

E 801782

②

DTIC FILE COPY

AFATL-TR-88-117, VOL IV

Program EAGLE User's Manual

Vol IV - Multiblock Implicit, Steady-State Euler Code

AD-A204 144

Jon S Mounts
Dave M Belk
David L Whitfield

AERODYNAMICS BRANCH
AEROMECHANICS DIVISION

SEPTEMBER 1988

DTIC
ELECTE
OCT 12 1988
S H

INTERIM REPORT FOR PERIOD OCTOBER 1986 - SEPTEMBER 1988

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE ARMAMENT LABORATORY

Air Force Systems Command ■ United States Air Force ■ Eglin Air Force Base, Florida

88 1011 223

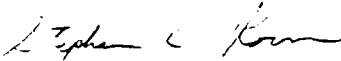
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States 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 as 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.

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

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



STEPHEN C. KORN
Technical Director, Aeromechanics Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify AFATL/FXA, Eglin AFB FL 32542-5434.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE			Approved for public release; distribution unlimited		
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Aerodynamics Branch Aeromechanics Division			6b. OFFICE SYMBOL (If applicable) AFATL/FXA		7a. NAME OF MONITORING ORGANIZATION Aerodynamics Branch Aeromechanics Division
6c. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin Air Force Base, FL 32542-5434			7b. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin Air Force Base, FL 32542-5434		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)				N/A	
			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO. 62602F	PROJECT NO. 2567	TASK NO. 03
			WORK UNIT ACCESSION NO. 08		
11. TITLE (Include Security Classification) Program EAGLE User's Manual, Volume IV - Multiblock Implicit, Steady-State Euler Code					
12. PERSONAL AUTHOR(S) Jon S. Mounts, Dave M. Belk, David Whitfield					
13a. TYPE OF REPORT Interim		13b. TIME COVERED FROM Oct 86 to Sep 88		14. DATE OF REPORT (Year, Month, Day) September 1988	
15. PAGE COUNT 134					
16. SUPPLEMENTARY NOTATION Availability of this report is specified on verso of front cover. This volume is a joint Contractor and AFATL effort. Therefore, it is in contractor format.					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Inviscid Flow Solver		
			Computational Aerodynamics		
			Program EAGLE-Flow Solver		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
This report documents the theory and usage of the Program EAGLE-Flow Solver; a multiblock implicit, steady-state Euler algorithm used for the aerodynamic analysis of advanced arbitrarily shaped airframe configurations in freestream and interference flowfields.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT			21. ABSTRACT SECURITY CLASSIFICATION		
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS			Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL John R. Cipola			22b. TELEPHONE (Include Area Code) (904) 882-3124		22c. OFFICE SYMBOL AFATL/FXA

PREFACE

Program EAGLE (Eglin Arbitrary Geometry Implicit Euler) - Flow Solver is specifically designed to interface with the Program EAGLE - Numerical Grid Generation System (which provides the necessary computational grid, Volumes I-III) to solve for the freestream aerodynamic characteristics of weapon airframe configurations, as well as the interference flowfield associated with in-carriage configurations.

Program EAGLE - Flow Solver has been a joint development effort between the Air Force Armament Laboratory's (AFATL) Aerodynamics Branch (FXA) and Mississippi State University's (MSU) Department of Aerospace Engineering. The principal investigators of the flow theory have been Dr David L. Whitfield of MSU and Dr Dave M. Belk and Mr L. Bruce Simpson of the Computational Fluid Dynamics Section (AFATL/FXA). The EAGLE - Flow Solver has been developed for public release by Jon S. Mounts, Dr Dave M. Belk and L. Bruce Simpson of the CFD Section (AFATL/FXA). The program manager for the development of Program EAGLE has been Dr Lawrence E. Lijewski.



Accession For	
NTIS GFA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution _____	
Availability Codes	
Dist	Avail. and/or Special
A-1	

TABLE OF CONTENTS

Section	Title	Page
I.	INTRODUCTION	1
II.	THEORETICAL ASPECTS	3
	1. Governing Equations	3
	2. Flux-Vector Splitting (Steger-Warming)	5
	3. Flux-Difference Splitting (Roe)	13
	4. Boundary Conditions	18
III	FLOW SOLVER INPUT	28
	1. General Input (FINPUT)	28
	2. Transformation Input (ROTATE)	36
	3. Surface Specification (SURFACE)	39
	4. Boundary Conditions (BCIN)	41
IV	FLOW SOLVER DESCRIPTION	49
	1. Main Routine (MISSE)	50
	2. Subroutine AEQLU	53
	3. Subroutine BC	53
	4. Subroutine CHECK	53
	5. Subroutine CUT	54
	6. Subroutine DELQ	54
	7. Subroutine DOOM	55
	8. Subroutine DOOP	55
	9. Subroutine DPMAP	56
	10. Subroutine FARFLD	56
	11. Subroutine FJMAT	57
	12. Subroutine FLUX	57
	13. Subroutine FORCE	58
	14. Subroutine GETBC	58
	15. Subroutine GETBBC	59
	16. Subroutine GETCP	59
	17. Subroutine GRIDIN	59
	18. Subroutine IC	60
	19. Subroutine LOCALT	60
	20. Subroutine MATM	60
	21. Subroutine METRIC	61
	22. Subroutine MINMOD	61
	23. Subroutine OPSSD	62
	24. Subroutine PASTE	62
	25. Subroutine POINT	62
	26. Subroutine PUTBC	63
	27. Subroutine PUTBBC	63
	28. Subroutine PVAR	63
	29. Subroutine READ	64
	30. Subroutine RESID	64
	31. Subroutine RLVECS	65
	32. Subroutine ROTX	65
	33. Subroutine ROTY	66
	34. Subroutine ROTZ	66
	35. Subroutine SETMEM	66
	36. Subroutine SETMPB	67

	37.Subroutine SOLID	67
	38.Subroutine STEP	67
	39.Subroutine SUPBEE	68
	40.Subroutine WRITE	68
V	APPLICATIONS	69
	1. NACA 0012 Airfoil	69
	2. Generic Finned Configuration	88
VI	RESULTS	98
	1. NACA 0012 Airfoil	98
	2. Generic Finned Configuration	100
	3. Blunt, Finned Body of Revolution W/Canards	105
	4. Mutual Interference Configurations	112
VII	SUMMARY	120
	REFERENCES	123

LIST OF FIGURES

Figure	Title	Page
1	Indexing of Cell Centers and Faces	9
2	Computational Coordinates Schematic for Characteristic Variable Boundary Conditions	21
3	Zero Pressure Gradient Boundary Condition	27
4	LIFTAX Representation	30
5	Positive TRANS Rotation Convention	37
6	Symmetry Case (BCTYPE = 3)	43
7	Block-to-Block Information (BCTYPE = 4)	44
8	Block-to-Block Across a Singularity (BCTYPE = 5)	45
9	Singularity on a Reflection Plane (BCTYPE = 6)	46
10	NACA 0012 Airfoil Grid	71
11	NACA 0012 Computational Region	73
12	NACA 0012 Airfoil Incidence Angle Rotation	75
13	Generic Finned Configuration	89
14	Generic Finned Configuration 4-Block Grid	90
15	Generic Finned Configuration Computational Region	96
16	NACA 0012 Airfoil - Flow Solver Output	99
17	Generic Finned Configuration - Flow Solver Output ($\alpha = 15$ Degrees)	101
18	Generic Finned Configuration - Flow Solver Output ($\alpha = 6$ Degrees)	106
19	Blunt, Finned Body of Revolution - Flow Solver Output	109
20	Multiple Body Configurations	113
21	Configuration Complexity Effects	114
22	Fin Effects on 2-Body Case	116
23	Angle of Attack Effects, 3-Body Case	117
24	Interference Effects on Fin Pressures, 3-Body Case	118

LIST OF TABLES

Table	Title	Page
1	Program EAGLE - Flow Solver NACA 0012 Airfoil Output	78

LIST OF SYMBOLS

a_{∞}	Freestream speed of sound
$A, B, C, \bar{K}$	Jacobians of the flux vectors
b	compression parameter
c	local speed of sound, airfoil chord length
CFL	Courant number
C_p	pressure coefficient
C_{p^*}	critical pressure coefficient
e	total specific energy
f, g, h	Cartesian coordinate flux vectors
F, G, H, K	Curvilinear coordinate flux vectors
I	identity operator
J	Metric Jacobian (cell volume)
l	reference length
L	limiting operator
M	Mach number
P	pressure
q	Cartesian coordinate dependent variable vector
Q	curvilinear coordinate dependent variable vector
ΔQ^n	change in dependent variable vector from time n to $n+1$
r	eigenvector
R	residual matrix
t	time
u, v, w	Cartesian velocity components
U, V, W	Contravariant velocities
V_{∞}	freestream velocity
X	intermediate solution vector
α	angle of attack, jump in characteristic variable
β	side slip angle
γ	ratio of specific heats

∇	Del operator
Δ	difference operator
δ	spatial difference operator
λ	eigenvalue
σ	Roe parameter
ψ	TVD parameter (ROC)
θ_k	$K_x U + K_y V + K_z W$
$\bar{\theta}_k$	$\theta_k / \nabla K $
ρ	density
τ	transformed time variable
ξ, η, ζ	curvilinear coordinates

Subscripts

i, j, k	mesh point location
l	split flux vector
∞	freestream values
ξ, η, ζ, K	partial differentiation
r	reference value

Superscripts

l	eigenvalue number
$+$	associated with positive eigenvalues
n	time level
$-$	associated with negative eigenvalues
-1	inverse
$\wedge$	indicates dimensional quantity
j	eigenvalue number (Roe)

SECTION I

INTRODUCTION

Accurate prediction of free-flight aerodynamic characteristics is of the utmost importance to the weapon airframe designer. Mission effectiveness is highly dependent on his ability to design an airframe with desirable aerodynamic qualities to deliver the payload on target with optimum maneuvering performance. Often this is a time-consuming iterative effort coupling various aerodynamic predictive methods and expensive wind tunnel testing for airframe validation.

As a result, the Air Force Armament Laboratory has embarked on a multi-year effort to derive accurate computational fluid dynamic techniques to enhance weapon airframe design procedures and reduce wind tunnel validation testing. The first product from this effort is Program EAGLE - Flow Solver (MISSE); a multi-block, implicit, steady-state Euler algorithm. Program EAGLE has been designed to aid in the analysis of both freestream and interference aerodynamic characteristics of advanced arbitrarily shaped, finned or unfinned configurations in the compressible subsonic, transonic and supersonic Mach ranges at relatively low incidence angles.

This technical report is designed to be a user's manual for the Program EAGLE - Flow Solver. In this report an indepth discussion is given on the numerical solution of the three-dimensional Euler equations (Section II). In addition, a section

is provided that details the entire set of input required by the Flow Solver including all default values (Section III). A detailed discussion of the main routine (MISSE), each of the 39 subroutines and the interaction between all portions of the code (Section IV), is also included for the user. And finally, in Section V (Applications) a complete discussion of several aerodynamic applications is given to familiarize the user with the detailed workings of the Program EAGLE - Flow Solver.

SECTION II THEORETICAL ASPECTS

1. GOVERNING EQUATIONS

The three-dimensional, time dependent Euler equations written in strong conservation law form are

$$(\partial \hat{q} / \partial \hat{t}) + (\partial \hat{f} / \partial \hat{x}) + (\partial \hat{g} / \partial \hat{y}) + (\partial \hat{h} / \partial \hat{z}) = 0 \quad (1)$$

where

$$\hat{q} = \begin{bmatrix} \hat{\rho} \\ \hat{\rho}\hat{u} \\ \hat{\rho}\hat{v} \\ \hat{\rho}\hat{w} \\ \hat{e} \end{bmatrix}, \quad \hat{f} = \begin{bmatrix} \hat{\rho} \\ \hat{\rho}\hat{u}^2 + \hat{P} \\ \hat{\rho}\hat{u}\hat{v} \\ \hat{\rho}\hat{u}\hat{w} \\ \hat{u}(\hat{e} + \hat{P}) \end{bmatrix}, \quad \hat{g} = \begin{bmatrix} \hat{\rho} \\ \hat{\rho}\hat{u}\hat{v} \\ \hat{\rho}\hat{v}^2 + \hat{P} \\ \hat{\rho}\hat{v}\hat{w} \\ \hat{v}(\hat{e} + \hat{P}) \end{bmatrix}, \quad \hat{h} = \begin{bmatrix} \hat{\rho} \\ \hat{\rho}\hat{u}\hat{w} \\ \hat{\rho}\hat{w}^2 + \hat{P} \\ \hat{\rho}\hat{v}\hat{w} \\ \hat{w}(\hat{e} + \hat{P}) \end{bmatrix}$$

and

$$\hat{e} = (\hat{P}/\gamma - 1) + (\hat{\rho}/2)[\hat{u}^2 + \hat{v}^2 + \hat{w}^2]$$

Here $\hat{\rho}$ is the mass density of the compressible fluid, $\hat{u}$, $\hat{v}$, and $\hat{w}$ are the velocity components in the $\hat{x}$, $\hat{y}$, and $\hat{z}$ Cartesian coordinate directions, respectively, $\hat{P}$ is the pressure, and $\hat{e}$ the total specific energy of the fluid. The relationship between $\hat{e}$ and $\hat{P}$ is the result of making the perfect gas assumption, and γ is the ratio of specific heats. Carets on the quantities in Equation (1) are used to indicate that dimensional quantities are being used. One way to non-dimensionalize Equation (1) is to define the following non-dimensional quantities:

$$\begin{aligned} x &= \hat{x}/l_r, \quad y = \hat{y}/l_r, \quad z = \hat{z}/l_r, \quad t = (a_{(\infty)} \hat{t})/l_r \\ u &= \hat{u}/a_{(\infty)}, \quad v = \hat{v}/a_{(\infty)}, \quad w = \hat{w}/a_{(\infty)} \\ \rho &= \hat{\rho}/\rho_{(\infty)}, \quad e = \hat{e}/(\rho_{(\infty)} a_{(\infty)}^2), \quad P = \hat{P}/(\rho_{(\infty)} a_{(\infty)}^2) \end{aligned} \quad (2)$$

where l_r is any convenient reference length, $a = (\gamma P_{(\infty)}/\rho_{(\infty)})^{1/2}$ the

speed of sound in undisturbed gas, and ρ_∞ is the density of undisturbed gas. After some manipulation, the non-dimensional Euler equations may be written in exactly the same form as Equation (1) with all carets dropped.

To simplify numerical treatment of boundary conditions around general geometries the Euler equations are recast in terms of general boundary conforming curvilinear coordinates. The curvilinear coordinates are defined as

$$\begin{aligned}\xi &= \xi(x, y, z) \\ \eta &= \eta(x, y, z) \\ \zeta &= \zeta(x, y, z) \\ \tau &= t\end{aligned}\tag{3}$$

Applying the transformation Equation (3) to the non-dimensional Euler equations (Equation (1) with all carets dropped) gives

$$(\partial Q / \partial \tau) + (\partial F / \partial \xi) + (\partial G / \partial \eta) + (\partial H / \partial \zeta) = 0\tag{4}$$

where

$$Q = J \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}$$

$$F = J \begin{bmatrix} \rho U \\ \rho u U + \xi_x P \\ \rho v U + \xi_y P \\ \rho w U + \xi_z P \\ U(e + P) \end{bmatrix} \quad \text{and } U = \xi_x u + \xi_y v + \xi_z w$$

$$G = J \begin{bmatrix} \rho V \\ \rho u V + \eta_x P \\ \rho v V + \eta_y P \\ \rho w V + \eta_z P \\ V(e + P) \end{bmatrix} \quad \text{and } V = \eta_x u + \eta_y v + \eta_z w$$

$$H = J \begin{bmatrix} \rho W \\ \rho u W + \zeta_x P \\ \rho v W + \zeta_y P \\ \rho w W + \zeta_z P \\ W(e + P) \end{bmatrix} \quad \text{and } W = \zeta_x u + \zeta_y v + \zeta_z w$$

in the above equation, J is the Jacobian of the inverse

transformation. i.e. $\partial(x,y,z)/\partial(\xi,\eta,\zeta)$ and is given by

$$J = x_{\xi}(y_{\eta}z_{\zeta} - z_{\eta}y_{\zeta}) - y_{\xi}(x_{\eta}z_{\zeta} - z_{\eta}x_{\zeta}) + z_{\xi}(x_{\eta}y_{\zeta} - y_{\eta}x_{\zeta}) .$$

The metric quantities are given by

$$\xi_x = J^{-1}(y_{\eta}z_{\zeta} - z_{\eta}y_{\zeta}), \quad \eta_x = J^{-1}(z_{\xi}y_{\zeta} - y_{\xi}z_{\zeta}), \quad \zeta_x = J^{-1}(y_{\xi}z_{\eta} - z_{\xi}y_{\eta})$$

$$\xi_y = J^{-1}(z_{\eta}x_{\zeta} - x_{\eta}z_{\zeta}), \quad \eta_y = J^{-1}(x_{\xi}z_{\zeta} - z_{\xi}x_{\zeta}), \quad \zeta_y = J^{-1}(z_{\xi}x_{\eta} - x_{\xi}z_{\eta})$$

$$\xi_z = J^{-1}(x_{\eta}y_{\zeta} - y_{\eta}x_{\zeta}), \quad \eta_z = J^{-1}(y_{\xi}x_{\zeta} - x_{\xi}y_{\zeta}), \quad \zeta_z = J^{-1}(x_{\xi}y_{\eta} - y_{\xi}x_{\eta})$$

The details of this transformation are given in Reference 1.

2. FLUX-VECTOR SPLITTING (Steger-Warming)

Hyperbolic partial differential equations, such as the Euler equations, are characterized by the existence of a limited domain of dependence. The solution at a point does not depend on every other point in the field; this means that information travels only in certain characteristic directions. Numerical schemes intended to solve hyperbolic equations are usually enhanced by insuring that the numerical method propagates information in the direction specified by the partial differential equation. This can be done by using an upwind method, or one in which the difference operator is taken in the direction from which the information should come. Stability properties are often improved by upwinding, and it is usually unnecessary to add smoothing terms or artificial viscosity to an upwind method.

The three-dimensional Euler equations, Equation (4), are a hyperbolic system of five equations and hence have five characteristic velocities in each of the three spatial directions. These characteristic velocities are determined from the quasilinear form of Equation (4),

$$(\partial Q / \partial \tau) + A(\partial Q / \partial \xi) + B(\partial Q / \partial \eta) + C(\partial Q / \partial \zeta) = 0 \quad (5)$$

where the matrices A, B, and C are given by

$$A = (\partial F / \partial Q), \quad B = (\partial G / \partial Q), \quad C = (\partial H / \partial Q). \quad (6)$$

The eigenvalues of A are the characteristic velocities in the ξ direction, the eigenvalues of B are the characteristic velocities in the η direction, and similarly, the eigenvalues of C are the characteristic velocities in the ζ direction.

Since F, G, and H are identical except that where ξ appears in F, η will appear in G and ζ will appear in H, extra writing can be avoided by letting K represent either F, G, or H, when k represents either ξ , η or ζ respectively. Then define

$$\bar{K} = (\partial K / \partial Q) \quad (7)$$

which corresponds to A, B, or C depending on the meaning of K. The general flux Jacobian matrix $\bar{K}$ and its eigenvalues are derived in Reference 2. The eigenvalues of the matrix $\bar{K}$ are

$$\begin{aligned} \lambda_k^1 &= \lambda_k^2 = \lambda_k^3 = (k_x u + k_y v + k_z w) = \theta_k \\ \lambda_k^4 &= \theta_k + c |\nabla k| = \theta_k + c(k_x^2 + k_y^2 + k_z^2)^{1/2} \\ \lambda_k^5 &= \theta_k - c |\nabla k| \end{aligned} \quad (8)$$

where c is the speed of sound.

It is possible to split the flux vector K into three parts, one corresponding to each of the distinct eigenvalues of $\bar{K}$ given above (for the details of this splitting, see Reference 2). The flux vector K is then written as

$$K = \lambda_k^1 K_1 + \lambda_k^4 K_4 + \lambda_k^5 K_5 \quad (9)$$

where

$$K_1 = J(\gamma-1)/\gamma \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ (\rho/2)(u^2 + v^2 + w^2) \end{bmatrix} \quad (10a)$$

$$K_4 = J/(2\gamma) \begin{bmatrix} \rho \\ \rho u + \rho c \bar{k}_x \\ \rho v + \rho c \bar{k}_y \\ \rho w + \rho c \bar{k}_z \\ e + P + \rho c \bar{\theta}_k \end{bmatrix} \quad (10b)$$

$$K_5 = J/(2\gamma) \begin{bmatrix} \rho \\ \rho u - \rho c \bar{k}_x \\ \rho v - \rho c \bar{k}_y \\ \rho w - \rho c \bar{k}_z \\ e + P - \rho c \bar{\theta}_k \end{bmatrix} \quad (10c)$$

and

$$\bar{k}_x = (k_x/|\nabla k|) = k_x/(k_x + k_y + k_z)^{1/2} \quad (10d)$$

$$\bar{k}_y = (k_y/|\nabla k|) \quad (10e)$$

$$\bar{k}_z = (k_z/|\nabla k|) \quad (10f)$$

$$\bar{\theta}_k = \bar{k}_x u + \bar{k}_y v + \bar{k}_z w. \quad (10g)$$

The sign of λ_k^1 in Equation (8) determines from which direction information should be used to determine the corresponding portion of flux, K_i for $i = 1, 4$, and 5 .

The discussion above provides a rationale for writing the flux vector K as the sum of two vectors, K^+ and K^- ;

$$K = K^+ + K^- \quad (11)$$

K^+ is associated with the eigenvalues that have positive signs and K^- is associated with the eigenvalues that have negative signs. The specific method used to calculate K^+ and K^- will be discussed in the next section.

a. Finite Volume Discretization

A finite volume discretization of Equation (4) balances the increase of the conserved quantity in a computational cell, or volume, with the flux of the quantity through the surface of the cell. Figure (1) depicts a portion of the computational domain with a typical cell labeled. Assume the dependent variables are constant in the interior of cell (i, j, k) , and that the flux vectors F , G , and H are constant over the constant ξ , η , and ζ surfaces of the cell, respectively. Then, an implicit discretization of Equation (4) is

$$\begin{aligned} & (Q^{n+1}_{i,j,k} - Q^n_{i,j,k})\Delta\xi\Delta\eta\Delta\zeta \\ & + (F^{n+1}_{i+1/2,j,k} - F^{n+1}_{i-1/2,j,k})\Delta\eta\Delta\zeta\Delta\tau \\ & + (G^{n+1}_{i,j+1/2,k} - G^{n+1}_{i,j-1/2,k})\Delta\xi\Delta\zeta\Delta\tau \\ & + (H^{n+1}_{i,j,k+1/2} - H^{n+1}_{i,j,k-1/2})\Delta\xi\Delta\eta\Delta\tau = 0 \end{aligned} \quad (12)$$

This can be written as

$$\Delta Q^n + \Delta\tau(\delta_\xi F^{n+1} + \delta_\eta G^{n+1} + \delta_\zeta H^{n+1}) = 0 \quad (13)$$

where δ_ξ , for example, is defined by

$$\delta_\xi F_{i,j,k} = (1/\Delta\xi)[F_{i+1/2,j,k} - F_{i-1/2,j,k}],$$

and δ_η and δ_ζ are defined analogously. Letting F^+ , F^- , G^+ , G^- ,

H^+ and H^- be the split flux vectors for F , G , and H as given by Equation (11), and the implicit split flux discretization is given by

$$\Delta Q^n + \Delta\tau[\delta_\xi(F^+ + F^-)^{n+1} + \delta_\eta(G^+ + G^-)^{n+1} + \delta_\zeta(H^+ + H^-)^{n+1}] = 0 \quad (14)$$

The fluxes are nonlinear functions of the dependent variables, and must be linearized to obtain a system of equations that can easily be solved. Beam and Warming³ and Briley and McDonald⁴ have used the linearization

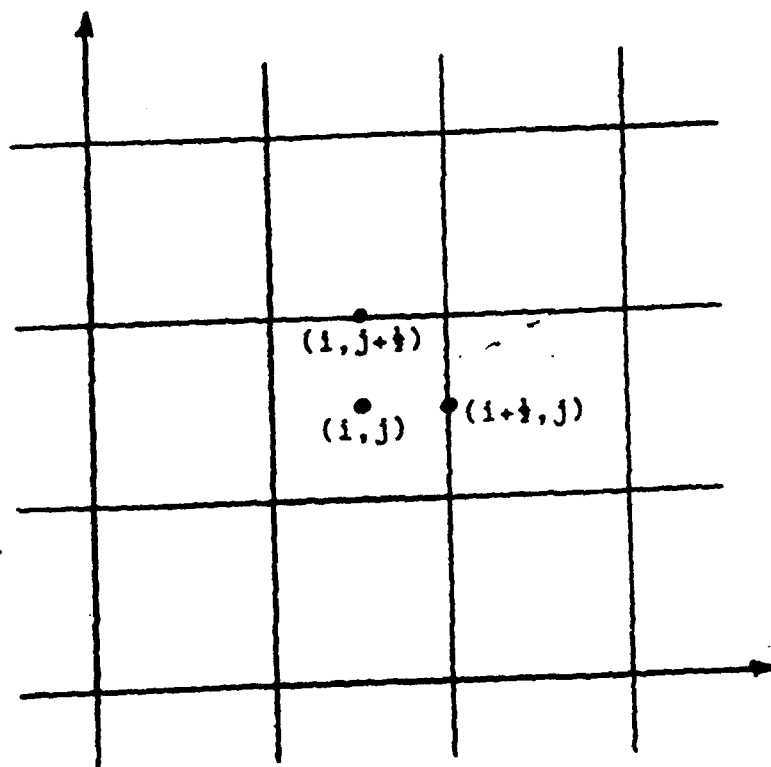


Figure 1. Indexing of Cell Centers and Faces

$$K^{n+1} = K^n + \bar{K}^n \Delta Q^n + O(\Delta T^2), \quad (15)$$

where $\bar{K}^n = (\partial K / \partial Q)^n$ for a typical flux term K . The flux Jacobian matrices required to linearize Equation (14) are

$$\begin{aligned} A^+ &= (\partial F^+ / \partial Q)^n, & A^- &= (\partial F^- / \partial Q)^n, \\ B^+ &= (\partial G^+ / \partial Q)^n, & B^- &= (\partial G^- / \partial Q)^n, \\ C^+ &= (\partial H^+ / \partial Q)^n, & C^- &= (\partial H^- / \partial Q)^n \end{aligned} \quad (16)$$

and are derived in Reference 1. Using this linearization, Equation (14) becomes

$$\begin{aligned} [I + \Delta T (\delta_\xi^i A^+ + \delta_\xi^i A^- + \delta_\eta^i B^+ + \delta_\eta^i B^- + \delta_\zeta^i C^+ + \delta_\zeta^i C^-)] \Delta Q^n \\ = -\Delta T (\delta_\xi^e F^n + \delta_\eta^e G^n + \delta_\zeta^e H^n) \end{aligned} \quad (17)$$

where a distinction has been made between the implicit spatial difference operators and the explicit spatial difference operators by using superscripts i and e respectively. The dots are used to indicate that the difference operators apply to the product of the Jacobian matrices with ΔQ^n . If δ_ξ^e , δ_η^e and δ_ζ^e are chosen such that

$$\delta_\xi^e = (\partial / \partial \xi) + O(\Delta \xi^2), \quad (18a)$$

$$\delta_\eta^e = (\partial / \partial \eta) + O(\Delta \eta^2), \quad (18b)$$

$$\delta_\zeta^e = (\partial / \partial \zeta) + O(\Delta \zeta^2), \quad (18c)$$

and δ_ξ^i , δ_η^i and δ_ζ^i are chosen such that

$$\delta_\xi^i = (\partial / \partial \xi) + O(\Delta \xi), \quad (18d)$$

$$\delta_\eta^i = (\partial / \partial \eta) + O(\Delta \eta), \quad (18e)$$

$$\delta_\zeta^i = (\partial / \partial \zeta) + O(\Delta \zeta), \quad (18f)$$

then substitution into Equation (17) yields a method that is second order accurate in space if $\Delta \xi$, $\Delta \eta$, and $\Delta \zeta$ are $O(\Delta T)$ and first-order accurate in time when used with Equation (17). The method has second order spatial accuracy when converged to

steady-state since ΔQ approaches 0.

For this algorithm, the implicit spatial differences need only be first order accurate and are evaluated using a one-point dependent variable extrapolation; however, the explicit differences in Eqs. (18) obtain second order accuracy by evaluating the flux at cell faces from two-point dependent variable extrapolations.

F^+ is evaluated using a two-point dependent variable extrapolation from the "positive" side and a two-point dependent variable extrapolation from the "negative" side for F^- . For a particular cell face, the scheme uses

$$Q_{j+1/2}^+ = (3/2)Q_j - (1/2)Q_{j-1} \quad (19a)$$

to calculate a set of "positive" eigenvalues, $\lambda^l(Q^+)$, and left split fluxes, $F_l(Q^+)$, and uses

$$Q_{j+1/2}^- = (3/2)Q_{j+1} - (1/2)Q_{j+2} \quad (19b)$$

to calculate a set of "negative" eigenvalues, $\lambda^l(Q^-)$, and right split fluxes, $F^-(Q^-)$. The flux at the cell face is then set to

$$F^+ = \sum_l (1/2)[\lambda^l(Q^+) + |\lambda^l(Q^+)|] F_l(Q^+) \quad (20a)$$

and

$$F^- = \sum_l (1/2)[\lambda^l(Q^-) - |\lambda^l(Q^-)|] F_l(Q^-). \quad (20b)$$

Note that if a "positive" and "negative" eigenvalue have different signs, then the corresponding split flux may either be summed from both sides, if the sign changes from positive to negative, or from neither side, if the sign changes from negative to positive.

This technique seems to require less storage and executes more quickly than previous methods. It also totally eliminates

upstream propagation of information in supersonic regions. Experience with this method has shown that the dispersive effects downstream of shocks, characteristic of previous methods, is also reduced.

b. Approximate Factorization

The finite volume scheme described by Equation (17) is not practical to use because the system of algebraic equations that are generated have a very large bandwidth. There are many possible approximate factorizations of this method that can be used to make the solution process easier, as noted by Steger and Warming⁵, for example. Several factorizations have been recently applied by Whitfield⁶ and by Anderson, Thomas, and Whitfield⁷. The approximate factorization used here is given by

$$\begin{aligned} [I + \Delta T (\delta_{\xi}^i A^+ + \delta_{\eta}^i B^+ + \delta_{\zeta}^i C^+)] \\ [I + \Delta T (\delta_{\xi}^i A^- + \delta_{\eta}^i B^- + \delta_{\zeta}^i C^-)] \Delta Q^n = -\Delta T R^n \end{aligned}$$

$$\text{where } R^n = \delta_{\xi}^e F^n + \delta_{\eta}^e G^n + \delta_{\zeta}^e H^n.$$

This two-factor scheme is very attractive in that it consists of the solution of a sparse block lower triangular system followed by the solution of a sparse block upper triangular system of equations given by

$$[I + \Delta T (\delta_{\xi}^i A^+ + \delta_{\eta}^i B^+ + \delta_{\zeta}^i C^+)] X^1 = -\Delta T R^n \quad (21a)$$

$$[I + \Delta T (\delta_{\xi}^i A^- + \delta_{\eta}^i B^- + \delta_{\zeta}^i C^-)] \Delta Q^n = X^1 \quad (21b)$$

Solution of (21a) is done by a simple forward substitution and solution of (21b) by a simple back substitution. A stability analysis for several factorization schemes was presented by Anderson, et al., in Reference 8. This two-factor

scheme was found to be least sensitive to Courant number and was stable for all Courant numbers up to 35, which was the maximum investigated.

3. FLUX-DIFFERENCE SPLITTING (Roe)⁹

The flux-vector split scheme due to Steger and Warming⁵ described in Section 2.2 has been shown to capture shocks in as few as two mesh points. However, a flux-difference split scheme due to Roe¹⁰ has been shown to capture shocks in as few as zero mesh points. Bram van Leer, et.al.¹¹ recently demonstrated that only one-fourth to one-half as many points were required to resolve a shear layer using a first order Roe scheme as compared to the number of points required by a second order flux-vector split scheme. To allow extension of the current capabilities of the Flow Solver to a partial or full Navier-Stokes code, a flux-difference split scheme has been implemented.

a. Approximate Riemann Solver and Roe Averaging

The one-dimensional Euler equations are considered for simplicity since in the one-dimensional analysis the waves move normal to the cell interface. In a finite-volume formulation for multi-dimensional flow the assumption is made that all waves move normal to the interfaces and the approximate Riemann solver is applied to each interface based on this assumption.

The one-dimensional Euler equations are:

$$(\partial Q / \partial t) + (\partial F / \partial x) = 0 \quad (22)$$

Roe's idea was to approximate the Riemann problem by obtaining exact solutions to an approximate equation, such as:

$$(\partial Q / \partial t) + \bar{A}[Q^i, Q^{i+1}](\partial Q / \partial x) = 0 \quad (23)$$

where, $\bar{A}[Q^i, Q^{i+1}]$, is Roe's matrix.

The constant matrix, $\bar{A}[Q^i, Q^{i+1}]$, is to be constructed so as to represent local conditions. Roe¹⁰ required the matrix to have the following set of properties, called Property U since it is intended to insure uniform validity across discontinuities:

- (i) It constitutes a linear mapping from the vector space Q to the vector space F .
- (ii) As Q^i approaches Q , $\bar{A}[Q^i, Q^{i+1}]$ approaches $A[Q]$, where $A = (\partial F / \partial Q)$.
- (iii) For any Q^i, Q^{i+1} : $\bar{A}[Q^i, Q^{i+1}] * (Q^{i+1} - Q^i) = F^{i+1} - F^i$.
- (iv) The eigenvectors of A are linearly independent.

Once the matrix A is obtained, then its eigenvalues λ^j are the wavespeeds of the Riemann problem and the right eigenvector, r^j , is proportional to the change in dependent variables, dQ , across each characteristic curve associated with a specific eigenvalue λ^j . Therefore there are two ways of obtaining the flux vector at $i+1/2$.

$$\bar{F}_{i+1/2} = f_i + \sum_{j=1}^m \alpha_j \lambda^{-(j)} r^{(j)} \quad (24)$$

where the $(-)$ superscript indicates that the summation includes only those characteristic curves that have negative slope, that is, only those associated with negative eigenvalues. Another way is

$$\bar{f}_{i+1/2} = f_{i+1} - \sum_{j=1}^m \alpha_j \lambda^{(+j)} r^{(j)} \quad (25)$$

where the (+) superscript indicates a summation only over positive eigenvalues. An averaging of these two flux vector representations yields

$$\bar{f}_{i+1/2} = (1/2)[f_i + f_{i+1}] - (1/2) \sum_{j=1}^m \alpha_j |\lambda^{(j)}| r^{(j)} \quad (26)$$

Equation (26) may also be written as

$$\bar{f}_{i+1/2} = (1/2)[f_i + f_{i+1}] - (1/2) |\bar{A}(Q_i, Q_{i+1})| (Q_{i+1} - Q_i) \quad (27)$$

To construct Roe's matrix, $\bar{A}[Q_i, Q_{i+1}]$, for the three-dimensional Euler equations the information on each side of an interface is used to compute

$$\rho = (\rho_i \rho_{i+1})^{1/2} \quad (28a)$$

$$u = (\rho_i u_i + \rho_{i+1} u_{i+1}) / (\rho_i + \rho_{i+1}) \quad (28b)$$

$$v = (\rho_i v_i + \rho_{i+1} v_{i+1}) / (\rho_i + \rho_{i+1}) \quad (28c)$$

$$w = (\rho_i w_i + \rho_{i+1} w_{i+1}) / (\rho_i + \rho_{i+1}) \quad (28d)$$

$$H = (\rho_i H_i + \rho_{i+1} H_{i+1}) / (\rho_i + \rho_{i+1}) \quad (28e)$$

$$a^2 = (\gamma P / \rho) = (\gamma - 1)[H - (1/2)(u^2 + v^2 + w^2)] \quad (28f)$$

where ρ is the density, P is the pressure, u , v , and w are the velocity components, and e is the total energy

$$e = (P / (\gamma - 1)) + (\rho / 2)(u^2 + v^2 + w^2)$$

and H is the total enthalpy

$$H = (1/\rho)(e + P).$$

Roe and Pike¹² showed that these averages are the only ones that have the required properties.

b. Higher Order Spatial Accuracy

The numerical flux derived thus far leads only to a first order accurate scheme. A family of numerical fluxes of high order accurate TVD schemes using Roe averaging was introduced by Osher and Chakravarthy.¹³ A slight variation of

that numerical flux vector at cell face $i+1/2$ that was found to work well in practice was obtained by using all Roe variables and metrics corresponding to cell face $i+1/2$ for the computation of the eigenvalues and eigenvectors given by

$$\begin{aligned} \bar{f}_{i+1/2} = [f(Q_i)]_{i+1/2} + \sum_{j=1}^m \alpha_{j,i+1/2}^- r_{i+1/2}^{(j)} + \\ \sum_{j=1}^m \{ (1-\gamma)/4 [L_j^+(-1,1) - L_j^-(3,-1)] + \\ (1+\gamma)/4 [L_j^+(1,-1) - L_j^-(1,3)] \} r_{i+1/2}^{(j)} \end{aligned} \quad (29)$$

where

$$\alpha_{j,i+p/2}^{\pm} = \lambda_{i+1/2}^{\pm(j)} \alpha_{j,i+p/2} \quad (30a)$$

and

$$\alpha_{j,i-1/2} = l_{i+1/2}^{(j)} \cdot (Q_i - Q_{i+1}) \quad (30b)$$

$$\alpha_{j,i+1/2} = l_{i+1/2}^{(j)} \cdot (Q_{i+1} - Q_i) \quad (30c)$$

$$\alpha_{j,i+3/2} = l_{i+1/2}^{(j)} \cdot (Q_{i+2} - Q_{i+1}) \quad (30d)$$

$$L_j^{\pm}(1,n) = \text{minmod}(\alpha_{j,i+1/2}^{\pm}, b\alpha_{j,i+n/2}^{\pm}) \quad (30e)$$

$$\text{minmod}(x,y) = \text{sign}(x) \max\{0, \min[|x|, y \text{ sign}(x)]\} \quad (30f)$$

$$b = (3-\gamma)/(1-\gamma). \quad (30g)$$

The parameter b is a compression parameter.¹⁰ The variables $l^{(j)}$, $r^{(j)}$ and $\lambda^{\pm(j)}$ are evaluated using the Roe averaged variables. The flux vectors $f_i = f(Q_i)$ in the first term on the right hand side of Equation (29) is evaluated using the dependent variable vector as indicated (not the Roe averaged variables), but the subscript $i+1/2$ on this bracket indicates the metrics at the $i+1/2$ are to be used.

When two arguments are used in the minmod (MINMOD) operator, the operator returns zero if the arguments are of opposite sign, and the smaller absolute value of the arguments otherwise. The consequences of this throughout the flow are discussed in Reference 14.

Another limiter that is used here is the the highly compressive flux limiter due to Roe¹⁴ called Superbee (SUPBEE).

$$L_j^{\pm}(l, n) = L_j^{\pm}(n, l)$$

The scheme is independent of γ when the Superbee limiter is employed. This would then appear to be like a second order Fromm scheme which occurs for $\gamma = 0$.

c. Implicit Solver

To simplify the solution matrix, linearization of only the first order numerical flux is considered. The argument list of the flux vector and Jacobian matrix is expanded in this section to include the metrics, denoted by M.

A way of formulating the flux difference can be carried out by using Equation (25) at cell face $i+1/2$ and Equation (24) at cell face $i-1/2$. By using a similar development as presented by Whitfield⁶, Equation (26) can be written as

$$\bar{f}_{i+1/2}^{n+1} = f_i^{n+1} + \bar{A}_{i+1/2}^{-} (Q_{i+1}^{n+1} - Q_i^{n+1}) \quad (31a)$$

$$\bar{f}_{i+1/2}^{n+1} = f_i^{n+1} - \bar{A}_{i+1/2}^{+} (Q_i^{n+1} - Q_{i+1}^{n+1}) \quad (31b)$$

The linearized flux difference at cell i using Eqs. (31a) and (31b) is

$$\begin{aligned} \bar{f}_{i+1/2}^{n+1} - \bar{f}_{i+1/2}^{n+1/2} &= \bar{f}_{i+1/2}^n - \bar{f}_{i+1/2}^n + \bar{A}_{i+1/2}^{-} \Delta Q_{i+1}^n - \bar{A}_{i+1/2}^{+} \Delta Q_{i+1}^n \\ &+ [A(Q_i^n, M_{i+1/2}^{n+1}) - \bar{A}_{i+1/2}^{-} - A(Q_i^n, M_{i+1/2}^n) + \bar{A}_{i+1/2}^{+}] \Delta Q_i^n \end{aligned} \quad (32)$$

By neglecting the location of the metrics in the Jacobian matrices, A, in Equation (32), then becomes

$$\bar{f}_{i+1/2}^{n+1} - \bar{f}_{i+1/2}^{n+1/2} = \bar{f}_{i+1/2}^n - \bar{f}_{i+1/2}^n + \delta \bar{A}^{-} \Delta Q^n + \delta \bar{A}^{+} \Delta Q^n \quad (33)$$

Equation (33) results in the same form of the solution matrix as that obtained using the flux-vector split scheme discussed earlier (Section 2.2). The Jacobian matrices in Equation (33);

however, are the Roe matrices, A , whereas the Jacobian matrices in the flux-vector split scheme correspond to the true partials of the positive and negative flux vectors. The solution matrix resulting from Equation (33), with Roe matrices, was used, as well as this same solution matrix but with the Jacobian matrices corresponding to the flux-vector split scheme. In all results obtained thus far, the Jacobian matrices corresponding to the flux-vector split scheme gave improved convergence rates and a more robust scheme than the Roe matrices. The behavior is similar to what happens in the flux-vector splitting, i.e. $A^+ = T \Lambda^+ T^{-1}$, are used in place of the true Jacobian matrices obtained from differentiating the positive and negative flux vectors.

Because the Jacobian matrices corresponding to the flux-vector split scheme work better on the left hand side in the solution matrix than the Roe matrices, this is the scheme presently being employed in the flow solver. This implementation of the Roe scheme is particularly simple, in that the computation of the residual vector, R^n , in the flux-vector split scheme need only to be replaced with the Roe scheme.

4. BOUNDARY CONDITIONS

To derive the characteristic variable boundary conditions, the Euler equations need to be cast in characteristic variable form. Therefore, the characteristic variables are given first from Reference 2, followed by the derivation of the boundary conditions for the specific cases of supersonic inflow and outflow, and impermeable surfaces.

From Reference 2, the characteristic variables correspond in order to the eigenvalues

$$\lambda_k^1 = k_x u + k_y v + k_z w = \theta_k \quad (34a)$$

$$\lambda_k^2 = \theta_k \quad (34b)$$

$$\lambda_k^3 = \theta_k \quad (34c)$$

$$\lambda_k^4 = \theta_k + c|\nabla k| \quad (34d)$$

$$\lambda_k^5 = \theta_k - c|\nabla k| \quad (34e)$$

where $k = \xi, \eta, \text{ or } \zeta$.

a. Characteristic Variable Boundary Conditions

The boundary conditions are derived below assuming locally one-dimensional flow. This assumption is probably better for far field boundary conditions than for boundary conditions applied on or near surfaces. However, numerical experiments using zero pressure gradient, normal pressure gradient, and locally one-dimensional characteristic variable boundary conditions, indicate that similar results can be obtained using any of these three methods for impermeable wall boundary conditions as long as the grid is not extraordinarily coarse. For the computations performed thus far, the locally one-dimensional characteristic variable impermeable wall boundary conditions are to be preferred over the other methods; hence, this method is developed below.

The eigenvalues indicate a direction in computational space, where one particular eigenvalue is associated with one particular characteristic variable. Each eigenvalue indicates the direction across the $k = \text{constant}$ computational surface that information contained in the associated characteristic variable propagates. This result is the basis for determining the boundary conditions referred to here as characteristic variable boundary conditions. Boundary conditions are now developed for the following five

specific cases:

1. farfield-supersonic inflow
2. farfield-supersonic outflow
3. farfield-subsonic inflow
4. farfield-subsonic outflow
5. impermeable surface

Farfield-Supersonic Inflow

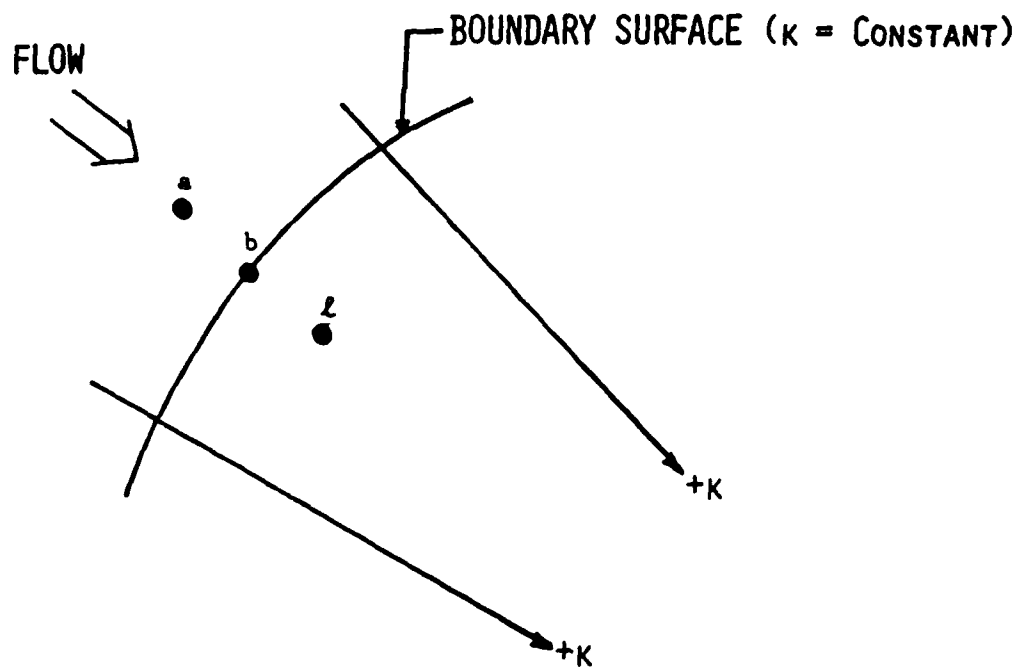
This is a situation where all eigenvalues have the same sign. Because flow is coming into the computational domain, all flow variables are specified.

Farfield-Supersonic Outflow

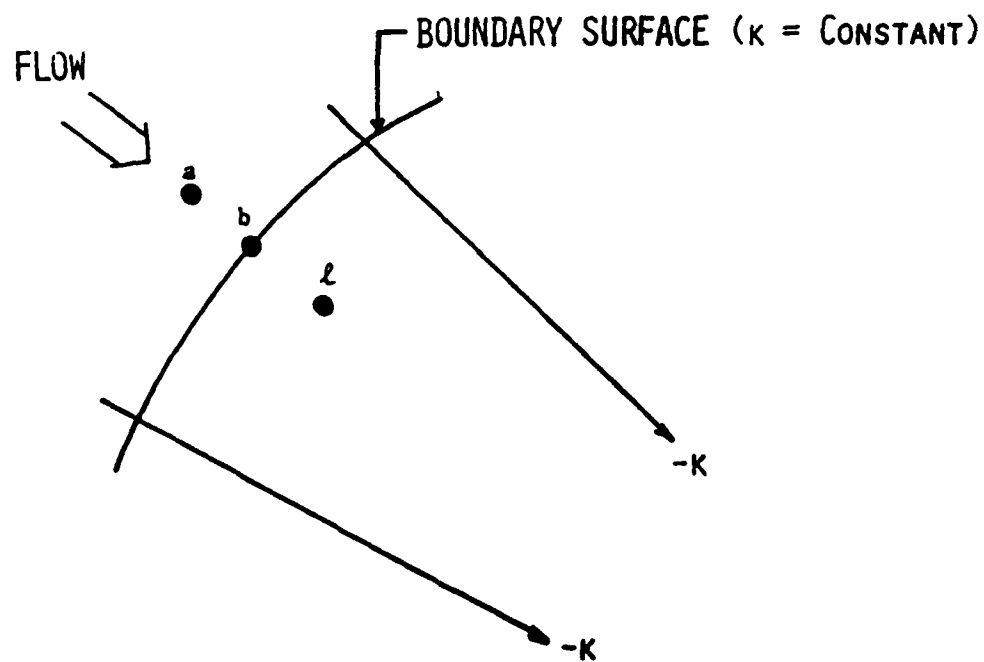
This is another situation where all eigenvalues have the same sign. Because flow is leaving the computational domain all flow variables at the boundary must be obtained from the solution in the computational domain. All flow variables are extrapolated from inside the computational domain to the boundary.

Farfield-Subsonic Inflow

This situation is characterized by four eigenvalues of the same sign and one of differing sign. For the subsonic inflow case shown in Figure 2a with the flow in the direction of increasing computational coordinate k , the first four eigenvalues are positive and the fifth is negative. For the subsonic inflow case shown in Figure 2b with the flow in the direction of decreasing computational coordinate k , the first three and fifth



a. Flow in Direction of $+k$.



b. Flow in Direction of $-k$.

Figure 2. Computational Coordinates Schematic for Characteristic Variable Boundary Conditions

eigenvalues are negative and the fourth eigenvalue is positive. For a totally general three-dimensional code either situation in Figure 2 could occur. Solving for the dependent variables across the boundary yields (Figure 2):

$$P_b = (1/2)\{P_a + P_l + \text{sign}(\lambda_k^1)\rho_o c_o[\bar{k}_x(u_a - u_l) + \bar{k}_y(v_a - v_l) + \bar{k}_z(w_a - w_l)]\} \quad (35a)$$

$$\rho_b = \rho_a + [(P_b - P_a)/c_o^2] \quad (35b)$$

$$u_b = u_a + \bar{k}_x[(P_a - P_b)/\rho_o c_o]\text{sign}(\lambda_k^1) \quad (35c)$$

$$v_b = v_a + \bar{k}_y[(P_a - P_b)/\rho_o c_o]\text{sign}(\lambda_k^1) \quad (35d)$$

$$w_b = w_a + \bar{k}_z[(P_a - P_b)/\rho_o c_o]\text{sign}(\lambda_k^1) \quad (35e)$$

Note that these signs correspond to the sign of the first three eigenvalues, and hence this is a means of writing the code for general applications with arbitrary orientation of the computation coordinates. The point a is outside the computational domain, point b is on the computational boundary, and point l is inside the computational domain.

Farfield-Subsonic Outflow

This situation is also characterized by four eigenvalues of the same sign and one of opposite sign. The development of subsonic outflow boundary conditions is similar to that for subsonic inflow, and Figure (2) can be used again for illustration. However, for subsonic outflow only one characteristic variable is specified and four are determined from information inside the computational domain, whereas, for subsonic inflow four characteristic variables were specified and one was determined from information inside the computational domain. Note, however, that although the equations are identical, point a is now inside the computational domain and point b is outside the computational

domain; whereas, just the opposite was true for subsonic inflow.

The remaining four equations can be solved for the remaining four variables giving

$$P_b = P_a \quad (36a)$$

$$\rho_b = \rho_a + [(P_b - P_a)/c_o^2] \quad (36b)$$

$$u_b = u_a + \bar{k}_x[(P_a - P_b)/\rho_o c_o] \text{sign}(\lambda_k^1) \quad (36c)$$

$$v_b = v_a + \bar{k}_y[(P_a - P_b)/\rho_o c_o] \text{sign}(\lambda_k^1) \quad (36d)$$

$$w_b = w_a + \bar{k}_z[(P_a - P_b)/\rho_o c_o] \text{sign}(\lambda_k^1) \quad (36e)$$

Impermeable Surface

For a boundary across which there is no flow the first three eigenvalues given by Equation (34) are zero, the fourth is positive, and the fifth is negative. One condition must, therefore, be specified. The condition specified is that there is no flow across the boundary.

Finite volume codes only require the pressure at an impermeable boundary; however, to facilitate the handling of points near boundaries and aid in code vectorization, phantom points are used in the present version of the code. The use of phantom points requires information of variables other than pressure to ensure, for example, zero flow across an impermeable boundary.

The solution for the remaining four variables is as follows:

$$P_b = P_r \mp \rho_o c_o (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (37a)$$

$$\rho_b = \rho_r + [(P_b - P_r)/c_o^2] \quad (37b)$$

$$u_b = u_r - \bar{k}_x (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (37c)$$

$$v_b = v_r - \bar{k}_y (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (37d)$$

$$w_b = w_r - \bar{k}_z (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (37e)$$

where the point r is the center of the first cell from the boundary and the minus sign in Equation (37a) is used if r is in the positive k direction from the boundary, and the plus sign is used

if r is in the negative direction from the boundary.

b. Phantom Points

Phantom points are denoted by the subscript p . The points are obtained from the relations

$$P_p = 2P_b - P_{in} \quad (38a)$$

$$\rho_p = 2\rho_b - \rho_{in} \quad (38b)$$

$$u_p = 2u_b - u_{in} \quad (38c)$$

$$v_p = 2v_b - v_{in} \quad (38d)$$

$$w_p = 2w_b - w_{in} \quad (38e)$$

where the subscript "in" refers to the center of the first cell inside the computational domain and can be any of the points a or r used in this section. For example, the phantom points for an impermeable surface are

$$P_p = P_r + 2\rho_0 c_0 (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (39a)$$

$$\rho_p = \rho_r + 2(P_b - P_r)/c_0^2 \quad (39b)$$

$$u_p = u_r - 2\bar{k}_x (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (39c)$$

$$v_p = v_r - 2\bar{k}_y (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (39d)$$

$$w_p = w_r - 2\bar{k}_z (\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \quad (39e)$$

The velocity vector components are the same as those used by Jacocks and Kneile.¹⁵

c. Blocked Grids

Blocked grids are handled in much the same manner in as much as phantom or image points are required at block boundaries to allow for second order differencing across the block-to-block interfaces. In the present code, the dependent variables that correspond to a given block interface are stored so that the adjacent block will have access to those values when they are required. These interfaces are identified in the CUT (Section

4.5) and PASTE (Section 4.24) subroutines and the values are retrieved and stored at the proper times by the GETBC (Section 4.14) and PUTBC (Section 4.26) subroutines. For this steady-state code, the dependent variables used are "unsynchronized" because the code takes whatever time level values are present when they are needed. This is opposed to storing additional time levels required for the block-to-block interfaces which would cause an increase in the amount of in-core storage. From Reference 1, the difference between synchronized and unsynchronized boundary conditions is substantial for unsteady (or time accurate) solutions; however, for the steady case where the error approaches zero there appears to be no degradation in convergence or stability.

d. Zero Pressure Gradient (ZPG) Condition

For several instances a "starting" solution may need to be implemented for the flow solver using a zero pressure gradient boundary condition. This type of BC is used to enable the code to transition from the freestream conditions at step "0" to the characteristic variable BC's (with impermeable wall BC) at step NZPG+1. This allows the code to "build up" to a complete solid body boundary in a specified number of zero pressure gradient iterations (NZPG). This BC forces the code to allow some flow through the body for several (NZPG) iterations as shown in Figure (3). To further enhance the codes ability to smoothly transition through the beginning iterations, the inputs allow the user to gradually apply the zero pressure gradient boundary condition

along impermeable surfaces (NGRAD). These boundary conditions are implemented in the following manner on a constant k impermeable surface:

$$P_p = P_{in} \quad (40a)$$

$$\rho_p = \rho_{in} \quad (40b)$$

$$u_p = u_{in} - F_z u_{in} \bar{k}_x \quad (40c)$$

$$v_p = v_{in} - F_z v_{in} \bar{k}_y \quad (40d)$$

$$w_p = w_{in} - F_z w_{in} \bar{k}_z \quad (40e)$$

where

$$F_z = \begin{cases} 2.0 & \text{for zero pressure gradient BCs} \\ 2.0 * (\text{step \#} / \text{NGRAD}) & \text{[less than or equal to 2.0]} \\ & \text{for gradually applied ZPG BCs} \end{cases}$$

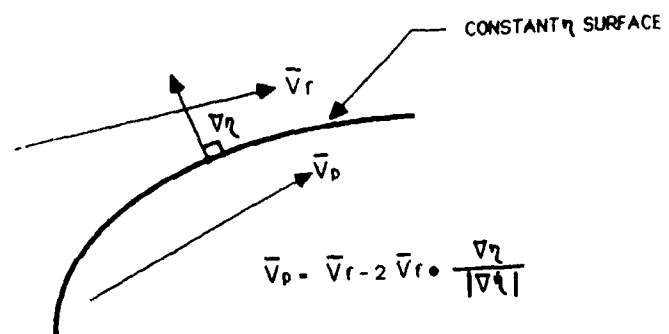


Figure 3. Zero Pressure Gradient Boundary Condition

SECTION III

FLOW SOLVER INPUT

Namelist input allows increased flexibility in the ability to group variables and to read those variables from an input deck. This capability has thus been implemented in this code to allow a better understanding of the interaction among the input variables. The following constitute the namelist headings:

Namelist /FINPUT/ : general flow solver input
Namelist /ROTATE/ : transformation angles
Namelist /SURFACE/ : surface identification
Namelist /BCIN/ : boundary conditions

1. GENERAL INPUT /FINPUT/

As stated previously, namelist FINPUT makes up the general flow solver input. FINPUT is the first of the namelists to be read and is only read once at the beginning of the main routine (MISSE).

a. CFL (real): Courant Number

The CFL variable is input to set the maximum time step allowed during local time stepping. The default value is <15.0> but is usually only set that high for simple cases such as airfoils at subsonic conditions. Typically, CFL should fall between 5.0 and 10.0 depending on angle of attack, bluntness ratio of configuration and Mach number.

Example: E\$ FINPUT CFL=8.0,...

b. `FSMACH` (real): freestream Mach number

`FSMACH` sets the ratio of freestream velocity to local speed of sound. The `FSMACH` is used to set the initial conditions for the velocity components in the x, y and z directions (u, v, and w). The default value for `FSMACH` is `<0.95>` but can be set to Mach numbers in the compressible subsonic, transonic and supersonic range.

Example: `E$ FINPUT CFL= 8.0, FSMACH = 0.80,...`

c. `LIFTAX` (integer): Lift axis

The `LIFTAX` variable is input to tell the code in which direction to place the lift vector. This value is set before any rotation is made to the right-handed coordinate system (namelist `ROTATE`); therefore, the `LIFTAX` variable is set to the positive y direction (+2), positive z direction (+3), the negative y direction (-2) or to the negative z direction (-3). The default value is `<+2>` and a representation of these situations is shown in Figure 4.

Example: `E$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2...`

d. `JFREQ` (integer): frequency of updates to the flux Jacobians

The `JFREQ` variable sets the number of iterations at which the flux Jacobian matrices are updated. If `JFREQ` is set to 10, the flux Jacobian matrices are recalculated every 10 iterations. The default value is `<1>` which requires the matrices to be

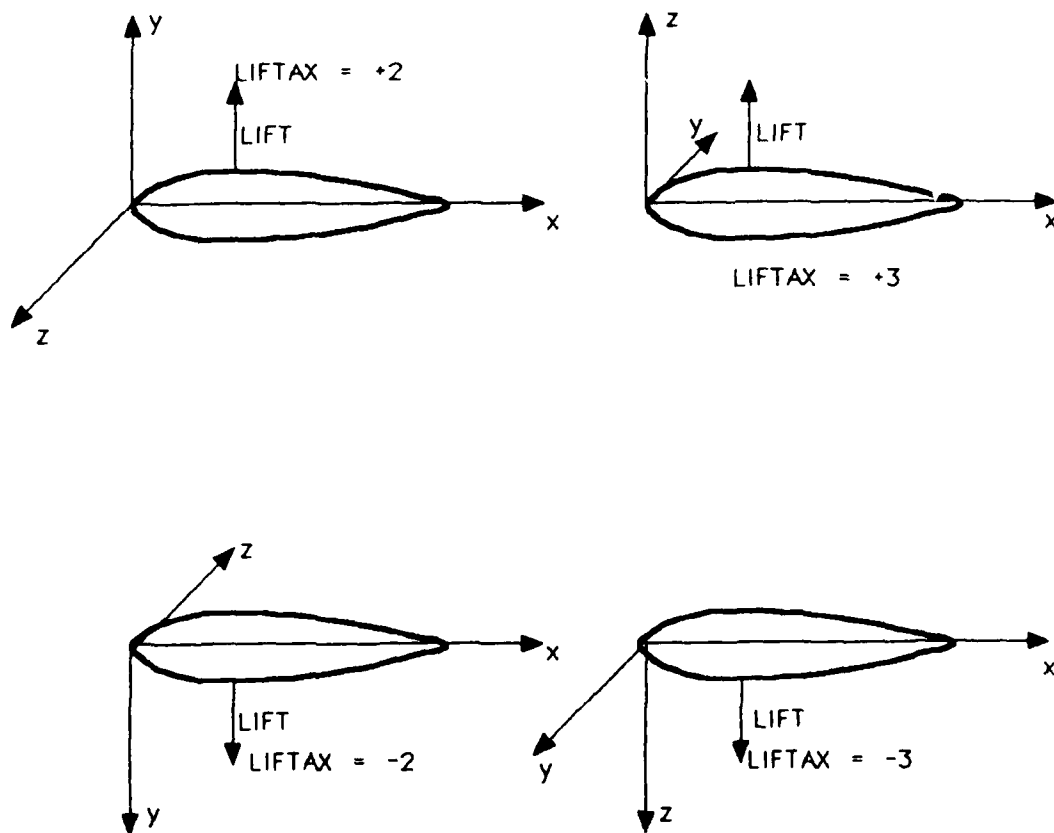


Figure 4. LIFTAX Representation

updated every iteration during the entire job. For steady state solutions, as we have here, results have shown that the two extremes (JFREQ=0 and JFREQ=1) yield the best results. Typically, for those cases with sharp gradients in the flow field the latter (JFREQ=1) situation will enable the solution to converge, although the runtime will be increased dramatically (updating the flux Jacobian matrices requires almost 50% of each iterations CPU time). For simpler conditions, the former input (JFREQ=0) will suffice and greatly reduce CPU time.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2,
JFREQ=5.0,...

e. PRINT (integer): Print-out specification

The PRINT variable sets the type of print-out the user requires. The following choices are made available to the user:

PRINT=0; forces and moments only
PRINT=1; x versus Cp and forces and moments
PRINT=123; x,y,z versus Cp and forces and moments

The default value for PRINT is <0>.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1,...

f. NPR (integer): number of printouts;

NIT (integer): number of iterations

The NPR variable sets the number of prints the user requires. NPR, in conjunction with NIT, also sets the total number of iterations (NPR*NIT). The NIT variable sets the number of iterations at which a printout will occur. Therefore, if the user requires 4 prints and a total of 200 iterations, NPR would be set to 4 and NIT to 50 (NPR=4, NIT=50). In this case, every 50 iterations a printout will be accomplished under the format set by PRINT (Section 3.1) There are no default values for

either NPR or NIT.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100,...\$

g. NCYC (integer): number of cycles for residual printout.

The NCYC variable sets the frequency of residual prints to allow the user to judge the convergence level of his/her run. Residual printouts are formatted as follows:

CYCLE	BLOCK	RTMAX	I	J	K	RTRMS	ETMAX	ETRMS	L2NORM	NSUP
-------	-------	-------	---	---	---	-------	-------	-------	--------	------

where

CYCLE: iteration number
BLOCK: block number
RTMAX: maximum residual for density (R)
I,J,K: I,J,K location at which RTMAX occurs
RTRMS: RMS value of density (log)
ETMAX: Maximum residual for energy (E)
ETRMS: RMS value of energy (log)
L2NORM: L2 norm value for dependent variables (log)
NSUP: Number of supersonic points in entire field

An NCYC value of 5 will yield a residual print out every fifth iteration (cycle). The default value is <1>.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2,
JFREQ=50, PRINT=1, NPR=6, NIT= 100,
NCYC=10,...

h. NSURF (integer): number of surfaces

The NSURF variable tells the code how many surfaces are used in calculating the forces and moments for the configuration. The default value is <1> and more details will be discussed in Section 3.3 (Namelist /SURFACE/).

Example: E\$ FINPUT CFL= 8.0, FSMACH= 0.80, LIFTAX=-2,
JFREQ=50, PRINT=1, NPR=6, NIT=100,
NCYC=10, NSURF=3,...

i. SPLIT (integer): type of splitting technique employed.

The SPLIT variable sets the type of splitting technique used in the code. The user has the following choices:

SPLIT=1; flux-vector-splitting of Steger and Warming

SPLIT=2; flux-difference-splitting of Roe

The default value is Roe's flux-difference-splitting <2> which is less dissipative than flux-vector-splitting. When SPLIT=2 is employed, a second variable, LIMIT (Section 3.1), must be invoked. The Steger and Warming splitting technique (SPLIT= 1) is available due to its tendency to reach a converged solution at a faster rate, when this splitting technique is employed the LIMIT input is ignored.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10, NSURF=3,
SPLIT=2,...

j. LIMIT (integer): type of flux limiter used during Roe averaging

The LIMIT variable is used to specify the type of flux limiter used during a Roe averaged flux-difference-split run (SPLIT=2). The following choices are given to the user:

LIMIT=1; calls subroutine MINMOD (Sec 4.22)

LIMIT=2; calls subroutine SUPBEE (Sec 4.39)

For more information on the limiters employed, refer to the sections describing the specific subroutines. The default value is <1>. If SPLIT=1 is employed then the LIMIT input is simply ignored and no error message is sent by the code.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10, NSURF=3,
SPLIT=2, LIMIT=2, ...

k. BORDER (integer): number of phantom points used

The BORDER variable is used to set the number of phantom points (Section 2.3) used in the block boundary calculations. The following choices are given to the user:

BORDER= 1; first order block boundary (one phantom point)
BORDER= 2; second order block boundary (two phantom points)

The default value is <2> and there has been no reason to go first order in any test cases to the present time.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10,
NSURF=3, SPLIT=2, LIMIT=2, BORDER=2,...

l. GAM (real): ratio of specific heats

The GAM variable allows the user to input a gamma value for any type of fluid medium. The default value, <1.4>, is set for air.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10,
NSURF=3, SPLIT=2, LIMIT=2, BORDER=2,
GAM=1.31,...

m. RESTRT (integer): restart flag

The RESTRT variable tells the code whether or not a restart file is being read in and from what file tape number the restart file is being read from. The following delineates the users' options:

RESTRT = 0; no restart
RESTRT > 1; restart will be read in on unit RESTRT and a new file will be written out on RESTRT + 1.

The default value of RESTRT is <0>.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10,
NSURF=3, SPLIT=2, LIMIT=2,
BORDER=2, GAM=1.131, RESTRT=8,...

n. NZPG (integer): number of zero pressure gradient BC iterations

The NZPG variable is input to tell the code how many zero pressure gradient boundary condition iterations are to be applied in a specific job. The default value is <10>; however, this value is typically only good for airfoils at fairly simple flow conditions (subsonic at low angles of attack). NZPG values of 50 to 100 are generally applied to three dimensional configurations depending on Mach number, bluntness ratio of shape and angle of attack.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10, NSURF=3,
SPLIT=2, LIMIT=2, BORDER=2, GAM=1.131,
RESTART=8, NZPG=75,...

o. NGRAD (integer): number of gradually applied ZPG BC iterations

The NGRAD variable sets the number of iterations that zero pressure gradient boundary conditions will be applied gradually (Section 2.4). NGRAD must always be less than NZPG. The default for NGRAD is <0> although typically, for three-dimensional configurations, NGRAD should be from 10 to 50 or more depending on angle of attack, Mach number and bluntness ratio.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10, NSURF=3,
SPLIT=2, LIMIT=2, BORDER=2, GAM=1.131,
RESTART=8, NGRAD=25, NZPG=75,...

p. RORDER (integer): Order of Roe scheme approximation

The RORDER input specifies the order of approximation used in the Roe averaging. The following choices are given to the user:

RORDER = 1 ; first-order approximation (no limiters)
RORDER = 2 ; second-order approximation
RORDER = 3 ; third-order approximation (MINMOD only)

The default value is <2> for a second-order approximation.

Example: E\$ FINPUT CFL=8.0, FSMACH=0.80, LIFTAX=-2, JFREQ=50,
PRINT=1, NPR=6, NIT=100, NCYC=10, NSURF=3,
SPLIT=2, LIMIT=2, BORDER=2, GAM=1.131,
RESTART=8, NGRAD=25, NZPG=75, RORDER=1, ...

2. TRANSFORMATION INPUT (ROTATE)

The transformation input section allows the user to manipulate a set of transformation matrices to rotate the configuration of interest about any of its three axes. This will effectively give the configuration an angle of attack, α , roll angle, β , or side-slip angle, ϕ , or any combination of these angles depending on how the configuration is oriented. The orientation is input as LIFTAX in the set of general input (Section 3.1) and tells the code in which direction to place the lift vector.

a. TRANS (integers): order in which the rotations will occur

TRANS acts to tell the code in which order to rotate the "Euler" angles. The integer "1" will rotate the configuration about the "x" axis, "2" will rotate the configuration about the "y" axis, and "3" will rotate the configuration about the "z" axis. All angles are counter-clockwise positive (Figure 5) looking towards the origin. The default values are all set to zero, effectively yielding no rotations. If only one rotation is required, say about the "y" axis, then only a "2" will be entered.

Example: E\$ ROTATE TRANS = 2, PHIY = -4.00 \$

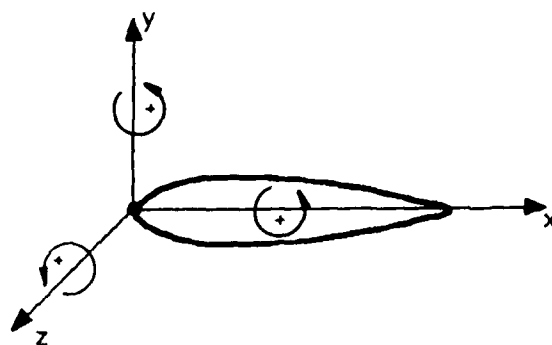


Figure 5. Positive TRANS Rotation Convention

In this case, the configuration will be only rotated clockwise about its "y" axis. The other two angles are assumed to remain zero and so their values for TRANS need not be entered. The same situation holds when only two rotations are requested.

Example: E\$ ROTATE TRANS = 3 , 1, PHIX = -2.43 , PHIZ = 5.84\$

In this case, the configuration will be rotated counter-clockwise first about the "z" axis and then rotated clockwise about the "x" axis. Note that the order of the actual angles (PHIX and PHIZ) is inconsequential.

It is possible to rotate about the same axis more than once in a given run; however, this case is limited by the fact that the magnitude of rotation must remain the same for both rotations.

Example: E\$ ROTATE TRANS = 3 , 1 , 3, PHIX= 2.43 , PHIZ= 5.84

Here the user is first rotating about the "z" axis in a counter-clockwise direction, then about the "x" axis in a counter-clockwise direction and then, again, about the "z" axis. Both rotations about the "z" axis are of the same magnitude and direction. There is no logic in the code to allow the user to rotate about the same axis more than once with different "PHI" values.

- b. PHIX (real): rotation about the "x" axis (degrees)
- PHIY (real): rotation about the "y" axis (degrees)
- PHIZ (real): rotation about the "z" axis (degrees)

The "PHI" angles are simply the amount of rotation about the given axis. Again, the angles are given in the counter-clockwise positive direction and work in conjunction with the TRANS input (Section 3.2) and the LIFTAX input (Section 3.1). These angles must be input in degrees as stated above and are converted into radians within the code.

Example: E\$ ROTATE TRANS= 1, PHIZ= -12.0

In this example, the user is attempting to attain an angle of attack of 12.0 degrees. The LIFTAX value entered in the general input section (FINPUT) was specified as <2>, the positive "y" axis. This means that to obtain a tail down, positive angle of attack of 12 degrees, the configuration is rotated clockwise about the "z" axis effectively yielding the required angle.

3. SURFACE SPECIFICATION (SURFACE)

This set of inputs allows the user to delineate the surfaces he/she wants to calculate forces and moments on or about and sets the reference lengths in the "x", "y" and "z" directions as well as the total reference length and area.

a. Surface Name (sixteen alpha-numeric characters)

A surface name (maximum of sixteen characters) is input prior to every namelist SURFACE. This aids in identifying the surfaces that will be specified in the force and moment print-out section.

b. SREF (integer): surface number

The value for SREF is simply a numerical identifier for the code and corresponds to the SURF values to be input with the namelist BCIN cards in Section 3.4. SREF= 1 must always be the entire configuration required in the force and moment calculations.

c. XREF (real): reference displacement off the "x" axis

YREF (real): reference displacement off the "y" axis

ZREF (real): reference displacement off the "z" axis

The value for XREF, YREF and ZREF allows the user to set a position about which the code will calculate moments. Default values are listed below:

$$XREF = 0.0, \quad YREF = 0.0, \quad ZREF = 0.0,$$

This shows that the default values for XREF, YREF and ZREF lie on the "x", "y" and "z" axes respectively.

d. AREF (real): reference area

LREF (real): reference length

The AREF and LREF values are input so as to allow the user to set a reference area (planform, cross-sectional or any sectional area required) and a reference length, be it chord length, axial body length or any section length required. The default values for these input values are as follows:

$$AREF = 1.0, \quad LREF = 1.0.$$

An example of the Surface specification input is given as follows

Example: Generic body

E\$ SURFACE SREF= 1, AREF= 17.93 , LREF= 142.53, XREF= 3.2

Fin leading edge

E\$ SURFACE SREF= 2, AREF= 56.78 , LREF= 3.42, XREF= .855

.

.

.

Nose cone

E\$ SURFACE SREF= 20, AREF= 23.45

This example shows the surface name of the first section of body to be specified, "Generic body", and it must encompass the entire configuration. This is labeled as SREF= 1 (corresponding to all of the sections specified in the BCIN input with SURF= 1) with a cross-sectional area of 17.93 units, a reference length of 142.53 units and a +3.2 unit displacement on the "x" axis for the moment reference point. The second section (corresponding to all sections of BCTYPE= 1 that includes a 2 in the SURF card) to be specified is the "Fin leading edge" with a fin planform area set at 56.78 units, a chord length of 3.42 units and the moment reference point set at the fin quarter chord. There can be a maximum of 20 surfaces specified, as indicated. All input values are echoed out at the end of a printout along with the forces and moments calculated for each specified surface.

4. BOUNDARY CONDITIONS (BCIN)

The following set of inputs requires the user to specify a boundary condition type for all points on the boundary of a given computational grid. There will be a total of six complete sides in a three-dimensional grid and four sides in a two-dimensional grid.

a. BCTYPE (integer): boundary condition type

The following shows the boundary condition types to specify for each patch of grid:

BCTYPE = 0; terminates input

This BCTYPE is the flag that tells the flow solver that there are no more boundary condition cards to be read. This card should be the last line of the input stream.

BCTYPE = 1; impermeable surface (solid body)

This BCTYPE sets the solid body boundary patches for the code and flags the main routine to call the impermeable surface boundary condition subroutine (SOLID-Section 4.37). The SURF input (Section 3.4) is required for BCTYPE = 1.

BCTYPE = 2; farfield conditions

This BCTYPE defines the patches of grid on which the code implements the inflow/outflow boundary conditions discussed earlier (Section 2.4) and flags the code to call the farfield boundary condition subroutine (FARFLD-Section 4.10)

BCTYPE = 3; symmetry case (reflection plane)

This BCTYPE allows the user to define a plane of symmetry for use in axisymmetric cases (Figure 6).

BCTYPE = 4; block-to-block information

This BCTYPE defines the block-to-block interfaces on which the flow solver must specify image (phantom) points (Figure 7). For this BCTYPE, both BLKA and BLKB inputs are required.

BCTYPE = 5; block-to-block across a singularity

This BCTYPE specifies the boundary condition across a singularity, such as the stagnation line out the nose of an aircraft (Figure 8), to give an image point position for use in differencing across the zero area cells created by the singularity. For this BCTYPE, both BLKA and BLKB inputs are required.

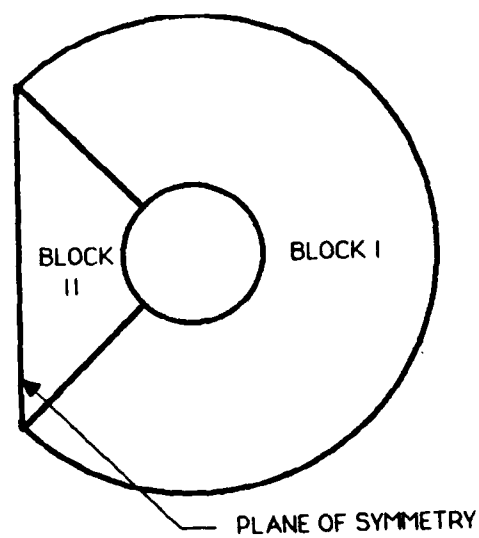
BCTYPE = 6; singularity on a reflection plane

This BCTYPE specifies the boundary condition across a singularity, such as the stagnation line out the nose of an aircraft, when that singularity lies on a reflection plane (Figure 9). This input allows the code to go first order in the first cell along the stagnation line since there is no cell to be specified as the image position for image point information.

BCTYPE = 7; blockout conditions (no calculations)

This BCTYPE allows the user to specify a portion of grid in which to make no flow solver calculations. The code simply skips over this region as if it did not exist; however, the rest of the BCIN cards must reflect this change.

Any boundary condition type can be specified on any patch of grid by stating the type (BCTYPE) of boundary condition and the



FRONT VIEW

Figure 6. Symmetry Case (BCTYPE = 3)

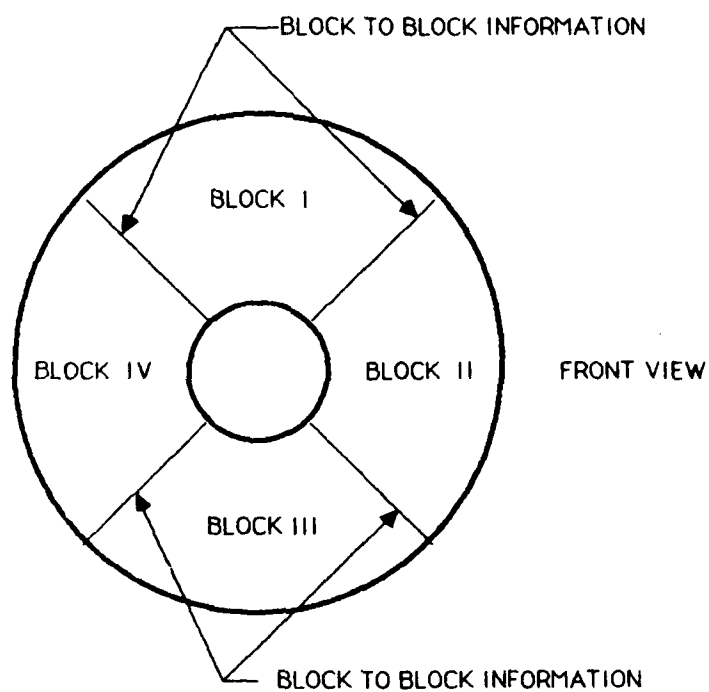


Figure 7. Block-to-Block Information (BCTYPE = 4)

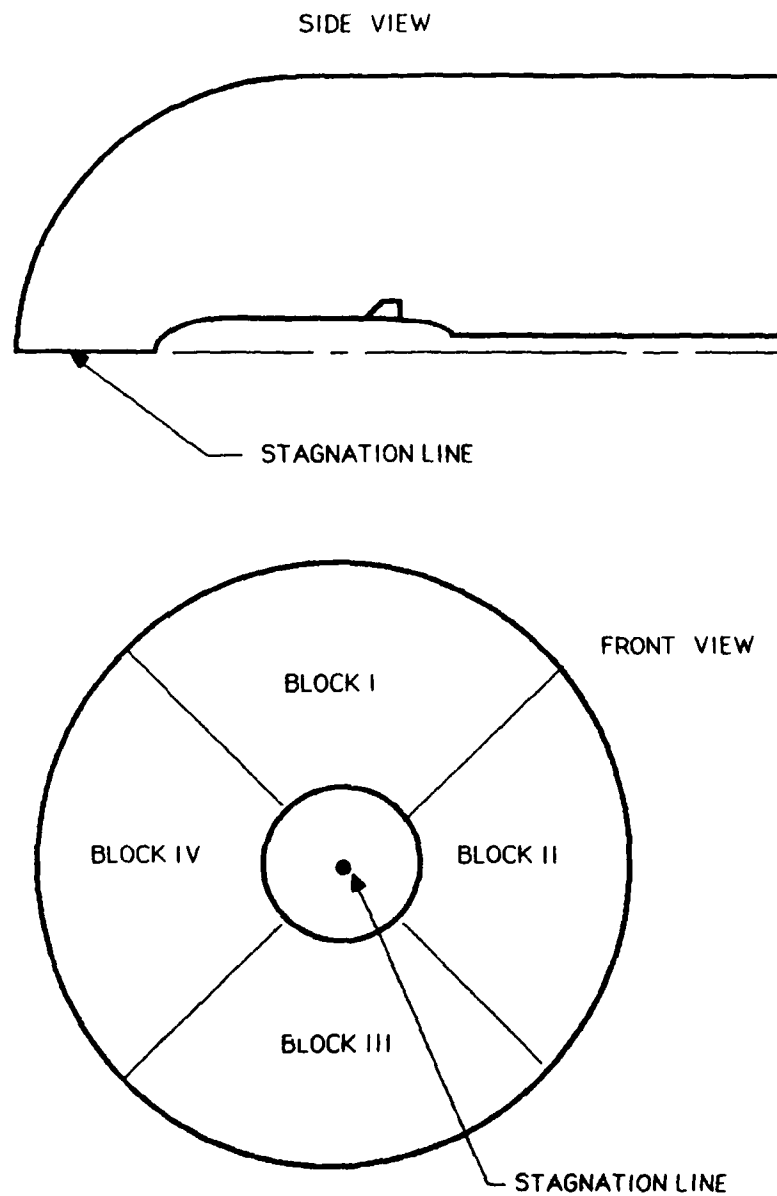


Figure 8. Block-to-Block Across a Singularity (BCTYPE = 5)

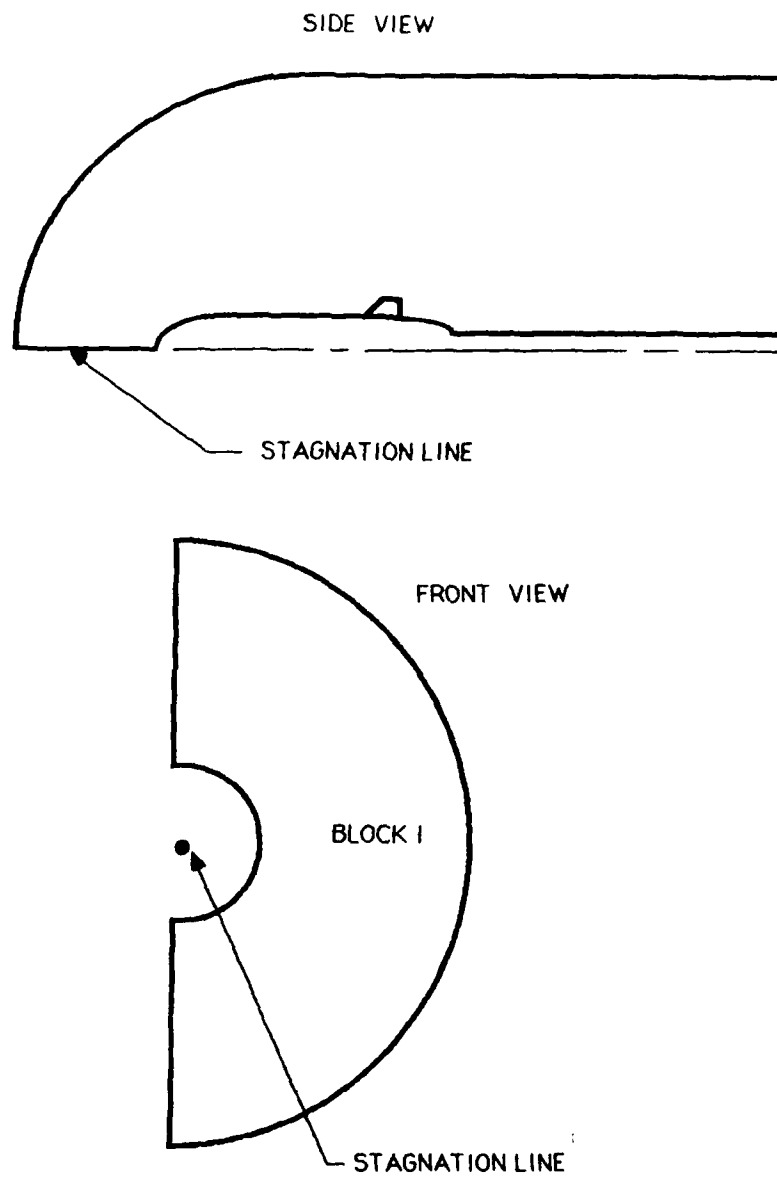


Figure 9. Singularity on a Reflection Plane (BCTYPE = 6)

start (STARTA) and end (ENDA) points for a constant plane in any block (BLKA) of computational grid.

b. SURF (integers): surface numbers with BCTYPE = 1

This SURF input allows the user to indicate in which surfaces this patch of impermeable surface (BCTYPE = 1) is included. As indicated, this input is only valid for a BCTYPE equal to 1 since only solid body boundaries are of interest for force and moment calculations.

Example: E\$ BCIN BCTYPE= 1 , SURF= 1, 2, 20, BLKA= ... ,
STARTA=..., ENDA= ...

In this example, surfaces numbered 1, 2 and 20 from the namelist SURFACE input (SREF=1, SREF=2 and SREF=20) all have this particular patch of solid body grid included in their force and moment calculations.

c. BLKA (integer): block number for a patch of grid

STARTA (integers): start location for a patch of grid

ENDA (integers): end location for a patch of grid

This set of inputs specifies the location of a patch of computational grid to be classified as a particular type of boundary condition with the BCTYPE input. BLKA lets the code know which block the patch is in, STARTA lets the code know the starting index of the patch and ENDA specifies the ending location index for the patch. The patch must specify a constant computational plane, and therefore, one index must remain constant in the STARTA and ENDA input combination.

Example: E\$ BCIN BCTYPE= 2, BLKA= 2, STARTA= 1,1,1,
ENDA= 100,25,1

This example is specifying a constant K plane as a farfield boundary in block 2. Note that both the I and J indices are in increasing order from start(STARTA) to end(ENDA) and that the K index is held constant.

- d. BLKB (integer): block number in block-to-block conditions
- STARTB (integers): start location for block-to-block conditions
- ENDB (integers): end location for block-to-block conditions

The "B" inputs are only required when a block-to-block boundary condition is implemented (BCTYPE equals 4 or 5). This set of inputs tells the code which patch of grid in the first block (STARTA, ENDA, BLKA) corresponds to which patch of grid in the second block (STARTB, ENDB, BLKB). Using this input, the code passes boundary information between blocks A and B.

Example: E\$ BCIN BCTYPE= 4,
BLKA= 3, STARTA= 1 ,1 ,1, ENDA= 100, 25, 1,
BLKB= 8, STARTB=125,75,1, ENDB= 25, 50, 1

In this example, the user has specified that block-to-block conditions exist between blocks 3 (BLKA) and 8 (BLKB). This occurs at a constant K plane (as indicated by the third index in both the "A" and "B" inputs being held constant). Note that the I and J indices in the "A" input are in increasing order from start to end, but that this is not required for the "B" input.

SECTION IV

FLOW SOLVER DESCRIPTION

The Program EAGLE - Flow Solver consists of a main program (MISSE), several subroutines (39) and accesses system routines resident on the Cray X-MP (X-MP version only). The Flow Solver is specifically designed to execute on a computer with a high speed secondary memory such as the solid state disk (SSD) device on a Cray X-MP. However, a Cray 2 version of this code exists that makes this requirement obsolete. An extensive software engineering effort has been made to enhance the codes ability to efficiently store and transfer data for use within the flow solver. Ribbon vector storage with dynamic memory management is employed to prevent wasting computer memory when working with blocks of widely varying dimensions. Only one block is in memory at a time while all other blocks are stored on SSD [X-MP version only]. The memory required to hold all arrays associated with a particular block is determined by the code and the memory size will be increased or decreased as each block is operated on in turn.

The term software engineering is used loosely here to emphasize the fact that considerable concern has been given to the efficient use of the hardware to be utilized. "Four properties that are sufficiently general to be accepted as goals for the entire discipline of software engineering are modifiability, efficiency, reliability and understandability"¹⁶. These essential tenets have been kept in mind

throughout the development of the Program EAGLE - Flow Solver.

1. MAIN ROUTINE (MISSE)

The main program in the flow solver is primarily responsible for the initialization of the dependent variables (Section 2.1), initialization of the ribbon vector storage, the dynamic memory allocation, the reading of namelist inputs (Section 3.1-3.4), the reading and writing of data files (computational grid or restart files), and the proper sequencing of subroutine calls. In addition, the MISSE routine is required to set defaults for namelist input values, check values that are set by namelist for any possible incorrect input and to ensure that enough memory is allocated at any given time during the running of the code.

Before the main iteration loop the following subroutines are called:

(1a) CALL MEMORY => gives MEMORY the username from job control list (JCL). [X-MP version only]

(1b) CALL MEMORY => gives MEMORY users requested amount of memory from JCL and gets back the maximum allowable memory available on the system. [X-MP version only]

(1c) CALL MEMORY => gives MEMORY the current amount of memory being requested by the Flow Solver and gets back the minimum initial amount available. [X-MP version only]

(2a) CALL ROTX => gives ROTX the magnitude of the requested rotation after having read the ROTATE input. MISSE gets back an updated transformation matrix through COMMON /TRANS(3,3)/.

(2b) CALL ROTY => gives ROTY the magnitude of the requested rotation after having read the ROTATE input. MISSE gets back an updated transformation matrix through COMMON /TRANS(3,3)/.

(2c) CALL ROTZ => gives ROTZ the magnitude of the requested rotation after having read the ROTATE input. MISSE gets back an updated transformation matrix through COMMON /TRANS(3,3)/.

(3) CALL SETMEM => gives the current location of the last memory location in the ribbon vector and allocates the proper amount of memory. [X-MP version only]

(4) CALL CHECK => gives CHECK all boundary condition input information after having read a BCIN input card. MISSE gets back a thorough check of all boundary condition input ensuring that all locations within the computational grid have boundary condition types (BCTYPE) associated with them.

(5) CALL CUT => gives CUT boundary condition input information for block-to-block conditions only. MISSE gets back a check on proper boundary alignment and phantom point locations.

(6) CALL SETMEM => gives the current location of the last memory location in the ribbon vector and allocates the proper amount of memory. [X-MP version only]

(7) CALL POINT => calculates the amount of memory required in the ribbon vector.

(8) CALL GRIDIN => reads the computational grid from filetape 10 and stores it in the proper memory location in the ribbon vector.

(9) CALL IC => calculates the initial conditions for the dependent variables and stores them in the proper memory locations in the ribbon vector.

(10) CALL READ => READ will read the grid and dependent variables from a restart file only if one is available and is initiated from namelist input RESTART > 0 [X-MP version only], or the specified restart file for the Cray 2 version.

(11) CALL PUTBC => gives PUTBC the memory locations of the dependent variables and then puts block-to-block boundary condition information in memory for use in the main iteration loop.

(12) CALL SETMPB => from the boundary condition input, SETMPB sets the number of phantom points used for block-to-block calculations and stores them in the proper memory locations in the ribbon vector.

(13) CALL DPMAP => sets the arrays to obtain the diagonal indices of the matrices for vectorization.

(14) CALL METRIC => calculates the areas of each cell face throughout the computational grid.

(15) CALL OPSSD => opens an SSD file and assigns it the requested number. [X-MP version only]

(16) CALL WRITE => writes file information to SSD [X-MP version only] or to the specified restart file [Cray 2 version].

Within the main iteration loop, including an inner loop over all the blocks, the following subroutines are called:

(1) CALL POINT => calculates the amount of memory required in the ribbon vector.

(2) CALL SETMEM => sets the amount of memory required at this point in the iteration loop if there is not enough allocated.

[X-MP version only]

(3) CALL OPSSD => opens a specific SSD file. [X-MP version only]

(4) CALL READ => reads information from the SSD or restart file opened previously.

(5a) CALL OPSSD => opens an SSD file and assigns it a specific number to be used in storing the flux Jacobian matrices from the positive direction. [X-MP version only]

(5b) CALL OPSSD => opens an SSD file and assigns it a specific number to be used in storing the flux Jacobian matrices from the negative direction. [X-MP version only]

(6) CALL SETMEM => sets the amount of memory required at this point in the iteration loop if there is not enough allocated. [X-MP version only]

(7) CALL READ => reads image point information from the SSD or restart files.

(8) CALL PVAR => calculates forces and moments for each individual surface and prepares them to be printed out.

(9) CALL BC => sets the proper boundary conditions for each cell on the boundaries.

(10) CALL GETBC => obtains the boundary condition information for block-to-block cases from memory.

(11) CALL SETMEM => sets the amount of memory required by this portion of the iteration loop if there is not enough allocated. [X-MP version only]

(12) CALL STEP => advances the flow solution one iterative cycle.

(13) CALL SETMEM => sets the amount of memory required by this portion of the iteration loop if there is not enough already allocated. [X-MP version only]

(14) CALL READ => reads image point information from the SSD files.

(15) CALL PUTBC => puts newly calculated block-to-block boundary condition information in memory for use in the appropriate block.

(16) CALL WRITE => writes out all information in the ribbon vector to SSD. [X-MP version only]

At the end of the main iteration loop, the Flow Solver then prints out the forces and moments calculated during the entire run.

2. SUBROUTINE AEQLU

AEQLU is called by DOOM (Section 4.7) and DOOP (Section 4.8) to compute an LU decomposition for the solution matrix on the left hand side of Equation (20). AEQLU receives the number of necessary points in the grid and accomplishes an LU decomposition of the 5 X 5 block matrices. AEQLU returns the decomposed matrices through COMMON /E/.

3. SUBROUTINE BC

BC is called by the main routine, MISSE, to apply the appropriate boundary conditions to the correct patch of computational grid. BC receives information from the Q array (the dependent variables => R, RU, RV, RW, E and pressure => P), the metrics (A, SA), and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB). BC returns the properly implemented boundary conditions for impermeable wall (BCTYPE=1) and farfield conditions (BCTYPE=2) by calling SOLID and FARFLD.

4. SUBROUTINE CHECK

CHECK is called by MISSE from within the read BCIN loop and checks each line of boundary condition input to ensure that one and only one boundary condition type (BCTYPE) is specified for each computational cell. CHECK receives all of the BCIN input information (BLKA, STARTA, ENDA, BLKB, STARTB, ENDB, BCTYPE), the maximum values, over all the blocks, for NI (NIX), NJ (NJX), and NK (NKX), the storage locations for the checking

planes (IPL, JPL, KPL), the total number of blocks in the region (NBB). CHECK returns the internal counter for the number of boundary condition cards (NUMBC), and the error flag (NERR => set equal to one if a fatal boundary condition error is encountered).

5. SUBROUTINE CUT

CUT is also called by the main routine during the reading of the BCIN inputs, but is only implemented when block-to-block conditions are raised. The CUT subroutine is used to specify the number of phantom points to be used in block-to-block calculations. CUT receives the BCIN information (BLKA, STARTA, ENDA, BLKB, STARTB, ENDB, BCTYPE). CUT returns the image point information (IMGE), and the error flag (NERR => set to one when a fatal error is encountered). Subroutine CUT calls PASTE (Section 4.24) which puts the boundary information along the interfacing boundary (BLKB, STARTB, ENDB).

6. SUBROUTINE DELQ

DELQ is called by STEP (Section 4.38) to calculate the change in the dependent variables using the Steger and Warming flux-vector splitting technique discussed earlier in Section 2.2 (SPLIT=1). DELQ receives the dependent variables and pressure (R, RU, RV, RW, E, P), the metrics (AIX, AJX, AKX, AIY, AJY, AKY, AIZ, AJZ, AKZ, SADAI, SADAJ, SADAK), the number of phantom points used on block boundaries (NP11, NP12, NP21, NP22, NP31, NP32), and the indices for the computational block

currently in memory (NI, NJ, NK, NI1, NJ1, NK1, IB). DELQ returns the residual vector for the current iteration (DR, DRU, DRV, DRW, DE). Subroutine DELQ calls FLUX (Section 4.12) to obtain the flux through a given set of cell faces (constant I, J, or K).

7. SUBROUTINE DOOM

DOOM is called by STEP (Section 4.38) to solve for the lower triangular system of equations using Doolittle's method. DOOM receives the change in the dependent variables (DR, DRU, DRV, DRW, DE) from Equation (21a) for the previous iteration, the time step size (DT), the coefficients of the flux Jacobian matrices (B4, B5, B6), the size of the computational block (NI, NJ, NK, NI1, NJ1, NK1, IB), the [MAP1, MAP2] information, and the SSD file to read the flux Jacobian matrices information from (JFILE). DOOM returns the updated changes in the dependent variables (DR, DRU, DRV, DRW, DE) for the left hand side of Equation (21b). Subroutine DOOM calls AEQLU to obtain the LU decomposition of the matrices.

8. SUBROUTINE DOOP

DOOP is called by STEP (Section 4.38) to solve for the upper triangular system of equations using Doolittle's method. DOOP receives the residual vector (DR, DRU, DRV, DRW, DE) for the left hand side of Equation (21a), the time step size (DT), the coefficients of the flux Jacobian matrices (B1, B2, B3), the size of the computational block (NI, NJ, NK, NI1, NJ1, NK1, IB),

the diagonal plane information for full vectorization (MAP1, MAP2), and the SSD file to read the flux Jacobian matrices information from (JFILE). DOOP returns the updated changes in the dependent variables (DR, DRU, DRV, DRW, DE) for the left hand side of Equation (21b). Subroutine DOOP calls AEQLU to obtain the LU decomposition of the matrices.

9. SUBROUTINE DPMAP

DPMAP is called by the main routine to set the diagonal planes in the solution matrix for vectorization. DPMAP coordinates the indices interdependencies so as to allow full vectorization of the loops. DPMAP receives the size of the block (NI, NJ, NK) and returns the diagonal plane information for full vectorization (MAP1, MAP2, and NDV).

10. SUBROUTINE FARFLD

FARFLD is called by BC (Section 4.3) to compute the boundary condition information required at the outer boundaries of the computational region. FARFLD receives the dependent variables and pressure (R, RU, RV, RW, E, and P), the metrics (A, SA), the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB), and the number of boundary condition input cards (M) to cycle through to check for farfield patches of grid (BCTYPE=2). FARFLD returns the dependent variables at the outer or farfield boundary locations. The farfield boundary condition implementation is discussed in further detail in Section 2.2.

11. SUBROUTINE FJMAT

FJMAT is called by STEP (Section 4.38), when required by the specified number of updates to the Jacobians (JFREQ), to calculate the coefficient matrix of the change in the dependent variables (the flux Jacobians). FJMAT receives values for the indices of loops (I1, J1, K1, SIGN, IB, JB, KB), the metrics (AX, AY, AZ, SADA), the dependent variables and pressure (R, RU, RV, RW, E, and P) and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IBLOCK). FJMAT returns updated values for the flux Jacobians (B).

12. SUBROUTINE FLUX

FLUX is called by DELQ (Section 4.6) to calculate the fluxes through all of the constant I, J, or K cell faces. FLUX calculates the eigenvalues associated with the right (EV1R, EV4R, EV5R) and left (EV1L, EV4L, EV5L) hand side of a cell face to determine the "upwind" side for differencing the dependent variables. FLUX receives a single value that determines the length of the loop and is the number of points in either the I (NI), J (NJ), or K (NK) direction and FLUX receives the dependent variables extrapolated to the left and right cell face through COMMON /A/ and /B/. FLUX returns, through COMMON /C/, the change in the dependent variables for the right hand side of Equation (20) (XR, XRU, XRV, XRW, XE).

13. SUBROUTINE FORCE

FORCE is called by PVAR (Section 4.28) to sum the forces and moments for each of the surfaces specified by SURF, SREF, and Surname and to set up the print format for the output of the pressure coefficients. FORCE receives pressure (P), cell face areas (A), axial coordinate locations (X), the size of a given block (NI1, NJ1, NK1, IB), and the number of boundary condition cards (M) to cycle through to obtain the correct patches of grid associated with impermeable wall boundary conditions (BCTYPE=1). FORCE returns pressure coefficients (COEP), and axial cell center locations (XC). Subroutine FORCE calls GETCP to obtain the pressure coefficients along the solid body boundaries.

14. SUBROUTINE GETBC

GETBC is called by BC (Section 4.3) to obtain the location of the block-to-block boundary condition information. GETBC receives the Q array information (the dependent variables => R, RU, RV, RW, E and pressure => P), the QBC array information (the dependent variables on the block boundaries => QBC), the DQ array (the residuals on the block boundaries => DR, DRU, DRV, DRW, DE), the image or phantom point information (IMGE), and the size of the grid (NI1, NJ1, NK1, IB). GETBC returns the necessary location of the block-to-block conditions on which the Q array information must be implemented. Subroutine GETBC calls GETBBC (Section 4.15) which passes the dependent variables (Q) and the residuals (DQ) back for use in GETBC.

15. SUBROUTINE GETBBC

GETBBC is called by GETBC (Section 4.14) to obtain the actual block-to-block boundary information. GETBBC receives the Q and DQ arrays (dependent variables => R, RU, RV, RW, E, pressure => P, residuals => DR, DRU, DRV, DRW, DE), the size of the block (NI1, NJ1, NK1, IB), and the image point information (IMGE, N, M). GETBBC returns the proper dependent variables and residuals for use in GETBC (Section 4.14).

16. SUBROUTINE GETCP

GETCP is called by FORCE (Section 4.13) to calculate the pressures and pressure coefficients along all of the solid body boundaries (BCTYPE=1). GETCP receives the size of the grid (NI1, NJ1, NK1, IB), and the number of boundary condition cards (M) to sort through to find the impermeable wall (BCTYPE=1) boundary condition patches of grid. GETCP returns the pressures (P) and pressure coefficients (COEP).

17. SUBROUTINE GRIDIN

GRIDIN is called by the main routine to read in the computational grid that discretizes the domain about the configuration of interest and to calculate the X, Y, and Z coordinates of the grid cells. GRIDIN receives the block number in which to calculate the coordinates (IB), and returns the coordinates of the grid (X, Y, Z). GRIDIN reads the grid file off of filetape 10 (FT10) [X-MP version

only] and file 1 for the Cray 2 version of the code.

18. SUBROUTINE IC

IC is called by the main routine to obtain the initial conditions in each cell throughout the entire computational domain. IC returns the dependent variables (R, RU, RV, RW, E, and P), and receives the size of the grid block (NI1, NJ1, NK1, IB). The dependent variables throughout the field are initialized to the freestream conditions (RINIT, RUINIT, RVINIT, RWINIT, EINIT, and PINIT) which are received through COMMON /INIT/ and COMMON/CONST/.

19. SUBROUTINE LOCALT

LOCALT is called by STEP (Section 4.38) to obtain the maximum allowable time step size (DT) for each cell in the field. LOCALT receives the dependent variables and pressure (R, RU, RV, RW, E, and P), the metrics (AIX, AJX, AKX, AIY, AJY, AKY, AIZ, AJZ, AKZ, SADAI, SADAJ, SADAK), and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB). LOCALT returns the value of the time step (DT) for each cell in the field (Local time-stepping) which is calculated using the speed at which the propagation of information is moving (C) and the Courant number (CFL) input by the user.

20. SUBROUTINE MATM

MATM is called by ROTX (Section 4.32), ROTY (Section 4.33), and/or ROTZ (Section 4.34) to obtain the current rotation matrix

to transform the initial velocities (RUINIT, RVINIT, RWINIT).
MATM receives a transformation matrix (MAT) from ROTX,
ROTY, or ROTZ and the current rotation matrix (T(3,3)) from
COMMON /TRANS/ and returns, through COMMON /TRANS/, an
updated rotation matrix (T(3,3)) for the calculation of the
initial velocities.

21. SUBROUTINE METRIC

METRIC is called by the main routine to calculate the areas
of the cell faces (metrics) throughout the entire computational
region. METRIC receives the coordinates of the grid (X, Y,
Z), and the size of the grid (NI, NJ, NK). METRIC returns
the areas of the cell faces (AIX, AJX, AKX, AIY, AJY, AKY, AIZ,
AJZ, AKZ), and the square root of the sum of the squares of the
metrics (SADAI, SADAJ, SADAK).

22. SUBROUTINE MINMOD

MINMOD is called by RESID (Section 4.30) when the LIMIT=1
option is selected. MINMOD is a flux limiter and is only
employed when the Roe flux-difference splitting technique
(SPLIT=2) is selected. MINMOD receives the start and end
indices for the limiter loop (NSF, NSL) and the value for (PSI).
MINMOD returns the flux limiter (SP) through COMMON
/SIGMA/.

23. SUBROUTINE OPSSD [X-MP version only]

OPSSD is called by the main routine to open SSD files for data storage. OPSSD receives the unit number to be opened (IU) and the name of the SSD file, SSDissd (ISSD). Subroutine OPSSD calls routine ASSIGN to open and assign the requested SSD files.

24. SUBROUTINE PASTE

PASTE is called by CUT (Section 4.5) to place the proper alignment of block-to-block conditions for use in GETBC (Section 4.14) and PUTBC (Section 4.26). PASTE receives the boundary condition information necessary to properly align the block-to-block interfaces (ITOREA, ICA, BLKA, STARTA, ENDA, ICB, BLKB, STARTB, NSC, DC), and the error flag (NERR). PASTE returns the appropriate interface information through the image arrays (IMAGA, M, IMAGE, N) for use in GETBC and PUTBC.

25. SUBROUTINE POINT

POINT is called by the main routine to calculate the length of the ribbon vector. POINT receives the total size of the blocks (IS3D, IS3D1, IS3DP1, IB) and the current location of the pointer (NEXLOC). POINT returns the updated location of the pointer by calculating the amount of memory required by the information needed in in-core memory (NEXLOC) and the location of the starting point of arrays in the ribbon vector through COMMON /PNTRS/.

26. SUBROUTINE PUTBC

PUTBC is called by the main routine to place the proper dependent variables in the QBC arrays for use in block-to-block calculations. PUTBC receives the dependent variables through the Q array (Q) and the size of the grid (NI1, NJ1, NK1, IB) and returns the proper values in the QBC arrays (QBC). Subroutine PUTBC calls PUTBBC (Section 4.27) to obtain the correct dependent variable values to place in the QBC arrays.

27. SUBROUTINE PUTBBC

PUTBBC is called by PUTBC (Section 4.26) to obtain the correct dependent variables to place in the QBC arrays. PUTBBC receives the dependent variables (Q), the size of the block (NI1, NJ1, NK1, IB), and the number of boundary condition cards to cycle through to coordinate the block-to-block patches of grid (M). PUTBBC returns the QBC array with the correct values for the block-to-block interface calculations (QBC).

28. SUBROUTINE PVAR

PVAR is called by the main routine to calculate the forces and moments for all of the surfaces specified by SURF, SREF, and Surname. PVAR receives the pressure (P), the areas (A), the axial location (X), the cell center locations along the "x" axis (XC), the number of boundary condition cards for obtaining the solid body boundary patches of grid (M), the coefficient of pressures (COEP), and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB). PVAR returns the forces and moments for

each surface through COMMON /SFACE/. Subroutine PVAR calls FORCE (Section 4.13) to obtain the intermediate forces and moments for each surface.

29. SUBROUTINE READ

READ is called by the main routine to read information from the restart files. READ receives the specific variables in-core memory location (A), the amount of memory required by that variable (N), and the SSD unit number (IU). READ returns the variable itself into the proper memory location.

30. SUBROUTINE RESID

RESID is called by STEP (Section 4.38) to employ the Roe averaged flux-difference splitting technique discussed in Section 2.3. RESID receives the dependent variables and pressure (R, RU, RV, RW, E, P), the metrics (AIX, AJX, AKX, AIY, AJY, AKY, AIZ, AJZ, AKZ, SADAI, SADAJ, SADAK), the flux limiter selection (LIMIT), the phantom point information for block-to-block calculations (NP11, NP12, NP21, NP22, NP31, NP32), and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, and IBLOCK). RESID returns the updated residuals (DR, DRU, DRV, DRW, DE) for the right hand side of Equation (21). Subroutine RESID calls RLVECS (Section 4.31) to obtain the eigenvalues and their corresponding right and left eigenvectors to calculate the flux, and calls MINMOD (Section 4.22) or SUPBEE (Section 4.39) when a flux limiter is required by the LIMIT variable input

by the user. For the X-MP version of the code, the eigenvalues and eigenvectors are stored in B1, B2, and B3 sharing the memory location with the flux Jacobian matrices. For the Cray 2 version of the code, the eigenvalues and eigenvectors are simply stored in locations identified by EVL, EVR, and EV.

31. SUBROUTINE RLVECS

RLVECS is called by RESID (Section 4.30) to calculate the eigenvalues and their corresponding right and left eigenvectors associated with the right hand side of Equation (20). RLVECS receives loop indices (I1, J1, K1, IB, JB, KB), the metrics (AX, AY, AZ, SADA), the dependent variables and pressure (R, RU, RV, RW, E and P), and the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IBLOCK). RLVECS returns the calculated eigenvectors and eigenvalues (B1, B2, B3 or EVL, EVR, EV).

32. SUBROUTINE ROTX

ROTX is called by the main routine to calculate a revised rotation matrix. ROTX receives the magnitude of the rotation about the "x" axis in radians (X) and implements that angle. ROTX returns a revised rotation matrix (T(3,3)) through the COMMON /TRANS/. Subroutine ROTX calls MATM (Section 4.20) to perform the matrix multiplication between the original transformation matrix and the requested rotation (matrix) to yield the revised rotation matrix.

33. SUBROUTINE ROTY

ROTY is called by the main routine to calculate a revised rotation matrix. ROTY receives the magnitude of the rotation about the "y" axis in radians (X) and implements that angle. ROTY returns a revised rotation matrix (T(3,3)) through the COMMON /TRANS/. Subroutine ROTY calls MATM (Section 4.20) to perform the matrix multiplication between the original transformation matrix and the requested rotation (matrix) to yield the revised rotation matrix.

34. SUBROUTINE ROTZ

ROTZ is called by the main routine to calculate a revised rotation matrix. ROTZ receives the magnitude of the rotation about the "z" axis in radians (X) and implements that angle. ROTZ returns a revised rotation matrix (T(3,3)) through the COMMON /TRANS/. Subroutine ROTZ calls MATM (Section 4.20) to perform the matrix multiplication between the original transformation matrix and the requested rotation (matrix) to yield the revised rotation matrix.

35. SUBROUTINE SETMEM [X-MP version only]

SETMEM is called by the main routine to set the memory requirements for the flow solver. SETMEM receives the amount of memory required (MNEED) and returns the sufficient amount of memory through COMMON /MEM/. Subroutine SETMEM calls the system routine MEMORY to obtain the required amount of memory.

36. SUBROUTINE SETMPB

SETMPB is called by the main routine in a loop over all of the boundary condition cards to set the amount of phantom points required for block-to-block calculations. SETMPB receives the dimensions of the particular patch of block (M2, M3), the indices for the loop over that patch (IS, JS, IE, JE), and the number of phantom points required (IIAP) set by the user input BORDER. SETMPB returns the MPB array (NP11, NP12, NP21, NP22, NP31, NP32) with the phantom point information.

37. SUBROUTINE SOLID

SOLID is called by BC (Section 4.3) to obtain boundary conditions for impermeable wall (BCTYPE=1) patches of grid. SOLID receives the dependent variables and pressure (R, RU, RV, RW, E, and P), the metrics (A, SA), the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB), and the number of boundary condition cards to cycle through (M). SOLID returns the values of the dependent variables on the solid body boundaries (R, RU, RV, RW, E).

38. SUBROUTINE STEP

STEP is called by the main routine to update the dependent variables. STEP receives the dependent variables and pressure (R, RU, RV, RW, E and P), the metrics (AIX, AJX, AKX, AIY, AJY, AKY, AIZ, AJZ, AKZ, SADAI, SADAJ, SADAK), the flux limiter selection (LIMIT), the phantom point information (NP11, NP12, NP21, NP22, NP31, NP32), the diagonal plane information for

full vectorization (MAP1, MAP2, NDV), the splitting technique to be utilized (SPLIT), the size of the block (NI, NJ, NK, NI1, NJ1, NK1, IB), and the print cycles (NCYC). STEP returns the updated dependent variables (R, RU, RV, RW, E and P). Subroutine STEP calls either RESID (Section 4.30) or DELQ (Section 4.6) to obtain the residual vector, LOCALT (Section 4.19) to obtain a value for the time step size (DT), FJMAT (Section 4.11) to obtain the flux Jacobian matrices (A^+ , A^- , B^+ , B^- , C^+ , C^-), WRITE (Section 4.40) to write out the flux Jacobians to SSD, and DOOP (Section 4.8) and DOOM (Section 4.7) to solve the block 5 X 5 upper or lower system of equations.

39. SUBROUTINE SUPBEE

SUPBEE is called by RESID (Section 4.30) to obtain flux limiters when the LIMIT=2 option is selected by the user. SUPBEE receives the loop indices (NSF, NSL) and returns the flux limiters through COMMON /SIGMA/.

40. SUBROUTINE WRITE

WRITE is called by the main routine to write information out to restart files. WRITE receives the in-core memory location of the specified variable (A), and the amount of memory required by the variable (N). WRITE returns the specified variable into the restart SSD file (IU).

SECTION V

APPLICATIONS

The Program EAGLE - Flow Solver has been extensively exercised on several aerodynamic problems to validate the codes predictive capabilities. These cases have been run through a range of Mach numbers (compressible subsonic, transonic, supersonic) and incidence angles (0 to 20 degrees). The Flow Solver yields excellent inviscid results when compared to experimental data and should provide a very reasonable estimate of the pressure distribution, as well as forces and moments for any arbitrary configuration. Presented in this section are two samples from this validation process that will, hopefully, aid in the understanding of the uses of the Program EAGLE - Flow Solver. A discussion is provided for each grid, describing the single or multi-block structure and the computational region on which the Flow Solver actually works. The Flow Solver input for each example case is presented with a discussion of the individual input statements.

1. NACA 0012 AIRFOIL

The first sample application involves a symmetric airfoil (NACA 0012) at transonic conditions (Mach = 0.80) at a small incidence (1.25 degrees). This case has been a primary check case throughout the development of the EAGLE code and has yielded some interesting results when comparing the different splitting techniques (Steger-Warming => SPLIT=1 and Roe => SPLIT=2).

a. One Block Grid

The Program EAGLE - Numerical Grid Generation System has been employed to generate a single block algebraic, two-dimensional C-type grid about the NACA 0012 airfoil (Figure 10a). The 221 X 20 X 2, C-type grid is used so as to allow a fine concentration of grid points around the leading edge of the section (Figure 10b). Grid lines have been concentrated parallel (J lines) to the airfoil body, and forward near the leading edge perpendicular (I lines) to the airfoil body, to resolve the sharp gradients in the flowfield expected in these regions. A view of the computational region is shown (Figure 11), with dimensions, to help with the input of boundary conditions (BCIN).

b. Flow Solver Input

The Program EAGLE - Flow Solver input is designed to be user oriented in that each namelist input has a specific, readable, meaning for the user. In this case the input stream is a rather simple one in that the two-dimensional grid requires little boundary condition input. The input for this case follows:

```

E$ FINPUT  FSMACH = 0.80, CFL  = 15.0 , GAM   = 1.4 ,
           NGRAD  = 0   , NZPG = 10   , JFREQ = 20   ,
           NPR    = 2   , NIT   = 500  , PRINT = 1   ,
           BORDER = 2   , SPLIT= 2    , LIMIT = 2    ,
           LIFTAX = 2   $

E$ ROTATE  TRANS  = 3   , PHIZ = -1.25                $

NACA 0012
E$ SURFACE SREF   = 1   , XREF = 0.25                $
```

The three general input types are presented here for ease of discussion. In the FINPUT section; the freestream Mach number is

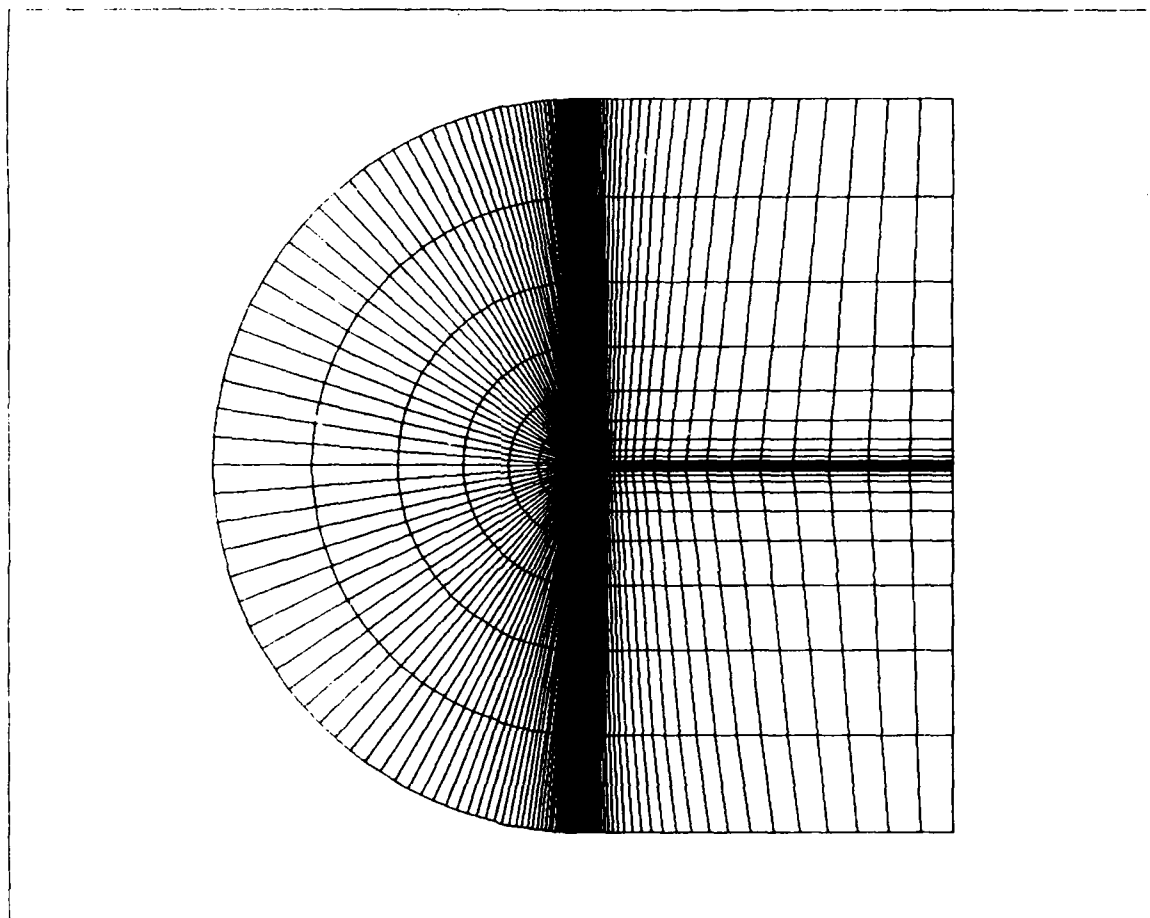


Figure 10a. NACA 0012 Airfoil Grid

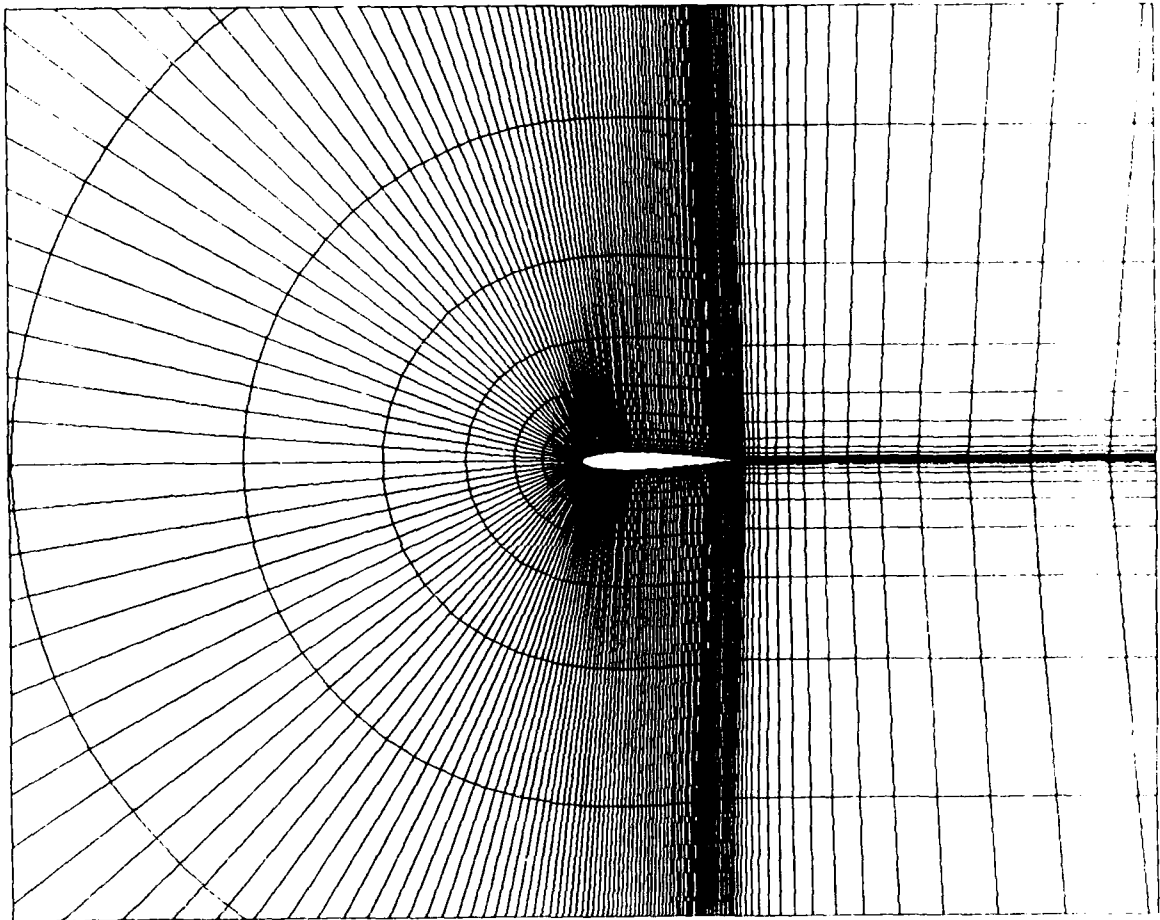


Figure 10b. NACA 0012 Airfoil Grid

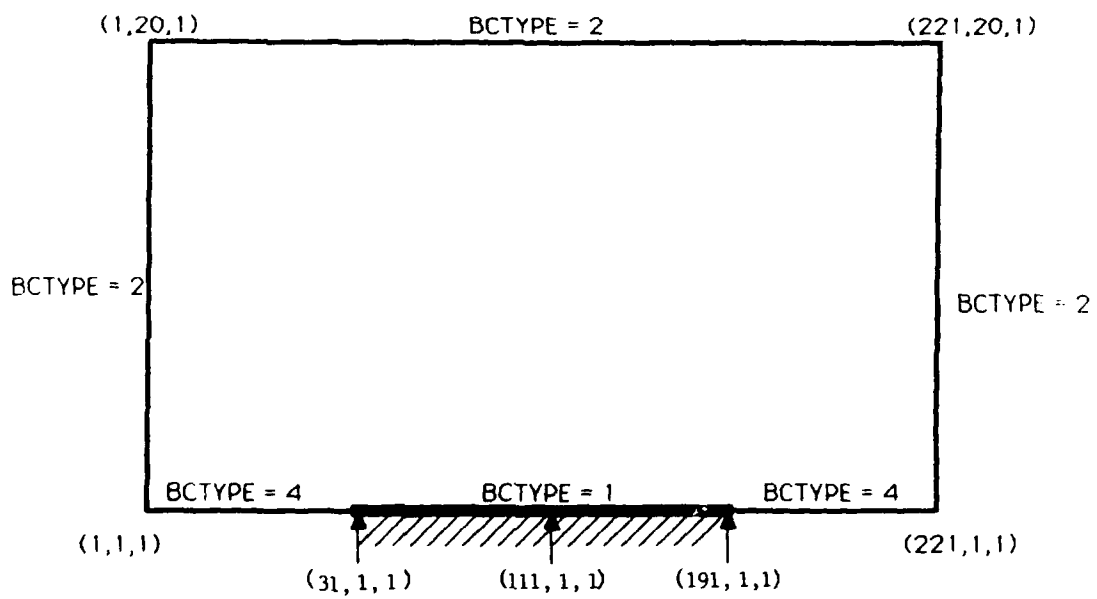


Figure 11. NACA 0012 Computational Region

set to 0.80 (FSMACH), the Courant-Friedrich-Lewy (CFL) condition is set at 15.0, the ratio of specific heats (GAM) 1.4 for air. There will be 10 zero pressure gradient boundary conditions (NZPG) and none will be gradually applied (NGRAD = 0). The flux Jacobian matrices will be updated every 20 iterations (JFREQ) and two phantom points will be used between blocks for block-to-block conditions (BORDER). The total number of iterations will be 1000 (NPR*NIT) and there will be five printouts (NPR) of pressures versus X location, forces and moments (PRINT) every 500 iterations (NIT). The splitting technique employed (SPLIT) will be a form of Roe⁹ averaged flux-difference splitting (Section 2.3) using the SUPBEE (LIMIT) routine (Section 4.39). And finally, the direction of the lift axis is set to the positive "y" axis (LIFTAX) for use with the ROTATE input card.

The ROTATE input is straightforward in that it describes a rotation about the positive "z" axis. The "z" axis is set as the axis of rotation (TRANS) and the magnitude of this angle is described as -1.25 degree (PHIZ). Figure 12 shows the details of this rotation. Since all rotations are counter-clockwise positive, and tail down is a positive angle of attack, to obtain a 1.25 degree angle of attack the rotations must be as indicated.

The first SURFACE input is a string of a maximum of sixteen alphanumeric characters (NACA 0012). This aids in the description of the surfaces to be specified. Following this card is the SURFACE input itself in which most of the default values are exercised (LREF = 1.0 for chord length). The surface specification throughout this run will be 1 (SREF) for the entire

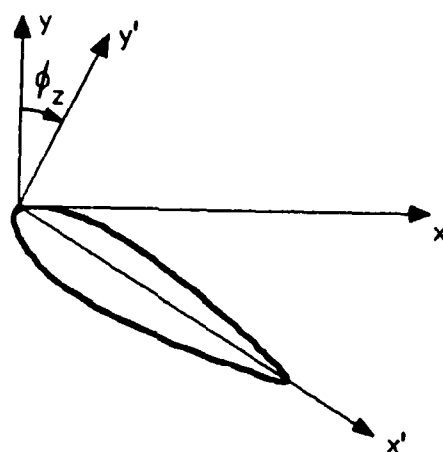


Figure 12. NACA 0012 Airfoil Incidence Angle Rotation

airfoil and will coincide with values of 1 for the SURF input on the BCIN cards (BCTYPE=1). The reference "x" location is set to the quarter chord (XREF) for force and moment calculations.

The following cards detail the BCIN input to the Flow Solver:

```

E$ BCIN BCTYPE = 1, SURF = 1, BLKA = 1, STARTA = 31,1, 1,
                                ENDA= 191, 1, 2 $
E$ BCIN BCTYPE = 2,           BLKA = 1, STARTA = 1, 1, 1,
                                ENDA = 1, 20, 2 $
E$ BCIN BCTYPE = 2,           BLKA = 1, STARTA = 221,1,1,
                                ENDA = 221,20, 2 $
E$ BCIN BCTYPE = 2,           BLKA = 1, STARTA = 1, 20,1,
                                ENDA = 221,20, 2 $
E$ BCIN BCTYPE = 4,           BLKA = 1, STARTA = 1, 1, 1,
                                ENDA = 31, 1, 2 ,
                                BLKB = 1, STARTB = 221,1,1,
                                ENDB = 191,1, 2 $

E$ BCIN BCTYPE = 0 $

```

For this example, impermeable surface (solid body) boundary conditions (BCTYPE) are applied in block 1 (BLKA) from 31,1,1 to 191,1,1 (STARTA and ENDA) a constant J line. This portion of solid body will be used in the force and moment calculations for surface 1 (SURF => SREF and NACA 0012). Farfield boundary conditions (BCTYPE) are applied in block 1 (BLKA) for three different planes (lines, in this two-dimensional example) from 1,1,1 to 1,20,2, a constant I line (the upper rear boundary); from 221,1,1 to 221,20,2, another constant I line (the lower rear boundary) and from 1,20,1 to 221,20,2, a constant J line (the outer boundary). Block-to-block conditions are applied in the wake region of this airfoil in block 1 (BLKA and BLKB) from 1,1,1 to 31,1,2 (STARTA and ENDA) corresponding to a patch (line in 2-D) of grid from 221,1,1 to 191,1,2 (STARTB and ENDB); again, a constant J line. Finally, the BCIN input deck is terminated by employing the ending input of zero (BCTYPE).

c. Flow Solver Output

Output for the Program EAGLE - Flow Solver is organized to echo all user inputs for checking purposes, print convergence information (residuals and number of supersonic points), print pressure coefficient data (if requested) and to print forces, moments and reference lengths and areas. Table 1 shows the output for this test case. Echo checking, from the namelist input streams and from specific information regarding solution procedures, is displayed as shown. The dimensions of the numerical grid generated are echoed for checking purposes and a header is displayed giving the type of splitting performed. The convergence information by iteration number (STEP) and block number (BLOCK) is shown. Residuals (RMS and MAX) are given for density and energy.

The Flow Solver output for this sample case is given in Table 1 and a plot of the pressure coefficient (C_p) versus chord length (X/C) is shown in Figure 16. The convergence for this example is quite good in that the L2 Norm (not shown) has converged to -2.88 or three orders of magnitude in 1000 iterations, the residuals for density are at E-4 and the number of supersonic points is constant at 451 for twelve iterations (although not shown, NSUP was constant at 450 for almost 200 iterations). All indications show that a good engineering level solution has been attained in 1000 iterations. Any further convergence would go essentially unnoticed in any graphical analysis of the solution. The maximum memory size was under 1.7 million words (maximum field length) and the run time was 115.835 CP seconds on the Cray X-MP.

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output

```
*****
*
*           Program EAGLE- Flow Solver
* Multi-block Implicit Steady-State Euler Algorithm*
*
*           Cray 2 version (20 July 1988)
*
*****
```

Air Force Armament Laboratory (AFATL)
Aeromechanics Division (AFATL/FX)
Aerodynamics Branch (AFATL/FXA)
Computational Fluid Dynamics Section (CFD)
Eglin AFB, FL 32542-5242

Questions or Comments call:
Jon S. Mounts or Dave M. Belk @
(904) 882-3124/2767 or AV 872-3124/2767

***** ECHO CHECK *****

Namelist /FINPUT/ :

Courant number (CFL) = 15.0
Freestream Mach number (FSMACH) = 0.8
Frequency of flux Jacobian matrix updates
(JFREQ) = 20
Number of zero pressure gradient BC iterations
(NZPG) = 10
Number of gradually applied ZPG BC iterations
(NGRAD) = 0
Lift will be calculated for the #2 axis
(LIFTAX)
Total number of surfaces (NSURF) = 1
Total number of iterations (NPR)*(NIT) = 1000

Computational region :

Block	NI	NJ	NK
1	221	20	1

Block 1 assumed to be two-dimensional

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

Namelist /ROTATE/ :

Rotation angle sequence (TRANS) = 3 , 0 , 0

Rotation angles :

PHIX = 0.

PHIY = 0.

PHIZ = -1.25

Differencing Technique :

SPLIT = 2

LIMIT = 2

RORDER= 2

Namelist /SURFACE/ :

Surface reference number (SREF) : 1

Reference area (AREF) = 1.0

Reference length (LREF) = 1.0

"x" reference (XREF) = 0.25

"y" reference (YREF) = 0.

"z" reference (ZREF) = 0.

Total memory allocated : 1664816

2 PRINTOUTS EVERY 500 CYCLES									
STEP	IB	RTMAX	IRM	JRM	KRM	RT RMS	ETMAX	ETRMS	
0	1	6.420e-01	111	3	2	0.	1.284e+00	0.	
1	1	3.819e-01	111	2	2	-2.323e-01	8.227e-01	-2.309e-01	
2	1	1.752e-01	113	4	2	-4.376e-01	3.581e-01	-4.619e-01	
3	1	-1.784e-01	95	2	2	-4.249e-01	-3.565e-01	-4.773e-01	
4	1	-2.303e-01	98	2	2	-3.450e-01	-5.230e-01	-3.825e-01	
5	1	-2.706e-01	111	2	2	-3.133e-01	-6.766e-01	-3.346e-01	
6	1	-2.649e-01	111	2	2	-3.354e-01	-6.775e-01	-3.425e-01	
7	1	-2.094e-01	112	4	2	-4.051e-01	-4.487e-01	-3.983e-01	
8	1	-1.617e-01	112	5	2	-4.733e-01	-3.672e-01	-4.688e-01	
9	1	1.405e-01	97	2	2	-4.960e-01	-3.263e-01	-5.113e-01	
10	1	3.739e-01	111	2	2	-2.892e-01	8.077e-01	-3.253e-01	
11	1	-1.327e-01	95	2	2	-5.449e-01	-2.983e-01	-5.505e-01	
12	1	-1.158e-01	96	2	2	-6.118e-01	-2.549e-01	-6.408e-01	
13	1	9.642e-02	89	5	2	-6.515e-01	-2.016e-01	-6.999e-01	
14	1	1.006e-01	88	6	2	-6.781e-01	1.772e-01	-7.330e-01	
15	1	9.985e-02	90	6	2	-6.993e-01	1.854e-01	-7.516e-01	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

485	1	-1.028e-03	60	2	2	-2.933e+00	-1.939e-03	-3.015e+00
486	1	-1.008e-03	60	2	2	-2.943e+00	-1.928e-03	-3.025e+00
487	1	1.016e-03	197	2	2	-2.939e+00	-1.905e-03	-3.021e+00
488	1	-1.001e-03	60	6	2	-2.946e+00	-1.935e-03	-3.026e+00
489	1	-1.022e-03	60	6	2	-2.949e+00	-1.971e-03	-3.029e+00
490	1	-9.777e-04	60	2	2	-2.953e+00	-1.909e-03	-3.034e+00
491	1	-9.636e-04	60	2	2	-2.949e+00	-1.867e-03	-3.033e+00
492	1	1.167e-03	196	2	2	-2.942e+00	1.916e-03	-3.029e+00
493	1	-9.166e-04	60	2	2	-2.959e+00	-1.753e-03	-3.045e+00
494	1	-9.001e-04	60	6	2	-2.968e+00	-1.700e-03	-3.053e+00
495	1	-8.905e-04	60	6	2	-2.970e+00	-1.666e-03	-3.054e+00
496	1	1.060e-03	195	2	2	-2.964e+00	1.791e-03	-3.049e+00
497	1	9.354e-04	195	2	2	-2.972e+00	-1.628e-03	-3.054e+00
498	1	-8.659e-04	60	2	2	-2.980e+00	-1.628e-03	-3.061e+00
499	1	-8.445e-04	60	2	2	-2.977e+00	-1.574e-03	-3.060e+00

ISTEP = 500 BLOCK = 1

	X	CP
32	0.9948	0.3623
33	0.9842	0.3165
34	0.9729	0.2636
35	0.9611	0.2231
36	0.9487	0.1907
37	0.9357	0.1620
38	0.9220	0.1366
39	0.9078	0.1130
40	0.8929	0.0917
41	0.8773	0.0711
42	0.8612	0.0515
43	0.8445	0.0320
44	0.8271	0.0133
45	0.8092	-0.0054
46	0.7907	-0.0241
47	0.7717	-0.0421
48	0.7521	-0.0591
49	0.7321	-0.0791
50	0.7117	-0.1008
51	0.6909	-0.1145
52	0.6698	-0.1244
53	0.6483	-0.1503
54	0.6266	-0.1821
55	0.6048	-0.1852
56	0.5828	-0.1820
57	0.5608	-0.2137
58	0.5387	-0.2838
59	0.5168	-0.3219
60	0.4949	-0.3050
61	0.4732	-0.2754
62	0.4517	-0.2841
63	0.4305	-0.3523
64	0.4097	-0.3981
65	0.3892	-0.4125
66	0.3692	-0.4066
67	0.3496	-0.3351
68	0.3305	-0.6208
69	0.3120	-0.6733

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

70	0.2940	-0.6638
71	0.2766	-0.6599
72	0.2598	-0.6523
73	0.2437	-0.6421
74	0.2281	-0.6291
75	0.2132	-0.6141
76	0.1989	-0.5980
77	0.1852	-0.5804
78	0.1722	-0.5617
79	0.1598	-0.5416
80	0.1480	-0.5197
81	0.1368	-0.4958
82	0.1262	-0.4703
83	0.1161	-0.4456
84	0.1066	-0.4191
85	0.0977	-0.3919
86	0.0893	-0.3608
87	0.0813	-0.3324
88	0.0739	-0.3004
89	0.0669	-0.2650
90	0.0603	-0.2318
91	0.0542	-0.1942
92	0.0485	-0.1523
93	0.0432	-0.1144
94	0.0382	-0.0648
95	0.0336	-0.0192
96	0.0294	0.0348
97	0.0254	0.0911
98	0.0218	0.1534
99	0.0185	0.2202
100	0.0155	0.2929
101	0.0127	0.3712
102	0.0103	0.4598
103	0.0081	0.5546
104	0.0062	0.6547
105	0.0045	0.7572
106	0.0031	0.8591
107	0.0020	0.9495
108	0.0012	1.0346
109	0.0006	1.0891
110	0.0002	1.1097
111	0.0000	1.1033
112	0.0000	1.0684
113	0.0002	1.0021
114	0.0006	0.9023
115	0.0012	0.7866
116	0.0020	0.6513
117	0.0031	0.5147
118	0.0045	0.3785
119	0.0062	0.2564
120	0.0081	0.1422
121	0.0103	0.0380
122	0.0127	-0.0546
123	0.0155	-0.1366
124	0.0185	-0.2079
125	0.0218	-0.2726
126	0.0254	-0.3395

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

127	0.0294	-0.3805
128	0.0336	-0.4327
129	0.0382	-0.4860
130	0.0432	-0.5126
131	0.0485	-0.5503
132	0.0542	-0.5936
133	0.0603	-0.6298
134	0.0669	-0.6575
135	0.0739	-0.6821
136	0.0813	-0.7087
137	0.0893	-0.7379
138	0.0977	-0.7655
139	0.1066	-0.7890
140	0.1161	-0.8090
141	0.1262	-0.8291
142	0.1368	-0.8522
143	0.1480	-0.8756
144	0.1598	-0.8971
145	0.1722	-0.9157
146	0.1852	-0.9318
147	0.1989	-0.9471
148	0.2132	-0.9641
149	0.2281	-0.9818
150	0.2437	-0.9989
151	0.2598	-1.0146
152	0.2766	-1.0287
153	0.2940	-1.0413
154	0.3120	-1.0527
155	0.3305	-1.0635
156	0.3496	-1.0743
157	0.3692	-1.0855
158	0.3892	-1.0963
159	0.4097	-1.1063
160	0.4305	-1.1152
161	0.4517	-1.1228
162	0.4732	-1.1291
163	0.4949	-1.1345
164	0.5168	-1.1391
165	0.5387	-1.1433
166	0.5608	-1.1472
167	0.5828	-1.1489
168	0.6048	-1.1284
169	0.6266	-0.1077
170	0.6483	0.0132
171	0.6698	0.0127
172	0.6909	0.0122
173	0.7117	0.0163
174	0.7321	0.0208
175	0.7521	0.0265
176	0.7717	0.0345
177	0.7907	0.0447
178	0.8092	0.0563
179	0.8271	0.0694
180	0.8445	0.0833
181	0.8612	0.0984
182	0.8773	0.1145
183	0.8929	0.1316

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

184	0.9078	0.1509
185	0.9220	0.1721
186	0.9357	0.1960
187	0.9487	0.2232
188	0.9611	0.2552
189	0.9729	0.2939
190	0.9842	0.3422
191	0.9948	0.3704

500 1 9.415e-04 194 2 2 -2.979e+00 1.637e-03 -3.062e+00

MACH NO.	PHIX	PHIY	PHIZ	CFL	LIFTAX
0.8000	0.00	0.00	-1.25	15.0	2

		REFERENCE			
SURFACE	NAME	AREA	X	Y	Z
1	NACA 0012 Test	1.00000	0.25000	0.00000	0.00000

		FORCES		MOMENTS	
1	NACA 0012 Test	FX =	0.009043	MX =	-0.163014
		FY =	0.326028	MY =	0.004522
		FZ =	0.000000	MZ =	0.029244

		COEFFICIENTS			
SURFACE	NAME	FORCES		MOMENTS	
1	NACA 0012 Test	CX =	0.009043	CMX =	-0.163014
		CY =	0.326028	CMY =	0.004522
		CZ =	0.000000	CMZ =	0.029244

		FORCES		COEFFICIENTS	
1	NACA 0012 Test	FL =	0.325753	CL =	0.325753
		FD =	0.016153	CD =	0.016153
		FS =	0.000000	CS =	0.000000

501	1	1.018e-03	194	2	2	-2.977e+00	1.744e-03	-3.060e+00
502	1	7.637e-04	57	8	2	-2.991e+00	1.540e-03	-3.074e+00
503	1	8.444e-04	57	8	2	-2.995e+00	1.617e-03	-3.078e+00
504	1	1.045e-03	197	2	2	-2.992e+00	1.778e-03	-3.076e+00
505	1	-1.009e-03	195	2	2	-3.003e+00	-1.813e-03	-3.087e+00
506	1	8.323e-04	58	5	2	-3.008e+00	1.584e-03	-3.092e+00
507	1	8.435e-04	58	5	2	-3.006e+00	1.608e-03	-3.094e+00
508	1	8.424e-04	58	5	2	-3.004e+00	1.622e-03	-3.093e+00
509	1	1.173e-03	196	2	2	-3.000e+00	1.968e-03	-3.090e+00
510	1	8.417e-04	58	6	2	-3.014e+00	1.585e-03	-3.103e+00
511	1	8.434e-04	58	6	2	-3.018e+00	1.572e-03	-3.107e+00
512	1	8.403e-04	58	6	2	-3.017e+00	1.567e-03	-3.107e+00
513	1	8.933e-04	195	2	2	-3.019e+00	1.566e-03	-3.110e+00
514	1	1.007e-03	195	2	2	-3.023e+00	1.725e-03	-3.115e+00
515	1	8.197e-04	58	7	2	-3.031e+00	1.610e-03	-3.120e+00
516	1	8.579e-04	58	7	2	-3.027e+00	1.660e-03	-3.112e+00
517	1	8.385e-04	58	7	2	-3.022e+00	1.648e-03	-3.106e+00
518	1	1.076e-03	194	2	2	-3.011e+00	1.942e-03	-3.093e+00
519	1	1.072e-03	194	2	2	-3.023e+00	1.905e-03	-3.105e+00
520	1	7.381e-04	58	6	2	-3.037e+00	1.481e-03	-3.118e+00
521	1	8.580e-04	197	2	2	-3.028e+00	1.452e-03	-3.111e+00
522	1	-1.048e-03	195	2	2	-3.031e+00	-1.879e-03	-3.114e+00

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

523	1	-1.076e-03	195	2	2	-3.034e+00	-1.917e-03	-3.115e+00
524	1	7.364e-04	58	7	2	-3.041e+00	1.431e-03	-3.120e+00
525	1	7.258e-04	58	7	2	-3.044e+00	1.480e-03	-3.126e+00
.								
.								
.								
975	1	-7.631e-04	197	2	2	-3.314e+00	-1.345e-03	-3.420e+00
976	1	8.539e-04	195	2	2	-3.339e+00	1.434e-03	-3.446e+00
977	1	9.282e-04	195	2	2	-3.343e+00	1.592e-03	-3.442e+00
978	1	3.958e-04	27	3	2	-3.352e+00	7.469e-04	-3.445e+00
979	1	-7.310e-04	196	2	2	-3.320e+00	-1.168e-03	-3.419e+00
980	1	7.057e-04	194	2	2	-3.343e+00	1.119e-03	-3.436e+00
981	1	9.820e-04	194	2	2	-3.343e+00	1.648e-03	-3.435e+00
982	1	8.784e-04	194	2	2	-3.345e+00	1.510e-03	-3.436e+00
983	1	7.618e-04	197	2	2	-3.331e+00	1.271e-03	-3.436e+00
984	1	-9.734e-04	195	2	2	-3.319e+00	-1.676e-03	-3.419e+00
985	1	-8.540e-04	195	2	2	-3.348e+00	-1.476e-03	-3.440e+00
986	1	-4.825e-04	198	2	2	-3.390e+00	-9.262e-04	-3.496e+00
987	1	-8.122e-04	194	2	2	-3.374e+00	-1.316e-03	-3.491e+00
988	1	-1.218e-03	194	2	2	-3.297e+00	-2.012e-03	-3.413e+00
989	1	9.981e-04	196	2	2	-3.313e+00	1.701e-03	-3.418e+00
990	1	-6.028e-04	193	2	2	-3.369e+00	-9.989e-04	-3.471e+00
991	1	-6.008e-04	193	2	2	-3.378e+00	-7.946e-04	-3.480e+00
992	1	-7.360e-04	197	2	2	-3.341e+00	-1.297e-03	-3.443e+00
993	1	9.029e-04	195	2	2	-3.356e+00	1.524e-03	-3.457e+00
994	1	8.924e-04	195	2	2	-3.372e+00	1.481e-03	-3.471e+00
995	1	-4.598e-04	193	2	2	-3.388e+00	-6.407e-04	-3.483e+00
996	1	-7.557e-04	196	2	2	-3.341e+00	-1.365e-03	-3.434e+00
997	1	-7.188e-04	196	2	2	-3.378e+00	-1.204e-03	-3.466e+00
998	1	7.838e-04	194	2	2	-3.385e+00	1.285e-03	-3.468e+00
999	1	9.871e-04	194	2	2	-3.356e+00	1.706e-03	-3.439e+00

ISTEP = 1000 BLOCK = 1

		2
	X	CP
32	0.9948	0.3675
33	0.9842	0.3198
34	0.9729	0.2663
35	0.9611	0.2258
36	0.9487	0.1930
37	0.9357	0.1643
38	0.9220	0.1392
39	0.9078	0.1153
40	0.8929	0.0934
41	0.8773	0.0724
42	0.8612	0.0528
43	0.8445	0.0337
44	0.8271	0.0153
45	0.8092	-0.0028
46	0.7907	-0.0206
47	0.7717	-0.0386
48	0.7521	-0.0565
49	0.7321	-0.0747
50	0.7117	-0.0930
51	0.6909	-0.1115

TABLE I. Program EAGLE - Flow Solv
NACA 0012 Airfoil Output (continued)

52	0.6698	-0.1300
53	0.6483	-0.1486
54	0.6266	-0.1676
55	0.6048	-0.1871
56	0.5828	-0.2068
57	0.5608	-0.2264
58	0.5387	-0.2470
59	0.5168	-0.2683
60	0.4949	-0.2901
61	0.4732	-0.3121
62	0.4517	-0.3350
63	0.4305	-0.3589
64	0.4097	-0.3766
65	0.3892	-0.4030
66	0.3692	-0.4143
67	0.3496	-0.3353
68	0.3305	-0.5631
69	0.3120	-0.6568
70	0.2940	-0.6574
71	0.2766	-0.6523
72	0.2598	-0.6453
73	0.2437	-0.6357
74	0.2281	-0.6237
75	0.2132	-0.6094
76	0.1989	-0.5934
77	0.1852	-0.5758
78	0.1722	-0.5571
79	0.1598	-0.5372
80	0.1480	-0.5156
81	0.1368	-0.4920
82	0.1262	-0.4665
83	0.1161	-0.4425
84	0.1066	-0.4151
85	0.0977	-0.3877
86	0.0893	-0.3572
87	0.0813	-0.3289
88	0.0739	-0.2970
89	0.0669	-0.2617
90	0.0603	-0.2285
91	0.0542	-0.1907
92	0.0485	-0.1490
93	0.0432	-0.1112
94	0.0382	-0.0614
95	0.0336	-0.0159
96	0.0294	0.0383
97	0.0254	0.0942
98	0.0218	0.1567
99	0.0185	0.2226
100	0.0155	0.2964
101	0.0127	0.3741
102	0.0103	0.4632
103	0.0081	0.5573
104	0.0062	0.6579
105	0.0045	0.7591
106	0.0031	0.8617
107	0.0020	0.9511
108	0.0012	1.0374

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (continued)

109	0.0006	1.0908
110	0.0002	1.1088
111	0.0000	1.1022
112	0.0000	1.0670
113	0.0002	0.9998
114	0.000	0.9000
115	0.0012	0.7841
116	0.0020	0.6487
117	0.0031	0.5123
118	0.0045	0.3758
119	0.0062	0.2541
120	0.0081	0.1394
121	0.0103	0.0357
122	0.0127	-0.0576
123	0.0155	-0.1392
124	0.0185	-0.2104
125	0.0218	-0.2747
126	0.0254	-0.3412
127	0.0294	-0.3833
128	0.0336	-0.4348
129	0.0382	-0.4884
130	0.0432	-0.5153
131	0.0485	-0.5520
132	0.0542	-0.5955
133	0.0603	-0.6318
134	0.0669	-0.6596
135	0.0739	-0.6842
136	0.0813	-0.7104
137	0.0893	-0.7398
138	0.0977	-0.7675
139	0.1066	-0.7908
140	0.1161	-0.8108
141	0.1262	-0.8309
142	0.1368	-0.8540
143	0.1480	-0.8772
144	0.1598	-0.8987
145	0.1722	-0.9174
146	0.1852	-0.9335
147	0.1989	-0.9489
148	0.2132	-0.9658
149	0.2281	-0.9834
150	0.2437	-1.0006
151	0.2598	-1.0163
152	0.2766	-1.0304
153	0.2940	-1.0429
154	0.3120	-1.0543
155	0.3305	-1.0650
156	0.3496	-1.0759
157	0.3692	-1.0870
158	0.3892	-1.0979
159	0.4097	-1.1079
160	0.4305	-1.1169
161	0.4517	-1.1246
162	0.4732	-1.1310
163	0.4949	-1.1364
164	0.5168	-1.1410
165	0.5387	-1.1452

TABLE I. Program EAGLE - Flow Solver
NACA 0012 Airfoil Output (concluded)

166	0.5608	-1.1486
167	0.5828	-1.1557
168	0.6048	-1.1557
169	0.6266	-0.3054
170	0.6483	0.0293
171	0.6698	0.0177
172	0.6909	0.0179
173	0.7117	0.0202
174	0.7321	0.0244
175	0.7521	0.0307
176	0.7717	0.0389
177	0.7907	0.0486
178	0.8092	0.0598
179	0.8271	0.0720
180	0.8445	0.0855
181	0.8612	0.1003
182	0.8773	0.1161
183	0.8929	0.1334
184	0.9078	0.1519
185	0.9220	0.1731
186	0.9357	0.1955
187	0.9487	0.2215
188	0.9611	0.2524
189	0.9729	0.2924
190	0.9842	0.3428
191	0.9948	0.3753

MACH NO.	PHIX	PHIY	PHIZ	CFL	LIFTAX
0.8000	0.00	0.00	-1.25	15.0	2

		REFERENCE			
SURFACE	NAME	AREA	X	Y	Z
1	NACA 0012 Test	1.00000	0.25000	0.00000	0.00000

		FORCES	MOMENTS		
1	NACA 0012 Test	FX =	0.009289	MX =	-0.167376
		FY =	0.334751	MY =	0.004645
		FZ =	0.000000	MZ =	0.031203

		FORCES	COEFFICIENTS		MOMENTS
1	NACA 0012 Test	CX =	0.009289	CMX =	-0.167376
		CY =	0.334751	CMY =	0.004645
		CZ =	0.000000	CMZ =	0.031203

		FORCES	COEFFICIENTS		
1	NACA 0012 Test	FL =	0.334469	CL =	0.334469
		FD =	0.016589	CD =	0.016589
		FS =	0.000000	CS =	0.000000

TOTAL CP TIME WAS 241.545929

2. GENERIC FINNED CONFIGURATION

This configuration is a generic weapon shape with fins consisting of a tangent ogive forebody and afterbody with a cylindrical center section and a set of four NACA 0008 airfoil fins. This airframe has been specifically designed to be used for CFD code validation. The Air Force Armament Laboratory has expended a substantial amount of resources to compile an extensive surface pressure, force and moment data base for this configuration (Figure 13), as well as the same configuration in two and three store mutual interference combinations, for the validation of computational aerodynamic techniques (Section 6.4).

a. Four Block Grid

The computational region for this body is divided into four circumferential blocks (Figure 14a) from -45 to 45 degrees (Block I), 45 to 135 degrees (Block II), 135 to 225 degrees (Block III), and from 225 to 315 or -45 degrees (Block IV). The entire grid is made up of these four 140 X 24 X 10 blocks divided equally between the four fins so as to allow the solid body (impermeable wall) boundary conditions to be implemented on a block boundary. As before, the grid lines are concentrated down along the solid body (the J=1 line) and along the leading edge of the fins (the I=68 lines). In both of these areas, sharp gradients are expected to occur in the flowfield (Figure 14b).

b. Flow Solver Input

The input for the generic finned configuration will specify

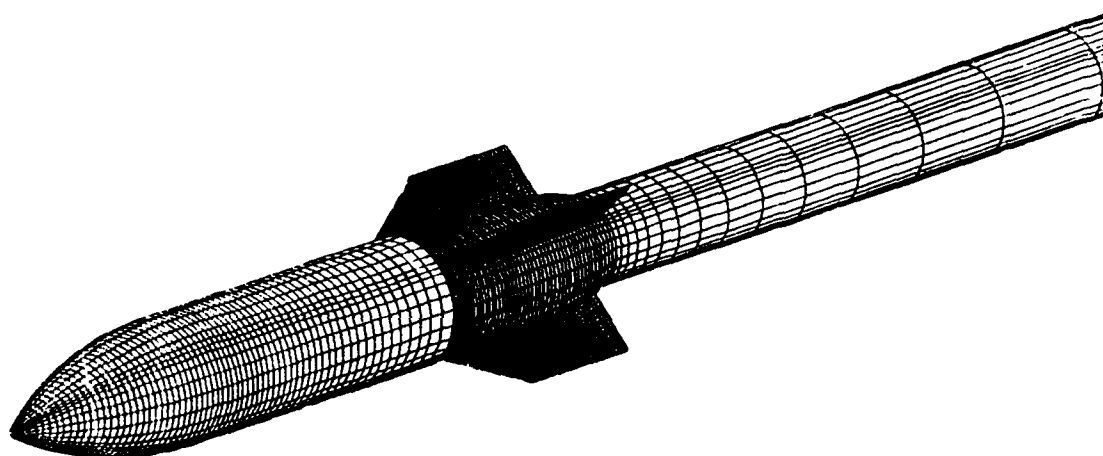


Figure 13. Generic Finned Configuration

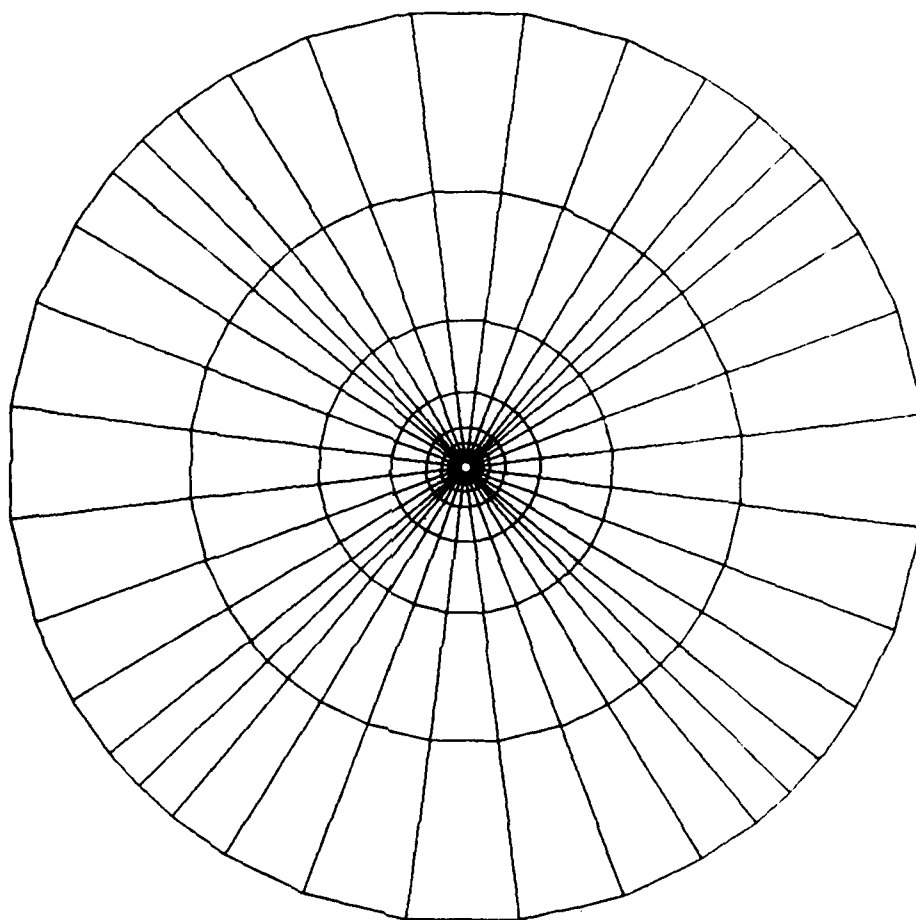


Figure 14a. Generic Finned Configuration 4 Block Grid
(Front View)

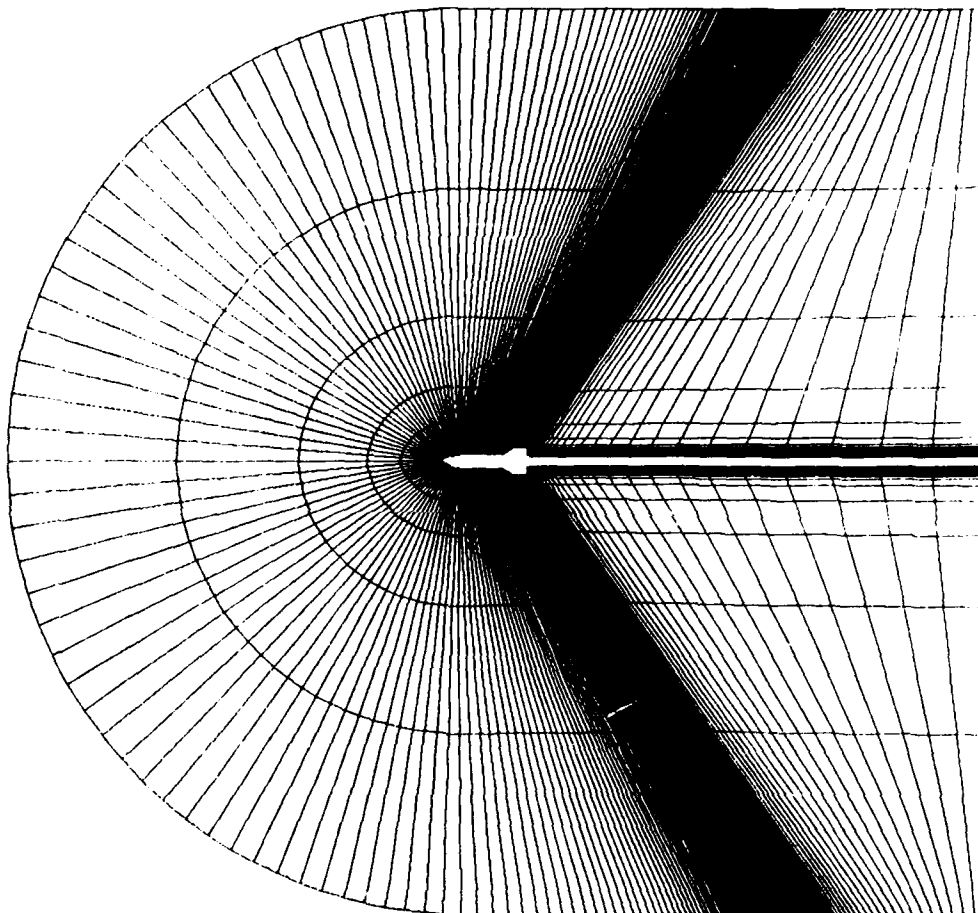


Figure 14b. Generic Finned Configuration 4 Block Grid
(Side View)

the first supersonic test case (Mach = 1.05) at a fifteen degree incidence angle. Here we have implemented a full 360-degree grid so as to allow future investigations into yaw and roll angle effects.

```
ES FINPUT CFL=5.0, FSMACH=1.05, JFREQ=1, SPLIT=2, LIMIT=1,  
          NSURF=4 , RORDER=2, NPR=1, NIT=1000, PRINT=123,  
          NGRAD=10, NZPG=50 $
```

```
ES ROTATE TRANS=3, PHIZ= -15.0$
```

Body w/sting

```
ES SURFACE SREF=1, XREF= 6.4 $
```

Fin Set

```
ES SURFACE SREF=2, XREF= 0.25 $
```

Body Alone

```
ES SURFACE SREF=3, XREF= 6.4 $
```

Generic Store

```
ES SURFACE SREF=4, XREF= 6.4 $
```

As before, the CFL condition is set first (CFL=5.0) along with the Mach number (FSMACH=1.05) and the number of updates to the flux Jacobian matrices (JFREQ=1). For this specific case, the Jacobians will be calculated every iteration. A form of Roe averaging will be employed (SPLIT=2) using the second-order (RORDER=2) MINMOD limiter (LIMIT=1). A total of 1000 iterations will be run (NPR*NIT) with print outs occurring once every 1000 iterations (NPR=1). The print format will be x, y, and z coordinates versus Cp (pressure coefficients), as well as all forces and moments for the specified 4 surfaces (NSURF=4). Zero pressure gradient boundary conditions will be applied for the first 10 iterations (NGRAD=10) with a total of 50 zero pressure gradient boundary conditions (NZPG=50) for the entire run. The rotations for this case will be fifteen degrees, and therefore will be input so as to rotate the body about the z axis. The surface inputs specify four different sections of the body to

calculate forces and moments on or about. As is always the case, the first portion of the body is the entire configuration (SREF=1) entitled "Body w/sting". The second section encompasses the entire fin set (SREF=2), entitled "Fin Set" and is specified in the BCIN input cards with SURF=2. The third portion is comprised of the ogive-cylinder-ogive body (SREF=3), called "Body Alone", and is specified in the BCIN input cards with SURF=3. And finally, the fourth specified portion is the "Generic Store" which is comprised of just the body and fins (not including the sting). The boundary condition inputs are as follows for this four circumferential block case:

```

E$ BCIN BCTYPE=1, SURF=1,3,4,BLKA=1, STARTA= 1, 1, 1,
      ENDA= 116, 1, 10 $
E$ BCIN BCTYPE=1, SURF=1,3,4,BLKA=2, STARTA= 1, 1, 1,
      ENDA= 116, 1, 10 $
E$ BCIN BCTYPE=1, SURF=1,3,4,BLKA=3, STARTA= 1, 1, 1,
      ENDA= 116, 1, 10 $
E$ BCIN BCTYPE=1, SURF=1,3,4,BLKA=4, STARTA= 1, 1, 1,
      ENDA= 116, 1, 10 $
E$ BCIN BCTYPE=1, SURF=1, BLKA=1, STARTA=117,1, 1,
      ENDA= 140,1 , 10$
E$ BCIN BCTYPE=1, SURF=1, BLKA=2, STARTA=117,1, 1,
      ENDA= 140,1 , 10$
E$ BCIN BCTYPE=1, SURF=1, BLKA=3, STARTA=117,1, 1,
      ENDA= 140,1 , 10$
E$ BCIN BCTYPE=1, SURF=1, BLKA=4, STARTA=117,1, 1,
      ENDA= 140,1 , 10$
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=1, STARTA=68, 1, 1,
      ENDA= 113,16, 1 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=1, STARTA=68, 1,10,
      ENDA= 113,16, 10 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=2, STARTA=68, 1, 1,
      ENDA= 113,16, 1 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=2, STARTA=68, 1,10,
      ENDA= 113,16, 10 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=3, STARTA=68, 1, 1,
      ENDA= 113,16, 1 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=3, STARTA=68, 1,10,
      ENDA= 113,16, 10 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=4, STARTA=68, 1, 1,
      ENDA= 113,16, 1 $
E$ BCIN BCTYPE=1, SURF=1,2,4,BLKA=4, STARTA=68, 1,10,
      ENDA= 113,16, 10 $
E$ BCIN BCTYPE=2, BLKA=1, STARTA= 1,24, 1,
      ENDA= 140,24, 10 $
E$ BCIN BCTYPE=2, BLKA=1, STARTA=140,1, 1,
      ENDA= 140,24, 10 $

```

E\$ BCIN BCTYPE=2,	BLKA=2, STARTA= 1,24, 1,
E\$ BCIN BCTYPE=2,	ENDA= 140,24, 10 \$
E\$ BCIN BCTYPE=2,	BLKA=2, STARTA=140,1, 1,
E\$ BCIN BCTYPE=2,	ENDA= 140,24, 10 \$
E\$ BCIN BCTYPE=2,	BLKA=3, STARTA= 1,24, 1,
E\$ BCIN BCTYPE=2,	ENDA= 140,24, 10 \$
E\$ BCIN BCTYPE=2,	BLKA=3, STARTA=140,1, 1,
E\$ BCIN BCTYPE=2,	ENDA= 140,24, 10 \$
E\$ BCIN BCTYPE=2,	BLKA=4, STARTA= 1,24, 1,
E\$ BCIN BCTYPE=2,	ENDA= 140,24, 10 \$
E\$ BCIN BCTYPE=2,	BLKA=4, STARTA=140,1, 1,
E\$ BCIN BCTYPE=4,	ENDA= 140,24, 10 \$
	BLKA=1, STARTA= 1, 1, 1,
	ENDA= 68,24, 1 ,
	BLKB=4, STARTB= 1, 1,10,
E\$ BCIN BCTYPE=4,	ENDB= 68,24, 10 \$
	BLKA=2, STARTA= 1, 1, 1,
	ENDA= 68,24, 1 ,
	BLKB=1, STARTB= 1, 1,10,
E\$ BCIN BCTYPE=4,	ENDB= 68,24, 10 \$
	BLKA=3, STARTA= 1, 1, 1,
	ENDA= 68,24, 1 ,
	BLKB=2, STARTB= 1, 1,10,
E\$ BCIN BCTYPE=4,	ENDB= 68,24, 10 \$
	BLKA=4, STARTA= 1, 1, 1,
	ENDA= 68,24, 1 ,
	BLKB=3, STARTB= 1, 1,10,
E\$ BCIN BCTYPE=4,	ENDB= 68,24, 10 \$
	BLKA=1, STARTA=68,16, 1,
	ENDA= 113,24, 1 ,
	BLKB=4, STARTB=68,16,10,
E\$ BCIN BCTYPE=4,	ENDB= 113,24, 10 \$
	BLKA=2, STARTA=68,16, 1,
	ENDA= 113,24, 1 ,
	BLKB=1, STARTB=68,16,10,
E\$ BCIN BCTYPE=4,	ENDB= 113,24, 10 \$
	BLKA=3, STARTA=68,16, 1,
	ENDA= 113,24, 1 ,
	BLKB=2, STARTB=68,16,10,
E\$ BCIN BCTYPE=4,	ENDB= 113,24, 10 \$
	BLKA=4, STARTA=68,16, 1,
	ENDA= 113,24, 1 ,
	BLKB=3, STARTB=68,16,10,
E\$ BCIN BCTYPE=4,	ENDB= 113,24, 10 \$
	BLKA=1, STARTA=113,1, 1,
	ENDA= 140,24, 1 ,
	BLKB=4, STARTB=113,1,10,
E\$ BCIN BCTYPE=4,	ENDB= 140,24, 10 \$
	BLKA=2, STARTA=113,1, 1,
	ENDA= 140,24, 1 ,
	BLKB=1, STARTB=113,1,10,
E\$ BCIN BCTYPE=4,	ENDB= 140,24, 10 \$
	BLKA=3, STARTA=113,1, 1,
	ENDA= 140,24, 1 ,
	BLKB=2, STARTB=113,1,10,
E\$ BCIN BCTYPE=4,	ENDB= 140,24, 10 \$
	BLKA=4, STARTA=113,1, 1,
	ENDA= 140,24, 1 ,
	BLKB=3, STARTB=113,1,10,
	ENDB= 140,24, 10 \$

```

E$ BCIN BCTYPE=5,          BLKA=1, STARTA= 1, 1, 1,
                             ENDA= 1,24, 10 ,
                             BLKB=3, STARTB= 1, 1, 1,
                             ENDB= 1,24, 10 $
E$ BCIN BCTYPE=5,          BLKA=2, STARTA= 1, 1, 1,
                             ENDA= 1,24, 10 ,
                             BLKB=4, STARTB= 1, 1, 1,
                             ENDB= 1,24, 10 $
E$ BCIN BCTYPE=0 $

```

The boundary condition input for this four block case is set up for ease of understanding since the bc's are essentially symmetric. The first four inputs are for the solid body (BCTYPE=1) condition along the J=1 surface in blocks 1-4 for the body alone. These inputs correspond to the SREF=1, 3 and 4 cards in the SURFACE input. The second four cards specify the sting portion of the configuration and correspond to the SREF=1 input in the SURFACE section. The next eight input cards are for the upper and lower surfaces of the fins in all four blocks (K=1 and 10). These are also impermeable wall conditions (BCTYPE=1) and correspond to the SURFACE input SREF=1, 2 and 4. Farfield conditions (BCTYPE=2) are then implemented for all four blocks at the outer (J=24) and back (I=140) boundaries. Block-to-block conditions are also implemented (BCTYPE=4), at the interface between the blocks at the -45, 45, 135, and 225 degree positions (K=1 and 10). The final two input cards involve the stagnation line out the front of the configuration, along which a symmetry condition must be implemented (BCTYPE=5). This aids in resolving the flux through what is effectively the zero area cells on the I=1 surface. And finally, the input stream is concluded with a BCTYPE=0 card terminating all input for this run. The computational grid is shown in Figure 15 to give the proper implementation of the boundary condition types (BCTYPEs).

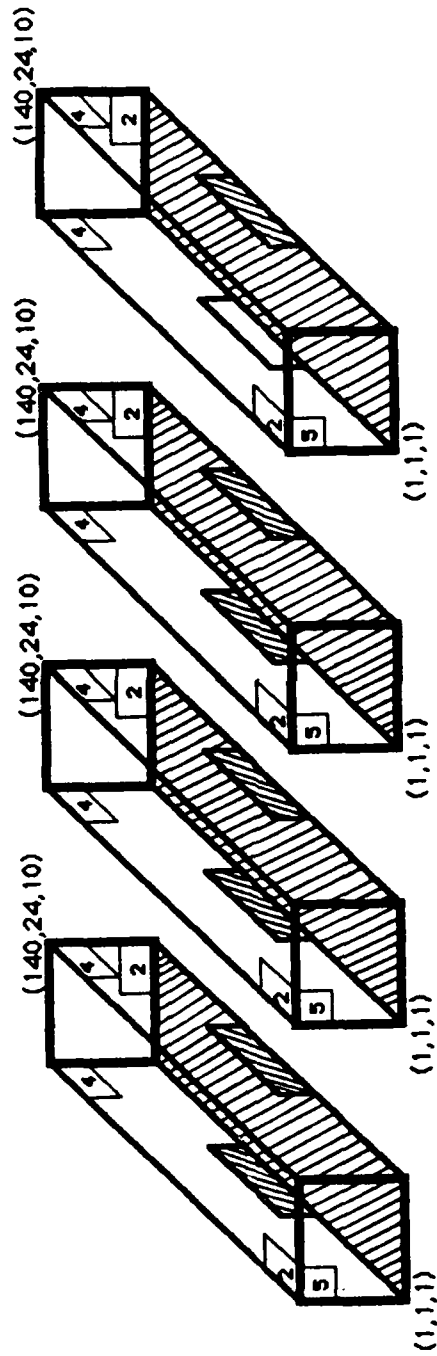


Figure 15. Generic Finned Configuration Computational Region

c. Flow Solver Output

The output for the Generic Store configuration shows the entire run took 10077.12623 CP seconds for 1000 iterations. The density residuals converged almost three orders of magnitudes as did the L2 Norm depending on block number. The number of supersonic points appears to be "frozen" for the last 20 iterations. This seems to be a fairly well converged solution and should yield excellent engineering approximations to the flow at these conditions.

SECTION VI

RESULTS

This section summarizes the results obtained with the EAGLE-Flow Solver over the past few months of development and validation. The code has been exercised thoroughly on configurations ranging from the NACA 0012 airfoil (using grids created by the EAGLE - Numerical Grid Generation System as well as the GRAPE code developed by Sorenson ¹⁷) to the three store mutual interference configuration (Section 6.4). An entire range of Mach numbers (from 0.40 to 6.81) and angles of attack (from 0 to 20 degrees) have been run to test the code's ability to solve for difficult flow conditions. This section provides just a sampling of these results.

1. NACA 0012 AIRFOIL

The NACA 0012 airfoil has been used as a test configuration due to its vast amount of documentation by other code development efforts and because it is so well understood from a theoretical standpoint. The test case run in Section 5.1 yields excellent results for the flow conditions specified (Mach= 0.80, AOA= 1.25 degrees). The code captures both the upper surface shock at the 55% chord location, as well as the weak lower surface shock present at the 35% chord location (Figure 16). No experimental data is shown; however, the solution compares favorably to other codes¹⁸ results.

NACA 0012 Airfoil (221 X 20 X 2)
Surface Pressure Distribution

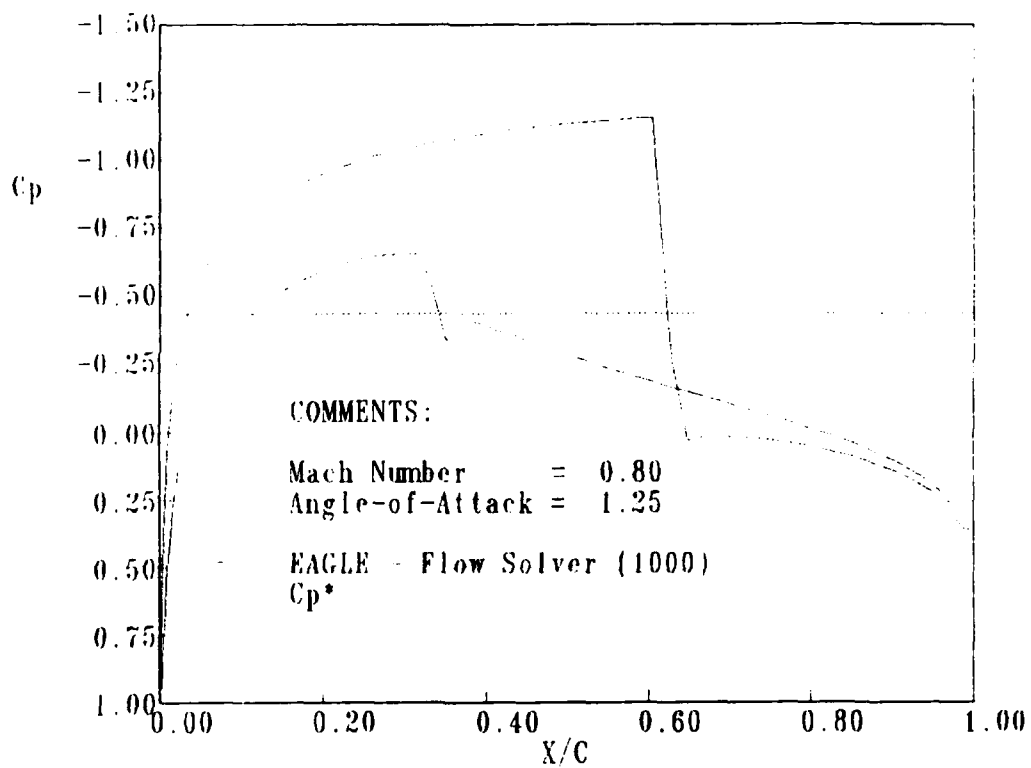


Figure 16. NACA 0012 Airfoil - Flow Solver Output

2. GENERIC FINNED CONFIGURATION

The Generic Finned Configuration (Ogive-Cylinder-Ogive with fins) has been exercised for several transonic Mach numbers (0.80 through 1.20) and angles of attack. This configuration has been the "bread and butter" test case for the Armament Laboratory for freestream, and interference flow, analysis in the transonic Mach range.^{19,20} The following results are provided at Mach 1.05 at two angles of attack to show the codes strengths and weaknesses as flow conditions become increasingly viscous in nature.

Figure 17 shows the inviscid solutions obtained for the sample case presented in Section 5.2 (Mach= 1.05 and AOA= 15 degrees) versus experimental data¹⁹. In Figure 17a, the body pressure distribution is shown at an orientation angle of 0 degrees (or twelve o'clock- the leeward side). The solver does an excellent job of following the flow as it accelerates over the ogive nose and expands near the shoulder region of the body. The inviscid approximation does begin to fail at about 80% as the flow separates over the ogive boattail section. Figure 17b shows the body pressures for the windward side (orientation of 180 degrees or six o'clock position). The flow solver again captures the flow as it expands over the nose and does a good job predicting the flow in the tail section (as the flow stays attached on the windward side). In Figure 17c fin pressures are shown versus experimental data at a fin orientation of 45 degrees and a span location of 20% on the leeward side of the fin. As expected, the inviscid solver doesn't capture the physics of the

Finned, Ogive Cylinder-Ogive Surface Pressure Distribution

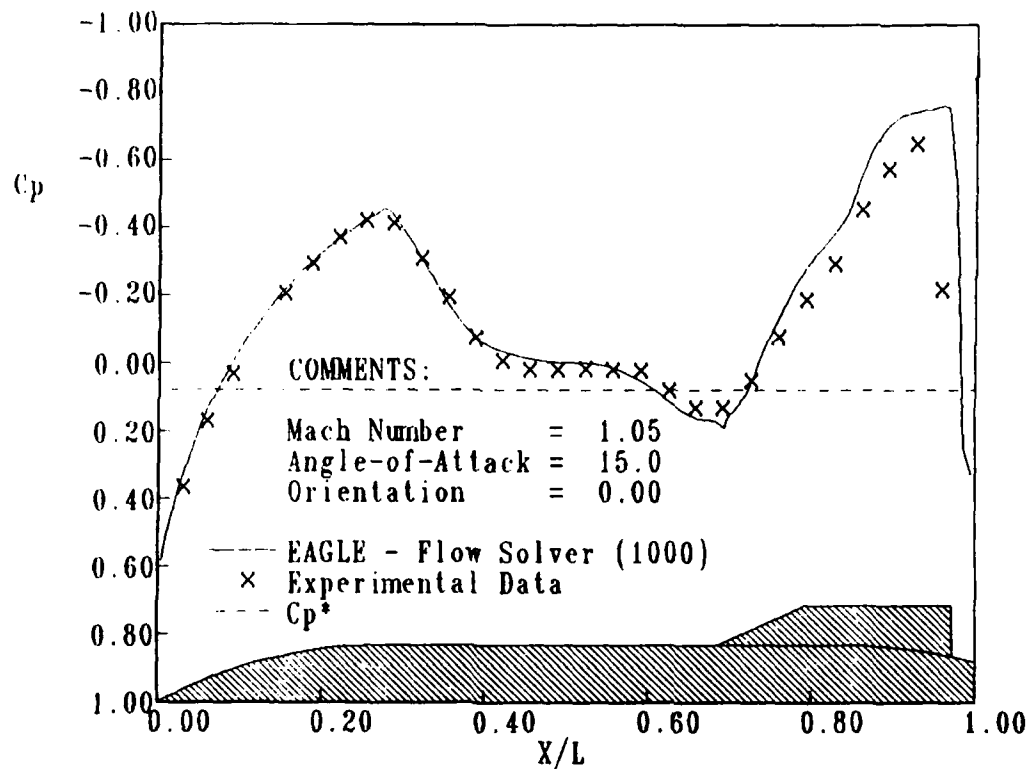


Figure 17a. Generic Finned Configuration - Flow Solver Output
Body Pressure Distribution ($\alpha = 15$ Degrees)

Finned, Ogive Cylinder Ogive Surface Pressure Distribution

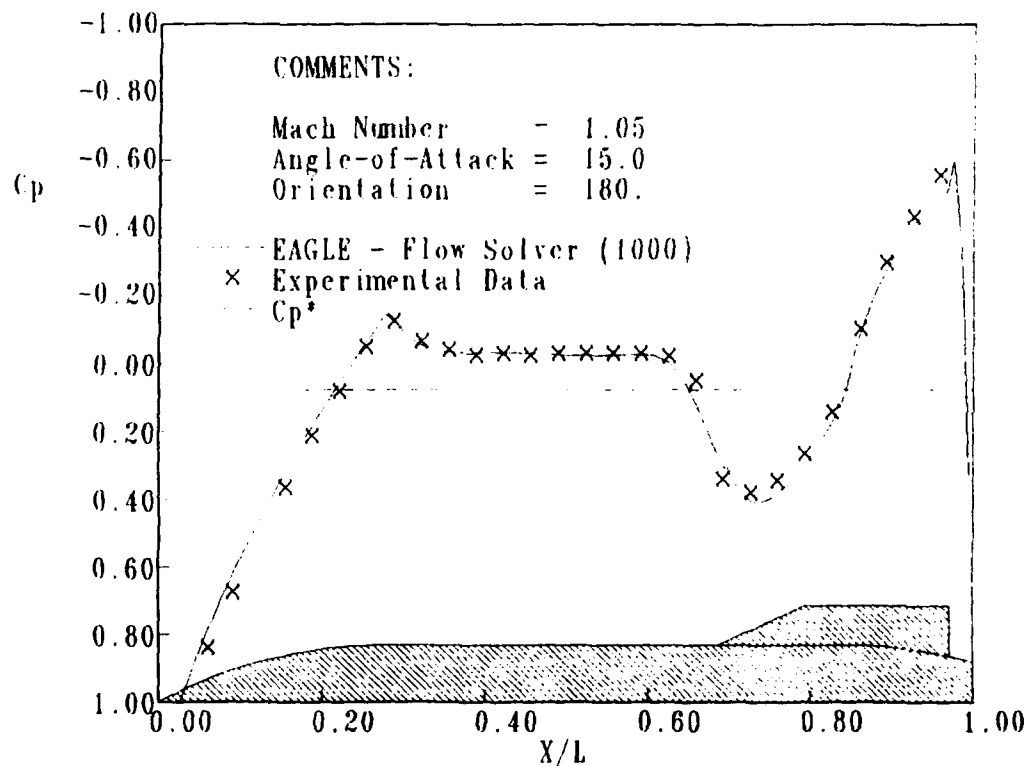


Figure 17b. Generic Finned Configuration - Flow Solver Output
Body Pressure Distribution ($\alpha = 15$ Degrees)

Finned, Ogive Cylinder Ogive Fin Pressure Distribution

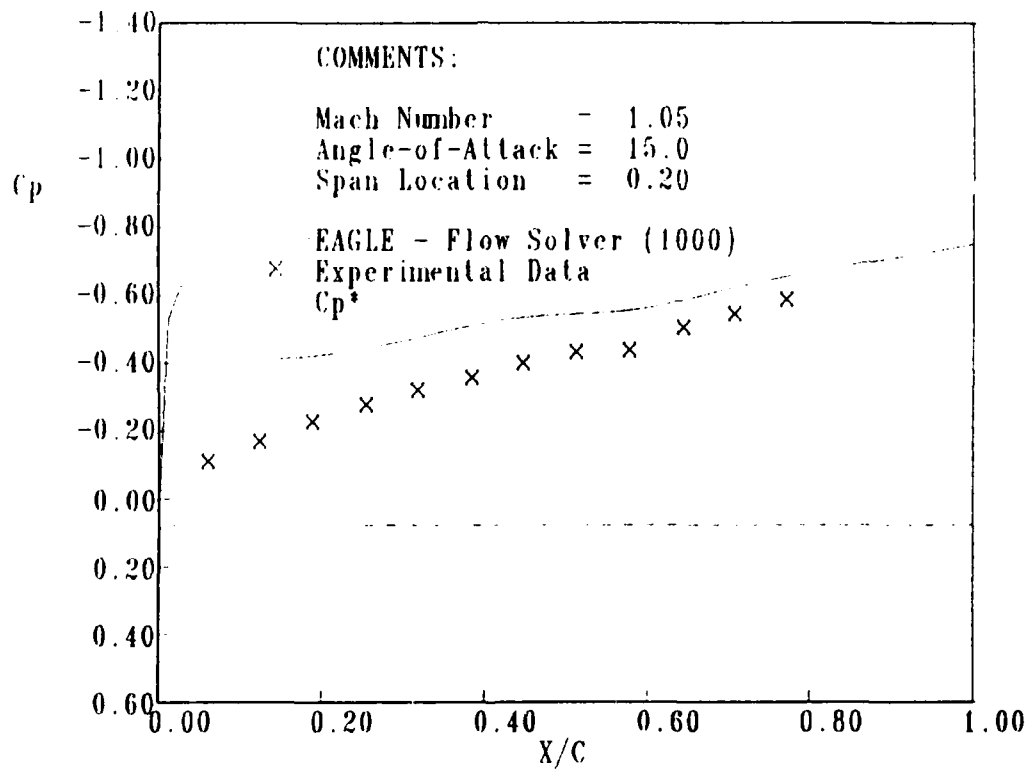


Figure 17c. Generic Finned Configuration - Flow Solver Output
 Fin Pressure Distribution ($\alpha = 15$ Degrees)

Finned, Ogive Cylinder Ogive Fin Pressure Distribution

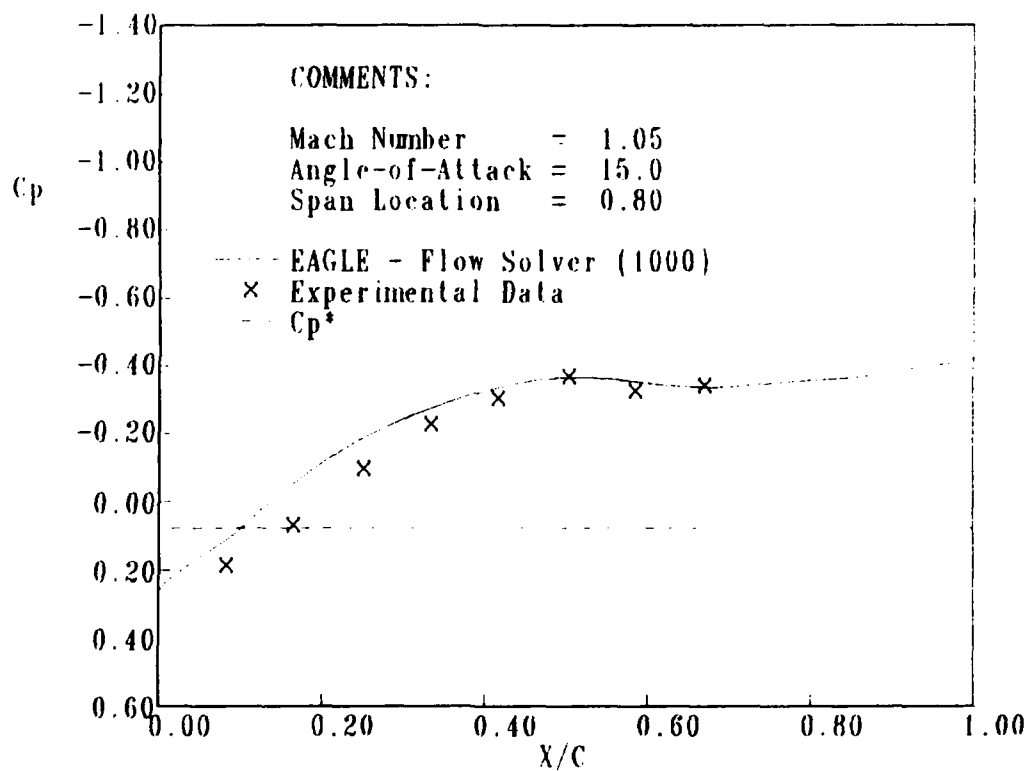


Figure 17d. Generic Finned Configuration - Flow Solver Output
Fin Pressure Distribution ($\alpha = 15$ Degrees)

flowfield in an area where separated flow predominates the flowfield (leeward side of the body on the leeward side of the fin). Figure 17d shows the windward side of the same fin (fin orientation of 45 degrees) at a span location near 80%. The solver does a better job of capturing the expansion over the leading edge and, because of a lack of data points, there is no judging the trailing edge.

Figure 18 gives results obtained by the flow solver at Mach = 1.05 and an angle of attack of 6 degrees. The code does an excellent job of capturing the expansion over the nose on to the shoulder region of the body at the leeward (0 deg.), windward (180 deg.) and the side meridian (90 deg.) positions (Figure 18a). Figure 18b,c shows results obtained for the windward fins (at 135 deg.) on the leeward side at four fin span locations and for the same fin (135 deg.) on the windward side at four fin span locations. In both cases the code does a good job of predicting the surface pressures along the chord length of the fin.

3. BLUNT, FINNED BODY OF REVOLUTION WITH CANARDS

This configuration has been used as a test case for the EAGLE - Numerical Grid Generation System, as well as for the EAGLE - Flow Solver, due to its difficult canard set and multi-block layout shown in Reference 21. Several studies have been accomplished on this body due to its ability to create a multiple shock structure.^{20,22} The results shown here in Figure 19 are for a Mach number of 0.95 and angle of attack of zero degrees for both the EAGLE - Flow Solver and a previous

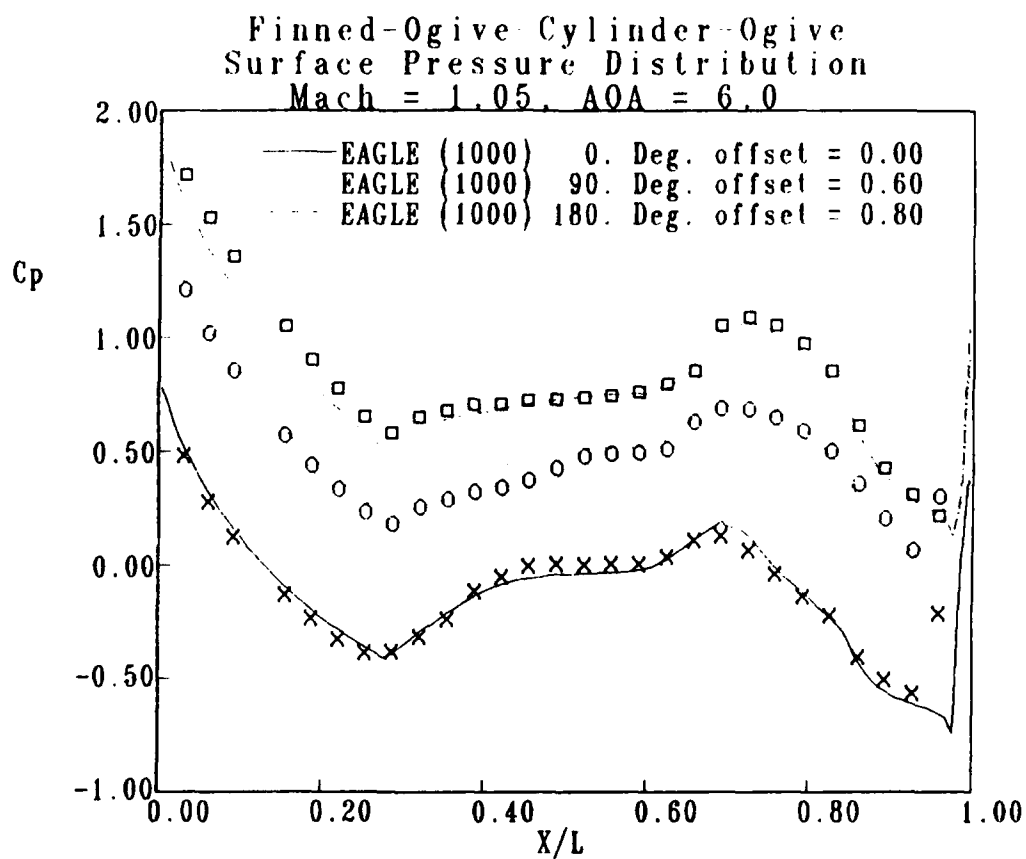


Figure 18a. Generic Finned Configuration - Flow Solver Output
Body Pressure Distribution ($\alpha = 6$ Degrees)

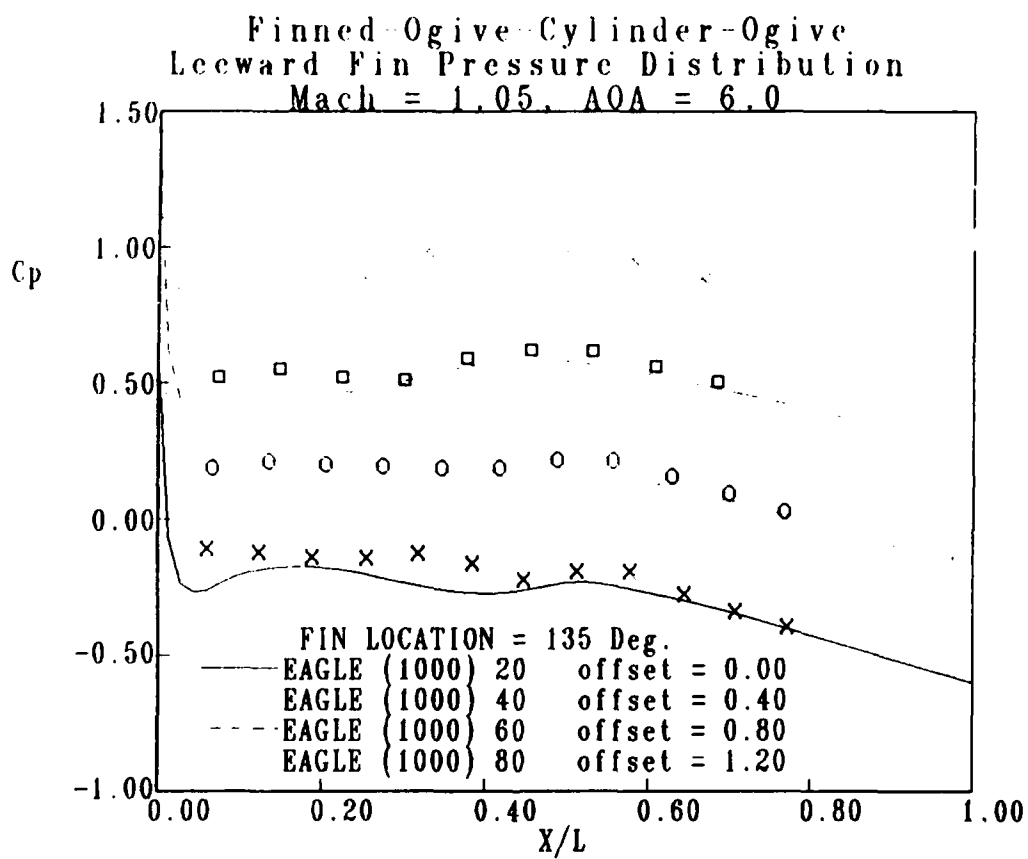


Figure 18b. Generic Finned Configuration - Flow Solver Output
 Fin Pressure Distribution ($\alpha = 6$ Degrees)

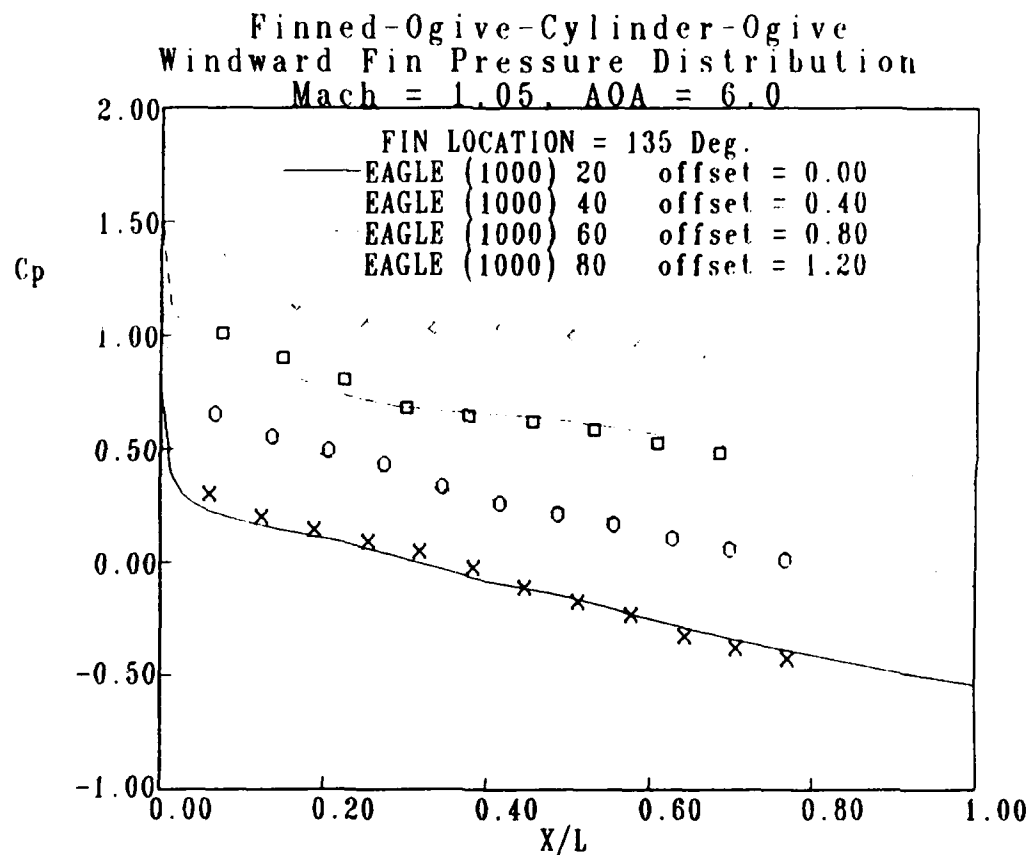


Figure 18c. Generic Finned Configuration - Flow Solver Output
Fin Pressure Distribution ($\alpha = 6$ Degrees)

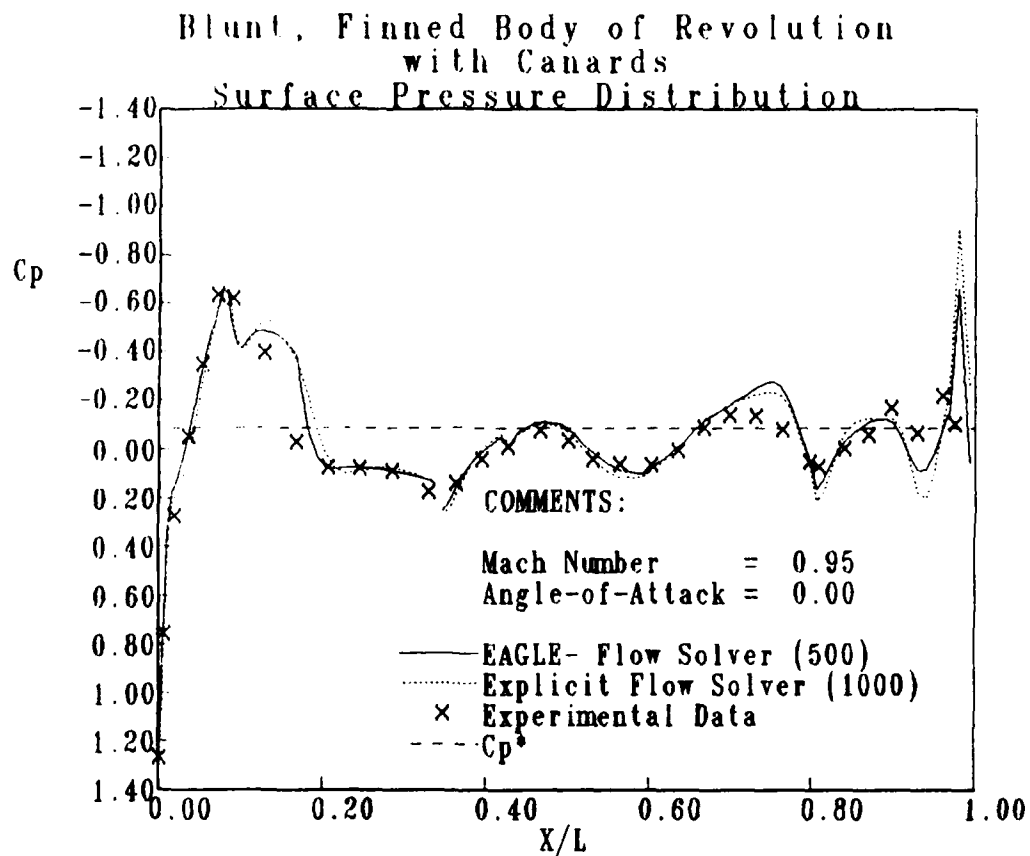


Figure 19a. Blunt, Finned Body of Revolution with High Taper Ratio Canards

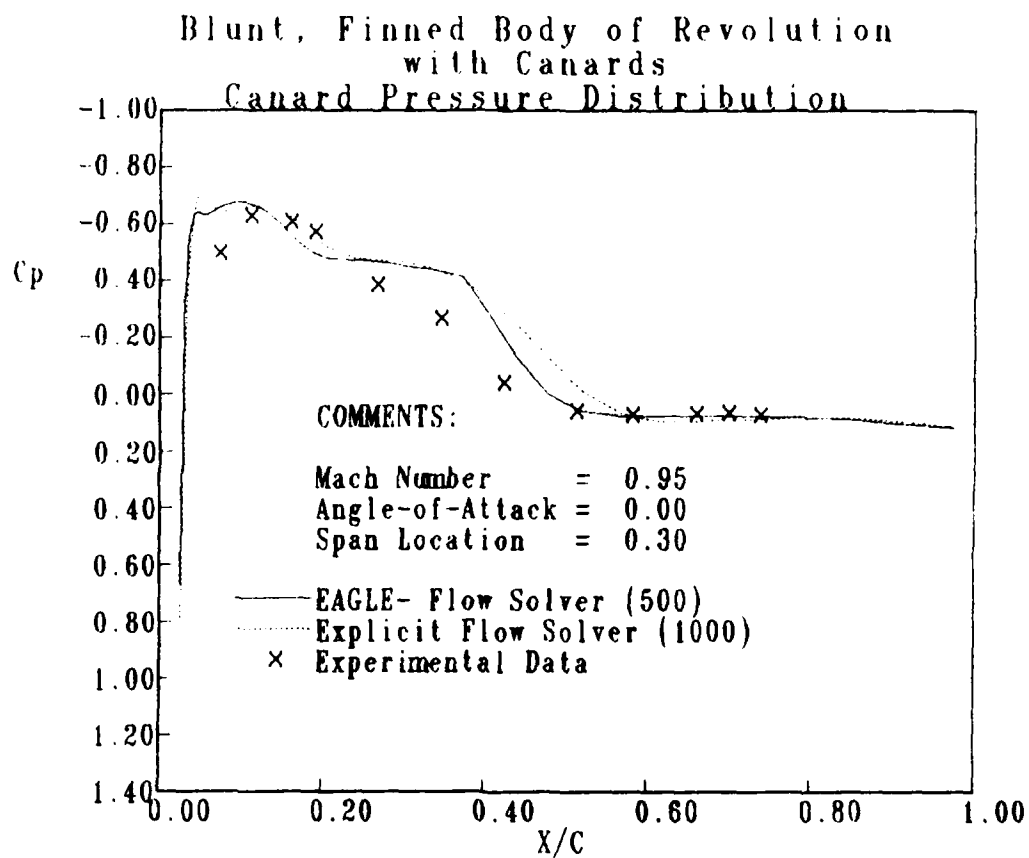


Figure 19b. Blunt, Finned Body of Revolution with High Taper Ratio Canards

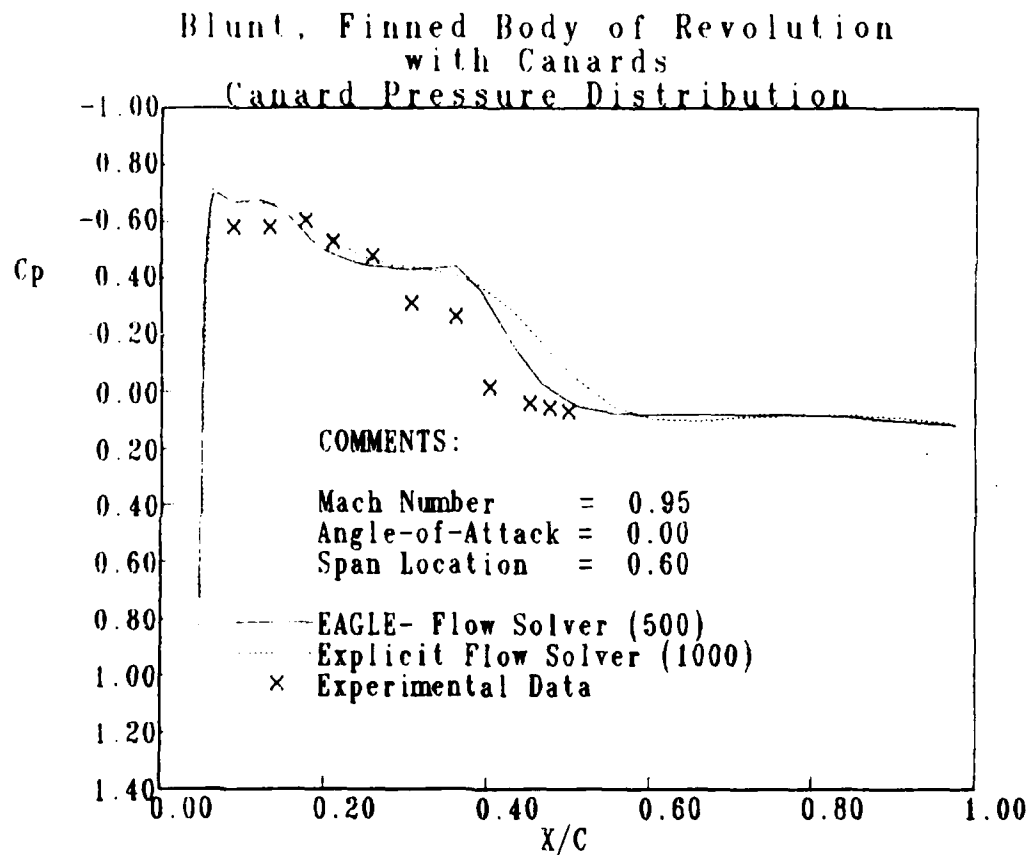


Figure 19c. Blunt, Finned Body of Revolution with High Taper Ratio Canards

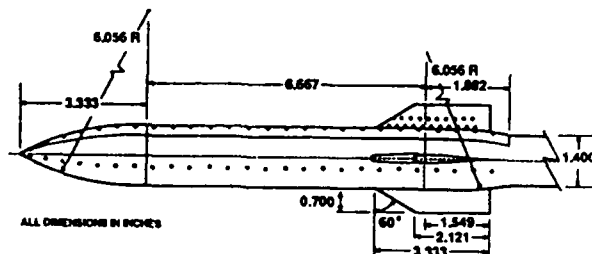
explicit algorithm that was a candidate for inclusion in the code²⁰. Both codes do an excellent job of following the expansion over the blunt nose and past the canards at the 15% - 20% locations. The solver predicts the complex flow pattern throughout the rest of the field, but fails at the boattail section as the flow separates off the back end (Figure 19a). In Figure 19b,c canard pressures are shown for two span locations (30% and 60%). In both cases the codes do well in following the pressure distribution over the canard shape.

4. MUTUAL INTERFERENCE CONFIGURATIONS

To illustrate the capability for predicting interference flow conditions, the flow solver was run for a set of configurations shown in Figure 20. The geometry is that shown in Figure 14 and a discussion of the single and triple store grids can be found in Volume I.

Solutions were obtained for the three store configurations at two Mach numbers (0.80 and 0.95) at zero and ten degrees angle of attack. For the one, two and three store cases, Figure 21 plots the surface pressure distribution on the inboard side of the body (265 deg. location). As expected, the interference increases as the second and third bodies are included. The aerodynamic interference manifests itself in stronger expansions just after the shoulder region that increases in size downstream as the other bodies are added. The flow solutions agree quite well with the experimental data; however, the code over predicts the expansions at the shocks as well as the shock locations.

WIND TUNNEL MODEL GEOMETRY



STORE CONFIGURATIONS

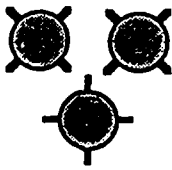
X FIN ATTITUDE	CONFIG
	TRIPLE
	DOUBLE
	SINGLE

Figure 20. Multiple Body Configurations

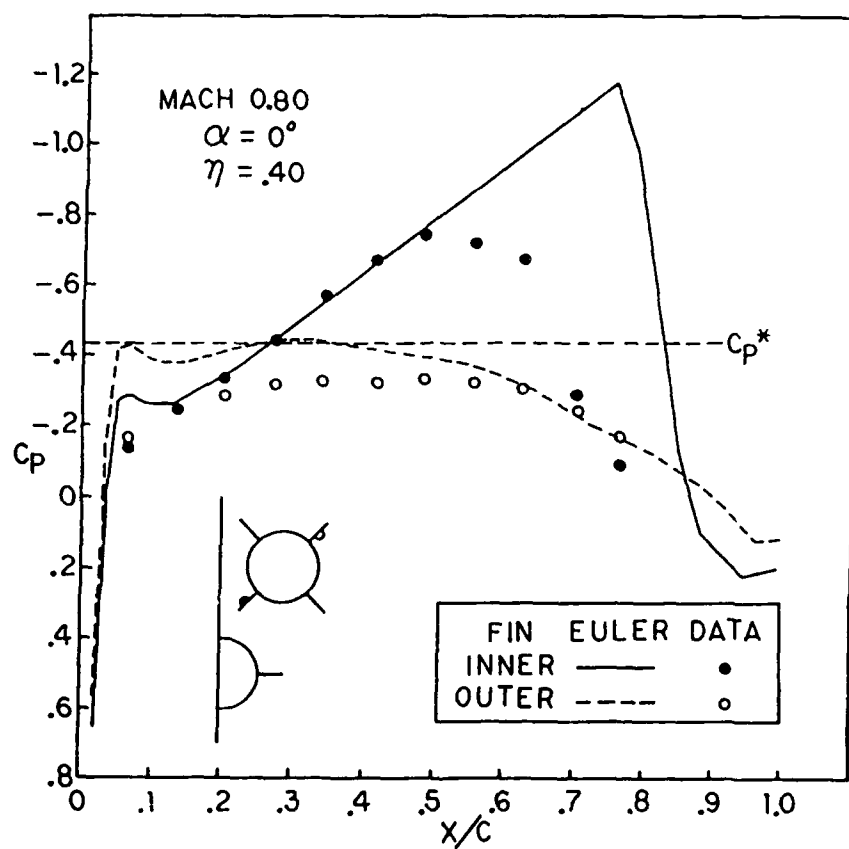


Figure 21. Configuration Complexity Effects

To investigate their effect, calculations were obtained for both the finned and unfinned configurations. Figure 22 shows the effects of the fins on the double (or two store) configuration at Mach= 0.95 and zero degrees angle of attack. The effect seen on the body in the interference flow region is a reduction of the expansion just aft of the nose for the finned body, but an increased expansion in the fin area. The fins also increase blockage in the flow, moving the forebody shock forward 3%.

To examine the interference effects at angle of attack, the triple store configuration with fins was run for both zero and ten degrees angle of attack at Mach= 0.80. The surface pressure distribution for these cases is shown in Figure 23. As the configuration increases in incidence angle, the forebody expansion decreases on the leeward side of the lower body. The upper two stores in the three store configuration act to redirect the flow over the lower body and effectively reduce the angle of attack seen by the lower body. This accounts for the weaker forebody expansion seen at the ten degree incidence angle.

The interference effect of the fins was investigated at Mach= 0.80 and zero degree incidence. Figure 24 illustrates the pressure distributions on the fins for the three store configuration at the positions of minimum and maximum interference. The pressure distribution on the minimum interference fin is nearly flat while a significant expansion and shock occurs on the maximum interference fin. This is expected since the flow accelerates through the region between the three

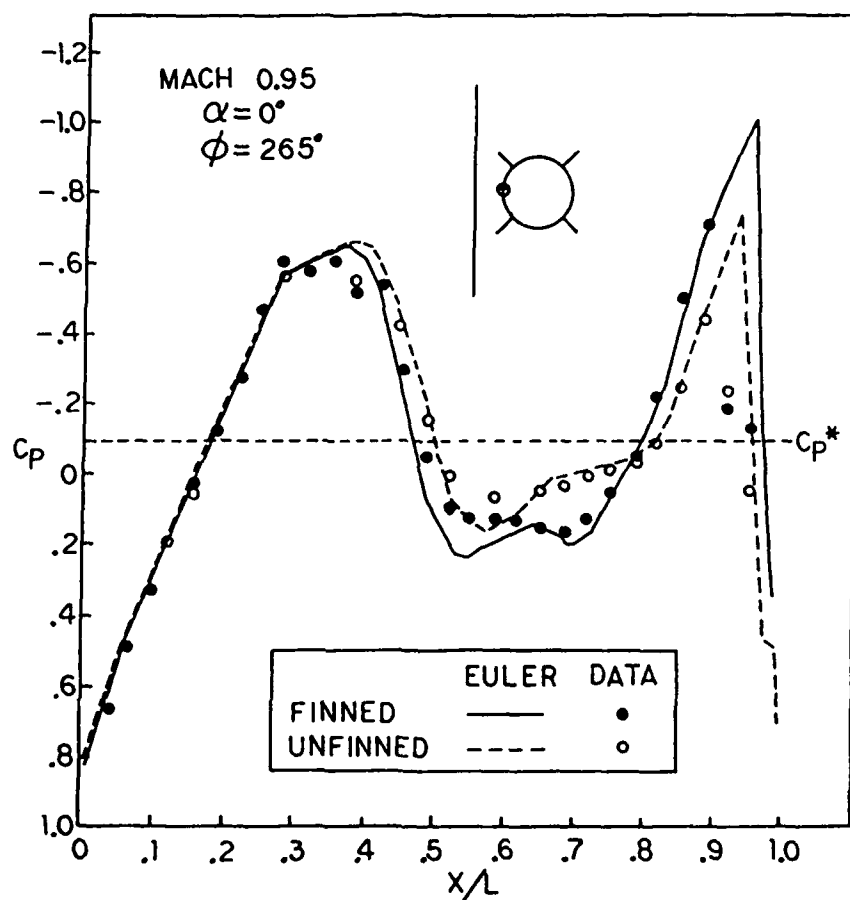


Figure 22. Fin Effects on 2-Body Case

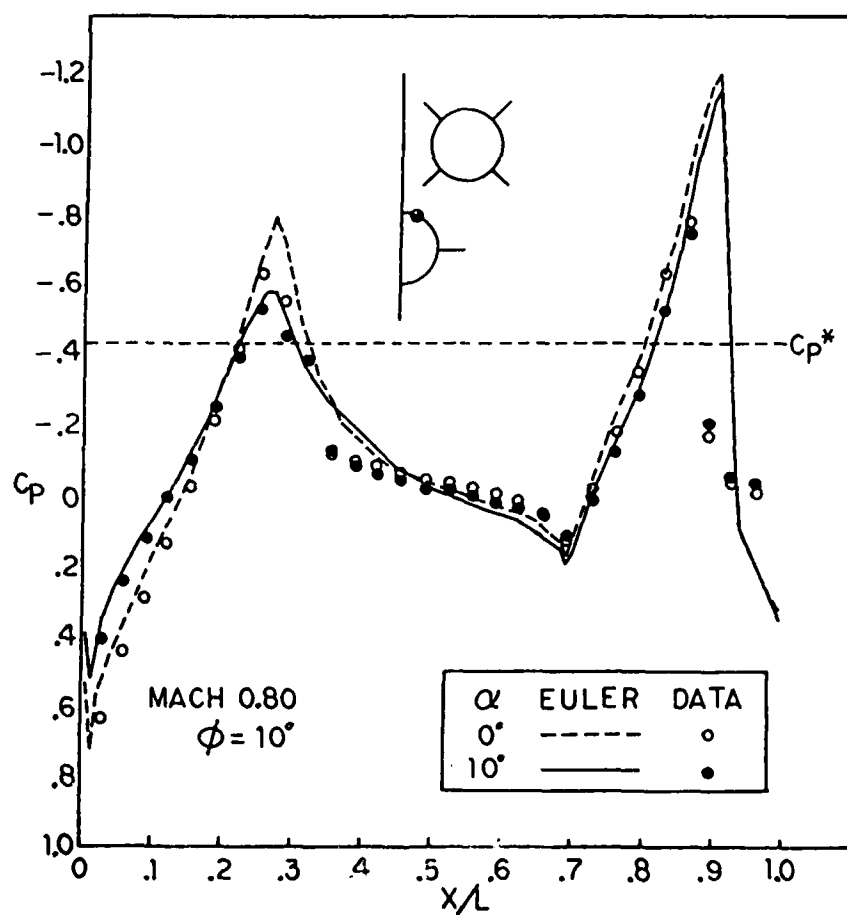


Figure 23. Angle of Attack Effects, 3-Body Case

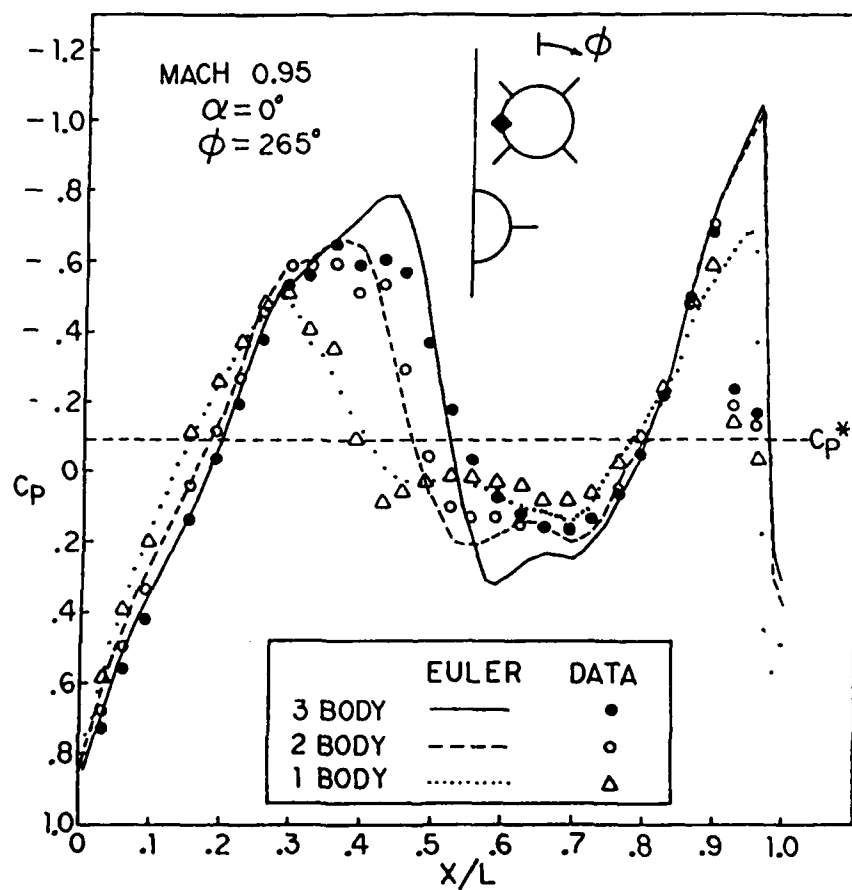


Figure 24. Interference Effects on Fin Pressures, 3-Body Case

bodies lowering the pressure. The flow solver does a fair to good job on the first 50% of the chord for both fins. The shock on the inboard fin is missed substantially both in strength and location.

SECTION VII

SUMMARY

As presented here, the Program EAGLE - Flow Solver can be employed as an extremely effective aid in the analysis of freestream and interference flow aerodynamic characteristics of advanced, arbitrarily shaped airframe configurations.

The code appears to be most effective in the high subsonic to low supersonic Mach range (0.75 to 2.0), but has shown excellent results outside of this range. As is evident from the inviscid nature of the code, the solver has difficulty resolving viscous dominated regions (separated flow) and; therefore is limited to relatively small incidence angles. However, the flow solver has shown excellent results on body pressure distributions at incidence angles as high as 15 degrees.

Good engineering solutions can also be obtained for complex geometries in interference flowfields as is evident from Section 6.4. The solutions obtained show that an inviscid approximation is quite adequate for most aspects of the fluid dynamic phenomena; however, the solution does break down in highly viscous regions such as the positions of maximum interference where the boundary layer is "sucked" back into the channel between the stores.¹⁹

In general, the EAGLE - Flow Solver is an excellent analytical tool for the practicing engineer in need of a compliment to experimental analysis. The EAGLE code should yield extremely accurate predictions of the aerodynamic characteristics

of any arbitrarily shaped advanced airframe configuration in freestream or interference flowfields. The recommended range of implementation is for subsonic to low hypersonic Mach numbers (where there are no real gas effects present) [$0.40 < \text{Mach} < 8.0$] at relatively low angles of incidence (where the separated region is relatively small) [0 to 20 degrees].

REFERENCES

1. Belk, D.M., Unsteady Three-Dimensional Euler Equations Solutions on Dynamic Blocked Grids, Ph.D. Dissertation, Mississippi State University, August 1986.
2. Janus, J.M., The Development of a Three-Dimensional Split Flux Vector Euler Solver with Dynamic Grid Applications, Masters Thesis, Mississippi State University, August 1984.
3. Beam, R.M. and Warming, R.F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Volume 16, Number 4, pp. 393-402, April 1978.
4. Briley, W.R. and MacDonald, H., "On the Structure and Use of Linearized Block Implicit Schemes," Journal of Computational Physics, Volume 34, pp. 54-73, 1980.
5. Steger, J.L. and Warming, R.F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite-Difference Methods," Journal of Computational Physics, Volume 40, Number 2, pp. 263-293, April 1981.
6. Whitfield, D.L., "Implicit Upwind Finite Volume Scheme for the Three-Dimensional Euler Equations," Mississippi State University Report, MSSU-EIRS-ASE-85-1, September 1985.
7. Anderson, W.K., Thomas, J.L. and Whitfield, D.L., "Multigrid Acceleration of the Flux Split Euler Equations," AIAA-86-0274, January 1986.
8. Anderson, W.K., Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations, Ph.D. Dissertation, Mississippi State University, August 1986.
9. Whitfield, D.L., Janus, J.M. and Simpson, L.B., "Implicit Finite Volume High Resolution Wave-Split Scheme for Solving the Unsteady Three-Dimensional Euler and Navier-Stokes Equations on Stationary or Dynamic Grids," Mississippi State University Report, MSSU-EIRS-ASE-88-2, February 1988.
10. Roe, P.L., "Approximate Riemann Solvers, Parameter Vector, and Difference Schemes," Journal of Computational Physics, Volume 43, pp. 357-372, 1981.
11. van Leer, B., Thomas, J.L., Roe, P.L. and Newsome, R.W., "A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations," AIAA-87-1104-CP, June 1987.
12. Roe, P.L. and Pike, J., "Efficient Construction and Utilization of Approximate Riemann Solutions," Computing Methods in Applied Sciences and Engineering, ed. R. Glowinski and J.L. Lions, 6:499-518, Amsterdam: North-Holland, 1984.
13. Osher, S. and Chakravarthy, S., "Very High Order Accurate TVD Schemes," ICASE Report Number 84-44, September 1984.

14. Roe, P.L., "Characteristic-Based Schemes for the Euler Equations," Annual Review of Fluid Mechanics, Volume 18, pp. 337-365, 1986.
15. Jacocks, J.L. and Kniele, K.R., "Computation of Three-Dimensional Flow Using the Euler Equations," AEDC-TR-80-49, July 1981.
16. Ross, D.T., Goodenough, J.B. and Irvine, C.A., "Software Engineering: Process, Principles and Goals," Computers, May 1975.
17. 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, May 1980.
18. Yee, H.C. and Harten, A., "Implicit TVD Schemes for Hyperbolic Conservation Laws in Curvilinear Coordinates," AIAA Journal, Volume 25, Number 2, AIAA-85-1513, pp. 266-274, February 1987.
19. Cottrell, C.J. and Lijewski, L.E., "A Study of Finned, Multi-body Aerodynamic Interference at Transonic Mach Numbers," AIAA-87-2480, August 1987.
20. Mounts, J.S., Belk, D.M. and Lijewski, L.E., "A Comparison of Explicit and Implicit Three-Dimensional Split Flux Euler Algorithms," AIAA-87-2413, August 1987.
21. Martinez, A. and Mounts, J.S., "Program EAGLE - Numerical Grid Generation System User's Manual: Volume I-Executive and Plotting Routines," AFATL-TR-87-15, April 1987.
22. Lijewski, L.E., "Transonic Flow Solutions on a Blunt, Finned Body of Revolution Using the Euler Equations," AIAA-86-1082, May 1986.