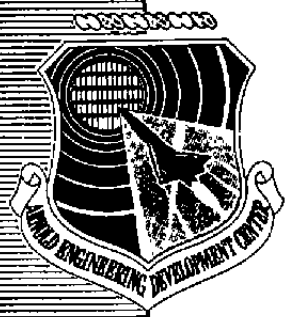
AEDC-TR-83-22

C.2



Numerical Solution of the Three-Dimensional Unsteady Euler Equations

K. Koenig
Mississippi State University

and

J. M. Barton
Sverdrup Technology, Inc.

August 1983

Final Report for Period January 1, 1982 to December 31, 1982

Approved for public release; distribution unlimited.

ARNOLD ENGINEERING DEVELOPMENT CENTER
ARNOLD AIR FORCE STATION, TENNESSEE
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE

NOTICES

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, or in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Qualified users may obtain copies of this report from the Defense Technical Information Center.

References to named commercial products in this report are not to be considered in any sense as an endorsement of the product by the United States Air Force or the Government.

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

APPROVAL STATEMENT

This report has been reviewed and approved.

Keith L Kushman

KEITH L. KUSHMAN
Directorate of Technology
Deputy for Operations

Approved for publication:

FOR THE COMMANDER

Marion L. Laster

MARION L. LASTER
Director of Technology
Deputy for Operations

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AEDC-TR-83-22
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: NUMERICAL SOLUTION OF THE THREE-DIMENSIONAL UNSTEADY EULER EQUATIONS

5. TYPE OF REPORT & PERIOD COVERED: Final Report Jan. 1, 1982 - Dec. 31, 1982

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): K. Koenig, Mississippi State University and J. M. Barton, Sverdrup Technology, Inc.

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Arnold Engineering Development Center/DOT, Air Force Systems Command, Arnold Air Force Station, TN 37389

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 65807F

11. CONTROLLING OFFICE NAME AND ADDRESS: Arnold Engineering Development Center/DOS, Air Force Systems Command, Arnold Air Force Station, TN 37389

12. REPORT DATE: August 1983

13. NUMBER OF PAGES: 42

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:

15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

Available in Defense Technical Information Center (DTIC).

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

Euler equations
compressible flow
MacCormack algorithm

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

A user's guide is presented for the three-dimensional, time-dependent Euler code, ARO-1. The guide provides a brief theoretical background for the explicit MacCormack algorithm and its application to the Euler equations. The main routines of the program are described and the calling order examined for a sample computational step. The primary code variables are defined, along with the nondimensionalization employed. Specification of the initial and boundary conditions is discussed from a theoretical and practical perspective. The required input data are presented for a sample case.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

PREFACE

The work reported herein was conducted by the Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC) under Project Number D173EW. The Air Force Project Manager was Dr. K. L. Kushman, AEDC/DOT. The results of the research were obtained by Sverdrup Technology, Inc., operating contractor for the propulsion test facilities, AEDC, AFSC, Arnold Air Force Station, TN 37389. Dr. Koenig contributed to the effort as a consultant from the Department of Aerospace Engineering, Mississippi State University, Mississippi State, Mississippi. The manuscript was submitted for publication on April 1, 1983.

CONTENTS

ILLUSTRATIONS

APPENDICES

1.0 INTRODUCTION

The governing field equations which describe the general cases of fluid flow are often referred to as the Navier-Stokes equations. If the fluid is assumed to be inviscid, then the equations of motion are often called the Euler equations. If the fluid is assumed to be inviscid and the flow is assumed to be irrotational, then the flow is termed potential. The full Navier-Stokes equations are a coupled system of second order, nonlinear partial differential equations; the Euler equations are a coupled system of first order, nonlinear partial differential equations, and the governing equation for potential flow is a second order, nonlinear partial differential equation. The Navier-Stokes equations can treat virtually any flow problem but are the most difficult to solve. The potential flow equation, although the easiest to solve, is the most limited in its applications. The Euler equations are intermediate in both respects and provide a practical means of solving fairly difficult problems in an economical way. If used properly, the Euler equations may be used to describe three-dimensional, unsteady flow at any Mach number and without assumptions of irrotational or isoenergetic conditions. The Euler equations are not subject to irrotational and isoenergetic assumptions with the important consequence that flows with curved shocks or multiple, intersecting shocks can be treated, thus making the equations applicable to transonic and supersonic flow past complex bodies or combinations of bodies.

2.0 BASIC CONCEPTS

The computer program, ARO-1 (Ref. 1), solves the three-dimensional, unsteady Euler equations in Cartesian coordinates using a finite volume (volume flux) approach. The basic numerical algorithm is the explicit predictor-corrector scheme of MacCormack (Ref. 2). Reference 1 discusses in detail this computer program, the fluid mechanics of the problem, and the numerical techniques involved in the solution to the equations. This report serves two purposes: (1) to introduce the principles of computational fluid dynamics to engineers and scientists unfamiliar with this subject and (2) to provide detailed instructions on the use of this particular computer solution of the Euler equations.

2.1 THE MESH

The Euler equations are partial differential equations. Numerical solutions to differential equations require approximating continuous functions (such as pressure) and continuous operations (mainly differentiation and integration) by discrete values, differences, and sums done for discrete points in space and time. The discrete points in space form the basis of what is called a computational grid or mesh. This mesh is an essential part of any CFD code. It must be fine enough to resolve important flow details, yet coarse enough to keep storage and computational time requirements reasonable. In addition, it should be laid

out and numbered in a consistent manner so that the computations can progress smoothly and unambiguously through the mesh. The mesh is also one of the portions of this program that users will need to modify or create on their own for particular problems; therefore, a thorough understanding of this mesh scheme is necessary.

Basically a mesh or grid should concentrate points near the body and spread them out farther away. Figure 1 presents an example of a very coarse grid about an axisymmetric body. Figure 1a shows the side view, Fig. 1b shows the front view midway back of the body, and Fig. 1c shows an oblique view of a portion of the grid. Nomenclature used in the program is also given. The mesh shown here would be referred to as a $10 \times 3 \times 7$ mesh; i.e., there are 10 planes perpendicular to the X axis, 3 surfaces enclosing the X axis, and 7 planes radially out from the X axis. The numbering scheme increases the indices with increasing values of X, radial location, and azimuthal location (clockwise from vertical). The subscripts (JX, JY, JZ) are not synonymous with the coordinates (X, Y, Z). For example, in this grid two points with the same JZ may have different Z values.

The network of lines forms six-sided volumes (some, on the X axis, degenerate to only five sides) with eight corners or vertices. To refer to a particular volume in the mesh, the subscripts for the corner in the most downstream, radially outward, clockwise location are used. The subscripts for this corner are also the largest values of JX, JY, and JZ which any corner in the volume has. For example, the shaded volume in Fig. 1c would be designated as volume (5, 3, 2). The physical dimensions of the cell volumes and surface areas are referred to as metrics. Determining these quantities is not as straightforward as might be expected; this will be discussed in more detail elsewhere in this report.

In ARO-1 the grid is generated (or input) in one subroutine, CONE, whereas the metrics are determined in another subroutine, VOLGRD. Usually the grid is generated in an independent mesh program, and CONE is used to transmit the coordinates (X, Y, Z) from disk storage to ARO-1. The subroutine VOLGRD is quite general and will work for any grid provided the numbering scheme described above is used.

2.2 THE EQUATIONS

The Euler equations describe the flow of an inviscid fluid. For negligible body forces (mainly gravity) these equations can be written in rectangular Cartesian coordinates as

Continuity
$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \bar{q}) = 0 \tag{1}$$

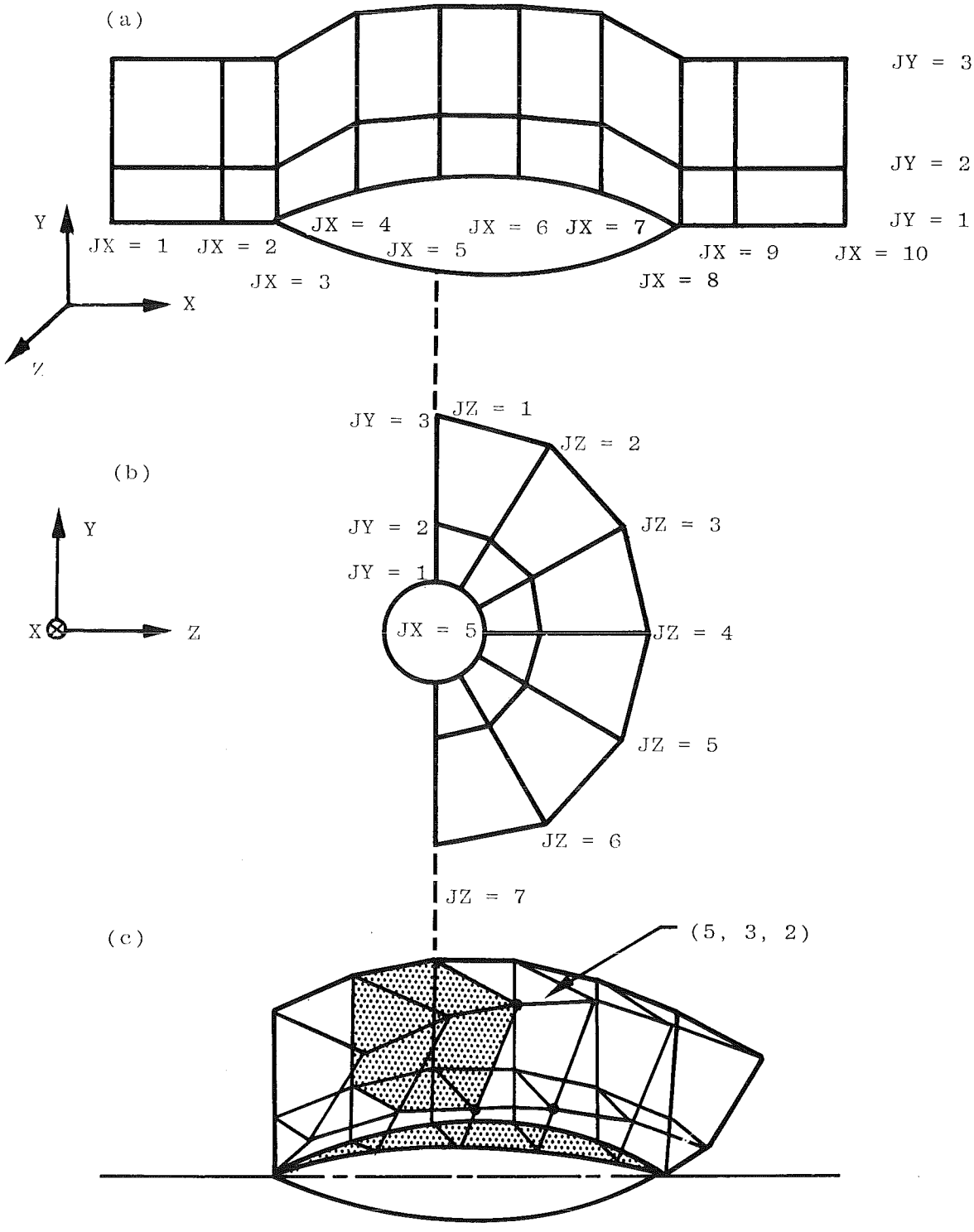


Figure 1. Simple axisymmetric grid.

Momentum

$$\frac{\partial(\varrho u)}{\partial t} + \nabla \cdot (\varrho u \bar{q}) = -\frac{\partial p}{\partial X}$$

$$\frac{\partial(\varrho v)}{\partial t} + \nabla \cdot (\varrho v \bar{q}) = -\frac{\partial p}{\partial Y}$$

$$\frac{\partial(\varrho w)}{\partial t} + \nabla \cdot (\varrho w \bar{q}) = -\frac{\partial p}{\partial Z} \tag{2}$$

Energy

$$\frac{\partial}{\partial t}\left[\varrho\left(\tilde{e} + \frac{q^2}{2}\right)\right] + \nabla \cdot \left[\varrho\left(\tilde{e} + \frac{q^2}{2}\right)\bar{q}\right] = -\nabla \cdot (\varrho \bar{q}) \tag{3}$$

where $\bar{q} = u\hat{i} + v\hat{j} + w\hat{k}$. In addition, an equation of state is required. Here a perfect gas is assumed, so that

$$p = \varrho RT$$

$$\tilde{e} = c_v T = \text{internal energy}$$

$$\gamma = c_p / c_v$$

$$c^2 = \gamma RT = \gamma p/\varrho = \text{speed of sound (squared)} \tag{4}$$

With some algebra these equations can be rearranged so all have the same strong conservation form:

$$\frac{\partial G}{\partial t} + \nabla \cdot F = 0 \tag{5}$$

Here G is a vector formed of the dependent flow variables ϱ, ϱu, ϱv, ϱw, and e; and F is a matrix of flux quantities, such as ϱuv (the flux of X momentum in the Y direction). More exactly, it is expressed as

$$G = \begin{bmatrix} \varrho \\ \varrho u \\ \varrho v \\ \varrho w \\ e \end{bmatrix} \quad F = \begin{bmatrix} \varrho u & \varrho v & \varrho w \\ \varrho u^2 + p & \varrho uv & \varrho uw \\ \varrho uv & \varrho v^2 + p & \varrho vw \\ \varrho uw & \varrho vw & \varrho w^2 + p \\ (e + p)u & (e + p)v & (e + p)w \end{bmatrix} \tag{6}$$

The energy e here is actually the sum of the internal and kinetic energy per unit volume:

$$e = \varrho\left(\tilde{e} + \frac{q^2}{2}\right) \qquad q^2 = u^2 + v^2 + w^2 \tag{7}$$

and is related to pressure by

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\varrho q^2 \tag{8}$$

If Eq. (5) is applied to one of the small stationary volumes, V, of the mesh it may be written as (see Section 4.0)

$$\frac{\partial \overline{G}}{\partial t} = -\frac{1}{V}\int_S F \cdot \hat{n}\, dS \tag{9}$$

where $\hat{n}$ is the outward unit normal to the surface, S, enclosing V, and $\overline{G}$ is the average value of G in the volume. Here this average value is approximated by the value at the center.

Equation (9) is the (schematic) governing equation of Euler flow. It describes three-dimensional unsteady flows, but is used in ARO-1 to calculate steady flows. To do so, a set of initial conditions for $\overline{G}$ (and therefore F) is assumed. These initial conditions are arbitrary but imposing free-stream conditions everywhere works well. Through the use of Eq. (9), the time rate of change of $\overline{G}$ is determined, and values of $\overline{G}$ at later times are found. Time is allowed to continue until the values of $\overline{G}$ at one time are sufficiently close to the values at the preceding time to consider the process as converged. These converged values of $\overline{G}$ are then the solution to the problem. The solution is, of course, constrained by boundary conditions. These boundary conditions ensure the uniqueness of the solution.

The ARO-1 program solves Eq. (9) using MacCormack's explicit predictor-corrector scheme. Details of this scheme as applied to Eq. (9) are described in the next section. Here it is sufficient to observe a few general characteristics of the technique and this equation. Equation (9) provides a means of finding $\overline{G}$ at a later time knowing F at the present time. The explicit nature of the technique means that at a given instant of time all values of F are known and $\partial\overline{G}/\partial t$ can be found by one equation. (Implicit techniques require solving simultaneous equations.) Because calculations are actually done using finite time steps and values of F known only at certain points (rather than continuous distributions), the time derivative and surface integrals in Eq. (9) can only be approximated. The predictor-corrector nature of the technique means that Eq. (9) is used twice, first to predict values of $\overline{G}$ and F at

at a later time, then to take these predicted values and obtain improved or corrected values of $\overline{G}$ and F at this later time. These corrected values can be shown to be formally second order accurate in the time and space intervals, which is sufficient for many applications.

3.0 IMPLEMENTATION OF MACCORMACK'S EXPLICIT PREDICTOR-CORRECTOR TECHNIQUE

The equations of motion are schematically written

$$\frac{\partial G}{\partial t} + \nabla \cdot F = 0 \tag{10}$$

where G represents the flow variables and F represents the flux vectors. If G were known at all times throughout space, the problem would be solved. The flow variable G is found as follows.

Suppose at some time t we know G everywhere (either because we assumed some values or have calculated them up to that time). The values of G at a later instant in time can be found by expanding G in a Taylor series about t, i.e,

$$G(t + \Delta t) = G(t) + \left(\frac{\partial G}{\partial t}\right)_t \Delta t + \frac{1}{2}\left(\frac{\partial^2 G}{\partial t^2}\right)_t \Delta t^2 + \ldots \tag{11}$$

where the subscript t on the derivatives means to evaluate the derivatives at time t. The larger the time step Δt is, the more terms are needed in Eq. (11) to give a good value for $G(t + \Delta t)$. For finite time steps, Eq. (11) might be *approximated* by

$$G(t + \Delta t) = G(t) + \left(\frac{\partial G}{\partial t}\right)_{avg} \Delta t \tag{12}$$

where $(\partial G/\partial t)_{avg}$ is some effective, or average, value of the time derivative over the interval t to $t + \Delta t$. Finding the new value of G now becomes a problem of finding $(\partial G/\partial t)_{avg}$. This is done through the use of Eq. (10). The flux vectors F in Eq. (10) are simply algebraic combinations of the flow variables G. If G is known everywhere at time t, then F is also known everywhere at the same instant, and Eq. (10) gives a way to find the time derivative of G.

To use Eq. (10) on a grid of *discrete* points, some manipulations are necessary. Equation (10) is rearranged and integrated over one of the small, stationary grid volumes to give

$$\int_V \frac{\partial G}{\partial t}\, dV = -\int_V \nabla \cdot F\, dV \tag{13}$$

Consider the left-hand side of Eq. (13). Since the volumes are assumed to be stationary, the time derivative can be taken outside. The mean value theorem of calculus can then be applied to the integral. This sequence yields

$$\int_V \frac{\partial G}{\partial t}\, dV = \frac{\partial}{\partial t} \int_V G dV = \frac{\partial}{\partial t} (\tilde{G} \int_V dV) = \frac{\partial}{\partial t} (\tilde{G} V) = V \frac{\partial \tilde{G}}{\partial t} \tag{14}$$

where $\tilde{G}$ is the mean value of G in the volume V. If the volume is small, then $\tilde{G}$ is approximately the same as the value of G at the volume center; therefore, Eq. (14) gives

$$\int_V \frac{\partial G}{\partial t}\, dV = V \frac{\partial G}{\partial t} \tag{15}$$

Note that Eqs. (13) and (14) are exact mathematically, given that Eq. (10) and Eq. (11) are also exact (with the proper conditions of G being differentiable and integrable). Equations (12) and (15) introduce approximations. By considering the right-hand side of Eq. (13), we use the divergence theorem to change the volume integral to a surface integral, i.e.,

$$\int_V \nabla \cdot F\, dV = \int_S F \cdot \hat{n}\, dS \tag{16}$$

where $\hat{n}$ is the unit outward normal of S. We use Eqs. (15) and (16) in Eq. (13) and divide through by V to give

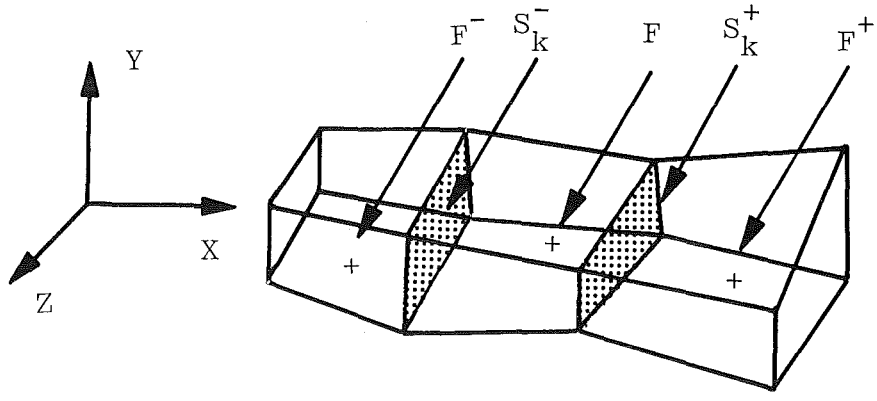
$$\frac{\partial G}{\partial t} = -\frac{1}{V} \int_S F \cdot \hat{n}\, dS \tag{17}$$

Equation (17) is now the means by which the first time derivative of G, evaluated at the volume center and at time t, can be found approximately. The quantity G on the right-hand side of Eq. (15) and in Eq. (17) is the same as the $\overline{G}$ in Section 2.2.

To eliminate possible confusion, the basic purpose must be reexamined. Our goal is to find $G(t + \Delta t)$ knowing G(t). We decided to do this by Eq. (12), but this equation requires finding $(\partial G/\partial t)_{avg}$ (which has has not been defined yet). A specific time derivative can be found using Eq. (17); therefore, we have a tool to use. However, there is an additional consideration to using this tool. Equation (17) requires an integration of F over the surfaces of a grid volume, but we only know F at discrete points in space, in fact, only at the centers of volumes. So we have two problems to solve in order to find $G(t + \Delta t)$: (1) how do we find $(\partial G/\partial t)_{avg}$ and (2) how do we perform the integration in Eq. (17)? These two problems are addressed simultaneously by MacCormack's explicit predictor-corrector technique. Justifying that the steps in MacCormack's method are valid and accurate to a particular

order is not that simple, but it can be done. Let us accept that the technique does work and is essentially second order accurate and instead worry only with how it is used and that it, at least, is reasonable.

A digression is necesary at this point to review the computational grid. The grid points form the corners of six-sided volumes (hexahedra), three of which are sketched below.



The crosses in the sketch are the volume centers where the flow variables and flux vectors are known, here denoted by F^-, F, and F^+. The two shaded faces, S_k^-, and S_k^+, are opposing faces of the middle volume to serve as examples.

The integration in Eq. (17) determines the net flux through the volume in question. Since each volume has three pairs of opposing sides, the integral can be treated as the sum of the integrals over each of the six surfaces, i.e.,

$$\int_S F \cdot \hat{n}\, dS = \sum_{k=1}^{3} \int_{S_k^+} F \cdot \hat{n}\, dS + \sum_{k=1}^{3} \int_{S_k^-} F \cdot \hat{n}\, dS \tag{18}$$

which is exact. However, F is not known over each surface but only at the volume center points, so the individual integrals must be approximated. This approximation can be done in a variety of ways. For example, the flux across S_k^- might be determined using an average of F^- and F, or an average of F^-, F plus other neighboring centers (not shown) or using F^-, or F alone. The MacCormack method essentially uses an average of F^- and F, but does so indirectly in two steps, as will be shown later in this report.

The term predictor-corrector means a solution for G (t + Δt) is first predicted, then this prediction is corrected. The quantity $(\partial G/\partial t)_{avg}$, defined by

$$\left(\frac{\partial G}{\partial t}\right)_{avg} = \frac{1}{2}\left[\left(\frac{\partial G}{\partial t}\right)_t + \left(\frac{\partial \bar{G}}{\partial t}\right)\right] \tag{19}$$

incorporates this idea; $(\partial G/\partial t)_t$ predicts a value and $(\partial \bar{G}/\partial t)$ corrects the result. (Note that the overbar here indicates a predictor-corrector quantity and is a different usage from that of Section 2.2.) The predicted value of G (t + Δt) is given by

$$\bar{G}(t + \Delta t) = G(t) + \left(\frac{\partial G}{\partial t}\right)_t \Delta t \tag{20}$$

Equations (17) and (18) are used to obtain $(\partial G/\partial t)_t$:

$$\left(\frac{\partial G}{\partial t}\right)_t = -\frac{1}{V}\left(\sum_{k=1}^{3} F \cdot S_k^+ + \sum_{k=1}^{3} F^- \cdot S_k^-\right) \tag{21}$$

Note here that the flux integrals are approximated by computing the scalar product of the flux vector "behind" the appropriate face with the area vector of that face. The quantities G (t + Δt) from Eq. (20) can now be used to find predicted values for the flux vectors, F(t + Δt). These new flux vectors are used to find the predicted derivative, which serves as a corrector to G(t + Δt):

$$\left(\frac{\partial \bar{G}}{\partial t}\right)_t = -\frac{1}{V}\left(\sum_{k=1}^{3} \bar{F}^+ \cdot S_k^+ + \sum_{k=1}^{3} \bar{F} \cdot S_k^-\right) \tag{22}$$

Here the flux integrals are approximated by using the flux vectors "ahead of" the appropriate face. Equation (22) is the corrector step and completes the flux averaging problem mentioned earlier.

In principle all the ingredients to find G(t + Δt) are now available. First, using the known conditions at time t, we find $(\partial G/\partial t)_t$ from Eq. (21). By substituting this result into Eq. (20) to find $\bar{G}(t + \Delta t)$, we then use $\bar{G}(t + \Delta t)$ to find $\bar{F}(t + \Delta t)$; and substituting $\bar{F}(t + \Delta t)$ into Eq. (22), Eq. (22) into Eq. (19), and Eq. (19) into Eq. (12) yields G(t + Δt). In the notation of Ref. 1, $G(t) = G^n$, $G(t + \Delta t) = G^{n+1}$, and $\bar{F}(t + \Delta t) = F$. Except for a factor ω, Eqs. (20) and (21) here form the first part of Eq. (7) in Ref. 1, whereas Eqs. (12), (19), (20), and (22) here combine to form the second part of Eq. (7) in Ref. 1.

4.0 DISCUSSION AND RECOMMENDATIONS

Time-dependent, predictor-corrector solution techniques of the three-dimensional, unsteady Euler equations can be effectively applied to a rather wide range of practical

situations, as demonstrated in Refs. 1, 3, and 4. The preceding sections have presented introductions to three essential aspects of this, or any, CFD program: the equations being solved, the computational grid on which the solution is determined, and the numerical algorithm used to solve the equations. Satisfactory application of this program requires users, especially those new to CFD, to understand thoroughly the details of the code. Also included in this report are a discussion of program organization (Appendix A), a discussion of program sequence (Appendix B), and a detailed description of the major subroutines (Appendix C). Although some of this information is not absolutely necessary in order to use the program, careful study of the Appendices will enable users to apply the code more efficiently to their particular problem. The full capabilities of this program can be determined and exploited only if the code is tested and used for a variety of problems; therefore, careful study before application is highly recommended.

REFERENCES

1. Jacocks, J. L. and Kneile, K. R. "Computation of Three-Dimensional, Time-Dependent Flow Using the Euler Equations." AEDC-TR-80-49 (AD-A102463), July 1981.

2. MacCormack, R. W. "The Effect of Viscosity in Hypervelocity Impact Cratering." AIAA Paper No. 69-354, 1969 .

3. Whitfield, D. L., Swafford, T. W., and Jacocks, J. L. "Calculation of Turbulent Boundary Layers with Separation and Viscous-Inviscid Interaction." *AIAA Journal*, Vol. 19, No. 10, October 1981.

4. Barton, J. M. "The Role of CFD in Aeropropulsion Ground Testing." AIAA Paper 83-0149, January 1983.

APPENDIX A
PROGRAM ORGANIZATION

1. Introduction

The program in its present form consists of a main calling program and 17 subroutines. The calculations involve considerable looping among the main program and the other 17 subroutines. Extensive use is made of labelled common regions to transfer data among these subroutines. Input to the program is quite minimal. Only four data cards are read although this is a little misleading, since the mesh coordinates are usually input through an unformatted READ statement in one of the subroutines and other information is provided through the use of PARAMETER and DATA statements.

2. Subroutines

A list of the main program and subroutines follows with very brief descriptions of the main activities of each unit. The list is in the order in which the routines appear in the program and not in the order in which they are called. Some of these subroutines will be described in greater detail in later sections.

a. ARO1 is the main calling program. It calls (in chronological order) the subroutines which (1) allow for restart, (2) establish the initial conditions, (3) generate (or read) the grid, (4) compute the grid metrics, (5) initialize certain arrays, (6) print the grid mesh point locations, (7) print the initial conditions, (8) control the computational cycles, and (9) stop the program when everything is finished.

b. START subroutine reads a restart file produced from a previous calculation and provides for creation of a subsequent restart file at the conclusion of additional calculations. This subroutine also initializes the flux accumulators, the boundary conditions, and time step.

c. PVAR subroutine is user-adaptable and produces the desired printed output of flow-field and geometry variables. The user can tailor the subroutine as required.

d. PARRAY subroutine writes any array passed to it.

e. CONE subroutine is the vehicle for inputting the mesh coordinates. Usually the coordinates are computed using a separate mesh generation code; however, for simple geometries, the coordinates could be computed in this subroutine.

f. ICEES subroutine computes the initial conditions throughout the flow field, based on the specified Mach number and attitude.

g. VOLGRD subroutine computes the grid metrics (projected areas and volumes).

h. BC subroutine enforces the boundary conditions [Eq. (9), Ref. 1] and calls subroutine INFLOW.

i. INFLOW subroutine enforces the inflow boundary conditions using a reference plane characteristic approach (see, for example, Kneile et al.*). For subsonic flow the subroutine uses the specified total temperature, total pressure, and flow angles to compute the required dependent variables. For supersonic flow, all variables are specified at inflow. INFLOW calls subroutine CENTER.

j. CENTER subroutine takes the mesh coordinates (X, Y, Z), which are coordinates of the volume corners, and computes coordinates of the volume centers.

k. FLUXX, FLUXY, FLUXZ are subroutines which compute the fluxes of mass, momentum, and energy through the faces of the grid volumes in the JX, JY, and JZ directions, respectively. Flux accumulators which sum the net flow for the predictor-corrector calculations are also computed. Second order smoothing of the dependent variables is also available, depending on the value of the parameter KSMOOTH.

l. UPDATE subroutine computes the new or updated values of the flow variables (G) using the flux accumulators. This is where parts of Eq. (7) of Ref. 1 appear explicitly.

m. DTCAL subroutine computes the new time step to be used [Eq. (8), Ref. 1].

n. STEP subroutine controls the calls to the FLUX, UPDATE, SMOOTH, and DTCAL subroutines.

o. BSTEP subroutine controls the calls to the STEP and PVAR subroutines.

p. SMOOTH subroutine computes smoothed or averaged values of the flow variables inside the grid.

---

*Kneile, K. R., Todd, D. C., and Jacocks, J. L. "Characteristic Boundary Conditions for ARO-1." AEDC-TR-82-28, May 1983.

3. Inputs

Information is transferred into the program in three ways: through READ, DATA, and PARAMETER statements. For normal operation only four data cards are read. However, if the program is to be run with a restart capability, then additional information may be read from a specified file.

The data cards will be discussed in the order in which they are encountered, with definitions and typical values of the variables being read. The following applies to calculations without a restart.

```
   READ (5,100) KIN, KOUT, KSMOOTH, TFLAG
100 FORMAT (3I3, E24.0)
```

This read is in subroutine START.

KIN	=	logical unit number of restart file to be read (0 = no restart)	= 0
KOUT	=	logical unit number of restart file to be written after calculations are complete	= 0
KSMOOTH	=	flag to turn on second order smoothing in FLUX subroutines (0 = no smoothing)	= 0
TFLAG	=	maximum time value permitted if internal clock of computer is used to time the computation, min	= 7.00

```
   READ (5,100) FACT, FSMACH, ALPHA, BETA, PHI
100 FORMAT (8E10.0)
```

This read is in subroutine ICEES.

FACT	=	parameter for computational stability (CFL number)	= 0.8
FSMACH	=	free-stream Mach number	= 0.8500

ALPHA	=	free-stream flow angle, deg	= 0
BETA	=	free-stream flow angle, deg	= 0 . . . no incidence = -10.00 . . . 10-deg angle of attack
PHI	=	free-stream flow angle, deg	= 0

More information on the free-stream flow angles is available in Appendix C, ICEES and Free-Stream Conditions.

```
   READ (5,100) NB, NT, NV, FACTX
100 FORMAT (3I5, F20.0)
```

This read is in the main routine, ARO1. A very important parameter is the product

NB*NT	=	total number of time steps to be use	= 4000 (for example)
NB	=	number of printouts, i.e., calls to PVAR	= 20
NT	=	(NB*NT)/NB, must be a multiple of 8	= 200
NV	=	number of times the boundary-layer subroutine is called for versions of the code with viscous/inviscid interaction capability (Ref. 3)	= 0
FACTX	=	for restart runs, a new value for FACT; otherwise set to zero (For restart runs, subroutine ICEES is not called and the value of FACT is taken from the restart file; thus, to change to a new value, FACTX is used.)	= 0

```
   READ (5,100)
100 FORMAT (2I3,E24.0)
```

This read is in subroutine START. A blank card must be the fourth card.

After NB calls to subroutine BSTEP, the variables NB, NT, NV, FACTX are read again. If a zero value for NB is encountered, execution is terminated. If, after a specified number of iterations, it is desired to alter the Courant number or the amount of printout, this data card can be used for that purpose by indicating the new values of NB, NT, NV, and FACTX desired, where FACTX is the new Courant number.

Besides these reads from TAPE5, there are several READ(KIN) statements for restart which appear in subroutine START.

A second form of input is the DATA statement. The occurrences of DATA are as follows:

In ARO1:

DATA GAM, GM1, GS2, RGM1 / . . . /, where

GAM = γ, GM1 = γ-1, GS2 = $\gamma/2$, RGM1 = $(\gamma\text{-}1)^{-1}$.

The other two DATA statements set initial values for certain counters and reference numbers.

In FLUXX, FLUXY, FLUXZ:

DATA KODE / . . ./ provides a way to cycle around the eight corners of a grid volume. These should *not* be changed.

The last input method is the PARAMETER statement. It sets values of certain quantities which then cannot be changed by any later program statement. It must appear in every subroutine using those quantities. Here only one statement is used, but it is in almost all of the subroutines.

PARAMETER (NNX = 80, NNY = 32, NNZ = 2, . . .)

This particular statement is for dimensioning arrays for an 80 × 32 × 2 grid. Other grid sizes can be specified by replacing these numbers. This must be done in *every* PARAMETER statement. One major benefit of using the PARAMETER statement is that variables may be dimensioned (in DIMENSION or COMMON statements, for example) with variable sizes, if these sizes are first specified in PARAMETER statements. This saves a tremendous amount of work in this program when grid sizes are changed. (The PARAMETER statement is a feature of Cray computers and will require replacement when ARO-1 is installed on a different computer.)

4. Variables

There are large numbers of variables in this program, many of them triply subscripted with dimensions of about (NNX+1, NNY+1, NNZ+1). This leads to large storage requirements. Most of these variables appear in labelled COMMON statements, and these labelled COMMON statements appear in most of the subroutines even though the subroutines may actually require only a few of the variables. The variable names are, in many cases, descriptive of the actual quantities they represent, thus helping in understanding and using the program. The following list gives the more important variables, their dimension size, definitions, and the subroutines in which they are actually used. This list does not include some insignificant variables and variables which appear exclusively in SMOOTH. (In the following, the designation NNX1 means NNX + 1, with analogous meanings for NNY1 and NNZ1.)

A (NNX1, NNY1, NNZ1)	=	dummy name for any of the flow variables	PARRAY
ADA	=	$\overline{A} \cdot \overline{A}$, magnitude of area vector squared	BC, DTCAL
ALPHA	=	free-stream flow angle	ICEES
AX,AY, AZ	=	average value of the x, y, and z components of the area vectors of opposing faces	VOLGRD
AX1, AY1, AZ1 (NNX, NNY1, NNZ1)	=	components of area vector of face with constant JX	VOLGRD, FLUXX, DTCAL, INFLOW, BC
AX2, AY2, AZ2 (NNX1, NNY, NNZ1)	=	components of area vector of face with constant JY	VOLGRD, FLUXY, DTCAL, BC
AX3, AY3, AZ3 (NNX1, NNY1, NNZ)	=	components of area vector of face with constant JZ	VOLGRD, FLUXZ, DTCAL, BC

BETA	=	free-stream flow angle, negative of angle of attack	ICEES
CLSTEP	=	coefficient of flux summation [Eq. (11), Ref. 1]	UPDATE
DE, DR, DRU, DRV, DRW (NNX1, NNY1, NNZ1)	=	flux accumulators for energy, density, and (X, Y, Z) momentum	START, SMOOTH, FLUX, UPDATE
DT, DTM	=	time step [Eq. (8), Ref. 1]	DTCAL
DTSTEP	=	coefficient of flux accumulators [Eq. (11), Ref. 1]	UPDATE DTCAL
DTSVOL	=	coefficient of flux accumulators	UPDATE
DUM	=	denominator of Eq. (11), Ref. 1	DTCAL
DX1, DY1, DZ1	=	components of one face diagonal	VOLGRD
DX2, DY2, DZ2	=	components of opposite face diagonal	VOLGRD
DXC, DYC, DZC	=	components of vector between centers of opposing faces	VOLGRD
E (NNX1, NNY1, NNZ1)	=	energy	ICEES, BC, FLUX, DTCAL
EINIT	=	initial value of energy	ICEES
FACT, FACTX	=	CFL number	ARO1, ICEES, DTCAL
FF	=	$2^* (\rho\bar{q} \cdot \bar{A})/(\bar{A} \cdot \bar{A})$, mirror condition [Eq. (9), Ref. 1]	BC

FSMACH	=	free-stream Mach number	PVAR ICEES
F1, F2, F3, F4, F5 (NNX1, NNY1)	=	flux of mass, (X, Y, Z) momentum, and energy	FLUXX, FLUXY, FLUXZ
GAM	=	γ, ratio of specific heats	ICEES PVAR
GM1	=	$\gamma - 1$	ICEES
GS2	=	$\gamma/2$	DTCAL
INC	=	counter, number of times STEP is called	STEP
JJ, JS, JT	=	indices for predictor-corrector	FLUX
JX, JY, JZ	=	primary indices	various places
JX1, JX2, JY1, JY2 JZ1, JZ2	=	indices for predictor-corrector	FLUX
KIN	=	logical unit number for restart input file	ARO1 START
KODE	=	code for predictor-corrector sequence	FLUX
KOUNT	=	counter for time steps	PVAR, BC, STEP
KOUT	=	logical unit number for restart ouput file	ARO1 START
LIMIT	=	counter limit for calls to STEP	STEP
NB	=	number of printouts	ARO1
NEX, NEY, NEZ	=	grid size -1	various places

NNX, NNY, NNZ	=	grid size	various places
NNX1, NNY1, NNZ1	=	grid size + 1	various places
NT	=	frequency of printouts	ARO1 BSTEP
NV	=	number of calls to boundary-layer subroutine	ARO1
OMEGA	=	artificial viscosity parameter of Eq. (7), Ref. 1	UPDATE
OMTO	=	coefficient of flux accumulators $(1 - 2\omega)$	UPDATE
P (NNX1, NNY1, NNZ1)	=	pressure	PVAR, BC, FLUX, START, UPDATE, DTCAL, SMOOTH
PHI	=	free-stream flow angle	ICEES
PI	=	π	CONE
PINIT	=	initial value of pressure	ICEES
QDA	=	$\bar{q} \cdot \overline{A}$, volume flux	BC FLUX
R (NNX1, NNY1, NNZ1)	=	density	START, ICEES, BC, FLUX, UPDATE, DTCAL, SMOOTH
RGM1	=	$(\gamma - 1)^{-1}$	DTCAL

RQDA	=	$\varrho\bar{q} \cdot \overline{A}$	DTCAL
RQDQ	=	$\varrho\bar{q} \cdot \bar{q}$	SMOOTH, UPDATE, DTCAL
RQDRQ	=	$\varrho\,\bar{q} \cdot \varrho\,\bar{q}$	DTCAL
RU, RV, RW (NNX1, NNY1, NNZ1)	=	(X, Y, Z) momentum	START, ICEES, BC, FLUX, UPDATE, DTCAL, SMOOTH
RUINIT, RVINIT, RWINIT	=	initial (X, Y, Z) momentum	ICEES
TFLAG	=	internal program time limit	START STEP
TIME	=	current flow time	PVAR, PARRAY, UPDATE
VOL (NNX1, NNY1, NNZ1)	=	grid cell volume	VOLGRD, UPDATE, DTCAL, SMOOTH
VOLMIN	=	minimum grid cell volume	VOLGRD
X, Y, Z (NNX1, NNY1, NNZ1)	=	grid coordinates	PVAR, CONE, VOLGRD
XC, YC, ZC (NNX)	=	coordinates of face centers	VOLGRD
XDUM	=	denominator of Eq. (11), Ref. 1	DTCAL

APPENDIX B
PROGRAM SEQUENCE

The major calls, loops, and computations (in the order they occur) are described below. This sequence is for a run without the boundary-layer calculations, the internal clock, or the restart option, and is for the analysis of one body started impulsively with free-stream conditions everywhere. The activities described are performed in the program or subroutine identified immediately above the description.

The input cards will be taken as

KIN	=	0	no restart
KOUT	=	0	
TFLAG	=	7.00	
FACT	=	0.8	
FSMACH	=	0.85	
ALPHA	=	0.0	
BETA	=	0.0	
PHI	=	0.0	
NB	=	20	4000 times steps with a printout every
NT	=	200	200 time steps
NV	=	0	no call to boundary-layer routine
FACTX	=	0.0	

These are typical values for a run without the boundary layer or restart.

ARO1

calls START

reads KIN, KOUT, KSMOOTH, TFLAG

if KIN = 0, returns to ARO1

calls ICEES

reads FACT, FSMACH, ALPHA, BETA, PHI

computes initial values of R, RU, RV, RW, E

returns to ARO1

calls CONE

grid points (X, Y, Z) input

returns to ARO1

calls VOLGRD

computes AX1, AY1, AZ1, AX2, AY2, AZ2, AX3, AY3, AZ3, VOL

returns to ARO1

calls entry MIDDLE in START

computes P

initializes flux accumulators DR, DRU, DRV, DRW, DE

calls BC

computes initial values of boundary conditions

returns to MIDDLE

calls DTCAL

initializes time step

if KIN = 0 and KOUT = 0, returns to ARO1

calls PVAR

writes initial flow variables

returns to ARO1

reads NB, NT, NV, FACTX

At this point a nested loop process begins which (for the most general situation) is controlled by the values of NB, NT, NV, and TFLAG. For the sequence described here where the internal program clock and boundary layer are not being used, only NB and NT control the process.

ARO1 calls BSTEP (NB times)

BSTEP calls STEP (NT times)

STEP increments KOUNT by 1

STEP calls FLUX and UPDATE (2 times for each)
FLUX computes DR, DRU, DRV, DRW, DE

UPDATE computes TIME, R, RU, RV, RW, E, P,
DR, DRU, DRV, DRW, DE

UPDATE calls BC

BC computes R, E, P, RU, RV, RW at boundaries

STEP calls SMOOTH after every 16 calls to FLUX and UPDATE

SMOOTH computes DR, DRU, DRV, DRW, DE,
R, RU, RV, RW, E, P

SMOOTH calls BC

BC computes R, E, P, RU, RV, RW at boundaries

STEP calls DTCAL after every 128 calls to FLUX and UPDATE

DTCAL computes DTSTEP

BSTEP calls PVAR

PVAR writes computed flow variables

When this loop sequence is finished, KOUNT will equal the product NB*NT; there will be NB printouts, not including the initial flow field, and the control will be back in ARO1. For the case of no restart, the remaining steps end the program.

ARO1 calls entry FINISH in START

if KOUT = 0, returns to ARO1

STOP

END

APPENDIX C
SUBROUTINES

1. CONE

This subroutine reads the (X, Y, Z) coordinates of a grid. The grid is produced in a separate mesh generation program. The mesh generation routines can be included in CONE if desired.

2. ICEES and Free-Stream Conditions

Reference 1 states that all variables are dimensionless, with the reference conditions usually taken to be free-stream density and sound speed, the latter given by

$$c = \left(\gamma \frac{p}{\varrho}\right)^{1/2}$$

The FORTRAN program variables are R, P, E, RU, RV, and RW. Since these are dimensionless, then, for example,

$$R = \frac{\varrho}{\varrho_\infty}$$

and, therefore,

$$R_\infty = \frac{\varrho_\infty}{\varrho_\infty} = 1$$

Similarly, if C is the nondimensional sound speed, then

$$C = \frac{c}{c_\infty}$$

and, therefore,

$$C_\infty = \frac{c_\infty}{c_\infty} = 1$$

Since ϱ_∞ and c_∞ are the reference values (by definition), then

$$P = \frac{p}{\varrho_\infty c_\infty^2}$$

and not p/p_∞ as one might guess. But from Eq. (4), Ref. 1, $p = 1/\gamma\, \varrho c^2$, so

$$P = \frac{1}{\gamma}\frac{\varrho c^2}{\varrho_\infty c_\infty^2} = \frac{1}{\gamma} RC^2$$

and

$$P_\infty = \frac{1}{\gamma}$$

From Eq. (3), Ref. 1, we have

$$e = \frac{p}{\gamma-1} + \frac{1}{2}\varrho q^2 = \frac{p}{\gamma-1} + \frac{1}{2}\gamma p M^2$$

$$E = \frac{e}{\varrho_\infty c_\infty^2} = \frac{1}{\gamma-1}\frac{p}{\varrho_\infty c_\infty^2} + \frac{\gamma}{2}\frac{p}{\varrho_\infty c_\infty^2} M^2$$

and

$$E_\infty = \left(\frac{1}{\gamma-1}\right)\frac{1}{\gamma} + \frac{1}{2} M_\infty^2$$

Summarizing these results, the free-stream nondimensional flow is decribed by

$$R_\infty = 1,\ C_\infty = 1,\ P_\infty = \frac{1}{\gamma},\ E_\infty = \left(\frac{1}{\gamma-1}\right)\frac{1}{\gamma} + \frac{1}{2} M_\infty^2$$

If we let $\bar{q}$ be the dimensional velocity vector and q be its magnitude, then

$$q = (u^2 + v^2 + w^2)^{1/2}$$

$$M = \frac{q}{c}$$

Also,

$$U = \frac{u}{c_\infty},\ V = \frac{v}{c_\infty},\ W = \frac{w}{c_\infty}$$

so that U_∞, V_∞, and W_∞ are simply the (X, Y, Z) components of M_∞. This completes the description of the free stream.

The initial conditions are taken to be free-stream conditions everywhere and are established in ICEES. There are five equations of interest in ICEES which prescribe initial values of RU, RV, RW, P, and E, i.e.,

$$\text{RUINIT} = R_\infty U_\infty = \text{X component of } M_\infty \text{ (since } R_\infty = 1)$$

$$\text{RVINIT} = R_\infty V_\infty = \text{Y component of } M_\infty$$

$$\text{RWINIT} = R_\infty W_\infty = \text{Z component of } M_\infty$$

$$\text{EINIT} = E_\infty = (1/\gamma - 1)\ 1/\gamma + 1/2\ M_\infty^2$$

$$\text{PINIT} = P_\infty = 1/\gamma$$

The components of M_∞ are found from an Euler angle-type approach (see, for example, Etkin*). The orientation is slightly unconventional, however. The coordinate frame remains fixed to the body, and the incoming flow is put at an angle. The angles α, β, ϕ refer to yaw, pitch, and roll, respectively, where α is the angle in the XZ-plane, β is the angle in the XY-plane, and ϕ is the angle in the YZ-plane. For the coordinate frame as established in Fig. 1, a positive angle of attack (nose up) is given by BETA equal to a negative angle. All angles in ICEES are in degrees.

3. VOLGRD

This subroutine computes the area vectors (or project areas) of the grid volume faces and the volume of the elements. Before the actual details of the calculations are described, a brief discussion of the philosophy of the scheme is in order.

This subroutine was written to be able to handle almost any conceivable grid, as long as the grid is consistently numbered. In many cases (such as in this program), the grid elements are taken from some sort of regular geometric shape with well-defined flat sides; however, this is not always the case. It is quite possible to form a grid such that the corners of a volume side are *not* all in the same plane, and, consequently, the surface is not flat, as depicted in Fig. C-1. This introduces an essential difficulty. When the side is flat, its area and normal vector (which together give the area vector) can be found unambiguously and exactly (assuming the corners are connected by straight lines). But, if the four corners of a

*Etkin, B. *Dynamics of Atmospheric Flight*. John Wiley & Sons, Inc., New York, 1972, pp. 112-117.

side are not coplanar, then the area is not well-defined since only the four corners are actually known. Even if the surface can be analytically described, it will have a normal vector which varies in direction over the surface. For a numerical solution, it is highly desirable to have each surface of a volume element defined by a single, constant area vector. This means the nonplanar surface must be represented by an effective area vector which must be determined solely in terms of the corner coordinates. In addition, the method for doing this should yield the exact answer when the surface is flat. Finally, the method should be capable of properly handling surfaces that degenerate to a line or point (as happens in almost any grid scheme). The scheme used in VOLGRD satisfies all these requirements.

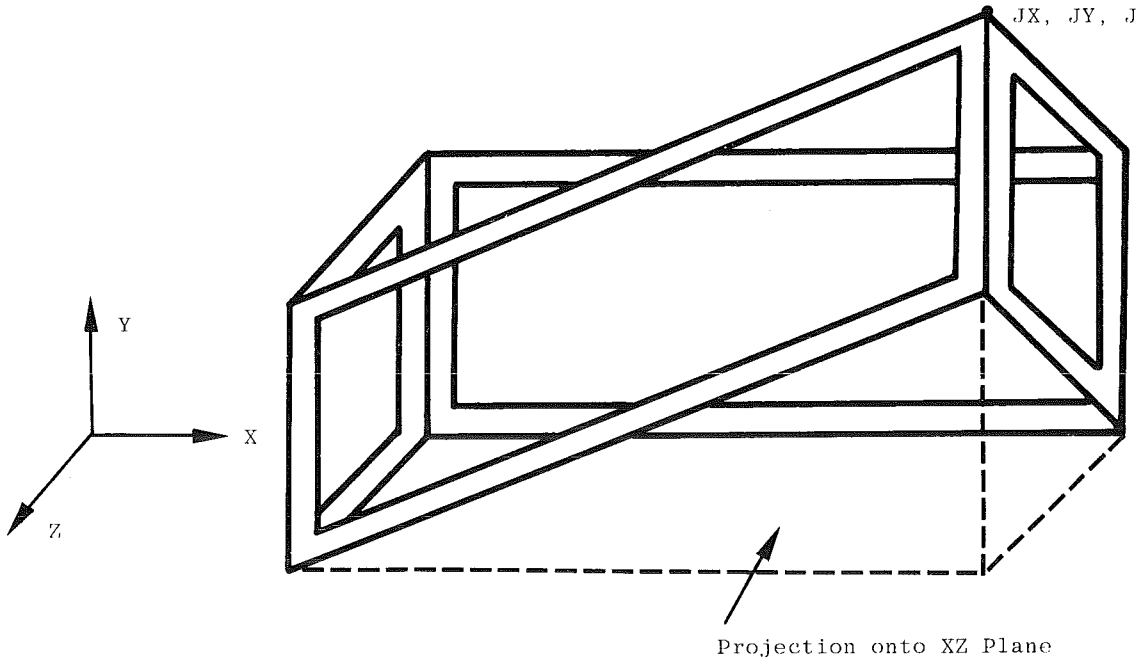


Figure C-1. Volume element with nonplanar surfaces.

Determining the volume of an element has similar difficulties. The volume of a flat-sided element can be found unambiguously and exactly (although the calculations may be quite tedious). For nonplanar surfaces, the element is poorly defined, and the volume becomes difficult to compute. For numerical solutions, this computation needs to be simplified and based only on the corner coordinates. Also, the result must be exact for flat-sided elements. The technique in VOLGRD satisfies these requirements as well.

A representative volume is shown in Fig. C-2. With the corners labelled as shown, this volume would be referred to by the subscripts (JX, JY, JZ). The surfaces of the element are surfaces on which the values of JX, JY, or JZ are constant and are denoted as A1, A2, and A3 respectively, there being two of each of these for any element. The two A1 surfaces have been singled out here to serve as examples. The calculations to be described are repeated for the other two surface pairs. All surfaces are denoted by the corner having the largest subscripts; therefore, in this example, the two A1 surfaces are referred to by (JX-1, JY, JZ) on the left and (JX, JY, JZ) on the right.

Each surface has two diagonals which can be treated as vectors . For surface A1 (JX, JY, JZ), the diagonal vectors are

$$D1 = [X(JX,JY,JZ) - X(JX,JY-1,JZ-1)]\hat{i} + [Y(JX,JY,JZ) - Y(JX,JY-1,JZ-1)]\hat{j} + [Z(JX,JY,JZ) - Z(JX,JY-1,JZ-1)]\hat{k}$$
$$= DX1\hat{i} + DY1\hat{j} + DZ1\hat{k}$$

$$D2 = [X(JX,JY-1,JZ) - X(JX,JY,JZ-1)]\hat{i} + [Y(JX,JY-1,JZ) - Y(JX,JY,JZ-1)]\hat{j} + [Z(JX,JY-1,JZ) - Z(JX,JY,JZ-1)]\hat{k}$$
$$= DX2\hat{i} + DY2\hat{j} + DZ2\hat{k}$$

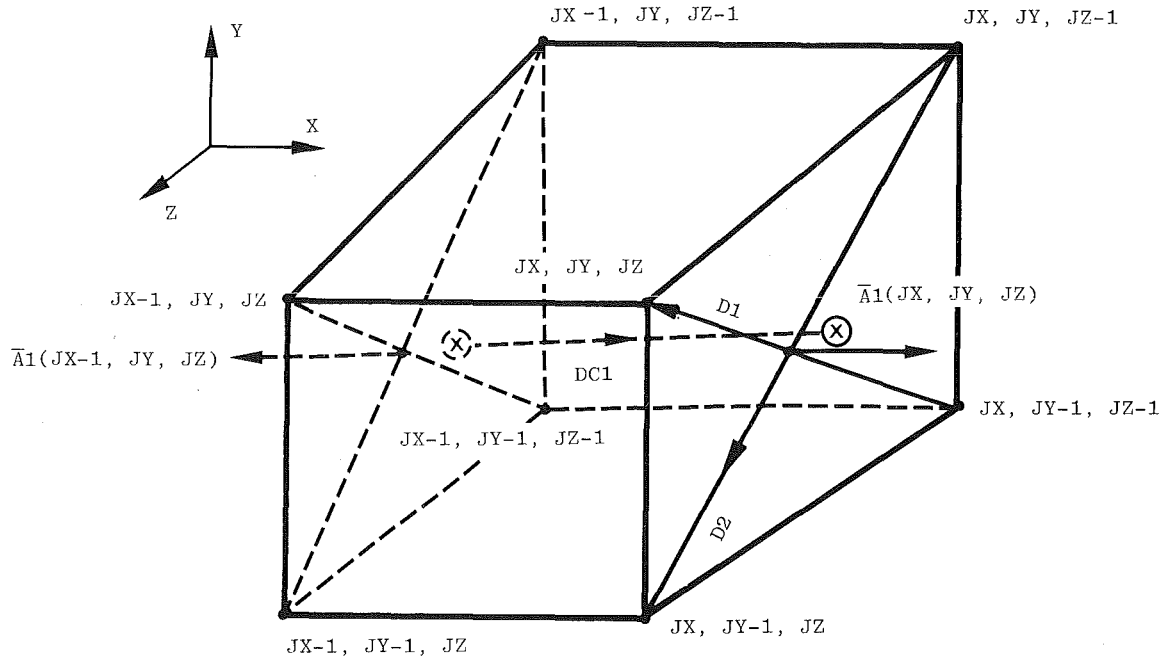


Figure C-2. Metrics of a volume element.

The area vector of surface A1(JX, JY, JZ) is *defined* to be one-half the cross product of the two diagonal vectors, i.e.,

$$A1(JX,JY,JZ) = \tfrac{1}{2}(D1 \times D2) = \tfrac{1}{2}(DY1*DZ2 - DZ1*DY2)\hat{i} + (DZ1*DX2 - DX1*DZ2)\hat{j} + (DX1*DY2 - DY1*DX2)\hat{k} = AX1\hat{i} + AY1\hat{j} + AZ1\hat{k}$$

which is exact for flat surfaces with straight lines connecting the corners. This vector is also shown in Fig. C-2. Similarly the diagonal vectors and area vector for side A1(JX-1, JY, JZ) are found.

Each side also has a center which is *defined* to be the average of the corner coordinates. For example, the center of A1(JX, JY, JZ) has an X value of

$$XC(JX,JY,JZ) = \tfrac{1}{4}[X(JX,JY,JZ) + X(JX,JY,JZ-1) + X(JX,JY-1,JZ-1) + X(JX,JY-1,JZ)]$$

The centers of opposing surfaces (here A1(JX-1, JY, JZ) and A1(JX, JY, JZ)) can be connected by a line which also can be expressed as a vector:

$$DC1 = [XC(JX,JY,JZ) - XC(JX-1,JY,JZ)]\hat{i} + [YC(JX,JY,JZ) - YC(JX-1,JY,JZ)]\hat{j} + [ZC(JX,JY,JZ) - ZC(JX-1,JY,JZ)]\hat{k}$$

$$= DXC\hat{i} + DYC\hat{j} + DZC\hat{k}$$

The two area vectors of opposing faces and the vector between the centers of these faces can be used to compute a volume (not necessarily the true volume of the element). For the example given here, this calculation is

$$VOL = \tfrac{1}{2}[A1(JX,JY,JZ) + A1(JX-1,JY,JZ)] \cdot DC1$$

i.e., the dot product between the average of the opposing face area vectors and the vector connecting the centers of these faces gives a volume. For a rectangular parallelepiped, this calculation would indeed be the true volume. Any pair of opposing faces could be used for a rectangular parallelepiped, and the result would be the same; or the average of this process for the three pairs could be used to recover the true volume of a rectangular parallelepiped. For an arbitrary element, the volume is *defined* by the average for the three pairs, ie.,

$$VOL = \tfrac{1}{3}\{[A1(JX) + A1(JX-1)] \cdot DC1 + [A2(JY) + A2(JY-1)] \cdot DC2 + [A3(JZ) + A3(JZ-1)] \cdot DC3\}$$

where the subscript notations for the area vectors have been shortened.

These definitions of area vectors and volumes are certainly not exact for arbitrary elements, but are not too far off for small volumes. For regular elements, they are exact and they are computationally efficient. These schemes are similar to those used in many modern numerical computational programs.

A final comment on the nomenclature used in the preceding discussion is necessary. The equations appearing in VOLGRD are in terms of the components of the various vectors; the vectors themselves never occur. For example, the symbol D1 has been used in this discussion for one of the diagonals. In VOLGRD only the components of D1, denoted DX1, DY1, DZ1, are used. Similarly, A1 never appears but AX1, AY1, and AZ1 are used, and so on. Hopefully, this difference in notation will not cause any confusion.

4. BC

This subroutine enforces the boundary conditions on the computed flow. Typical boundary conditions for a body in an unbounded free stream are free-stream conditions far away from the body and a specified normal velocity at the body surface (usually zero). Implementing conditions like these on a computational mesh of finite extent requires that these conditions be handled in a special way. The technique used here is based on the concept of phantom points, i.e., points which are physically outside of the computational mesh.

Phantom points are the centers of volume elements which lie just outside of the computational mesh. Figure 1 shows a (coarse) grid for an axisymmetric body which would be useful for zero incidence or angle-of-attack problems (hence, the JZ lines go halfway around the body). The first volume elements in the mesh to be encountered as one proceeds downstream are those having JX = 2, since the volume element is designated by a downstream corner (see Fig. C-3a). If one referred to a volume (1, JY, JZ) this would be immediately upstream of the mesh. The center of this volume is a phantom point. Now, referring to Figs. C-3b and C-3e, a volume denoted by (JX, JY, 1) would be just to the left (or counter clockwise) of the grid volumes above the body, since the volume element is designated by a clockwise corner. The center of this volume is also a phantom point. In a similar manner, the centers of volumes (JX, JY, 8), where NNZ + 1 = 8, are phantom points just clockwise of the lower grid volumes (Figs. C-3b and C-3d.). The centers of volumes (NNX+1, JY, JZ) are phantom points just downstream of the last streamwise

volumes (Fig. 1a). Finally, the centers of volumes (JX, 1, JZ) are phantom points just inside the body, since volumes are designated by radially outward corners (Fig. C-3c).

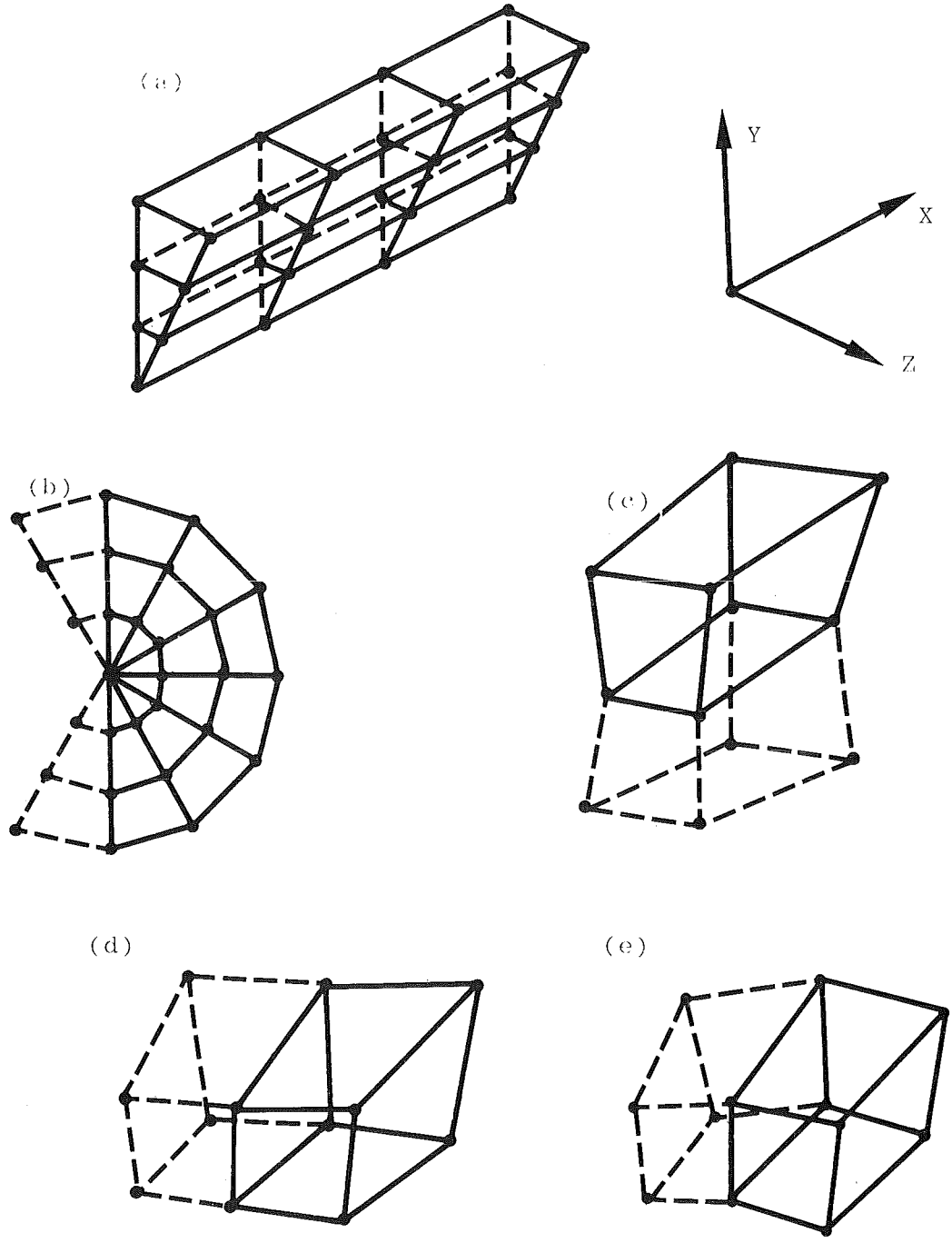


Figure C-3. Volume elements and phantom points.

The phantom points are used in the following way. At planes of symmetry (for example, lines JZ = 1 and JZ = 7 in Fig. 1) and body surfaces (JY = 1), mirror conditions are applied. Normally on a solid body surface or on a plane of symmetry, we would simply require that $\bar{q} \cdot \hat{n} = 0$ ($\bar{q}$ = velocity vector, $\hat{n}$ = normal to surface). However, in the formulation of this program we do not know conditions on surfaces but only at the centers of volume elements. Consequently, we only know conditions slightly off the body or to one side of a plane of symmetry. Mirror conditions dictate that the velocity vector at the phantom point inside the body surface or on the other side of the symmetry plane has a normal component equal and opposite to the normal component of velocity at the neighboring true volume center. One might say that the phantom point velocity vectors are "mirror images" of the neighboring true velocity vectors, hence the name mirror conditions. Then, for small volume elements these conditions essentially impose a zero normal velocity on the surface in question. Implementing the mirror condition leads to Eq. (9) of Ref. 1:

$$\bar{q}_{\text{phantom point}} = \bar{q}_{\text{true}} - 2\left(\frac{S \cdot \bar{q}_{\text{true}}}{S \cdot S}\right)S$$

where S is the area vector of the surface in question. In subroutine BC, these mirror conditions are imposed at the phantom points. The density, energy, and pressure at these same phantom points are determined by imposing a zero normal gradient condition; i.e., R, E, and P do not change from the phantom point to the neighboring volume center. Thus, we have R(JX, 1, JZ) = R(JX, 2, JZ), R(JZ, JY, 1) = R(JX, JY, 2), and R(JX, JY, NNZ+1) = R(JX, JY, NNZ), and so on.

At the downstream boundary (JZ = NNX) the zero normal gradient condition is imposed on all the flow quantities R, E, P, and RU, RV, RW. Thus, we have R(NNX+1, JY, JZ) = R(NNZ, JY, JZ) and so on. This condition may not be physically true, especially if the downstream boundary is not far away from the body, but it has been found to work virtually as well as more correct (and much more complicated) outflow conditions and, therefore, has been retained in the program.

5. UPDATE

This subroutine essentially evaluates Eq. (7) of Ref. 1 [more appropriately, its alternate form, Eq. (11)]. This equation, in two parts, is the net result of applying the predictor-corrector technique to this problem. In a *very* schematic way, both parts of Eq. 7 [or Eq. (11)] look like

$$G_{\text{new}} = G_{\text{old}} - \text{constant} * \Sigma F \cdot S \qquad \text{(C-1)}$$

The term G represents any of the quantities R, RU, RV, RW, E, and the term F represents the fluxes correponding to these quantities (see Section 3.0). The quantity F • S is referred to as the flux accumulator.

Suppose G = R. The flux accumulator for R is the variable DR. The constant in Eq. (C-1) involves Δt and volume, i.e, DT and VOL, and is called DTSVOL. So, if we write Eq. (C-1) above for G = R, we obtain

$$R(JX,JY,JZ) = R(JX,JY,JZ) - DTSVOL(JX) * DR(JX,JY,JZ) \qquad (C\text{-}2)$$

which is precisely one of the equations in subroutine UPDATE. This equation *updates* the old value of R by the net mass which has entered the volume during the time interval Δt, hence, the subroutine name.

Now, Eq. (C-1) represents either part of Eq. (7), Ref. 1; i.e., the predictor and the corrector steps have the same form. Equation (C-2) above is valid for either step also. Which step is actually being computed at some instant in the program depends on a complicated manipulation of indices and counters which takes place in several subroutines. These manipulations are beyond the scope of the present discussion, other than to observe that they occur.

6. DTCAL

This subroutine determines the time step, Δt, to be used in the predictor-corrector calculations. It would be desirable for this time step to be as large as possible so that a solution could be obtained quickly and, thus, economically. However, the computational scheme is unstable for time steps greater than some critical value. By unstable we mean the solution diverges as time increases rather than converges as we desire. The critical value is determined by the Courant-Friedrichs-Lewy stability criterion*, which says that the maximum allowable time step is the minimum of

$$\Delta t = \frac{V}{|\bar{q} \cdot S| + c|S|}$$

throughout the mesh, where V is the volume of an element, S and $\bar{q}$ are area and velocity vectors, respectively, and c is the local speed of sound. This criterion essentially says that the

*Anderson, J. D. *Modern Compressible Flow*. McGraw-Hill, New York, 1982, pp. 312-313.

time step can be no greater than the least time it takes for an acoustic signal to travel from one volume center to the next, which is certainly physically reasonable.

DTCAL proceeds by computing at a given time

$$\frac{V}{|\bar{q} \cdot S| + c|S|}$$

for every volume in the mesh and searching for the minimum value. It is interesting and instructive to look at a few details of how this quantity (called DT) is actually calculated.

Careful study of DTCAL shows that, in the notation of this program but leaving off the subscripts, DT is determined by the expression

$$DT = \frac{VOL * R}{|RQDA| + \left(RQDRQ * ADA * \frac{GS2}{E/P - RGM1} \right)^{1/2}}$$

where

$$RQDA = \varrho \bar{q} \cdot \bar{A}$$

$$RQDRQ = \varrho \bar{q} \cdot \varrho \bar{q} = \varrho^2 q^2$$

$$ADA = \bar{A} \cdot \bar{A} = A^2$$

and GS2/[(E/P) - RGM1] = $1/M^2$, if the definitions of GS2, RGM1, E, and P are used. This expression for DT can then be written as

$$DT = \frac{\varrho * VOL}{\varrho|\bar{q} \cdot \bar{A}| + (\varrho^2 q^2 A^2 / M^2)^{1/2}}$$

But M = q/c, so this reduces to

$$DT = \frac{VOL}{|\bar{q} \cdot \bar{A}| + cA}$$

which is the same as the equation for Δt.

7. FLUX

The subroutines FLUXX, FLUXY, and FLUXZ (collectively referred to as FLUX) compute the fluxes of the various flow variables across each pair of surfaces of a volume

element. Thus, FLUXX computes the flux across the two A1 surfaces, FLUXY computes the A2 fluxes, and FLUXZ determines the transport across the two A3 surfaces. These routines also increment the flux accumulators, which symbolically are F • S, with appropriate care for the signs. The three routines are identical in form and differ only in the subscripts and variables used.

The computations in FLUX are straightforward. Central to these calculations is the definition of the flux of something:

$$\text{flux} = \int_S \left(\frac{\text{something}}{\text{unit volume}} \right) \bar{q} \cdot d\bar{A}$$

which, for planar surfaces with constant flow properties, reduces to

$$\text{flux} = \left(\frac{\text{something}}{\text{unit volume}} \right) \bar{q} \cdot \bar{A}$$

The "something" here is the dimensionless mass, (X, Y, Z) momentum, or energy, and is simply R, RU, RV, RW, and E. The area used in the dot product depends on the subroutine being used. Thus,

$$\begin{aligned} \bar{q} \cdot \bar{A} = QDA &= (RU*AX1 + RV*AY1 + RW*AZ1)/R \quad \text{in} \quad FLUXX, \\ &= (RU*AX2 + RV*AY2 + RW*AZ2)/R \text{ in FLUXY, and} \\ &= (RU*AX3 + RV*AY3 + RW*AZ3)/R \quad \text{in} \quad FLUXZ \end{aligned}$$

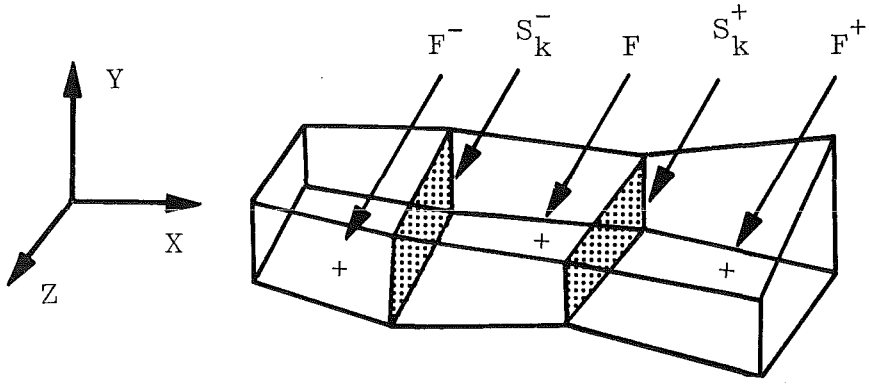
The equations in FLUX then follow from combinations of the "something per unit volume" and the QDA terms.

What is not so straightforward in FLUX (and in some of the other subroutines as well) is the way in which the indices (or subscripts) and counters are manipulated. For instance, in FLUXX we find the following indices and DO LOOP parameters being used: KODE, JJ, JX1, JX2, NEX, NNY, NNZ, JS, JX, JY, JZ, and JT. FLUXY and FLUXZ have similar quantities. Actually, this rather complicated situation is the result of economizing the computations. The solution process involves performing a set of calculations, namely those in Eq. (11), Ref. 1, for every volume element in the computational mesh and at every time step. Within Eq. (11) we see that there is a need for repeated determinations of F • S, although the subscripts and superscripts appearing in Eq. (11) suggest that different values of F and S are to be used in the various terms. This is indeed true, although with careful

study of the overall computational process we find that certain calculations are repeated exactly. For example, at a given time step the quantity $F \cdot S_k^+$ for one volume element is exactly the same (except for a sign) as $F^- \cdot S_k^-$ for a neighboring volume element. (Reference 1 and Section 4.0 may help clarify this.) In any case, the important idea is that certain types of calculations, namely F • S, are done very many times in this program and some are repeated exactly. The design of the FLUX subroutines is such that the equations for mass, momentum, and energy flux can be used for every volume, every time step, and each form of F • S in Eq. (11). The versatility of these subroutines comes from the manipulation of the indices and counters.

A number of subroutines besides FLUX are involved in these manipulations; however, much of this activity does take place in FLUX, and a feel for what goes on can be obtained by studying some of the details in the FLUX routines, which we shall now do (on a very limited scale).

FLUX evaluates the quantity F • S. The particular values of F and S used in any given calculation depend on which volume is being considered, which face of the the volume is being considered, the time step, and which step of the predictor-corrector sequence is being executed. Suppose we fix the volume, face, and time step in question. A representative situation is sketched below.



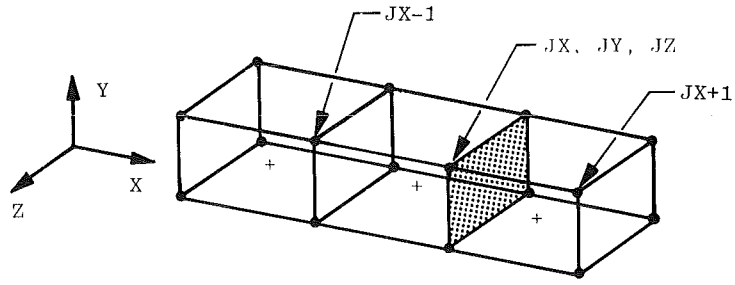
The volume in question is the middle volume, the face of interest is the shaded one labelled S_k^+. With the time, volume, and face fixed, we only need to worry about which step in the predictor-corrector cycle we are in. The two steps in the predictor-corrector cycle are represented by the two equations in Eq. (11), Ref. 1. Looking at these equations we see that we use $F \cdot S_k^+$ in one step and $\bar{F}^+ \cdot S_k^+$ in the other. (Don't worry about the bar over F^+ right now.) So, in one step of the predictor-corrector, we use the value of F "behind" the face in question; in the other step we use the value of F "ahead". This can be accomplished by using the same equation for $F \cdot S$ and simply changing the subscript of F in the k direction. (Remember, the subscripts of a flow quantity refer to the volume in which it occurs.) If the k direction, in this example, is in the X direction, then we increment JX to go from one side of the face to the other. Such a procedure must be done for all six faces of this volume. The face labelled S_k^- in the sketch would use F^- and F, and so on.

Besides determining $F \cdot S$ at each face, FLUX sums or accumulates the net flux into each volume by incrementing the flux accumulators DR, DRU, DRV, DRW, and DE. However, care must be taken with the signs. For instance, in the sketch, the product $F \cdot S_k^+$ might represent the flux *out* of the middle volume or the flux *into* the right-hand volume. For one volume this number is positive, for the other it is negative. The subscripts and signs must be carefully matched to properly sum or accumulate the fluxes.

The foregoing discussion has established the reasons for the multitude of indices and index operations in the program. The following brief summary shows how some of the subscripts provide a means to switch from one side of a surface to another and to switch from one volume to another.

Each of FLUXX, FLUXY, and FLUXZ has a dimensioned variable KODE (JJ) whose 16 values are specified in a DATA statement in each subroutine. The value of JJ is established in STEP and is transferred through the labelled COMMON CONST. JJ tells FLUX which phase of the predictor-corrector sequence to compute by picking a value of KODE. KODE, in turn, sets JX1, JY1, or JZ1 depending on which FLUX subroutine we are using. JX1 (or JY1 or JZ1), in turn, sets the value of JT. JT sets the value of JS, and JS tells FLUX whether to use the flow value on one side or the other of a surface in evaluating $F \cdot S$ and whether to give this quantity a plus or minus sign when it is added to the flux accumulator (schematically represented here by D).

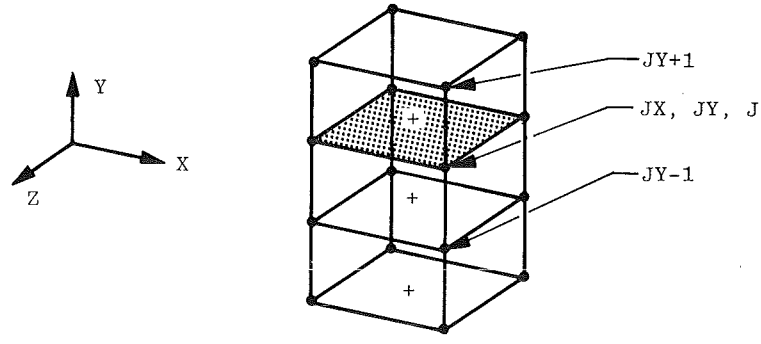
Figure C-4 presents the values of and relations between the various indices as the calculations in FLUX proceed. Also depicted are representative volume elements and reference surfaces to illustrate the predictor-corrector selection and flux summation.



JX1 = 1 → JT = 2, JS = JX, D(JS) = D(JS) - F(JS-1) + F(JS);
q upstream of A1

JX1 = 2 → JT = 3, JS = JX-1, D(JS) = D(JS) - F(JS) + F(JS+1);
q downstream of A1

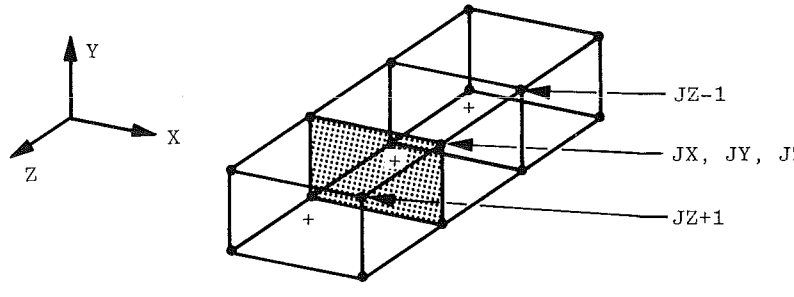
a. FLUXX



JY1 = 1 → JT = 2, JS = JY, D(JS) = D(JS) - F(JS-1) + F(JS);
q below A2

JY1 = 2 → JT = 3, JS = JY-1, D(JS) = D(JS) - F(JS+1);
q above A2

b. FLUXY



JZ1 = 1 → JT = 2, JS = JZ, D(JS) = D(JS) - F(JS-1) + F(JS);
q counterclockwise of A3

JZ1 = 2 → JT = 3, JS = JZ-1, D(JS) = D(JS) - F(JS) + F(JS+1);
q clockwise of A3

c. FLUXZ

Figure C-4. FLUX calculations and representations.