 \tag{17}$$

In terms of the parameters introduced in section 2 we have for the nonconservative form:

$$\nabla^2 f = \frac{1}{J\sin\theta} \Big\{ A f_{\xi\xi} - \cos\theta\, f_{\xi\eta} - A^{-1} f_{\eta\eta} + [A_\xi - A\theta_\xi \cot\theta + \theta_\eta \csc\theta] f_\xi + \Big[\frac{-A_\eta}{A^2} - \frac{\theta_\eta}{A} \cot\theta + \theta_\xi \csc\theta \Big] f_\eta \Big\} = 0 \tag{18}$$

The conservative form is

$$\partial_\xi (A \csc\theta\, f_\xi - \cot\theta\, f_\eta) + \partial_\eta (-\cot\theta\, f_\xi + A^{-1} \csc\theta\, f_\eta) = 0 \tag{19}$$

4. TRUNCATION ERROR

Suppose that we apply central differences in our finite difference scheme, so that the first and second differences of f are given by

$$\delta_\xi f = \frac{f(\xi_{i+1}) - f(\xi_{i-1})}{2\Delta\xi} = f_\xi + \frac{\Delta\xi^2}{6} f_{\xi\xi\xi} + O(\Delta\xi^4)$$

$$\delta_{\xi\xi} f = \frac{f(\xi_{i+1}) - 2f(\xi_i) + f(\xi_{i-1})}{\Delta\xi^2} = f_{\xi\xi} + \frac{\Delta\xi^2}{12} f_{\xi\xi\xi\xi} + O(\Delta\xi^4) \tag{20}$$

The function f can be either the solution variable or the physical coordinate x^i. Hence there will be an error of order $(\Delta\xi)^2$ in the Jacobian matrix components. By the chain rule we can find the errors in the metric tensor and other geometric quantities derived above. For example the truncation error Δg_{11} in the metric tensor component g_{11} is

$$\Delta g_{11} = \frac{\Delta\xi^2}{6} [2x_\xi x_{\xi\xi\xi} + 2y_\xi y_{\xi\xi\xi}] \tag{21}$$

Using the definitions above we may express these errors in terms of the new parameters, albeit at the cost of some tedious manipulation. Here, for example, a partial reparameterization yields the result that

$$\Delta g_{11} = \frac{\Delta\xi^2}{6}\left[-\frac{1}{2}g_{11,1}^2 + g_{11,11} - g_{11}\alpha_{,1}^2\right] \tag{22}$$

Here, $x^1 = \xi$ and a comma denotes partial differentiation. This shows us that, although the metric tensor is an invariantly defined quantity, its truncation error is not, since the first derivative of the rotation angle α appears.

The truncation error analysis for this problem contains a great deal of algebra, but should be relatively simple to do on a computer, since the manipulations are straightforward. For the moment, we turn to a simple example.

5. POLAR COORDINATES

We consider the transformation to (unclustered) polar coordinates

$$\begin{aligned} x &= r\cos\phi \\ y &= r\sin\phi \end{aligned} \tag{23}$$

so that $(\xi,\eta) = (r,\phi)$. The parameters corresponding to this are

$$J = r \quad A = r \quad \theta = \pi/2 \quad \alpha = -\phi \tag{24}$$

In the nonconservative case, Laplace's equation is

$$\frac{1}{r}\{rf_{rr} - \frac{1}{r}f_{\phi\phi} + f_r\} = 0 \tag{25}$$

Since the Jacobian of the transformation is computed by finite differences also, the corresponding truncation errors in the metric tensor and Christoffel symbols must be computed. We obtain as a result of this computation, in the non-conservative case, the modified equation

$$\nabla^2 f = (TE)_f + (TE)_g + O(\Delta^4)$$

$$(TE)_f = \frac{\Delta r^2}{12}\left(f_{rrrr} + \frac{2}{r}f_{rrr}\right) + \frac{\Delta\phi^2}{12}\left(\frac{1}{r^2}f_{\phi\phi\phi\phi}\right)$$

$$(TE)_g = \frac{\Delta\phi^2}{12}\left(\frac{4}{r^2}f_{\phi\phi} + \frac{3}{r}f_r\right)$$

The truncation errors can be segregated into two groups. The "f" errors are due to differencing of the function while the "g" errors are due to differencing the metrics. In the conservative case we obtain the respective terms.

$$(TE)_f = \frac{r^2}{12}\left(4 f_{rrrr} + \frac{8}{r} f_{rrr}\right) + \frac{\Delta\phi^2}{12}\left(\frac{4}{r^2} f_{\phi\phi\phi\phi}\right)$$

$$(TE)_g = \frac{\Delta\phi^2}{12}\left(\frac{2}{r^2} f_{\phi\phi} - 2f_{rr} - \frac{2}{r} f_r\right) \tag{26}$$

Additional characteristic features of truncation errors are shown here. First and foremost, we observe that every term depends upon the derivatives of the (unknown) solution, so that a complete a priori determination of the error is impossible and any a priori estimate depends upon our prediction of the solution. Second, the truncation error depends on the form of the equation (strongly or weakly conservative, or non-conservative). Of course, changing the type and order of the difference scheme will also change the error.

6. ANALYSIS APPLIED TO THE SOLUTION OF THE FULL POTENTIAL CODE TAIR[5,6]

Some numerical results will be presented here. The results will be in terms of a post-mortem. For a given mesh and numerical solution based on the full potential code (Holst[5]) as modified by Dougherty[6] the truncation errors will be analyzed to see if the mesh indeed was adequate.

The full potential equation written in strongly conservative form for the velocity potential ϕ is

$$(\rho\phi_x)_x + (\rho\phi_y)_y = 0$$

$$\rho = \left[1 - \frac{\gamma - 1}{\gamma + 1}\left(\phi_x^2 + \phi_y^2\right)\right]^{\frac{1}{\gamma-1}} \tag{27}$$

where the density, ρ, and the velocity ϕ_x, ϕ_y, are nondimensionalized with respect to the stagnation density, ρ_o, and the critical sound speed, a^*, respectively; x and y are the Cartesian coordinates and γ is the ratio of specific heats.

In the computational space the equation becomes

$$(J\rho U)_\xi + (J\rho V)_\eta = 0 \tag{28}$$

where

$$U = (g_{22}\phi_\xi - g_{12}\phi_\eta)/J^2$$

$$V = (-g_{12}\phi_\xi + g_{11}\phi_\eta)/J^2$$

The density is now given by

$$\rho = \left[1 - \frac{\gamma - 1}{\gamma + 1}\left\{\left(\frac{g_{22}}{J^2}\right)\phi_\xi^2 - 2\left(\frac{g_{12}}{J^2}\right)\phi_\xi\phi_\eta + \left(\frac{g_{11}}{J^2}\right)\phi_\eta^2\right\}\right]^{\frac{1}{\gamma-1}} \tag{29}$$

In this second order differential equation the metric tensor components appear directly. This is unlike a first order differential system where only the components of the Jacobian matrix appear.

Since for full potential equations the density is usually the desired solution from which the pressure and Mach numbers can be determined we will examine the truncation error of equation (29) only. For simplicity simple central differencing will be used throughout the entire flow field whether the local Mach numbers are sub- or supersonic. Let the exact density be given by ρ_E and the numerically derived density by ρ_N. The relation between the two is

$$\rho_E = \rho_N + \Delta\rho + \text{higher order terms}$$

$$\tilde{=} \rho_N\left(1 + \frac{\Delta\rho}{\rho_N}\right) \tag{30}$$

The $\Delta\rho$ can be derived from the modified equation form of equation (29), as follows: replace all the derivatives by central differences and expand in a Taylor series. Obtain thereby the modified equation

$$\rho_E = \rho_N - \frac{1}{\gamma+1}\left\{\frac{g_{22}}{J^2}\Delta\left(\phi_\xi^2\right) - \frac{2g_{12}}{J^2}\Delta\left(\phi_\xi\phi_\eta\right) + \frac{g_{11}}{J^2}\Delta\left(\phi_\eta^2\right)\right\}$$

$$- \frac{1}{\gamma+1}\left\{\phi_\xi^2\Delta\left(\frac{g_{22}}{J^2}\right) - 2\phi_\xi\phi_\nu\Delta\left(\frac{g_{12}}{J^2}\right) + \phi_\eta\Delta\left(\frac{g_{11}}{J^2}\right)\right\} \tag{31}$$

The individual truncation errors in this modified equation are computed by applying the chain rule. For example,

$$\Delta\left(\frac{g_{11}}{J^2}\right) = \frac{\Delta\xi^2}{3}\left(\frac{x_\xi x_{\xi\xi\xi} + y_\xi y_{\xi\xi\xi}}{J^2}\right) - \frac{g_{11}}{3J^3}\Delta J$$

and

$$\Delta J = \Delta\eta^2(x_\xi y_{\eta\eta\eta} - y_\xi x_{\eta\eta\eta}) + \Delta\xi^2(y_\eta x_{\xi\xi\xi} - x_\eta y_{\xi\xi\xi}) \tag{32}$$

Here again, the effect of the truncation errors of the metrics and the solution (ϕ) are somewhat separable. In equation (31) the first group of terms on the right hand side is the truncation due to the differencing of the solution. This part of the error still has the metrics as coefficients. The second group of terms is the mesh-induced truncation error and also depends on the solution. Only in this sense are the two errors separable. Thus, for a given solution, one could check the adequacy of a mesh by computing the two separate errors as above and require that the mesh induced errors be no larger than the solution-induced errors over the entire solution domain.

This is not a very satisfactory result in that a mesh-induced truncation error only has meaning with respect to a particular flow field solution, so that there is no completely independent criterion for "mesh goodness."

7. NUMERICAL RESULTS

The grid is shown in Figure 2. Shown is a C-type mesh around a NACA 0012 airfoil. The far field boundaries (not shown) are approximately six chord lengths from the airfoil in all directions. It consists of 175×31 node points and is generated by use of the Poisson equation with some grid control (Sorenson[7]). A solution in terms of density is shown in Figure 3 for a free stream Mach number $M_\infty = 0.80$ and angle of attack $\alpha = 0°$. These results are presented as carpet plots with computational variables I, J (representing the "circumferential" computational variable $\xi(I)$ and the "radial" computational variable $\eta(J)$, respectively) as independent variables. The airfoil surface is given by J = 31 and the far field boundary by J = 1. The leading edge of the airfoil is at I = 88 and the trailing edge is at I = 31. The plots are distorted, but this should cause no confusion. The viewpoint of Figure 3 and all the subsequent figures, except where noted, is from the far field boundary and the trailing edge, or the lower left hand corner of Figure 2. The rise of the density at the leading edge stagnation point and at the shock wave is quite apparent.

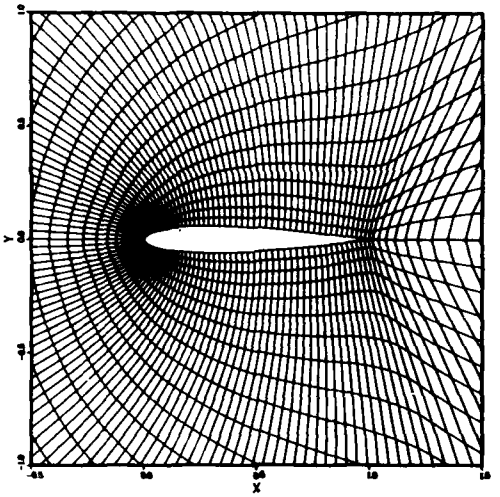


Fig. 2. Grid plot of a C-mesh around a NACA 0012 airfoil - 175×31 mesh points.

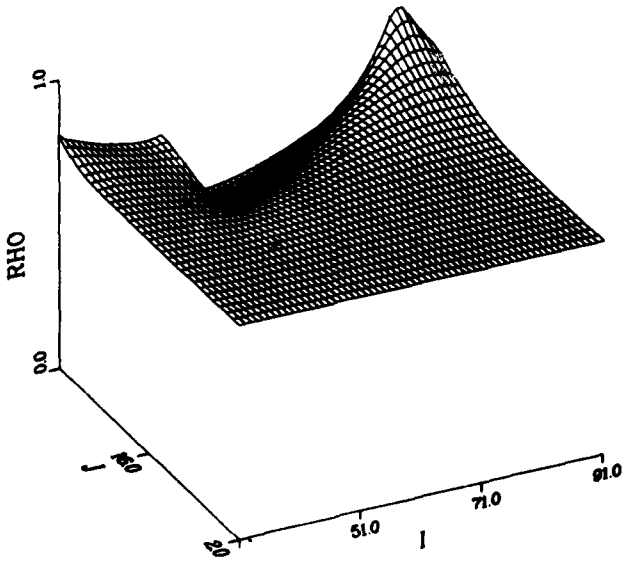


Fig. 3. Carpet plot of density solution (density vs. grid index) as solved by TAIR C (ref. 6) on lower half of NACA 0012 airfoil, $M_\infty = 0.8$, $\alpha = 0.$ deg. Viewpoint near trailing edge and far field boundary (lower left hand corner of Fig. 2).

The following set of four figures present the components of the Jacobian matrix. These are x_ξ, y_ξ, x_η, and y_η in Figures 4, 5, 6, and 7, respectively. These figures indicate that the largest variations of the metrics occur near the trailing edge and at the far field boundary. They are quite uniform near the leading edge. This indicates that the mesh has been sufficiently refined there to reduce the orientation angle variation to acceptable levels. It may be possible, however, that extra resolution is needed to resolve the peak of stagnation density accurately.

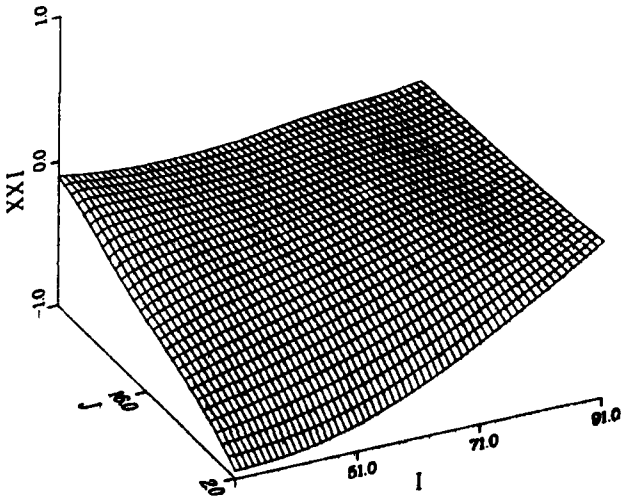


Fig. 4. Carpet plot of Jacobian matrix component x_ξ normalized by 0.1773.

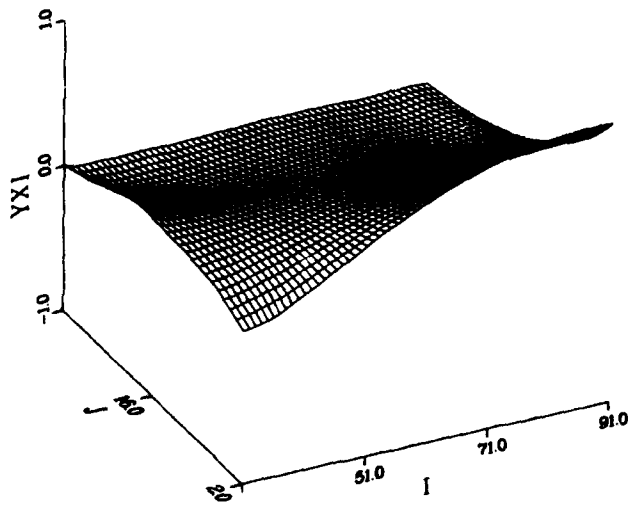


Fig. 5. Carpet plot of Jacobian matrix component y_ξ normalized by 0.1773.

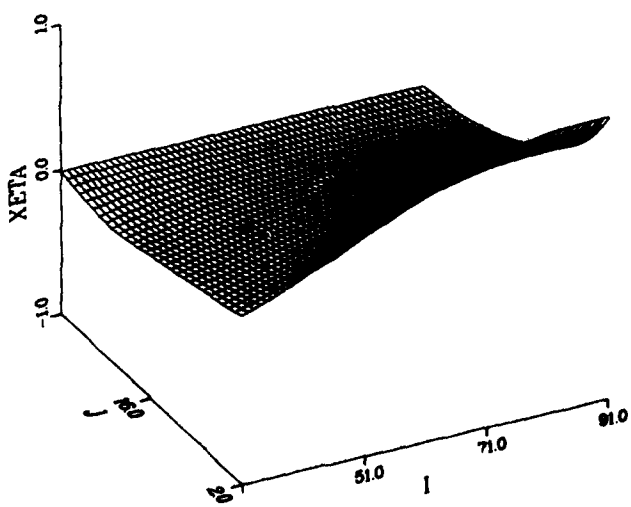


Fig. 6. Carpet plot of Jacobian matrix component x_η normalized by 0.6653.

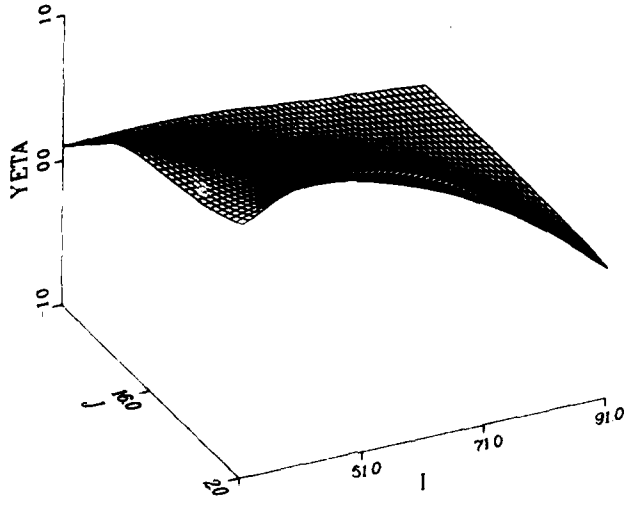


Fig. 7. Carpet plot of Jacobian matrix component y_η normalized by 0.3993.

As before, we separate out the rotation parameter α. Its cosine is plotted. in Figure 8. The sharp turning of the mesh about the leading edge is clearly in evidence, and as noted before, this can lead to significant mesh-induced truncation error.

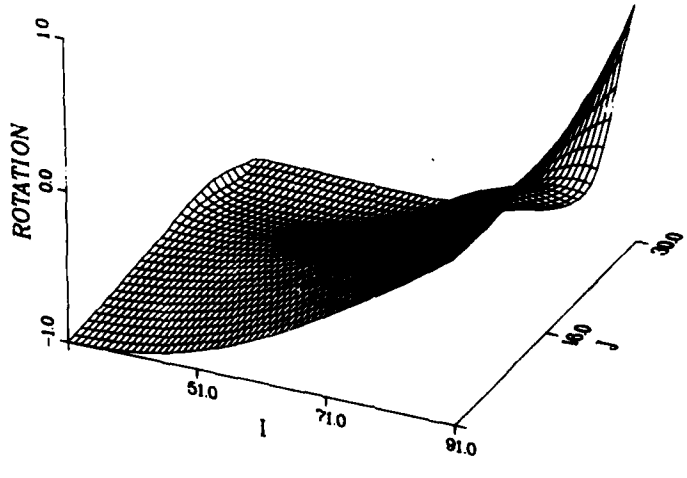


Fig. 8. Carpet plot of cell orientation (rotation) parameter cosα, normalized by 1.00. Viewpoint is from lower right hand side of Fig. 2, far field, leading edge.

The next two Figures (9,10) show the metric tensor components g_{11} and g_{22}, normalized by factors of 0.03143 and 0.4427, respectively. They are nearly uniform except near the trailing edge and the far field boundary, with the largest value g_{22} being about an order of magnitude more important (its normalization factor is about ten times larger).

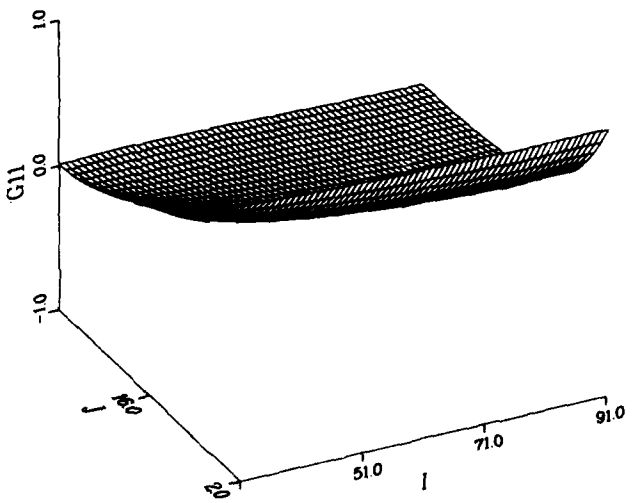


Fig. 9 Carpet plot of metric tensor component g_{11} normalized by 0.0314.

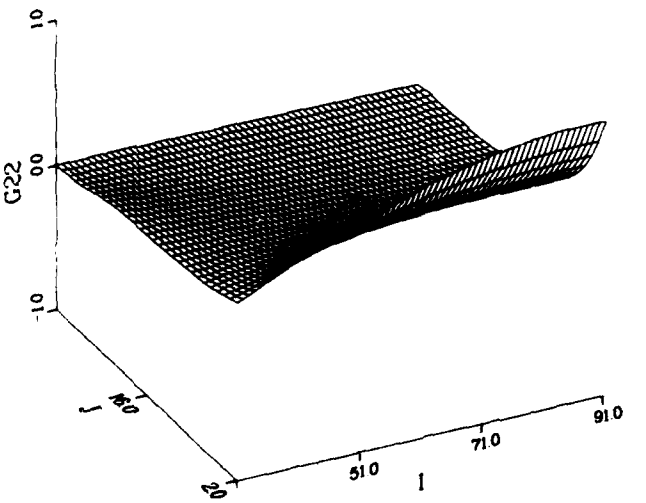


Fig. 10. Carpet plot of metric tensor component g_{22} normalized by 0.4427.

The parameters based on the decomposition of the metric tensor are shown next. Figure 11 shows the Jacobian determinant $|\partial(x,y)/\partial(\xi,\eta)|$. As one would expect, the only large values occur near the far field, and the variation with grid index is reasonably smooth there. Of more importance is the variation near the airfoil surface, illustrated by the plot of the inverse Jacobian in Figure 12. Here one can see the near singular behavior of the coordinate system near the leading edge, although here again, the variation is fairly smooth.

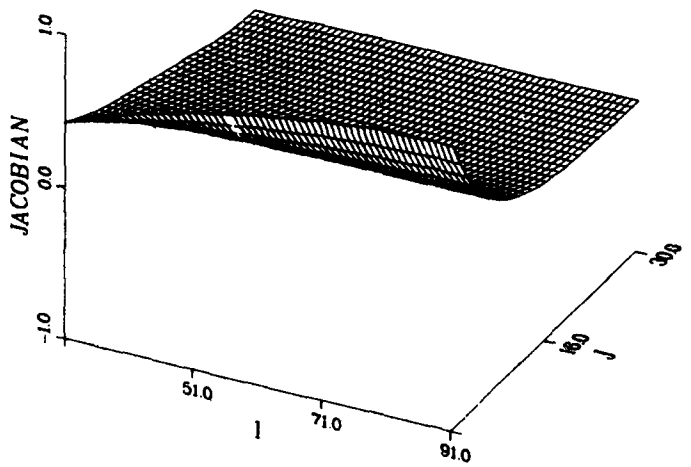


Fig. 11. Carpet plot of the Jacobian determinant (cell volume parameter) $J = |\partial(x,y)/\partial(\xi,\eta)|$ normalized by 0.1179. Viewpoint at far field leading edge.

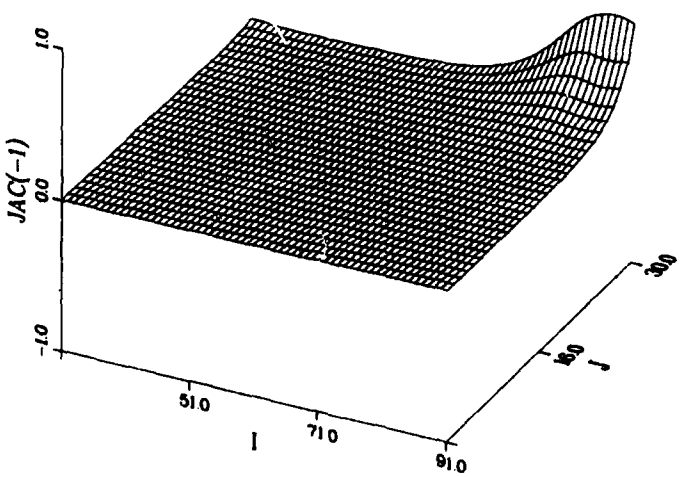


Fig. 12. Carpet plot of the inverse Jacobian determinant $J^{-1} = |\partial(\xi,\eta)/\partial(x,y)|$ normalized by 32460.

The cell aspect ratio A is plotted in Figure 13, normalized by 3.870. The cells with the large aspect ratio occur near the cut at the trailing edge. The cell orthogonality parameter cosθ is plotted in Figure 14. There are significant departures from orthogonality in the entire near field, and rapid variations near the leading and trailing edges.

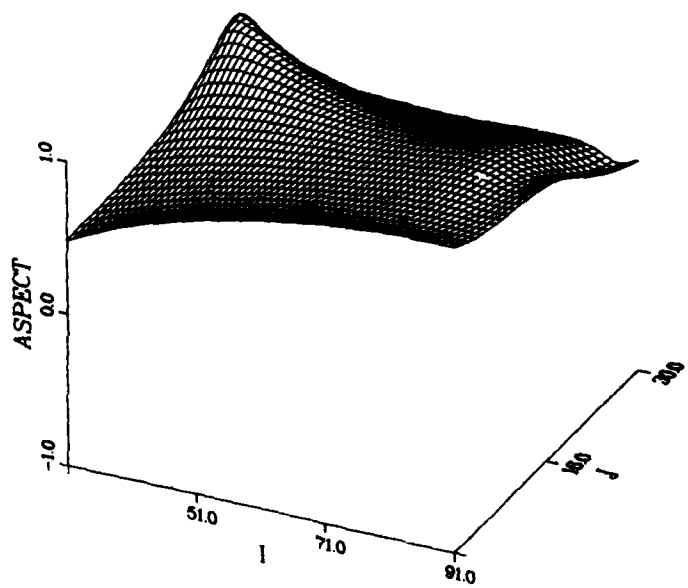


Fig. 13. Carpet plot of cell aspect ratio parameter A, normalized by 3.870.

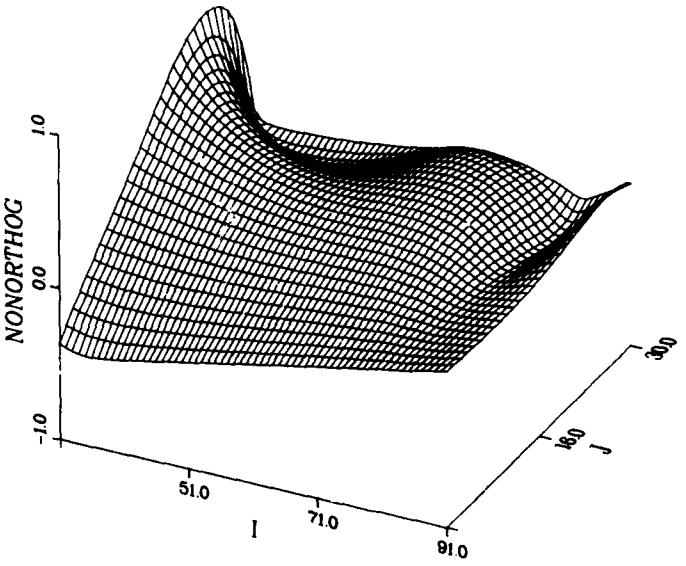


Fig. 14. Carpet plot of cell orthogonality parameter cosθ, normalized by 1.00.

Based on these results one would expect most of the mesh-induced truncation errors to occur near the trailing edge and at the far field boundary. That this is indeed the case is confirmed in Figure 15. Shown here is the second group of terms of equation (31) in terms of percentage of ρ_N (the numerically computed density). The mesh-induced errors occur mostly near the trailing edge and the far field boundary. They are on the order of 2 to 3%.

The solution-induced truncation error [first group of terms of equation (31)] is shown in Figure 16. These errors are much larger. They are on the order of 1-2% near the leading edge, 5% at the shock wave, <5% at the trailing edge and 2% at the far field. The total error is shown in Figure 17. As can be seen some of the errors at the trailing edge and far field have been cancelled. This is, however, a fortuitous circumstance and, in general, should not be expected to occur. The errors are still approximately 2% at the leading and trailing edges and 5% at the shock wave.

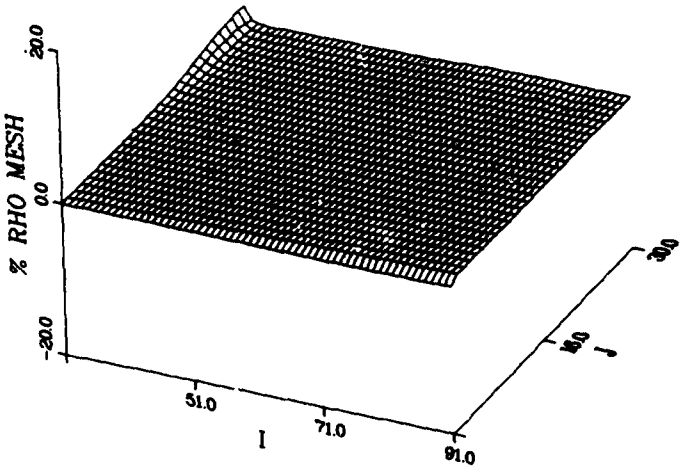


Fig. 15. Truncation error due to the mesh $(TE)_g$, expressed as a percentage of density ρ_N.

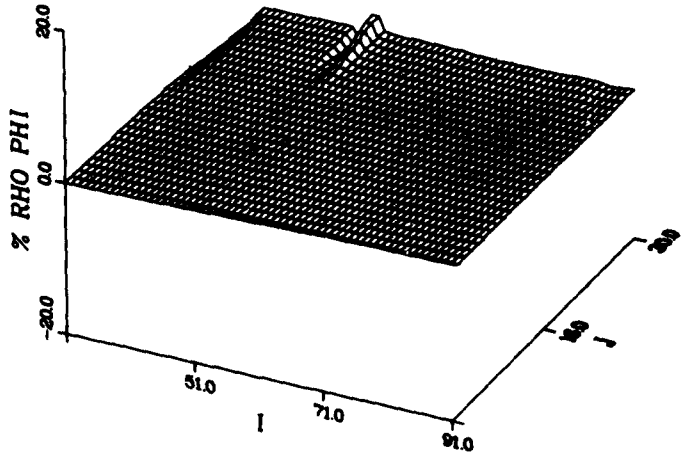


Fig. 16. Truncation error due to differencing the solution $(TE)_f$, expressed as a percentage of density ρ_N.

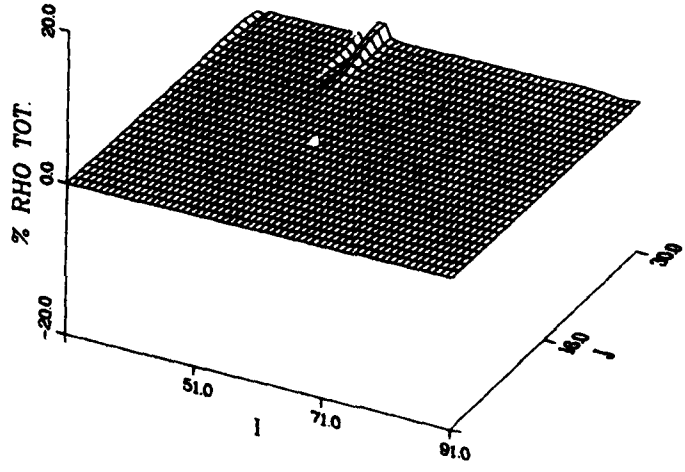


Fig. 17. Total truncation error in the numerical solution for the density, expressed as a percentage of density ρ_N.

One concludes that the mesh of Figure 2 is probably adquate for controlling the mesh-induced errors, if 5% solution errors are acceptable. If drag calculations are to be attempted with this mesh and numerical procedure, then 2% accuracy at the trailing and leading edge is probably not acceptable. The errors would need to be reduced by further mesh refinement.

8. FURTHER APPLICATIONS AND CONCLUSIONS

The parameters introduced in this paper are useful in several ways. First, they provide a quantitative expression for the qualities of each mesh cell, which is useful in its own right.

Second, the local truncation error due to the mesh is expressible in terms of the components of $(\partial x^i/\partial \xi^j)$ and derivatives. These can be recomputed, with much manipulation, in terms of the mesh cell parameters, and their derivatives. For a given numerical scheme, it is possible to see how the various mesh properties affect the error. This information can then be used to modify the original mesh either initially or in the course of the calculation.

These parameters are also natural choices for the Lagrange multipliers for a mesh generation algorithm like that of Brackbill,[8] which is based on a variational principle. Depending on the scheme used, one can also construct variational principles which minimize the leading truncation error of the algorithm.

The variables J, A, θ introduced above are related to Brackbill's variables which describe volume, smoothness, and orthogonality. In addition, we have introduced the local rotation α. This parameter can be important in fluid flow problems where there are regions of high curvature, like the regions near the leading and trailing edges of wings.

The extension of these methods to three dimensions is conceptually clear. The manipulations involved are, however, so long that machine calculation of the algebra is a practical necessity.

REFERENCES

1. Klopfer, G. H. (1982) Analysis of a Finite Difference Grid. Nielsen Engineering TR 267.
2. Warsi, Z. U. A. and Thompson, J. F. (1980) Numerical Generation of Two-Dimensional Orthogonal Curvilinear Coordinates in Euclidean Space. NASA CP-2166.
3. Do Carmo, M. P. (1976) Differential Geometry of Curves and Surfaces. Prentice-Hall.
4. Schouten, J. A. (1954) Ricci-Calculus. Springer-Verlag.
5. Holst, T. L. (1979) An Implicit Algorithm for the Conservative Transonic Full Potential Equation Using an Arbitrary Mesh. AIAA Journal, Vol. 17, pp. 1038-1056.
6. Dougherty, F. C. (1982) TAIC Code, in preparation. NASA Ames Research Center.
7. Sorenson, R. L. and Steger, J. L. (1980) Numerical Generation of Two-Dimensional Grids by the Use of Poisson Equations with Grid Control at Boundaries. NASA Conference Publications 2166.
8. Brackbill, J. U. and Saltzman, J. (1980) "An Adaptive Computational Mesh for the Solution of Singular Perturbation Problems," in Proceedings of the Workshop on Numerical Grid Generation Technique for Partial Differential Equations, NASA CP-2166.

AD P001004

Copyright 1982 by Elsevier Science Publishing
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

AN IMPLICIT SCHEME FOR WATER WAVE PROBLEMS

MARTHA B. ASTON[+] AND J. W. THOMAS[++]
[+]Ph.D. student, present address: Jet Propulsion Laboratories, Pasadena, Calif.
[++]Professor, Dept. of Mathematics, Colorado State Univ., Ft. Collins, Colo.

ABSTRACT

This paper presents an implicit scheme for numerically simulating fluid flow in the presence of a free surface. The scheme couples numerical generation of a boundary-fitted coordinate system with an efficient alternating-Direction-Implicit solution of the finite difference equations. Solutions using the scheme presented here indicate that the scheme can be applied successfull to free surface fluid flow problems.

INTRODUCTION

This paper presents the initial development of a scheme to numerically simulate two-dimensional flow of an incompressible viscous fluid with a free surface. Currently established numerical simulation methods are capable of solving free surface problems accurately[1,2,3]. However, these methods are usually most successful for particular geometries or types of problems. In addition, these methods generally depend on explicit or iterative calculation of the flow boundary conditions and the free surface location. The numerical scheme presented in this paper is an entirely implicit solution technique and demonstrates efficient calculation of implicit boundary conditions and free surface location. The scheme also incorporates a curvilinear coordinate generation technique which can follow the free surface and hence should be adaptive to the solution of problems involving complicated geometries.

Numerical simulation of free surface fluid flows is complicated because the deforming free surface creates a time dependent and often complex geometry. Furthermore, location of the free surface must be part of the solution. Also, boundary conditions on the free surface are nonlinear and must be applied accurately because they strongly influence flow throughout the entire domain. Numerically generating a dynamically adaptive boundary-fitted coordinate system can help overcome these problems. Such a coordinate system can be generated automatically by an appropriate set of partial differential equations which yield coordinate lines coincident with all boundaries of the physical problem. Several authors have shown the applicability of numerical grid generation schemes to the solution of free surface problems[4,5]. One aim of the present

PREVIOUS PAGE IS BLANK

work is to improve the numerical efficiency with which such grid generation schemes may be implemented to solve free surface fluid flow problems.

To achieve increased efficiency and favorable stability properties, the numerical solution method presented here uses a full implicit backward difference scheme coupled with an Alternating-Direction-Implicit (ADI) matrix method. No iteration is required to compute the solution for a single time step. To improve adaptivity of grid generation techniques to the implicit numerical method, a variation of the usual grid generating equations is used. The mapping functions in this new scheme satisfy parabolic equations rather than the usual elliptic equations. The parabolic grid generating equations may be solved simultaneously with the fluid equations. The reason we have used parabolic equations for coordinate generation is to try to improve the ability of the calculational domain to follow the flowing fluid.

The aim of this work is to investigate the feasibility of implementing the above described numerical scheme to the solution of free surface fluid flow problems. Consequently, much effort has been expended on correctly coupling the various elements in the solution scheme to produce a reasonable system of equations which may be solved without the imposition of restrictive stability criteria. This emphasis, plus the limited computational resources available to the authors, has resulted in only preliminary testing to date of a few simple fluid flow geometries. Also, as of yet the scheme includes no capability to cluster grid lines. However, the scheme could allow for such terms in the mapping equations and there are plans to add such terms. All results concerning stability and parameter values are due to numerical experimentation.

MATHEMATICAL FORMULATION

Consider the problem of determining the two-dimensional flow of an incompressible, viscous fluid through a channel with a rough bottom. Figure 1 illustrates schematically such a geometry. In Figure 1, the free and the bottom surfaces are denoted by $y = H(x, t)$ and $y = F(x)$, respectively, and the physical region of flow by D_F.

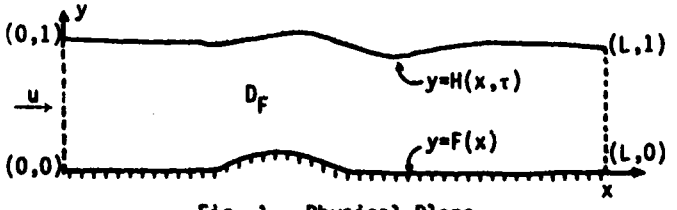


Fig. 1. Physical Plane

The equations of motion are the Navier-Stokes equations in the primitive variable form. Continuity of mass is prescribed by the divergence equation. Written in non-dimensional form, the equations to be solved are the following:

$$\underset{\sim}{u}_t + (\underset{\sim}{u}\cdot\nabla)\underset{\sim}{u} = -\nabla p + \frac{1}{Re}\nabla^2\underset{\sim}{u} + E \quad \text{in } D_F \tag{1}$$

$$\nabla\cdot\underset{\sim}{u} = 0 \quad \text{in } D_F \tag{2}$$

where $\underset{\sim}{u}$ denotes the velocity vector, p denotes the pressure, Re is the Reynolds number and E is the external force (specifically, E contains the gravity term, $\frac{1}{Fr^2}$, where Fr is the Froude number).

Using an ADI scheme to solve equation (1), the velocity field may be advanced in time. However, the role of pressure in these equations and equation (2) presents computational difficulties. Introducing an artificial compressibility term into the continuity equation makes it possible to determine the pressure in a natural and efficient way. The equation of artificial compressibility is

$$\beta p_t = -\nabla\cdot\underset{\sim}{u} \tag{3}$$

where $\beta << 1$ is the coefficient of artificial compressibility. This equation fits naturally into an ADI scheme. It should be noted that Steger[6] and Chorin[7] have used the concept of artificial compressibility to successfully simulate incompressible fluid flows. Indeed, Teman[8] proves, for a certain numerical scheme, that the solution of equations (1) and (3) with appropriate boundary conditions converges to the solution of equations (1) and (2) as $\beta \to 0$.

The most important boundary associated with free surface problems is the free surface itself. Boundary conditions on the free surface are specified as follows. Let the pressure be constant on the free surface, i.e.,

$$p = P_0 \quad \text{on } y = H(x, t). \tag{4}$$

Movement of the free surface, $y - H(x, t) = 0$ is determined by requiring the convective derivative to vanish. This results in the condition

$$H_t + uH_x - v = 0 \quad \text{on } y = H(x, t). \tag{5}$$

To enhance the computational stability of the free surface we add a numerical viscosity term[11] and hence, consider the following equation:

$$H_t + uH_x - v - \nu H_{xx} = 0. \tag{6}$$

The quantity ν is defined as the coefficient of artificial viscosity. Finally, continuity of stress is imposed at the free surface. Applying condition 4 one obtains on the free surface

$$(7)\qquad \begin{cases} \frac{1}{R_e}[2u_xH_x - (u_y + v_x)] = 0 \\ \frac{1}{R_e}[(u_y + v_x)H_x - 2v_y] = 0 \end{cases} \qquad \text{on } y = H(x, t).$$

Along the bottom, free slip boundary conditions are imposed (though no slip conditions could also be applied). At the inflow and outflow boundaries, a uniform flow with hydrostatic pressure is assumed. For computational requirements, the inflow and outflow conditions are set at finite values of the channel length. The equations describing this uniform flow at the inflow and outflow boundaries take the form

$$(8)\qquad n \cdot \underset{\sim}{u} = u_0$$

$$(9)\qquad p = P_0 + (H(x, t) - y)/Fr^2 \quad \text{on} \quad x = 0,\ x = L.$$

Here n is the unit normal to the boundary (in the direction of flow) and Fr is the Froude number. An accelerated start is used to initialize the fluid flow. Beginning with zero fluid velocities and hydrostatic pressure, an acceleration term is used to increase the velocity terms from zero to their uniform flow values.

The goal of the mathematical scheme presented here is to find the solution to equations (1), (2), (4), (5) and (7). This solution is sought for the boundary conditions defined by equations (8) and (9) and is determined by actually solving equations (1), (3), (4), (6) and (7) for small values of the coefficient of artificial compressibility, β.

NUMERICAL GRID GENERATION

Some sort of grid generation is very useful when solving free surface problems. Many physical problems that must be solved numerically have complicated geometries. Free surface problems have both complicated and time dependent geometries. For this reason, the authors feel that numerical mapping techniques hold much promise for accurate solutions of free surface problems.

The basic grid generating technique employed here follows a scheme similar to those used by Shanks and Thompson[5]. Assume that the physical region D_F illustrated in Figure 1 is mapped onto a fixed rectangle, Q, in the $\xi - \eta$ plane, with boundaries of D_F mapping to boundaries of Q. For example, suppose that the free surface, AB, maps onto the top of rectangle Q, the bottom, DC, maps onto the bottom of Q and the sides, AD and BC, map onto the left and right sides of Q, respectively. Such a mapping is shown in Figure 2. Note that it is not necessary that we map the region D_F in such a straightforward way to the $\xi - \eta$ plane. In more complex geometries the mapping will be more complex. For

the preliminary results presented here, the above described configuration and mapping was considered sufficient to test overall feasibility of the scheme.

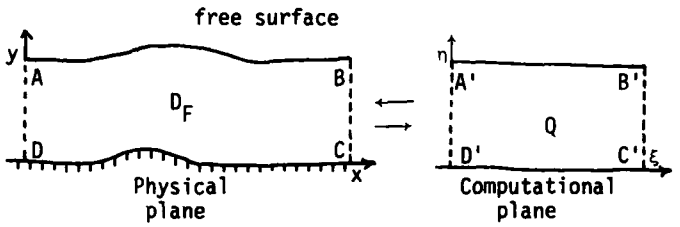


Fig. 2. Free surface problem transformation

Let $\xi = \xi(x, y, t)$ and $\eta = \eta(x, y, t)$ denote the mapping functions. The variables ξ and η are chosen to satisfy the following partial differential equations:

$$\varepsilon\xi_t - \nabla^2\xi = 0 \tag{10}$$

$$\varepsilon\eta_t - \nabla^2\eta = 0. \tag{11}$$

(Boundary conditions for these equations will be discussed later.) Other authors[5] frequently require that each mapping function satisfy a Poisson equation and thus maintain some control over coordinate line spacing. The present scheme could be modified in this manner. It is felt that the inclusion of coordinate placement capabilities will be necessary before the scheme presented here can be successfully applied to more difficult geometries such as flow about a cylinder.

The difference between grid generating equations (10) and (11) and other more common grid generating equations is the addition of the $\varepsilon\xi_t$ and $\varepsilon\eta_t$ terms. Note that by taking ε small (or even zero), the usual schemes can be approximated. However, the primary purpose for introducing the time derivative terms in equations (10) and (11) is to allow experimentation based on different values of ε. Specifically, it is hoped that providing a truely dynamic grid may better represent the flow of the coordinate system in union with the changing physical domain.

As is true for any such grid generating scheme, equations (1), (3), (4), (6), (7), (10) and (11) are solved in the $\xi - \eta$ plane rather than the $x - y$ plane. After transforming the system to the $\xi - \eta$ plane the following system

of equations is obtained:

$$(12)\quad (J\underset{\sim}{q})_t + \{J[\underset{\sim}{q}(y_t x_\eta - x_t y_\eta) + \underset{\sim}{f} y_\eta - \underset{\sim}{g} x_\eta]\}_\xi + \{J[\underset{\sim}{q}(x_t y_\xi - y_t x_\xi) - \underset{\sim}{f} y_\xi + \underset{\sim}{g} x_\xi]\}_\eta = \frac{1}{R_e}\{[\frac{y_\eta}{J}(Dy_\eta \underset{\sim}{q}) + \frac{x_\eta}{J}(Dx_\eta \underset{\sim}{q})]_\xi - [\frac{y_\eta}{J}(Dy_\xi \underset{\sim}{q}) + \frac{x_\eta}{J}(Dx_\xi \underset{\sim}{q})]_\eta\}_\xi + \frac{1}{R_e}\{[\frac{x_\xi}{J}(Dx_\xi \underset{\sim}{q}) + \frac{y_\xi}{J}(Dy_\xi \underset{\sim}{q})]_\eta - [\frac{x_\xi}{J}(Dx_\eta \underset{\sim}{q}) + \frac{y_\xi}{J}(Dy_\eta \underset{\sim}{q})]_\xi\}_\eta + E$$

$$(13)\quad \varepsilon J^2 x_t - \alpha x_{\xi\xi} + 2\beta_1 x_{\xi\eta} - \gamma x_{\eta\eta} = 0$$

$$(14)\quad \varepsilon J^2 y_t - \alpha y_{\xi\xi} + 2\beta_1 y_{\xi\eta} - \gamma y_{\eta\eta} = 0$$

$$(15)\quad H_t + \frac{u}{J}(H_\xi y_\eta - H_\eta y_\xi) - v + \frac{\nu}{J}\{[\frac{y_\eta}{J}(y_\eta H_\xi - y_\xi H_\eta)]_\xi - [\frac{y_\xi}{J}(y_\eta H_\xi - y_\xi H_\eta)]_\eta\} = 0$$

$$(16)\quad u_\eta = \frac{1}{\gamma}(\beta_1 u_\xi - J v_\xi)$$

$$(17)\quad v_\eta = \frac{1}{\gamma}(J u_\xi + \beta_1 v_\xi)$$

$$(18)\quad p = P_0$$

where $\underset{\sim}{q} = [u, v, \beta p]^T$, $\underset{\sim}{f} = [p + u^2, uv, u]^T$, $\underset{\sim}{g} = [uv, p + v^2, v]^T$, J is the Jacobian of the transformation,

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$\alpha = x_\eta^2 + y_\eta^2$, $\beta_1 = x_\xi x_\eta + y_\xi y_\eta$, $\gamma = x_\xi^2 + y_\eta^2$ and equations (15)-(18) are evaluated along the top of the rectangle in the $\xi - \eta$ plane. Several remarks should be made. First, equation (12) is in conservation law form. Secondly, equations (15) through (18) are on a line of constant η which maps to the line $y - H(x, t) = 0$. This makes it possible to eliminate the free surface height, H, from equation (15). Simplifying gives

$$(19)\quad y_t + u\frac{y_\xi}{x_\xi} - v + \nu[\frac{x_\xi y_{\xi\xi} - y_\xi x_{\xi\xi}}{x_\xi^3}] = 0.$$

Thirdly, the mapping functions $x = x(\xi, \eta, t)$ and $y = y(\xi, \eta, t)$ are time dependent unknowns which must be determined at each time step. For applications of grid generating schemes to problems without a free surface, the mapping functions are time independent and hence need to be determined only once. Finally, the boundary conditions on the bottom, inflow and outflow are unchanged by the transformation, except they have a new domain.

Boundary conditions for equations (13) and (14) are defined so that the boundary of Q maps onto the boundary of D_F (note that this is the inverse of

the mapping given by equations (10) and (11)). This results in the requirements that $x(0, \eta, t) = 0$, $y(0, \eta, t) = \eta$, $x(1, \eta, t) = L$, $y(1, \eta, t) = \eta$, $x(\xi, 0, t) = L\xi$ and $y(\xi, 0, t) = F(x(\xi, 0, t), t)$ where for simplicity Q has been defined as the unit square.

NUMERICAL SCHEME

The goal of the numerical scheme is to provide an efficient, noniterative and stable method for solving equations (12)-(18) presented in the previous section. As mentioned, an ADI scheme is used for numerical efficiency. In addition, all unknowns, including the flow variables and mapping variables both within the domain of flow and on the free surface are advanced through time simultaneously. Hence computational efficiency is improved by having one iteration advance the entire solution one time step. Moreover, the authors feel that, because the free surface strongly effects the solution throughout the entire domain, free surface values should be calculated with the values for the rest of the fluid. It is hoped that ultimately this totally implicit method will permit larger time steps and speed convergence to steady state.

The numerical scheme presented here is analogous to those described in reference 9. The format of the scheme includes:

i) linearization of the nonlinear terms by Taylor series expansion about the previous (known) time level,

ii) a backward time, centered space differencing,

iii) an expression of the equations as a system of coupled, linear difference equations, and

iv) a Douglas-Gunn[12] splitting to generate the ADI system of equations.

The final result is two one-dimensional linear difference equations that can be solved efficiently by standard block elimination techniques. The above outline is simplified, ignoring the many computations (and seemingly endless detail) necessary to get the equations in their final form. In fact, the authors found it necessary to analyze each equation individually. Each equation, including boundary conditions, was linearized, differenced and reduced to a canonical form. The equations were then put back together and the two ADI block tridiagonal equations were determined. Note, again, that the final ADI system includes finite difference expressions of all flow equations and boundary conditions.

When $\underset{\sim}{\mu}^n = [u, v, p, x, y]^T$ at time step $n = n\Delta t$ is known, the ADI system determines $\underset{\sim}{\mu}^{n+1}$ in the two steps shown below:

$$[A^n - \Delta t D_\xi^n](\underset{\sim}{\mu}^* - \underset{\sim}{\mu}^n) = \Delta t[(D_\xi^n + D_\eta^n + D_{\xi\eta}^n)\underset{\sim}{\mu}^n + B^n] \tag{19}$$

$$[A^n - \Delta t D_\eta^n](\underset{\sim}{\mu}^{n+1} - \underset{\sim}{\mu}^n) = A^n(\underset{\sim}{\mu}^* - \underset{\sim}{\mu}^n) \tag{20}$$

where A^n, D_ξ^n, D_η^n and $D_{\xi\eta}^n$ are 5 x 5 matrices and B^n is a 5 x 1 vector. The matrices and the vector B^n are constructed from known values at the n^{th} time step. The matrix D_ξ^n contains the ξ spatial difference operators while D_η^n contains the η spatial difference operators. Similarly, $D_{\xi\eta}^n$ defines the cross difference operator; this operator is lagged in the ADI system. The term $\underset{\sim}{u}^*$ is an intermediate solution.

It is possible to speed computation by reducing the number of arithmetic operators required to solve (19) and (20). Each of the matrices A^n, D_ξ^n and D_η^n, that determine the block elements of the tridiagonal systems has the following form

$$\left[\begin{array}{ccc:cc} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ \hdashline 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{array}\right]$$

where x denotes a nonzero entry. Thus, the block tridiagonal solver need only invert 3 x 3 and 2 x 2 blocks. This partition bypasses the computationally more difficult inversion of the original 5 x 5 blocks. Computation time should also be shortened by adapting the ADI system to a vector processors. Unfortunately, such a facility was not available to the authors.

NUMERICAL RESULTS

Most of the authors' computation effort to date has been concentrated on the problem illustrated in Figure 1. The bump on the bottom of the channel has a height corresponding to 20% of the total fluid depth. Initially, the fluid depth, $H(x, t) - F(x)$, is set to one with initial velocities $u = 0$ and $v = 0$. An acceleration term is applied over the entire domain on each of the first five time steps to increase the velocities to their uniform (non-dimensional) flow values of $u = 1.0$ and $v = 0.0$. The fluid flow is specified by a Reynolds number, $Re = 20.0$ and a Froude number, $Fr = 2.0$. Computational parameters are $\beta = 0.02$, $\varepsilon = 20.0$ and we usually used $\Delta t = 0.02$. The calculational domain is defined by a 37 x 11 grid. At the end of sixteen iterations, the results appear to be reasonable, although convergence to a steady solution has not been achieved. The shape of the free surface generally follows trends expected from analytic results. Free surface location at time $t = .32$ is shown in Figure 3.

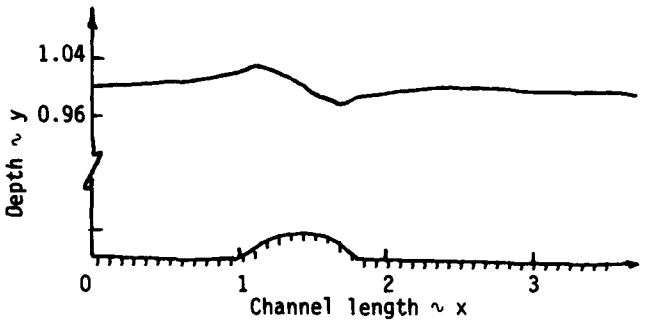


Fig. 3. Numerical results, Re = 20.0, Fr = 2.0, time t = .32 (Δt = 0.02).

At this point it is difficult to fairly access the stability of the scheme. Limited computer resources have prevented the authors from completing more than sixteen iterations for the 37 x 11 grid. With such a few iterations it is not possible to perform meaningful experimentation with a variety of time steps. Nonetheless, the scheme behaves well with time steps up to $\Delta t = 0.05$ for the iterations completed to date. This time step seems to indicate acceptable numerical stability. Further stability analysis and numerical experimentation are merited. It should be noted that we have not yet added any artificial dissipation[10]. In the future, numerical dissipation terms will be added.

The two parameters, ε and β appear to have a significant effect on the solution. Although optimum values of these parameters have not been determined some observations have been made. Values of ε with size on the order of the Reynolds number ($\varepsilon \sim \mathcal{O}(Re)$) permit the grid to move well with the fluid. Unexpectedly, when ε was chosen small ($\varepsilon \sim \mathcal{O}(\frac{Re}{100})$) the grid disassociated itself from the fluid flow and numerical instability resulted. Various values of β were also examined for their effect on the solution. In keeping with the principle of artificial compressibility, it was expected that obtaining reasonable solutions would be dependent on small values of β. However, for β in the range of 2×10^{-4} to 2×10^{-2}, results were nearly identical. Since larger values of β appeared to provide better stability for larger values of Δt, β was set at the value 2×10^{-2}.

The computed results given above were obtained on a CYBER 170. Each iteration required roughly 1.5 mins of CPU execution time. It should be noted that no attempt has been made to optimize computer programming efficiency. Thus, the time per iteration, though currently longer than desired, seems

reasonable when compared to iteration times for other methods which solve free surface flow problems.

SUMMARY

This paper presents an entirely implicit scheme for numerically simulating fluid flow in the presence of a free surface. The scheme includes numerical generation of a boundary-fitted coordinate domain in which all calculations are performed. An efficient solution of the flow equations is achieved by applying an Alternating-Direction-Implicit method to solve the transformed, linearized, differenced equations. Algebraic complexity of the system is reduced by determining a canonical form for the individual equations. Efficiency is increased by inclusion of all boundary conditions in the final matrix equations. Solutions obtained using this numerical scheme indicate that it can be applied successfully to free surface fluid flow problems.

ACKNOWLEDGEMENTS

The authors would like to thank the late J. C. Bell for his helpful discussions and Graeme Aston for his assistance in preparing this manuscript.

REFERENCES

1. Schot, Joanna and Salvesen, Nils, eds., (1975) Proc. of the First Int'l. Conf. on Numerical Ship Hydrodynamics, David Taylor Naval Ship Research and Development Center, Bethesda, MD.
2. _____, (1977) Proc. of the Second Int'l. Conf. on Numerical Ship Hydrodynamics, Univ of Calif., Berkeley.
3. _____, (1981) Proc. of the Third Int'l. Conf. on Numerical Ship Hydrodynamics, Paris, France.
4. Haussling, H.J. and Coleman, R.M., (1977) Finite-difference computations using body-fitted coordinates for free surface potential flows generated by submerged bodies, Second Int'l. Conf. on Numer. Ship Hydrodynamics, Univ. of Calif., Berkeley.
5. Shanks, S.P. and Thompson, J.F. (1977) Numerical solution of the Navier-Stokes equation for 2D hydrofoils in or below a free surface, Second Int'l. Conf. on Numer. Ship Hydrodynamics, Univ. of Calif., Berkeley.
6. Steger, Joseph L. and Kutler, Paul. (1976) Implicit finite-difference procedures for the computation of vortex wakes, AIAA Paper 76-385.
7. Chorin,A.J. (1967) A numerical method for solving incompressible viscous flow problems, J. of Comp. Phys., 2, 12-26.
8. Teman, Roger. Navier-Stokes Equations: Theory and Numerical Analysis, North Holland.
9. Briley, W.R. and McDonald, H. (1980) On the structure and use of linearized block implicit schemes, J. of Comp. Phys. 34, 54-73.
10. Pulliam, T.H. and Steger, J.L. (1978) On implicit difference simulations of three-dimensional flow, AIAA 16th Aerosp. Sci. Mtg, Huntsville, AL.
11. Nichols,B.D. and Hirt,C.W. (1973) Calculating Three-Dimensional Free Surface Flows in the Vicinity of Submerged and Exposed Structures, J. Comp. Phys. 12, 234-246.
12. Douglas,J. and Gunn,J.E. (1964) A General Formulation of Alternating Direction Methods, Numer. Mathemati 6, 428-453.

AD P001005

Published 1982 by El:
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

A VECTORIZED, FINITE-VOLUME, ADAPTIVE-GRID ALGORITHM FOR NAVIER-STOKES CALCULATIONS

PETER A. GNOFFO[+]

[+]Aero-Space Technologist, Aerothermodynamics Branch, Space Systems Division, NASA Langley Research Center, Hampton, Virginia 23665, Graduate Student, Department of Mechanical and Aerospace Engineering, Princeton University.

ABSTRACT

An adaptive grid, finite-volume method has been used to solve the Navier-Stokes equations for complete (forebody and afterbody) flowfields around blunt bodies. The code, which is applicable for axisymmetric or two-dimensional flows, allows the mesh to adjust during the computation to provide a closer spacing of mesh points in regions of high gradients, thus minimizing the number of required computational points. The solution technique is explicit, utilizing a maximum time-step advancement at each grid point to accelerate convergence to the steady state. The code has been fully vectorized for efficient solution on the CYBER 203 computer. A very flexible rezoning routine is used to concentrate mesh points anywhere in the field, either by a user-defined weighting function or by allowing high gradient regions to adjust the grid. The grid adjustment routine is implicit in nature and represents a very small portion of the total computational cost. Currently, the code runs in approximately 1.6×10^{-5} ≈ 0.00001 seconds per grid point per iteration.

INTRODUCTION

The finite-volume method of numerically solving systems of conservation laws has been successfully applied to a wide variety of problems in fluid mechanics.[1-6] Its ability to maintain conservation of mass, momentum, and energy from cell to cell, even in rather complex nonorthogonal coordinates, makes it particularly attractive for use with adaptive grid techniques. For the purposes of this paper, a finite-volume formulation (FVF) is defined as a discrete approximation to a conservation law written in integral form which (1) uniquely defines control volumes in such a way that control volumes (cells) do not overlap nor are gaps left in physical space and (2) uniquely defines fluxes and forces acting through cell walls so that summability without residue[7] (conservation) is guaranteed. It differs from a finite difference formulation (FDF) only in the way a problem is approached. For example, given a system of conservation laws we might consider the FDF as a discrete approximation to the

differential form of the laws while the FVF is a discrete approximation to the integral form of the laws. A review of various FVF's suggests that all FVF's can be written as FDF's (and usually appear that way in the literature) but not all FDF's can be written as FVF's. Furthermore, discretizing the divergence form[8] of the conservation laws in general coordinates to obtain the FDF does not guarantee that the formulation can be expressed as an FVF because conservation is not maintained due to some confusion of how to properly define the metric coefficients. Various ways of defining the metric coefficients to overcome this problem have been given in References 5 and 9. The discussion there leads one to conclude that while there may be a certain amount of ambiguity in how to properly define or discretize the metrics the problems can be overcome without a great deal of difficulty. It should also be noted that the FDF of a conservation law written in nondivergence form can in fact be a conservative FVF. An example of such a scheme is presented in Reference 10.

The discussion up to this point has been concerned with taking a differential form of a conservation law, approximating it by an FDF, and testing if it satisfies the requirements of an FVF (or forcing it to satisfy the requirements by suitably defining the metrics). The approach taken in the presentation which follows is to start with the integral form of the conservation law, approximate it with an FVF on some generalized grid, and expand the formulation to see what FDF results. It will be shown that no special treatment of metrics is required and in fact no terms which are readily recognizable as metric coefficients ever appear. It is only when the FVF is expanded into a format more familiar as an FDF do these metrics appear as ratios of the dimensions of cell walls to cell volumes. Furthermore, the unexpanded form of the FVF is ideally suited for vectorization.

Within this framework a grid adaption routine has been developed which greatly facilitates placement of mesh points where needed. A description of how the adaption algorithm has evolved from the groundwork established in references 11 and 12 to the current implicit adaption scheme will be presented. Finally, sample calculations of supersonic flow over a Viking Aeroshell, including flow in the wake, demonstrate the versatility of both the FVF and the adaption algorithm through its ability to concentrate points not only in the boundary layer but also in the free shear layer.

FINITE-VOLUME FORMULATION

Consider a system of two-dimensional conservation laws which can be expressed by the vector relation

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{1a}$$

$$\iiint \frac{\partial U}{\partial t}\, dv + \iint (F\bar{i} + G\bar{j}) \cdot d\bar{s} = 0 \tag{1b}$$

where Equations (1a) and (1b) are the differential and integral forms of the conservation laws respectively, and $\bar{i}$ and $\bar{j}$ are unit vectors in the x and y directions, respectively.

A control volume in two-dimensional (or axisymmetric) space is determined by a quadrilateral whose vertices are defined by adjacent mesh points. The $[i,j]$ cell refers to the cell with vertices (i,j), $(i+1,j)$, $(i,j+1)$, $(i+1,j+1)$. Points of constant j index define ξ coordinate lines, increasing ξ in the direction of increasing i. Points of constant i index define η coordinate lines, increasing η in the direction of increasing j (see Fig. 1). The flux through the wall defined by the points $<(i,j), (i,j+1)>$ is calculated using information from the (i,j) vertex on the predictor step and from the $(i,j+1)$ vertex on the corrector step for odd time steps. The order is reversed for even time steps. The flux through the wall defined by the points $<(i,j), (i+1,j)>$ follows a similar pattern of definition, using information from the (i,j) vertex on the predictor step. The FVF for the $[i,j]$ cell is written:

$$\left.\frac{\partial U}{\partial t}\right|_{i,j}^{\overline{n+1}} A_{i,j} + F^{n}_{i+1,j}(y_{i+1,j+1} - y_{i+1,j}) - G^{n}_{i+1,j}(x_{i+1,j+1} - x_{i+1,j})$$

$$- F^{n}_{i,j}(y_{i,j+1} - y_{i,j}) + G^{n}_{i,j}(x_{i,j+1} - x_{i,j})$$

$$- F^{n}_{i,j+1}(y_{i+1,j+1} - y_{i,j+1}) + G^{n}_{i,j+1}(x_{i+1,j+1} - x_{i,j+1})$$

$$+ F^{n}_{i,j}(y_{i+1,j} - y_{i,j}) - G^{n}_{i,j}(x_{i+1,j} - x_{i,j}) = 0 \tag{2a}$$

$$\left.\frac{\partial U}{\partial t}\right|_{i,j}^{\overline{n+1}} A_{i,j} + \overline{F^{n+1}_{i+1,j+1}}(y_{i+1,j+1} - y_{i+1,j}) - \overline{G^{n+1}_{i+1,j+1}}(x_{i+1,j+1} - x_{i+1,j})$$
$$- \overline{F^{n+1}_{i,j+1}}(y_{i,j+1} - y_{i,j}) + \overline{G^{n+1}_{i,j+1}}(x_{i,j+1} - x_{i,j})$$
$$- \overline{F^{n+1}_{i+1,j+1}}(y_{i+1,j+1} - y_{i,j+1}) + \overline{G^{n+1}_{i+1,j+1}}(x_{i+1,j+1} - x_{i,j+1})$$
$$+ \overline{F^{n+1}_{i+1,j}}(y_{i+1,j} - y_{i,j}) - \overline{G^{n+1}_{i+1,j}}(x_{i+1,j} - x_{i,j}) = 0 \qquad (2b)$$

$$U^{n+1}_{i,j} = .5\left(\overline{U^{n+1}_{i,j}} + \overline{\overline{U^{n+1}_{i,j}}}\right) \qquad (3)$$

where A_{ij} is the area of the $[i,j]$ cell. Equation (2) must approximate the integral form of Equation (1). The first term of Equation (2) must approximate the integral of $\partial U/\partial t$ over the entire cell. In the FVF we set

$$\left.\frac{\partial U}{\partial t}\right|_{i,j}^{\overline{n+1}} = \frac{\overline{U^{n+1}_{i,j}} - U^{n}_{i,j}}{\Delta t_{ij}} \qquad (4a)$$

$$\left.\frac{\partial U}{\partial t}\right|_{i,j}^{\overline{n+1}} = \frac{\overline{U^{n+1}_{i+1,j+1}} - U^{n}_{i+1,j+1}}{\Delta t_{i+1,j+1}} \qquad (4b)$$

Note the use of a Δt which varies from point to point in Equation (4).

Now consider a finite-difference formulation, FDF, of Equation (1) which can be transformed to

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial F}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial G}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial G}{\partial \eta}\frac{\partial \eta}{\partial y} = 0 \qquad (5)$$

where $x = x(\xi,\eta)$, $y = y(\xi,\eta)$. The FDF of (5) is written:

$$\overline{U^{n+1}_{i,j}} = U^{n}_{i,j} - \Delta t_{i,j}\left[\frac{(F_{i+1,j} - F_{i,j})^{n}}{\Delta\xi}\,\xi^{+}_{x_{ij}} + \frac{(G_{i+1,j} - G_{i,j})^{n}}{\Delta\xi}\,\xi^{+}_{y_{ij}} + \frac{(F_{i,j+1} - F_{i,j})^{n}}{\Delta\eta}\,\eta^{+}_{x_{ij}} + \frac{(G_{i,j+1} - G_{i,j})^{n}}{\Delta\eta}\,\eta^{+}_{y_{ij}}\right] \tag{6a}$$

$$\overline{\overline{U^{n+1}_{i,j}}} = U^{n}_{i,j} - \Delta t_{i,j}\left[\frac{(F_{i,j} - F_{i-1,j})^{\overline{n+1}}}{\Delta\xi}\,\xi^{-}_{x_{ij}} + \frac{(G_{i,j} - G_{i-1,j})^{\overline{n+1}}}{\Delta\xi}\,\xi^{-}_{y_{ij}} + \frac{(F_{i,j} - F_{i,j-1})^{\overline{n+1}}}{\Delta\eta}\,\eta^{-}_{x_{ij}} + \frac{(G_{i,j} - G_{i,j-1})^{\overline{n+1}}}{\Delta\eta}\,\eta^{-}_{y_{ij}}\right] \tag{6b}$$

$$U^{n+1}_{i,j} = \frac{1}{2}\left(\overline{U^{n+1}_{i,j}} + \overline{\overline{U^{n+1}_{i,j}}}\right) \tag{6c}$$

The FVF (Eqs. (2) and (4)) and the FDF (Eq. (6)) are equivalent if

$$\xi^{+}_{x_{ij}} \equiv (y_{i+1,j+1} - y_{i+1,j})\,\Delta\xi/A_{ij} \tag{7a}$$

$$\eta^{+}_{x_{ij}} \equiv (y_{i,j+1} - y_{i+1,j+1})\,\Delta\eta/A_{ij} \tag{7b}$$

$$\xi^{+}_{y_{ij}} \equiv (x_{i+1,j} - x_{i+1,j+1})\,\Delta\xi/A_{ij} \tag{7c}$$

$$\eta^{+}_{y_{ij}} \equiv (x_{i+1,j+1} - x_{i,j+1})\,\Delta\eta/A_{ij} \tag{7d}$$

$$\xi^{-}_{x_{ij}} \equiv (y_{i-1,j} - y_{i-1,j-1})\,\Delta\xi/A_{i-1,j-1} \tag{7e}$$

$$\eta^{-}_{x_{ij}} \equiv (y_{i-1,j-1} - y_{i,j-1})\,\Delta\eta/A_{i-1,j-1} \tag{7f}$$

$$\xi^{-}_{y_{ij}} \equiv (x_{i-1,j-1} - x_{i-1,j})\ \Delta\xi/A_{i-1,j-1} \tag{7g}$$

$$\eta^{-}_{y_{ij}} \equiv (x_{i,j-1} - x_{i-1,j-1})\ \Delta\eta/A_{i-1,j-1} \tag{7h}$$

A linearized analysis of the FDF (Eq. (6)), after it has been expanded into a one-step scheme shows the following: (1) Equation (6) is a consistent representation of Equation (5), (2) the truncation error for variable Δt_{ij} is first order in time and can be expressed

$$E_1 = \frac{-\Delta t_{ij}}{2}\left(\frac{\partial h_{ij}}{\partial \xi} C + \frac{\partial h_{ij}}{\partial \eta} D\right)\left(C \frac{\partial U}{\partial \xi} + D \frac{\partial U}{\partial \eta}\right) \tag{8}$$

where

$$C = \frac{\partial F}{\partial U} \xi_x + \frac{\partial G}{\partial U} \xi_y,\ D = \frac{\partial F}{\partial U} \eta_x + \frac{\partial G}{\partial U} \eta_y$$

and

$$h_{ij}(\xi,\eta) = \Delta t(\xi,\eta)/\Delta t(\xi_{ij},\eta_{ij})$$

Note that $E_1 = 0$ if Δt_{ij} = constant, and (3) the spatial truncation error is of $0(\Delta^2)$ where Δ is proportional to a cell wall dimension if all $\xi^{\pm}_x$, etc., are first-order-accurate approximations to ξ_x, etc., and $(\xi^{+}_x + \xi^{-}_x)/2$, etc., , are second-order-accurate approximations to ξ_x, etc. These conditions are formally satisfied by Equation (7). For example

$$(\xi^{+}_x + \xi^{-}_x)/2 = \xi_x + \Delta\xi\{\Delta\xi\ [(Jy_{\xi\eta})_\xi + Jy_{\xi\xi\eta}] + \Delta\eta(Jy_{\xi\eta})_\eta\}/2 \tag{9}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$.

Consequently, it is seen that while the FDF or FVF is second-order-accurate (for Δt_{ij} = constant) excessive stretching or skewness of the grid can introduce large factors of second-order terms. Furthermore, these factors multiply terms like $\partial^2 U/\partial\xi^2$, etc., and they contribute to the overall rtificial viscosity of the method. At present, grid-induced errors are

checked by comparing results obtained for the same problem on the different grids constructed from more points or from different adjusting parameters (see section on ADAPTIVE GRID). It should also be noted that while Equation (5) is not in strict conservation form, the FVF which models Equation (1b) is conservative in the strict sense of summation without residue.[7] For example, regardless of the grid used, the FVF returns an exact uniform flow if a uniform flow is used as an initial condition.

It is noted that the equations and analysis in this section are for two-dimensional flow. An axisymmetric extension introduces only minor complications. Finally we point out that the FVF (Eqs. (2) and (4)) reduces to MacCormack's method[13] in Cartesian coordinates. There are an infinity of other ways to define cells and flux through cells on a general grid, each of which lead to a unique FVF and unique definitions of metric coefficients for that particular FVF.

BOUNDARY CONDITIONS

Only a cursory description of boundary conditions can be provided due to space limitations.

Wall:	$u = v = 0$. Adiabatic wall or constant wall temperature. $\partial p/\partial n = 0$. ($n$ coordinate normal to wall)
Shock:	Rankine-Hugoniot relations for a discrete moving shock are applied. Pressure behind shock obtained from extrapolation from interior points.
Symmetry:	Limiting form of differential form of governing equations solved using MacCormack's method.
Outflow:	Primitive variables obtained by extrapolation.

ADAPTIVE GRID

The present grid adjustment scheme has evolved from one which imparted velocities to grid points based on gradients in the field as in References 11 and 12 to one which rezones the computational mesh using an implicit algorithm at any desired frequency (i.e., once per time step or once per 1000 time steps). Originally every point (i,j) in the computational plane was connected to the four adjacent points (i±1,j), (i,j±1) by springs whose spring constants K^j_i, K^i_j were determined by a function of the gradient of some dependent variable between the points. For example, the spring along the ith row connecting points (i,j) and (i,j+1) was defined

$$K_j^i = 1 + c\left|f_{i,j} - f_{i,j+1}\right|/\Delta\bar{r} \tag{10}$$

where c is a constant and $\Delta\bar{r}$ is the distance between the points. Grid points on the boundaries were free to move along the boundary as if on a frictionless rod. Grid points on the corners of the domain were held fixed. Small adjustments in the x and y directions were assigned to grid points as determined by the net forces on the grid points in the respective directions. After every time step, the grid was updated and checked to make sure that no grid overlap was imminent.

In a noninteractive test case where $p(x,y)$ was prescribed to model the pressure field of an oblique shock crossing a uniform flowfield ($p(t) \equiv 0$), the grid evolved from one of equally spaced rectangular cells to the one shown in Figure 2. In an interactive test case where the solution $p(x,y,t)$ was allowed to evolve with time, the final grid distribution (Fig. 3) is seen to be much more erratic than that of the previous case. The pressure field for this case is shown in Figure 4.

By sacrificing the column to column (or row to row) influence of the spring system (i.e., no springs between column i and column i+1) a superior algorithm in terms of computational costs and smoothness of grid distribution can be constructed. (Smoothness of grid distribution is a subjective judgment. A coordinate line whose direction or length changes erratically from point to point is judged nonsmooth.)

Consider the coordinate line of constant index i as shown in Figure 5. Let s_j be the length of the line in physical space from the first point to the jth point as defined below.

$$s_1 = 0;\ s_j = \sum_{l=2}^{l=j>1} \left[(x_l - x_{l-1})^2 + (y_l - y_{l-1})^2\right]^{1/2} \tag{11}$$

Let $f_{j,m}$ be a table of dependent variables at the point s_j defined by

$$f_{j,1} = x_j,\ f_{j,2} = y_j,\ f_{j,3} = \rho_j,\ f_{j,4} = u_j,\ f_{j,5} = v_j,\ f_{j,6} = e_j \tag{12}$$

The grid points in physical (x,y) space are mapped onto a straight line in s space. Let each grid point be connected to its neighbor in s space by a spring with spring constant K_j between the j and $j+1$ points to be defined later.

Let Δn_j be the distance between the j and $j+1$ points in s space. Then $K_j \Delta n_j$ equals a constant. Therefore,

$$\Delta n_j = K_1 \Delta n_1 / K_j \tag{13}$$

The known total length of s can be expressed as

$$S_{JN+1} = \sum_{j=1}^{JN} \Delta n_j = \Delta n_1 K_1 \sum_{j=1}^{JN} 1/K_j = \Delta n_1 K_1 \text{ SUM} \tag{14}$$

where JN = total number of intervals. Consequently,

$$\Delta n_1 = S_{JN+1}/(K_1 \text{ SUM})$$
$$\Delta n_j = S_{JN+1}/(K_j \text{ SUM}) \tag{15}$$

The constants K_j can be defined explicitly or implicitly. An explicit definition of K_j assigns a value to the spring constant independent of the position of the spring in s space. For example,

$$K_j = \exp(-c_1 \cdot j/JN) \tag{16}$$

provides an exponentially increasing spacing of grid points from $j = 1$ to $j = JN$ where the constant c_1 is used to control stretching. This definition provides no adaptive capability but does provide a very quick way of concentrating points near boundaries.

An implicit definition of K_j assigns a value to the spring constant which is a function of the position of the spring in s space. For an implicit case, the algorithm is implemented as follows:

(1) Start at the first column (or row) of data and compute and store all the values of s_j and $f_{j,m}$.

(2) Initialize $\bar{n}_j$ and n_j equal to s_j for all j, ITER = 0.

(3) Set ITER = ITER + 1.

(4) Using the known stored values of independent variable s_j and dependent variables $f_{j,m}$, compute the new values of the dependent variables $d_{j,m}$ at all n_j with a univariate interpolation routine where $d_{j,1} = x_j$, $d_{j,2} = y_j$, etc.

(5) Compute $K_j = K_j(d_{j,m})$.

6) Compute the new coordinates n_j from Equation (15); set $n_{JN+1} = s_{JN+1}$ to keep end point values unchanged.

(7) Compute an error norm to determine convergence

$$L_2 = \sum_{j=2}^{JN} (n_j - \bar{n}_j)^2 \quad ^{1/2} \tag{17}$$

If $L_2 < 0.0001$, proceed to step (9).

If ITER > 100, stop the procedure.

(8) Set $\bar{n}_j = n_j$ for all j and go to step (3).

(9) Replace x_j, y_j, ρ_j, etc., with $d_{j,1}$, $d_{j,2}$, $d_{j,3}$, etc., go to next column of data and repeat all steps.

Typical definitions of K_j employ gradients of velocity, internal energy, or Mach number. For example,

$$K_j = 1 + c_3 |e(n_{j+1}) - e(n_j)| / (n_{j+1} - n_j) \tag{18}$$

where c_3 can be used to control the concentration of mesh points in regions of high gradient. Examples of grid distribution using these gradient adaptive mechanisms will be presented in the RESULTS section.

In cases where a high gradient develops rapidly, as in the vicinity of an expansion corner of a planetary probe, the adaption algorithm can have trouble converging. This problem is overcome as follows.

(1) After step (4) in the adaption algorithm, the overall change in the new grid distribution n_j is damped by writing

$$n_j = c_4 n_j + (1 - c_4)\bar{n}_j \tag{19}$$

where $0 < c_4 \leq 1$ (c_4 is typically of order 1/2)

(2) After step (5) in the adaption algorithm, the values of the spring constants are filtered by writing

$$K_j^{n+1} = (K_{j-1} + 2K_j + K_{j+1})^n / 4; \quad j = 2, JN-1 \tag{20}$$

Often the convergence problems can be overcome by keeping the rezoning interval small or by avoiding large changes of c_3 in Equation (18) between rezonings.

The major advantage of the rezoning procedure described herein is that it provides a very quick way of providing an adaptive capability to the

finite-volume formulation. The option to control rezoning frequency allows the grid to dynamically adapt to developing features of the flow (i.e., shock, shear layers) and then to be held fixed as steady state is approached. The rezoning procedure can also be used as a separate program to study the effects of monitoring various gradients in the field or using different definitions of K_j.

RESULTS AND DISCUSSION

The finite-volume, adaptive grid algorithm described in the previous sections has been used to compute supersonic flowfields over spheres, cylinders and two planetary probe configurations. The most comprehensive tests to date for testing the adaption algorithm have been on the Viking Aeroshell (Fig. 6), a configuration which protected the Viking lander for its descent through the Martian atmosphere. This section will deal exclusively with those results. The freestream conditions, $M_\infty = 2$, $\gamma = 1.285$, $Re_\infty = 5000$, have been chosen to permit comparison with the results of Reference 14. Subsequent calculations at higher Reynolds numbers have been generated to demonstrate the capabilities of the adaption algorithm.

The grid shown in Figure 6 was achieved with an explicit definition of K_j (Eq. (16)) with $c_1/JN = 0.15$, 90 points around the body and down the wake centerline and 31 points between the body/wake centerline and the bow shock. The pressure distribution around the body/wake centerline is presented in Figure 7. This case was run to convergence in approximately 35 minutes of computing time with a timing of $\approx 3 \times 10^{-5}$ sec/iteration/grid point. Recent changes in the code affecting the manner in which data are stored have decreased timing to $\approx 1.6 * 10^{-5}$ sec/iteration/grid point.

An adaptive grid calculation which monitored velocity gradients was applied to this same problem. In the previous case, the grid was not sufficiently stretched to adequately resolve the boundary layer at the separation point. Since the separation region is very important in determining the nature of the flow in the wake,[15] it was decided to run this case with 91 points across the shock layer. The converged grid for $c_3 = 0.5$ is shown in Figure 8. In defining the values of K_j near the wall, a test was included that would make $K_j = 3$, $j \leq 4$ if K_j was previously less than 3. From $j = 5$ to 9, the minimum allowable value of K_j was decreased linearly to zero. This forces a concentration of points near the body and in the wake. Pressure distribution is plotted in Figure 7. Comparisons with numerical results of

Reference 14 for shock shape and pressure distribution down the wake centerline show generally good agreement in Figures 9 and 7, respectively.

Solutions for the higher Reynolds number cases ($5 * 10^4$, $5 * 10^6$) were generated sequentially using the previous results as initial conditions. All of these results are for laminar flow and so while the highest ones are physically in error they test the adaption routine's ability to resolve the large gradients associated with such flows.

Some interesting results of these high Reynolds number studies are presented in Figures 10 through 17. The converged grid showing a concentration of points in the free shear layer is presented in Figure 10. The streamline pattern around the expansion corner of the vehicle and associated Mach number contours are presented in Figures 11 and 12. The outer extent of these figures is defined by the physical location of the 25th mesh point. The velocity and grid point distributions along the η coordinate line (i=22) located just ahead of the expansion corner are given in Figure 13 for $Re_\infty = 10^4$ and 10^6. These results were obtained using twenty passes through Equation (20) with $c_3 = 0.5$ and 1.0 respectively. The global internal energy contour plot in Figure 14 shows the high gradient regions in the shear layer behind the expansion corner, the captured recompression shock in the wake and a small, high gradient shock-like region on the symmetry line just behind the base where recirculating velocities rapidly change from supersonic to subsonic. This phenomenon was also observed in the results of Reference 14 using a different numerical approach. The high gradient free shear layer extends approximately one-fourth maximum body radius beyond the corner after which point the gradients rapidly decrease and the flow turns toward the axis. There is a large region just below the shear layer of low density ($.2 \leq \rho/\rho_\infty \leq .7$) recirculating flow. A Mach number distribution along an η coordinate line (i=34), Figure 15, shows the high gradient region and distribution of mesh points through the shear layer. The complexity of the flow in this region and the ability of the adaption process to concentrate mesh points in the high gradient regions away from the wall further demonstrate the versatility and sensitivity of the adaption process.

CONCLUDING REMARKS

The Finite-Volume Formulation (FVF) described herein has been shown to be a consistent and conservative approximation to the Navier-Stokes equations. The formulation has a first order error in wave speed, proportional to the local value of Δt, when the local maximum time step is used to advance the

solution at every point. The formulation is second order accurate when constant time increment is used to advance the solution. However, at large Reynolds numbers (small cell volumes) the constant Δt advancement of the solution, using one value of Δt that is stable for all points, is impractical because many more iterations are required to converge the solution. Converged solutions for complete blunt body flows using approximately 3000 mesh points can be obtained within 30 minutes using the variable Δt option and adaptive grid rezoning on the CYBER 203 computer.

In all of the problems considered herein adaption along only one coordinate line is quite sufficient for the purpose of moving mesh points to high gradient regions in the flow. Restricting grid motion along one coordinate direction permits application of a highly efficient implicit grid adaption procedure. The routine can be implemented at any desired frequency thus permitting dynamic adaption or cost savings by adapting only after large increments of iteration count. Grid point concentration in the boundary layer and free shear layer have demonstrated the success and versatility of the adaption algorithm.

REFERENCES

1. Rizzi, Arthur W., Inouye, Mamoru (1973) A Time Split Finite-Volume Technique for Three-Dimensional Blunt Body Flow, AIAA Paper No. 73-133.
2. MacCormack, R. W. and Paullay, A. J. (1974) The Influence of the Computational Mesh on Accuracy for Initial Value Problems with Discontinuous or Nonunique Solutions, Computers and Fluids, Vol. 2, pp. 339-361.
3. Thomas, P. D. and Lombard, C. K. (1978) The Geometric Conservation Law - A Link Between Finite-Difference and Finite-Volume Methods of Flow Computation on Moving Grids, AIAA Paper No. 78-1208.
4. Caughey, D. A. and Jameson, A. (1980) Progress in Finite Volume - Calculations for Wing-Fuselage Combinations, AIAA Journal, Vol. 18, No. 11.
5. Lombard, C. K., Davy, W. C. and Green, M. J. (1980) Forebody and Base Region Real Gas Flow in Severe Planetary Entry by a Factored Implicit Numerical Method - Part I (Computational Fluid Dynamics), AIAA Paper No. 80-0065.
6. Allen, J. S. and Cheng, S. I. (1970) Numerical Solutions of the Compressible Navier-Stokes Equations for the Laminar Near Wake, Physics of Fluids, Vol. 13, No. 1, pp. 37-52.
7. Cheng, Sin I. (1974) A Critical Review of the Numerical Solution of Navier-Stokes Equations, Lecture Notes in Physics, No. 41, Springer Verlag, 1974
8. Roache, Patrick J. (1972) Computational Fluid Dynamics, Hermosa Publishers, Albuquerque, New Mexico.
9. Pulliam, T. H. and Steger, J. L. (1978) An Implicit Finite-Difference Simulations of Three-Dimensional Flow, AIAA Paper 78-10.
10. Shubin, Gregory and Cheng, Sin I. (1979) Gas Dynamic Modeling and Computational Accuracy, Journal of Computational Physics, Vol. 32.
11. Dwyer, H. A., Kee, R. J. and Sanders, B. R. (1979) An Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer, AIAA Paper No. 79-1464.

12. Rai, M. M. and Anderson, D. A. (1981) The Use of Adaptive Grids in Conjunction with Shock Capturing Methods, AIAA Paper 81-1012.
13. MacCormack, R. W. (1969) The Effect of Viscosity in Hypervelocity Impact Cratering, AIAA Paper 69-354.
14. Gnoffo, Peter A. (1980) Complete Supersonic Flowfields Over Blunt Bodies in a Generalized Orthogonal Coordinate System, NASA TM 81784.
15. Berger, Stanley A. (1971) Laminar Wakes, American Elsevier Publishing Company, Inc., New York.

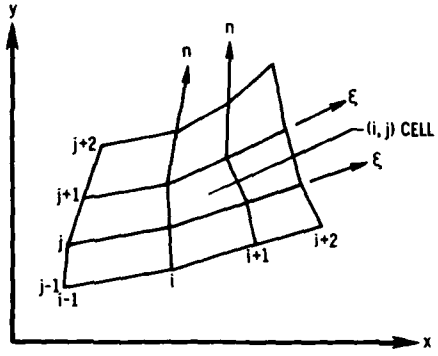


Fig. 1 Schematic showing relation of indexing system, physical (x,y) coordinates, and computational (ξ,η) coordinates.

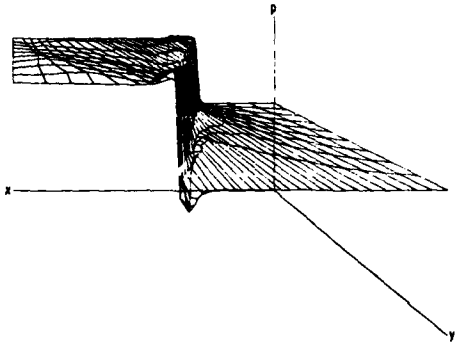


Fig. 4 Perspective view of pressure on (x,y) plane for oblique shock calculation on interactive grid.

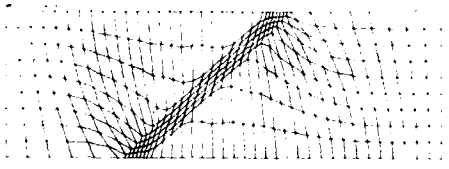


Fig. 2 Noninteractive example of grid adapting to prescribed pressure distribution with two degrees of adaptive freedom.

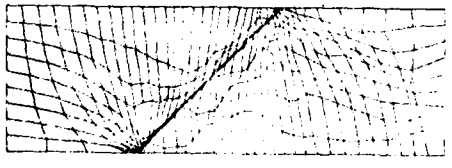


Fig. 3 Interactive example of grid adapting to computed pressure distribution across oblique shock with two degrees of adaptive freedom.

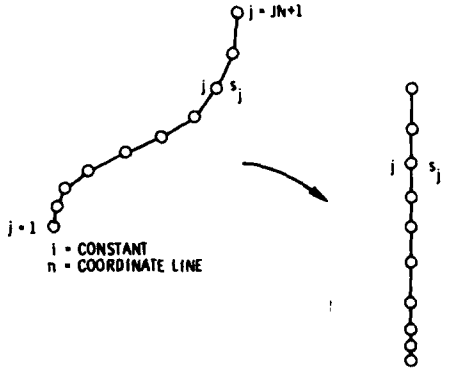


Fig. 5 Schematic showing spring system for one degree of adaptive freedom along η coordinate line.

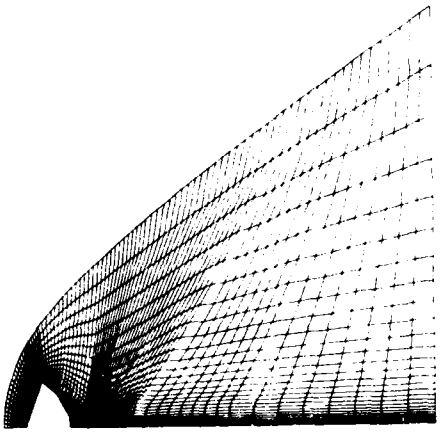


Fig. 6 Grid over Viking Aeroshell obtained with explicit definition of K_j, $C_1/JN=.15$.

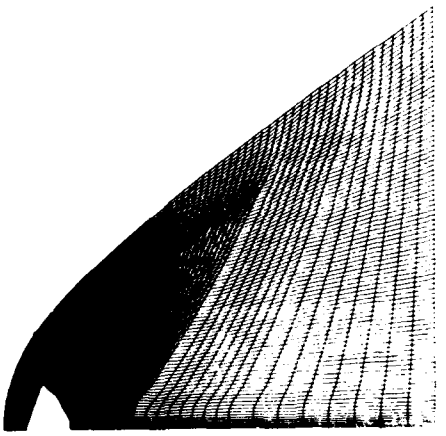


Fig. 8 Grid over Viking Aeroshell obtained with implicit definition of K_j, $Re_\infty=5000$.

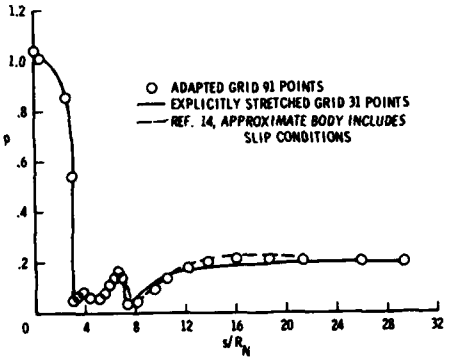


Fig. 7 Pressure distribution around body/wake centerline of Viking Aeroshell. $M_\infty=2$, $Re_\infty=5000$, $\gamma=1.285$.

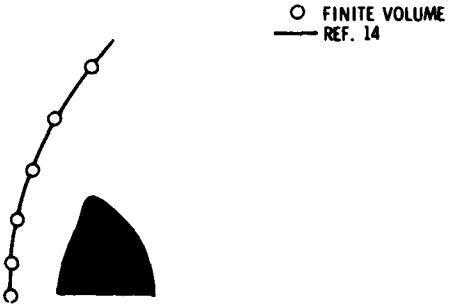


Fig. 9 Bow shock over Viking Aeroshell for $M_\infty=2$, $Re_\infty=5000$.

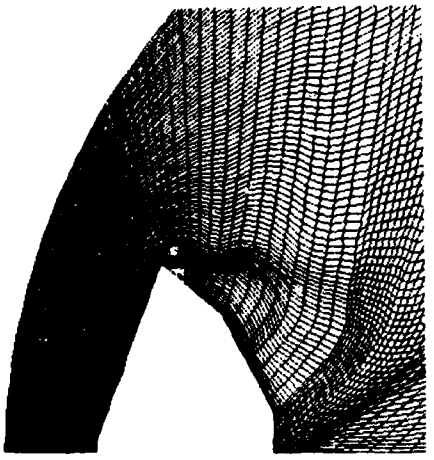


Fig. 10 Grid over Viking Aeroshell obtained with implicit definition of K_j, $Re_\infty=10^6$.

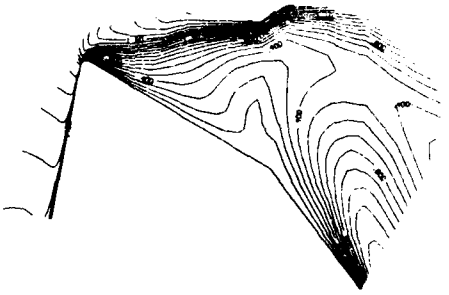


Fig. 12 Mach number contours around expansion corner of Viking Aeroshell, $M_\infty=2$, $Re_\infty=10^6$.

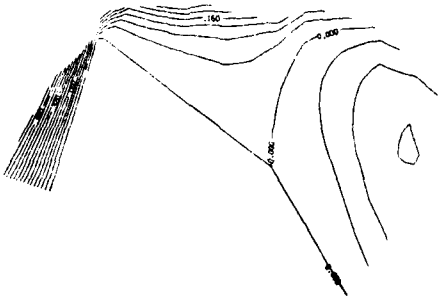


Fig. 11 Streamlines around expansion corner of Viking Aeroshell, $M_\infty=2$, $Re_\infty=10^6$.

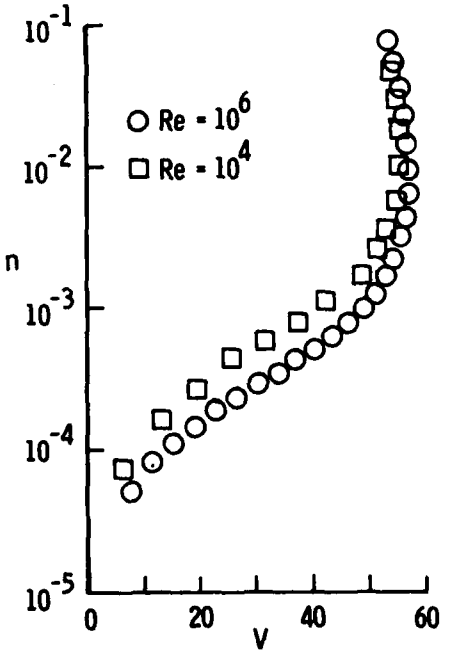


Fig. 13 Velocity and grid spacing distribution along η coordinate line ahead of expansion corner, $Re_\infty=10^4$ and 10^6.

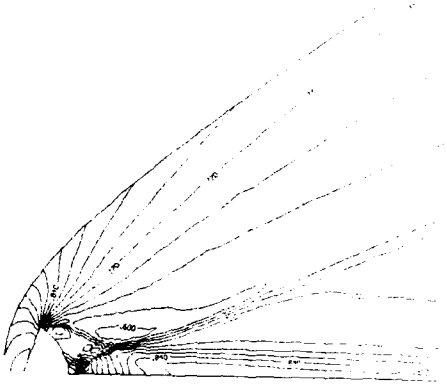


Fig. 14 Contour plot of internal energy over Viking Aeroshell for $M_\infty=2$, $Re_\infty=10^6$.

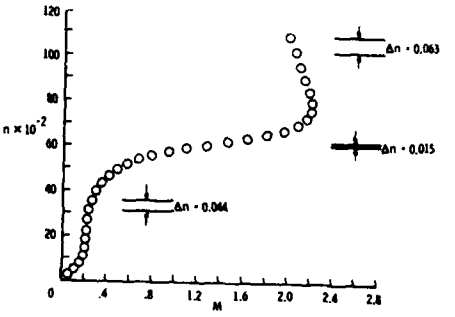


Fig. 15 Mach number and grid spacing distribution through separated shear layer behind expansion corner along η coordinate line.

AD P001006

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

IDEALIZED DYNAMIC GRID COMPUTATION OF PHYSICAL SYSTEMS

JOSHUA C. ANYIWO
NASA Langley Research Center
Hampton, VA 23665 USA

INTRODUCTION

This paper considers the construction and utility of an idealized computational space (henceforth called tau space) for finite-difference computation of physical systems.

The tau space idealization is with respect to the following features: (a) the "physical system-adaptive" grid discretizing tau space is to be always uniform and orthogonal; (b) the transformed equations, initial and boundary conditions are to be not more complicated than the subject physical problem space; and (c) the transformation relations between tau space and physical space are to be reasonably simple and easy to implement in conjunction with desired finite-difference schemes.

The construction of such an idealized computational space requires: (a) the ability to discretize a physical problem space with an orthogonal grid which is everywhere and always adaptive to the collective influences of the geometry, the physics and the number of grid cells (or discretization scale) of the subject physical system; and (b) the ability to formulate relatively simple transformation relations between such a "physical system-adaptive" grid and a prototype uniform and orthogonal grid.

One method of approach is to first construct a reasonably simple measure field which is independent of reference frame and which unifies the interactive influences of the geometry and physics of a system and the scale of the grid discretizing that system. Once such an invariant, unified measure field is specified, the dynamic (or moving adaptive) grid for a physical system in a reference frame would logically be defined as the grid in that reference frame which instantaneously equidistributes the subject system's invariant, unified measure field. Furthermore, a computational space, idealized in the aforementioned sense, may then be constructed by basing the metric of the system's problem space on the instantaneous values of that system's invariant, unified measure field.

This paper successfully implements the above approach and also outlines a reasonably simple procedure, based upon tau space transformation, for efficient and accurate finite-difference computation of physical systems.

PREVIOUS PAGE IS BLANK

A number of specialized grid methods for finite-difference computation of physical systems have been proposed in the past; for example, References 1 and 2. Most of them, however, employ static grid discretization with or without adaptation to the system's physics. A few, such as References 3 to 6 have incorporated some sort of dynamic grid adaptation. But none of these methods has been developed to explicitly and efficiently incorporate the interactive influences of scale, geometry and physics in the computation of systems. The ideas presented in this paper not only fill the above needs but also open the way for further developments towards efficient and accurate physical system solvers.

STRAIN FIELD MODEL OF PHYSICAL SYSTEMS

Consider a physical system to be always covered by a uniform, regular, orthogonal base coordinate system $X\{x^i,t\}$, which may be chosen arbitrarily to properly describe the system. Further, let the physical system also always be referred to an orthogonal, curvilinear coordinate frame $S\{s^i,t\}$, which is always fitted to the system's boundaries. The reference frame $S\{s^i,t\}$, then represents the physical system's geometry, and the distributions $T_k(s^i,t)$, $k \geq 1$, $i \geq 1$, of the system's material properties T_k, relative to $S\{s^i,t\}$ characterize the system's physics.

It is desired to construct a reasonably simple, invariant measure field which unifies the interactive influences of the geometry, physics and discretization scale of a finitely discretized physical system space. The procedure used in this paper is sketched in Table 1 and described below.

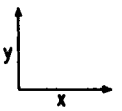
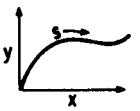
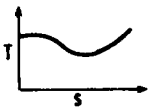
First a common basis must be established with which to unify the scale, geometry and physics of a physical system.

Consider a finite discretization of a physical system. It may be perceived as a process of deforming the base coordinate system $X\{x^i,t\}$ relative to some undeformed absolute frame A. As the number N, of discretizing grid cells tends to infinity one recovers the undeformed absolute frame. But, since fewer than three grid cell nodes may not be used to meaningfully discretize a space, the base coordinate system X, may be perceived as becoming infinitely deformed relative to A as $N \to 2$. Thus, the discretization scale of a finitely discretized physical system may be specified in terms of a deformation moment μ_A, of the base coordinate system covering the physical system's space. And μ_A may be defined as an inverse function of $(N - 1)$.

The geometry of a physical system may be uniquely specified from knowledge of the system's boundaries and the distribution of the principal geometric

TABLE 1

A STRAIN FIELD MODEL OF PHYSICAL SYSTEMS

PHYSICAL SYSTEM	BASE GEOMETRY	REFERENCE GEOMETRY	PHYSICS
REPRESENTATION			
SLOPE (OR SHEAR), θ	$\theta_A = 0$	$\theta_G = y_x$	$\theta_T = T_s$
CIRCULAR CURVATURE, μ	$\mu_A = 1/N$	$\mu_G = y_{xx}/\left(1+y_x^2\right)^{3/2}$	$\mu_T = T_{ss}/\left(1+T_s^2\right)^{3/2}$
TORSION, β	$\beta_A = 0$	β_G: DEFINED IN TEXT	β_T: NEGLECTED
TOTAL CURVATURE $(\mu + \beta)$	$(\mu_A + \beta_A)$	$(\mu_G + \beta_G)$	$(\mu_T + \beta_T)$
DISCRETIZATION SCALE, ϕ	$\phi = 1/\ln(N-1)$		
DEFORMATION FIELDS, σ	$\sigma_X(A) \equiv \kappa_1 \mu_A \phi$	$\sigma_S(X) \equiv \kappa_2 \{\overline{\mu_G + \beta_G}\} \phi$	$\sigma_T(S) \equiv \kappa_3 \{\overline{\lvert\theta_T\rvert + \lvert(\mu_T + \beta_T)\rvert}\} \phi$
ABSOLUTE DEFORMATION FIELD, γ	$\gamma = \{\sigma_X(A) + \sigma_S(X) + \sigma_T(S)\}$		
ABSOLUTE STRAIN DENSITY FIELD	$\Gamma = e^{\gamma}$		

deformation moment--total curvature $(\mu_G + \beta_G)$--of the system's reference frame $S\{s^i,t\}$, relative to $X\{x^i,t\}$.

Finally, the physics of a physical system may be uniquely specified from knowledge of the boundary conditions of the system's material properties and the distribution of the principal material deformation moments--slope (or shear) θ_T, and total curvature $(\mu_T + \beta_T)$--of the system's material properties relative to the reference frame $S\{s^i,t\}$.

Thus, the scale, geometry and physics of a physical system each represents a deformation field which is uniquely specified by some set of principal deformation moments. Grid adaptation becomes meaningless as the number N, of discretizing grid cells tends to infinity, since all grids would tend to the same size. Therefore, the discretization scale influence must be made to permeate both the geometric and the material deformation moments, too.

Let the magnitude of a scale, geometric or material deformation be presumed to be invariant with respect to all reference frames. The unified influence of the scale, geometry and physics of a system may, therefore, be represented by a "resultant absolute deformation field," whose total deformation magnitude remains invariant with respect to all reference frames. This total deformation

field would be the sum of the principal deformation moments of $X\{x^i,t\}$ relative to A, of $S\{s^i,t\}$ relative to $X\{s^i,t\}$, and of $T_k(s^i,t)$ relative to $S\{s^i,t\}$.

Let the scale deformation field be specified by the parameter $\sigma_X(A)$, which is defined by some inverse function of $(N - 1)$, where N is the number of discretizing grid cells. A relation which models this scale influence, but which may not necessarily be the ideal relation is:

$$\sigma_X(A) = \kappa_1/\{N \ln(N - 1)\} \tag{1}$$

Let the geometric deformation field be specified by the invariant parameter $\sigma_S(X)$, which is defined as a function of the invariant total curvature of the reference frame $S\{s^i,t\}$, relative to $X\{x^i,t\}$. This function must be modified by the scale influence; a simple relation is:

$$\sigma_S(X) = \kappa_2(\mu_G + \beta_G)/\ln(N - 1) \tag{2}$$

The sign of the curvature is significant to a unique specification of geometry.

Let the material deformation field be specified by the invariant parameter $\sigma_T(S)$. For the purpose of grid adaptation the direction of the material deformation field is not significant. Therefore, $\sigma_T(S)$ may be defined as the simple sum of the magnitudes of the principal material deformation moments θ_T and $(\mu_T + \beta_T)$. This function must also be modified by the scale influence; thus:

$$\sigma_T(S) = \kappa_3\left\{|\theta_T| + |(\mu_T + \beta_T)|\right\}/\ln(N - 1) \tag{3}$$

If a physical system is characterized by more than one property (or dependent variable) T_k, $k = 1, 2, \ldots, M$, then:

$$\sigma_T(S) = \sum_{k=1}^{M} \sigma_{T_k}(S) \tag{4}$$

The resultant absolute deformation field of a physical system may then be specified by γ where:

$$\gamma = \{\sigma_X(A) + \sigma_S(X) + \sigma_T(S)\} \tag{5}$$

For compatibility each of the above three deformation field components should be normalized, using, for instance, the maximum value.

Let the measure of deformation be referred to as a "strain," and let the strain per unit of a reference frame be called the "strain density" in that reference frame. It follows from the foregoing that a finitely discretized physical system may be modeled in a reference frame S, as a strain density field $\Gamma(S)$; and such a model would uniquely identify the physical system except with respect to the direction of gradients and curvatures of the material properties of the system.

The strain density field $\Gamma(S)$, measuring the invariant total absolute deformation field γ, may be expressed as some function of γ. Desirably, a simple function with features which enhance the utility of the resulting measure should be chosen. The exponential function is simple, smooth, makes the measure field positive definite and, above all, permits the measure field to be split conveniently along desired coordinate directions of a reference frame. Because of these desirable features the exponential function will be employed to express the strain density field of physical systems. Thus:

$$\Gamma(S) = e^{\gamma} \tag{6}$$

For finite-difference computation, the reference frame $S\{s^i,t\}$, of a physical system is usually represented as a finite set of curvilinear coordinate curves, s^i, sequenced along the time coordinate, t. The components of the deformation and the strain density fields along each s^i coordinate curve are then:

$$\sigma_{xi}(A) = \kappa_1 \mu_{A_i} \psi_i = \kappa_1 \psi_i / N_i \tag{7a}$$

$$\sigma_{si}(x^i) = \kappa_2 \{\mu_{G_i} + \beta_{G_i}\}\psi_i \tag{7b}$$

$$\sigma_T(s^i) = \kappa_3 \left\{|\theta_{Ti}| + |(\mu_{Ti} + \beta_{Ti})|\right\}\psi_i \tag{7c}$$

and

$$\Gamma_i(s^i) = \exp\left[\kappa_1 \psi_i / N_i + \kappa_2 \overline{\{\mu_{Gi} + \beta_{Gi}\}}\psi_i + \kappa_3 \overline{\left\{|\theta_{Ti}| + |(\mu_{Ti} + \beta_{Ti})|\right\}}\psi_i\right] \tag{7d}$$

where overbars denote appropriately normalized quantities; and $\psi_i = 1/\ell n(N_i - 1)$. Note the crude similarity between Equation (7d) and the grid stretching ideas of Dwyer et al.[5]

The values of the coefficients κ_1, κ_2, κ_3 may be chosen arbitrarily within certain bounds, and may therefore be used to produce desired weighting effects on scale, geometry and physics, respectively. In that sense, κ_1, κ_2, κ_3 may be called grid clustering intensity coefficients. From experimenting with a few examples the following typical values seem adequate:

$$\kappa_1 = 1; \quad \kappa_2 = -0.1; \quad 0 \leq \kappa_3 \leq 5 \tag{8}$$

For dynamical systems with advection and diffusion of properties, κ_3 is really a function of the average cell Reynolds number $Re_c \equiv UL/(\nu N)$, where U,L are respectively the scaling parameters for velocity and length in the physical system, and ν is the relevant diffusion coefficient.

Any value of $\kappa_3 > 0$ produces grid clustering adaptive to a subject system's physics at an intensity proportional to κ_3. But values of $\kappa_3 > 5$ may tend to overpack the grids leading to both numerical errors and reduced computational efficiency.

Given the invariance, with respect to reference frames, of the total absolute strain of a system, one may refer to a strain field model of a physical system in a reference frame as an "invariant map" of that system. The process of generating such strain field models in different reference frames may then be called "invariant-mapping."

Sample computation of strain density fields

Consider the following simple physical system: a plane parabolic curve--$y = 1 - x^2$, $0 \leq x \leq 1$--with a sine wave distribution $T = \sin(2\pi x)$, of temperature along the curve. Let the system be discretized by N cells. The absolute strain density field may then be computed as follows:

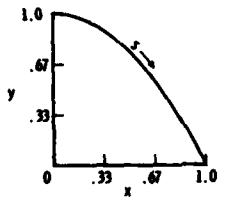
scale influence parameter	$= \psi = 1/\ell n(N - 1)$
base geometric space curvature	$= \mu_A = 1/N$
reference geometric space circular curvature (where: $y_x = \partial y/\partial x$, $y_{xx} = \partial^2 y/\partial x^2$)	$= \mu_G = y_{xx}/(1 + y_x^2)^{3/2}$
reference geometric space torsion	$= \beta_G = 0$
material shear	$= \theta_T = T_s$
material circular curvature (where: $T_s = \partial T/\partial s$, $T_{ss} = \partial^2 T/\partial s^2$)	$= \mu_T = T_{ss}/(1 + T_s^2)^{3/2}$

geometric strain density $= \exp\{\kappa_1 \mu_A + \kappa_2 \bar{\mu}_G)\psi\}$

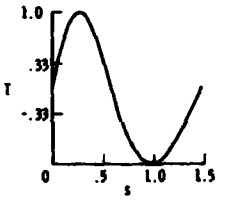
material strain density $= \exp\left\{\kappa_3 \psi \left(|\theta_T| + |\mu_T|\right)\right\}$

absolute strain density $= \exp\left[\psi\left\{(1/N) + \kappa_2 \bar{\mu}_G + \kappa_3 \left(|\theta_T| + |\mu_T|\right)\right\}\right]$

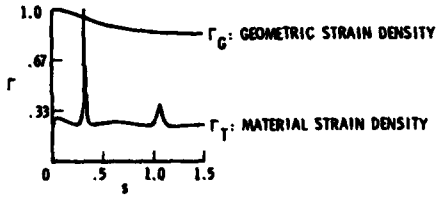
Figure 1 displays these strain density fields, for the parameter values: $\kappa_1 = 1$, $\kappa_2 = -0.1$, $\kappa_3 = 0.05$, $N = 10$.



(a) geometric field



(b) material field



(c) strain density fields

Fig. 1. Sample strain density field model of a physical system

DYNAMIC GRID GENERATION USING INVARIANT-MAPPING

A fundamental requirement of a dynamic grid is that the grid intensity should at all times mimic the unified influence of the scale, geometry and physics of the subject system in the reference frame. This may be assured by equidistributing any proper measure of the unified influence of the scale, geometry and physics of a system among the discretizing grid cells.

Recalling that the absolute strain field of a physical system, as conceptualized in this paper, instantaneously measures the unified influence of the scale, geometry and physics of the system in the system's reference frame, it follows that: "the dynamic grid of a system in a reference frame may be constructed by requiring that the instantaneous absolute strain field of the system in the subject reference frame be equidistributed among the grid cells."

The equidistribution of the instantaneous absolute strain of a system among its discretizing grid cells implies the relation:

$$\Gamma_k(S) \cdot \Delta V_k(S) = \text{constant} = \Phi(t)/N \tag{9}$$

where $\Gamma_k(S)$ is the local value of the strain density in the kth discretizing cell of size $\Delta V_k(S)$, in the reference frame $S\{s,^i,t\}$; $\Phi(t)$ is the instantaneous total absolute strain for the subject physical system; and N is the number of discretizing grid cells.

The invariance of the total absolute strain of a system which respect to reference frames then implies that:

$$\sum_{k=1}^{N} \Gamma_k(S) \cdot \Delta V_k(S) \equiv \Phi(t) \tag{10}$$

$$= \text{instantaneous constant in all reference frames}$$

In view of relations (9) and (10) it follows that between any two invariant-maps of a physical system, each discretized by the same number N, of grid cells, the following relation must hold at any instant:

$$\Gamma_k(R_1) \cdot \Delta V_k(R_1) = \Gamma_k(R_2) \cdot \Delta V_k(R_2) \equiv \Phi(t)/N \tag{11}$$

where $\Gamma_k(R_1)$, $\Gamma_k(R_2)$ are local values of the subject system's strain density fields in corresponding cells of sizes $\Delta V_k(R_1)$, $\Delta V_k(R_2)$ in the reference frames $R_1\{r_1^i,t\}$, $R_2\{r_2^i,t\}$, respectively.

Relation (11) implies that provided a physical system is not an infinitely deformed space a transformation with nonvanishing Jacobian exists by which a physical space dynamic grid for the system may be related to some prototype (or computational) space dynamic grid of prescribed features.

THE TAU COMPUTATIONAL SPACE

In light of the invariant-mapping and dynamic grid generation schemes already presented in this paper it may be observed that if an invariant-map of a physical system is discretized with a dynamic grid whose metric is based upon the instantaneous absolute strain of the subject physical system, then such a dynamic grid will always be uniform and regular. The change of the

space metric will essentially appropriately "stretch" all "equi-strain" regions to become "equi-sized" regions.

The new reference frame represented by this uniform, regular dynamic grid is referred to in this paper as tau space, $T\{\xi^i,\tau\}$.

In order to map a physical system, discretized by N grid cells, from its physical space reference frame $S\{s^i,t\}$ to tau space $T\{\xi^i,\tau\}$, relations (9) and (10) are combined to form the following special tau space invariant mapping relation:

$$\left\{\Gamma_k(S) \cdot \Delta V_k(S)\right\} / \left\{\sum_{k=1}^{N} \Gamma_k(S) \cdot \Delta V_k(S)\right\} = 1/N = \Delta V_k^*(T) \qquad (12)$$

where $\Delta V_k^*(T)$ is the unit measure of grid cell size in tau space. Since $\Delta V_k^*(T) = 1/N$, this unit measure of grid cell size in tau space may be set equal to unity and made time-independent by multiplying relation (12) by N. In so doing, tau space becomes an integer space, with $N \cdot \Delta V_k^*(T) \equiv \Delta V_k(T) = 1$.

A grid dynamism constraint

Applying the tau space relation independently along the boundary-fitted curvilinear coordinate lines one may state in a limiting form the following transformation relation between $S\{s^i,t\}$ and $T\{\xi^i,\tau\}$:

$$\partial s^i/\partial \xi^i = s^i_{\xi^i} = \left(\int_C \Gamma_i \, ds^i\right) / (N_i \Gamma_i) \equiv V_i \qquad (13)$$

where V may be called a "strain function" field, and V_i is its component in the i-coordinate direction; N_i is the number of grid cells along the i-coordinate direction; and, no summation over repeated indices is intended.

Equations (13) show clearly that the transformation metric coefficients relating a system's physical space reference frame $S\{s^i,t\}$ and tau space $T\{\xi^i,\tau\}$ are explicit functions of the subject physical system's absolute strain. Since a system's absolute strain is formulated to mimic the scale, geometry and physics of the system at all times, it follows that Equations (13) constitute a valid set of generation equations for dynamic grids--moving grids adaptive to the scale, geometry and physics of a subject physical system. Therefo , Equations (13) may be called a grid "dynamism constraint" relation.

The grid dynamism constraint is equivalent to the following requirement:

Physical-Tau Space Transformation
Jacobian Field, J = Physical System Strain Function Field, V
= Physical Space Dynamic Grid Cell Size Field

Steger and Sorenson[7] imposed this requirement in their grid method but employed an empirical static model for V instead of the dynamic strain function field introduced in this paper.

One may further establish the transformation metric coefficients in terms of the base coordinate system $X\{x^i,t\}$. This requires knowledge of the transformation relation between $X\{x^i,t\}$ and $S\{s^i,t\}$. If $X\{x^i,t\}$ is Cartesian, then the quadratic form: $(ds^i)^2 = g_{ij}\, dx^i\, dx^j$ may be expressed in terms of the angle of slope α_{ij}, of the curvilinear coordinate curve s^i relative to the base coordinate line x^j; the result is:

$$ds^i = dx^j \cos \alpha_{ij} \tag{14}$$

For a two-dimensional problem space the use of Equation (14) permits the general first-order transformation metric coefficient, $x^i_{\xi^j} = \partial x^i/\partial \xi^j$ to be written in terms only of the strain function field and the angle of slope fields, as:

$$x^i_{\xi^j} = V_j \cos \alpha_{ji} \quad \text{(no summation)} \tag{15}$$

For a three-dimensional problem space the cosine function would be replaced by a more complicated function of α_{ij}. Higher order metric coefficients may be obtained by repeated differentiation of Equations (15).

A grid orthogonality constraint

A requirement of orthogonality would impose constraints on the α_{ij}-fields, commensurate with the constraint that the transformation metric tensor $g_{ij} \equiv x^k_{\xi i} x^k_{\xi j}$ have only diagonal entries.

The α_{ij}-fields may be computed by developing and solving their governing differential equations, for instance as suggested by Warsi.[8] Alternatively, the α_{ij}-fields may be obtained by interpolation between known boundary values, as discussed in Appendix 1 of this paper.

A grid smoothness constraint

In addition to $x^i_{\xi^j}$ and higher derivatives thereof there are also the transformation metric coefficients $x^i_\tau = \partial x^i/\partial\tau$, which indicate the speed of the dynamic grid. The evaluation of x^i_τ has significant consequences on the quality and utility of any adaptive grid method. Rai and Anderson[3] have attempted an empirical model of x^i_τ with encouraging results. A nonempirical expression for x^i_τ may, however, be given, based upon a generalized principle of continuity, namely: "that a property Q, of an object evolves in the space of that object always according to the generalized advection-diffusion law:

$$(\partial Q/\partial t) = \nabla\{-u^* Q + \nu^* \nabla Q\} \tag{16}$$

where t, ∇ are respectively time coordinate and the ∇-operator in the object-space, and u^*, ν^* are respectively the advection and diffusion coefficient fields of the subject property in the object-space."

Such a continuity principle merely specifies a "smooth" field of the subject property in the object-space.

The object-space of interest here is the computational (or tau) space $T\{\xi^i,\tau\}$. In tau space the physical space base coordinates, x^i, are dependent variables which evolve by advection-diffusion processes. Therefore, from Equation (16) a natural constraint on the physical space base coordinates x^i in tau space is:

$$x^i_\tau = \nabla\{-u^* x^i + \nu^* \nabla x^i\} \tag{17a}$$

A physics-adaptive geometry for a problem space essentially requires that: (a) the diffusion coefficient for x^i be identical with the diffusion coefficient ν, of the dependent variable(s) driving x^i, and (b) the advection velocity for x^i mimic the corresponding advection velocity field u, of the dependent variable(s) driving x^i, but always remain smaller in magnitude. That is $\nu^* = \nu$; $u^* = c^* u$ and $0 \leq c^* < 1$. Thus, Equation (17a) may be rewritten as:

$$x^i_\tau = \nabla\{-c^* u x^i + \nu \nabla x^i\} \tag{17b}$$

Equations (17) in conjunction with Equations (15) are a complete set of generation equations for the physics-adaptive geometry of a physical system. Since they derive from the generalized principle of continuity, which

essentially is a "smoothness" constraint on the evolution of property in space, Equations (17) constitute a grid "smoothness constraint" relation.

Equations (17) may further be rewritten in an elliptic form as follows:

$$\nabla(\nu\nabla x^i) = \{x_\tau + \nabla(c^* u x^i)\} \tag{18a}$$

$$= P(\xi^i,\tau)$$

For a boundary-fitted but nonadaptive geometry, $x^i_\tau = 0$ and $c^* = 0$. Thus:

$$\nabla(\nu\nabla x^i) = 0 \tag{18b}$$

One may infer from Equations (18) that the popular elliptic grid generator methods of Thompson, et al[9] constitute an imposition of the grid smoothness constraint in an elliptic form, which requires either no adaption to a system's physics or the empirical formulation of a complicated grid forcing function $P(\xi^i,\tau)$.

A TAU COMPUTATIONAL SPACE METHOD

In general one would first transform a subject physical system from its physical space to a tau space representation. Then, using a suitable finite difference method, one solves the transformed system instant by instant on the uniform, regular, fixed grid of tau space. By computing also the physical space grid corresponding to the fixed tau space grid at each desired instant, one then has an instantaneous physical space solution of the subject system.

The detailed tau computational space method is now presented and illustrated with two sample problems.

Step 0: specify the physical problem space. The physical problem space consists of the set of governing equations and the initial and boundary conditions that describe the geometry and physics of the subject system in physical space.

Step 1: obtain the tau problem space. Transform the physical problem space into the equivalent tau problem space; that is, obtain the transformed governing equations, initial and boundary conditions that describe the geometry and the physics of the subject system in the idealized reference frame--tau space. For this purpose, the relations in Appendix 2 are used. This consists of the application of (a) a grid dynamism constraint--to continuously adapt

the geometry to the physics and scale of the problem; (b) a grid orthogonality constraint--to reduce the number and complexity of the transformation metric coefficients; and (c) a grid smoothness constraint--to functionally express the grid speeds x^i_τ in terms of the system's physics, geometry and discretization scale. The result is a set of equations and conditions containing only T_k, V_i, and α_{ij} as the dependent variables and (ξ^i,τ) as the independent variables.

Step 2: set-up a tau space finite-difference scheme. Set up the desired finite-difference scheme for the tau problem space. For this, it would seem that a locally one-dimensional, weighted-mean finite-difference method would be most compatible. Such a scheme: (a) has nearly identical perspective of the "grid cell" as the tau computational space concept of this paper; (b) employs a three, five, and seven point operator for one-, two-, and three-dimensional spaces, respectively; (c) is antisymmetric in relation to the velocity field; and (d) may be used in conjunction with an explicit or implicit solution method.

Step 3: obtain the tau solution space. Implement the finite-difference solution of the tau problem space, for instance in the manner shown in Figure 2.

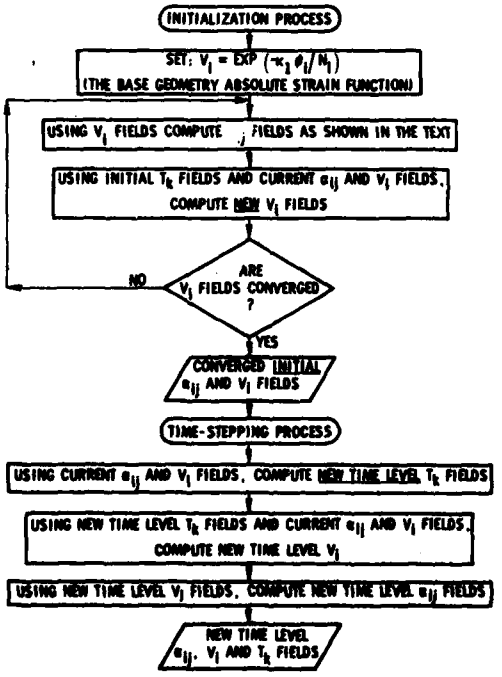


Fig. 2. Flowchart of tau space computational method

It should be noted that the two key parameter fields--α_{ij} and V--critical to these solutions are evaluated in tau space as follows: (a) α_{ij} are evaluated by the simple interpolation scheme outlined in Appendix 1; and (b) V is evaluated along each coordinate direction as:

$V_i = \left(\int_c \Gamma_i \, ds^i\right)/(N_i \Gamma_i)$, with no summation over repeated indices. The tau solution space consists of the instantaneous values of T_k, V and α_{ij} at the tau space grid nodes.

Theoretically, in the numerical solution process each time level (or instant) should be iterated until T_k, V_i, and α_{ij} simultaneously converge everywhere. In practice, however, since one desires only that the discretizing grid mimic (but not necessarily exactly match) the unified influence of the scale, geometry and physics of the subject system, it is usually sufficient for the adaptive grid to lag the physical property fields by one time level. For the initial time level, however, all fields must be solved to simultaneous convergence everywhere.

Step 4: obtain the physical solution space. At desired instants obtain the physical solution space by computing the base coordinates x^i at each tau space grid node, corresponding to the already computed T_k, V, and α_{ij} fields. For this, the grid dynamism constraint relations (15) are integrated along each coordinate curve, using boundary values of x^i.

The integration procedure used in this paper is the following:

$$x^i(\xi^i) = x^i(1) + \left\{x^i(N_i) - x^i(1)\right\}\left\{\int_1^{\xi} V_i \cos\alpha_{ii} \, d\xi^i\right\}/\left\{\int_1^{N_i} V_i \cos\alpha_{ii} \, d\xi^i\right\}; \quad 1 \le \xi^i \le N_i \tag{19}$$

no summation over repeated indices. This normalized form ensures that no boundary overshoot occurs.

This completes the solution of the physical system at the desired time instant.

Illustrative example 1. Consider the following 1-D Burger's equation:

$$u_t = -uu_x + \nu u_{xx} \quad ; \quad -1 \leq x \leq 1$$

$$u(x,o) = 1 \quad ; \quad -1 \leq x < 1$$

$$= -1 \quad ; \quad x = 1$$

$$u(-1,t) = 1 \quad ; \quad u(1,t) = -1 \tag{20}$$

Using the 1-D version of Appendix 2, the transformed equation in tau space is:

$$u_\tau = -u^* u_\xi + \nu^* u_{\xi\xi} \quad ; \quad \xi \geq 1$$

where $$\nu^* = \nu/V^2 \quad ; \quad V = x_\xi \tag{21}$$

$$u^* = cu/V \quad ; \quad c \geq 1 \quad (c = 1/(1 - c^*))$$

and c^* is as defined in Appendix 2. The initial and boundary conditions in tau space conform with the given physical space conditions.

Equation (21) is discretized with a weighted-mean, finite-difference scheme such as is given by Fiadeiro and Veronis.[10] An implicit solution method is employed.

The results are presented in Figures 3 and 4. For all Reynolds numbers convergence to steady state was rapid and the maximum error at steady state was less than 0.01%, provided $0 \leq \kappa_3 \leq 5$ and $1 \leq c \leq 10$ are judiciously chosen--in this case, in proportion to the average cell Reynolds number $Re_c = 2/(VN)$, with $\kappa_3 \to 5$, $c \to 10$ as $Re_c \to \infty$ and $\kappa_3 \to 0$, $c \to 1$ as $Re_c \to 0$.

Illustrative example 2. Consider the following two-dimensional advection-diffusion equation:

$$u_t = -a_1 u_x - a_2 u_y + \nu(u_{xx} + u_{yy}); \quad 0 \leq x \leq 1; \quad 0 \leq y \leq 1 \tag{22}$$

where a_1, a_2 are positive constants or variables.

This is to be solved in a square domain with the initial and boundary conditions:

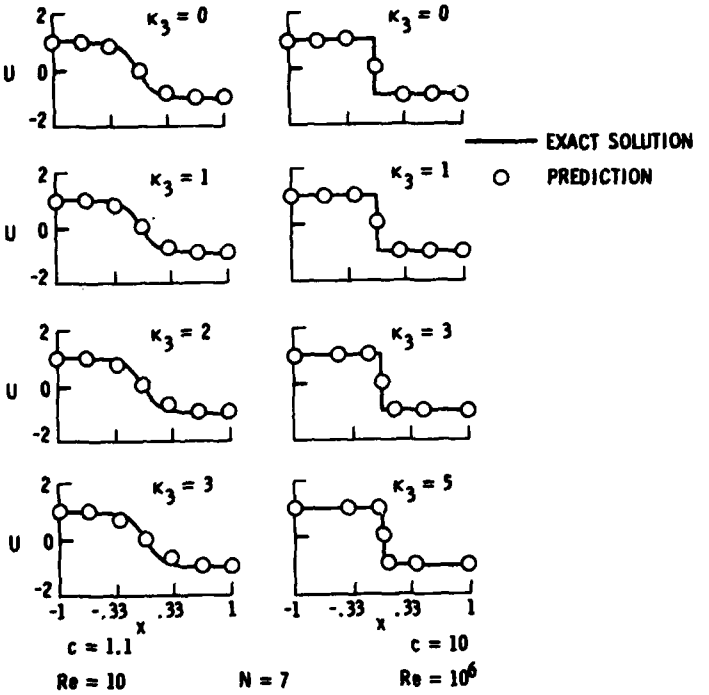
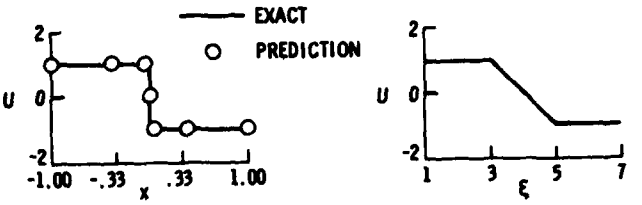
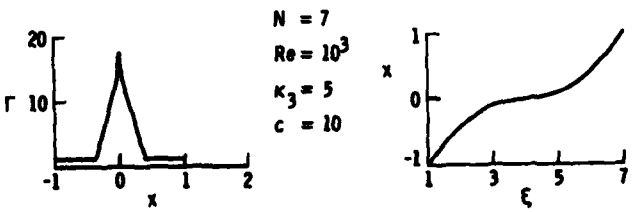


Fig. 3. Steady state solutions of sample problem 1



(a) u-field in physical space (b) u-field in tau space



(c) strain density field (d) x-field in tau space

Fig. 4. Steady state fields of dependent variables in sample problem 1

$$u(x,0,t) = 1 + \{1 - \exp((x - 1)/\nu)\}/\{1 - \exp(-1/\nu)\}$$

$$u(0,y,t) = 1 + \{1 - \exp((y - 1)/\nu)\}/\{1 - \exp(-1/\nu)\}$$

$$u(x,y,0) = 1; \quad x > 0, \quad y > 0$$

$$u(x,1,t) = u(1,y,t) = 1$$

Using the relations of Appendix 2, the transformed equation in tau space is:

$$u_\tau = -u_1 u_\xi - u_2 u_\eta + \nu_1 u_{\xi\xi} + \nu_2 u_{\eta\eta}, \qquad \xi \geq 1; \quad \eta \geq 1 \tag{23}$$

where $u_1 = c_1(a_1 \cos\alpha + a_2 \sin\alpha)/V_1$

$u_2 = c_2(a_2 \cos\alpha - a_1 \sin\alpha)/V_2$

$c_1, c_2 \geq 1$

$\nu_1 = \nu/V_1^2$

$\nu_2 = \nu/V_2^2$

The initial and boundary conditions in tau space conform with the given physical space conditions.

Equation (23) is discretized on a (7 × 7) grid in the ξ, η direction, respectively, using a weighted-mean finite-difference scheme, Reference 10. An implicit locally one-dimensional solution method is employed.

The results for $Re = 5$, $\kappa_3 = 0$, $\kappa_3 = 1$ and $c_1 = c_2 = 1.1$ are presented in Figures 5 and 6. Convergence to steady state was rapid and the maximum error at steady state was very small. As indicated in the previous example, κ_3 and c_1, c_2 are functions of the relevant grid cell Reynolds number Re_c, which is defined in Appendix 2 of this paper.

CONCLUDING REMARKS

The tau computational space method proposed in this paper is a dynamic grid, weighted-mean finite-difference scheme for computing physical systems. All computations are performed in tau space, which at any instant represents the dynamic grid of a subject physical system appropriately stretched into a uniform, orthogonal grid. An implication of this method is that the error in a finite-difference computation of a physical system would be minimized if the finite-difference scheme is of the weighted-mean type and proportionately

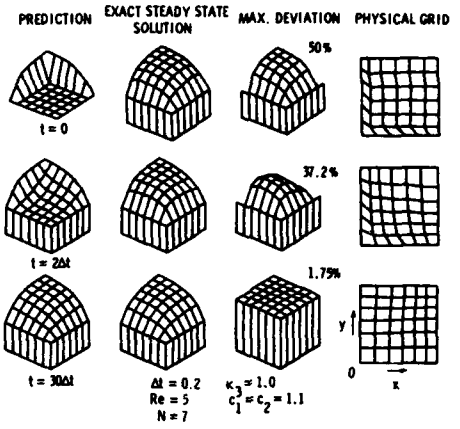


Fig. 5. Solution of sample problem 2, with grid clustering

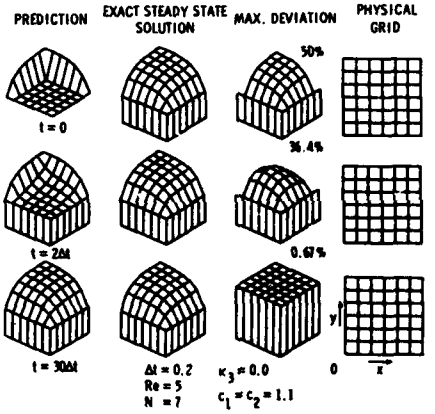


Fig. 6. Solution of sample problem 2, without grid clustering

employs the interactive influences of the physics, geometry and discretization scale of the physical system in the scheme's weighting functions. In such a finite-difference scheme the number $N \geq 3$, of the discretizing grid cell nodes would no longer be an explicit constraint, since N is involved in the scheme's weighting functions.

The relationship, if any, between the error minimization scheme implied in the tau space method and that employed by Gough, et al[4] and Brackbill and Salzman[6] has not been investigated in this paper. Such a study needs to be carried out.

The tau space method employs the three essential classes of constraints--dynamism, orthogonality and smoothness--in a reasonably simple fashion to appropriately regulate the geometry of a physical system. Each of these three

essential classes of constraints can independently provide a control on the geometry of a physical system, as References 3, 11 and 9 have shown. But, the concerted action of all three classes of constraints should be most desirable from the point of view of enhanced finite-difference computation of physical systems.

Other desirable features of the tau space method include the following:

(i) dynamical similarity is inherently maintained between the physical and tau space, by virtue of the strain field idea;

(ii) the strain field, which is a critical element of the tau space method, is formulated as a relatively simple function of the geometrical, physical and scale deformation fields of the subject physical system;

(iii) by specifying a system's geometry in terms of the angle of slope fields, relating the physical reference frame and a uniform base geometry of the system, the system's geometry is functionally interpolated in a rather simple but accurate manner, using only the known boundary geometry and the strain fields of the system. No complicated boundary geometry and governing equations need be derived or solved in order to adapt a system's geometry to its physics. This lends the tau space method the desirability of the algebraic grid generator methods;

(iv) all transformation metric coefficients are evaluated as simple analytic functions of the strain field and the angle of slope fields. This obviates those computational errors usually introduced by a finite-difference recursive evaluation of metric coefficients;

(v) the implementation of the tau space method is reasonably straightforward; the transformation relations are reasonably simple; and the resulting computational efficiency and accuracy, using only a few grid cells, appear to be quite high; and finally

(vi) dynamical systems of more than one dependent variable can be handled without increased difficulty.

By being able to operate accurately and efficiently with minimal number of grid cells, the tau computational space approach may make it possible to solve, at reasonable costs, some large physical systems which hitherto easily overtasked available computer facilities.

Many interesting features of tau space still remain, however, to be fully understood.

REFERENCES

1. Proceedings of Numerical Grid Generation Technique Workshop. NASA CP-2166, October 1980.
2. Eiseman, P. R.: Geometric Methods in Computational Fluid Dynamics. ICASE Report No. 80-11, NASA Langley Research Center, Hampton, VA, April 1980.
3. Rai, M. M. and Anderson, D. A.: Grid Evolution in Time Asymptotic Problems. Proceedings of Numerical Grid Generation Techniques Workshop. NASA CP-2166, October 1980.
4. Gough, D. O., Spiegel, E. A. and Toomre, J.: Highly Stretched Meshes As Functionals of Solutions. Proc. 4th Inter. Conf. Numer. Meth. Fluid Dyn., 1974.
5. Dwyer, H. A., Kee, R. J. and Sanders, B. R.: Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer. AIAA Journal, Vol. 18, No. 10, October 1980.
6. Brackbill, J. U. and Saltzmann, J. S.: An Adaptive Computation Mesh for the Solution of Perturbation Problems. Proceedings of the Numerical Grid Generation Technique Workshop, NASA CP-2166, October 1980.
7. Steger, J. L. and Sorenson, R. L.: Use of Hyperbolic Partial Differential Equations to Generate Body-Fitted Coordinates. Proceedings of Numerical Grid Generation Techniques Workshop. NASA CP-2166, October 1980.
8. Warsi, Z. U. A.: Tensors and Differential Geometry Applied to Analytic and Numerical Grid Generation. Mississippi State University Report, MSSU-EIRS-ASE-81-1, January 1981.
9. Thompson, J. F., Thames, F. C., and Mastin, W. C.: Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing any Number of Arbitrary Two-Dimensional Bodies, NASA CR-2729, July 1977.
10. Fiadeiro, M. E. and Veronis, G.: On Weighted-Mean Schemes for the Finite Difference Approximation to the Advection-Diffusion Equation, Tellus (1977), 29, pp. 512-522.
11. Coleman, R. M.: Generation of Orthogonal Boundary-Fitted Coordinate Systems. Proceedings of Numerical Grid Generation Techniques Workshop, NASA CP-2166, October 1980.

Appendix 1: distribution of α_{ij} in two-dimensional tau space

The general scheme for computing the α_{ij}-fields of a subject physical system in tau space is not given here but may be readily formulated.

If a physical space is a plane and if the orthogonality constraint is imposed, then only one $\alpha_{ij} = \alpha$ field requires to be known. For such cases the distribution of α on the bounding curves would usually be known. Recognising that: (a) at any instant the intermediate curves between the bounding curves must smoothly develop from one to the other; (b) the spacing of these intermediate curves is determined by the transverse component V_2, of the strain function field; and (c) the distribution of points on each bounding or intermediate curve is determined by the longitudinal component V_1, of the strain function field, it follows that if one can establish the adaptive distribution of points on the bounding curves, then along each

ξ = constant curve, transverse to the η = constant curves, one knows the boundary values of α. The intermediate values of α may then be estimated by the following interpolation:

$$\alpha = (\alpha_I + \alpha_G)\ f(\bar{\eta}) \tag{A1}$$

where

$$\alpha_I = \alpha(\xi,1) + \bar{\eta}\{\alpha(\xi,N_2) - \alpha(\xi,1)\}$$

$$\alpha_G = \int_1^{\eta} \Omega\, d\eta - \bar{\eta} \int_1^{N_2} \Omega\, d\eta$$

$$\bar{\eta} = (\eta - 1)/(N_2 - 1)$$

$$f(\bar{\eta}) = 1 - 4\bar{\eta}(1 - \bar{\eta})$$

$$\Omega = \partial v_2/\partial\xi$$

Equation (A1) appears to give results which satisfactorily approximate the solutions of the wave-type equations (Reference 8) governing the evolution of α_{ij} on a plane surface. It contains two parts. The first, $\alpha_I\ f(\bar{\eta})$, is a nonlinear direct interpolation of α between its boundary values explicitly independently of the problem's inner field physics. The second part, $\alpha_G f(\bar{\eta})$ is a "generation-of-α" component. It is explicitly independent of the boundary values of α and reflects and direct influence of the inner field physics on the system's geometry.

Appendix 2: transformation metric coefficients

When a physical system is invariantly mapped onto tau space the governing equations must be rewritten in terms of tau-space coordinates. In general, if $f = F_1(x^i) = F_2(\xi^i)$, $i = 1, 2, 3, 4$, then:

$$\partial f/\partial\xi^j = (\partial f/\partial x^i)(\partial x^i/\partial\xi^j) \tag{A2}$$

From Equation (A2) and higher derivatives thereof generalized transformation relations can be fully constructed for derivatives of the dependent variable f.

From the grid dynamism constraint relations outlined in this paper the transformation metric coefficients for a two-dimensional space may be written as follows:

$$x_\xi = V_1 \cos\alpha;\ x_\eta = -V_2 \sin\alpha$$

$$y_\xi = V_1 \sin\alpha;\ y_\eta = V_2 \cos\alpha$$

$$x_\tau = -c_1^* u_1 x_\xi - c_2^* u_2 x_\eta + \nu_1 x_{\xi\xi} + \nu_2 x_{\eta\eta}$$

$$y_\tau = -c_1^* u_1 y_\xi - c_2^* u_2 y_\eta + \nu_1 y_{\xi\xi} + \nu_2 y_{\eta\eta}$$

$$J = V_1 V_2 = (x_\xi y_\eta - x_\eta y_\xi)$$

$$(x_\xi x_\eta + y_\xi y_\eta) = 0. \qquad \text{(A3)}$$

where u_1, u_2 are the transformed advective velocities of the dependent variable(s) in the ξ and η-directions, respectively, ν_1, ν_2 are the corresponding transformed diffusion coefficients; and $0 \leq c_k^* < 1$, $k = 1, 2$ are constant coefficients whose values tend to unity as $Re_c = UL_k/(\nu_k N_k)$ tends to infinity and tend to zero as $Re_c \to 0$; Re_c is the average grid cell Reynolds number based upon the average physical space grid size (L_N/N_k), the scaling velocity U and the relevant property diffusion coefficient ν_k.

AD P001007

Published 1982 by Elsevier Science Publishing Company, Inc.
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

EQUIDISTANT MESH FOR GAS DYNAMIC CALCULATIONS

C. M. ABLOW
SRI International
Menlo Park, CA 94025

This document states fact

The accuracy of finite difference solution of the differential equations of fluid flow is increased by fitting the mesh to flow boundaries and refining it where the solution has sharp variation. Fitting a mesh to flow boundaries is a subject of current research effort.[1] Refinement in layers at the boundaries is readily accomplished in a boundary-fitting grid.

The wave fronts and shocks of many gas dynamic flows present additional layers of sharp variation away from the boundaries. Before the benefits of mesh refinement in these regions can be obtained, simultaneous flow solution and grid adjustment are needed. A split calculation that alternates between steps of solution and steps of mesh improvement would be nearly as fast and easier to implement. The results reported here are derived for the mesh improvement step of the split calculation. A properly refined mesh is obtained that fits a given solution function.

An optimally refined grid for numerical computation minimizes the truncation error of the difference equations being solved. This grid has proven to be expensive and difficult to construct for one-dimensional problems.[2]

A readily constructed grid that can reduce truncation error uses distance s along the solution curve as the equally incremented computational coordinate. Let x be the independent variable and z the dependent one, and let variable subscripts denote differentiation. Then

$$x_s^2 + z_s^2 = 1$$

so that x_s and z_s are restricted to being less than 1 in magnitude for any slope z_x. Even though a system with two dependent variables, x and z, needs to be solved, much more accurate results are obtained on several examples[2] than with the original one-dependent-variable system. Adding multiplesof the curvature[3] to the distance s further increases the accuracy but also produces a system of higher order.

A generalization to two-dimensional surfaces of the one-dimensional curvilinear distance coordinate is the equidistant mesh, in which the grid points divide each grid line into segments of equal lengths. If the flow region is

compact and simply connected, a computational domain with the same topology may be taken to be the unit square. After the corners of the square are located on the boundary, the boundary segments are mesh lines with equidistant mesh points. An alternating direction implicit solution can then be carried through to locate the interior mesh points.

The difference equations to be solved are written in terms of the grid as projected into the (x,y) plane. For low truncation error the projected grid should be as nearly orthogonal as possible.[4] The corner points are therefore moved, by an outer iterative calculation, toward achieving orthogonality. The algorithm for this minimizes the sum of the squares of the cosecants of all the angles, a quantity readily calculated from the vector products of the sides of each angle.

Figure 1, taken from a previous report,[5] shows the grid obtained with a rather academic example. The mesh points are properly clustered where the solution surface is steep. The corner points are located so that at least some of the angles are nearly 90 degrees.

The gas dynamic example in Figure 2 represents the flow of air past a semi-infinite slab with a wedge-shaped leading edge. The wedge half-angle has been chosen to be 14.3 degrees so that the bow shock is at 45 degrees to the on-coming Mach 2 flow.[6] The rarefaction from the shoulder is contained between characteristics of slope 58.1 and 30.8 degrees. The flow region is a rectangle 8 slab-widths long and 4 wide that extends one width upstream of the leading edge. The region is not wide enough for the shock and rarefaction wave to interact.

Function z has been taken as a smooth approximation to the Mach number. With the origin of Cartesian coordinates at the leading edge, the flow is approximated by

$$z = 1.728 + 0.272 \tanh 2(y-x)$$

upstream of the rarefaction wave. Downstream of the rarefaction wave, $z = 1.95$. For the rarefaction wave, there is a cubic spline interpolation in the slope between $z = 1.46$ and the $z = 1.95$. To avoid a singular point on the boundary, the center for the approximation to the rarefaction wave was moved 0.1 slab width below the shoulder.

The 8 x 8 equidistant grid for the above function z is shown in Figure 3. Little effect of the function itself is evident. Also the corners of the flow region apparently inhibit the motion of the computational corners, which remain very close to initially guessed locations.

To increase the effects of functional variations, the function values were multiplied by 5 and 10 with the resulting grids shown in Figures 4 and 5. The functional effects are increasingly evident: the grid lines cluster near the shock and roughly fan out in the rarefaction. In Figure 6 the 4 x 8 grid is presented for the same function as in Figure 5. Differences between Figures 5 and 6 are small. One may conclude that mesh refinement will not appreciably alter the grid.

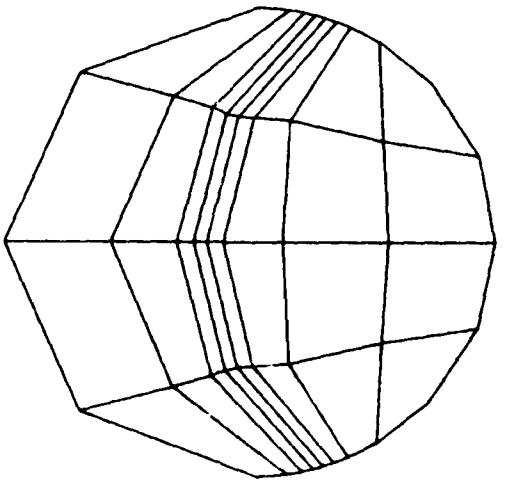
The equidistant grids satisfactorily cluster near the shock and the flow corner, but are so distorted as to be unlikely to increase the accuracy of the difference calculation. An alternative to the equidistant mesh on the solution surface is an isometric mesh. Isometric grid lines are orthogonal to one another and become more nearly equidistant at any point the finer the mesh.[7] The isometric grid is found by solving Beltrami's equations for the two grid coordinates. The use of these equations has been suggested[8] and some calculations made[9] by analogy with the widely used Laplace grid generators[10] to which they reduce when the surface is plane. A disadvantage of the isometric grid is that it is orthogonal on the surface rather than in its projection on the (x,y) plane.

One may conclude that the equidistant mesh does automatically provide refinement in regions of large variation. However, the mesh can be useful only if a way of obtaining more nearly orthogonal grids is found.

Support of this work by the Air Force Office of Scientific Research is gratefully acknowledged.

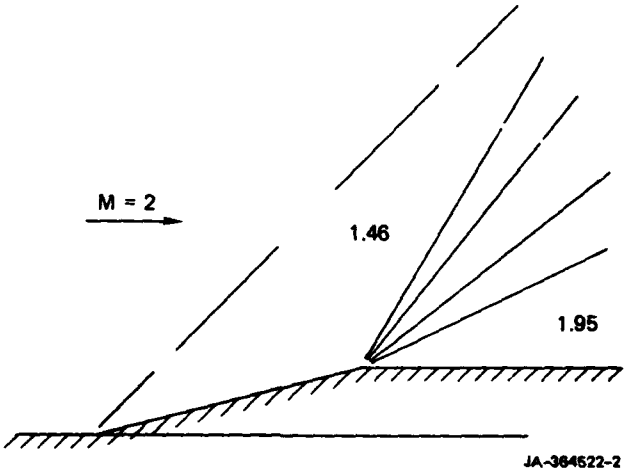
REFERENCES

1. Eiseman, P. R. (1980) Geometric Methods in Computational Fluid Dynamics, ICASE Report 80-11, NASA Langley, Hampton, VA.
2. Ablow, C. M., Schechter, S., and Swissler, W. H. (submitted to J. Comp. Phys.) Node Selection for Two-Point Boundary-Value Problems.
3. Ablow, C. M. and Schechter, S. (1978) J. Comp. Phys., 27, 351-362.
4. MacCormack, R. W. and Paullay, A. J. (1974) Computers and Fluids, 2, 339-361.
5. Ablow, C. M. and Schechter, S. (1980) Generation of Boundary and Boundary-Layer Fitting Grids, in Numerical Grid Generation Techniques, NASA CP 2166, NASA Langley RC, Hampton, VA.
6. Handbook of Supersonic Aerodynamics (1950) NAVORD Report 1488, Bureau of Ordinance, U.S. Navy Dept., Washington, D.C.
7. Eisenhart, L. P. (1960) A Treatise on the Differential Geometry of Curves and Surfaces. Dover Publications, New York.
8. Warsi, Z.U.A. (this symposium) Basic Differential Models for Coordinate Generation.
9. Warsi, Z.U.A. and Ziebarth, J. P. (this symposium) Numerical Generation of Three-Dimensional Coordinates Between Bodies of Arbitrary Shapes.
10. Thompson, J. F. and Mastin, C. W. (1980) Grid Generation Using Differential Systems Techniques, in Numerical Grid Generation Techniques, NASA CP 2166, NASA Langley RC, Hampton VA.



JA-364522-1

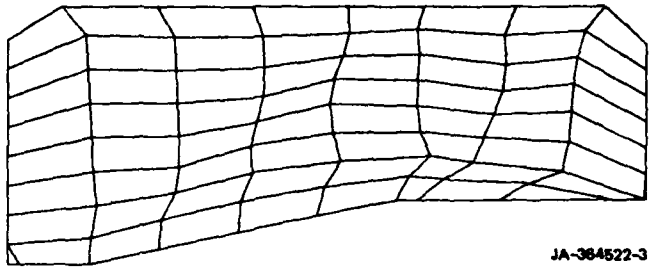
FIGURE 1 EQUIDISTANT MESH FOR z = tanh 8(r -1.2) IN THE UNIT DISK, WHERE r IS THE DISTANCE FROM THE POINT FURTHEST TO THE RIGHT



JA-364522-2

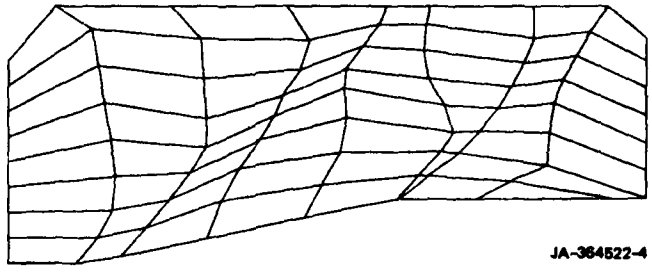
FIGURE 2 AIRFLOW OVER A WEDGE-CUT SEMI-INFINITE SLAB

Mach numbers shown for regions of uniform velocity.



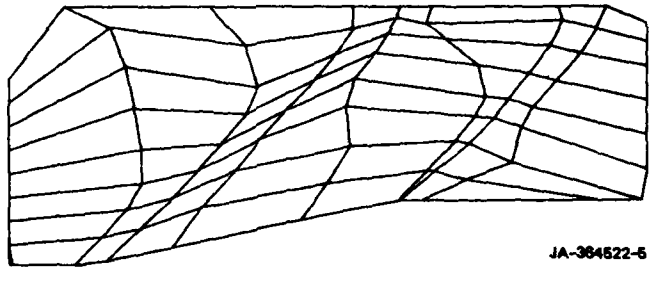
JA-364522-3

FIGURE 3 EQUIDISTANT GRID FOR FLOW MACH NUMBER



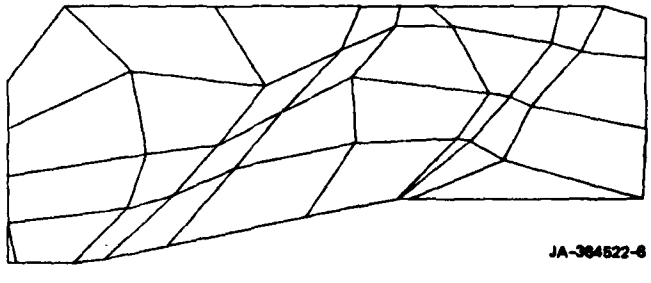
JA-364522-4

FIGURE 4 EQUIDISTANT GRID FOR FLOW MACH NUMBER TIMES 5



JA-364522-5

FIGURE 5 EQUIDISTANT GRID FOR FLOW MACH NUMBER TIMES 10



JA-364522-6

FIGURE 6 EQUIDISTANT GRID FOR FLOW MACH NUMBER TIMES 10

AD P001008

Published 1982 by Else
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

*Applications and Generalizations of Variational Methods for Generating Adaptive Meshes

Jeffrey Saltzman and Jeremiah Brackbill
Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545

INTRODUCTION

Generating computation meshes for irregular regions have been of interest to a lot of people in many areas of research for a long time. One technique that has met with success over the long run has been to generate meshes using an elliptic equation or a system of elliptic equations.

The technique in its simplest form, uses a system of Laplace equations which are solved by direct or iterative methods. As people gained more experience with this method, source terms were added to the Laplace equations to gain additional control of the mesh. In addition, variable coefficients of the derivatives were added for further flexibility.

The authors

In this paper ~~we~~ work with a method that systematically generates a set of elliptic equations without having to explicitly perturb a set of Laplace equations with source terms and variable coefficients. This technique uses the variational methods often associated with elliptic equations.

Following this introduction, ~~we will~~ they briefly discuss the variational formulation in two-dimensional cartesian geometry. Then the formulation will be generalized to three dimensions. Next, several three-dimensional test problems will be shown. After displaying these three-dimensional results, we will then exhibit an application of the mesh generation technique in two dimensions. This application involves generating an adaptive mesh for a supersonic flow past a step in a wind tunnel.

*This work performed under the auspices of the Department of Energy
LAUR-81-3335

PREVIOUS PAGE IS BLANK

THE VARIATIONAL FORMULATION IN TWO DIMENSIONS

Finite difference schemes on nonuniform meshes all have the obvious characteristic that the independent variables in the calculation must be kept track of as well as the dependent variables. This is usually done by introducing an indexing scheme that tags the independent variables in the same was as the dependent variables. For example, in two dimensions, this is done by assigning two indices to a variable such as the fortran dimension statement

```
dimension x(30,40), y(30,40)
```

This is quite suggestive of a mapping. This mapping is one from a rectangular array of integers to a set of real coordinates. By filling in between the points using an interpolation scheme of some sort we now have a continuous mapping of a rectangular region into some two-dimensional region. This mapping is illustrated in figure 1. To be more systematic we will call the collection of points formed from the indices of the grid points the parameter space while we will call the collection of points formed from the grid points the physical space. With this mapping, we now have a tool to describe qualities of the given mesh in a quantitative manner.

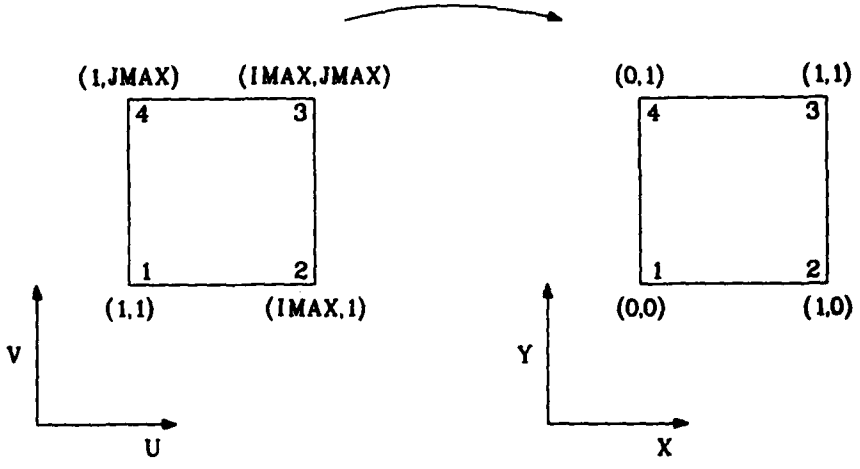


Fig. 1

As illustrated in the first figure we consider a mapping from the two-dimensional parameter space (u,v) to the space (x,y). We can quantify a mapping between these two spaces using the following functionals.

$$I_s = \iint (\nabla_{x,y}u)^2 + (\nabla_{x,y}v)^2 \, dxdy \tag{1}$$

$$I_o = \iint (\nabla_{x,y}u \cdot \nabla_{x,y}v)^2 \, dxdy \tag{2}$$

$$I_v = \iint w(x,y)J \, dxdy \tag{3}$$

where

$$J = \frac{\partial(x,y)}{\partial(u,v)} \tag{4}$$

and w is a given function of x and y.

We now describe the meaning of each functional.

The integral in equation (1) measures the smoothness of the mapping from (u,v) to (x,y). In particular, the gradients in the integrand measures the spacing of the constant u and v lines. It seems plausible that a mesh that has smooth changes in spacing would have a functional value less than a jaggedly spaced mesh. We will call this integral the smoothness functional. The integral in equation (2) measures the orthogonality of constant u and v lines. If the mesh were perfectly orthogonal then the integral would be zero. We will call this integral the orthogonality functional. The integral in equation (3) measures how well the volume elements are conforming to a given weight function w(x,y). If we were to minimize this integral, we would predict that where w is large J should be small and conversly where J is large w should be relatively small. Further, if J is small in a neighborhood of some point P then the grid should have many points close together in a neighborhood of the point P. We will call this last integral the volume weighting functional.

The functionals in the first three equations all measure useful qualities of a mesh. Both smoothness and orthogonality of a mesh are important to maintain accurate differencing. The volume weighting functional is useful in measuring

how a mesh is is adapting to a given function. In addition to measuring mesh qualities, these functionals can be used to generate meshes as well. This is done by deriving a system of elliptic equations from the functionals. This process is broken into several steps.

The first step in this process is to write the integrals using (u,v) as the independent variables. This is useful later on when we will difference some equations. Next we take a linear combination of the integrals. The lambdas are all chosen positive and their relative size determines the importance given to each integral. With a single sum defined, we can minimize this integral using the methods of the calculus of variations.

$$I = \lambda_s I_s + \lambda_o I_o + \lambda_v I_v \tag{5}$$

$$\left\{\frac{\partial}{\partial x} - \frac{\partial}{\partial u}\frac{\partial}{\partial x_u} - \frac{\partial}{\partial v}\frac{\partial}{\partial x_v}\right\} F = 0 \tag{6}$$

$$\left\{\frac{\partial}{\partial y} - \frac{\partial}{\partial u}\frac{\partial}{\partial y_u} - \frac{\partial}{\partial v}\frac{\partial}{\partial y_v}\right\} F = 0 \tag{7}$$

where F is the integrand of the right hand side of equation (5).

To do so we calculate the Euler derivative of the integrand of the sum in equation (5). These expressions are listed in equations (6) and (7). Notice that having written the integrals in terms of (u,v) we can now difference the corresponding elliptic equations using symmetric differences on a rectangular region. We have chosen to solve the equations using an iterative scheme. This scheme is the classical Gauss-Jacobi iteration which is very amenable to vectorization. Now that we have given a sketch of the two dimensional equations, we move onto three dimensions. References [1,2] cover the two dimensional equations in more detail. One particular detail that should not be left to the references is that in practice the orthogonality term is multiplied by the term J^3 to counter problems with rounding errors. This new term slightly alters the effect of the functional in that regions where J is large are orthogonalised more than regions where J is small.

GENERALIZATION OF THE VARIATIONAL FORMULATION TO THREE DIMENSIONS

The variational formulation easily generalizes to three dimensions. The first step in generalizing the equations is to define the appropriate mapping. This is simply done by adding one space dimension to both (u,v) and (x,y) to get (u,v,w) and (x,y,z). Equations (8), (9), (10) and (11) are the appropriate generalizations of equations (1) thru (4).

$$I_s = \iiint (\nabla_{x,y,z} u)^2 + (\nabla_{x,y,z} v)^2 + (\nabla_{x,y,z} w)^2 \, dxdydz \tag{8}$$

$$I_o = \iiint (\nabla_{x,y,z} u \cdot \nabla_{x,y,z} v)^2 + (\nabla_{x,y,z} u) \cdot \nabla_{x,y,z} w)^2 + (\nabla_{x,y,z} v) \cdot \nabla_{x,y,z} w)^2 \, dxdydz \tag{9}$$

$$I_v = \iiint w(x,y,z) J \, dxdydz \tag{10}$$

where

$$J = \frac{\partial(x,y,z)}{\partial(u,v,w)} \tag{11}$$

The only real additional complexity is found in the orthogonality functional where two additional terms must be added to insure that orthogonality of all coordinate lines are measured. Again linear combinations of the integrals are taken and the Euler derivative of the integrand of the sum of these integrals is calculated. Equations (12) thru (15) list these expressions.

$$I = \lambda_s I_s + \lambda_o I_o + \lambda_v I_v \tag{12}$$

$$\left\{\frac{\partial}{\partial x} - \frac{\partial}{\partial u}\frac{\partial}{\partial x_u} - \frac{\partial}{\partial v}\frac{\partial}{\partial x_v} - \frac{\partial}{\partial w}\frac{\partial}{\partial x_w}\right\} F = 0 \tag{13}$$

$$\left\{\frac{\partial}{\partial y} - \frac{\partial}{\partial u}\frac{\partial}{\partial y_u} - \frac{\partial}{\partial v}\frac{\partial}{\partial y_v} - \frac{\partial}{\partial w}\frac{\partial}{\partial y_w}\right\} F = 0 \tag{14}$$

$$\left\{\frac{\partial}{\partial z} - \frac{\partial}{\partial u}\frac{\partial}{\partial z_u} - \frac{\partial}{\partial v}\frac{\partial}{\partial z_v} - \frac{\partial}{\partial w}\frac{\partial}{\partial z_w}\right\} F = 0 \qquad (15)$$

Again, in practice, the integrand of the orthogonality term is multiplied by J^3 to reduce problems with rounding errors.

To demonstrate how these variational principles work in three dimensions several model problems will be presented. The first problem has a cubic physical region. Within this region a spherical exponential weight function is defined about the center of the cube. This problem will show how the weight functional influences the mesh. The second problem has a physical region that looks like an inverted pyramid with the tip cut off. This frustum problem will use the orthogonality functional to show its effect on the mesh. The last model problem uses the smoothness functional to generate a mesh around a cylindrical fuselage and an attached wing.

Figure 2 shows the mapping for the first model problem. The figure is a bit misleading since the cube actually rests at the origin. There are 20 points in each coordinate direction yielding a total of 8000 points. However only 18 x 18 x 18 (5832) points are actually iterated on since the others are fixed at the boundary. We use the following weight function. In this problem the

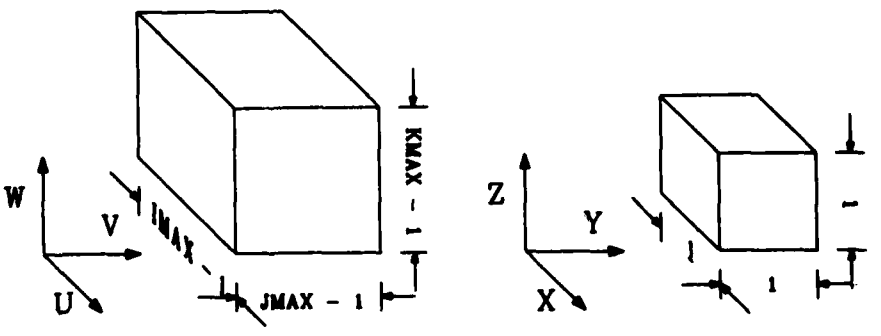


Fig. 2

weight λ_s for the smoothing functional will be set to one while the weight of the orthogonality functional will be set to zero. The weight for the volume weighting functional will vary between zero and one. To be able to show the full variation in the effect of the weight function while varying the parameter between zero and one a geometry dependent normalization was introduced for both the weighting functional and the orthogonality functional. These normalizations are introduced by dividing the lambdas by the lambda primes introduced in equations (16), (17) and (18).

$$\lambda_o' = \left(\frac{1}{\bar{l}}\right)^7 \tag{16}$$

$$\lambda_v' = \left(\frac{1}{\bar{l}}\right)^5 \hat{w} \tag{17}$$

$$\hat{w} = \frac{1}{V} \iiint w(x,y,z)\, dxdydz \tag{18}$$

where

V is the physical volume region.

l is the physical length scale.

$\bar{l}$ is the parameter length scale (number of points in a direction).

These normalizations were arrived at by using dimensional analysis. The "dimensions" of the weighting functional and the orthogonality functional were normalized to those of the smoothing functional. Finally the weighting function w(x,y,z) is defined in equation (19).

$$w(x,y,z) = 1000 \exp[(0.25 - (x^2 + y^2 + z^2)^{1/2})^2 / 0.05] \tag{19}$$

Figure 3 shows the behavior of the weighting functional as a function of lambda. The behavior of the curve is what one expects since as more weight is given to the functional its influence should be felt more.

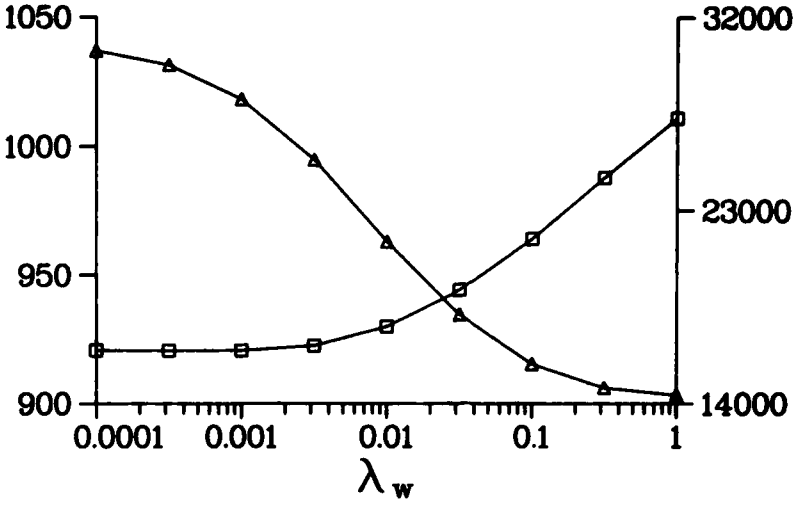


Fig. 3
The triangles mark values (right axis) of the volume weighting functional and the squares mark values (left axis) of the smoothing functional.

Figure 4 shows various cross-sections of the mesh when λ_v is set to one.

Figure 5 illustrates the frustum used in the second model problem. The coordinates of the bottom plane of the frustum are (x,y,z) = (0.25,0.25,0.5), (0.25,0.75,0.5), (0.75,0.75,0.5) and (0.75,0.25,0.5). The coordinates of the top plane are (x,y,z) = (0.0,1), (1,0,1), (1,1,1) and (0,1,1). In this problem the weighting for the smoothness functional is set to one while the weight of the orthogonality functional varies between 0 and 1000 . The volume weighting functional is not used and subsequently has weight zero. Figure 6 shows the behavior of the orthogonality functional as a function of λ_o for the given range 0 to 1000 . Again the curve exhibits a predictable behavior. Figure 7 shows several cuts made into the frustum. The cuts display how the mesh generation algorithm pushes the grid planes upward to make as many points as orthogonal as possible.

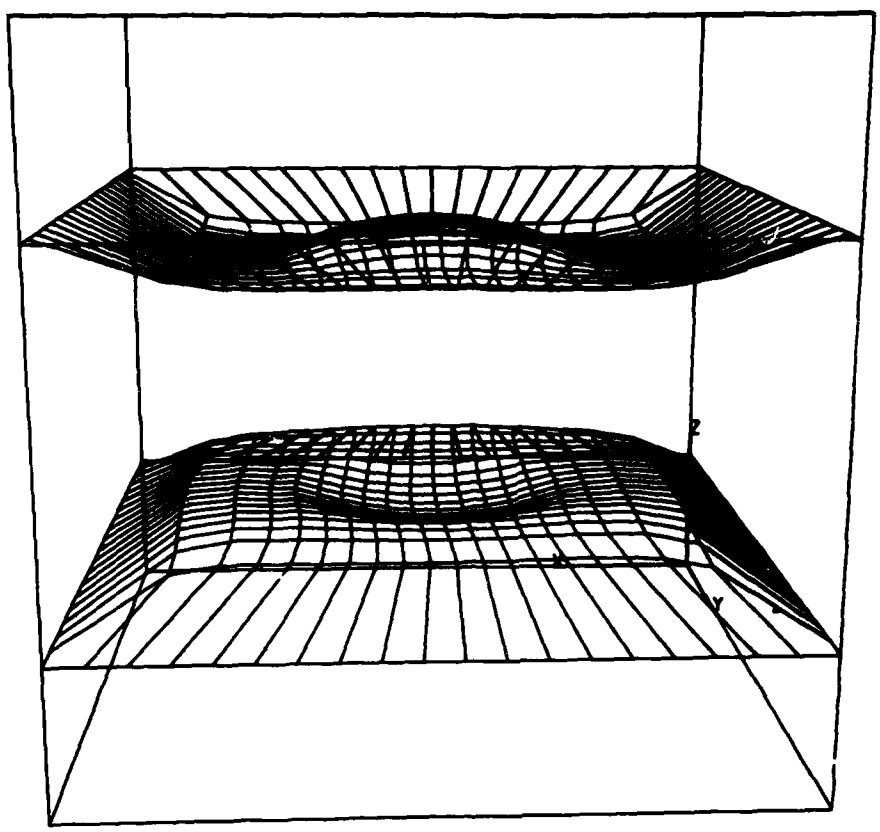


Fig. 4

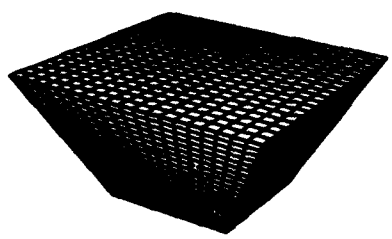


Fig. 5

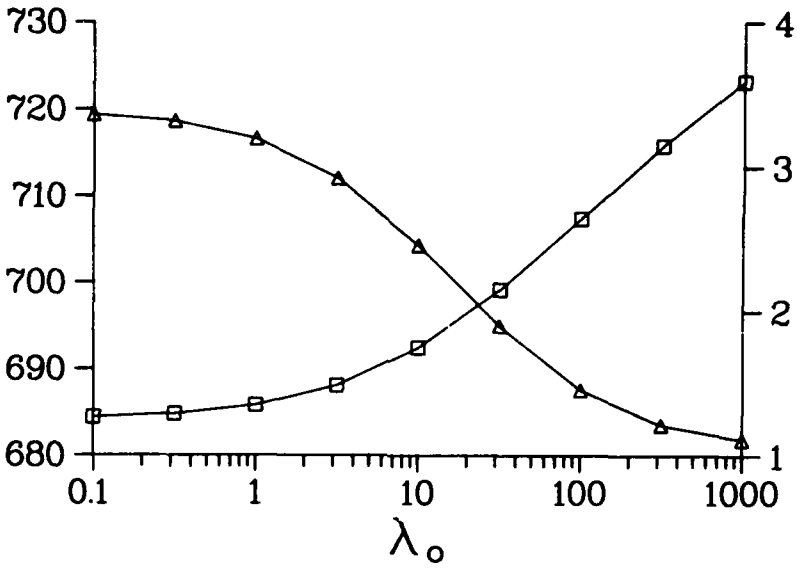


Fig. 6

The triangles mark values (right axis) of the orthogonality functional and the squares mark values (left axis) of the smoothing functional.

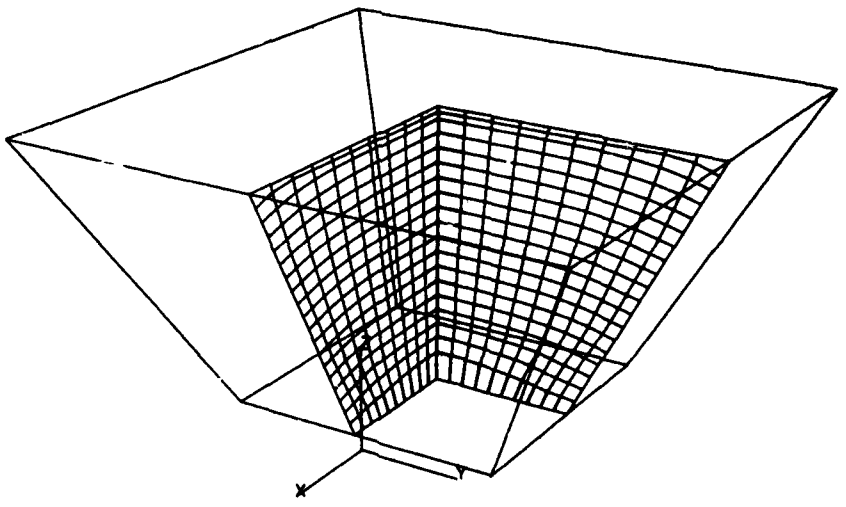


Fig. 7

Figure 8 displays the last model problem used to demonstrate the mesh generator. This figure shows a cylindrical fuselage with a wing attached.

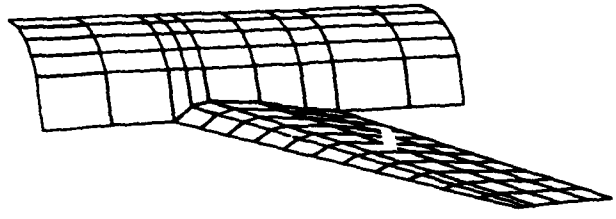
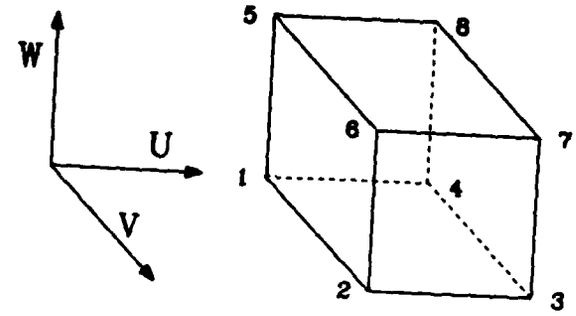


Fig. 8

The fuselage is extended so that the cylinder extends -90 degrees from the horizontal. In addition, a wall is constructed perpendicular to the wing. With these extensions a rectangular grid is wrapped around the wing as illustrated in figure 9 . In this problem the smoothness functional will first be used without either of the other functionals to generate a regular mesh. In figure 10 a cut, seen from the front of the fuselage, of the grid is shown after 50 iterations. The mesh appears to be regular but has one troubling characteristic. One notices that the mesh is tightly spaced around the edge of the wing. After examining the equations derived from the smoothing functional it becomes clear by using electrostatic analogies that the mesh lines should bunch around the sharp wing edge much as potential lines concentrate about a lightning rod.

One possible way to fix the problem at the edge is to use the weighting functional to redistribute the grid points away from the edge by choosing a weight function appropriately. Figure 11 shows the region where the weight function is large. Elsewhere the weight function is very small or zero. The form of the function is unimportant. However, its location is away from the edge making it a good candidate to pull the mesh lines away from the wing edge. Finally figure 12 shows a cut made in the same manner as in figure 10.



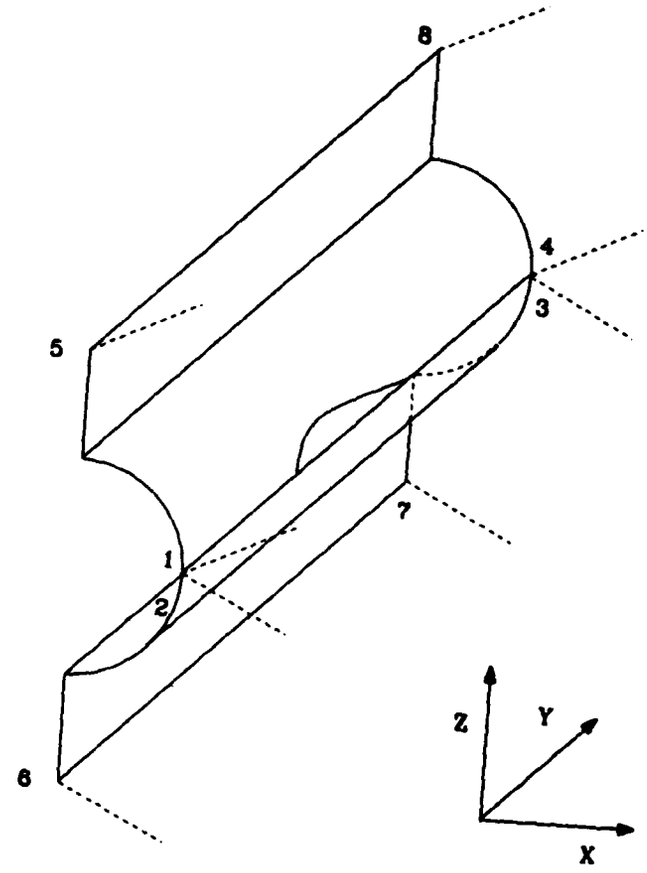


Fig. 9

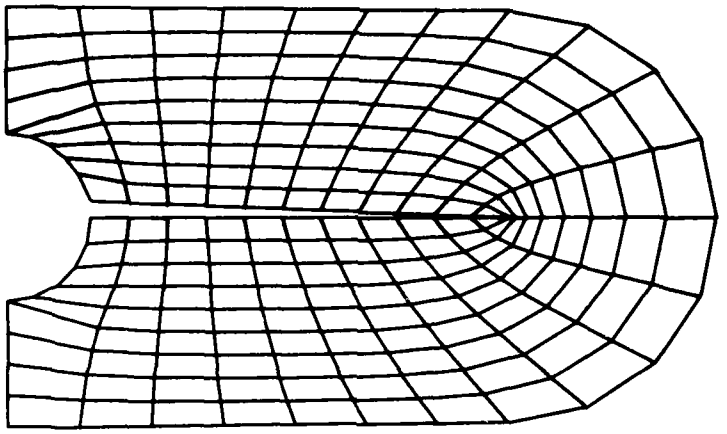


Fig. 10

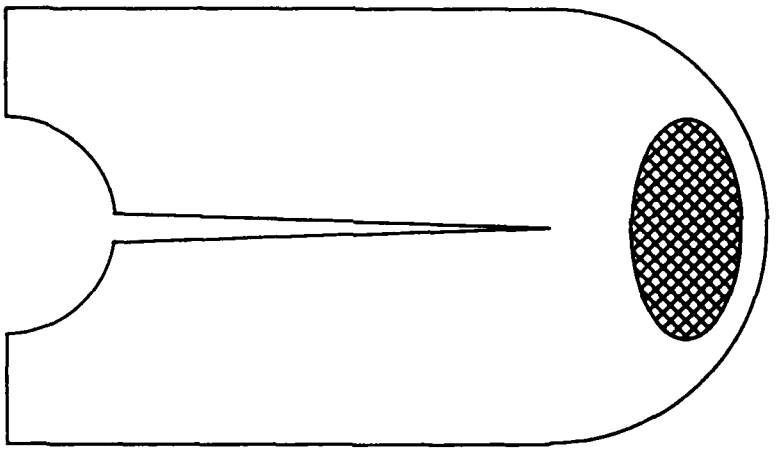


Fig. 11

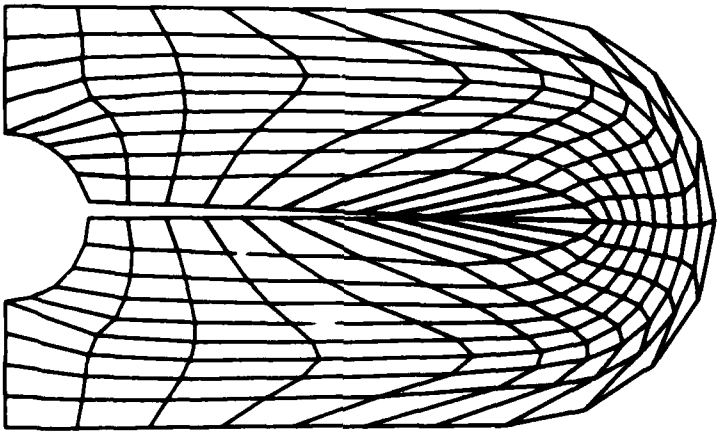


Fig. 12

AN APPLICATION IN TWO DIMENSIONS

In this section we will show how the mesh generator can be combined with a two-dimensional cartesian hydrodynamics code. All three functionals will be used to make the cartesian code adaptive. The step in a wind tunnel problem will be solved numerically to demonstrate the power of the method. This problem has been studied by many people and more information about its background can be found in references [2,3]. We first discuss the hydrodynamics code.

An appropriate model for the wind tunnel problem is the compressible inviscid fluid equations in two dimensional-cartesian geometry. These equations are listed below using a lagrangean formulation and an ideal equation of state.

$$\frac{d\rho}{dt} + \rho \nabla \cdot \vec{u} = 0 \tag{20}$$

$$\rho \frac{d\vec{u}}{dt} + \nabla p = 0 \tag{21}$$

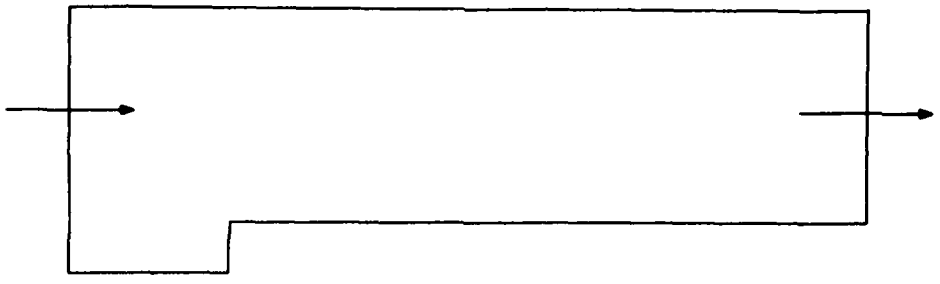
$$\rho \frac{dI}{dt} + p \nabla \cdot \vec{u} = 0 \tag{22}$$

$$p = (\gamma - 1) \rho I \tag{23}$$

The standard variable names are used. ρ is the density, p is the pressure (which could include viscous contributions), $\vec{u}$ is the velocity, I is the internal energy, t is time and gamma is the ratio of specific heats. Although the fluid equations are inviscid, a viscous pressure will be added to prevent ringing at shock fronts.

The equations will be differenced in two steps. The first step, called the Lagrangean phase, will approximate the above equations. Following the Lagrangean phase a remap phase will follow allowing an arbitrary mesh to be used. Because of length constraints on this paper, we cannot go into detail on the differencing. However, some general comments should be made about the differencing.

In the Lagrangean phase a conservative differencing scheme is used on a staggered quadrilateral grid. The velocities and masses are vertex centered while pressure and internal energy are cell centered. Differencing is explicit in time and stability is maintained using a courant limitation on the time step. More details can be found in reference [2]. The remap phase of the calculation is also conservative making the entire scheme conservative. The central idea employed in the remap phase is the utilization of the fully two-dimensional FCT algorithm of Zalesak generalized to an arbitrary



INFLOW PARAMETERS

$\rho = 1.4$
$P = 1.0$
$u = 3.0$
$v = 0.0$
$\gamma = 1.4$

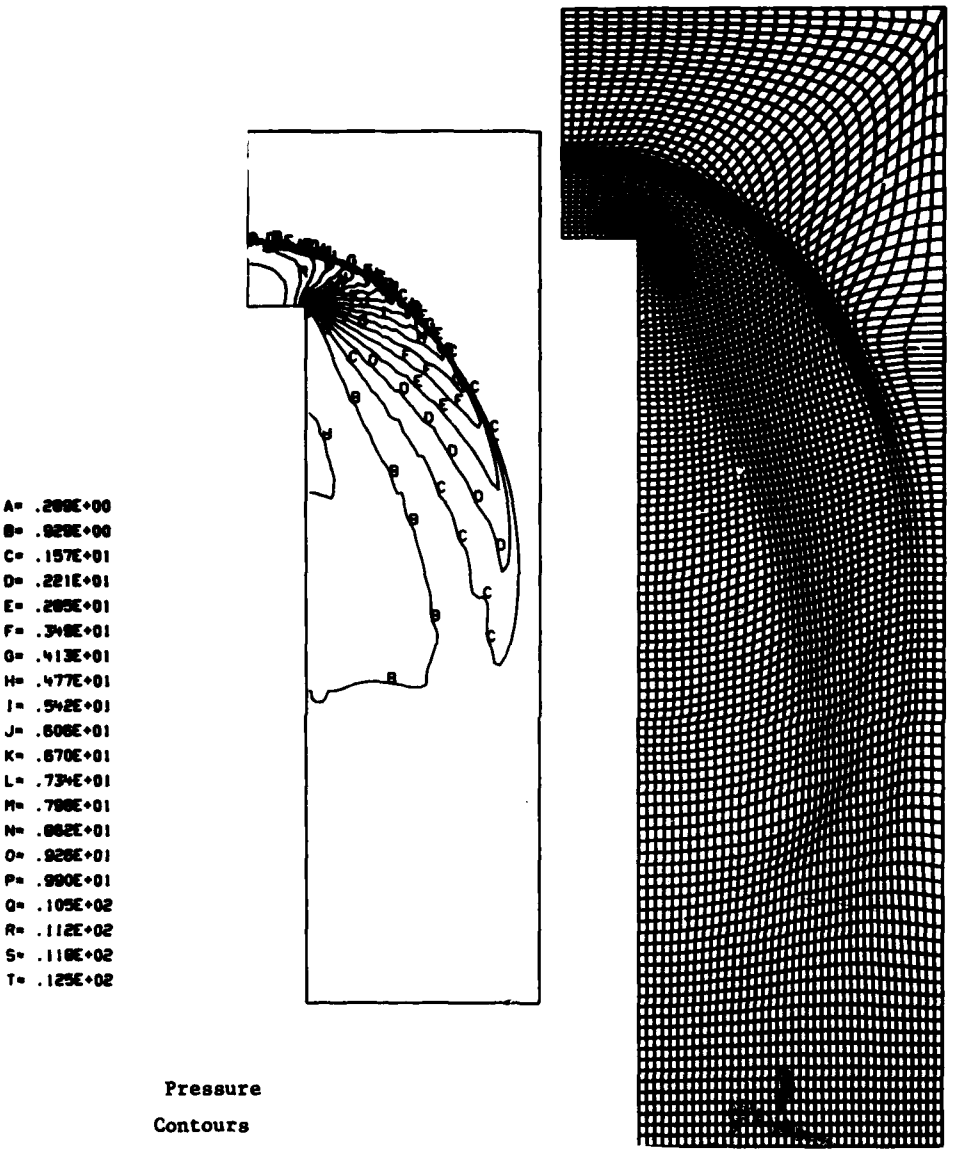
Fig. 13

quadrilateral mesh (references [2,4]). The remap phase is dissipative insuring stability.

Figure 13 is a schematic of the initial conditions and boundary of the wind tunnel problem. Free slip boundary conditions are used on the top and bottom boundaries since the model is inviscid. At the corners of the step, the velocities are constrained to a direction parallel to a chord formed using adjacent boundary points. The outflow boundary conditions simply set all normal derivatives to zero. As it turns out, the flow is always supersonic at the outflow boundary making the boundary conditions irrelavent. The inflow boundary is set to make the flow Mach 3 . The interior initially also has a uniform Mach 3 flow in the horizontal direction. Next, the initial mesh (120 x 40 cells) is created using the smoothness functional. Finally we introduce the weight function used in the volume weighting functional. It is well known for this problem that multiple shock structures develope throughout the region. To resolve these structures we have chosen the gradient length of pressure listed in equation (24).

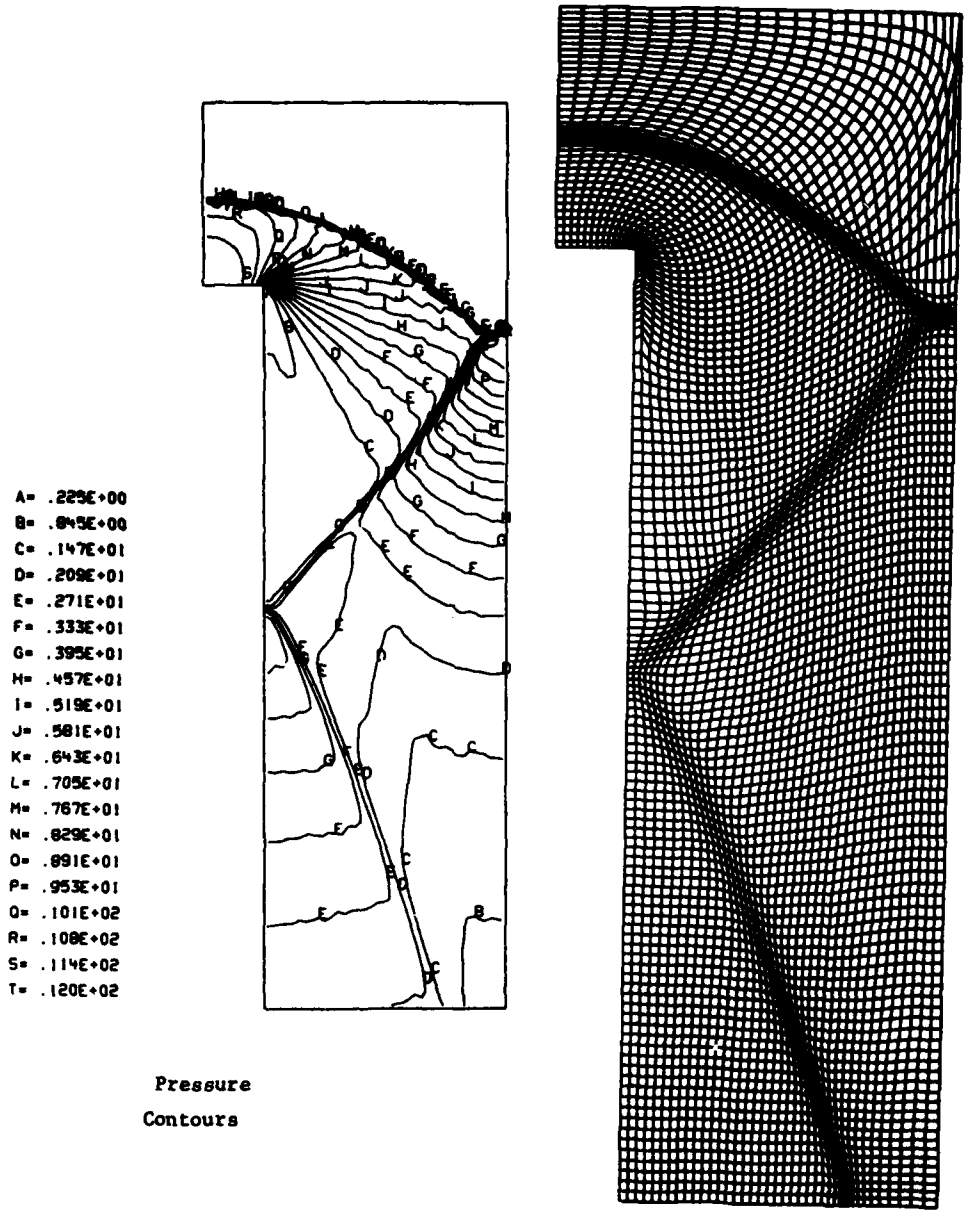
$$w(x,y) = \left|\frac{\nabla p}{p}\right|^2 \qquad (24)$$

Figures 14 and 15 are snapshots of the computation at times 0.5 and 2.0 respectively. The primary purpose of these illustrations is to show that the grid changes drastically over time in order that it may follow the structure of the flow. Further comparisons can be made at fixed times to show that the grid is concentrated around gradients in the pressure. Another property of the computation mesh that is observed is how it aligns itself with the gradients of the pressure. The cells contract in a direction parallel to the gradients which enhances the resolution more than if the cells shrunk in a uniform manner. Computations were carried out without using the weighting functional as well. The primary shock structures were still recognized in this computation, but shock thicknesses were doubled. Also finer structures that appeared near the shocks were lost completely. In this particular problem, very little extra computation time was expended in running with an adaptive mesh for two reasons. The first reason is that the adaptive algorithm adds



Pressure
Contours

Fig. 14



Pressure
Contours

Fig. 15

little expense since few iterations were needed to update the mesh each time step. A more important factor was that the Courant condition for this problem was most limiting at the step corner. As a result both problems ran with almost the same number of time steps for a fixed interval of time.

CONCLUSIONS

To conclude, we summarize our results. We have shown that the variational formulation for generating meshes can easily be extended to three dimensions. Further the mesh generation equations behave in an easily predictable manner as illustrated with the three model problems given. We have also outlined a successful two-dimensional application of the mesh generator to a problem with moving multiple structures. The mesh generator moved the computation grid with the shock fronts and enhanced the resolution of the difference scheme significantly.

REFERENCES

[1] J. Brackbill and J. Saltzman, Adaptive Zoning for Singular Problems in Two-Dimensions, To be published in the Journal of Computational Physics.

[2] J. Saltzman, "A Variational Method for Generating Multidimensional Grids", Thesis, New York University, October 1981

[3] B. van Leer, Towards the Ultimate Conservative Difference Scheme V. A Second Order Sequel to Godunov's Method, Journal of Computational Physics 32, 1979, pp. 101-136

[4] S. Zalesak, Fully Multidimensional Flux Corrected Transport, NRL Memorandum, Report 3716.

AD P001009

Copyright 1982 by Elsev
NUMERICAL GRID GENERATION
Joe F. Thompson, editor

ORTHOGONAL COORDINATE MESHES WITH MANAGEABLE JACOBIAN

C. I. Christov
Dept. of Fluid Mechanics, Center of Mathematics and Mechanics,
Bulgarian Academy of Sciences, P.O. Box 373, Sofia 1090, Bulgaria

INTRODUCTION

Recently, the problem of numerical generating curvilinear coordinate meshes has received a vast exploration because of its outstanding importance in solving the partial differential equations of continuous mechanics. The major advantage of this method is that the boundary of the region becomes a coordinate line which decidedly simplifies the numerical schemes for approximate integration of boundary value problems. In some sence the method of adapted coordinates is an alternative to the method of finite elements.

In two dimension the most natural way to create curvilinear meshes was, perhaps, the inversion of conformal mapping[1,2]. This approach was generalized by means of variational principle[3]. The coordinates obtained in this way, however, were not orthogonal in general and the Jacobian assumed in some cases incomfortable values approaching zero or infinity. It was due to the rigid prescription of the boundary points. The orthogonality has been restored only after reducing the conditions on the boundary points to the natural ones for a conformal mapping[4,5]. In present note an other approach ensuring the Jacobian to be a priory prescribed function is attempted.

GOVERNING EQUATIONS

Consider a region D in the plane Oxy with boundary ∂D. The transformation

(1) $\quad x = x(\xi,\eta) \quad$ and $\quad y = y(\xi,\eta)$

relates D to a region D' of the plane $O\xi\eta$ (see fig.1). Respectively ξ = const and η = const represent two families of curves in Oxy which are desired to be orthogonal. Then

(2) $\quad x_\xi x_\eta + y_\xi y_\eta = 0$

where subscribing denotes a partial differentiation.

Since eq.(2) is not enough to define the two functions x and y

one more relation is needed. In present work it is chosen to impose a condition on Jacobian, namely

$$(3) \qquad x_\xi y_\eta - x_\eta y_\xi = f^{-1}(x,y) = F^{-1}(\xi,\eta)$$

where $f(x,y)$ is certain arbitrary function. Respectively $F(\xi,\eta) = f(x(\xi,\eta),y(\xi,\eta))$. In the case when $f(x,y)=1$ the coordinates obtained can be named "uniform".

The boundary conditions for the system (2),(3) are:

$$(4) \qquad \Phi(x,y) = 0 \quad \text{at} \quad (x,y)\in \delta D, \text{ i.e. at } (\xi,\eta)\in \delta D'$$

where $\Phi(x,y)$ is the analytical representation of the curves which comprise δD. For complete definiteness it is necessary to prescribe the rule of correspondence between the corners A,B,C,E and A',B', C',E' (see fig.1).

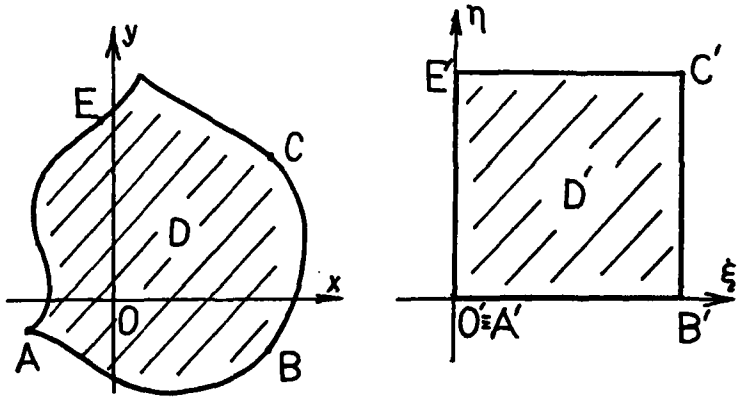


Fig.1. Geometry of the problem

The boundary value problem (2),(3) and (4) in its present form is very inconvenient for direct numerical integration. In addition its correctness is not obvious. However, (2) and (3) are readily transformed to

$$(5) \qquad f(x,y)x_\xi(x_\eta^2 + y_\eta^2) = y_\eta \quad \text{and} \quad f(x,y)y_\xi(x_\eta^2 + y_\eta^2) = -x_\eta$$

which is certain nonlinear generalization of the Cauchy-Riemann problem. This is which assures one that the problem is of elliptic type and could be expected to possess a soluti under the boundary conditions (4).

After obvious manipulations one finds

$$(6) \qquad f(x,y)(x_\xi^2 + y_\xi^2)(x_\eta^2 + y_\eta^2) = 1$$

and then eqs.(5) are easily transformed into following

$$(7) \qquad f(x,y)y_\eta(x_\xi^2 + y_\xi^2) = x_\xi \quad \text{and} \quad f(x,y)x_\eta(x_\xi^2 + y_\xi^2) = -y_\xi$$

The last set of equations is not independent of (5) and it is outlined only for further convenience.

Since sufficiently fast numerical method for direct integration of Cauchy-Riemann problem is not known it is convenient to render (5) and (7) into more standart form, namely:

$$(8) \qquad \frac{\partial}{\partial\xi}f(x,y)H_\eta^2\frac{\partial x}{\partial\xi} + \frac{\partial}{\partial\eta}f(x,y)H_\xi^2\frac{\partial x}{\partial\eta} = 0,$$

$$(9) \qquad \frac{\partial}{\partial\xi}f(x,y)H_\eta^2\frac{\partial y}{\partial\xi} + \frac{\partial}{\partial\eta}f(x,y)H_\xi^2\frac{\partial y}{\partial\eta} = 0,$$

where $H_\xi^2 = x_\xi^2 + y_\xi^2$ and $H_\eta^2 = x_\eta^2 + y_\eta^2$ and the squares of the coordinate scale factors. This system of two second-order differential equations is equivalent to a boundary value problem of Cauchy-Riemann type only when one of the first-order equations is satisfied at the boundary ∂D. In present work it is assumed to employ the orthogonality condition (2) as a boundary condition for (8),(9) along with (4).

METHOD OF SOLUTION

In order to solve the set of nonlinear elliptic equations (8), (9) with the boundary conditions (2),(4) the method of convergence[6] is chosen. Therefore derivatives of x and y with respect to some fictitious time t are added into (8) and (9) respectively,

$$(10) \qquad -\frac{\partial x}{\partial t} + \frac{\partial}{\partial\xi}fH_\eta^2\frac{\partial x}{\partial\xi} + \frac{\partial}{\partial\eta}fH_\xi^2\frac{\partial x}{\partial\eta} = 0,$$

$$(11) \qquad -\frac{\partial y}{\partial t} + \frac{\partial}{\partial\xi}fH_\eta^2\frac{\partial y}{\partial\xi} + \frac{\partial}{\partial\eta}fH_\xi^2\frac{\partial y}{\partial\eta} = 0.$$

The finite-difference scheme is constructed on the basis of the alternating-direction method[6]:

$$(12)\quad \frac{\bar{x}_{ij}-x^n_{ij}}{0.5\tau} = \Lambda^n_\xi \bar{x} + \Lambda^n_\eta x^n \quad , \quad \frac{\bar{y}_{ij}-y^n_{ij}}{0.5\tau} = \Lambda^n_\xi \bar{y} + \Lambda^n_\eta y^n \; ;$$

$$(13)\quad \frac{x^{n+1}_{ij}-\bar{x}_{ij}}{0.5\tau} = \Lambda^n_\xi \bar{x} + \Lambda^n_\eta x^{n+1} \quad , \quad \frac{y^{n+1}_{ij}-\bar{y}_{ij}}{0.5\tau} = \Lambda^n_\xi \bar{y} + \Lambda^n_\eta y^{n+1} \; .$$

Here $x_{ij}=x(\xi_i,\eta_j)$, $\xi_i=(i-1)h_\xi-0.5h_\xi$, $\eta_j=(j-1)h_\eta-0.5h_\eta$, where h_ξ and h_η are the uniform grid spacings in and directions respectively. The differential operators are approximated with second order of approximation as follows:

$$h^2_\xi \Lambda^n_\xi \varphi = (fH^2_\eta)^n_{i+\frac{1}{2},j}\,\varphi_{i+1,j} + (fH^2_\eta)^n_{i-\frac{1}{2},j}\,\varphi_{i-1,j} - \left[(fH^2_\eta)^n_{i+\frac{1}{2},j} + (fH^2_\eta)^n_{i-\frac{1}{2},j}\right]\varphi_{i,j}$$

$$h^2_\eta \Lambda^n_\eta \varphi = (fH^2_\xi)^n_{i,j+\frac{1}{2}}\,\varphi_{i,j+1} + (fH^2_\xi)^n_{i,j-\frac{1}{2}}\,\varphi_{i,j-1} - \left[(fH^2_\xi)^n_{i,j+\frac{1}{2}} + (fH^2_\xi)^n_{i,j-\frac{1}{2}}\right]\varphi_{i,j}$$

where $\varphi_{i,j}$ is either x or y.

The above scheme is completed with a second-order approximation of the boundary conditions (2) and (4):

$$(x^n_{1,j+1} - x^n_{1,j-1} + x^n_{2,j+1} - x^n_{2,j-1})(\bar{x}_{2,j} - \bar{x}_{1,j}) + (y^n_{1,j+1} - y^n_{1,j-1} + y^n_{2,j+1} - y^n_{2,j-1})(\bar{y}_{2,j} - \bar{y}_{1,j}) = 0 \; ,$$

(12^a)

$$\frac{\bar{x}_{1,j} + \bar{x}_{2,j}}{2}\frac{\partial F^n}{\partial x} + \frac{\bar{y}_{1,j} + \bar{y}_{2,j}}{2}\frac{\partial F^n}{\partial y} = -F^n + \frac{x^n_{1,j} + x^n_{2,j}}{2}\frac{\partial F^n}{\partial x} + \frac{y^n_{1,j} + y^n_{2,j}}{2}\frac{\partial F^n}{\partial y} \; .$$

$$(x^n_{M,i+1} - x^n_{M,i-1} + x^n_{M+1,i+1} - x^n_{M+1,j-1})(\bar{x}_{M+1,i} - \bar{x}_{M,j})$$

$$+ (y^n_{M,j+1} - y^n_{M,i-1} + y^n_{M+1,i+1} - v^n_{M+1,j-1})(\bar{y}_{M+1,j} - \bar{v}_{M,j}) = 0 \ ,$$

(12^b)

$$\frac{\bar{x}_{M,j} + \bar{x}_{M+1,j}}{2} \frac{\partial F^n}{\partial x} + \frac{\bar{y}_{M,j} + \bar{y}_{M+1,j}}{2} \frac{\partial F^n}{\partial v} = - F^n$$

$$+ \frac{x^n_{M,j} + x^n_{M+1,j}}{2} \frac{\partial F^n}{\partial x} + \frac{y^n_{M,j} + y^n_{M+1,j}}{2} \frac{\partial F^n}{\partial y} \ .$$

$$(x^n_{i+1,1} - x^n_{i-1,1} + x^n_{i+1,2} - x^n_{i-1,2})(x^{n+1}_{i,2} - x^{n+1}_{i,1})$$

$$+ (y^n_{i+1,1} - y^n_{i-1,1} + y^n_{i+1,2} - y^n_{i-1,2})(y^{n+1}_{i,2} - y^{n+1}_{i,1}) = 0 \ ,$$

(13^a)

$$\frac{x^{n+1}_{i,1} + x^{n+1}_{i,2}}{2} \frac{\partial F^n}{\partial x} + \frac{y^{n+1}_{i,1} + y^{n+1}_{i,2}}{2} \frac{\partial F^n}{\partial y} = - F^n$$

$$+ \frac{x^n_{i,1} + x^n_{i,2}}{2} \frac{\partial F^n}{\partial x} + \frac{y^n_{i,1} + y^n_{i,2}}{2} \frac{\partial F^n}{\partial y} \ .$$

$$(x^n_{i+1,N} - x^n_{i-1,N} + x^n_{i+1,N+1} - x^n_{i-1,N+1})(x^{n+1}_{i,N+1} - x^{n+1}_{i,N})$$

$$+ (y^n_{i+1,N} - y^n_{i-1,N} + y^n_{i+1,N+1} - y^n_{i-1,N+1})(y^{n+1}_{i,N+1} - y^{n+1}_{i,N}) = 0 \ ,$$

(13^b)

$$\frac{x^{n+1}_{i,N} + x^{n+1}_{i,N+1}}{2} \frac{\partial F^n}{\partial x} + \frac{y^{n+1}_{i,N} + y^{n+1}_{i,N+1}}{2} \frac{\partial F^n}{\partial y} = - F^n$$

$$+ \frac{x^n_{i,N} + x^n_{i,N+1}}{2} \frac{\partial F^n}{\partial x} + \frac{y^n_{i,N} + y^n_{i,N+1}}{2} \frac{\partial F^n}{\partial y} \ .$$

The major significance of the scheme proposed here is that the functions x and y are being computed together as a vector at each fractional step. This appear to be crucial because of the struc-

ture of the boundary conditions containing both the functions. In fact, on each half-time step an algebraic system of following type is solved:

$$(14) \quad A\begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} - C\begin{pmatrix} x_i \\ y_i \end{pmatrix} + B\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} f_i \\ \eta_i \end{pmatrix} .$$

Here A,B,C are matrices 2x2. It is easily proven that they satisfy the sufficient conditions for stability of the Gaussian-elimination -type method proposed[7] for system with the above structure.

RESULTS AND DISCUSSION

The method outlined here can be used in two major ways. The first is the construction of regular orthogonal meshes for domains with curved boundaries. The simpliest and, perhaps, the most natural definition of regularity is the requirement Jacobian to be equal to unity, i.e. f=1. The meshes related to this condition can be informally named "orthonormal".

First of all, it has been checked out wether cartesian coordinates appear to be among these uniform coordinates. Indeed, when the above boundary value problem has been solved for the case when the region D is simply the unit square, then the mesh obtained has resulted into cartesian set with very good accuracy.

The next test has been to generate an uniform orthogonal mesh for the curvilinear rectangular domain shown on fig.2. It is specially selected to possess right angles in order to avoid some difficulties connected with the breaking of conformity at the corner points when the angles are not right. It is not a restriction in general, but the case with arbitrary angles requires a special care when spacing the grid points near the corners. The latter goes beyond The frame of present short note. In addition three straight rims are selected and only one curved boundary is allowed according to the formula $y=1.5 - .5\cos(\pi x)$. This is fully enough to display the method. The grid is chosen to be uniformly spaced in both directions with space steps h_ξ and h_η respectively. In order to obtain higher accuracy as well as to check and verify the computations two different meshes are employed with number of grid cells 21x21 and 41x41 respectively. Results turn out to vary slightly with the reduce of the grid size. On the basis of the two solutions

and by means of Richardson extrapolation a solution with order of approximation $O(h_\xi^4+h_\eta^4)$ on the mesh 21x21 is constructed. On fig.2 are shown several characteristic coordinate lines of this solution, while on fig.3 are plotted the coordinates obtained simply from the conformal mapping. It is well seen the spoiled behaviour of the Jacobian in the last case as well as the significant improving attained after the method of the present work is applied.

The second way of application of the proposed method is in construction of optimal meshes. One of the possible usefull definitions of optimality is to seek for a mesh which is more dense in the regions where the profile of certain given function is more steep. A similar idea was employed[8] but on the basis of a variational principle. In present work the instrument for governing the mesh appears to be function f. It is assumed that f(x,y) is nothing but the two-dimensional slope of a given surface

$$(15) \quad f(x,y) = \sqrt{1 + u'^2_x + u'^2_y}$$

where $z = u(x,y)$ is the equation of that surface. Eq.(3) obviously yields:

$$dxdy = (1 + u'^2_x + u'^2_y)^{-\frac{1}{2}} d\xi d\eta .$$

Tha latter relation asserts that if one tekes a regularly spaced grid in the region D' one obtains coordinate lines in D which are more dense in the regions where the two-dimensional slope of function u(x,y) is greater, i.e. where function u(x,y) is steeper.

To avoid the unnecessary complications and to demonstrate the idea of optimality in its pure form it is considered here a square domain in the plane Oxy. It should be mentioned that several different "leading functions" $u(x,y)$ has been used in calculations. To give a better feeling of the results the simplest function $u(x,y) = 1 + x^2 + y^2$ has been chosen among others to expose the method. The optimal mesh obtained with this function is shown on fig.4. The shape of function u along with the coordinate mesh is plotted on fig.5. There can be seen the uniform portions in which the surface area is divided by the coordinate lines.

In the end it should be mentioned that the rate of convergence of the method for the optimal meshes has been much greater than that for the uniform ones with curved boundaries.

GENERALIZATION FOR THREE DIMENSIONS

It is important to note that the present method is easily generalized in three dimensions, i.e. when three cartesian coordinates x,yz are transformed to three curvilinear coordinates ξ,η,ζ. In this case there exist three conditions of orthogonality:

$$x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta = 0 ,$$

$$x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0 , \tag{16}$$

$$x_\zeta x_\xi + y_\zeta y_\xi + z_\zeta z_\xi = 0 ,$$

but only two of them are independent. The condition on Jacobian is

$$x_\xi y_\eta z_\zeta + x_\eta y_\zeta z_\xi + x_\zeta y_\xi z_\eta - x_\zeta y_\eta z_\xi - x_\eta y_\xi z_\zeta - x_\xi y_\zeta z_\eta = f^{-1} . \tag{17}$$

Once again, it is easy to show that

$$f^2(x,y,z)\, H_\xi^2\, H_\eta^2\, H_\zeta^2 = 1 \tag{18}$$

and to derive the following set of equations

$$\frac{\partial}{\partial\xi} f H_\eta^2 H_\zeta^2 \frac{\partial\varphi}{\partial\xi} + \frac{\partial}{\partial\eta} f H_\zeta^2 H_\xi^2 \frac{\partial\varphi}{\partial\eta} + \frac{\partial}{\partial\zeta} f H_\xi^2 H_\eta^2 \frac{\partial\varphi}{\partial\zeta} = 0 , \tag{19}$$

where φ is either x,y or z.

The boundary conditions for this system of equations are the equation of boundary surface:

$$F(x,y,z) = 0 \text{ at } (x,y,z) \in \delta D \tag{20}$$

on one hand and two of the othogonality conditions (16) on the other. At each side of the unit cube in D' have to be chosen those two of (16) which are normal to this side.

CONCLUSIONS

In present work a method for constructing orthogonal coordinates when their Jacobian satisfies certain condition is outlined. The respective equation of this condition along with the orthogonality condition yield a system of equations which is a nonlinear analog to the Cauchy-Riemann problem. The role of a boundary condition is played by the equation of the boundary. This system is rendered to a pair of coupled second-order elliptic equations which is solved by means of a kind of splitting method.

Two general ways of application are displayed: generation of "uniform" meshes with unit Jacobian, and "optimal" meshes for which the Jacobian is governed by the magnitude of the two-dimensional slope of a given function. The latter assures one that the mesh is more dense in the regions where computed function is steeper.

Generalization of the method for the case with three dimensions is sketched.

REFERENCES

1. Godunov, S.K. and Prokopov, G.P. (1967) Journal of Num. Math. and Math. Phys. (Moscow), 7, 1032-1059.
2. Thomson, Joe F., Thames, F.C. and Mastin, C.W. (1974) J. Comp. Phys., 15, 299-319.
3. Belinsky, P.P., Godunov, S.K., Ivanov, Yu.B. and Yanenko, I.K. (1975) Journal of Num. Math. and Math. Phys.(Moscow), 15, 1499-1511.
4. Barfield, W.D. (1970) J. Comp. Phys., 5, 23-33.
5. Prokopov, G.P. (1970) Reprint No 45 of Inst. of Appl. Math. of Acad. of Sci. of USSR, Moscow.
6. Yanenko, N.N. (1971) The method of fractional steps, McGraw-Hill.
7. Samarsky, A.A. and Nikolaev, E.S. (1978) Methods for solving difference equations, Nauka, Moscow.
8. Yanenko, N.N., Danaev, N.T. and Liseykin, V.D. (1977) Numerical Methods of Continuous Mechanics (Novosibirsk), 8(4), 157-163.

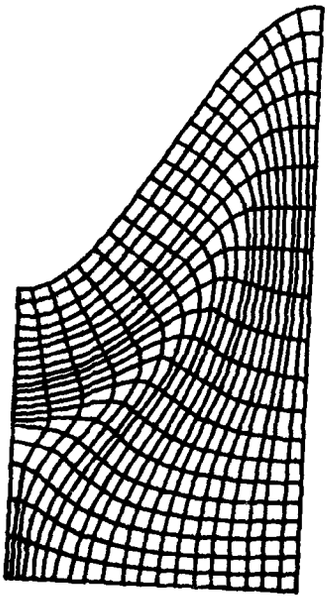


Fig. 2.

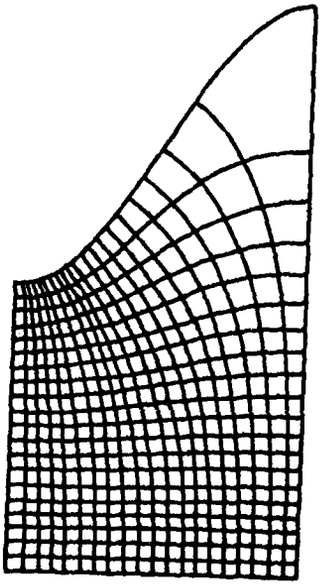


Fig. 3.

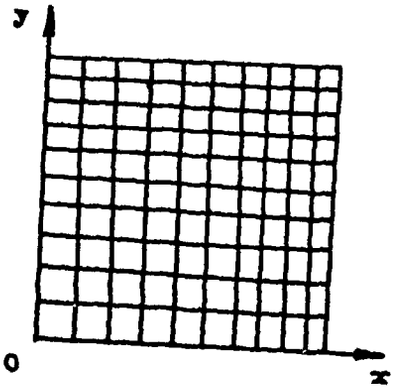


Fig. 4.

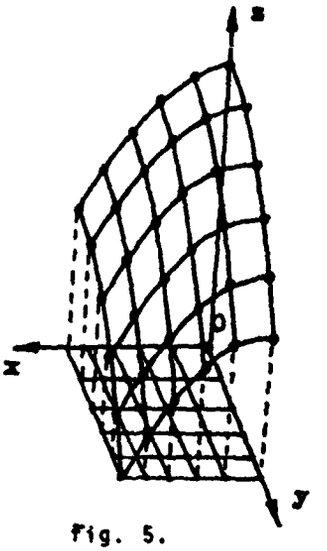


Fig. 5.

INDEX

A

ADI scheme, 102, 241, 304, 346-7, 690, 767, 809-10, 815, 818, 860, 887
Acceleration parameters, 102, 270
Accuracy, 31, 37, 194, 221, 254, 277, 283-8, 295, 302-5, 343, 348, 486, 496, 504, 731, 767, 787, 861, 890
Adaptive grids, 29, 41, 53, 61, 194, 277, 317, 339, 362, 787, 819, 837-842, 848, 851, 865
Advection-diffusion, 847, 851
Aerodynamics, 296
Aircraft, 235, 242, 471, 486
Airfoil, 113-129, 236, 252, 303-5, 358-362, 375, 451, 457-461, 508, 549, 565-581, 598, 653, 667, 677, 684-5, 704, 742-9, 752-7, 776, 781-6, 796-7, 802
Algebraic grid generation, 96, 107-8, 112, 131, 137, 173, 361, 447, 671, 855
Analytic and numerical evaluations, 195
Analytic continuity, 483
Angle of intersection, 82, 100, 104, 665-8, 765, 769
Annular mapping, 597, 598
Annulus, 374
Approximate-factorization, 295, 304
Arc length, 19-21, 85-7, 95-8, 112, 157-63, 198, 204-6, 219, 232, 238-40, 326, 339, 342, 365, 369, 374, 452, 456, 530, 605, 610, 672, 675, 678, 692, 701, 705, 731
Area, 19
Artificial compressiblity, 811-2, 817
Artificial corner, 376, 384
Artificial viscosity, 775, 811, 824
Aspect ratio, 19, 103, 198, 219, 220, 229, 231, 555, 659, 729, 787, 790, 803
Astronomical map, 113
Attraction, 90, 93, 103, 254, 399, 401, 428, 634, 690
Auxiliary constraints, 176-7

B

B-spline, 229, 465
Base vectors, 19, 23, 68
Basis functions, 477
Beltrami equations, 41, 47, 54, 74, 718, 721, 861
Bidirectional interpolation, 157-8
Biharmonic equation, 260-1, 761
Bilinear transfinite mapping, 111, 117, 174-7, 185-6
Biquadratically blended interpolant, 176
Bivariate transfinite mapping, 177
Blending functions, 111, 117, 139, 146-9, 152-4, 171, 174-7, 159-60, 176, 162-4, 168, 185-6
Block boundaries, 236-46
Block-line SOR, 347
Block-structured, 3, 12-3, 79, 235-8, 469, 611, 658
Body/wake, 829
BODYFIT, 625, 629-32
BODYFIT-1FE, 621
Boolean sum projection, 174-5, 179-80, 210, 217, 448
Boundary charges, 609, 612-3
Boundary conditions, 261-2, 272, 298, 306, 392, 502

Boundary integral-equation technique, 596
Boundary layer, 37, 287, 358-9, 376-7, 519-21, 564, 829, 830, 832
Boundary line string, 468
Boundary movement, 385
Boundary point correspondence, 127
Boundary point distribution, 82, 96, 99, 101
Boundary point speeds, 326
Boundary potentials, 602-7, 612, 614
Boundary slope discontinuity, 6, 80, 232
Box mapping, 645-50
Branch cuts, 5, 8-10, 12-16, 54, 79, 92-3, 95, 101, 104, 116, 123, 257, 358, 376, 381, 410, 416, 549, 553-4
Breaking wave, 400-403
Brown's square, 211
Burger's equation, 322-3, 326-7, 850

C

C-type, 9, 11, 15, 92, 103, 166-7, 305, 314, 358-62, 735, 381, 549-53, 560, 563-4, 569-73, 580, 638, 640, 645, 650, 656-9, 796-7
CPU time, 130, 349-50, 517, 519, 572, 585, 588, 663, 677, 695, 706, 716, 817
Canonical, 118
Canonical contour, 111, 123, 131
Canonical domain, 120, 122
Cardinality conditions, 173-4
Carpets, 483-5
Cascade, 113, 166-7, 123, 129, 139, 303-5, 314-5, 357-62, 381, 459-61, 549-59, 563, 569-70, 578, 580, 638, 643-5, 649
Cauchy-Riemann conditions, 74, 229-32, 526, 533, 886-7, 892
Cell size, 283, 653, 845
Cell string, 468
Cell volume, 231-2
Chain rule, 31, 65, 108, 141, 212, 225, 287, 298, 301, 735, 792, 795
Change in grid spacing, 33, 39
Charge distribution, 602-16
Chebyshev, 496, 498, 504
Christoffel symbols, 44, 67-8, 90, 719, 793
Circle preserving mappings, 126
Circular cylinder, 37
Clustering, 34, 193-6, 202, 207, 229, 232, 295, 303-4, 323, 368-72, 375, 401, 452-4, 457-9, 563, 566, 573, 581, 658, 660, 769, 783, 810, 842, 854, 860-1
Collapsed edge, 236
Combustion, 339-40, 348
Commutativity, 211
Compact stencil, 304
Compatibility requirements, 47, 69
Complex potential, 124, 509-10, 590-1, 595, 602-3, 608, 614
Complex transformation, 554, 560
Component adaptive grid interfacing, 654
Composite grids, 193-4, 447-9, 455, 667, 75-6, 682, 769
Compressible flow, 297, 303, 359, 525, 529, 537, 585
Compressor, 116
Computational fluid dynamics, 306, 437, 518, 742, 775
Computer time, 123, 127, 333, 482, 521, 779, 829
Concave, 14, 81, 231
Concentration, 29, 41, 51, 53, 110, 164, 346, 368, 417, 521, 529, 581, 706, 719, 722, 724, 819-20, 827, 829, 830, 832, 880

Configurations, 1, 2, 4, 15, 17, 29, 103, 104, 194, 633-6, 638, 645
Conformal mapping, 50, 64, 66, 72, 107, 172, 193-4, 198-9, 206-7, 228, 230-1, 280, 359, 361, 410, 415-6, 419, 481-2, 508, 519, 525-6, 528-31, 536-7, 554, 563-4, 565, 569, 570-2, 578, 579, 585, 601, 885, 891
Conformal metric, 789, 790
Conical surface, 112-3
Conjugate gradient, 504, 761, 767-8
Connectivity, 442
Conservative, 22-7, 35, 195, 202, 284, 289, 295-6, 306, 419, 426, 485, 493, 525-6, 530, 735, 787, 792, 794, 814, 819-22, 825, 879
Continuity, 236, 484-5, 653, 663, 847
Contraction, 38, 58, 73, 723, 725, 726, 727
Contravariant, 20-3, 43, 76, 298, 526-7, 792
Control, 79, 144, 164, 168, 194, 196, 204, 207, 210-1, 229, 235-43, 277, 295, 324-329, 357-62, 406, 426, 447, 529, 586, 594, 596, 619-20, 633, 636, 653, 655-8, 667-72, 678, 687-93, 700, 731, 761-5, 778, 781,827, 865, 896
Control functions, 81-5, 90, 95-104, 137, 154, 156, 159-63, 168, 700, 705-7, 714-5
Control parameters, 671-6
Control points, 660-1
Control surfaces, 139, 447
Control volumes, 621, 822
Convection diffusion equation, 283
Convective transport, 285
Convex, 19, 81, 102, 124, 376, 724
Coordinate redistribution function, 721
Corner, 6, 14, 103, 178, 180, 236, 266, 273, 376, 587-8, 596, 598, 610-1, 616, 692, 765, 769, 826, 860, 890
Corner-removing mappings, 587, 589, 597
Cost, 195, 241, 349, 607, 819
Covariant, 20-1, 42-3, 76, 791
Crack, 184
Critical points, 603-9, 612-5
Cross derivative, 62, 194, 241, 733
Cryptic language, 465, 472
Cube cluster, 465, 467, 469-71
Curl, 21, 26
Curvature, 33, 49, 67, 73, 81, 86-9, 97-9, 119, 124, 199, 200-2, 216-7, 288, 290, 294, 345, 364, 440, 452, 485, 554, 573, 668, 670-3, 678, 697, 701, 721, 732-3, 747, 788, 807, 839-42, 859
Curvature clustering, 448
Curvature constraint, 222, 225-31
Curvature tensor, 41, 66
Curve definition, 450

D

Darboux's Theorem, 217
Defect correction, 503-4
Deferred correction, 733-5
Deflection contours, 270-5
Deformation field, 839-41, 855
Deformation moment, 838-40
Depth-averaged, 412
Derivative correspondance across cuts, 16
Derivatives, 29, 31, 37, 137, 143, 167, 169, 480, 482, 486-7, 611, 795

DERMOD, 487
Difference approximations, 31-6, 101, 221, 300, 304-5, 344, 367, 423, 479, 490, 494, 502, 867, 879
Difference metric, 220
Difference orthogonality, 202, 222, 454
Differencing metrics, 301, 794
Differential equations, 410
Differential geometry, 29
Differential models, 41
Diffuser, 357, 360, 362, 372, 374, 381, 520-2
Discontinuities, 103, 493, 595, 645, 700, 704
Discretized transfinite mappings, 177
Distortion, 37, 176, 326-9, 376, 480, 705
Distribution, 79, 288, 441, 448, 462, 529, 659, 667, 669, 670, 704-5
Distributive lattice, 179
Divergence, 21, 25
Divergence form, 35, 296, 820
Divergence Theorem, 21
Donor cell differencing, 289
Douglas-Gunn splitting, 815
Droplet combustion, 340
Ducts, 129, 360-4, 376-7, 507-10, 521, 527-8, 531, 543, 563, 565-6
Dynamic grid, 813, 837, 844-5
Dynamism, 854
Dynamism constraint, 845, 850, 857

E

Edges, 97, 178, 180, 675
Efficiency, 304, 348, 553, 818
Elastic torsion, 253
Elasticity, 254, 258, 275
Electric field equations, 729-33
Electrode, 731, 734
Electrostatic analogy, 550, 875
Ellipse, 274, 455, 457
Ellipsoid, 201, 217, 676
Elliptic, 6, 9-12, 79, 107-8, 117, 195, 206-7, 211, 230, 241, 320, 359-64, 410, 416-7, 619, 653, 667, 688, 695, 717, 727, 729, 739, 761-3, 768, 775-8, 781, 848, 865, 868, 887, 892
Embedded, 10-1, 194, 447, 453
Engine, 465
Equidistant mesh, 859
Error, 31-2, 37, 103, 144, 220, 253, 266-8, 272-4, 277, 282, 284, 292, 299, 300, 305, 320, 323-5, 334, 340, 395, 502-3, 528-9, 610-1, 768, 787, 792-4, 804, 806, 824, 842, 853
Estuarine hydrodynamics, 409
Euler angles, 788
Euler characteristic, 470
Euler derivative, 868
Euler equations, 86, 279, 280, 282, 287, 290, 305, 328, 332, 333, 482, 486
Euler Time-Stepping, 465
Execution time, 542
Expansion of grid spacing, 34, 38
Exponential, 34
External flow, 295, 361, 508, 529, 531
Extremum principles, 80

F

Faces, 180, 671, 673, 768
Factorization, 347-8, 437, 697
Fan-in wing, 472
Fast Fourier techniques, 118-21, 125, 127, 130, 589
Fast Poisson solver, 128, 130, 131
Fictitious corner, 236-7, 243, 246, 376
Fictitious points, 766
Field boundaries, 236
Filtering, 394-5, 827
Finite element, 127, 243, 385, 410, 441-2, 488, 495
Finite volume, 110, 127, 131, 243, 295, 303, 547, 553-4, 668, 819
First fundamental form, 46
Fisher's equation, 286, 289, 293
Five-sided cells, 645
Flame, 340, 343-4, 348-56
Flap, 659-62
Flooding boundaries, 428
Flux corrected transport, 286, 423, 879
Fluxes, 202, 284-5, 304, 493
Forcing function, 53, 363-75, 687-94, 848
Forgiving algorithms, 242-3, 250-2
Four-color, 393
Fourier transform, 499
Free shear layer, 820
Free stream correction, 301
Free stream metric variation, 302
Free surface, 385-95, 401-6, 645, 651, 809-18
Fronts, 346
Full-potential, 656, 659, 663-4, 787, 794-5
Fundamental Differential Equation, 211, 213, 221
Fuselage, 236, 239, 529, 573, 676, 724, 870, 875
Fuselage-wing, 783

G

GEM Codes, 729
GIM code, 139
GRID 3C, 563, 572-3, 582-3
Garrick transformation, 122
Gas dynamic calculations, 859
Gauss Equations, 41, 45, 496, 717
Gauss-Jacobi iteration, 868
Gauss-Seidel iteration, 419, 761, 767-8
Gaussian curvature, 199, 200, 216, 225
Gaussian-elimination, 890
Gaussian surfaces, 89
Generalized coordinates, 172-3
Geodesic curvature, 216
Gradient, 21, 25, 109, 254, 277, 282, 285-7, 290, 303-6, 320, 326, 332, 339, 342-5, 353, 356, 564, 659, 747, 819, 825-30, 841, 867, 880
Gradient length, 292, 880
GRAPE, 11, 101, 633, 653-9, 663
Graphics, 131, 137, 168, 169, 447, 455
Grid changes, 296
Grid distortion, 315

Grid distribution, 689
Grid motion, 832
Grid refinement, 437-45, 729, 861
Grid resolution, 520, 573
Grid spacing, 31, 34, 38-9
Grid speeds, 318-34, 849

H

H-type, 399, 549, 553-60, 564, 659, 691-4, 783-4
Halo corners, 484
Halo width, 482
Halo zone, 483
Handles, 471
Harmonic functions, 63, 611, 740, 744-7
Harmonic mean, 493
Heat and mass transfer, 303, 339, 357, 81, 531, 619
Helmholty equation, 243, 252
Hermite blending functions, 176
Hermite interpolation, 139, 209-10, 438, 502
Hermite projector, 209-10, 217
Hinge point transformations, 120
Homotopic mappings, 138
Homotopy argument, 603
Hosted algorithm, 236-7, 242-3, 730
Hybrid grids, 598-9
Hybrid iteration, 767-8
Hydrodynamics, 409-10, 415, 422, 878
Hydrofoil, 645, 651
Hyperbolic, 79, 213, 217, 231-2, 303, 362, 775
Hypersonic scramjet flows, 360

I

Impeller, 113
Incompressible flow, 110, 124, 303
Infinity, 125, 564, 568, 570
Inhomogeneous terms, 81, 363, 655
Initial guess, 102, 368, 370, 373, 382-3, 635, 730
Inlet, 115-6, 120, 122, 127-9, 303, 360-2, 376, 472, 518-20, 755-6
INMESH, 633-8, 645, 651
Instability, 168, 286, 301, 344, 394, 817
Integral-equation, 585
Integral form, 820
Integrals, 23, 29
Interactive, 168-9, 437, 636
Interactive graphics, 171, 362
Interfaces, 176-7, 305, 469
Internal flows, 357, 507, 563
Interpolation, 97-9, 102, 130, 137-9, 146-7, 151-4, 160, 168, 173-5,178-80,208-11, 217, 220, 240-1, 284, 344, 354, 358, 363, 370-1, 377, 437-40, 448, 459, 477, 482, 487-8, 490-9, 503-4, 519, 565, 567, 670-5, 702-6, 768-9, 777, 828, 846, 850, 855, 857, 866
Intersection angle, 101
Invariant mapping, 842-5
Invariant measure field, 838
Inviscid flow, 110, 297, 312-3, 525

Inviscid shear flows, 302
Irregularities, 242-3
Isentropic, 227
Isentropic elastic plates, 253
Islands, 409, 419, 428
Isometric grid, 861
Isomorphism, 179
Isoparametric, 441, 564
Isothermic coordinates, 49, 55-6, 64, 71, 75-6, 198
Iterative solutions, 101-3
Jacobian, 19-20, 36, 108, 142-3, 199, 212-3, 231, 240-1, 278, 297, 301, 319, 323, 326, 376, 418, 496, 504, 526-34, 669, 674, 699, 701, 732-3, 787, 814, 844, 846, 885
Jacobian damping, 324, 329
Jacobian factorizations, 347
Jacobi iteration, 282
Joukowski transformation, 115
Juncture, 675

K

Karman-Trefftz transformation, 115, 117-8, 121, 587-8, 596
Keyseat, 269-70
Kinks, 706

L

L-shaped, 5, 6, 167
LU decomposition, 347
Lagrange interpolation, 488, 490-5
Lagrange multipliers, 787, 806
Lagrangean phase, 879
Lame's equation, 70
Laplace, 21, 26, 33, 37, 41, 49, 50, 58-66, 76, 80-3, 86, 128, 172, 219, 228, 242, 250, 254, 365, 388, 401, 495, 499, 530, 591, 596, 602, 634, 636, 699, 744, 749, 787, 791-3, 861, 865
Lasers, 729-30, 733
Laterally averaged, 410
Lax-Wendroff, 525
Least-squares, 490-4
Length element, 43
Line integral, 24, 28, 68
Linear equation solvers, 130
Linear grid generation equations, 241
Linear lofting, 175
Little's square, 211
Lobed mixers, 361
Local coordinate system, 15
Lofting, 178
Logarithmic transformation, 117 123, 592
Lost corners, 246
Lubrication, 113

M

MACSYMA, 735
MacCormack, 304, 323, 333, 525, 825
Mach reflections, 525
Magnetic reconnection, 288
Magnetosphere, 294
Mainardi-Codazzi equations, 47
Mapping modulus, 109-11
Marching scheme, 195-6, 222, 231, 303, 729, 775
Marker-and-cell, 385
Mass-residue, 622-4
Matrix splittings, 347
Matrix techniques, 130
Maximum principle, 362, 765
Median faces, 469
Median line, 467-8
Median point, 467
Median surface, 469-70, 474
Membrane analogy, 254-5
Memory, 110, 445, 542, 572, 776
Mercator projections, 113
Mesh density, 550, 554
Mesh intervals, 343-6
Mesh refinement, 345, 350, 806
Mesh-sensitive, 295
Method of characteristics, 529
Metric, 19, 109, 194, 196, 198-204, 224-232, 304, 306, 341, 353, 496-7, 796, 837, 844
Metric coefficients, 25, 41, 43, 50-1, 55-7, 66-8, 71, 75-6, 212-8, 222, 224, 295, 298-9, 359, 508, 511, 517-8, 529, 536, 670, 719, 820, 825, 845-7, 855, 857
Metric conservation law, 300
Metric constraint, 225
Metric differencing, 299
Metric equations, 217, 224
Metric error, 301
Metric identities, 300
Metric relations, 301
Metric rule on coordinate changes, 226
Metric specification, 226
Metric tensor, 20, 43, 76, 530, 787-95, 801-2, 846
Minervo approximation, 495
Minicomputer, 437
Mixed derivatives, 211, 225, 230
Modulus of the domain, 722
Moving aileron, 303
Moving boundaries, 275, 288
Moving control volume, 289
Moving grid, 28, 286, 320
Multi-block grid, 235-7, 243-6
Multi-body problems, 636-8
Multi-channel configurations, 360
Multielement airfoils, 585-6, 653, 659-63, 749
Multi-grid technique, 37, 304, 360, 465-8, 504, 564, 706, 732
Multi-material calculations, 290
Multiple block structures, 688
Multiple bodies, 122, 123, 129, 166, 303, 410, 416, 639

Multiple-connectedness, 633
Multiple grid systems, 306
Multiple valuedness, 107
Multiply connected regions, 601, 675
Multisurface method, 137-9, 150, 208-11, 447-8, 453-4, 462
Multivalued nature, 113-4
Multi-Volume Data Structure, 465

N

Nacelle, 115, 120, 305
Natural tangents, 197, 202, 204, 214, 220
Navier-Stokes, 68, 305, 411, 514, 621, 623, 656, 659, 697, 731, 811, 819
Near-circle, 116-7, 121-2, 130-1
Nearly-orthogonal mesh, 128, 131, 304, 357-62, 368, 372, 550, 555, 655, 657
Neumann boundary conditions, 4, 14, 368, 699, 741
Newton iteration, 257, 344, 347-8, 367, 372, 570, 572, 591, 607, 611
Node ordering, 443
Noise, 490
Non-conservative, 22-7
Nonorthogonality, 36, 61, 77, 193, 199, 426, 573, 579, 789
Normal, 19, 23, 358
Normal derivative, 23, 27
Normal derivative boundary condition, 766
Nozzles, 361-2, 517, 529
Nuclear reactors, 619, 625
Numerical diffusion, 277, 282, 285-90
Numerical instabilities, 368
Numerical viscosity, 286
Numerically orthogonal, 195
NUMESH Code, 11

O

O-type, 9, 92, 166, 359, 549-50, 553, 560, 564-5, 576, 592-3, 636-9, 748, 783-4
Ocean, 426, 428
One-to-one mapping, 2, 80, 362
Optimal mesh, 343, 891
Orientation, 787-91, 798
Orthogonal, 55, 97-9, 107-12, 156, 172, 267-8, 273, 357-61, 391, 410, 415-6, 419, 453, 457, 459, 462, 494, 507-9, 523, 550, 569, 671-4, 741, 837-8, 860-1
Orthogonal grid generation, 19, 49, 63-6, 70, 77, 124-31, 193, 279, 362, 459, 481, 789, 853, 885
Orthogonal metric, 194, 198, 216, 228
Orthogonal surface coordinates, 201
Orthogonal trajectories, 195-6, 204, 222-3, 231, 454
Orthogonality, 31, 65, 68, 73, 86, 88, 139, 156, 193, 212, 217, 222, 278, 280, 296, 360-4, 367-8, 371-6, 437, 457, 533, 571, 671, 678, 702, 765, 782, 787-90, 803, 807, 854, 860, 868, 885, 887, 892
Orthogonality at the boundary, 4
Orthogonality constraint, 846, 849
Orthogonality control, 781, 783
Orthogonality functional, 867, 869-72, 874
Orthogonality parameter, 804
Orthonormal, 890
Oscillations, 38, 305, 323, 330-3, 352-3, 394, 622, 731, 733
Outer surface method, 147

Overlapped grid, 102, 176, 187, 305, 315, 635, 826
Overlapping zones, 483
Overrelaxation, 102

P

Packaged tools, 130
Padé approximation, 499
Panel method, 124, 242, 243, 250-1, 385, 465, 596, 599, 742, 751
Parabolic equations, 775, 810
Parabolic interpolant, 493
Parabolic transformation, 239
Parabolized Navier Stokes, 303, 311, 529
Paraboloids, 217
Parametric transformation, 239
Particle-in-cell method, 290
Partition boundary, 466
Partitioned fields, 465
Patched coordinate systems, 235, 305-6, 447, 450, 482
Penalty method, 279
Perturbation form, 302
Pitching moment, 316
Planetary probe, 829
Plates, 13, 253-65, 272-5
Poincare-Lemma, 225
Point density, 588, 593-4
Point distribution, 98-9, 163, 593-4, 600, 659-63, 706, 830
Point method, 144
Point relaxation, 767
Point-SOR, 722
Poisson equations, 76, 81, 83, 207, 211, 260-4, 272, 350, 363-4, 390, 395, 417, 530, 633-4, 655, 671, 687, 796, 813
Poles, 124, 517
Polynomial interpolation, 487, 499, 777
Potential, 219, 243, 388, 603-16
Potential flow, 37, 124, 243, 301, 405, 465, 481-2, 509-11, 515, 547, 550, 553, 586, 590-5, 645
Potential flow streamlines, 507
Prandtl-Meyer expansions, 525
Precision sets, 179-80
Primitive function, 171, 173, 178
Principal curvature, 678
Principal normal, 87
Principle of reflection, 114
Product projection, 174-5
Projectiles, 303
Projector, 173-80, 206, 209-10, 239, 448, 671, 674, 861
Prolate ellipsoid, 56, 58, 724-7
Propeller, 547, 563, 573, 576, 582-3
Pseudo-spectral method, 499, 504

Q

Quadratic approximations, 493
Quadralateral, 610-11
Quality, 237, 239, 242, 304, 441, 787
Quasi-circular, 587

Quasiconformal mapping, 74

R

Rate-of-change of the grid spacing, 31
Rayleigh-Taylor instability, 288
Reaction-diffusion equations, 340
Reclustering, 661
Rectangular solids, 13
Red-black, 393
Redistribution, 51, 75, 722
Re-entrant boundary, 5, 8, 10, 15-6, 34, 79, 107
Reentry body, 529
Reference material, 128
Refinement, 204, 454, 564, 859, 861
Reflection mapping, 116
Reflection principle, 126
Remap phase, 879-80
Resolution, 398, 677, 798
Review, 128
Rezoning, 819, 825, 827, 832
Richardson extrapolation, 891
Riemann-Christoffel curvature tensor, 69
Riemann sheet, 113-4, 123, 563, 595
Riemann Tensor, 41, 66, 69
Riemannian metric, 43
Rivers, 409-10, 419, 426-8
Rod-bundle, 621, 625
Root selection, 115-7, 131
Rotor, 564-6
Rounding errors, 268, 868, 870
Runge-Kutta, 534-6

S

SCM method, 327-8
SLOR-iteration, 102, 304, 716
SOR iteration, 253-4, 257, 260, 265, 270, 272, 345, 367
Salinity, 411, 413, 423, 436
Schwartz-Christoffel techniques, 117, 119, 207, 361, 507, 525, 531-4, 544, 607
Scramjet engine, 361
Secant method, 607
Second fundamental form, 46
Segmented grids, 596-7, 633-8, 645, 651, 761
Sequence of surfaces, 89
Sequential mappings, 117
Semidirect, 729
Shafts, 253-4, 257-9, 266-73
Shear layers, 829-30, 832, 835
Shear stresses, 258-60, 267-71, 359, 842
Sheared conformal approach, 206
Shearing transformation, 36, 122, 205-10, 214, 217, 359, 361, 549, 554-5, 560, 563-6, 570-1, 577, 579, 585
Sheet, 16, 92
Ship, 402-4
Ship performance, 386
Ship wave, 386,406

Shock, 243, 284-5, 287, 296, 303-4, 327, 329, 331, 358, 467, 482, 525, 531, 564, 749, 796, 804, 826, 829-33, 861, 878, 880, 883
Shock aligning, 330, 334
Shock capturing, 301, 330, 334
Shroud, 659, 662
Side matching table, 442-3
Single-fours technique, 376, 384
Singular regions, 698
Singular surfaces, 698
Singularities, 124-5, 167-8, 212, 223, 229, 246, 266, 376, 395-6, 398, 465-7, 471, 476-7, 482-6, 508, 549, 553, 585, 594-6, 609-10, 677, 697, 705, 741, 775
Singularity method, 739
Six-sided cells, 395
Skewness, 253, 266-8, 272-5, 295, 304-6, 314, 363-4, 550, 653, 661, 824
Slabs, 8-10, 13
Slat, 757
Slit, 7-11, 13-4, 376, 384, 570, 613, 638, 641-4, 732-4
Slit-plane, 697
Slope changes, 124
Slope continuity, 656, 659
Slope discontinuities, 6-7, 104, 121-2, 176, 242, 447, 459
Slope interpolation, 448
Slopes, 653, 657, 663, 671, 673, 762, 765, 839, 846, 855
Smoothing, 81-2, 290, 323-4, 700, 775, 777, 871, 874
Smoothness, 31, 34, 37, 80, 86, 103 169, 194, 237-9, 240-3, 278-9, 282-3, 288, 295, 301, 304, 306, 343-4, 359, 440-1, 447, 488, 490, 496, 659, 667, 677, 769, 807, 826, 854
Smoothness constraint, 847, 849
Smoothness functional, 867, 870, 872, 875, 880
Solid mechanics applications, 253
Source terms, 360, 393, 398, 401, 671, 777, 865
Spacing, 36, 81, 83, 99-101, 104, 137, 139, 144, 154, 156, 160, 237-8, 242, 254, 266-78, 300, 304-5, 310, 355-62, 416, 418, 426, 586, 592, 594-5, 598, 619-20, 634, 656-60, 668, 673, 688, 690, 692, 694, 700, 747-8, 762-5, 768, 778, 781-3, 813, 834, 867, 875, 890
Spacing parameters, 780
Sparse areas, 592-4, 598
Special points, 14, 104, 236
Sphere, 55-6, 58, 201, 217, 724-7
Spherical projection, 724
Spline interpolation, 118, 130, 157, 164, 367, 488, 565, 567, 572, 610, 660-1, 860
Splittings, 347-8
Springs, 825-8, 831
Square root transformation, 114
Stability, 277, 282, 286, 355, 486, 880
Stacking, 13
Staggered-cell, 621, 628, 879
Stereographic projections, 113
Stiff, 346-7
Storm surge, 410
Strain, 838-46, 849, 852, 855-6
Stream function, 303, 743, 751
Stream-function-vorticity equations, 37, 39
Streamline coordinate system, 515, 554, 586, 590, 740, 742, 748
Streamline plane, 744-5, 752, 755, 757, 759
Streamlines, 741

Stress, 258-60, 266-73
Stretching transformation, 109-10, 127-8, 238, 242, 295, 304-6, 316, 354, 359, 364, 368, 375, 410, 416, 550, 563-73, 579, 675, 688, 691, 704, 706, 824, 828-9, 842, 845
String mapping, 588-9, 598-9
Strong conservation law form, 301
Strongly-implicit procedure, 102
Structuring, 236
Subdomains, 235, 437-40
Subgrid boundaries, 761, 769, 773
Subgrids, 761-3, 768-70
Submerged hydrofoil, 396
Subregions, 176-7, 667, 675-7, 681, 761
Successive line overrelaxation, 241, 677, 703
Successive overrelaxation, 367, 393, 634
Successive surface systems, 13
Supersonic flow, 113, 243, 360, 529, 865
Surface, 41, 44-9, 53, 55, 72-3, 86, 89-90, 99, 100, 107, 111, 145, 171, 178, 193-6, 201, 217, 224, 226, 239, 242, 447-53, 467, 472-3, 548-9, 565, 576, 669-76, 697, 717-23, 727
Surface Christoffel symbols, 45, 55
Surface decomposition techniques, 177
Surface grids, 71, 137-9, 156-7, 200, 202, 206, 210-1, 238, 241-2, 249, 447, 450-4, 462, 667, 672-7, 680, 684, 686
Surface integral, 24, 28
Surface metric, 194
Surface operators, 451
Surface pressure, 303
Survey of coordinate system generation, 29, 123, 232
Symbolic Manipulation, 729, 734
Symm's method, 605

T

T-form, 395, 697, 709
Tangent, 19
Tangential derivatives, 23, 27
Tau Computational Space, 844, 848
Taylor series, 35, 504, 795, 815
Tensor densities, 526, 531, 534
Tensor form, 18, 29, 531
Tensor product, 229, 488-9, 499
Tesselation, 467-71
Theodorsen-Garrick transformation, 116-7
Theorema egregium, 47, 199
Thermal-hydraulic, 621
Three-dimensional grid, 13, 41, 99, 128, 137, 171, 177, 193, 201, 237-8, 547, 563, 667, 687, 695, 717, 761, 865, 869, 892
Tidal cycle, 427
Time-dependent, 277, 284-5, 342, 345, 386, 406, 809
Time derivatives, 28, 428, 813
Timings, 445, 731
TOMCAT code, 10, 81, 90, 419, 426, 633
Topology, 137, 139, 168-9, 235, 465, 471-2, 475
Torques, 259
Torsion, 253-5, 258, 272-3, 839, 842
Tracking, 115-7

Trailing edges, 580-1
Transfinite interpolation, 137-9, 144, 153-4, 164, 171, 173, 175, 177, 229, 448
Transformation relations, 1, 2, 18, 29, 104, 143, 203, 226, 262, 855
Transient problems, 342
Transonic flows, 109, 243, 252, 304, 361, 481, 508, 547, 560, 573, 576, 585, 653, 739-41, 744, 747-9
Transport equations, 284
Triangular grid, 177, 274, 489, 495
Triangular plate, 272
Tridiagonal solution, 776, 781
Tri-element, 653
Trilinear interpolant, 177-8
Triply orthogonal system, 201
Truncation error, 31-40, 86, 282-6, 320-1, 324, 329, 334, 340, 342, 344, 347, 426, 787, 792, 794, 800, 804-6, 859-60
Two-boundary technique, 137-9, 154, 159-60, 164
Turbine, 116, 118, 507-9, 517-23
Turbomachinery, 113, 117, 360-1, 507, 563-4
Turbulence model, 358-9, 377, 516
Two-element airfoil, 758

U

Umbilic, 201
Unified measure field, 837
Uniformity, 159, 206
Unsteady, 350
Unit normal vector, 45

V

VAHM, 410, 422, 426, 434
Variable node formulation, 339, 344, 347
Variational formulation, 85, 277-9, 282, 287, 806, 865-6, 885
Vector processing, 295, 393, 498, 732, 819, 868
Velocity potential, 110, 303
Vertically averaged, 410, 412, 414, 419, 422
Viscous flow, 37, 507, 572, 653, 809
Viscous stresses, 358
Viscous sublayer, 364
Volume, 19, 85, 787, 790, 807
Volume integral, 24, 28
Volume sheet, 469-70
Volume strings, 469
Volume variation, 200
Volume weighting functional, 867, 871-2, 880
Vortex sheets, 467

W

Wake, 359, 549, 553, 564, 568-73, 578, 581, 829-30, 833
Water quality problems, 409
Water surface elevation, 423, 425, 435-6
Water wave, 385-6, 809
Waterways, 360
Wave equation, 243, 251
Wave motion, 385

Wave peak, 401
Weak conservation law form, 301
Wedge, 539-43, 860, 862
Weight function, 86, 282-90, 342-6, 353, 355, 819, 854, 870, 875
Weighted volume variation, 278
Weingarten equations, 46
WESCOR, 9
Wind tunnel, 361, 749, 759-60, 865, 878
Wing, 114, 236, 239, 303, 305, 529, 662, 664, 686, 870, 875
Wing-body, 236-7, 242, 245-9, 313, 465, 472-3, 667, 676, 680, 688
Wing-tip, 236, 573, 667, 677, 684-5, 695, 697, 706

Z

Zonal boundaries, 657-63
Zonal grid, 656-9, 663