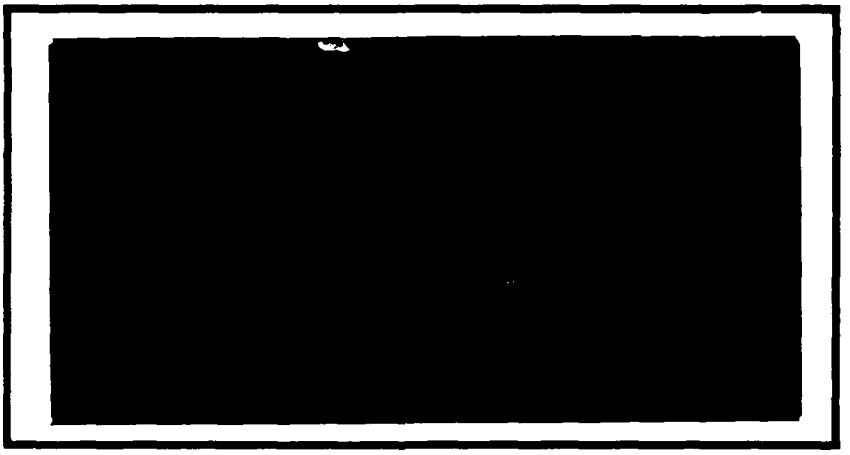


DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GAE/AA/88D-02

Navier-Stokes Solution For A NACA 0012 Airfoil With Mass Flux (Fan)

Thesis

Paul D. Boyles
Captain, USAF

AFIT/GAE/AA/88D-02

Approved for public release; distribution unlimited

A-1

AFIT/GAE/AA/88D-02

NAVIER-STOKES SOLUTION FOR A NACA 0012 AIRFOIL WITH MASS FLUX (FAN)

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Aeronautical Engineering

Paul D. Boyles

Captain, USAF

December 1988

Preface

The purpose of this study was to numerically simulate the flow field for NACA 0012 airfoil with mass transfer through the surface (fan). Recent emphasis in V/STOL and the advancements in today's technology has created an environment where a detailed knowledge of the flow characteristics involved in lift augmentation is needed. One particular area especially lacking in research is the "Fan-in-wing" device. Such a device was not practical in the 60's when it was studied for its vertical take off and hovering performance. However, with today's advances in material science, propulsion and aviation technology, such a device could be practical.

In two-dimensions, a "fan" is characterized by suction normal to the upper airfoil surface and vectored blowing on the lower airfoil surface. This study examines the lift and drag effects of such a device with varying suction velocity, ejection angle and angle of attack. The current "state of the art" in computational fluid dynamics (CFD) makes it possible to do this initial study by numerical simulation. It is hoped that the results obtained in this work can be an aid in the design of future experimental tests which will verify these numerical results and perhaps lead to V/STOL applications.

I would like to thank Dr. Halim for supervising this

work, and Capt Beran for his support and aid in this research. I would also like to thank Dr Shang, the sponsor of this research, and the wonderful group of people that work for him at the Flight Dynamics Laboratory. Without their technical guidance and sharing of computer resources, this thesis could not have been accomplished.

Finally, My wife and two small children need a special thanks for their patience, understanding and support during the many late nights and busy weekends over the past five months. I can now begin to try and make it up to them.

Table of Contents

	Page
Preface	ii
List of Figures	vi
List of Tables	ix
List of Symbols	x
Abstract	xiv
I. Introduction	1
II. Analysis	14
Governing Equations	14
III. Numerical Solutions of the NS Equations	28
Implicit Navier-Stokes Code	28
Grid Generation	33
Convergence Criteria	34
IV. Results and Discussion	37
Code and Grid Verification	37
Suction and Ejection Separately	44
Convergence Criterion for Periodic Solutions	55
The Effect of Varying Suction Rate	70
The Effect of Varying Ejection Angle	78
The Effect of Varying Angle of Attack	85
V. Conclusions and Recommendations	91
Appendix A: Non-Dimensional Variables	93
Appendix B: Jacobian Matrices for the Navier-Stokes Equations	96
Appendix C: Boundary and Initial Conditions	98
Boundary Conditions Without Blowing	98
Boundary Conditions for Suction and Blowing	100
Initial Conditions	103
Appendix D: Derivation of Force Coefficients C_l and C_d	105

Appendix E: Computer Resource Requirements..... 109

Appendix F: Navier-Stokes Code, Fortran Listing.. 111

Appendix G: Hyperbolic Grid Generating Code, Fortran Listing..... 112

Appendix H: Data Reduction Program 1, Fortran Listing..... 113

Appendix I: Data Reduction Program 2, Fortran Listing..... 114

Appendix J: Geometric Progression Program, Fortran Listing..... 115

Appendix K: Grid Redistribution Program, Fortran Listing..... 116

Bibliography..... 117

Vita..... 121

List of Figures

Figure		Page
1.	A "Fan-in-Wing".....	2
2.	A "Fan-in-Wing" Aircraft.....	2
3.	Propulsion System for the Ryan XV-5A.....	3
4.	V/STOL Aircraft Configurations.....	4
5.	Effect of Boundary Layer Geometry on Stability.....	5
6.	Effect of Suction on the Boundary Layer.....	6
7.	Boundary Layer Control a)Blowing, b)Slotted Wing, c)Suction.....	7
8.	Blowing Non-Tangent to the Surface.....	8
9.	Three-Dimensional Effects of a Fan.....	9
10.	General Transformation from the Physical Domain to the Computational Domain b.....	18
11.	Final Grid Configuration for a NACA 0012 Airfoil.....	39
12.	Final Grid Configuration, Close-Up.....	40
13.	Comparisons of C_p with Experimental Data....	43
14.	Comparisons of C_p for Transition Point Effects.....	45
15.	Comparisons of C_f for Transition Point Effects.....	46
16.	Comparisons of C_p for Suction and Blowing Effects.....	48
17.	Comparison of C_f for Suction and Blowing Effects.....	49

18. Velocity Boundary Layer Profile above Suction Surface..... 51

19. Velocity Profiles above Suction and Blowing Surfaces..... 53

20. Velocity Profiles of Trailing Edge Vortex and Shear Layer..... 54

21. Streamline Comparisons for one Period in C_l. 56

22. Vorticity Contour Comparisons for one Period in C_l..... 57

23. Force Coefficients vs Iterations, Number 3.. 59

24. Force Coefficients vs Iterations, Number 4.. 60

25. Force Coefficients vs Iterations, Number 5.. 61

26. Force Coefficients vs Iterations, Number 6.. 62

27. Force Coefficients vs Iterations, Number 7.. 63

28. Force Coefficients vs Iterations, Number 8.. 64

29. Force Coefficients vs Iterations, Number 9.. 65

30. Force Coefficients vs Iterations, Number 10. 66

31. Force Coefficients vs Iterations, Number 11. 67

32. Force Coefficients vs Iterations, Number 14. 68

33. Force Coefficients vs Iterations, Number 17. 69

34. Force Coefficients vs Suction Velocity..... 71

35. Comparisons of C_p for Different Suction Velocities..... 73

36. Comparisons of C_f for Different Suction Velocities..... 75

37. C_p Comparisons for one Period in C_l..... 77

38. Streamline and Vorticity Comparisons for Different Suction Velocities..... 79

39. Force Coefficients vs Ejection Angle..... 80

40. Streamline and Vorticity Comparisons for Different Ejection Angles..... 83

41. Streamline and Vorticity Comparisons for Different Ejection Angles..... 84

42. Force Coefficients vs Angle of Attack..... 86

43. Comparisons of C_p for Different Angles of Attack..... 87

44. C_l vs C_d for Angle of Attack..... 88

45. Streamline and Vorticity Comparisons for Different Angles of Attack..... 90

46. Boundary Conditions..... 101

List of Tables

Table		Page
1.	NACA 0012 Grid Configurations.....	38
2.	Separation Point Comparison.....	41
3.	Lift and Drag Comparison with Experimental Data.....	42
4.	C_l and C_d vs Re and Transition Point.....	42
5.	C_l and C_d Comparison with Experimental and Theoretical Data.....	44
6.	C_l and C_d Comparison, Cv = .05, $\delta = 90^\circ$.....	47
7.	Velocity and density Comparison at the Suction Surface.....	50
8.	List of Model Configurations for M = .3 and Re = 1,000,000.....	58
9.	C_l and C_d Component Comparison vs Suction Velocity.....	74
10.	C_l and C_d Component Comparison vs Ejection Angle.....	81
11.	C_d Calculation for Surface Blowing.....	82
12.	C_l and C_d Component Comparison vs Angle of Attack.....	89
13.	Computer Resource Comparison.....	109

List of Symbols

Symbol	Definition
A	Area
$\mathbf{A}$	Jacobian matrix
a	Speed of sound
$\mathbf{B}$	Jacobian matrix
C_d	Drag coefficient
C_l	Lift coefficient
C_p	Pressure coefficient
c	Airfoil cord
c_p	Specific heat at constant pressure
CFL	Courant number
Cr	Fan compression ratio
Cv	Suction velocity
D_t, D_θ	Damping Coefficients
d	Indicates derivative
E_t	Total internal energy
e	Specific internal energy
F, $\hat{F}$	Flux vectors
$\bar{f}$	Body force vector
G, $\hat{G}$	Flux vectors
I	Identity matrix
IL	number of grid points in η - direction
J	Transformation Jacobian
JL	Number of grid points in ξ - direction

k	Coefficient for thermal conductivity
k_t	Coefficient for turbulent thermal conductivity
L	Reference length
M	Mach number
M	Jacobian matrix
$\dot{m}$	Mass flow rate
N	Number of iterations
N	Jacobian matrix
P	Pressure
Pr	Molecular Prandtl number
Pr_t	Turbulent Prandtl number
Q	Internal heat generation
$\bar{q}$	Heat conduction vector
R	Perfect gas constant
Re	Reynolds number
s	Surface area
T	Temperature
t	Time
U, $\hat{U}$	Vectors of dependable variables
U_∞	Free stream velocity
u	Stream wise velocity component
u	Contravariant velocity
$\bar{V}$	Velocity vector
v	Normal velocity component
v	Contravariant velocity
x	Stream wise direction coordinate

y	Normal direction coordinate
α	Angle of attack
γ	Specific heat ratio
Δ	Difference operator
δ	Ejection angle
δ	Finite difference operator
ε	Turbulent eddy viscosity
η	Coordinate in computational space normal to the body
θ	Slope on the airfoil surface
ξ	Coordinate in computational space tangent to the body
λ	Coefficient of viscosity
λ_t	Coefficient of viscosity including turbulent properties
μ	Molecular dynamic viscosity
ρ	Density
$\underline{\sigma}$	Shear stress tensor
$\underline{\tau}$	Viscous stress tensor
ω	Damping coefficients
∂	Indicates partial derivative

Superscripts

T	Transpose of matrix

Subscripts

i,j,k	Index notation
i,j	Mesh indices

x	Partial derivative with respect to x
y	Partial derivative with respect to y
η	Partial derivative with respect to η
ξ	Partial derivative with respect to ξ
∞	Denotes free stream conditions

Abstract

The purpose of this study was to determine, through numerical simulation, the two-dimensional effect of mass transfer (fan) on a NACA 0012 airfoil. A "fan" is characterized by suction on the upper surface of an airfoil and corresponding blowing from the lower surface. The results are presented through the comparisons of lift and drag coefficients, pressure and skin friction coefficient profiles, and streamline and vorticity contours. The numerical code used in this study is based on the Beam-Warming approximate factorization algorithm which is used to solve the mass-averaged, compressible Navier-Stokes equations for viscous, unsteady flows. The grid used in this study was a c-grid produced from a hyperbolic grid generating code.

This study examined the effect of varying suction velocity (1, 5 and 10 percent of the free stream velocity), ejection angle (15, 45 and 90 degrees) and angle of attack (0, 2 and 4 degrees). The results indicate that suction has only a small influence on lift; but, produces a large viscous drag proportional to the suction velocity. Blowing produces a large lift component due to a modified pressure distribution surrounding the leading edge and only a small drag penalty. Blowing also creates an unsteady periodic solution. The unsteady behavior is the result of vortex

shedding caused by the interaction of a broad shear layer formed behind the ejection surface which induces a circulation around the trailing edge. The shedding of the trailing edge vortex creates periodic oscillations in lift and drag, but is not responsible for the large mean increase seen in lift.

Increasing suction velocities produced a linear increase in lift. A maximum drag penalty occurred at medium values of suction (5%). The mass flow for the suction velocities studied were too small to provide substantial lift; but, did provide a substantial thrust component at an ejection angle of 15° and a suction velocity of 10%.

Ejection angles mildly affected lift; but reduced drag considerably at lower ejection angles (15°) due to thrust. Increasing angles of attack (α) produced a large linear increase in lift and a corresponding reduction in drag near that of a clean airfoil.

The results indicate that, in two-dimensions, a "fan" type device can produce aerodynamic benefits and warrants additional experimental and numerical studies, with an emphasis on the lift curve slope and stall behavior. The effect of different fan geometries, boundary conditions and cambered airfoils should also be investigated.

NAVIER-STOKES SOLUTION FOR A NACA 0012 AIRFOIL WITH MASS FLUX (FAN)

I Introduction

The effect of mass flux on a NACA 0012 airfoil section is the focus of this research project. This study is accomplished through the use of numerical analysis based on the mass averaged, compressible Navier-Stokes (NS) equations. For this report mass flux is defined by the intake of mass, suction, on the upper surface of the airfoil, and the ejection of mass, blowing, on the lower surface. The mechanism providing the suction and blowing will be referred to as a "fan" (see Figure 1). Though in two-dimensions this behavior could be contributed to a variety of devices.

The concept of a "fan-in-wing" (see Figure 2) has been around since the late 50's, reached its peak in the late 60's and was essentially discarded in the mid 70's as being impractical. The "fan-in-wing" was first considered in the 50's for its vertical/short take off and landing (V/STOL) ability (23:8). However, the technology and material sciences of the times forced designs that were heavy and large. A typical size fan covered between 40 to 50 percent of the wing area, "flying fan".

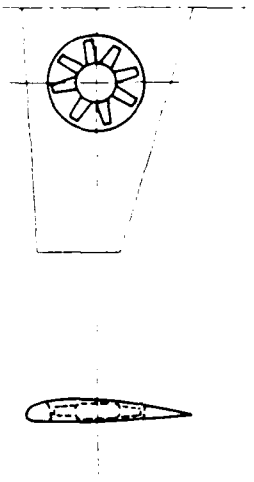


Figure 1. A "Fan-in-Wing" (23:23)

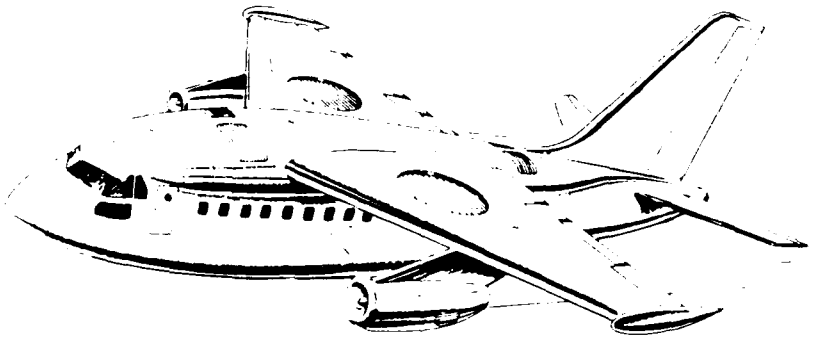


Figure 2. A "Fan-in-Wing" Aircraft (23:8)

The 60's technology produced some aircraft employing a "fan-in-wing" concept. The most notable being the Ryan XV-5a which employed a jet powered wing lifting fan (see Figure 3) (20:38). But, most research and development concentrated on the "tilt wing" concept (28:2-28) which has found many applications with the implementation of the modern turbo fan.

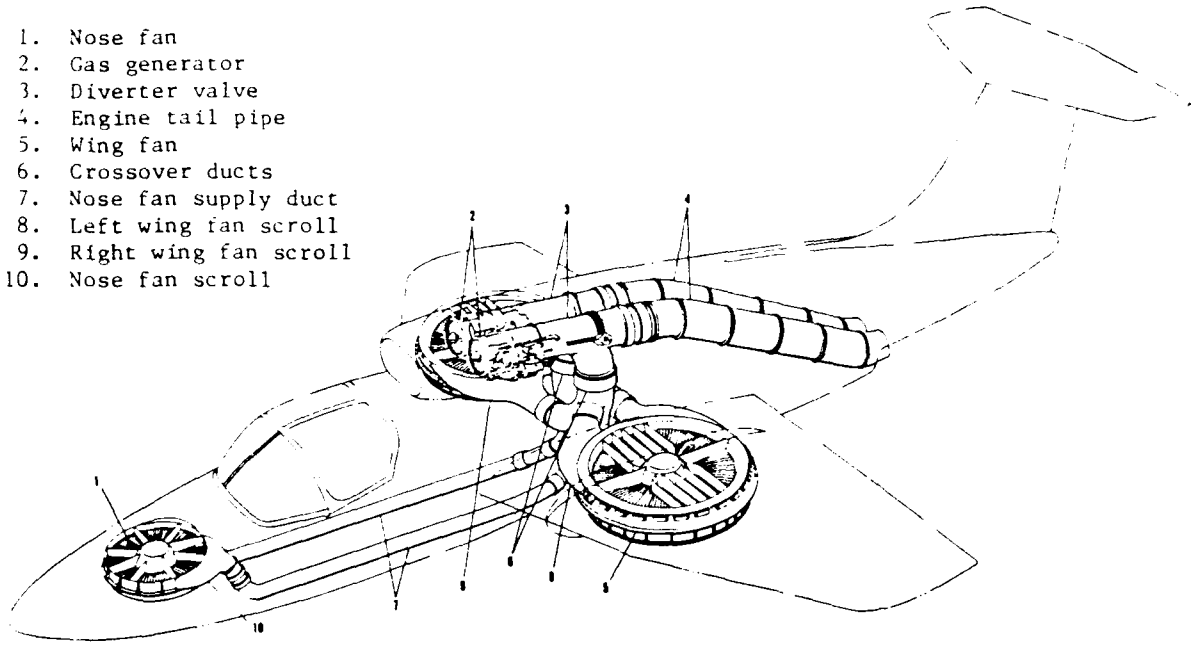


Figure 3. Propulsion System for the Ryan XV-5A (20:38)

In the 70's the fan gave way to jet technology which was lighter and more powerful (see Figure 4) and led to the development of the Harrier. "But in 1980 there is still only one true VSTOL aircraft in the free world that has reached the operational stage, the Hawker Siddeley Harrier (20:3)." Renewed interest in VSTOL and current

development in material science and compressor technology suggest that a second look is warranted for the "fan-in-wing"; with an emphasis on the flow characteristics and possible applications to future STOL technology.

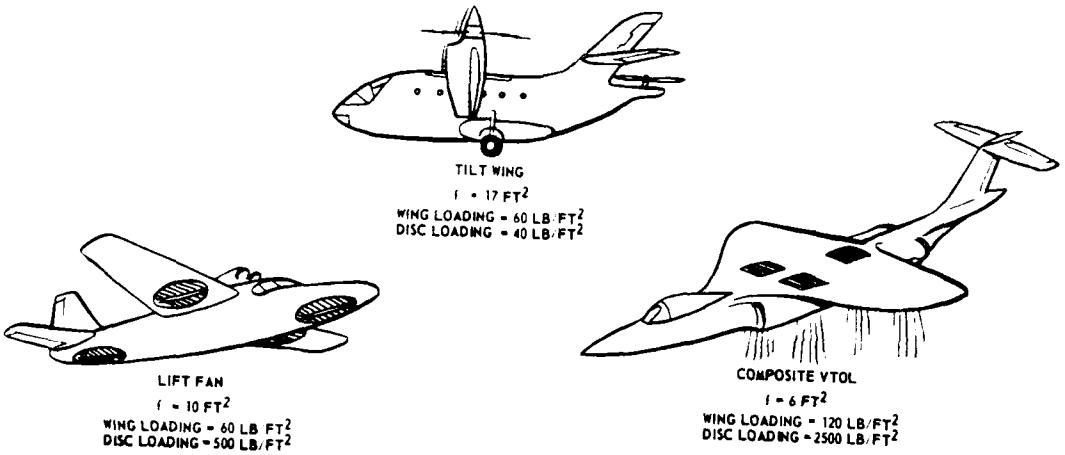


Figure 4. V/STOL Aircraft Configurations (28:14)

The positive effect of suction on boundary layer control and thus aircraft performance has been well documented and is in practical use (31:43-44). The fluid particles in the boundary layer over an airfoil surface have a much lower kinetic energy than do the fluid particles outside the boundary layer, due to the no slip condition and skin friction at the airfoil surface

(31:24-25). As the low energy particles over the airfoil surface mix with the outer boundary they increase the thickness of the low energy boundary layer thus decreasing its stability (see Figure 5) (31:25). And under unfavorable conditions separation can occur causing a substantial increase in drag and a decrease in lift which can lead to early stall behavior at high angles of attack (22:411-413). In boundary layer control the low energy boundary layer is sucked away and the the outer boundary is free to accelerate while a new smaller boundary layer reforms, thus improving lift performance and delaying separation (31:381,382).

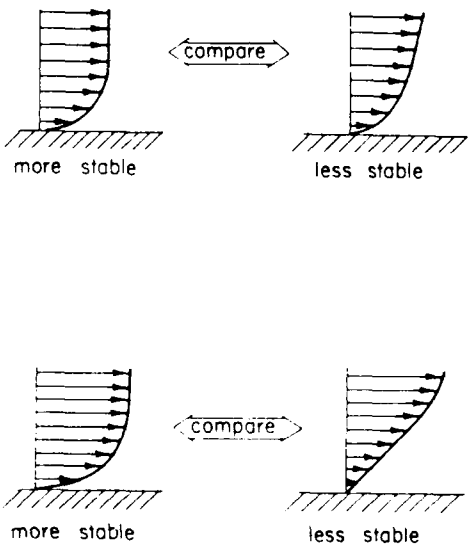


Figure 5. Effect of Boundary Layer Geometry Stability (23:265)

In principle the effects of boundary layer control by suction can be extended to a fan, the results are not as promising. The effect of suction produced by a fan is

not as well understood. The best results for suction are obtained when the slot widths are of the same order as the displacement thickness (see Figure 6) which is much much smaller than a fan width (23:269).

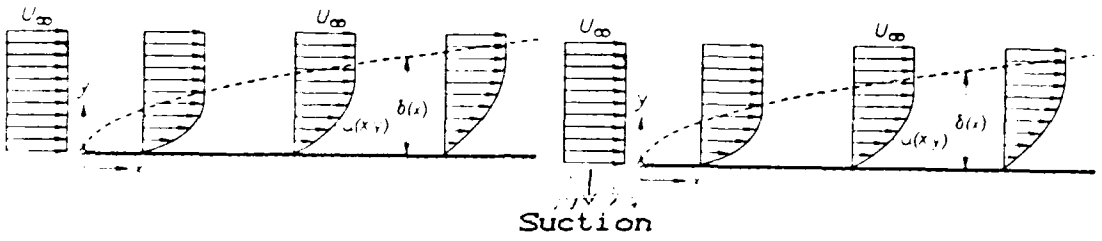


Figure 6. Effect of Suction on the Boundary Layer

The effect of blowing on the other hand is not well understood. Most research on blowing has concentrated on surface blowing which is tangent to the airfoil. This research has led to practical applications such as the "jet flap" and "slotted wing" (see Figure 7) (31:380,381); while the effect of blowing normal to the surface has been carefully ignored.

> "it is mandatory to pay careful attention to the shape of the slit in order to prevent the jet from dissolving into vortices at a short distance behind the exit section (31:380).

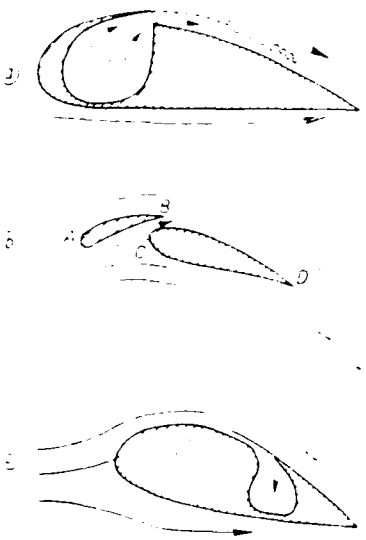


Figure 7. Boundary Layer Control a)Blowing b)Slotted Wing c)Suction (31:381)

At the very least, blowing which is non-tangent to the surface will produce vortices which will cause separation and produce a large separated area behind the blowing exit (see Figure 8) (31:28-29). What is not understood is the effect of blowing on the pressure distribution of the airfoil surface and the effective camber produced by turning the flow. If the blowing configuration is such that vortices are shed then an unsteady solution can be expected.

Some studies were done in the 60's on the optimal configuration for a "fan-in-wing" (23:254). These studies focused on the macroscopic behavior of lift and drag and did not study the flow interaction. These studies do

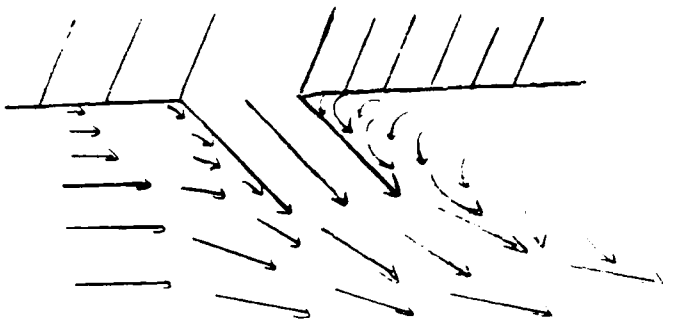


Figure 8. Blowing Non-Tangent to the Surface.

indicate that a horizontal "fan-in-wing" can produce additional lift and reduce drag (see Figure 9).

> The effect of the fan on the wing is somewhat similar to the section of a jet flap. The efflux from the fan is discharged in a direction normal to the plane of the wing and then turns in the downstream direction. (23:254)

These experimental results suggest that the use of inlet vanes to guide the incoming flow is detrimental to both lift and drag. However, the use of exit vanes or vectoring on the ejecting flow can reduce drag without significantly affecting lift or pitching moments. These results also indicate that ejection angles of 20 to 30 degrees with respect to the cord produce the best performance (23:257).

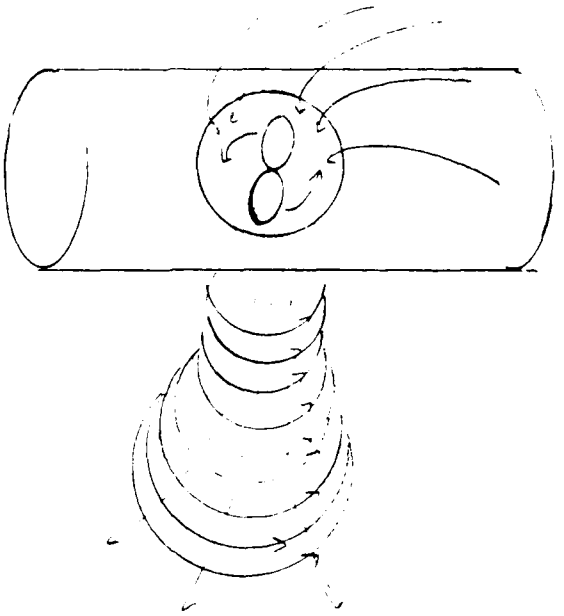


Figure 9. Three-Dimensional Effects of a Fan

As long as the concept has been around, the performance and flow field characteristics of the fan are still not well understood. "There is still no acceptable theory for predicting the performance of a fan-in-wing configuration. At best we will probably have to resort to a numerical solution (23:258-259)." With the current "state of the art" in Computational Fluid Dynamics (CFD), it is possible to numerically simulate complex flow field such as a "fan-in-wing".

In a numerical study many more configurations can be tested in less time and at lower cost than in an experimental study (2:45). The problem with numerical studies is that they can only simulate a problem but can not ensure its validity. This is why numerical solutions must be compared with experimental studies to validate their accuracy before small departures from the validated

model configuration and test conditions can be introduced (32). The new solutions can then provide valuable insight for future experimental and numerical studies. This can then become a continuous cycle of validation, insight, test, validation...

The main difficulty in predicting the effect of the fan on the wing is that it is characterized by a three-dimensional flow field. A vortex is produced in the plane of the wing and interacts with the flow normal to the fan (see Figure 9). This problem can be simplified into two dimensions by ignoring the vortex formed by a rotating fan and calculating only the effects of suction and blowing normal to the plane of the airfoil.

This report looks only at the two-dimensional problem, suction and blowing and as such can be described by any number of mechanisms; however, the term "fan" will be retained in describing the model. The focus of this report is on the effect of mass flux on the lift and drag performance of an airfoil section, and the characteristic behavior of the flow. The problem is further simplified by keeping the geometry and flow condition fixed (appendix C).

The model used for this report is based on the NACA 0012 airfoil section because its behavior has been well documented over a wide range of conditions (1:462,463; 24) that will serve as a basis for validating the numerical

solution without blowing. The fan model used assumes no inlet vanes but ejection vanes are used in order to allow smooth, vectored, parallel ejection. The effect of varying suction velocities, ejection angles and angles of attack was studied.

The model conditions were picked for optimum numerical performance. Compressible codes perform better at higher Mach numbers (34) and for this reason M = 0.3 was chosen. The Reynolds number (Re) can also have a dramatic affect on the solution. If Re is too low current turbulence models have difficulty in describing the physics correctly and if it is too high then the strong compressible effects require extremely high numerical resolution. For these reasons Re = 1,000,000 was chosen. This value is well above the laminar flow region (18) and much smaller than the critical Re for stall (22:413) for this airfoil which could create problems at high angles of attack (34).

Boundary conditions on the upper surface were chosen such that the velocity on the surface was normal to the surface with a constant fractional value of the free stream velocity (see appendix C). Because enhancements were being sought, the velocity ejecting from the lower surface was chosen to be constant in magnitude and direction. The ejection angle was measured from the cord assigned a value between 15 and 90 degrees. The

parameters considered for comparison and analysis were the coefficients of lift, C_l, drag, C_d and pressure, C_p.

The Navier-Stokes code used in the analysis (appendix F), was a modified version of a code developed by Miguel Visbal (36) at the Air Force Wright Aeronautical Laboratories (AFWAL) Flight Dynamics Lab (FDL). This code is based on the "full" compressible Navier-Stokes equations for viscous flow and uses the Beam-Warming Implicit Factored Scheme (4). The turbulence model is based on work done by Baldwin and Lomax (8). The "full" NS code was chosen over the approximate NS (ANS) and the parabolized NS (PNS) codes because of the large stream wise variation and the dynamic behavior of the flow field. The suction and blowing alter dramatically both the direction of the flow and the viscous behavior at the surface. And for the first numerical attempt at this problem a more accurate code was needed to insure numerical accuracy. A compressible code was also needed because of the compressibility imposed on the flow by the fan boundary conditions. Changes to the code involved boundary conditions, input/output, turbulence modeling, force and residual calculations and loop structure to reduce memory requirements.

Tests on different grids were performed in order to determine an optimum spacing configuration. The grids were developed using a hyperbolic grid generator (appendix

G) also provided by the Flight Dynamics Laboratory (FDL) The grids were tested using the above code in order to determine the optimum spacing needed to resolve the problem accurately. Tests were done on the code using a symmetric grid representing a NACA 0012 airfoil and comparing results with known solutions for both laminar and turbulent cases. The results from the final unsymmetric grid were then compared for grid effects. Details on the grid analysis are presented in the Results and Discussion section. After the grid was verified the fan modifications were added and results were obtained. There was no available fan data for comparison and as such the results remain to be verified through future experiments and additional numerical simulations.

Computer resources were provided by FDL and included accounts on the Aeronautical Systems Division Cyber Computer and Cray XMP. The Cray was used for running the Code (appendix E) and performing data analysis too large for the Cyber; all other analysis and plots were performed on the Cyber.

II Analysis

Governing Equations

The mathematical equations which describe the behavior of a fluid (gas) in motion are based on the Principles of Conservation of Mass, Momentum, and Energy. These equations are often grouped together and referred to as the Navier-Stokes (NS) Equations though only the momentum equation is credited to Navier and Stokes (41:71). These equations are listed below in vector notation and are valid for unsteady compressible and viscous flows (6:88-96).

Continuity (Conservation of Mass)

$$\frac{\partial \rho}{\partial t} + \bar{\nabla} \cdot (\rho \bar{V}) = 0 \qquad (1)$$

Momentum (Conservation of Momentum)

$$\frac{\partial (\rho \bar{V})}{\partial t} + \bar{\nabla} \cdot (\rho \bar{V} \bar{V}) = \rho \bar{f} + \bar{\nabla} \cdot \underline{\underline{\sigma}} \qquad (2)$$

Energy (Conservation of Energy)

$$\frac{\partial E_t}{\partial t} + (\bar{V} \cdot \bar{\nabla}) E_t = \frac{\partial Q}{\partial t} + \rho \bar{f} \cdot \bar{V} + \bar{\nabla} \cdot (\underline{\underline{\sigma}} \cdot \bar{V}) - \bar{\nabla} \cdot \bar{q} \qquad (3)$$

where the shear stress term $\underline{\underline{\sigma}}$ is related to the pressure, P, and viscous stress tensor, $\underline{\underline{\tau}}$ (16:6)

$$\underline{\underline{\sigma}} = -PI + \underline{\underline{\tau}} \tag{4}$$

The definition of variables used can be found in the list of symbols near the beginning of this report.

Three additional equations are needed to relate P, τ and heat conduction, q. If a perfect fluid/gas is assumed then the equation of state can be used (16:7)

$$P = \rho RT = \rho e(\gamma-1) \tag{5}$$

and if a Newtonian fluid is assumed then Stoke's law of friction can be used (14:12)

$$\tau = \lambda(\bar{\nabla}\cdot\bar{V})I + \mu(\bar{\nabla}\bar{V} + (\bar{\nabla}\bar{V})^T) \tag{6}$$

and the third equation relates the heat flux vector to the temperature gradient by Fourier's law of heat conduction

$$\bar{q} = -k\bar{\nabla}T \quad ; \quad \bar{\nabla}\cdot\bar{q} = -\bar{\nabla}\cdot(k\bar{\nabla}T) \tag{7}$$

where the conductivity k is a function of the temperature. (14:11-12)

Total internal energy, E_t can be related to the specific internal energy, e, and velocity by the following

relationship (15:13)

$$Et = \rho\left(e + \frac{u^2 + v^2}{2}\right) \tag{8}$$

The above equations can be non-dimensionalized (see appendix A) by applying the following definitions (an * represents a dimensional quantity)

$$x = \frac{x^*}{L^*} \qquad y = \frac{y^*}{L^*} \qquad u = \frac{u^*}{U_\infty^*} \qquad v = \frac{v^*}{U_\infty^*}$$

$$P = \frac{P^*}{\gamma M_\infty^2 P_\infty^*} \qquad T = \frac{\gamma P M_\infty^2}{\rho} \qquad \rho = \frac{\rho^*}{\rho_\infty^*} \qquad \mu = \frac{\mu^*}{\mu_\infty^*}$$

$$Et = \frac{Et^*}{\gamma M^2 P_\infty^*} \qquad e = \frac{Et}{\rho} \qquad \varepsilon = \frac{\varepsilon^*}{\mu_\infty^*} \qquad q = \frac{q^*}{q_\infty^*} \tag{9}$$

$$M_\infty = \frac{U_\infty}{\sqrt{\gamma RT}} \qquad Re = \frac{\rho_\infty^* U_\infty^* L^*}{\mu_\infty^*} \qquad t = \frac{t^* U_\infty^*}{L^*} \qquad L^* = 1$$

The non-dimensional NS equations are written below in conservation-law form for a 2-D cartesian coordinate system.

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \tag{10}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + P - \tau_{xx})}{\partial x} + \frac{\partial(\rho uv - \tau_{xy})}{\partial y} = \rho \bar{f}_x \qquad (11)$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho v^2 + P - \tau_{yy})}{\partial y} + \frac{\partial(\rho uv - \tau_{yx})}{\partial x} = \rho \bar{f}_y \qquad (12)$$

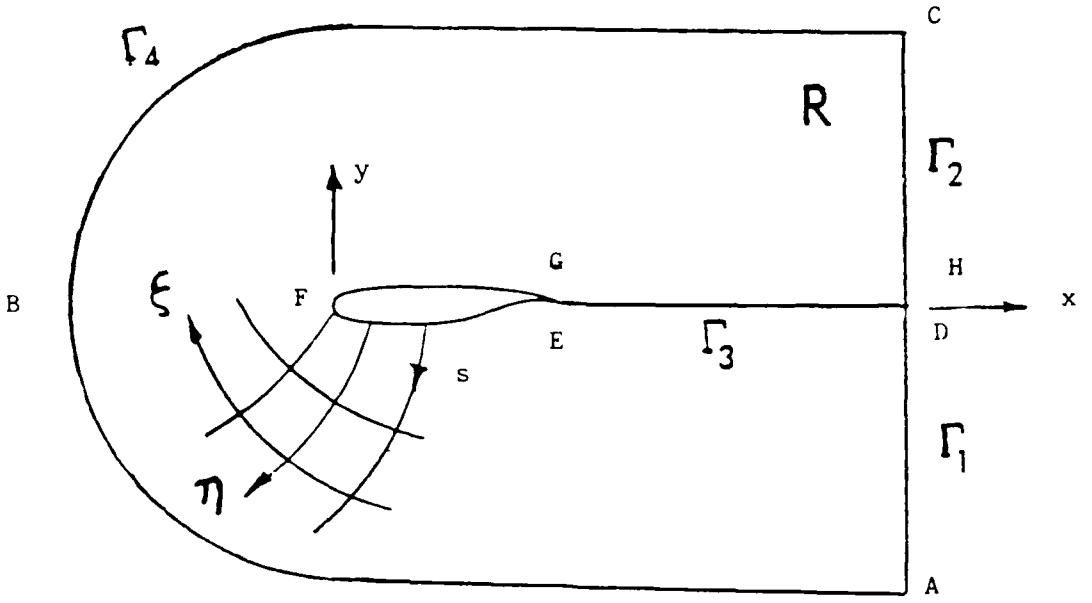
$$\frac{\partial E_t}{\partial t} - \rho(\bar{f}_x u + \bar{f}_y v) + \frac{\partial(E_t u + Pu - u\tau_{xy} - v\tau_{xy} + q_x)}{\partial x} + \frac{\partial(E_t v + Pv - u\tau_{xy} - v\tau_{yy} + q_x)}{\partial y} = 0 \qquad (13)$$

where the internal heat generation, Q, is assumed constant in time (15:32).

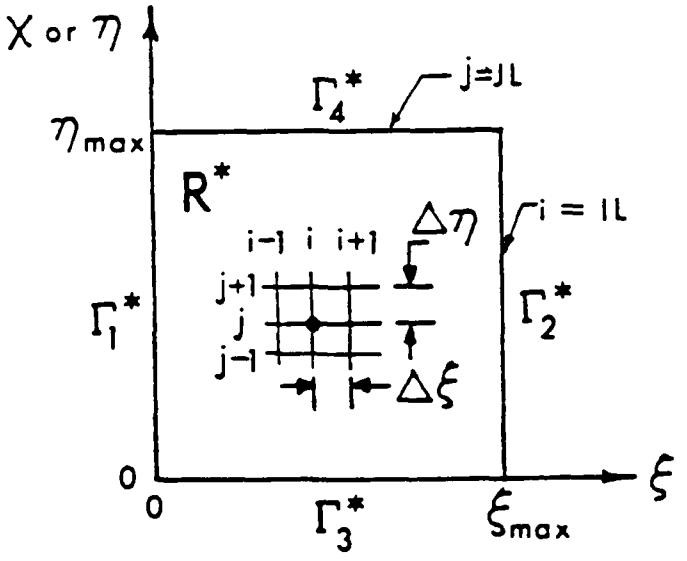
The non-dimensionalized stress terms in index notation are (16:6)

$$\tau_{ij} = -P\delta_{ij} + \frac{\mu}{Re}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \delta_{ij}\lambda \frac{\partial u_k}{\partial x_k} \qquad (14)$$

The above equations can be transformed for varying geometries by applying a coordinate transformation. For any given geometry the coordinates can be related to a square with unit axes of 1 (see Figure 10) by using the chain rule of partial differentiation to relate the



(a) Physical Plane



(b) Transformed Plane

Figure 10 General Transformation from The Physical Domain a to the Computation Domain b (34:47)

physical domain (x,y) to the computational domain (ξ,η).

Where (2:252)

$$\xi = \xi(x,y) \quad , \quad \eta = \eta(x,y) \tag{15}$$

$$\frac{\partial}{\partial x} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \tag{16}$$

$$\frac{\partial}{\partial y} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} \tag{17}$$

The metrics are determined in the following manner (2:252)

$$d\xi = \xi_x dx + \xi_y dy \tag{18}$$

$$d\eta = \eta_x dx + \eta_y dy \tag{19}$$

In matrix form

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{20}$$

and

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} = \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{21}$$

Therefore

$$\begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}^{-1} = J \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} \qquad (22)$$

Where the Jacobian J is given by (21:416)

$$J = \frac{\partial(\xi,\eta)}{\partial(x,y)} = \begin{vmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial y} \\ \frac{\partial\eta}{\partial x} & \frac{\partial\eta}{\partial y} \end{vmatrix} = \xi_x\eta_y - \xi_y\eta_x \qquad (23)$$

$$J = \frac{1}{\frac{\partial(x,y)}{\partial(\xi,\eta)}} = \frac{1}{\begin{vmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial y} \\ \frac{\partial\eta}{\partial x} & \frac{\partial\eta}{\partial y} \end{vmatrix}} = \frac{1}{(x_\xi y_\eta - x_\eta y_\xi)} \qquad (24)$$

and the transformation metrics are given by (35:7)

$$\xi_x = y_\eta J \qquad \xi_y = -x_\eta J \qquad (25)$$

$$\eta_x = -y_\xi J \qquad \eta_y = x_\xi J \qquad (26)$$

Further simplifications can be applied after the coordinate transformation is completed. The chain-rule conservative NS equations without body forces $\bar{f}$ written in terms of general curvilinear coordinates (ξ,η) are as follows (35:5-6)

$$\frac{\partial U}{\partial t} + \xi_x \frac{\partial F}{\partial \xi} + \xi_y \frac{\partial G}{\partial \xi} + \eta_x \frac{\partial F}{\partial \eta} + \eta_y \frac{\partial G}{\partial \eta} = 0 \qquad (27)$$

Where

$$U = [\rho, \rho u, \rho v, \rho e]^T \qquad (28)$$

$$F = \begin{bmatrix} \rho u \\ \rho u^2 - \tau_{xx} \\ \rho v^2 - \tau_{xy} \\ (\rho e + P)u - F_4 \end{bmatrix} \qquad (29)$$

$$G = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 - \tau_{yy} \\ (\rho e + p)v - G_4 \end{bmatrix} \qquad (30)$$

Stokes hypothesis gives (31:60)

$$\lambda = -\frac{2}{3}\mu \qquad (31)$$

Where μ is defined as the molecular dynamic viscosity. It can be extended to include turbulent eddy viscosity, ε (35:6)

$$\lambda_t = -\frac{2}{3}(\mu+\varepsilon) \qquad (32)$$

The thermal conductivity, k, is separated into k for the fluid and k_t for the turbulent properties. Then the viscous stress and heat conduction terms can be written in the following form (35:6)

$$\tau_{xx} = -\tfrac{2}{3}\lambda_t u_x + \lambda_t(u_x + v_y) \tag{33}$$

$$\tau_{xy} = -\frac{\tfrac{3}{2}}{2}\lambda_t(u_x + v_y) \tag{34}$$

$$\tau_{yy} = -\tfrac{2}{3}\lambda_t v_y + \lambda_t(u_x + v_y) \tag{35}$$

$$F_4 = u\tau_{xx} + v\tau_{xy} - q_x \tag{36}$$

$$G_4 = u\tau_{xy} + v\tau_{yy} - q_y \tag{37}$$

$$P = \rho(\gamma-1)\left(e - \frac{1}{2}(u^2 + v^2)\right) \tag{38}$$

$$q_x = -(k + k_t)\,T_x \quad ; \quad q_y = -(k + k_t)\,T_y \tag{39}$$

$$k = c_p\frac{\mu}{Pr} \quad , \quad k_t = c_p\frac{\varepsilon}{Pr_t} \tag{40}$$

where the molecular Prandtl number is defined by (35:6)

$$Pr = \frac{\mu c_p}{k} \cong .72 \tag{41}$$

and the turbulent Prandtl number is defined by (35:7)

$$Pr_t = \frac{\varepsilon c_p}{k} \cong .9 \tag{42}$$

where the specific heat at constant pressure, c_p for a perfect gas is defined by (12)

$$c_p = \frac{\gamma R}{\gamma - 1} \cong 1{,}000 \text{ J/Kg-K} \tag{43}$$

The NS equations can be written in the following strong conservation form by applying the coordinate transformation identities previously discussed (35:7)

$$\frac{\partial \hat{U}}{\partial t} + \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} = 0 \tag{44}$$

where

$$\hat{U} = \frac{U}{J} \tag{45}$$

$$\hat{F} = \frac{(\xi_x F + \xi_y G)}{J} \tag{46}$$

$$\hat{G} = \frac{(\eta_x F + \eta_y G)}{J} \tag{47}$$

Rewriting equation (44) allows for an easier implementation of the implicit algorithm scheme (section III)

$$\frac{\partial \hat{U}}{\partial t} + \frac{\partial E_1}{\partial \xi} + \frac{\partial E_2}{\partial \eta} = \tag{48}$$

$$\frac{\partial V_1(\hat{U}, \hat{U}_\xi)}{\partial \xi} + \frac{\partial V_2(\hat{U}, \hat{U}_\eta)}{\partial \xi} + \frac{\partial W_1(\hat{U}, \hat{U}_\xi)}{\partial \eta} + \frac{\partial W_2(\hat{U}, \hat{U}_\eta)}{\partial \eta}$$

where (35:8)

$$E_1 = \frac{1}{J}\begin{bmatrix} \rho u \\ \rho u u + \xi_x P \\ \rho v u + \xi_y P \\ (P + \rho e) u \end{bmatrix} \tag{49}$$

$$E_2 = \frac{1}{J}\begin{bmatrix} \rho v \\ \rho u v + \eta_x P \\ \rho v v + \eta_y P \\ (P + \rho e) v \end{bmatrix} \tag{50}$$

$$V_1 = \frac{1}{J}\begin{bmatrix} 0 \\ b_1 u_\xi + b_2 v_\xi \\ b_2 u_\xi + b_3 v_\xi \\ b_1 u u_\xi + b_2 (v u_\xi + u v_\xi) + b_3 v v_\xi + b_4 T_\xi \end{bmatrix} \tag{51}$$

$$V_2 = \frac{1}{J}\begin{bmatrix} 0 \\ c_1 u_\eta + c_2 v_\eta \\ c_3 u_\eta + c_4 v_\eta \\ c_1 u u_\eta + c_2 u v_\eta + c_3 v u_\eta + c_4 v v_\eta + c_5 T_\eta \end{bmatrix} \tag{52}$$

$$W_1 = \frac{1}{J}\begin{bmatrix} 0 \\ c_1 u_\xi + c_3 v_\xi \\ c_2 u_\xi + c_4 v_\xi \\ c_1 u u_\xi + c_2 v u_\xi + c_3 u v_\xi + c_4 v v_\xi + c_5 T_\xi \end{bmatrix} \quad (53)$$

$$W_2 = \frac{1}{J}\begin{bmatrix} 0 \\ d_1 u_\eta + d_2 v_\eta \\ d_2 u_\eta + d_3 v_\eta \\ d_1 u u_\eta + d_2 (v u_\eta + u v_\xi) + d_3 v v_\eta + d_4 T_\eta \end{bmatrix} \quad (54)$$

u and v denote the contravarient velocities (35:9)

$$u = \xi_x u + \xi_y v \quad (55)$$

$$u = \eta_x u + \eta_y v \quad (56)$$

The viscous coefficients are (35:4):

$$b_1 = -\frac{3}{2}\lambda_t \left[\frac{4}{3}\xi_x^2 + \xi_y^2\right] \quad (57)$$

$$b_2 = -\frac{1}{2}\lambda_t \xi_x \xi_y \quad (58)$$

$$b_3 = -\frac{3}{2}\lambda_t \left[\frac{4}{3}\xi_y^2 + \xi_x^2\right] \quad (59)$$

$$b_4 = c_p \left(\frac{\mu}{Pr} + \frac{\varepsilon}{Pr_t}\right)\left[\xi_x^2 + \xi_y^2\right] \quad (60)$$

$$c_1 = \frac{Re}{2} \lambda_t \left[\frac{4}{3} \xi_x \eta_x + \xi_y \eta_y \right] \quad (61)$$

$$c_2 = \frac{Re}{2} \lambda_t \left[\frac{2}{3} \xi_x \eta_y - \xi_y \eta_x \right] \quad (62)$$

$$c_3 = -\frac{Re}{2} \lambda_t \left[\xi_x \eta_y + 2/3 \xi_y \eta_x \right] \quad (63)$$

$$c_4 = \frac{Re}{2} \lambda_t \left[\xi_x \eta_x + 2/3 \xi_y \eta_y \right] \quad (64)$$

$$c_5 = -c_p \left[\frac{\mu}{Pr} + \frac{\varepsilon}{Pr_t} \right] \left[\xi_x \eta_x + \xi_y \eta_y \right] \quad (65)$$

$$d_1 = -\frac{Re}{2} \lambda_t \left[\frac{4}{3} \eta_x^2 + \eta_y^2 \right] \quad (66)$$

$$d_2 = -\frac{1}{2} \lambda_t \eta_x \eta_y \quad (67)$$

$$d_3 = -\frac{Re}{2} \lambda_t \left[\frac{4}{3} \eta_y^2 + \eta_x^2 \right] \quad (68)$$

$$d_4 = c_p \left[\frac{\mu}{Pr} + \frac{\varepsilon}{Pr_t} \right] \left[\eta_x^2 + \eta_y^2 \right] \quad (69)$$

The above NS equations written in terms of general curvilinear coordinates are valid for both subsonic and supersonic, unsteady, compressible, viscous flows. The governing equations form a hybrid hyperbolic-parabolic system. Their parabolic nature comes from the second-order derivatives in the momentum and energy equations which provide dissipative effects. The continuity equation contains only first-order terms and is

hyperbolic; without the unsteady density term it would be elliptic. These relationships dictate the manner in which the initial and boundary conditions can be applied. (25:312)

Additional approximations must be made in order to solve a finite difference problem (discussed in section III). The main difficulties are in the area of turbulent flow, where at best, turbulent models are only approximate and rely heavily on empirical data.

III Numerical Solution of the NS Equations

The numerical method for solving the NS equations as previously presented is discussed in this chapter. The finite-difference scheme presented is the result of work by R. F. Warming and Richard M. Beam and is known as the Beam-Warming approximate factorization algorithm (4). The original two-dimensional Navier-Stokes code was written and verified by Dr Miguel Visbal of the Flight Dynamics Laboratory (36). Modifications made by the Author and discussed in this chapter concern the airfoil surface boundary conditions for mass exchange (fan). Finally, critical details such as convergence criteria and time steps will be presented.

Implicit Navier-Stokes Code

The implicit code solves the strong conservative formulation of the NS equations (1 - 3) using the Beam-Warming approximate factorization algorithm (4:85). The scheme is written in "delta" form with a first-order Euler time-differencing written as follows (35:19):

$$\left\{ I + \Delta t \left[\frac{\partial A^n}{\partial \xi} - \frac{\partial^2 M^n}{\partial \xi^2} \right] \right\} \left\{ I + \Delta t \left[\frac{\partial B^n}{\partial \eta} - \frac{\partial^2 N^n}{\partial \eta^2} \right] \right\} \Delta \hat{U}^n =$$

$$- \Delta t \left[\frac{\partial}{\partial \xi} (E_1 - V_1 - V_2)^n + \frac{\partial}{\partial \eta} (E_2 - W_1 - W_2)^n \right] \quad (70)$$

$$\hat{U}^{n+1} = \hat{U}^n + \Delta \hat{U} \quad (71)$$

where n represents the temporal index (i.e. $\hat{U}^n = \hat{U}(n\Delta t)$) and A, B, M and N represent the Jacobian matrices. See appendix B for details.

$$A = \frac{\partial E_1}{\partial \hat{U}} \quad , \quad B = \frac{\partial E_2}{\partial \hat{U}} \quad (72)$$

$$M = \frac{\partial V_1}{\partial \hat{U}_\xi} \quad , \quad N = \frac{\partial W_2}{\partial \hat{U}_\eta} \quad (73)$$

After applying second order differencing the space derivatives become (35:19-20):

$$\left\{ I + \Delta t \left[\mu_\xi A_{i,j} - \delta_\xi^2 M_{i,j} \right] \right\} \Delta \hat{U}^*_{i,j} = \quad (74)$$

$$- \Delta t \left[\mu_\eta (E_1 - V_2)_{i,j} + \mu_\eta (E_2 - W_1)_{i,j} \right]$$

$$\left\{ I + \Delta t \left[\mu_\eta B_{i,j} - \delta_\eta^2 N_{i,j} \right] \right\} \Delta\hat{U}_{i,j} = \Delta\hat{U}^*_{i,j} \qquad (75)$$

$$\xi_i = (i - 1)\Delta\xi \quad ; \quad 1 \le i \le IL \qquad (76)$$

$$\eta_j = (j - 1)\Delta\eta \quad ; \quad 1 \le j \le JL \qquad (77)$$

and the finite-difference operators are (35:20):

$$\delta_\xi f_{i,j} = (f_{i+1/2,j} - f_{i-1/2,j}) / \Delta\xi \qquad (78)$$

$$\delta_\eta f_{i,j} = (f_{i,j+1/2} - f_{i,j-1/2}) / \Delta\eta \qquad (79)$$

$$\mu_\xi f_{i,j} = (f_{i+1,j} - f_{i-1,j}) / 2\Delta\xi \qquad (80)$$

$$\mu_\eta f_{i,j} = (f_{i,j+1} - f_{i,j-1}) / 2\Delta\eta \qquad (81)$$

The transformation derivatives (x_ξ, x_η, y_ξ, y_η) are computed from the body-fitted grid by using one-sided approximations at the boundaries and second-order central-differencing approximations at the interior points. A solution is arrived at by first solving equation (74) during a sweep along each η line ($2 \le i \le IL-1$), and then solving equation (75) during a sweep along each ξ line ($2 \le j \le JL-1$). The boundaries (i=1,IL ; j=1,JL) are then solved explicitly.

Artificial damping is needed to maintain the stability of the algorithm and to speed up convergence of the solution. Instabilities can occur in compressible flow computations at high Reynolds numbers. These instabilities can be caused by nonlinear effects, rapid changes of flow direction in separated regions, large pressure gradients and the presence of walls and outer boundaries in the computational domain. They can create finite oscillations which can slow down convergence considerably. (25:322)

These instabilities and oscillations are removed by adding an explicit fourth-order damping term D_e to right-hand-side of equation (74) and inserting two second order damping terms Di_ξ and Di_η within the respective implicit operators in equation (75) (35:20):

$$D_e = -\omega_e \Delta t J^{-1}_{i,j} \left(\delta^4_\xi + \delta^4_\eta\right) U^n \tag{82}$$

$$Di_\xi = -\omega_i \Delta t J^{-1}_{i,j} \delta^2_\xi J_{i,j} I \tag{83}$$

$$Di_\eta = -\omega_i \Delta t J^{-1}_{i,j} \delta^2_\eta J_{i,j} I \tag{84}$$

Where ω_i is of order one and $\omega_i \geq 2\omega_e$.

For certain well posed problems convergence to steady-state can be accelerated by employing local time $\Delta t_{i,j}$ which is dependent upon the flow conditions and metrics at each individual point in the grid (35:22):

$$\Delta t_{i,j} = (CFL)\,\Delta t\max_{i,j} \tag{85}$$

Where $\Delta t\max_{i,j}$ is calculated using the modified Courant-Friedrichs-Levy stability criteria (35:22).

$$\Delta t\max_{i,j} = \left[\frac{|u|}{\Delta\xi} + \frac{|v|}{\Delta\eta} + a\sqrt{\frac{1}{\Delta S_\xi^2} + \frac{1}{\Delta S_\eta^2}} + \frac{2\gamma}{\rho \Delta S_\eta^2}\left(\frac{\mu}{Pr} + \frac{\varepsilon}{Prt} \right) \right]^{-1} \tag{86}$$

$$\Delta S_\xi = \left(x_\xi^2 + y_\xi^2 \right)^{1/2} \Delta\xi \tag{87}$$

$$\Delta S_\eta = \left(x_\eta^2 + y_\eta^2 \right)^{1/2} \Delta\eta \tag{88}$$

$$u = \xi_x u + \xi_y v \tag{89}$$

$$v = \eta_x u + \eta_y v \tag{90}$$

The Courant Number (CFL) for an explicit algorithm must be less than one for stability. However, for the implicit scheme CFL can be much higher and is typically set equal to 20.

Grid Generation

As previously mentioned the grids used for this report were numerically generated from a " hyperbolic grid generator" code (appendix G). This code is based on an algorithm developed by Steger and Chaussee (33) which solves a system of hyperbolic partial differential equations. The advantage of using a hyperbolic grid generator over an elliptic one is that only the inner boundary needs to be specified (2:530), in this case the airfoil surface. Also, systems of hyperbolic equations are easier to solve than elliptic equations and the above algorithm is unconditionally stable.

The detailed theory is beyond the scope of this thesis. An excellent reference for theory used in this code is AFWAL-TM-84-191 (19), prepared by Kinsey and Barth. The purpose of using a grid generator is to produce a body conformal grid in the physical domain that can be related to a uniform rectangular computational domain with grid spacing of unity. For best results the grid coordinates should be orthogonal (2-193).

The particular grid topology used in this study was the C-grid. C-grids require less points than many other type grids (19:11) and therefore require less computer resources to resolve a typical problem. C-grids also allow uniform boundary conditions which are easy to implement.

However, for unsteady flows they can present a problem in the wake region of the airfoil. Where along the branch cut all variables are obtained by numerical averaging (see appendix C).

Convergence Criteria

There are three types of converged solutions in numerical simulations, steady-state, periodic and unsteady. Even though no solution is perfectly steady, there is always some oscillation, the unsteady behavior may be so small as not to affect the solution visibly. When this is the case the solution can be said to have reached a steady-state (27). The steady-state solution is the easiest to resolve. It is characterized by the uniform convergence of some criteria after an initial transition region. In most cases local time can be used with a CFL = 20 or more and convergence can be expected in less than 2,000 iterations. Even faster convergence can be achieved if restart solutions are used where the restart solutions represent a variation in parameters such as Reynolds number or transition point.

The criteria for convergence for this report were the coefficients of lift (C_l) and drag (C_d). When the variations in C_d were within 1-2 drag counts (1 drag count = 0.0001) and the amplitude in C_l was less than 0.1% then

the solution was considered converged to a steady state (35:25).

The periodic solutions can be characterized by an initial unsteady behavior in a transition region followed by periodic oscillations in C_l and C_d. Local time can not be used to resolve such a problem, and a fixed constant time step must be used. The oscillations in C_l and C_d may have more than one harmonic mode on an overriding wave (27). If one harmonic dominates or if a superposition period can be found then convergence is determined when the change in the maximum value of C_l and C_d over one characteristic time unit is less than one percent (7). A characteristic time unit is non-dimensionless and represents the time it takes a particle to travel the length of an airfoil.

$$t = t_{min} N \tag{91}$$

Where t_{min} is the smallest constant time step in the computation and N is the number of iterations. Care should be taken to insure that at least four periods are in the analysis, to take into account any unsteady trends (7). Periodic solutions can result from disturbances in the flow field, such as vortex shedding, large separated regions and stall characteristics at high angles of attack.

A truly unsteady solution is characterized by oscillations which are not periodic. Such a solution can not be resolved and often results when the model does not represent true physical behavior such as laminar separation at a turbulent Reynolds number or when numerical instability predominates the solution. This type of behavior can not be resolved except as a function of time and depending on the problem might not represent physical behavior.

IV Results and Discussion

In this chapter the numerical results obtained from the Navier-Stokes code previously mentioned are presented. The verification of both the code and grid are discussed and the final results for a "fan-in-wing" are presented. The results are presented through the comparison of lift and drag coefficients, streamline and vorticity contour plots and surface pressure and skin friction profiles. Data reduction was performed with two codes written by the author and included in appendices J and K. This investigation concentrated on the effect of mass flux on varying suction rates Cv, ejection angles δ and angle of attack α.

Code and Grid Verification

Before the fan problem could be resolved it was necessary to determine a compatible grid configuration and to insure that the NS code was predicting the flow field accurately. This was done by testing the various grids listed in Table 1 under known, verifiable conditions.

grid	wall spacing	η_{pt}	ξ_{pt}	outer boundary	symmetry
1	.001	199	40	5	yes
2	.0005	137	44	5	yes
3	.0001	199	50	10	yes
4	.0001	299	100	20	no
5	.0001	299	100	20	no

Table 1. NACA 0012 Grid Configurations

Grids 1, 3, 4 and 5 were produced using the hyperbolic generator and grid 2 was produced from an elliptical grid generator. Grid 4 and 5 are unsymmetrical because points on the body were clustered over the suction and ejection surfaces which, is evident in Figure 11. The number of points in the η direction, η_{pt}, were increased on grid 4 to allow better resolution and a greater cord distance to the outer boundary. The resolution on the airfoil surface was improved by increasing the number of points in the ξ direction, ξ_{pt}. At the leading edge the spacing is 0.003, at the trailing edge it is 0.005, and over the injection and suction surfaces it is 0.005. The wall spacing on 4 was refined to 0.0001 in order to resolve the boundary layer better. This resolution can be seen on the blown-up portions of the grid in Figure 12. Grid 5 was produced from grid 4 by using two codes (appendix I) to redistribute the points in the η direction in order to force a maximum spacing at the outer boundary of less than one cord length.

All of the grids in Table 1 were tested against

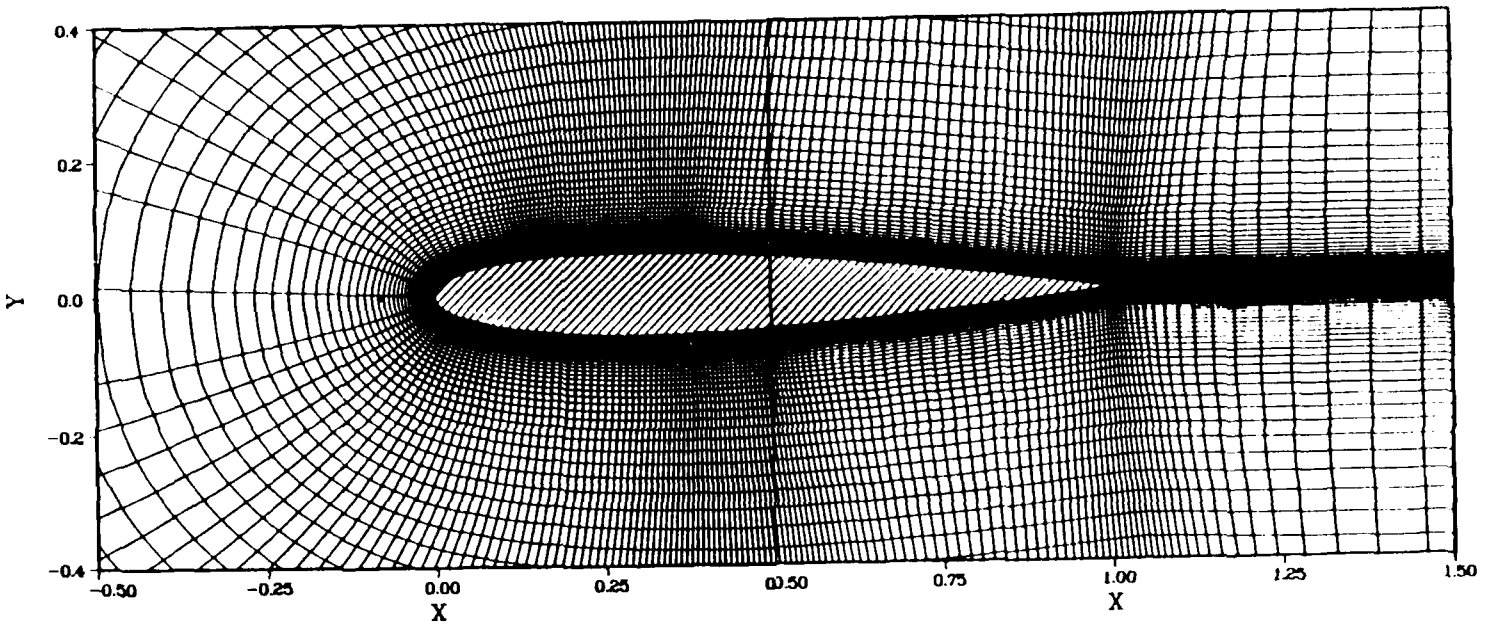


Figure 11. Final Grid Configuration for a NACA 0012 Airfoil

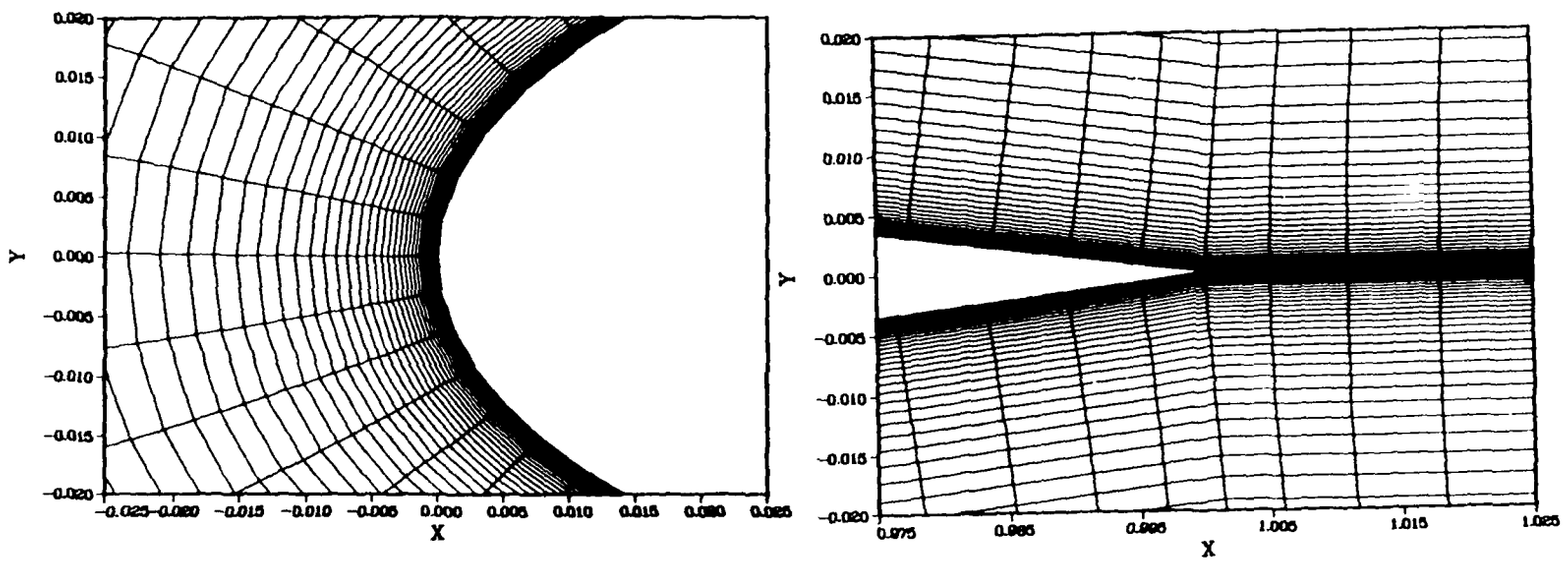


Figure 12. Final Grid Configuration, Close-up

separation point results from a PNS code (13:5). The results are compared in Table 2. The conditions of the test were laminar flow at Re = 12,500 and M = 0.3. From Table 2 it is obvious that a wall spacing of 0.001 is too coarse. The other grids predict reasonably well. Grid 2 has similar spacing to the grid used for the PNS analysis and for this reason gave the best results. Grid 4 and 5, with smaller wall spacing, predicted separation early. This may be due to better numerical resolution of the boundary layer.

grid	separation
1	.000
2	.817
3	.784
4	.787
5	.787
ref(13:5)	.817

Table 2. Separation Point Comparison

C_l and C_d were then compared at an extreme case for grid 4 and 5 (Table 3). The numerical data was computed using a interactive boundary layer code (10:8). Both the experimental data and the numerical solution fixed the transition at 5 percent cord. For this case Re = 6,000,000 and M = 0.3. These conditions are near stall ($\cong 14^\circ$), presented a difficult case and were a good test for the code's turbulence model.

grid	Cl	Cd
4	1.08	.0129
5	1.07	.0124
ref (10:8)	.95	.0092
ref (1:462)	1.11	.0111

Table 3. Lift and Drag Comparison with Experimental Data, $\alpha = 10^{\circ}$, M = .3, Re = 6,000,000

The C_l results in Table 3 are in good agreement with the experimental data, but the C_d results are higher than expected. Considering the extreme Re and the high angle of attack, this comparison still indicates that the code predicts a reasonable solution. The next case compares surface C_p form grid 5 with experimental data, Figure 13. The difference in Re has only a small effect, C_l and C_d are compared in Table 4 for transition effects. The effect of transition is more pronounced both in the C_p and the C_l, C_d comparisons. Though not in perfect agreement, the NS solution predicts slightly higher values for C_p, the agreement is still reasonable.

Re x10^6	transition %c lower surface	Transition %c upper surface	Cl	Cd
1.00	5	40	.455	.0128
1.00	5	5	.465	.0108
1.86	5	5	.460	.0098

Table 4. Cl and Cd vs Re and Transition Point, $\alpha^{\circ} = 4.04$

A final comparison was made for c_l and c_d versus inviscid theory and experimental data for grid 5, Table 5, and transition points. The transition point on the

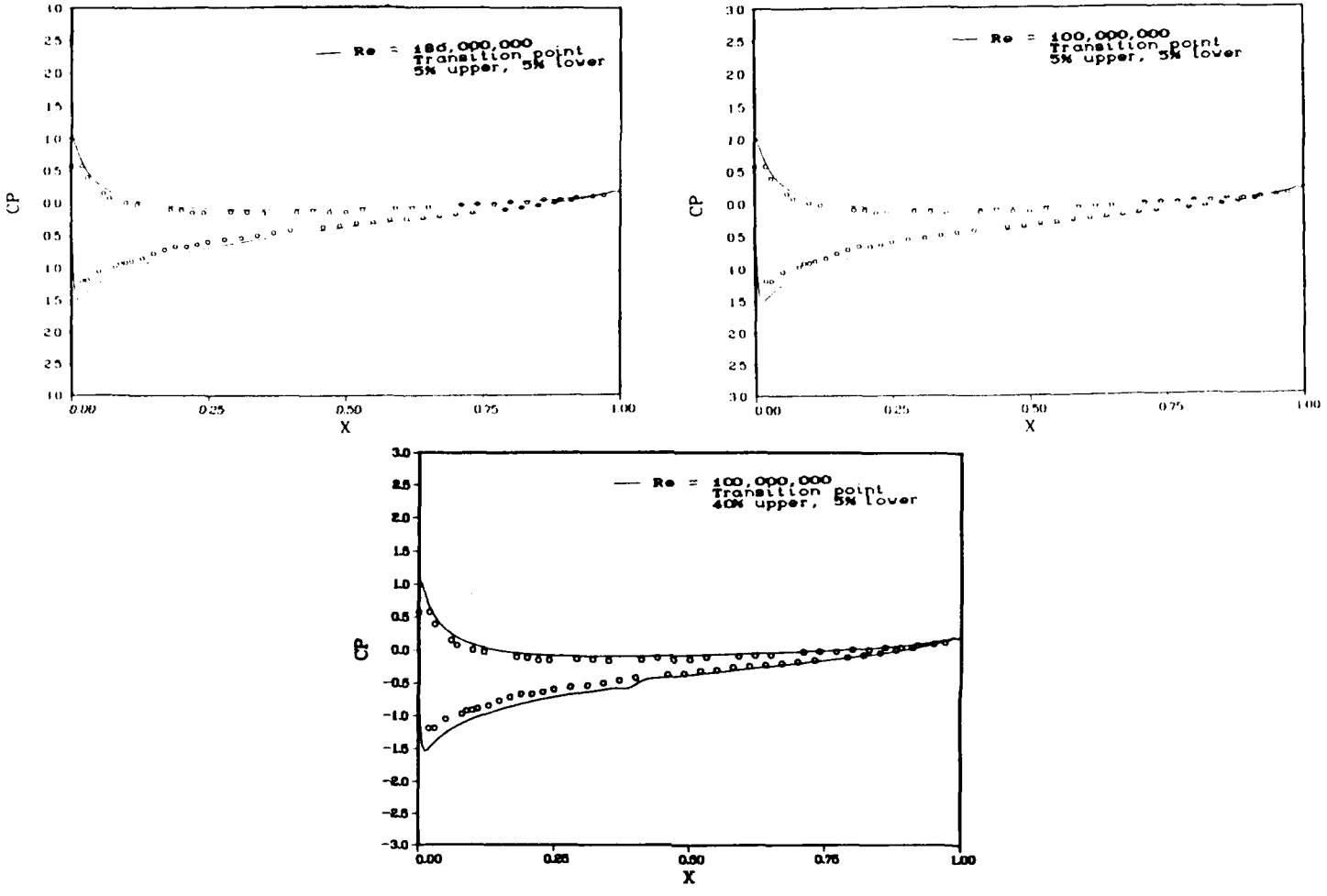


Figure 13. Comparisons of Cp with Experimental Data (25: A1-10)
(Re=1,860,000, M=.3, α=4.04°)

upper surface was fixed at 40 percent of the cord as measured from the leading edge, which is 5 grid points behind the suction region of the fan.

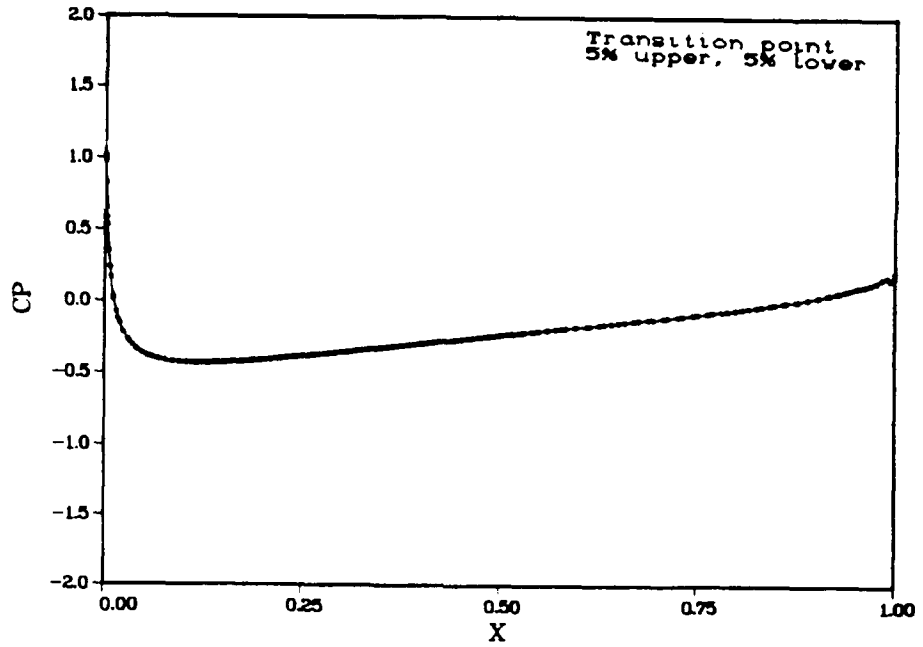
Transition %c	surface lower	surface upper	$\alpha=0^\circ$ Cl	$\alpha=2^\circ$ Cl	$\alpha=4^\circ$ Cl
	5	40	0.023	.246	.478
	5	5	0.000	.229	.454
exp (1:460)			0.0	.22	.44
inviscid theory					
flat plate $2\pi\alpha$		(31:93)	0.0	.219	.439
Joukovski t/c = .12		(27:95)	0.0	.238	.477

Table 5. Cl and Cd Comparison with Experimental and Theoretical Data
(exp Re = 3,000,000, num Re = 1,000,000)

The C_p and C_f surface comparisons are presented in Figures 14 and 15. From this data it is also apparent that the transition point has an effect on the solution. The C_p curves in Figure 14 indicate a pressure difference around the leading edge. This pressure difference is responsible for the lift created by the unsymmetrical transition points. The C_f plots in Figure 15 also indicate a small change due to transition. These differences are small when compared with angle of attack which will be presented later.

Suction and Ejection Separately

The first step taken in determining the effect of a



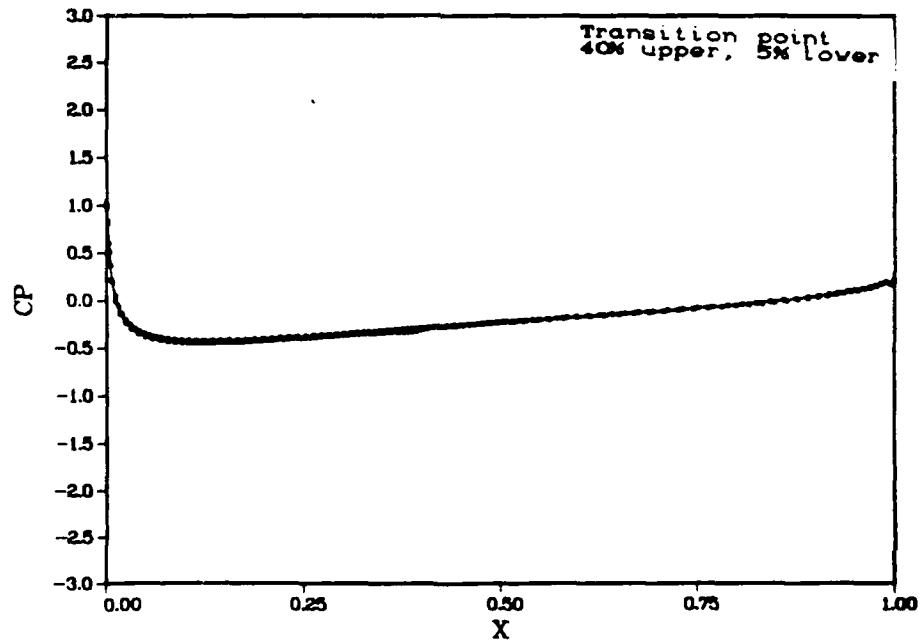


Figure 14. Comparisons of Cp for Transition Point Effects (Re=1,000,000, M=.3, $\alpha=0^{\circ}$)

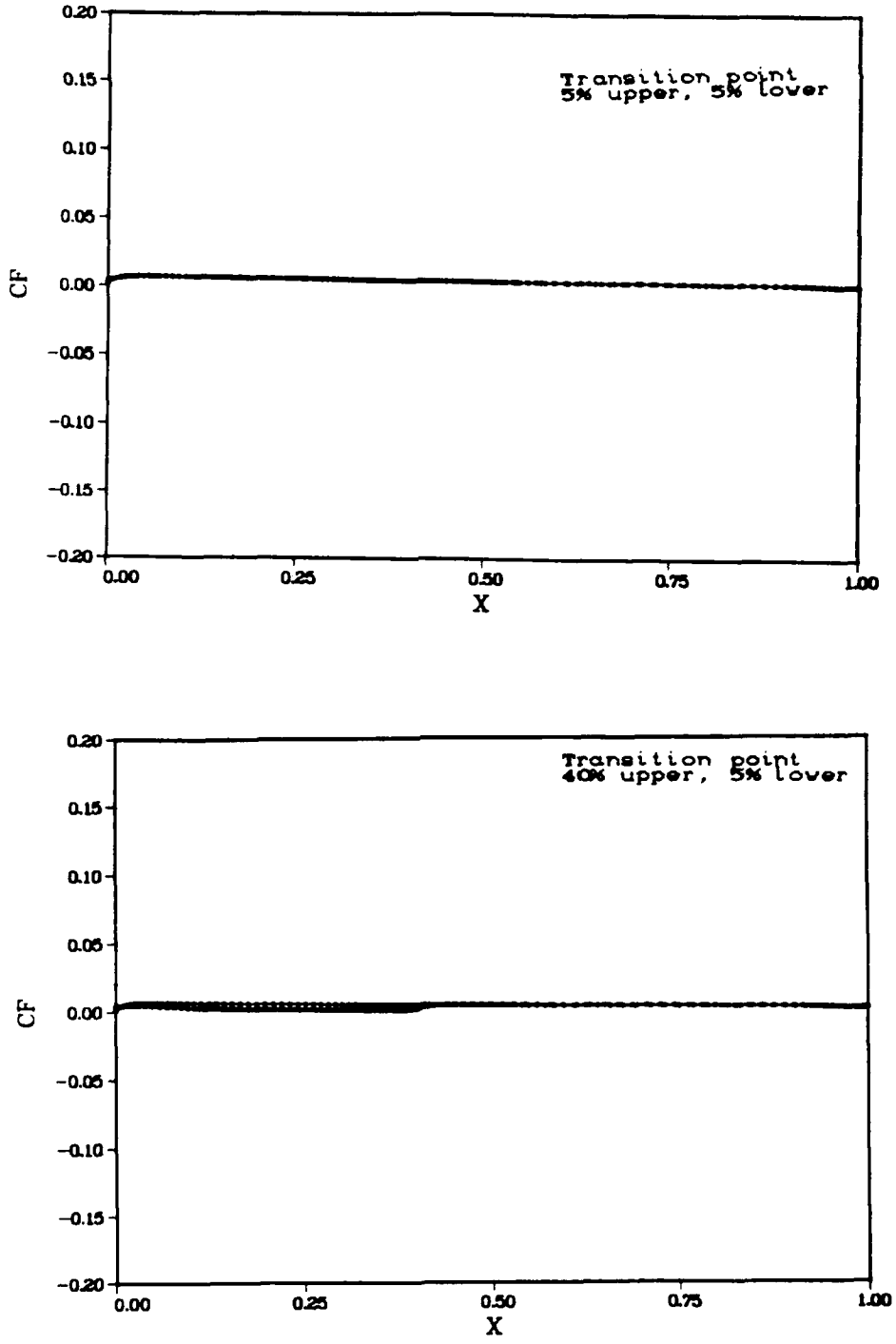


Figure 15. Comparisons of Cf for Transition Point Effects (R_e=1,000,000, M=.3, α=0°)

fan was a comparison of the components, suction and ejection separately see Table 6. For this analysis ejection was set at $\delta = 90^{\circ}$ and suction velocity Cv = 0.05. As with the solutions above, the solution for suction reached a steady-state condition quickly; however, blowing converged to a periodic solution which is discussed in some detail in the following section on convergence. Suction has the effect of increasing lift only marginally while ejection has a surprisingly high effect on lift. The differences can be clearly seen by comparing the C_p curves in Figure 16. The drag effects are reversed. Normal ejection has almost no effect on drag while suction produced an astonishingly high skin friction profile, Figure 17, which results in a large viscous drag component.

case	Cl_{total}	Cl_{press}	$Cl_{viscous}$	Cl_{mass}
no fan	.01907	.01921	-.00014	.00000
suction	.09968	.08477	.00018	-.00126
ejection	.21473	.21264	-.00018	.00226
fan	.22849	.22725	.00010	.00107

case	Cd_{total}	Cd_{press}	$Cd_{viscous}$	Cd_{mass}
no fan	.00954	.00164	.00790	.0000
suction	.01484	-.00926	.01806	.00004
ejection	.02694	.02151	.00543	.00000
fan	.03971	.01800	.01567	.00004

Table 6. Cl and Cd Comparison for Suction and Blowing Only ($Cv = .05$, $\delta^{\circ} = 90^{\circ}$)

The skin friction profile of suction indicates the effect is related to the immediate suction surface.

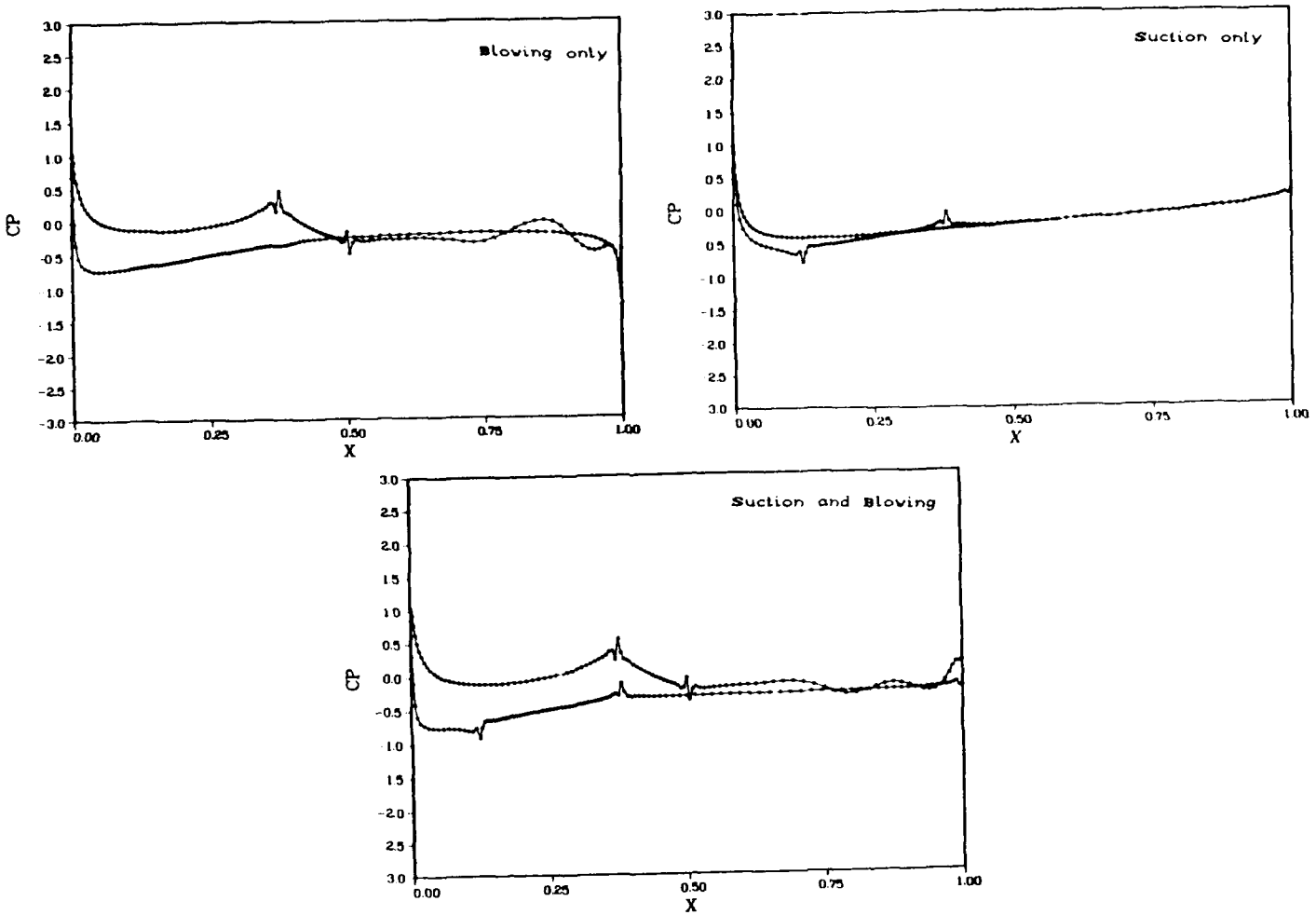


Figure 16. Comparisons of Cp for Suction and Blowing effects
(R_e=1,000,000, M=.3, α=0°)

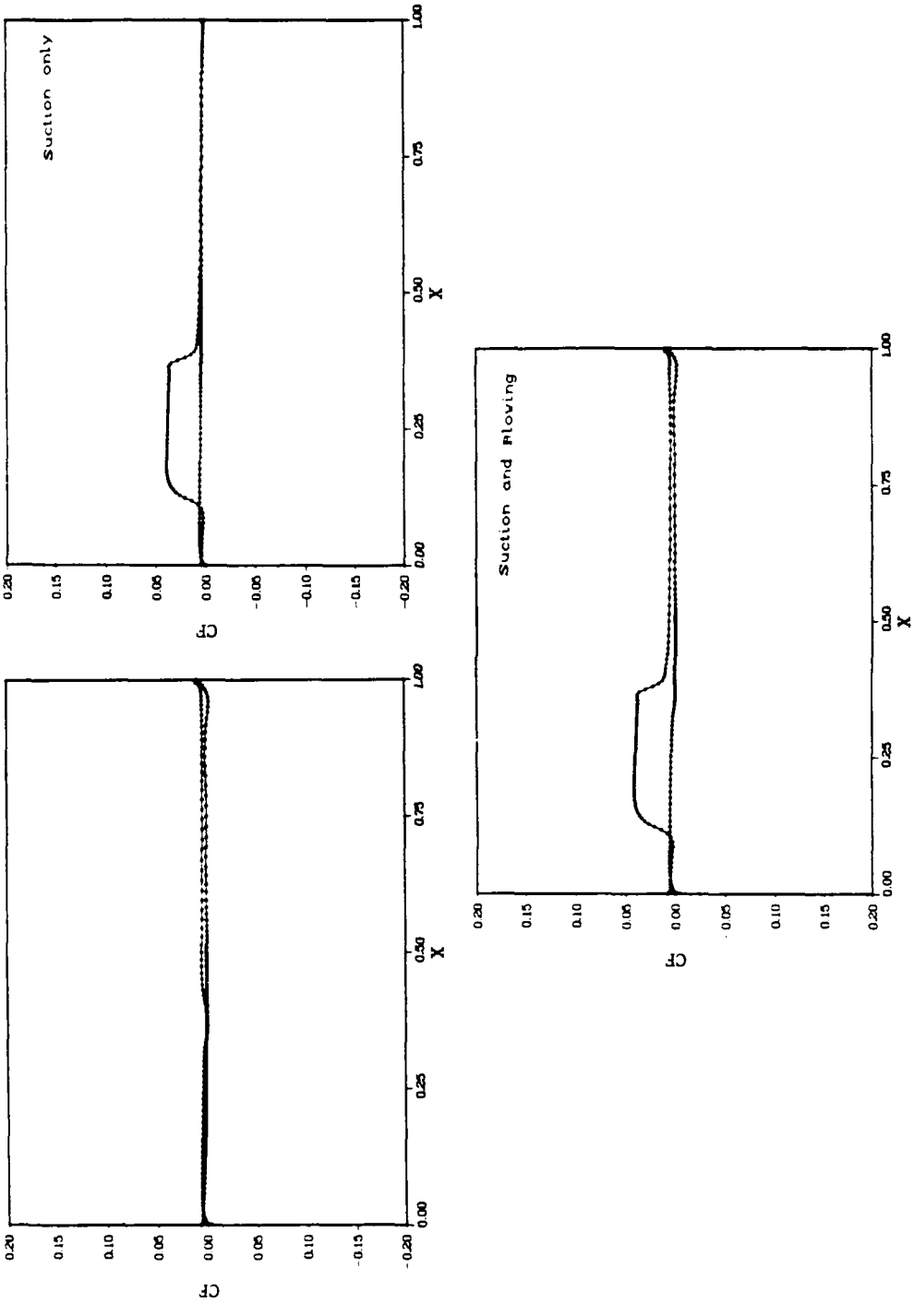


Figure 17. Comparisons of Cf for Suction and Blowing effects (Re=1,000,000, M=.3, α=0°, δ=90°, Cv=.05)

Analysis of the velocities at one point above the surface indicate that suction creates a large velocity gradient above the surface, see Table 7 and Figure 18. This result can be directly linked to the boundary conditions by considering the continuity equation at the suction surface

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \qquad (10)$$

surface values $\eta = 1$ x	y	ρ	u	v
.1110	.0483	.9938	.0000	.0000
.1176	.0492	.9870	.0000	.0000
.1234	.0500	.9860	.0065	-.0496
.1284	.0507	.9504	.0062	-.4961
.1334	.0513	.9619	.0059	-.4965

one point above surface $\eta = 2$ x	y	ρ	u	v
.1110	.0484	.9944	.2510	.0371
.1176	.0493	.9884	.4547	.0587
.1234	.0501	.9879	.7899	.0594
.1284	.0506	.9526	1.0575	.0796
.1334	.0514	.9698	1.2623	.0999

Table 7. Velocity and Density Comparison at the Suction Surface (Suction begins at x=.12345)

As v increases suddenly at the suction surface so must u along the surface and the growth is very fast as indicated by Table 7. The velocity of u before the suction indicates that it is in the boundary layer and at the point above suction the boundary layer is removed see Figure 18 and $u > 1$. The velocity profile quickly reduces to the free stream value, as can be seen in Figure 19. The effect of suction is confined very close to the body with

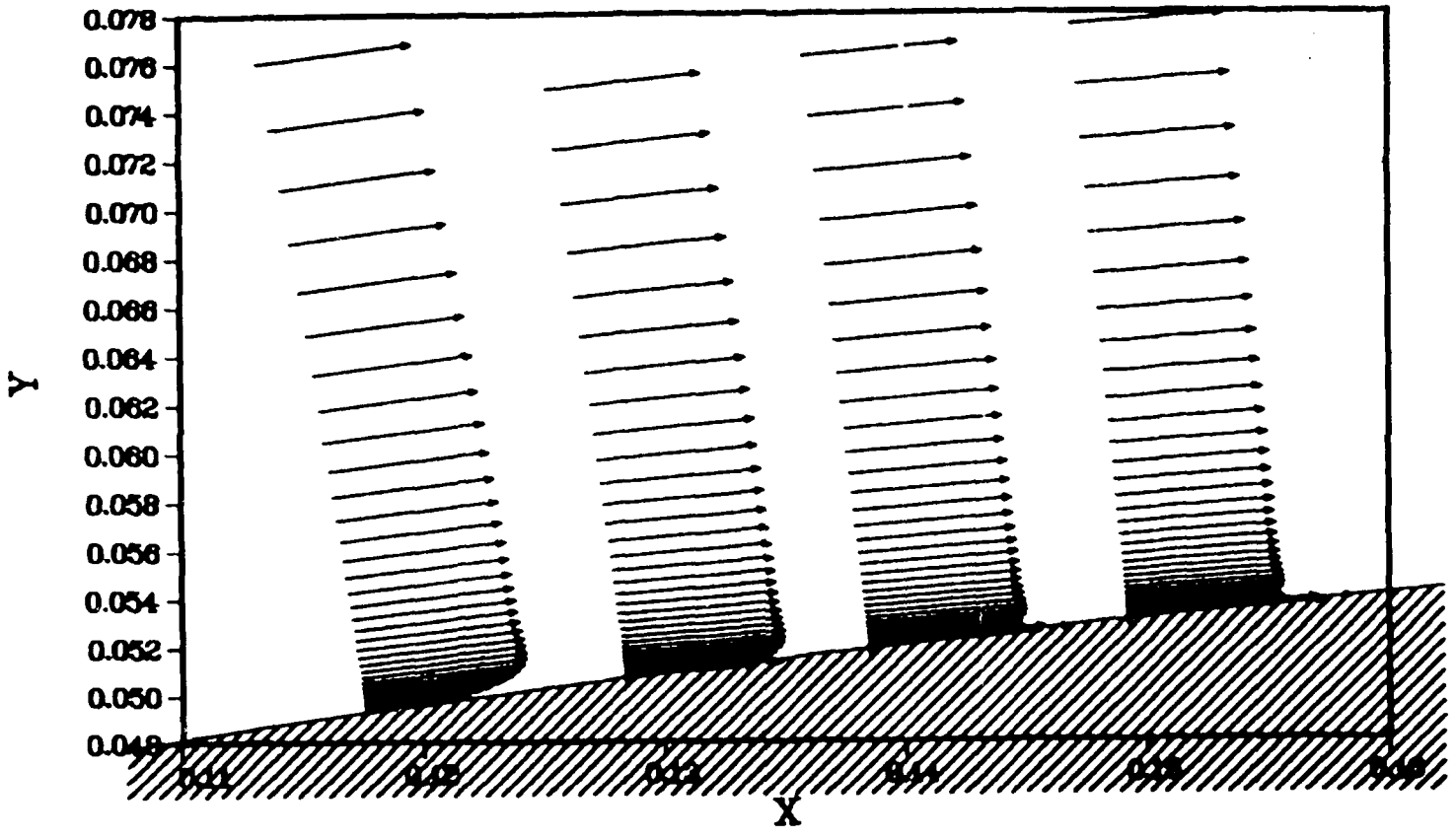
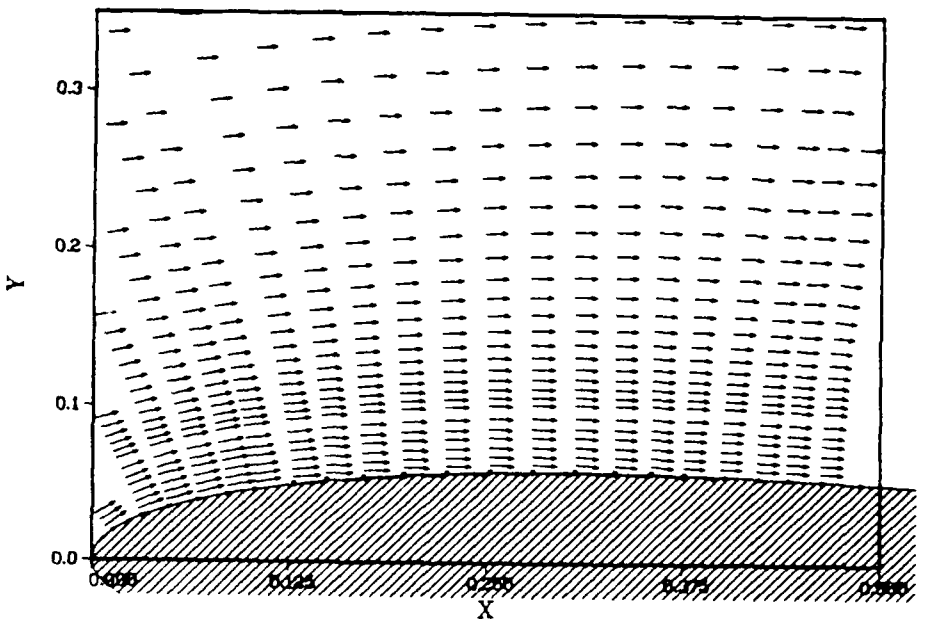


Figure 18. Velocity Boundary Layer Profile above Suction Surface (Cv=.05)

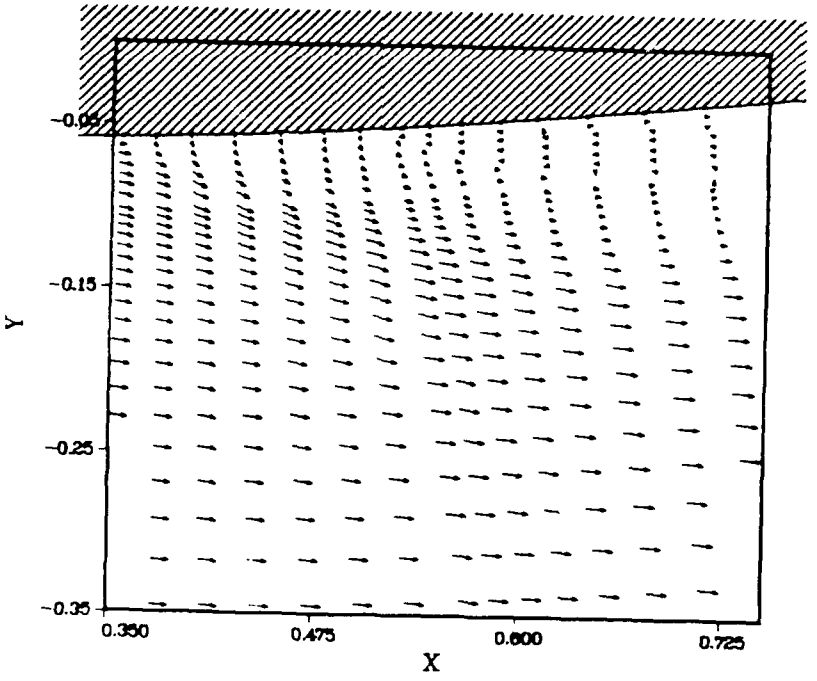
a dimension less than 0.002 as shown in Figure 18. This large surface gradient in velocity produces vorticity and thus has a large effect on the viscous force drag. A better model of suction might decrease drag but should still only have a small effect on lift.

The results for blowing can be explained in this way: at a 90 degrees ejection angle, density is constant along the ejection surface and there is no v component to velocity; therefore, continuity is satisfied if du = 0 with both du/dx and dv/dy $\cong$ 0 the skin friction is negligible right above the surface. However the effect of blowing produces vortices immediately behind the ejection surfaces shown in Figure 19. The vortices are shed through interaction with a shear layer formed by blowing as it turns the free stream At the trailing edge these counter-clockwise rotating vortices induce circulation around the trailing edge which leads to the formation of a clockwise rotating vortex.(5:196)

The trailing edge vortex is more powerful than the vortex produced by ejection as shown in Figure 20. This is due to the higher energy of the flow on the upper surface and the lower energy flow between the shear layer and the lower surface. These vortices are shed in a complex interaction as shown graphically by the streamline and

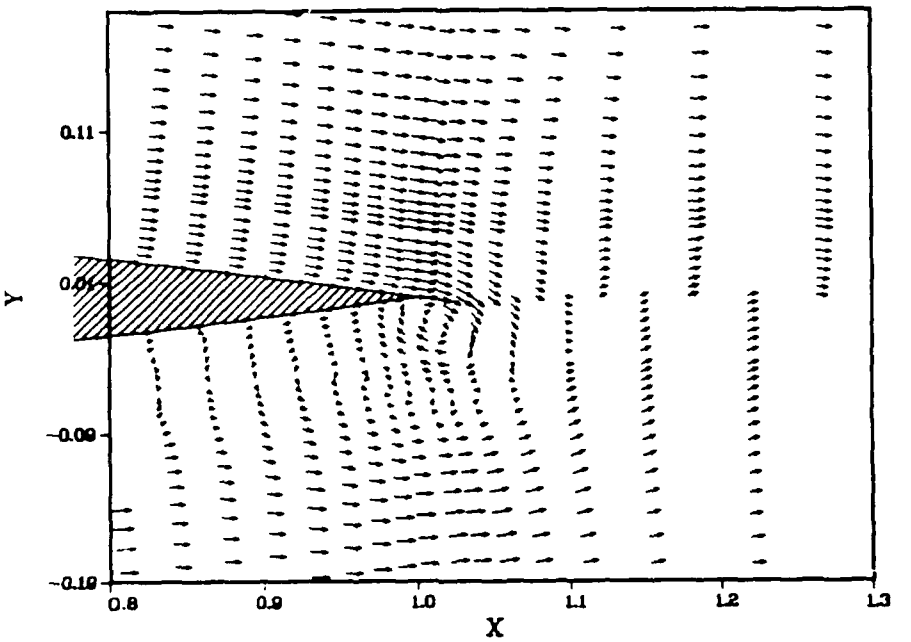


Upper Surface, Suction

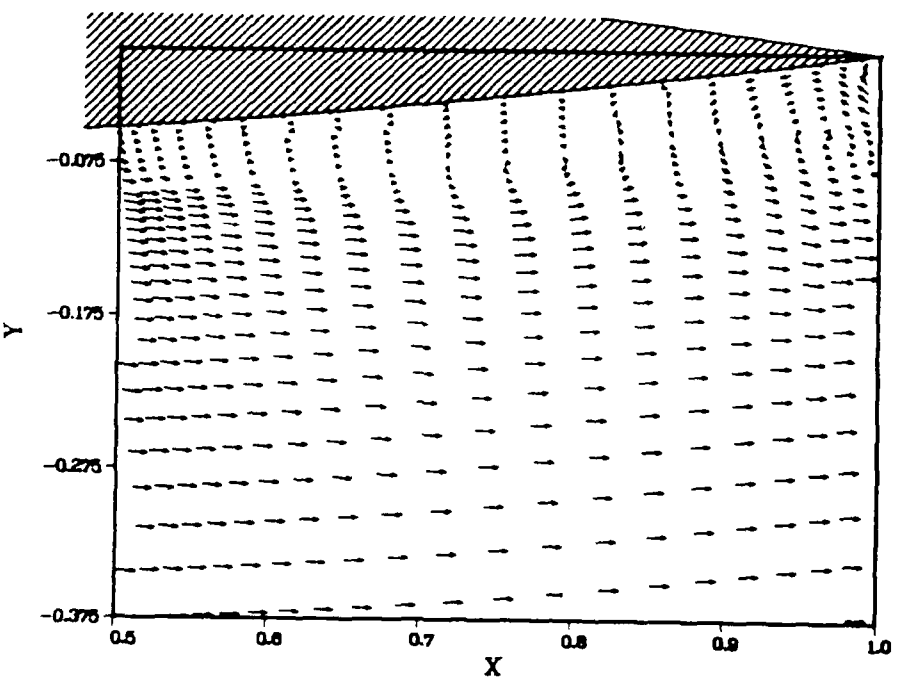


Lower Surface, Blowing

Figure 19. Velocity Profiles above Suction and Blowing Surfaces (Cv=.05)



Trailing Edge Vortex



Ejection induced Shear Layer

Figure 20. Velocity Profiles of Trailing Edge Vortex and Shear Layer (Cv=.05)

vorticity plots in Figures 21 and 22. As the vortices shed they produce oscillations in lift and drag. But these oscillations can not account for the higher mean values seen in lift due to blowing only.

An analysis of the surface pressure (Figure 16) reveal that the primary component to lift (Table 6) due to blowing is the pressure difference between the upper and lower surfaces on the forward portion of the airfoil. The primary effect of the vortex shedding is confined in the airfoil region behind the ejecting surface. The C_p difference producing lift can be explained by the compression effect the shear layer has on the flow by forcing it to turn.

Convergence Criterion for Periodic Solutions

Before an analysis could be performed, each case had to meet a convergence test as discussed in chapter three. Seventeen cases were run for Re = 1,000,000 and M = 0.3. Table 8 contains information on each case and pertinent information including the "number" by which they will be referred to. Of the seventeen cases listed the ten with ejection converged periodically. The remaining seven converged to a steady-state solution. Whenever possible, restarts from different converged solutions were used to reduce the transition period and thereby the number of

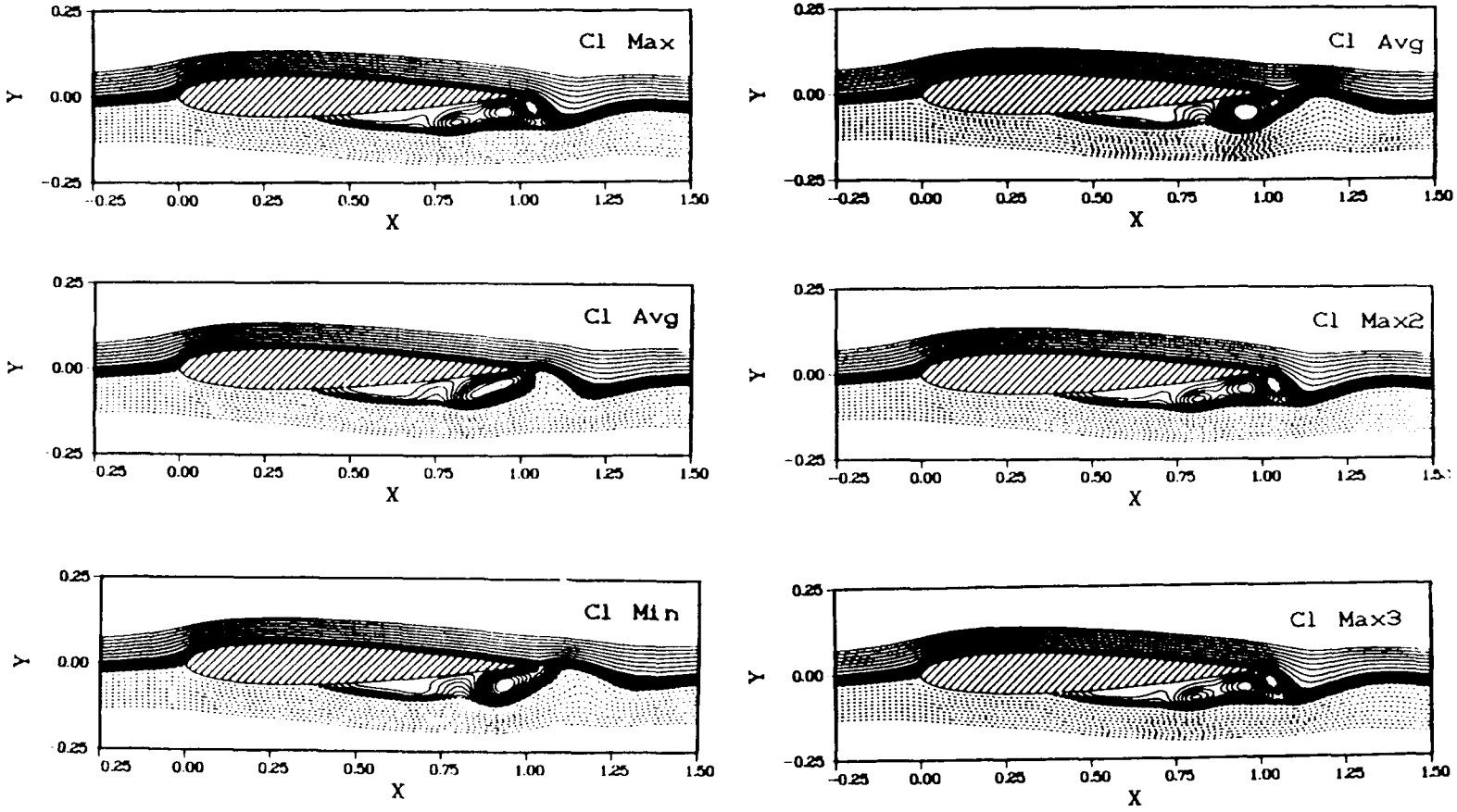


Figure 21. Streamline Comparisons for One Period in Cl (Cv=.05, δ=90°)

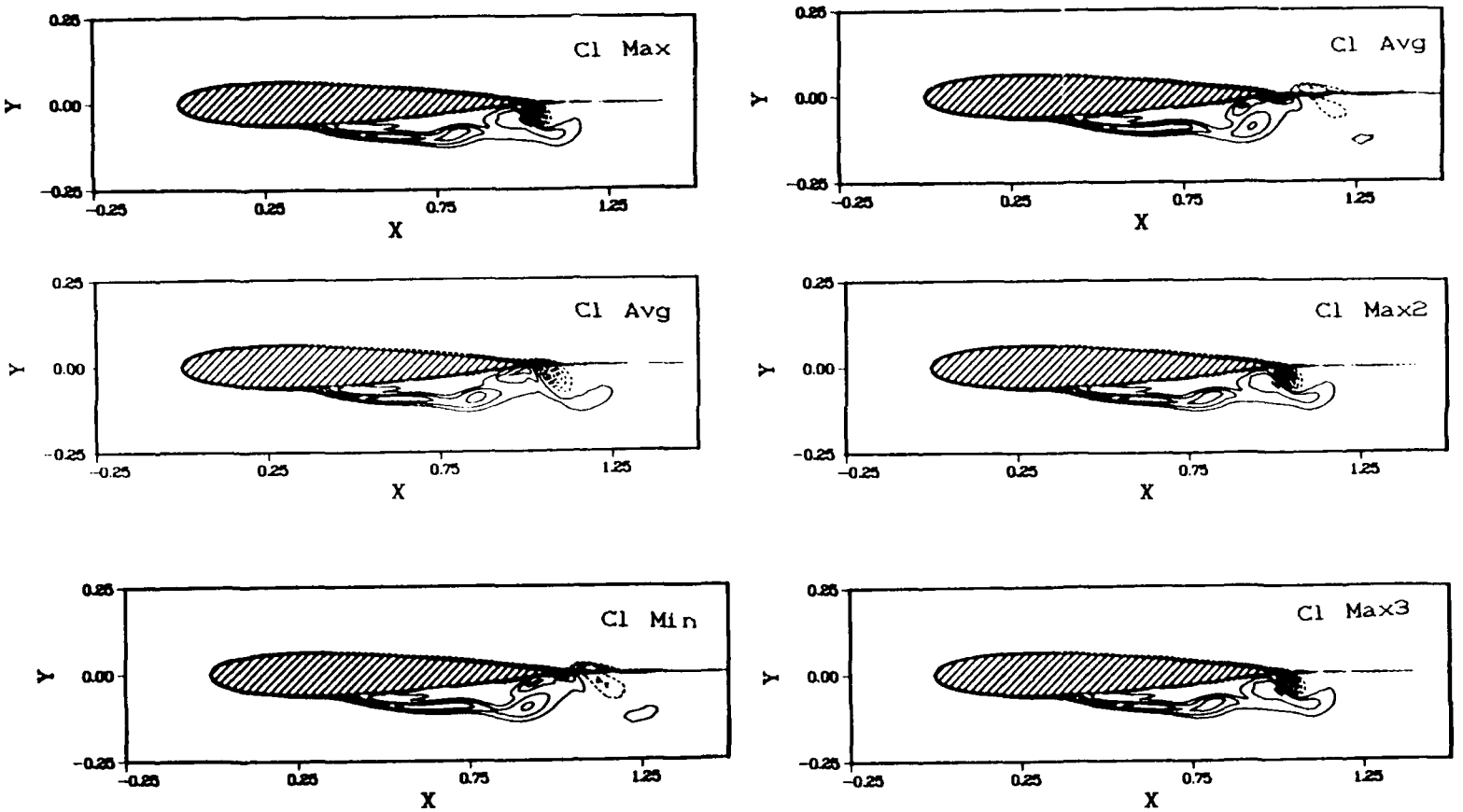


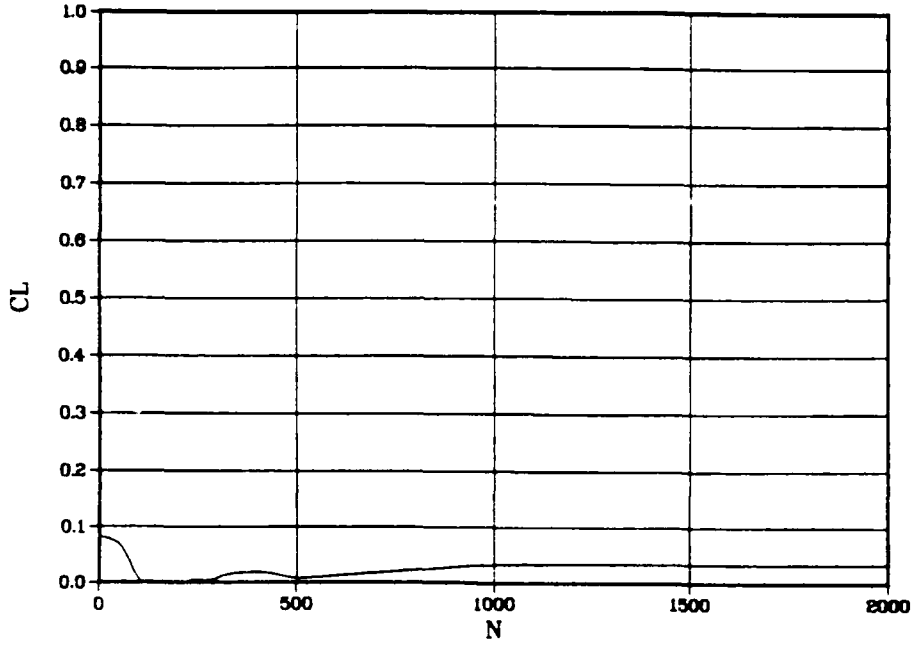
Figure 22. Vorticity Contour Comparisons for One Period in Cl (Cv=.05, δ=90°)

#		restart	Cv	δ°	α°	transition %c upper	lower
1	clean	2	0.00	0	0	1.5	1.5
2	clean	new	0.00	0	0	5	40
3	suction	2	0.05	0	0	5	40
4	blowing	2	0.05	90	0	5	40
5	fan	6	0.01	90	0	5	40
6	fan	2	0.05	90	0	5	40
7	fan	6	0.05	45	0	5	40
8	fan	7	0.05	15	0	5	40
9	fan	6	0.10	90	0	5	40
10	fan	9	0.10	45	0	5	40
11	fan	10	0.10	15	0	5	40
12	clean	13	0.00	00	2	5	5
13	clean	new	0.00	00	2	5	40
14	fan	13	0.05	45	2	5	40
15	clean	16	0.00	00	4	5	5
16	clean	new	0.00	00	4	5	40
17	fan	16	0.05	45	4	5	40

Table 8. List of Model Configurations for M=.3, Re = 1,000,000

iterations needed to converge to the new solution.

Plots of C_l and C_d for each case can be found in Figures 23-33. A comparison of the amplitude and slope of convergence provides some interesting insight into the behavior characteristics of different Cv, δ and α. The plots indicate that as the suction rate increased the amplitudes in C_l and C_d decreased. Increasing δ and α had the opposite effect. From the figures it appears, in general, that as the amplitudes decreased in size they increased in frequency. Even for the cases of Number 11 (Figure 31) and number 5 (Figure 25) which may look similar to steady state solutions (Figure 23), detailed analysis shows that low amplitude and high frequency oscillations are present.



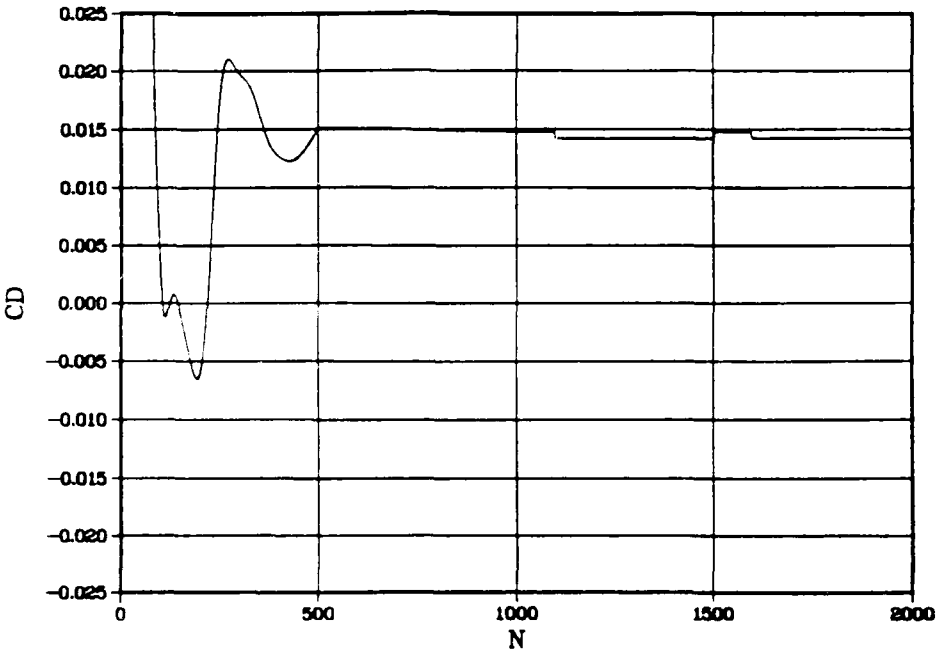
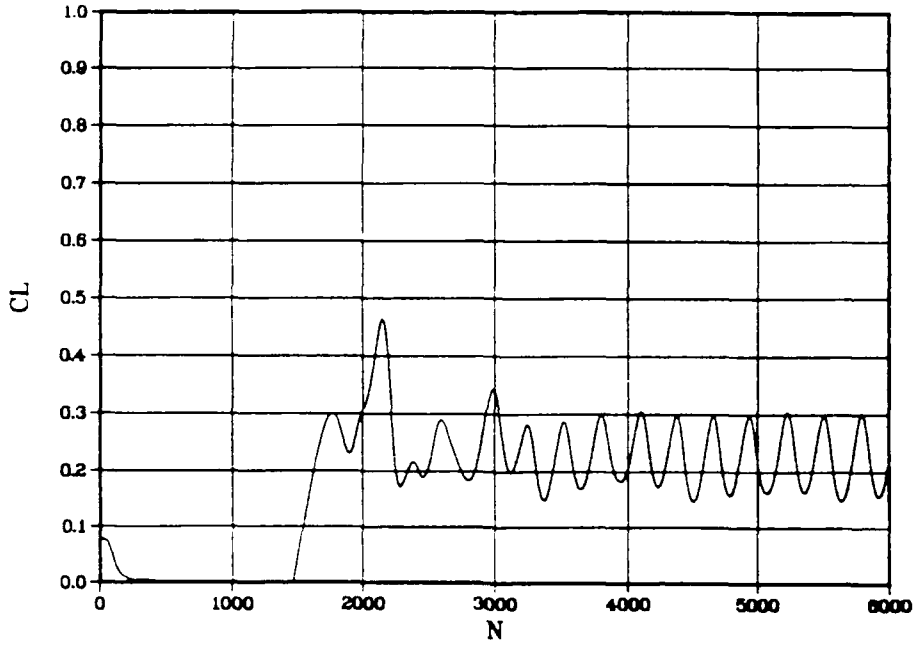


Figure 23. Force Coefficients vs Iterations, Number 3 (Suction only Cv=.05, $\alpha=0^{\circ}$)



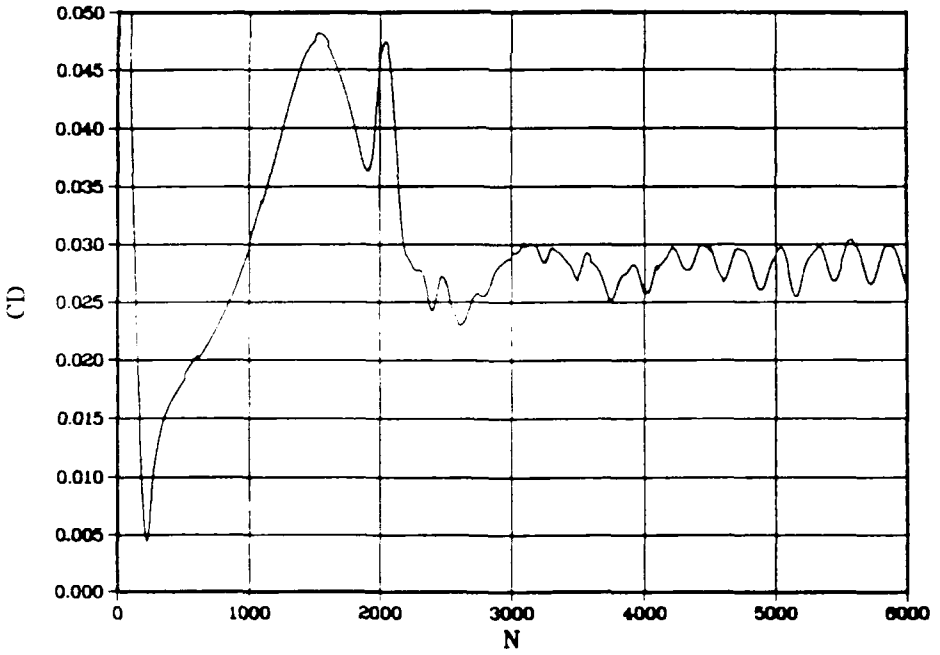
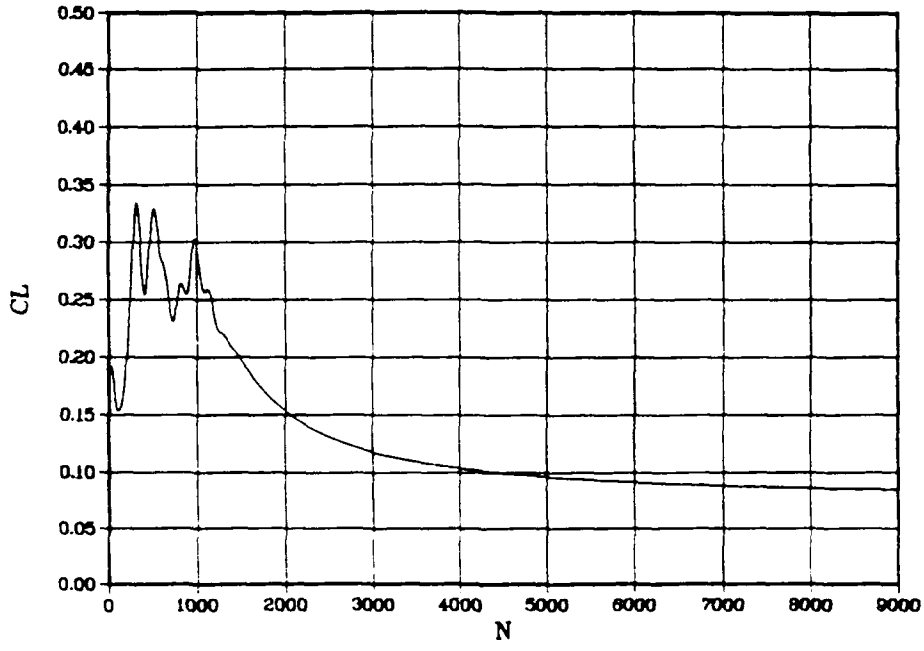


Figure 24. Force Coefficients vs Iterations, Number 4
(Blowing only Cv=.05, δ=90°, α=0°)



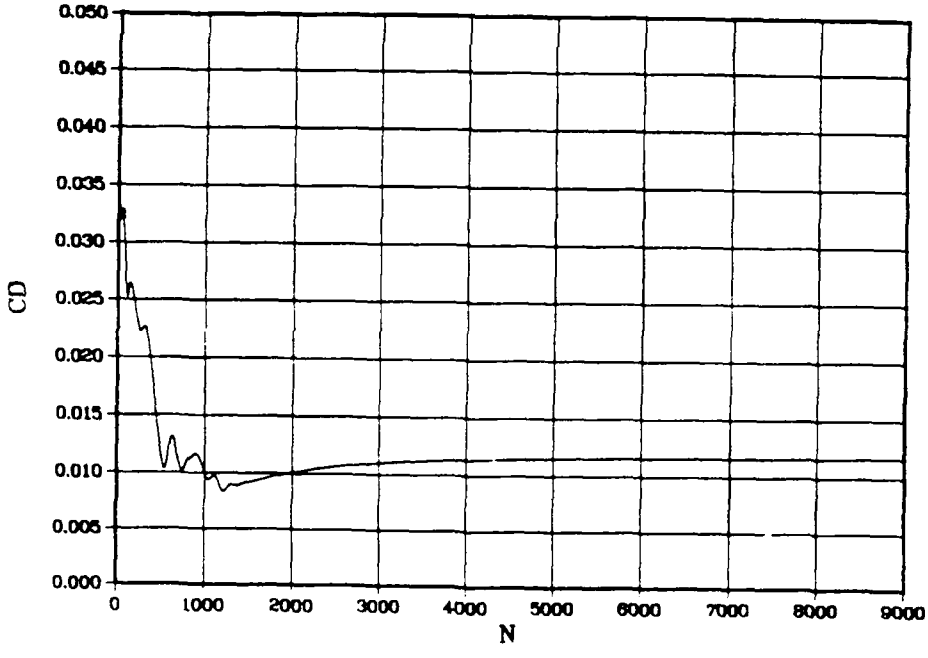


Figure 25. Force Coefficients vs Iterations, Number 5 (Cv=.01, δ=90°, α=0°)

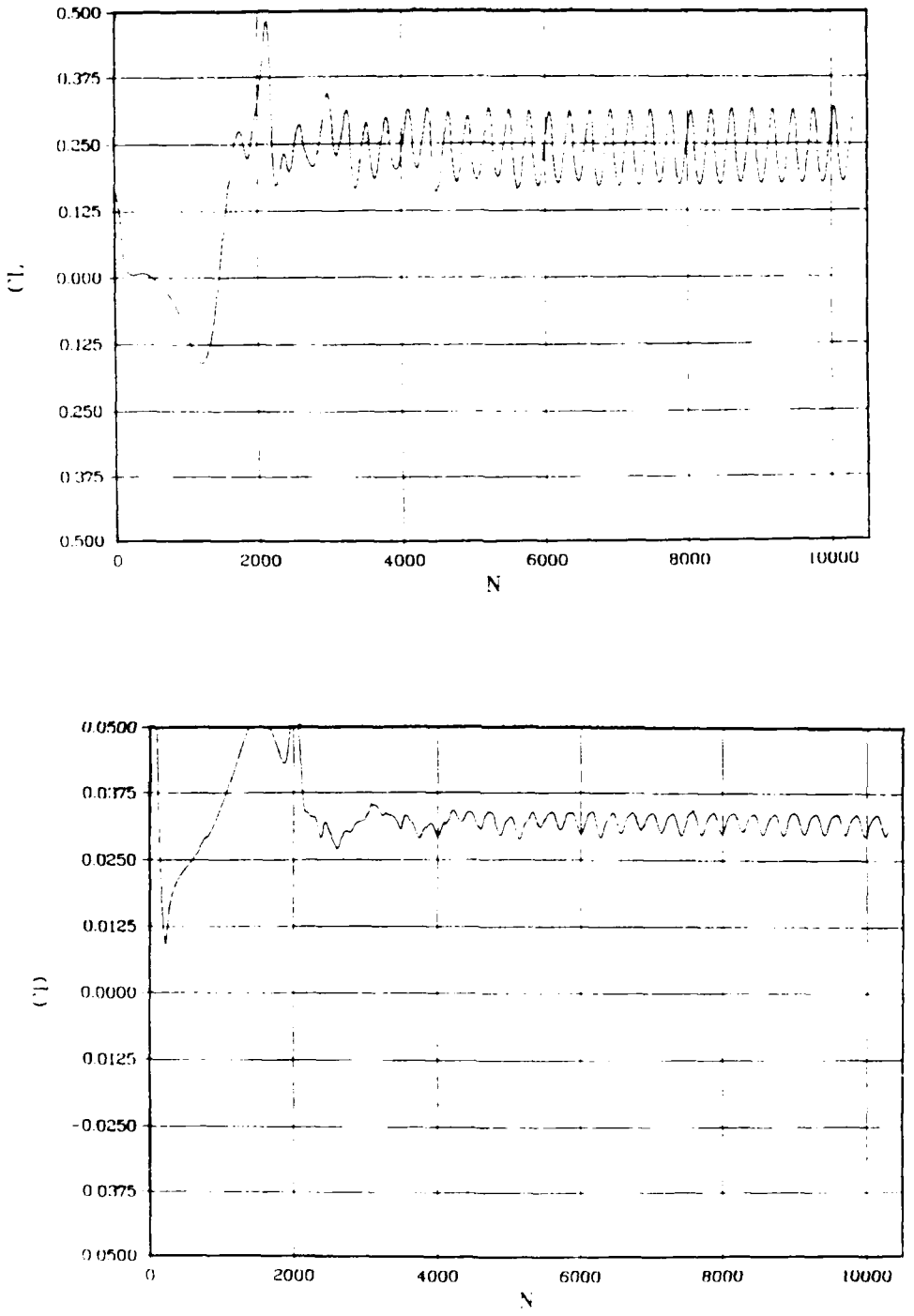


Figure 26. Force Coefficients vs Iterations, Number 6 (Cv=.05, δ=90°, α=0°)

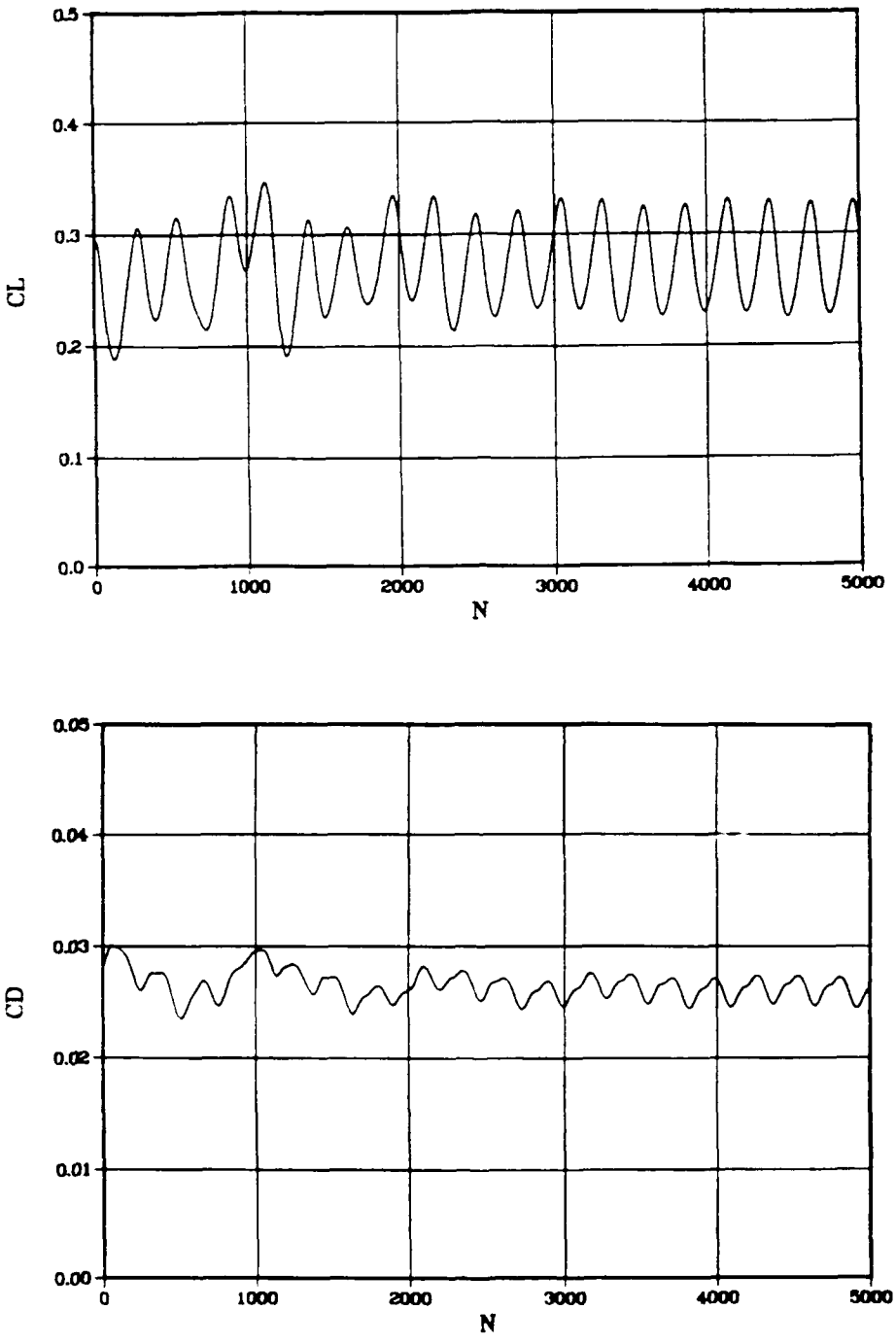


Figure 27. Force Coefficients vs Iterations, Number 7 (Cv=.05, δ=45°, α=0°)

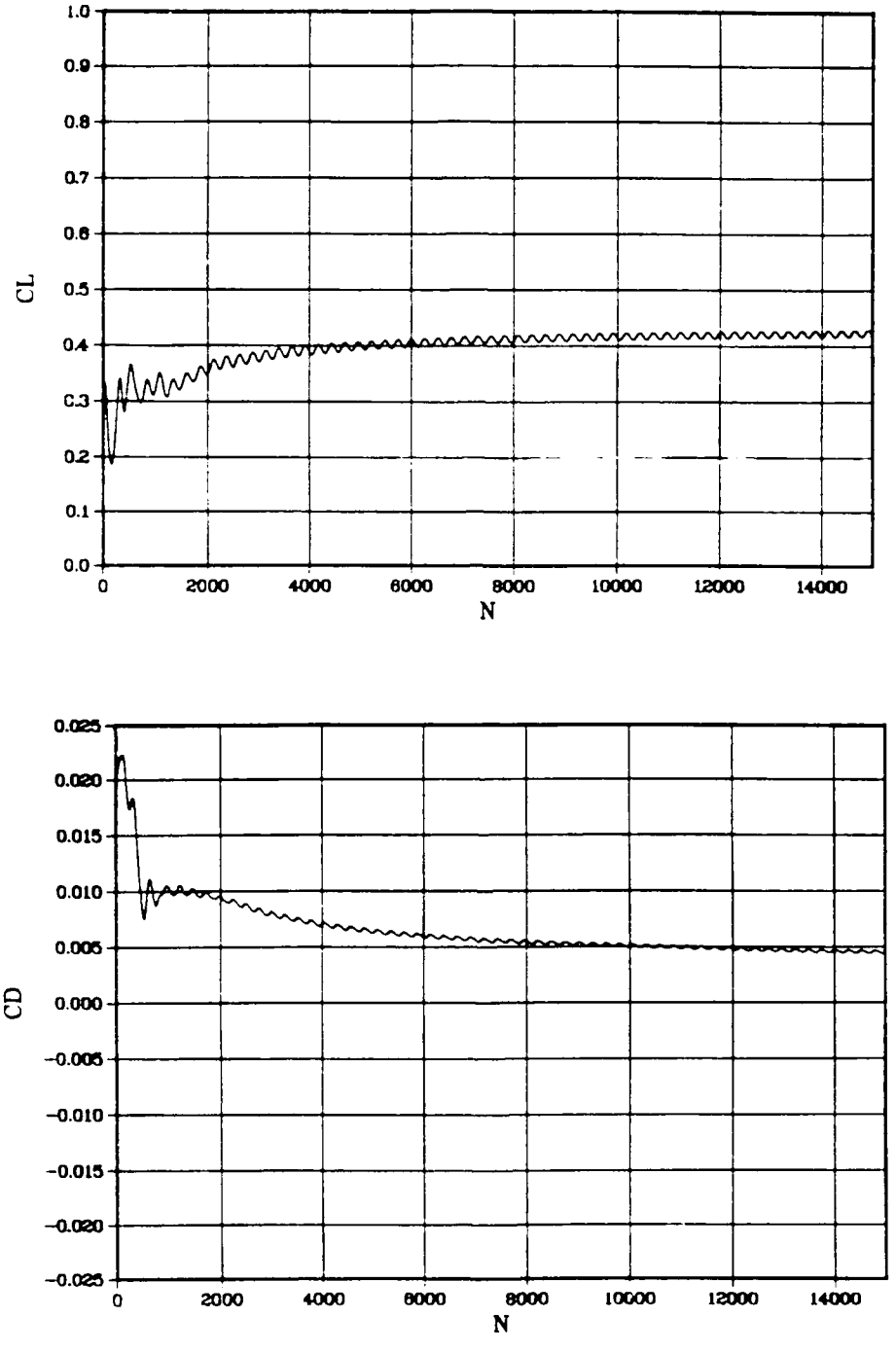


Figure 28. Force Coefficients vs Iterations, Number 8 (Cv=.05, δ=15°, α=0°)

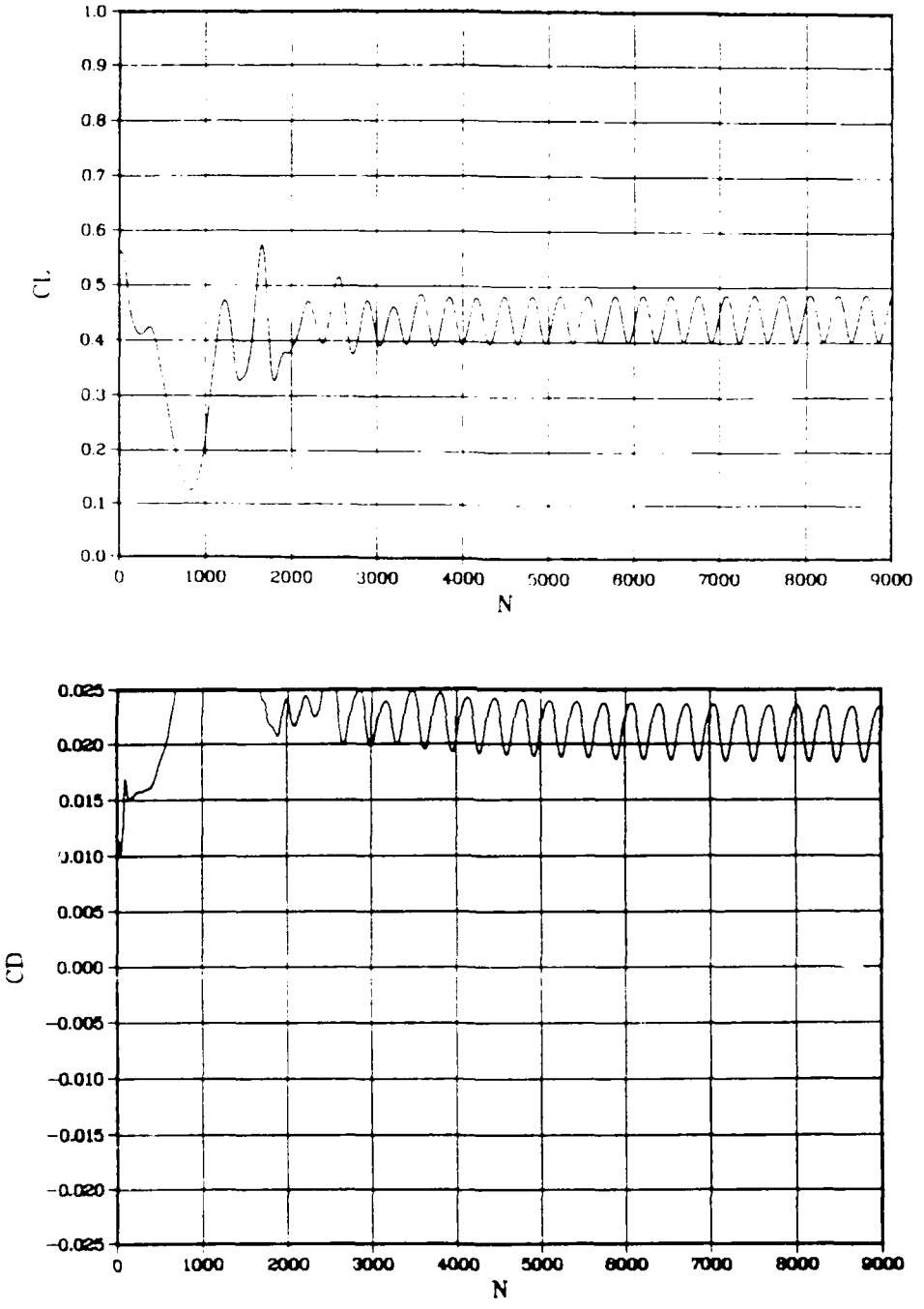
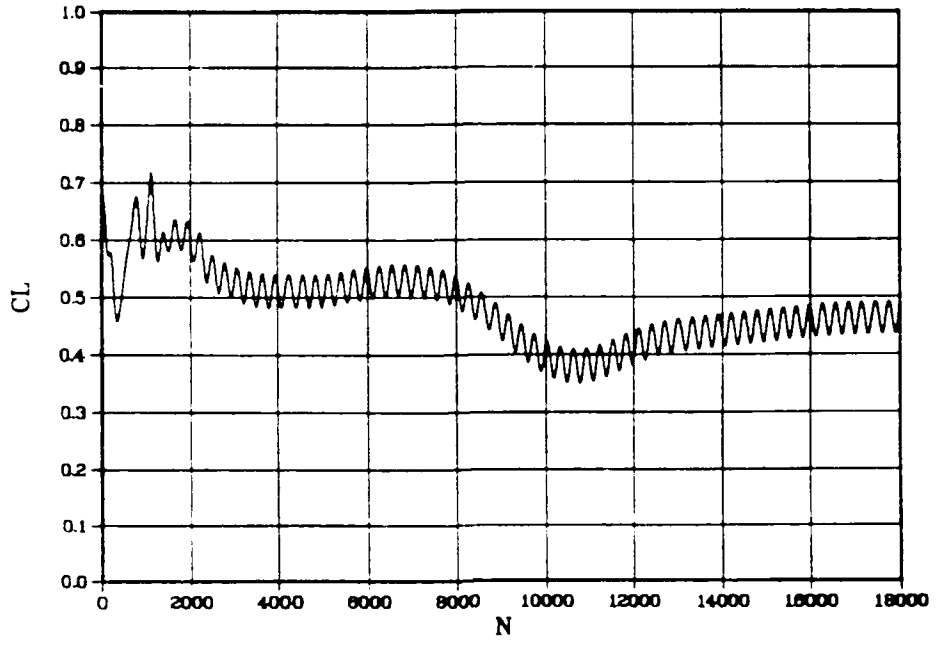


Figure 29. Force Coefficients vs Iterations, Number 9 (Cv=.10, δ=90°, α=0°)



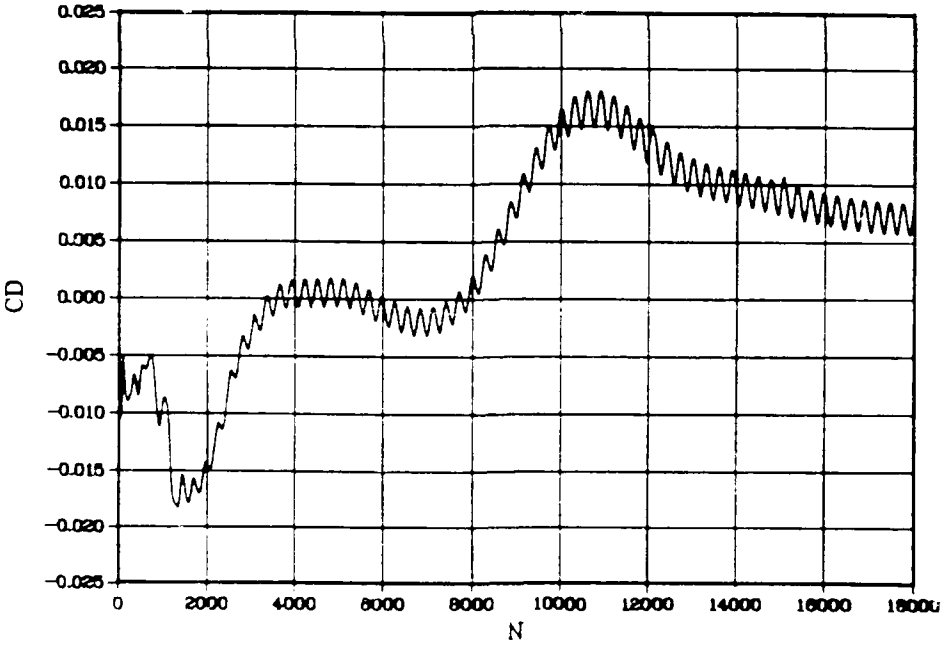


Figure 30. Force Coefficients vs Iterations, Number 10 (Cv=.10, δ=45°, α=0°)

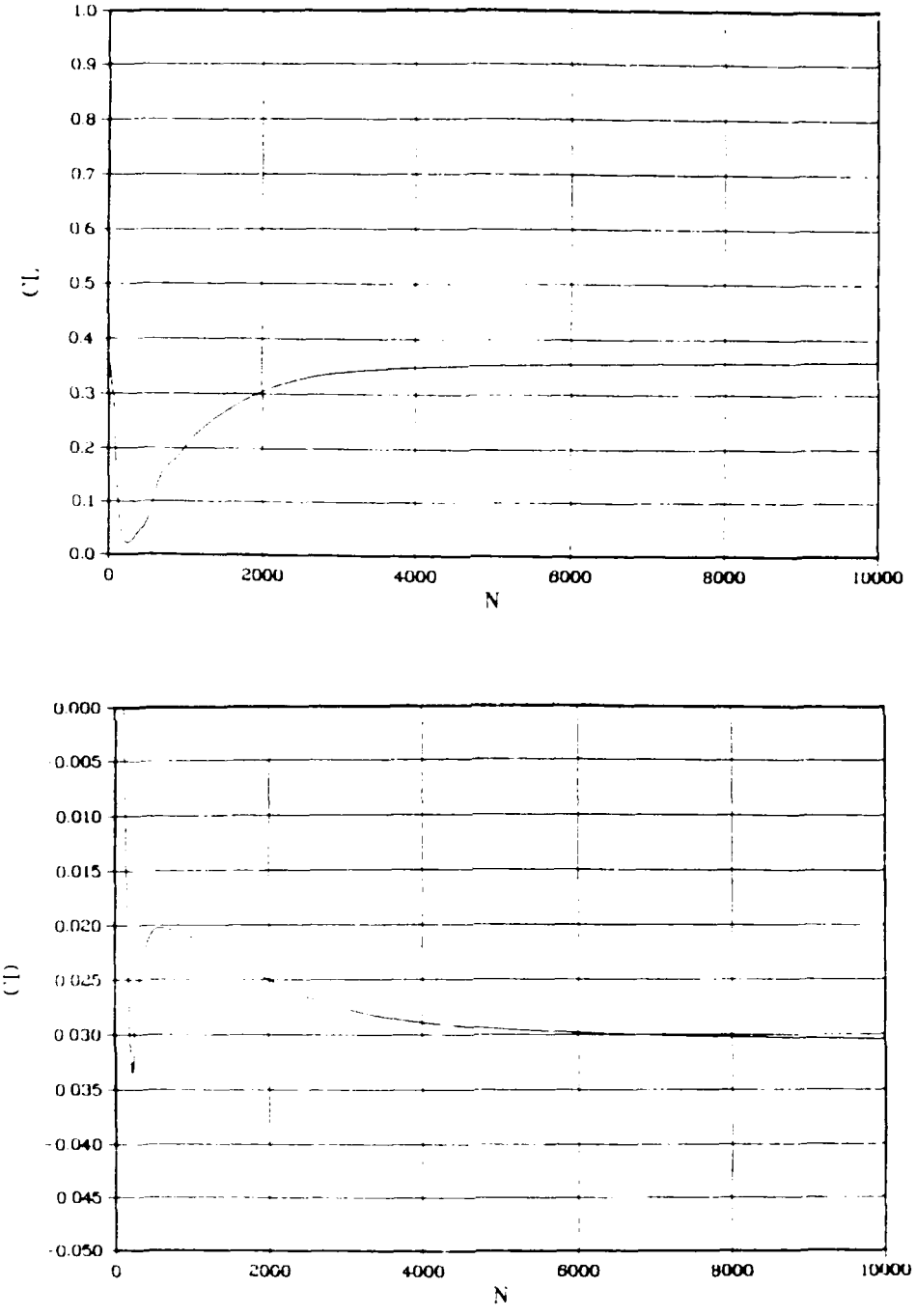


Figure 31. Force Coefficients vs Iterations, Number 11 (Cv=.10, δ=15°, α=0°)

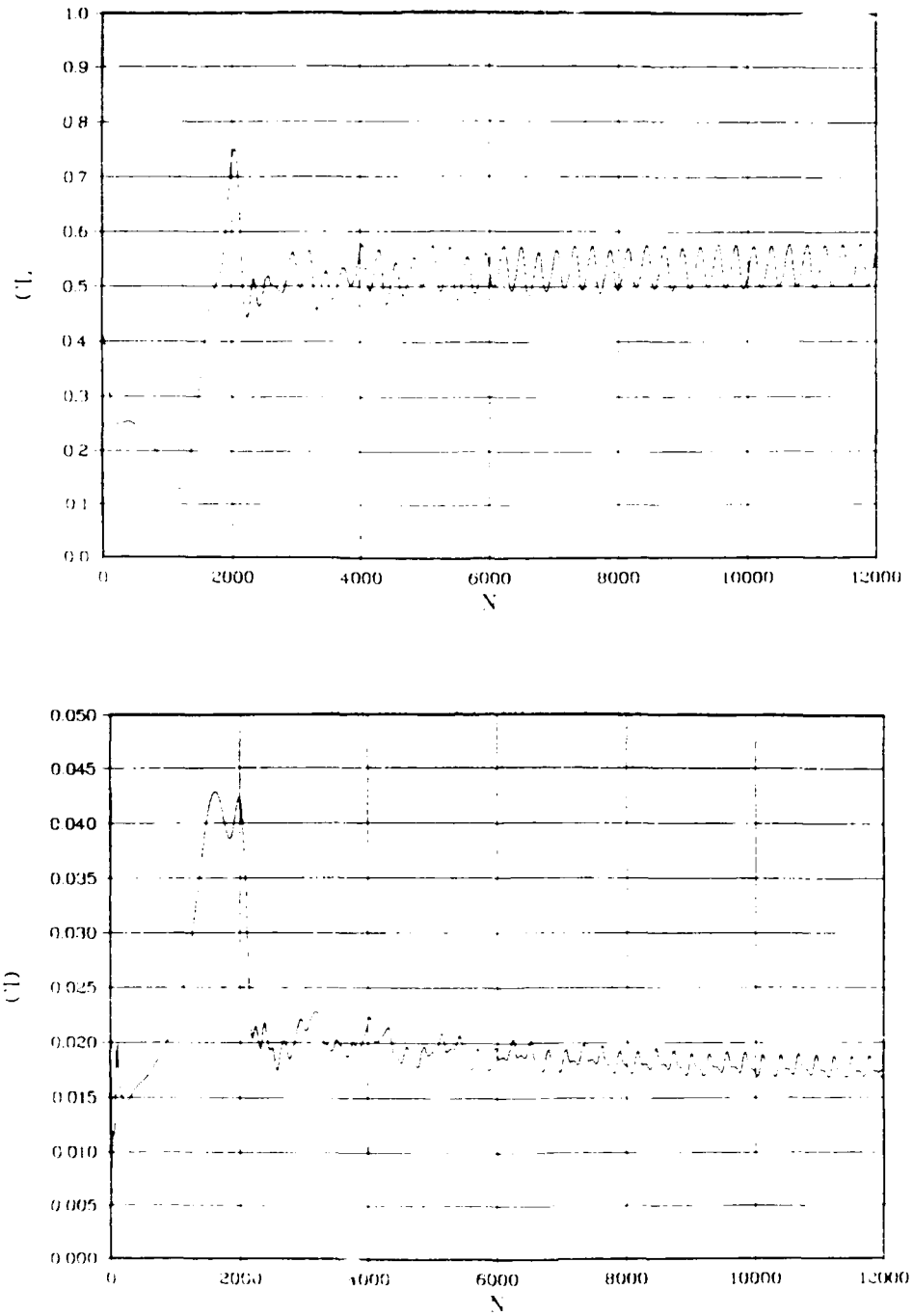


Figure 32. Force Coefficients vs Iterations, Number 14 (Cv=.05, δ=45°, α=2°)

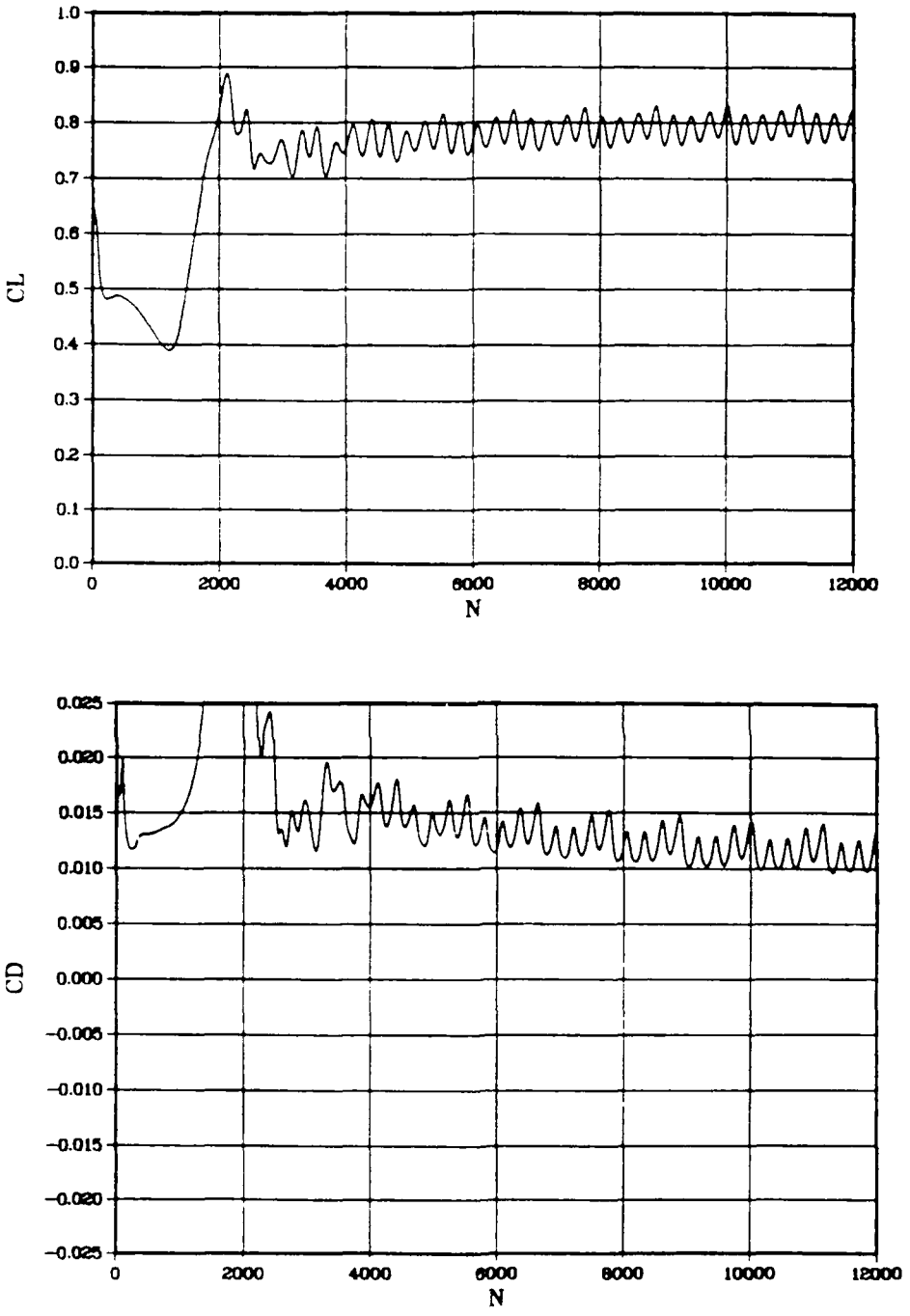
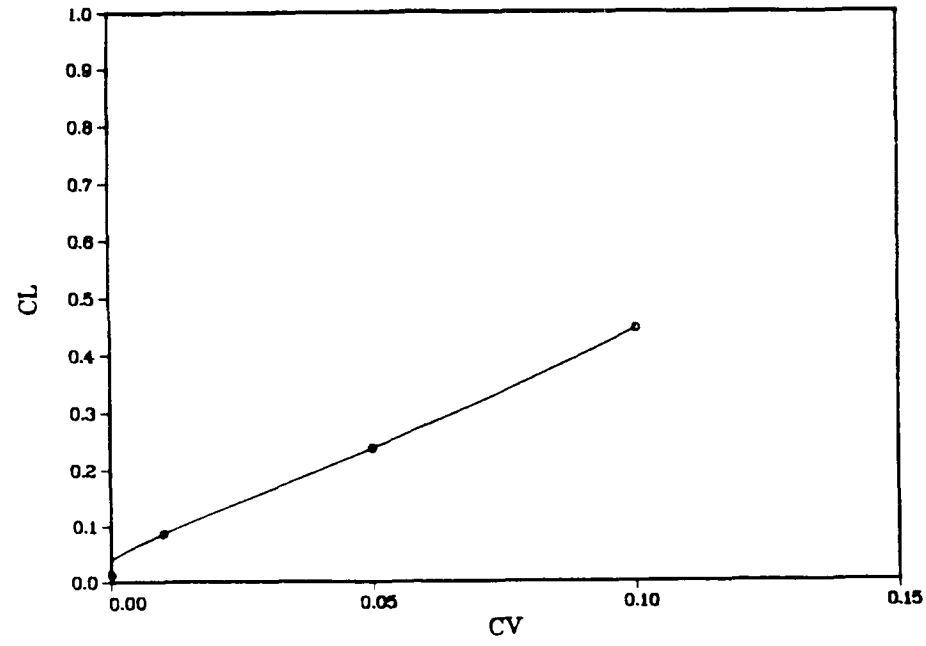


Figure 33. Force Coefficients vs Iterations, Number 17 (Cv=.05, δ=45°, α=4°)

Upon close inspection, small harmonics or secondary waves riding on the dominant wave can be seen, especially in the C_d plots. This is probably due to the interaction of the shedding vortices as mentioned above. Except at angle of attack, these harmonics are to small to affect the convergence criterion used. At angle of attack the harmonics are more pronounced. For Number 14, Figure 32, a set of four distinct oscillations with the same frequency formed early and later converged to a single oscillation. At four degrees, Number 17 (Figure 33), a similar quartet persisted and became periodic. This is true for both C_l and C_d. This behavior is probably due to a more complex interaction of the shedding vortexes induced by the angle of attack. A Fourier analysis could provide more information on the interactions but it is beyond the scope of this report.

The Effect of Varying Suction Rate

In this study three different suction, Cv, cases were studied, at 1%, 5% and 10% of free stream velocity. For purpose of comparison all three cases were run at an ejection angle of 90°. The ejection velocities were almost twice that for suction as a result of a 2:1 area ratio. The C_l vs Cv plot, Figure 34, indicates the expected result that increasing suction velocities and



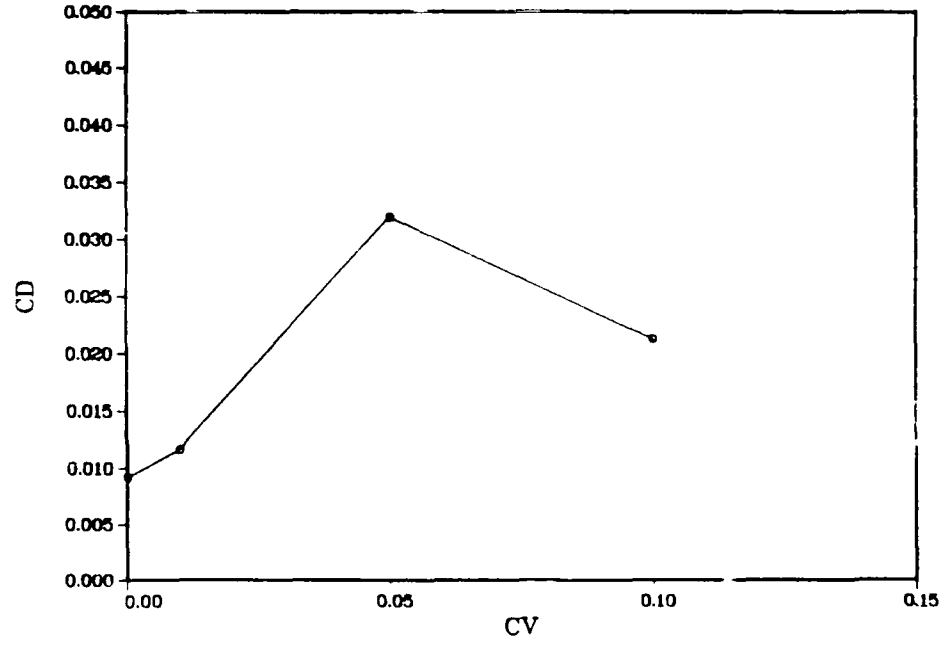


Figure 34. Force Coefficients vs Suction Velocity (δ=90°, α=0°)

corresponding increasing ejection velocities produce a near linear increase in lift. The surprising results in the C_d plot, Figure 34, suggests that there is a maximum drag produced near Cv equal 5%. Cv increasing beyond this critical value reduces drag. More points are need to draw firm conclusions concerning this particular behavior.

The normal (90°) blowing produced no induced drag due to mass flux at the ejection surface and minimal shear drag on the suction surface. The mass flux component of lift is almost negligible due to the small values of Cv as is demonstrated by an approximation to the momentum flux

$$\text{Lift} \cong \dot{m}\cdot\bar{V}\,\hat{j} = -\rho A(\bar{V}\cdot\bar{V}\hat{j}) = (\rho A v^2)_{lower} - (\rho A v^2)_{upper} \quad (92)$$

$$C_l = \frac{2L}{\rho_\infty V_\infty^2} = 2L \quad , \; v_{lower} \cong Cv \; , \; v_{upper} \cong 1.82Cv \quad (93)$$

For Cv = 0.1, $\rho \cong 1$ and A = 0.25 and 0.125, $C_l = 0.0012$ a very small value.

The lift and drag behavior can be directly related to the pressure distribution forward of the fan as the lift coefficients in Table 9 and the C_p surface distribution in Figure 35 indicate.

The biggest surprise is that varying the suction rate does not affect the pressure distribution on the lower surface nearly as much as on the upper surface. Since the

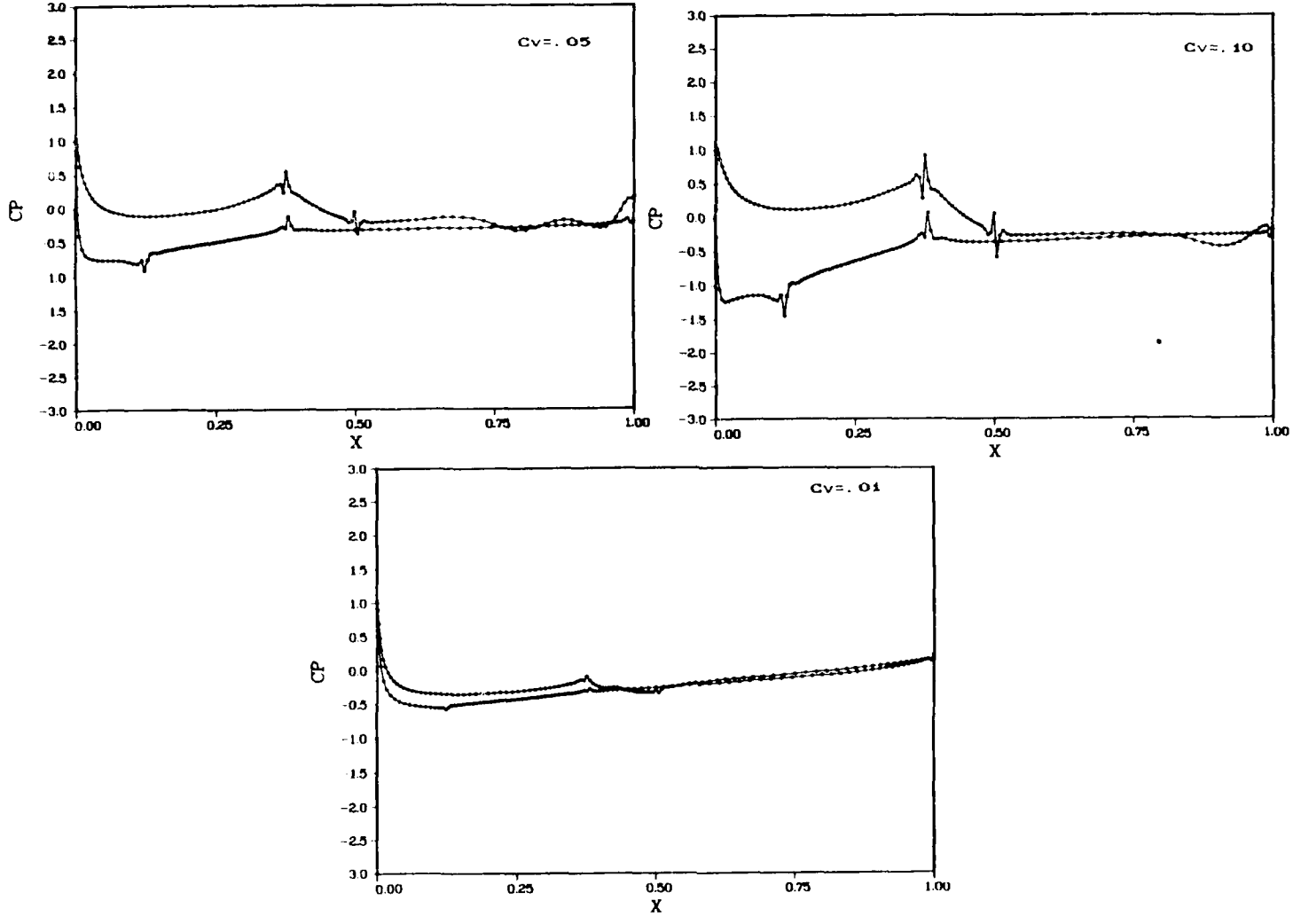


Figure 35. Comparisons of Cp for Different Suction Velocities (δ=90°, α=0°)

Number	Cl total	Cl press	Cl viscous	Cl mass
5	.08445	.08456	-.00016	.00004
6	.31719	.31599	.00011	.00108
9	.48454	.47984	.00036	.00434
	Cd total	**Cd press**	**Cd viscous**	**Cd mass**
5	.01223	.00190	.01033	.00000
6	.03232	.01629	.01599	.00004
9	.02392	.00578	.01797	.00016

Table 9. Cl and Cd Component Comparison vs Suction Velocity

negative pressure coefficient on the upper surface exerts a force normal to the surface and the greatest changes near the leading edge where the surface area has the largest vertical component (y) component. This trend can be verified. The following equation approximates pressure drag and lift (see appendix D)

$$P_{lift} \cong \sum P_i dy \quad , \quad P_{drag} \cong \sum P_i dx \qquad (94)$$

If the above equations are applied to the C_p curves the lift coefficients in Table 9 can be obtained. The mechanism creating these pressure differences in the Cp curves can not be easily explained. On the other hand The viscous drag is directly related to the suction velocity as is evident in the cf surface plots of Figure 36. The larger the suction the greater the drag over the suction region. Here again it is the pressure distribution which influences the total drag.

Some mechanism relating the ejection velocity to

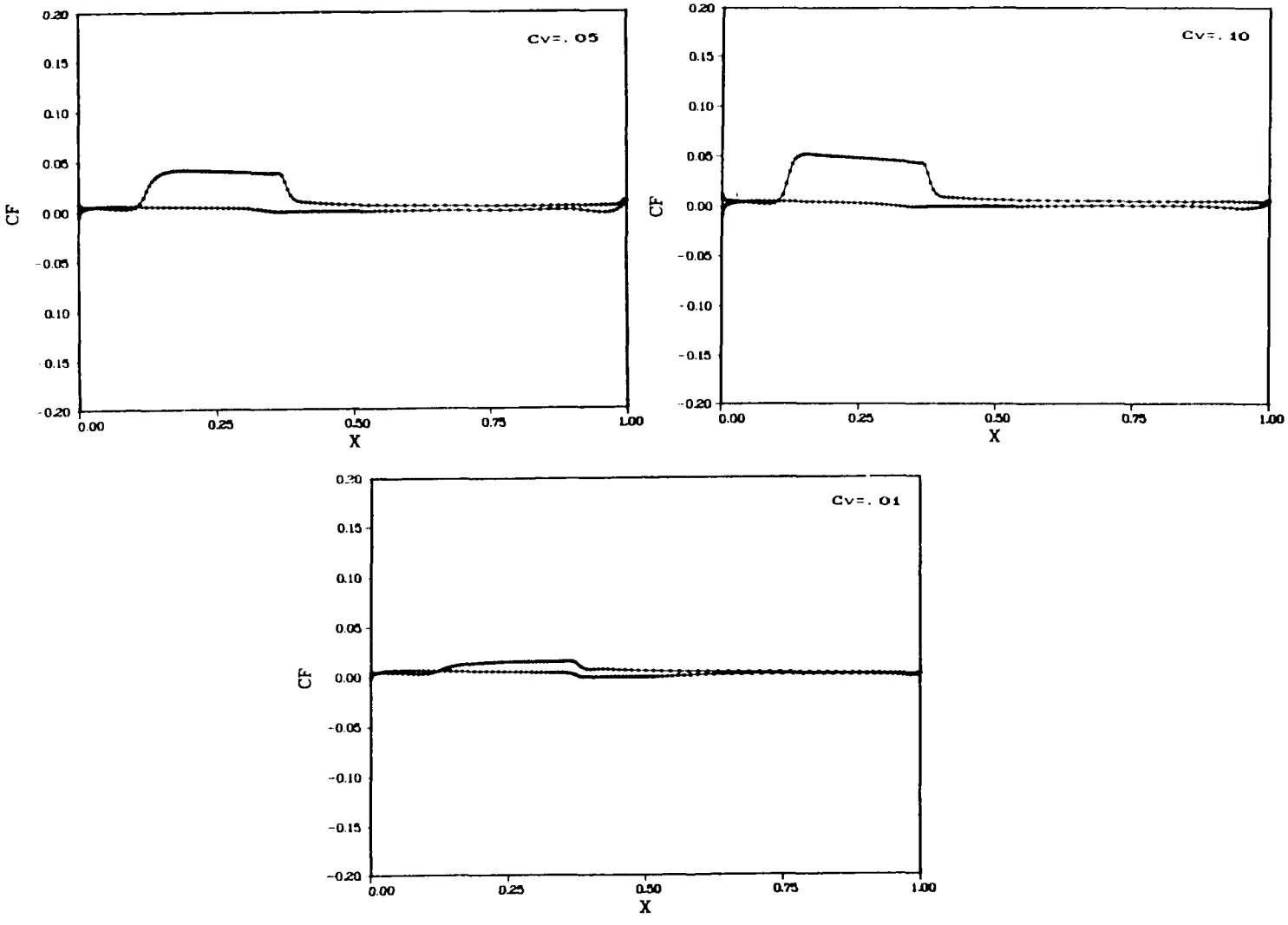


Figure 36. Comparisons of Cf for Different Suction Velocities (δ=90°, α=0°)

pressure distribution must account for this; but it is not the vortex shedding. Figure 37 compares the C_p plots for different points in the lift curve, Figure 25, for Number 6. The corresponding streamlines and vorticity contours were discussed earlier, see Figures 21 and 22. A comparison indicates that vortex shedding has no discernible effect on the pressure on the forward section of the airfoil. The amplitudes in the lift curve can be accounted for in the region near the trailing edge. The C_p differences near the trailing edge are not nearly as large as at the leading edge and the mean value seems to be near zero.

The streamline and vorticity contours demonstrate the period of the oscillations very well. The streamlines do not depict the shear layer in the ejection wake but do reveal the vortex at the trailing edge and the oscillations in the airfoil wake.

The vorticity contours reveal the shear layer by many small vortex cells. The maximum lift corresponds to a well formed vortex at the tail and the location of the wake at a minimum point below the cord. As the vortex breaks down the lift drops to a minimum and no tail vortex is present. As the vortex reforms the lift begins to climb again and the wake moves to a maximum point above the cord. The drag behavior is similar except that it is slightly out of phase, but with the same period.

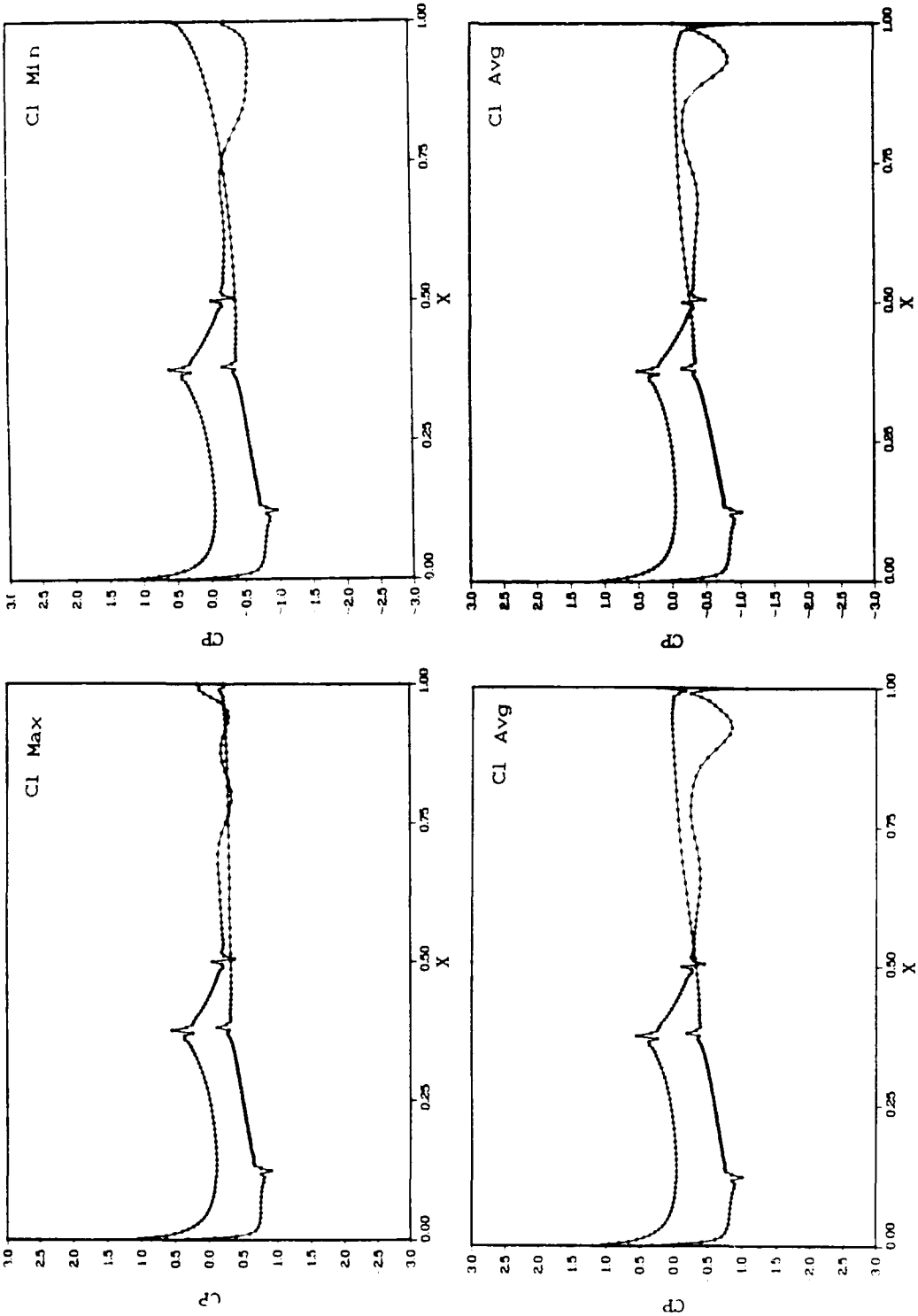


Figure 37. Cp Comparisons for One Period in Cl (CV=.05, δ=90°)

The streamline and vorticity comparison in Figure 38 demonstrate that Number 9, Cv = 0.1, has the strongest vortex and the broadest shear layer. Figure 38 also indicates that for Cv = 0.01 that there is no trailing edge vortex and a very thin shear layer. It is evident that the greater the suction velocity the greater the disturbance at the blowing surface which creates stronger ejection vortices and a wider shear layer. This broad shear layer may acts as a buffer on the rear body and decrease the shedding effects. This could account for the decreasing trend in amplitude with increasing suction velocity shown in the C_l and C_d plots.

The Effect of Varying Ejection Angle

For this portion of the study, suction rates of 5 and 10 percent were compared at ejection angles of 90, 45 and 15 degrees. The C_l and C_d vs δ curves in Figure 39 indicate different behavior for the two suction rates. For Cv = 0.1 the trend is an increase in both lift and drag. For Cv = 0.05 the trend in lift is reversed. The drag can be explained by examining the mass flux component of lift, Table 10. At any ejection angle the normal component of velocity v does not change; it must satisfy the constant mass flow condition

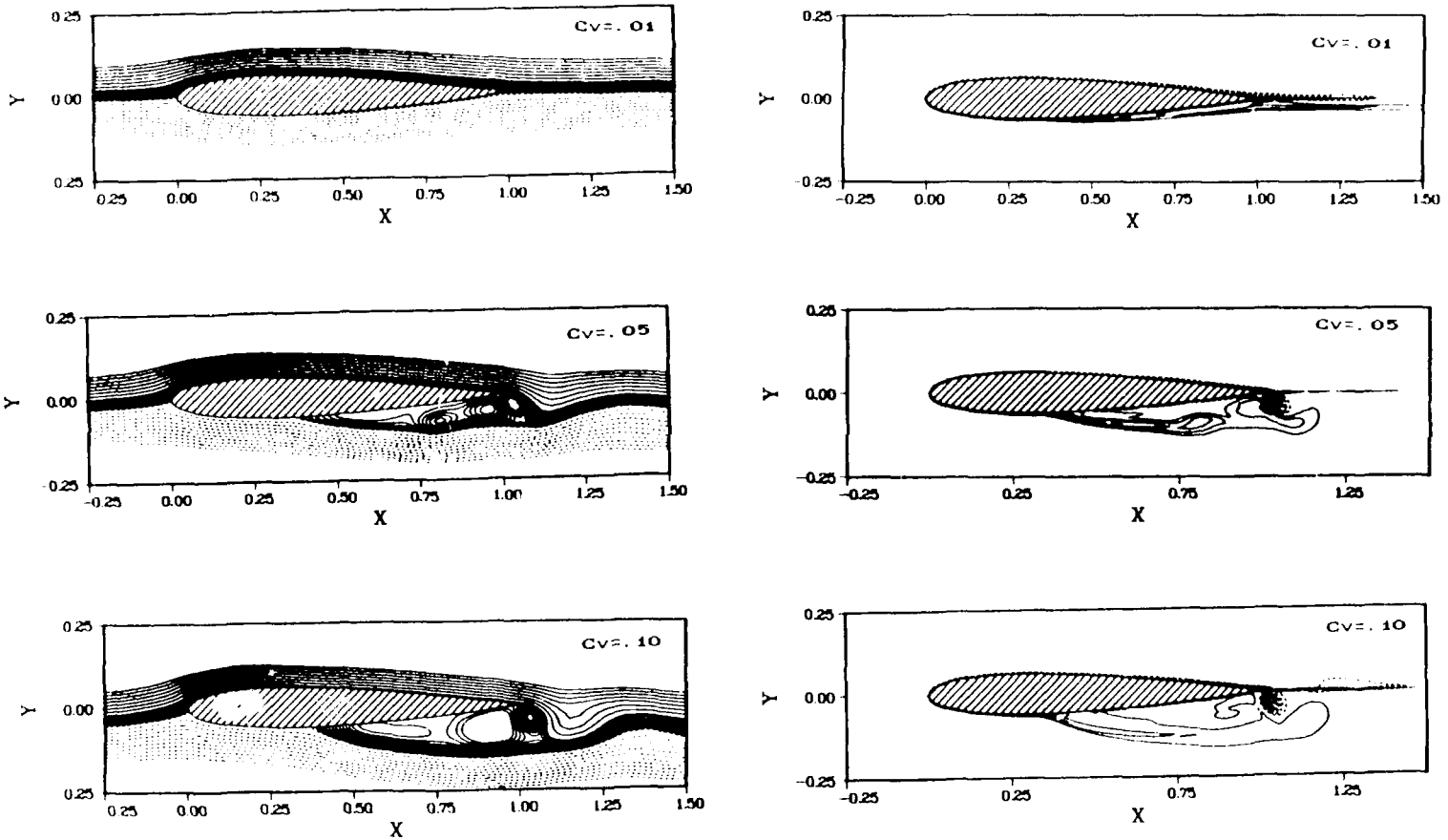
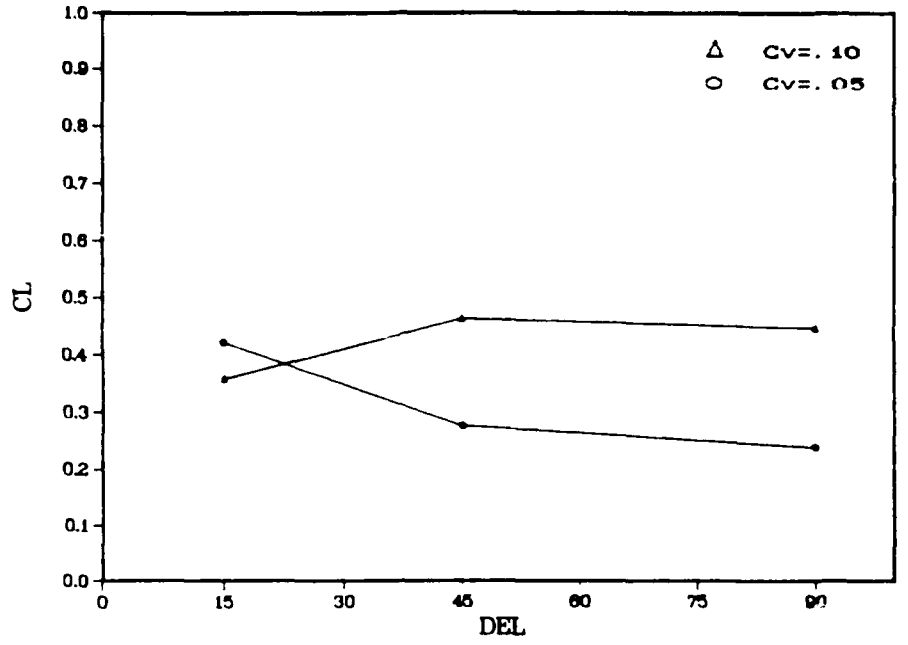


Figure 38. Streamline and Vorticity Comparisons for Different Suction Velocities (δ=90°, α=0°)



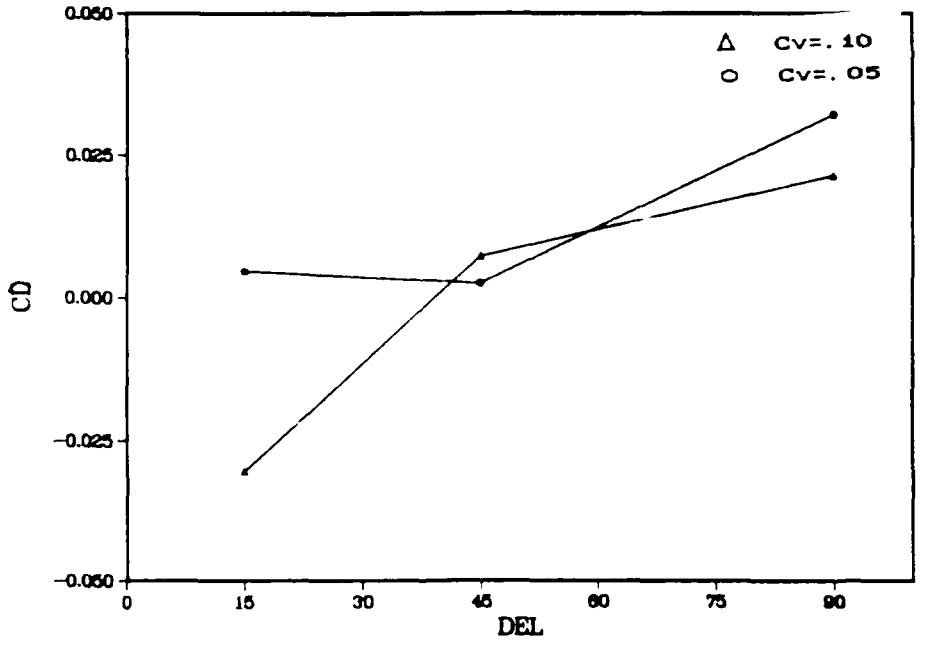


Figure 39. Force Coefficients vs Ejection Angle (Cv=.05, $\alpha=0^{\circ}$)

Number	Cl_{total}	Cl_{press}	$Cl_{viscous}$	Cl_{mass}
6	.31719	.31599	.00011	.00108
7	.29903	.29773	.00012	.00119
8	.41154	.40985	.00020	.00620
9	.48454	.47984	.00036	.00434
10	.43543	.43027	.00037	.00479
11	.35953	.35320	.00029	.00605

code	Cd_{total}	Cd_{press}	$Cd_{viscous}$	Cd_{mass}
6	.03232	.01629	.01599	.00004
7	.02675	.01320	.01595	-.00240
8	.00620	.00003	.01638	-.01020
9	.02392	.00578	.01797	.00016
10	.00889	.00048	.01809	-.00967
11	-.07989	-.00056	.01810	-.04128

Table 10. Cl and Cd Component Comparison vs Ejection Angle

$$\dot{m}_{in} = \dot{m}_{out} = \rho A V_{normal} \tag{95}$$

Where A is the surface area of the airfoil and does not change with ejection angle and V_{normal} is equal to v; therefore the mass flux component to lift does not change. Drag behaves differently. The velocity u is equal to $v/\tan(\delta)$ which is approximately $2CvCr/\tan(\delta)$ and for 45° approximately 1.8Cv and for 15° approximately 6.78Cv. As δ goes to zero u would go to infinity.

$$C_d = 2\ drag \cong -2\ \dot{m}\cdot\bar{V}\ \hat{i} = -2\ \rho A\left(\frac{v^2}{\tan^2\delta}\right) \tag{96}$$

The approximate calculations are listed in Table 11.

Cv	δ	Cd
.05	90	-0.0000
.05	45	-0.0011
.05	15	-0.0042
.10	90	-0.0000
.10	45	-0.0091
.10	15	-0.0334

Table 11. Cd Calculations for Surface Blowing (A = .125, ρ = 1.1 and V = 2 Cv/Cr, Cr = 1.1)

This explains the decrease in drag vs δ but does not account for the lift behavior.

As with the effect of varying suction velocities the lift comes mostly from the leading edge pressure differences. The influence of this pressure by changing the ejection angle can not be determined without more data.

The streamline and vortex plots are presented in Figures 40 and 41 for comparison. The vortex patterns at 90° are discussed above. As shown in Figures 40 and 41 the trend at 45° is a smaller and weaker vortex structures which accounts for the reduced amplitude in the C_l and C_d curves. The shear layer is also much smaller with a higher energy imparted by the u component of the ejection. These two reasons account for the smaller frequencies seen in the same curves. The smaller higher energy shear layer is closer to the body and prevents the build up of a strong trailing edge vortex. The smaller trailing edge vortices then must shed at a quicker rate than the ones at 90° where time was required for them to build in strength.

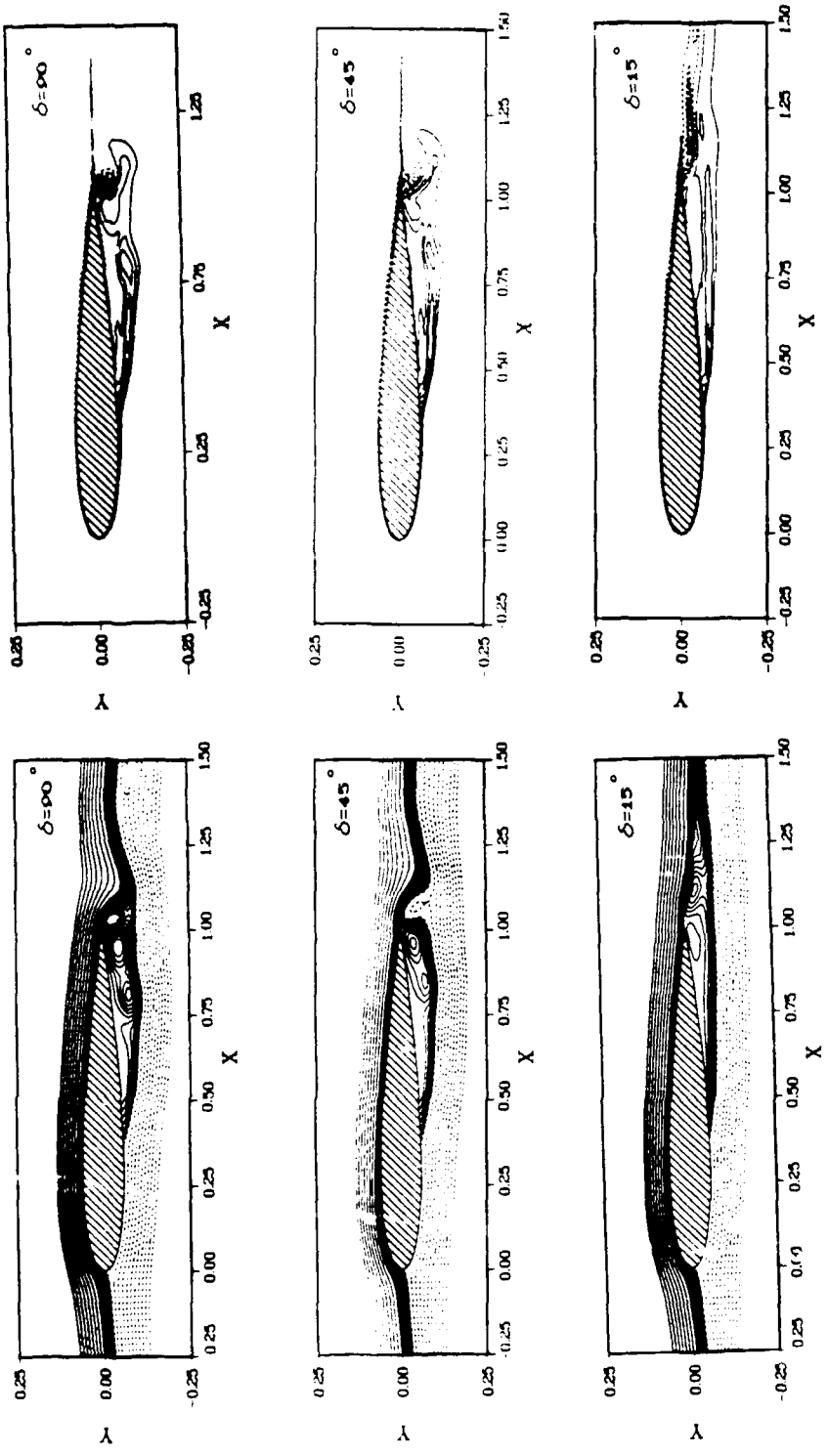


Figure 40. Streamline and Vorticity Comparisons for Different Ejection Angles (Cv=.05, α=0°)

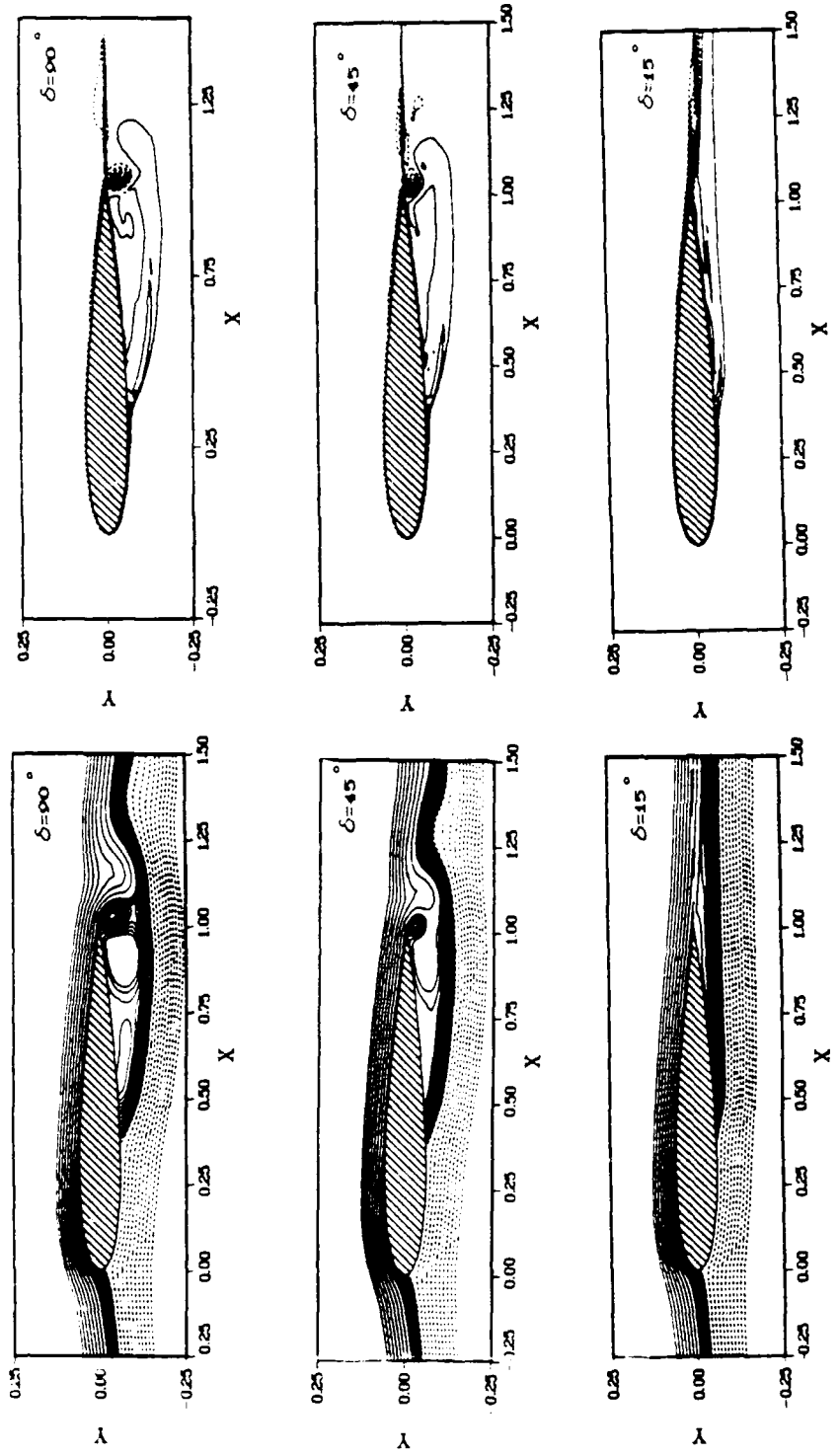
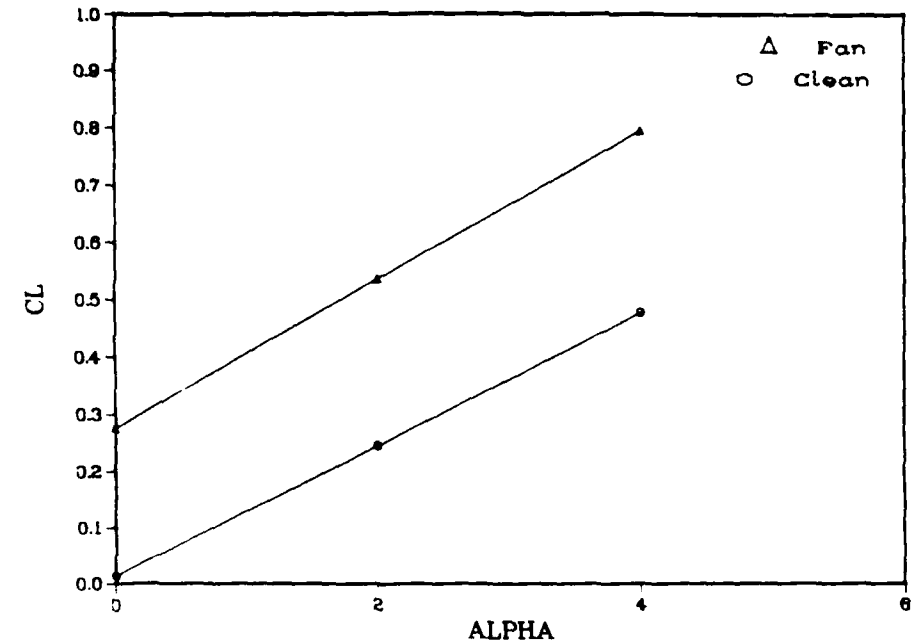


Figure 41. Streamline and Vorticity Comparisons for Different Ejection Angles (Cv=.10, α=0°)

At 15° the trend is even more pronounced. For Cv = 0.1 A trailing edge vortex was prevented from forming as shown in Figure 41.

The Effect of varying Angle of Attack

The most interesting results were obtained by varying the angle of attack. The C_l and C_d vs α curves are given by Figure 42. The lift curve shows the expected behavior. A delta C_l above that of a clean airfoil and constant with respect to α. Here again the major contributor to lift is the pressure distribution around the leading edge, see Figure 43 and Table 12. The drag curve behavior is unexpected. The trend for increasing α is a decrease in drag almost to the level of a clean airfoil. This behavior is unusual and cannot be explained by thrust due to mass flux. As α goes up the thrust component actually goes down because the direction is increasing the ejection angle by α. This is in agreement with the data in Table 12. Also from this figure the viscous components of drag are nearly constant; it is the contribution of pressure drag that is exhibiting the unusual behavior. At 4° angle of attack the pressure drag is acting as a thrust component. The mechanism causing this behavior is not known. The C_l vs C_d curve, Figure 44 is instructive. The fan curve retains the



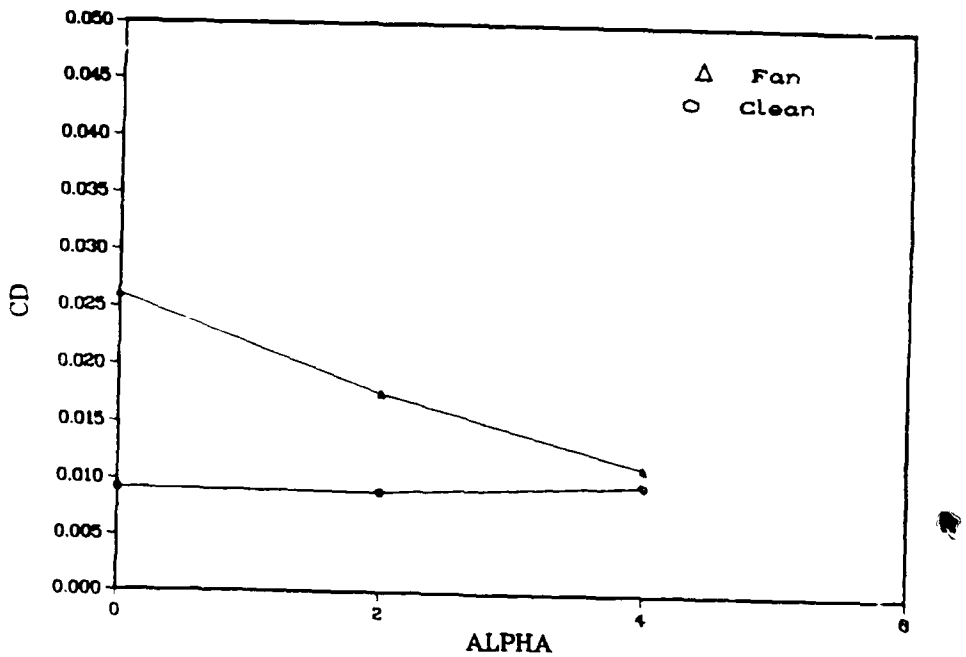


Figure 42. Force Coefficients vs Angle of Attack (Cv=.05, δ=45°)

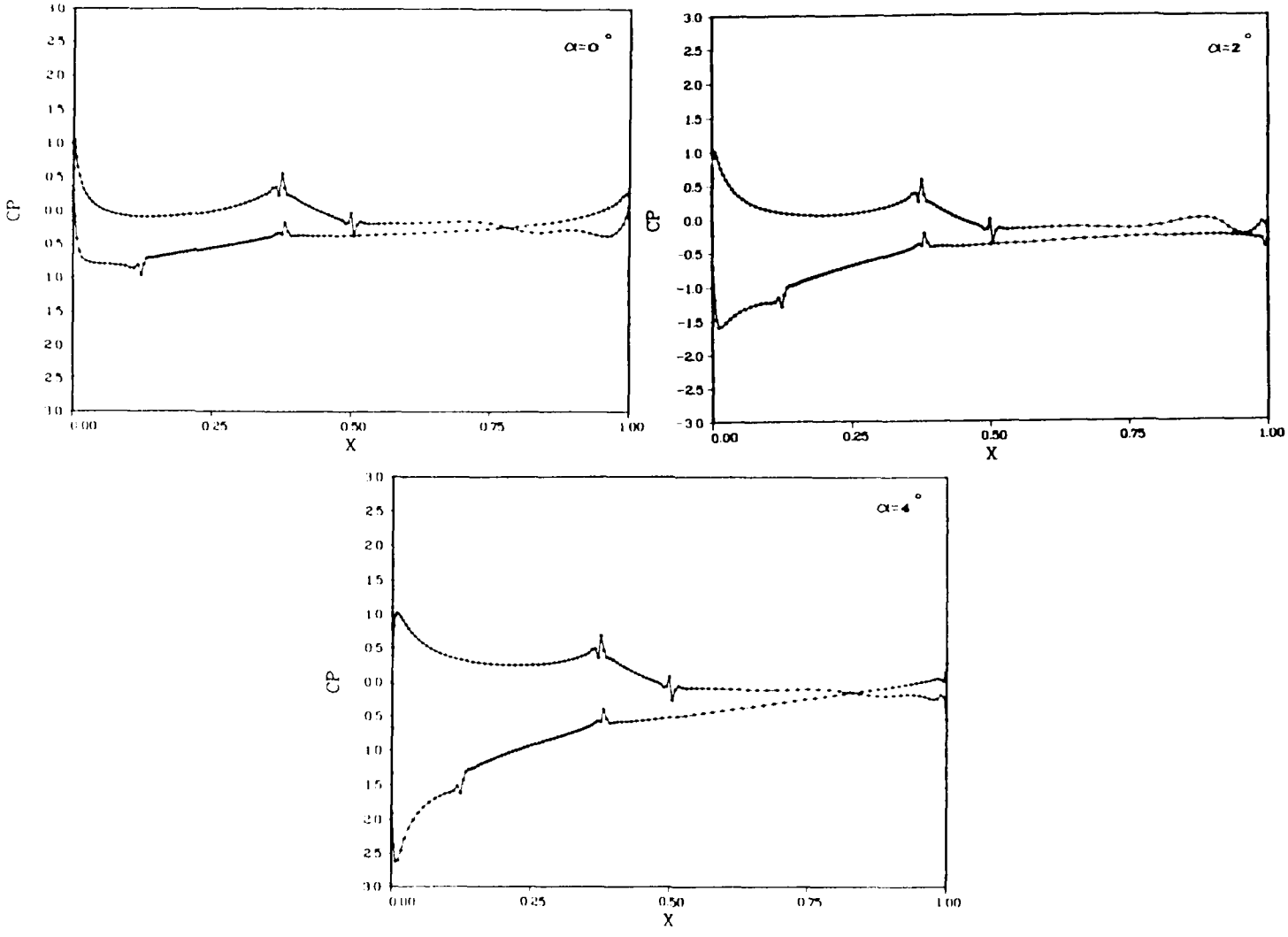


Figure 43. Comparisons of Cp for Different Angles of Attack (Cv=.05, δ=45°)

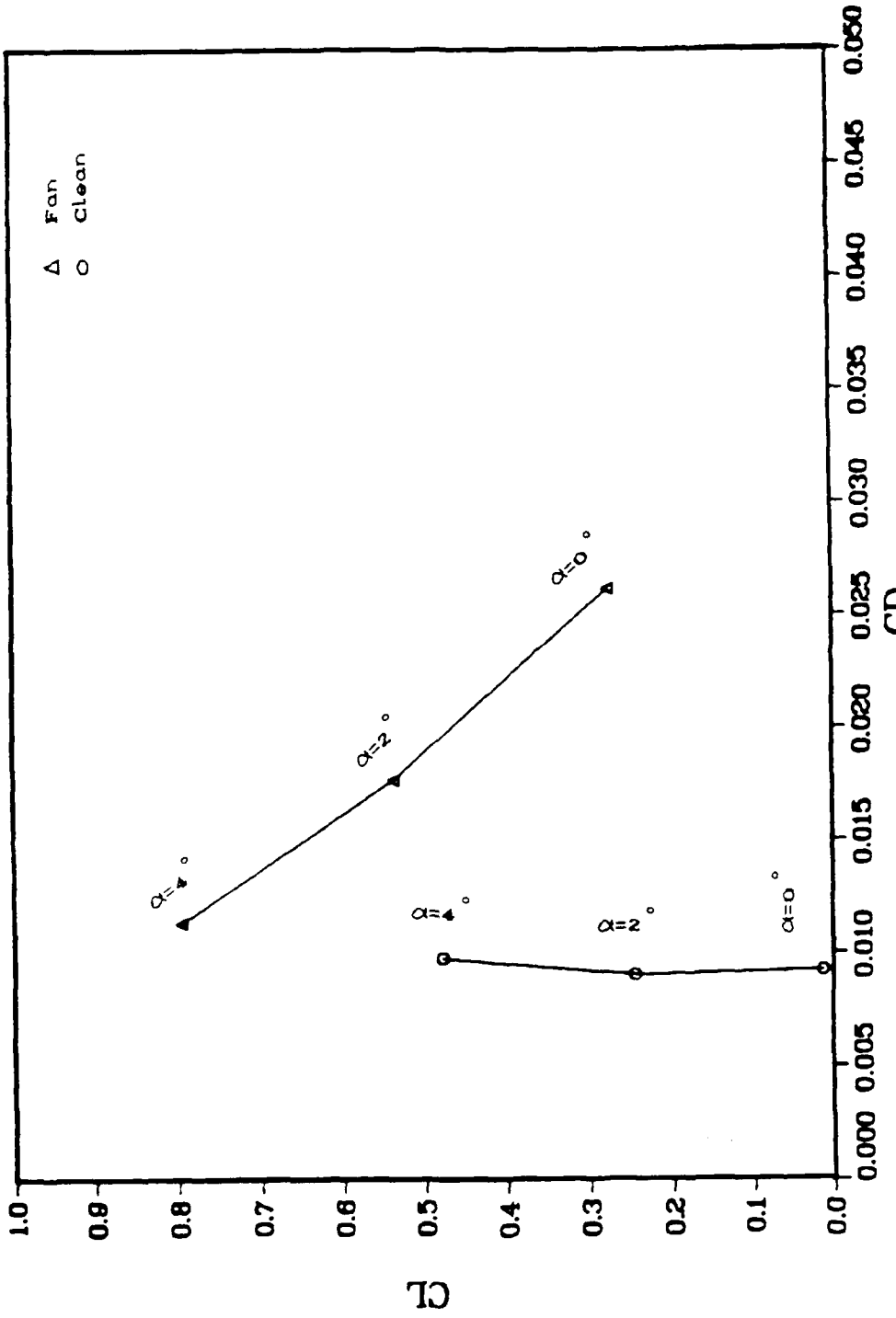


Figure 44. Cl vs Cd for Angle of Attack (Cv=.05, δ=45°)

Number	Cl total	Cl press	Cl viscous	Cl mass
2	.01307	.01321	-.00014	.00000
13	.24582	.24601	-.00018	.00000
16	.47828	.47849	-.00021	.00000
7	.29903	.29773	.00012	.00119
14	.56600	.56494	-.00021	.00126
17	.82326	.82255	-.00062	.00133

code	Cd total	Cd press	Cd viscous	Cd mass
2	.00954	.00164	.00790	.00000
13	.00951	.00196	.00755	.00000
16	.01020	.00831	.00698	.00000
7	.02675	.01320	.01595	-.00240
14	.01763	.00845	.01652	-.00294
17	.01989	-.00049	.01653	-.00227

Table 12. Cl and Cd component comparison vs α

general shape of the clean airfoil but is rotated and shifted up. The lift and drag oscillations contain strong harmonics as discussed above (Figures 32 and 33). At four degrees this is especially pronounced. As shown in Figure 45 the vortex pattern of shedding seems to be only mildly affected by the normal component of the free stream velocity, and the thickness of the shear layer above the surface is reduced by a small amount with increasing angle of attack. The streamlines in Figure 45 indicate the stagnation point at the leading edge is also affected by a small amount.

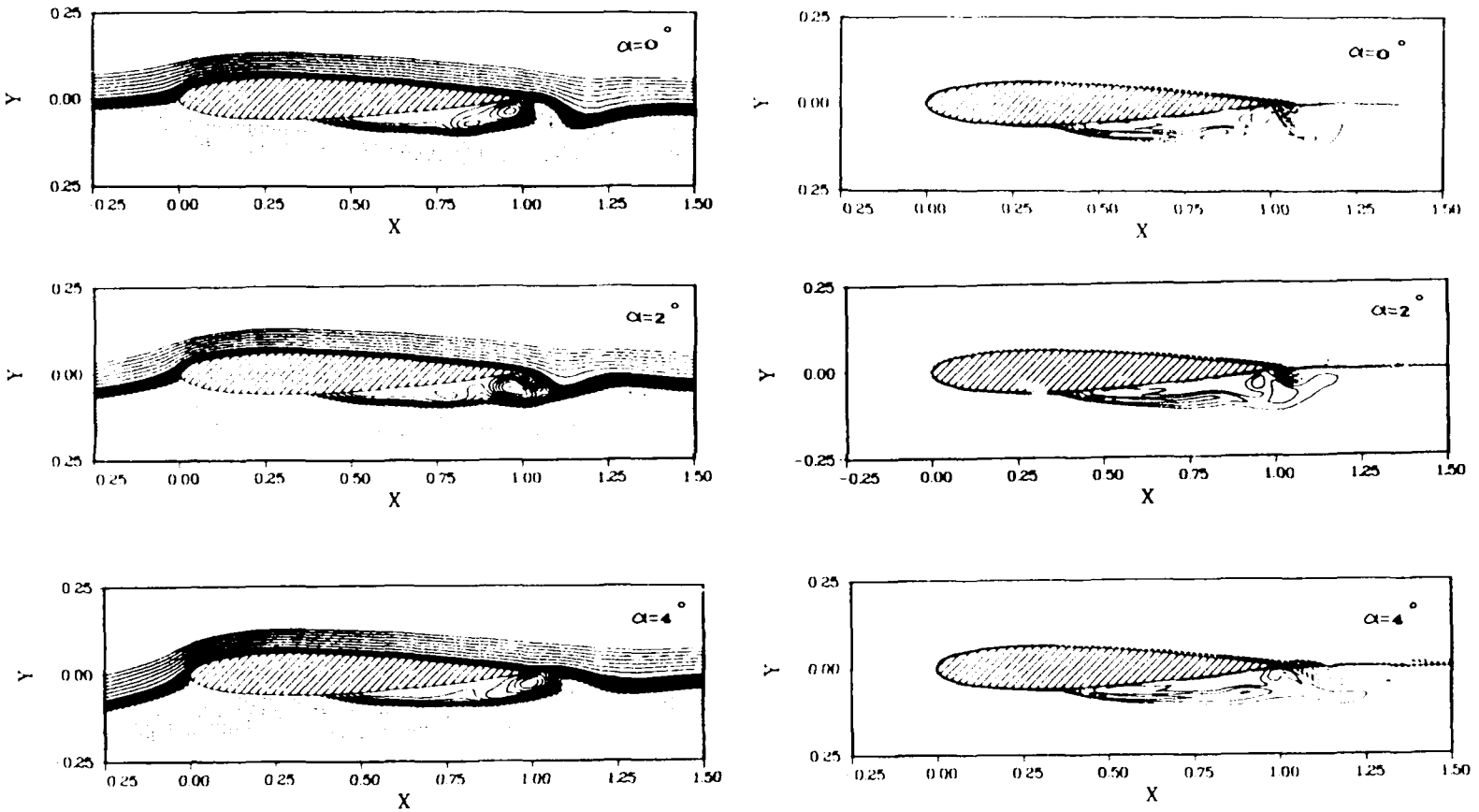


Figure 45. Streamline and Vorticity Comparisons for Different Angles of Attack (Cv=.05, δ=45°)

V Conclusions and Recommendations

The solutions obtained using the implicit compressible Navier-Stokes equations compared favorably with both experimental and other numerical results. For lack of verifying data such comparisons could not be made with solutions obtained from the model with mass flux. However, based on the codes performance the solutions should give a fair approximation to the physical problem.

The results indicate that a fan type device imbedded in an airfoil can produce favorable aerodynamic effects, improved lift performance with minor drag penalties. The best lift performance was achieved at an angle of attack of four degrees. The best drag performance was achieved with ejection angles of 15 degrees where thrust was produced.

Because the model only represented one possible configuration out of a multitude the results simply indicate trends in the "fan-in-wing" behavior. Many more tests both experimental and numerical are needed before firm conclusions can be drawn. At best the fan-like device tested indicates interesting and unusual behavior with possible applications to V/STOL technology.

Experimental tests should be performed to confirm this behavior. Emphasis should be given to high angles of attack in order to determine the maximum C_l and the

airfoil stall effects for a fan-like device.

Appendix A: Non-Dimensional Variables (2:191-193)

The variables used in the Navier-Stokes equations can be written in a non-dimensional form by dividing by L as the length scale, the free stream velocity U_∞ as the velocity scale and ρ_∞ as the mass scale. All other units can be expressed in terms of the three scalar factors, except viscosity and heat. The non-dimensional relationship are as follows (an * represents a non-dimensional quantity)

$$x^* = \frac{x}{L} \quad , \quad y^* = \frac{y}{L} \quad , \quad u^* = \frac{u}{U_\infty} \quad , \quad v^* = \frac{v}{U_\infty}$$

$$T^* = \frac{T}{T_\infty} \quad , \quad \rho^* = \frac{\rho}{\rho_\infty} \quad , \quad P^* = \frac{P}{\rho_\infty U_\infty^2} \quad , \quad e^* = \frac{e}{U_\infty^2}$$

$$Et^* = \frac{Et}{\rho_\infty U_\infty^2} \quad , \quad \mu^* = \frac{\mu}{\mu_\infty} \quad , \quad \varepsilon^* = \frac{\varepsilon}{\mu_\infty} \quad , \quad q^* = \frac{q}{q_\infty}$$

$$t^* = \frac{tU_\infty}{L} \quad , \quad L^* = \frac{L}{L} = 1 \qquad \text{(A-1)}$$

and the non-dimensional Reynolds number is defined as

$$Re = \frac{\rho_\infty U_\infty L}{\mu_\infty} \qquad \text{(A-2)}$$

The temperature, pressure, and energy equations can be expressed in terms of the Mach number

$$M_\infty = \frac{U_\infty}{a_\infty} \qquad \text{(A-3)}$$

where a is the local speed of sound

$$a_\infty = \sqrt{\gamma R T_\infty} \qquad \text{(A-4)}$$

pressure can be written using the equation of state

$$P = \rho R T \qquad \text{(A-5)}$$

$$P^* = \frac{P}{\rho_\infty U_\infty^2} = \frac{P}{\rho_\infty U_\infty^2 \dfrac{\gamma R T_\infty}{a_\infty^2}} = \frac{P}{P_\infty \gamma M_\infty^2} \qquad \text{(A-6)}$$

Temperature and energy can be modified in the same way

$$T^* = \frac{T}{T_\infty} = \frac{\dfrac{P}{\rho R}}{\dfrac{P_\infty}{\rho_\infty R}} = \frac{\rho_\infty P}{\rho P_\infty} = \frac{P^* \gamma M_\infty^2}{\rho^*} \qquad \text{(A-7)}$$

$$E_t^* = \frac{E_t}{\rho_\infty U_\infty^2} = \frac{E_t}{P_\infty \gamma M_\infty^2} \qquad \text{(A-8)}$$

$$e^* = \frac{\frac{E_t}{\rho}}{U_\infty} = \frac{E_t}{\rho\gamma R T_\infty M_\infty^2} = \frac{E_t}{\rho\gamma P_\infty M_\infty^2} = \frac{E_t^*}{\rho^*} \quad \text{(A-9)}$$

An example of how the NS equations are non-dimensionalized is demonstrated on the continuity equation below

$$\frac{\partial \rho^*}{\partial t} + \frac{\partial(\rho^* u^*)}{\partial x} + \frac{\partial(\rho^* v^*)}{\partial y} = 0 \quad \text{(A-10)}$$

$$\frac{\partial\left[\frac{\rho}{\rho_\infty}\right]}{\partial\left[\frac{\rho_\infty U_\infty}{L}\right]} + \frac{\partial\left[\frac{\rho u}{\rho_\infty U_\infty}\right]}{\partial\left[\frac{x}{L}\right]} + \frac{\partial\left[\frac{\rho v}{\rho_\infty U_\infty}\right]}{\partial\left[\frac{y}{L}\right]} = 0 \quad \text{(A-11)}$$

The constants can come out of the derivatives and divide through leaving

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \quad \text{(A-12)}$$

In the momentum and energy equations the constants do not divide out. They form the Reynolds number presented above.

Appendix B: Jacobian Matrices for the Navier-Stokes Equations (35:45-46)

The Jacobian matrices presented in equations (72-73) are defined below. They are obtained by differentiating the vectors E_1, E_2, V_1 and W_2 defined in equations (49-52)

(B-1)

$$A = \begin{bmatrix} 0 & \xi_x & \xi_y & 0 \\ \xi_x\phi - uu & u - (\gamma-2)\xi_x u & \xi_y u - (\gamma-1)\xi_x v & (\gamma-1)\xi_x \\ \xi_y\phi - vu & \xi_x v - (\gamma-1)\xi_y u & u - (\gamma-2)\xi_y v & (\gamma-1)\xi_y \\ (2\phi - \gamma e)u & (\gamma e-\phi)\xi_x-(\gamma-1)uu & (\gamma e-\phi)\xi_y-(\gamma-1)vu & \gamma u \end{bmatrix}$$

(B-2)

$$B = \begin{bmatrix} 0 & \eta_x & \eta_y & 0 \\ \eta_x\phi - u\upsilon & \upsilon - (\gamma-2)\eta_x u & \eta_y u - (\gamma-1)\eta_x v & (\gamma-1)\eta_x \\ \eta_y\phi - v\upsilon & \eta_x v - (\gamma-1)\eta_y u & \upsilon - (\gamma-2)\eta_y v & (\gamma-1)\eta_y \\ (2\phi - \gamma e)\upsilon & (\gamma e-\phi)\eta_x-(\gamma-1)u\upsilon & (\gamma e-\phi)\eta_y-(\gamma-1)v\upsilon & \gamma\upsilon \end{bmatrix}$$

where

$$\phi = 1/2\ (\gamma-1)\ (u^2 + v^2) \qquad \text{(B-3)}$$

and the Jacobian Matrix M and N are defined by

(B-4)

$$M = \frac{1}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 \\ -(b_1 u + b_2 v) & b_1 & b_2 & 0 \\ -(b_2 u + b_3 v) & b_2 & b_3 & 0 \\ -(b_1^2 + 2b_2 uv + b_3 v^2) & b_1 - b_4\gamma(\gamma-1)u & b_2 u + b_3 & b_4\gamma(\gamma-1) \\ +b_4\gamma(\gamma-1)(u^2+v^2-e) & +b_2 v & -b_4\gamma(\gamma-1) & \end{bmatrix}$$

(B-4)

$$N = \frac{1}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 \\ -(d_1 u + d_2 v) & d_1 & d_2 & 0 \\ -(d_2 u + d_3 v) & d_2 & d_3 & 0 \\ -(d_1^2 + 2d_2 uv + d_3 v^2) & d_1 - d_4\gamma(\gamma-1)u & d_2 u + d_3 & d_4\gamma(\gamma-1) \\ +d_4\gamma(\gamma-1)(u^2+v^2-e) & +d_2 v & -d_4\gamma(\gamma-1) & \end{bmatrix}$$

Appendix C: Boundary and Initial Conditions

Boundary Conditions Without Blowing.(35:10:11)

The boundary conditions on the C-grid and on the airfoil surface must be completely specified. In the code used the boundary conditions are explicit as opposed to implicit. This makes it easier to define and alter the conditions; but, requires a separate routine to reinitialize the conditions after each iteration. Referring to the airfoil computational domain shown in Figure 10, the following boundary conditions are *prescribed.*

Free stream conditions are given in non-dimensional form for all the flow variables along the outer boundary ABC.

$$P = P_\infty = \frac{1}{\gamma M_\infty^2} , \quad \rho = \rho_\infty = 1 , \quad \varepsilon = \varepsilon_\infty = 0 \qquad \text{(C-1)}$$

$$u = U_\infty \cos(\alpha) , \quad v = U_\infty \sin(\alpha) , \; U_\infty = 1 \qquad \text{(C-2)}$$

This is the only way in which angle of attack enters the problem and as such requires that restart solutions must be at the same angle of attack. Also, pressure P is

uniquely defined by Mach Number and requires all restart solutions to share the same Mach number.

On the downstream or far field boundaries AD and HC the flow variables are computed by the following relationships

$$P = P_{\infty} \tag{C-1}$$

$$\frac{\partial}{\partial \xi} \begin{bmatrix} \rho \\ \varepsilon \\ u \\ v \end{bmatrix} = 0 \tag{C-3}$$

This allows for compressible non-uniform flow at the boundary and loss in momentum due to skin friction at the airfoil surface.

Along the airfoil surface EFG, the no-slip adiabatic conditions are assumed

$$u = v = 0 \tag{C-4}$$

and the gradient of pressure and temperature normal to the surface is assumed to be zero, where surface density is a function of surface pressure and temperature.

$$\frac{\partial}{\partial \eta} \begin{Bmatrix} P \\ T \end{Bmatrix} = 0 \quad , \quad \rho = \rho(P,T) \tag{C-5}$$

Finally, through the Wake cut ED and EH continuity is

assured by

$$\frac{\partial}{\partial \eta}\begin{bmatrix} P \\ \rho \\ \varepsilon \\ u \\ v \end{bmatrix}_{\substack{\text{Lower} \\ \text{Wake}}} = \frac{\partial}{\partial \eta}\begin{bmatrix} P \\ \rho \\ \varepsilon \\ u \\ v \end{bmatrix}_{\substack{\text{Upper} \\ \text{Wake}}} \tag{C-6}$$

which reduces to a three point averaging

$$\begin{bmatrix} P \\ \rho \\ \varepsilon \\ u \\ v \end{bmatrix}_{\text{Wake}} = \frac{1}{2}\begin{bmatrix} P \\ \rho \\ \varepsilon \\ u \\ v \end{bmatrix}_{\substack{\text{Lower} \\ \text{Wake}}} + \frac{1}{2}\begin{bmatrix} P \\ \rho \\ \varepsilon \\ u \\ v \end{bmatrix}_{\substack{\text{Upper} \\ \text{Wake}}} \tag{C-7}$$

Boundary Conditions for Suction and Blowing (32)

Since the boundary conditions only affect specific sections of the airfoil section, refer to Figure 46, all other boundaries will be calculated as above. On the suction surface (su) P, ρ and T are calculated the same as the solid surface; however, u and v are determined by the constraint that $\bar{V}_{su}$ be normal to the surface and is given by the slope ,θ at the surface

$$\theta = \frac{dy}{dx} \tag{C-8}$$

$$u = V_{su}\sin(\theta) \quad , \quad v = -V_{su}\cos(\theta) \tag{C-9}$$

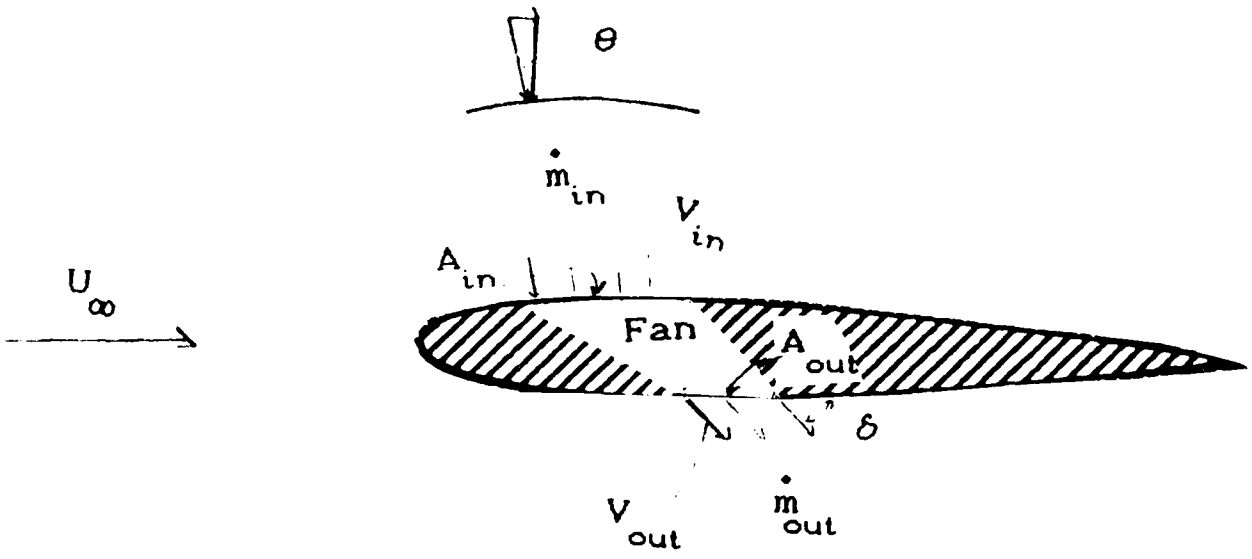


Figure 46. Boundary Conditions

where the magnitude V_{su} is a constant equal to some fraction of the free stream velocity

$$V_{su} = CvU_{\infty} = Cv \ , \quad U_{\infty} = 1 \qquad \text{(C-10)}$$

In order to prescribe conditions on the lower blowing surface (bl) the mass flow $\dot{m}$ must be determined and is done in the following manner (12)

$$\dot{m}_{su} = \dot{m}_{bl} = \dot{m} = \rho AV \qquad \text{(C-11)}$$

where A is the area of the opening and V is the velocity magnitude normal to the surface. On the upper surface ρ is averaged and changes with each iteration, A is assumed to be delta X and for this analysis was picked to be 0.25 or 25 percent of the airfoil cord and V is a known constant Cv. On the lower surface P is determined as above and ρ is fixed and assumed to be compressed by the action of the fan and is how work done by the fan is introduced into the model (11:4-5)

$$de = dq - Pd(1/\rho) = Pd(1/\rho) \qquad \text{(C-12)}$$

for no heat transfer, dq = 0

and

$$\rho_{bl} = Cr\ avg\ (\rho)_{su} \qquad (C-13)$$

where Cr is a constant and equal to 1.1 for this analysis.

Using the above relationships the magnitude of the normal velocity on the lower surface, V_{bl} can be determined. If vanes are assumed on the lower surface and A_{bl} is assumed to be equal to dx, where dx is picked to be 0.125 for this analysis, then the boundary conditions simplify to the following equations

$$\rho_{\underset{su}{avg}} Cr A_{bl} V_{bl} = \rho_{\underset{bl}{avg}} A_{su} Cv \qquad (C-14)$$

$$v = -V_{bl} = \frac{Ca\ Cv}{Cr}\ ,\ u = V_{bl}\tan(\delta) \qquad (C-15)$$

$$Ca = \frac{A_{su}}{A_{bl}} \qquad (C-16)$$

where δ is the angle of the vane with respect to the airfoil cord and is equal to 90 degrees for blowing normal to the lower surface area A_{bl}.

Initial Conditions

For this code the first run made for a particular angle of a attack and Mach Number assumes a Uniform flow field, where the non-dimensional flow variables are given by the following relationships

$$P = P_{\infty} = \frac{1}{\gamma M_{\infty}^2} \quad , \quad \rho = \rho_{\infty} = 1 \qquad \text{(C-1)}$$

$$u = U_{\infty}\cos(\alpha) \quad , \quad v = U_{\infty}\sin(\alpha) \quad , \; U_{\infty} = 1 \qquad \text{(C-2)}$$

An initial run of one hundred iterations with turbulence turned off is made in order to develop the flow and a solution or restart file is saved. Additional runs can then be made using the restart file as long as the angle of attack, Mach Number and grid are not changed. The variables which can be manipulated include the transition points, Reynolds Number, Cv, Cr, δ and the damping coefficients. A considerable number of iterations can be saved by using a near convergent solution as a restart for a new problem within the above Constraints.

Appendix D: Derivation of the Force Coefficients Cl and Cd

The Calculation of lift and drag on an airfoil surface with non-zero surface velocity and mass flux can be accomplished through a special application of the NS equations. Starting with the linear momentum equation in integral form.(41:)

$$\frac{D}{Dt}\iiint(\rho\bar{V})\,d\upsilon = \iiint(\bar{f}_b + \nabla\cdot\underline{\underline{\sigma}})\,d\upsilon \qquad (D-1)$$

By assuming a constant control volume with respect to time and applying the material derivative equation. The above equation the reduces to the following form.

$$\iiint\frac{\partial(\rho\bar{V})}{\partial t}d\upsilon = \iiint(-(\rho\bar{V}\cdot\bar{\nabla})\bar{V} + \bar{f}_b + \bar{\nabla}\cdot\underline{\underline{\sigma}})\,d\upsilon \qquad (D-2)$$

$$\iiint\left[\frac{\partial(\rho\bar{V})}{\partial t} - \bar{f}_b\right]d\upsilon \qquad (D-3)$$

$$= \iint(-(\rho\bar{V}\bar{V})\cdot\hat{n} + \underline{\underline{\sigma}}\cdot\hat{n})\,ds$$

Where ds represents a spatial increment on the airfoil surface. In 2-d cartesian coordinates the components reduce to (42)

$$\hat{n} = \frac{-y_\xi \hat{i} + x_\xi \hat{j}}{\sqrt{x_\xi^2 + y_\xi^2}} = \frac{-y_\xi \hat{i} + x_\xi \hat{j}}{ds} \qquad (D-4)$$

$$\bar{V} = u\hat{i} \quad v\hat{j} \qquad (D-5)$$

$$\iiint \frac{\partial(\rho\bar{V})}{\partial t} d\upsilon = \iiint \left[\rho\frac{\partial\bar{V}}{\partial t} + \bar{V}\frac{\partial\rho}{\partial t}\right] d\upsilon = 0 \qquad (D-6)$$

$$\rho\bar{V}\bar{V}\cdot n = \rho(u\hat{i} + v\hat{j})\left(\frac{-uy_\xi + vx_\xi}{ds}\right) \qquad (D-7)$$

$$\underline{\underline{\sigma}}\cdot\hat{n} = \left(Py_\xi - \tau_{xx}y_\xi + \tau_{xy}x_\xi\right)\hat{i} \qquad (D-8)$$

$$+ \left(-Px_\xi + \tau_{yy}x_\xi - \tau_{yx}y_\xi\right)\hat{j}$$

for velocity and density on the airfoil boundary equal to either a constant value or zero.

$$\iiint \bar{f}_b \, d\upsilon = \bar{F}_b = F_b\hat{i} + F_b\hat{j} \qquad (D-9)$$

$$= -F_x - F_y$$

Putting it all together and summing over the airfoil surface $\iint(\,)ds = \Sigma\, ds$ the equations reduce to

$$Fx = -\sum \rho u \left(\frac{-uy_\xi + vx_\xi}{ds} \right) + \sum \left(-Px_\xi + \tau_{yy} x_\xi - \tau_{yx} y_\xi \right) \quad \text{(D-10)}$$

$$Fy = -\sum \rho v \left(\frac{-uy_\xi + vx_\xi}{ds} \right) + \sum \left(-Px_\xi + \tau_{yy} x_\xi - \tau_{yx} y_\xi \right) \quad \text{(D-11)}$$

and by using Stokes approximation $\lambda = -2/3\mu$ the non-dimensional shear stress terms become

$$\tau_{xx} = \frac{1}{Re}\mu \left(\frac{4}{3} \frac{\partial u}{\partial x} - \frac{2}{3} \frac{\partial v}{\partial y} \right) \quad \text{(D-12)}$$

$$\tau_{yy} = \frac{1}{Re}\mu \left(\frac{4}{3} \frac{\partial v}{\partial y} - \frac{2}{3} \frac{\partial u}{\partial x} \right) \quad \text{(D-13)}$$

$$\tau_{xy} = \tau_{yx} = \frac{1}{Re}\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \quad \text{(D-14)}$$

Lift and drag can be related to the body forces and the angle attack by the following relationships

$$Lift = -Fx \sin(\alpha) + Fy \cos(\alpha) \quad \text{(D-15)}$$

$$Drag = Fx \cos(\alpha) + Fy \sin(\alpha) \quad \text{(D-16)}$$

and the coefficients are

$$Cl = \frac{Lift}{\frac{1}{2}\rho_\infty U_\infty^2} \qquad Cd = \frac{Drag}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{D-17}$$

and for non-dimensional variables $\rho_\infty = U_\infty = 1$ and equation (D-17) reduces to

$$Cl = 2\ Lift \quad , \qquad Cd = 2\ Drag \tag{D-18}$$

Appendix E: Computer Resource Requirements

The model simulations were accomplished by using the NS code as discussed in section III and in appendix F. The code was run on the Cray XMP computer located at Wright-Patterson Air Force Base. Table 13 compares the memory and CPU requirements for the original and modified codes.

	Grid			
Version	IL	JL	Memory	CPU/100
Unmodified	199	50	780,800	31.157
Modfied	199	50	611,328	29.991
Modified	299	100	1,397,760	71.202

Table 13. Computer Resource Comparisons

The memory modifications to the code reduced the memory requirements by 27.7 percent and the CPU requirements by 4.1 percent. The memory improvements were necessary in order to decrease turn around time for the larger grid used in the analysis. The maximum available core memory on the Cray is 1,860,000 words. The unmodified code memory requirements for the larger grid would have been close to this value. The main reason the improvement was needed is based on how the priority system on the Cray works. Each job with the same priority is rotated in the queue and, if memory is available, is allowed to run for a

set number of CPUs. If no memory is available, then the job is passed over. This pattern continues until the job is completed.

For jobs requiring a lot of memory (50% core), the availability of memory goes down. Jobs with higher priority maintain their memory and it is not available for jobs of a lower priority. However, lower priority jobs requiring less memory than that being used by the higher priority jobs can run. For this reason, the turn around time for a job with the larger grid at regular day priority was approximately 1.4 minutes per 100 iterations, while at an overnight priority it was between 1 hour and 3 days.

Appendix F: Navier-Stokes, Fortran Listing (36)

The attached fortran listing is the modified version of the code used in this work.The original code was supplied by the Air Force Wright Aerodynamics Laboratories, Flight Dynamics Laboratory, Computational Fluid Dynamics Group. The Code was developed by Dr. Miguel Visbal. It uses the Beam-Warming approximate factorization algorithm to solve the two-dimensional, mass-averaged, compressible Navier-Stokes equations for viscous, unsteady flows.

Modifications, made by the author, for this study involved the boundary conditions for the fan, the turbulence model for blowing effects, the force and residual calculations, input/output and loop structure to reduce memory requirements (see appendix E). The Code was run on the Cray XMP Computer.

```
C ------------------------------------------------------------------
C  VECTORIZED NAVIER-STOKES CODE        --      BEAM-WARMING  ALGORITHM
C  CONFIGURATION :  AIRFOIL , C-GRID;   AUG. 1985
C BY MIGUEL R. VISBAL
C PROGRAM MODIFIED TO INCLUDE INJECTION AND EJECTION ON THE AIRFOIL
C SURFACE. THIS ALLOWED SIMULATION OF A FAN IN THE WING CONCEPT. CHANGES
C WERE MADE IN THE SUBROUTINE BNDRY. THESE CHANGES WERE MADE BY
C CAPT PAUL D BOYLES IN SUPPORT OF AN AFIT THESIS, AUG 1988
C ------------------------------------------------------------------
C ****************  I M P O R T A N T  **********************
C BEFORE USING THIS PROGRAM :
C MODIFY DIMENSION STATEMENTS
C CHECK BOUNDARY CONDITIONS SUBROUTINE
C CHECK INPUT DATA
C ***********************************************
C
C THIS PROGRAM SOLVES THE COMPRESSIBLE NAVIER-STOKES EQUATIONS
C IN ARBITRARY TWO-DIMENSIONAL DOMAINS USING BODY-FITTED (TIME-
C INVARIANT) CURVILINEAR COORDINATES AND THE IMPLICIT FACTORED
C SCHEME OF BEAM & WARMING. ************
C TIME DIFFERENCING : EULER IMPLICIT
C SPATIAL DIFFERENCING : SECOND ORDER CENTERED DIFFERENCES
C IN THIS PROGRAM THE SPACE INCREMENTS OF THE TRANSFORMED
C VARIABLES XI AND ETA ARE BOTH ASSUMED EQUAL TO ONE ( I.E.,
C DEL(XI)=DEL(ETA)=1.0).
C
C ------------------------------------------------------------------
C    LIST OF  SYMBOLS
C ------------------------------------------------------------------
C X,Y : CARTESIAN COORDINATES
C XI,ETA : TRANSFORMED COORDINATES
C XJAC : JACOBIAN OF INVERSE TRANSFORMATION:D(X,Y)/D(XI,ETA).
C XXI,YXI,XETA,YETA: TRANSFORMATION DERIVATIVES
C U,V : VELOCITY COMPONENTS
C RHO : DENSITY
C P   : PRESSURE . IS NON DIMENSIONALIZED BY 1/(GMX*MACH#**2)
C XMU : VISCOSITY
C RE  : REYNOLDS NO.
C XM1 : MACH NO.
C PR  : PRANDTL NO.
C S1  : NON-DIMENSIONAL PARAMETER FOR SUTHERLAND'S LAW
C PRT : TURBULENT PRANDTL NO.
C EDDY : TURBULENT EDDY VISCOSITY
C S(I,J): DISTANCE ALONG THE XI-LINES FROM ETA=0.
C NCONV : THE CONVERGENCE TEST IS APPLIED EVERY NCONV STEPS
C IL   : NO. OF POINTS IN XI-DIRECTON
C JL   : NO. OF POINTS IN ETA-DIRECTION
C WE,WI : DAMPING COEFFICIENTS
C INMAX : MAX. NO. OF TIME STEPS ALLOWED
C JLAST : LOCATION IN THE  GRID ABOVE WHICH TURB IS NOT APPLIED
C ILE   : I LOCATION OF THE LEADING EDGE
C ITEU  : I LOCATION OF THE TRAILING EDGE ON THE UPPER SURFACE
C ITEL  : I LOCATION OF THE TRAILING EDGE ON THE LOWER SURFACE
C ITRSU : I LOCATION OF TRANSITION ON THE UPPER SURFACE
C ITRSL : I LOCATION OF TRANSITION ON THE LOWER SURFACE
C IWKST : I LOCATION WHERE MODIFIED TRANSITION IN THE WAKE STARTS
C IWKE  : I LOCATION WHERE MODIFIED TRANSITION IN THE WAKE ENDS
C
C ------------------------------------------------------------------
C                      MAIN   PROGRAM
C ------------------------------------------------------------------

      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TW,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SIMA,COSA,PRI,PRTI,REI,XL,XL1,XLN,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DAMP/ WE,WI,WPE,WPI,ISPECT
      COMMON/TURB1/ITRSU,ITRSL,IWKST,IWKE,JLAST,
     -              XRELAX
      COMMON/TURB2/XMUTI(JM),XMUTO(JM),D(JM),W(2*JM),F(2*JM),
     -             GAMMA(IM)
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/NORM/ UNORM,VNORM
      COMMON/BND/ NBMD,XMREF,TREF,TINF,CV,CR,CT,DEL,L1,L2,L3,L4
C
C ------------------------------------------------------------------
C                      INPUT DATA
C ------------------------------------------------------------------
C
      OPEN(UNIT=5,FILE='DATA',STATUS='OLD')
      OPEN(UNIT=1,FILE='GRID',STATUS='OLD',
      OPEN(UNIT=6,FILE='OUTPUT',STATUS='NEW')
      OPEN(UNIT=2,FILE='SOLN',STATUS='NEW')
      OPEN(UNIT=3,FILE='STUFF',STATUS='NEW')

      READ (5,510) NSTEP1
      READ (5,520) TAU
      READ (5,510) IL
      READ (5,510) JL
      READ (5,510) ILE
      READ (5,510) ITEL
      READ (5,510) ITEU
  510 FORMAT(I10)
      READ (5,520) XM1
      READ (5,520) RE
      READ (5,520) TW
      READ (5,520) S1
      READ (5,520) ALFA
  520 FORMAT (F15.10)
      READ (5,510) NTURB
      READ (5,510) IFREQ
      READ (5,510) ITRSU
      READ (5,510) ITRSL
      READ (5,510) IWKST
      READ (5,510) IWKE
      READ (5,510) JLAST
      READ (5,520) XRELAX
      READ (5,510) NDTAU
      READ (5,520) CFL
      READ (5,520) BETA
      READ (5,520) DTVIS
      READ (5,510) ISPECT
      READ (5,520) WE
      READ (5,520) WI
```

```
      READ (5,520) MPE
      READ (5,520) MPI
      READ (5,520) EDST
      READ (5,510) INMAX
      READ (5,510) NCONV
C  IF RESTART THEN IRESTART = 1 ELSE = 0
      READ (5,510) IREST
C  IPLOT = 1 TURNS PRINT ROUTINES ON FOR DEBUGGING
      READ(5,510) IPLOT
      READ(5,510) IPLOT1
C FAN INFORMATION
      READ(5,510) NBND
      READ(5,520) XMREF
      READ(5,520) CV
      READ(5,520) CR
      READ(5,520) CT
      READ(5,520) DEL
      READ(5,510) L1
      READ(5,510) L2
      READ(5,510) L3
      READ(5,510) L4
C
C ------------------------------------------------------------------
C                    CONSTANTS FOR AIR
C ------------------------------------------------------------------
      GAM=1.40
      PR=0.720
      PRT=0.90
      PRES = 1. / ( GAM*XM1*XM1 )
C     RHO = 1
      TINF  = 1
      TREF = XM1*XM1/(XMREF*XMREF)
      ICHECK = 0
C
C ------------------------------------------------------------------
C                       INITIALIZATION
C ------------------------------------------------------------------
C
      CALL INITIA
C
      UINF = COS(ANGLE)
      VINF = SIN(ANGLE)
C ------------------------------------------------------------------
C      PRINT OUT THE INPUT DATA
C ------------------------------------------------------------------
C
      WRITE (6,*)'C'
      WRITE (6,*)'C'
      WRITE (6,202)
      WRITE (6,203)
      WRITE (6,204) IL,JL,ILE,ITEL,ITEU
      WRITE (6,205) NDTAU,CFL,BETA,DTVIS
      WRITE (6,206) XM1,RE,S1,ALFA
      WRITE (6,213) ITRSU,ITRSL,IMKST,IMKE,
     -              JLAST,XRELAX
      WRITE (6,214) NTURB,IFREQ
      WRITE (6,208) ME,MI,MPE,MPI,ISPECT
      WRITE (6,*)' RESTART (Y/N, 1/0) => ',IREST
      WRITE (6,*)' '
      IF ( NBND .EQ. 2 ) THEN
        WRITE(6,*)'      SUCTION ONLY       '
      END IF
      IF ( NBND .EQ. 3 ) THEN
        WRITE(6,*)'      BLOWING ONLY       '
      END IF
      IF ( NBND .EQ. 0 ) THEN
        WRITE(6,*)'      FAN IS TURNED OFF '
        GOTO 111
      END IF
      WRITE (6,*)'       FAN INFORMATION '
      WRITE (6,209) L1,L2,L3,L4,DEL
      WRITE (6,210) CV,CR,CT
      WRITE (6,215) XMREF,XM1,TREF,TINF
C
C ------------------------------------------------------------------
C       INPUT RESTART SOLUTION (I.E. X,Y,U,V,P,RHO,EDDY)
C ------------------------------------------------------------------
C
 111  REWIND 1
      DO 281 J=1,JL
      DO 281 I=1,IL
      IF ( IREST .EQ. 0 ) GOTO 382
      READ (1,1000) X(I,J),Y(I,J),U(I,J),V(I,J),
     -              P(I,J),RHO(I,J),EDDY(I,J)
      GOTO 281
 382  READ(1,1001) X(I,J),Y(I,J)
      P(I,J) = PRES
      RHO(I,J) = 1.0
      U(I,J) = UINF
      V(I,J) = VINF
      EDDY(I,J) = 0.0
      IF ( J .LE. JLAST ) EDDY(I,J) = EDST
 281  CONTINUE
1000  FORMAT (1X,7E16.8)
1001  FORMAT (2E15.8)
C
C ------------------------------------------------------------------
C                  COMPUTE TRANSFORMATION METRICS
C ------------------------------------------------------------------
C
      CALL METRIC
C
C INITIALIZE SPECTRAL RADIUS SPECT(I,J)
      DO 282 J=1,JL
      DO 282 I=1,IL
      SPECT(I,J)=XJAC(I,J)
 282  CONTINUE
C
C ------------------------------------------------------------------
C                  COMPUTE TIME STEP (DTAU)
C ------------------------------------------------------------------
C  CALCULATE MOLECULAR VISCOSITY XMU(I,J)
      CALL CXMU(ICHECK)
      CALL TMSTEP
      WRITE(6,211) DTMIN,DTMAX
C
C ------------------------------------------------------------------
C                  PREPARE FOR FIRST ITERATION
C ------------------------------------------------------------------
C
 608  NSTEP=NSTEPI
      INDEX=0
```

```
      ICONV=0
      IDT=0
      ITURB=0
C
C COMPUTE TURBULENCE TRANSITION FACTOR GAMMA(J)
      CALL TRANSI
C
C-----------------------------------------------------------------------
C                     ADVANCE TIME STEP
C-----------------------------------------------------------------------
C
  100 NSTEP=NSTEP+1
      INDEX=INDEX+1
      ICONV=ICONV+1
      IF(ICONV.EQ.NCONV) ISWTCH=1
      IDT=IDT+1
      TAU=TAU+DTMIN
      ITURB=ITURB+1
      IF (INDEX.GT.INMAX) GO TO 200
C
C     COMPUTE NEW DTAU IF DESIRED
      IF (IDT.LT.NDTAU)  GO TO 400
      IDT=0
      CALL TMSTEP
      WRITE(6,212) NSTEP,DTMIN,DTMAX
  400 CONTINUE
C
C-----------------------------------------------------------------------
C             S T E P 1 :   XI - SWEEP
C-----------------------------------------------------------------------
C
      CALL STEPX (ISWTCH,DRMAX,RMSDR)
C
C-----------------------------------------------------------------------
C             S T E P 2 :   ETA - SWEEP
C-----------------------------------------------------------------------
C
      CALL STEPY
C
C-----------------------------------------------------------------------
C                     UPDATE VARIABLES AT INTERIOR POINTS
C-----------------------------------------------------------------------
C
      CALL CONVRT
C
      DO 25 K=1,4
      DO 25 J=2,JL1
      DO 25 I=2,IL1
      TEMP(I,J,K)=TEMP(I,J,K)+RHS(I,J,K)
   25 CONTINUE
C
      CALL UCONVRT
C
C-----------------------------------------------------------------------
C                 IMPLEMENT  BOUNDARY CONDITIONS
C-----------------------------------------------------------------------
C
      CALL BNDRY
C
      CALL CXMU(ICHECK)
      IF ( ICHECK .EQ. 1 ) GOTO 200
C
C-----------------------------------------------------------------------
C           UPDATE   TURBULENT EDDY VISCOSITY
C-----------------------------------------------------------------------
C
      IF (ITRSU.GT.IL) GO TO 256
      IF (INDEX.LT.NTURB) GO TO 256
      IF (ITURB.LT.IFREQ) GO TO 256
C
      ITURB=0
C
C     IF (INDEX.EQ.INMAX) IPLOT=1
C     IF (INDEX.EQ.INMAX' IPLOT1=1
      CALL TURB (IPLOT,IPLOT1)
C
  256 CONTINUE
C
C-----------------------------------------------------------------------
C                CONVERGENCE  CRITERIA
C-----------------------------------------------------------------------
C
      IF (ICONV.LT.NCONV) GO TO 100
      ICONV=0
C
      CALL MONITR (NSTEP,TAU,DRMAX,RMSDR)
C
      GO TO 100
C
C-----------------------------------------------------------------------
C                     SAVE CURRENT SOLUTION.
C-----------------------------------------------------------------------
C
  200 CONTINUE
C
      REWIND 2
      DO 257 J=1,JL
      DO 257 I=1,IL
      WRITE (2,1000) X(I,J),Y(I,J),U(I,J),V(I,J),
     -               P(I,J),RHO(I,J),EDDY(I,J)
  257 CONTINUE
C
C           PRINT OUT  SOLUTION
C
      CALL OUTPUT
C
C-----------------------------------------------------------------------
C                           FORMATS
C-----------------------------------------------------------------------
C
  202 FORMAT (1H1,30X,'**** BEAM-WARMING METHOD ****',///)
  203 FORMAT (30X,'     NACA 0012 AIRFOIL          ',///)
  204 FORMAT (10X,'IL=',I3,3X,'JL=',I3,3X,
     -        'ILE=',I3,3X,'ITEL=',I3,3X,'ITEU=',I3,//)
  205 FORMAT(15X,'NDTAU=',I5,5X,'CFL=',F12.3,5X,'BETA=',F12.3,
     -   5X,'DTVIS=',F12.5,/)
  206 FORMAT (1X,'REFERENCE MACH NO.=',F12.5,3X,
     1   'REYNOLDS NO.=',F12.1,3X,'VISCOSITY PARAMETER=',F12.5,
     1   3X,'ANGLE OF ATTACK.=',F12.6,//)
  208 FORMAT (15X,'DAMPING PARAMETERS',5X,'ME=',F12.5,
     -   5X,'MI=',F12.5,5X,'MPE=',F12.5,5X,
     -   'MPI=',F12.5,5X,'ISPECT=',I5,//)
  211 FORMAT(10X,'DTMIN=',E12.5,5X,'DTMAX=',E12.5,/)
  215 FORMAT.2X,'MREF=  ',F8.3,5X,'MINF=  ',F8.3,5X,'TREF=  '.F8.3,5X,
```

```
     -          'TINF=  ',F8.3,/)
  210 FORMAT (2X,'COEFECIENTS :  CV=  ',F8.2,5X,'CR=  ',F8.2,5X,
     -        'CT=',F8.2,/)
  209 FORMAT(2X,'FAN PARAMETERS : L1=',I5,2X,
     -  'L2=',I5,2X,'L3=',I5,2X,'L4=',I5,2X,'DEL=',F8.2,2X,/)
  212 FORMAT (10X,'TIME STEP=',I5,5X,'DTMIN=',E12.5,5X,
     -        'DTMAX=',E12.5,/)
  213 FORMAT(2X,'TURBULENCE PARAMETERS : ITRSU=',I5,2X,
     - 'ITRSL=',I5,2X,'IWKST=',I5,2X,'IWKE=',I5,2X,'JLAST=',I5,2X,
     - 'XRELAX=',F12.5,/)
  214 FORMAT(2X,'TURBULENCE SWITCHED ON AFTER TIME STEP NO.=',
     - I5,2X,'AND UPDATED EVERY',I5,2X,'TIME STEPS',/)
C
C ------------------------------------------------------------------
C
  300  CLOSE(5)
       CLOSE(6)
       CLOSE(1)
       CLOSE(2)
       CLOSE(3)

       STOP
       END

C SUBROUTINE "INITIA" COMPUTES COMMON PARAMETERS
C
       SUBROUTINE INITIA
C
       PARAMETER(IM=299,JM=100)
       COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
      1XMU(IM,JM),EDDY(IM,JM)
       COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
      1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
       COMMON /P1/ IL,JL,ILE,ITEL,ITEU
       COMMON /P2/ GAM,PR,PRT,XM1,RE,TM,S1,ALFA,ANGLE
       COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
      -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
      -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
       COMMON/DUMMY/ TEMP(IM,JM,4)
       COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
       COMMON/BND/ NBND,XMREF,TREF,TINF,CV,CR,CT,DEL,L1,L2,L3,L4
C
C ---------- COMPUTE  PARAMETERS ----------
       IL1=IL-1
       IL2=IL-2
       IL3=IL-3
       JL1=JL-1
       JL2=JL-2
       JL3=JL-3
       GAM1=GAM-1.0
       GAM2=GAM-2.0
       GAM3=GAM-3.0
       XM12=XM1**2
       GXM=GAM*XM12
       S1P=1.+S1
       REI=1.0/RE
       PI=3.141592654
       RAD=PI/180.
       ANGLE=RAD*ALFA
       DEL = RAD*DEL
       GAM1I=1.0/GAM1
-------------------------------------------------------------------
       PRI=1.0/PR
       GPR=GAM/PR
       PRTI=1.0/PRT
       XL=-2.0/3.0
       XLM=4.0/3.0
       XL1=1.0/3.0
       ILEP1=ILE+1
       ILEM1=ILE-1
       ITELP1=ITEL+1
       ITELM1=ITEL-1
       ITEUP1=ITEU+1
       ITEUM1=ITEU-1
       PINF=1./GXM
       SINA=SIN(ANGLE)
       COSA=COS(ANGLE)
C
C INITIALIZE PSMU(I,J) FOR PRESSURE DAMPING TERM
       DO 1 J=1,JL
       DO 1 I=1,IL
       PSMU(I,J)=0.0
     1 CONTINUE
C
       RETURN
       END
C SUBROUTINE "METRIC" EVALUATES THE COORDINATE TRANSFORMATION
C DERIVATIVES AND THE JACOBIAN.
C NOTE: DXI=DETA=1.0
C
       SUBROUTINE METRIC
       PARAMETER(IM=299,JM=100)
       COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
      -  XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
       COMMON /P1/ IL,JL,ILE,ITEL,ITEU
       COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
      -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
      -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
C
C COMPUTE DISTANCE SS1(I) ALONG ETA=0.0
       SS1(1)=0.0
       DO 1 I=2,IL
       DS=SQRT((X(I,1)-X(I-1,1))**2+(Y(I,1)-Y(I-1,1))**2)
       SS1(I)=SS1(I-1)+DS
     1 CONTINUE
C COMPUTE DISTACE S(I,J) ALONG THE XI-LINES
       DO 2 I=1,IL
       S(I,1)=0.0
       DO 2 J=2,JL
       DS=SQRT((X(I,J)-X(I,J-1))**2+(Y(I,J)-Y(I,J-1))**2)
       S(I,J)=S(I,J-1)+DS
     2 CONTINUE
C
C EVALUATE TRANSFORMATION METRICS
C INTERIOR POINTS
       DO 7 J=2,JL1
       DO 7 I=2,IL1
       XXI(I,J)=.5*(X(I+1,J)-X(I-1,J))
       YXI(I,J)=.5*(Y(I+1,J)-Y(I-1,J))
       XETA(I,J)=.5*(X(I,J+1)-X(I,J-1))
       YETA(I,J)=.5*(Y(I,J+1)-Y(I,J-1))
     7 CONTINUE
C XI=0.0
```

```
      DO 8 J=2,JL1
      XXI(1,J)=.5*(-X(3,J)+4.*X(2,J)-3.*X(1,J))
      YXI(1,J)=.5*(-Y(3,J)+4.*Y(2,J)-3.*Y(1,J))
      XETA(1,J)=.5*(X(1,J+1)-X(1,J-1))
      YETA(1,J)=.5*(Y(1,J+1)-Y(1,J-1))
    8 CONTINUE
C
C XI=XIMAX
      DO 9 J=2,JL1
      XXI(IL,J)=-.5*(-X(IL2,J)+4.*X(IL1,J)-3.*X(IL,J))
      YXI(IL,J)=-.5*(-Y(IL2,J)+4.*Y(IL1,J)-3.*Y(IL,J))
      XETA(IL,J)=.5*(X(IL,J+1)-X(IL,J-1))
      YETA(IL,J)=.5*(Y(IL,J+1)-Y(IL,J-1))
    9 CONTINUE
C
C ETA=0.0
      DO 10 I=2,IL1
      XXI(I,1)=.5*(X(I+1,1)-X(I-1,1))
      YXI(I,1)=.5*(Y(I+1,1)-Y(I-1,1))
      XETA(I,1)=.5*(-X(I,3)+4.*X(I,2)-3.*X(I,1))
      YETA(I,1)=.5*(-Y(I,3)+4.*Y(I,2)-3.*Y(I,1))
   10 CONTINUE
C
C
C ETA=ETAMAX
   98 DO 11 I=2,IL1
      XXI(I,JL)=.5*(X(I+1,JL)-X(I-1,JL))
      YXI(I,JL)=.5*(Y(I+1,JL)-Y(I-1,JL))
      XETA(I,JL)=-.5*(-X(I,JL2)+4.*X(I,JL1)-3.*X(I,JL))
      YETA(I,JL)=-.5*(-Y(I,JL2)+4.*Y(I,JL1)-3.*Y(I,JL))
   11 CONTINUE
C
C CORNER POINTS
   99 XXI(1,1)=.5*(-X(3,1)+4.*X(2,1)-3.*X(1,1))
      YXI(1,1)=.5*(-Y(3,1)+4.*Y(2,1)-3.*Y(1,1))
      XETA(1,1)=.5*(-X(1,3)+4.*X(1,2)-3.*X(1,1))
      YETA(1,1)=.5*(-Y(1,3)+4.*Y(1,2)-3.*Y(1,1))
C
      XXI(1,JL)=.5*(-X(3,JL)+4.*X(2,JL)-3.*X(1,JL))
      YXI(1,JL)=.5*(-Y(3,JL)+4.*Y(2,JL)-3.*Y(1,JL))
      XETA(1,JL)=-.5*(-X(1,JL2)+4.*X(1,JL1)-3.*X(1,JL))
      YETA(1,JL)=-.5*(-Y(1,JL2)+4.*Y(1,JL1)-3.*Y(1,JL))
C
      XXI(IL,1)=-.5*(-X(IL2,1)+4.*X(IL1,1)-3.*X(IL,1))
      YXI(IL,1)=-.5*(-Y(IL2,1)+4.*Y(IL1,1)-3.*Y(IL,1))
      XETA(IL,1)=.5*(-X(IL,3)+4.*X(IL,2)-3.*X(IL,1))
      YETA(IL,1)=.5*(-Y(IL,3)+4.*Y(IL,2)-3.*Y(IL,1))
C
      XXI(IL,JL)=-.5*(-X(IL2,JL)+4.*X(IL1,JL)-3.*X(IL,JL))
      YXI(IL,JL)=-.5*(-Y(IL2,JL)+4.*Y(IL1,JL)-3.*Y(IL,JL))
      XETA(IL,JL)=-.5*(-X(IL,JL2)+4.*X(IL,JL1)-3.*X(IL,JL))
      YETA(IL,JL)=-.5*(-Y(IL,JL2)+4.*Y(IL,JL1)-3.*Y(IL,JL))
C
C EVALUATE JACOBIAN
      DO 12 J=1,JL
      DO 12 I=1,IL
      XJAC(I,J)=XXI(I,J)*YETA(I,J)-XETA(I,J)*YXI(I,J)
   12 CONTINUE
C
      XJMIN=1.0E+10
      DO 122 J=1,JL
      DO 122 I=1,IL
      IF ( XJAC(I,J) .LT. XJMIN ) THEN
        II = I
        JJ = J
      END IF
       IF ( ABS(XJAC(I,J)) .LT. 1.0E-8)
     + WRITE (6,211) I,J,XJAC(I,J),X(I,J),Y(I,J)
      XJMIN=AMIN1(XJAC(I,J),XJMIN)
  122 CONTINUE
      IF(XJMIN.GT.0.0) GO TO 500
      WRITE(6,205) XJMIN
      WRITE(6,206) II,JJ,X(II,JJ),Y(II,JJ)
C
  211 FORMAT (2X,'I,J,XJAC,X(I,J),Y(I,J) = ',2I5,E10.5,2F7.3)
C PRINT METRICS IF DESIRED
      WRITE (6,207)
      DO 13 I=1,IL
      WRITE (6,200) I
      WRITE (6,201)
      DO 13 J=1,JL
      WRITE (6,202) J,X(I,J),Y(I,J),XJAC(I,J),XXI(I,J),XETA(I,J),
     1    YXI(I,J),YETA(I,J)
   13 CONTINUE
C
C FORMATS
  200 FORMAT (20X,'COLUMN',I3)
  201 FORMAT (4X,'J',5X,'X',8X,'Y',5X,'XJAC(I,J)',6X,'XXI(I,J)',
     1    6X,'XETA(I,J)',6X,'YXI(I,J)',6X,'YETA(I,J)')
  202 FORMAT (2X,I3,7(3X,E10.3))
  205 FORMAT (10X,'*** WARNING : JACOBIAN LESS OR EQUAL TO ZERO',
     -    3X,'MIN(JACOBIAN)=',E15.7,/)
  206 FORMAT (1X,'LOCATION IS AT I,J = ',2I3,2X,' AND X,Y = ',2F7.3,/)
  207 FORMAT (1H1,10X,'TRANSFORMATION DERIVATIVES',//)
  500 RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE "TMSTEP" COMPUTES LOCAL TIME STEP (DTAU(I,J)).
C IF BETA=0. DTAU(I,J) IS A CONSTANT EQUAL TO DTMIN.
C IF BETA.NE.0., LOCAL CFL NUMBER IS USED.
C
      SUBROUTINE TMSTEP
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRI,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAMI1,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLH,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/TIMEW/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
C
      X1=2.*GAM*REI
C
      DO 1 J=1,JL
      DO 1 I=1,IL
```

```
      UC=(YETA(I,J)*U(I,J)-XETA(I,J)*V(I,J))/XJAC(I,J)
      VC=(-YXI(I,J)*U(I,J)+XXI(I,J)*V(I,J))/XJAC(I,J)
      DSET2=XETA(I,J)**2+YETA(I,J)**2
      DSXI2=XXI(I,J)**2+YXI(I,J)**2
      T=GXM*P(I,J)/RHO(I,J)
      DTAU(I,J)=1./(ABS(UC)+ABS(VC)+SQRT(T*(1./DSET2+1./DSXI2))/XM1
     -  +(X1*(XMU(I,J)+PRI+EDDY(I,J)*PRTI))/(RHO(I,J)*DSET2) )
      DTAU(I,J)=CFL*DTAU(I,J)
    1 CONTINUE
C
C FIND DTMIN,DTMAX
      DTMAX=0.
      DTMIN=1.0E+20
      DO 2 J=1,JL
      DO 2 I=1,IL
      DTMIN=AMIN1(DTMIN,DTAU(I,J))
      DTMAX=AMAX1(DTMAX,DTAU(I,J))
    2 CONTINUE
C OVERRIDE CFL-CONDITION .THIS IS DESIRABLE FOR VERY FINE
C GRIDS AND/OR IN VISCOUS REGIONS
      DO 22 J=1,JL
      DO 22 I=1,IL
      IF(DTAU(I,J).GT.DTVIS) GO TO 22
      DTAU(I,J)=DTVIS
   22 CONTINUE
C
      IF(BETA.NE.0.0) GO TO 100
      DO 3 J=1,JL
      DO 3 I=1,IL
      DTAU(I,J)=DTVIS
    3 CONTINUE
C
  100 RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE "STEPX", PERFORMES XI - SWEEP
C
      SUBROUTINE STEPX(ISWTCH,DRMAX,RMSDR)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DAMP/ WE,WI,WPE,WPI,ISPECT
      COMMON/SPECTR/SPECT(IM,JM),PSMU(IM,JM)
C
C ------------ COMPUTE INVISCID JACOBIAN MATRIX A=DF/DQ ------------
      CALL CMATA
C ------------ COMPUTE RHS-VECTOR ----------------------------------
C
      CALL RHSV
C
C ------------ ADD FOURTH-ORDER EXPLICIT DAMPING -------------------
C
      CALL CONVRT1
      IF(ISPECT.EQ.0) GO TO 10
      CALL SPECR
   10 CONTINUE
      CALL DAMPEY (WE,WPE)
      CALL DAMPEX (WE,WPE)
C
C ------------ MULTIPLY RHS-VECTOR BY -DTAU ------------------------
C
      DO 1 K=1,4
      DO 1 J=2,JL1
      DO 1 I=2,IL1
      RHS(I,J,K)=-DTAU(I,J)*RHS(I,J,K)
    1 CONTINUE
C
C ------------ COMPUTE MAX. & L2-NORM OF RESIDUAL ------------------
      IF(ISWTCH.EQ.0) GO TO 20
      CALL CONVRG(DRMAX,RMSDR)
   20 CONTINUE
C
C ------------ SOLVE BLOCK-TRIDIAGONAL LINEAR SYSTEM ---------------
C
      CALL BTRIDX
C
      RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE "STEPY", PERFORMES ETA - SWEEP
C
      SUBROUTINE STEPY
C
      PARAMETER(IM=299,JM=100)
      COMMON/DAMP/ WE,WI,WPE,WPI,ISPECT
C
C ------------ COMPUTE INVISCID JACOBIAN MATRIX B=DG/DQ ------------
      CALL CMATB
C ------------ SOLVE BLOCK-TRIDIAGONAL LINEAR SYSTEM ---------------
C
      IF(WPE.LT.1.0E-05) GO TO 10
      CALL PSMOOTH(2)
   10 CONTINUE
      CALL BTRIDY
C
      RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE "DAMPEX" IMPLEMENTS  FOURTH-ORDER EXPLICIT
C DAMPING  IN XI-DIRECTION
C
      SUBROUTINE DAMPEX (W,WP)
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     -              XMU(IM,JM),EDDY(IM,JM
```

```
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     -  XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      IF(MP.LT.1.0E-05) GO TO 10
      CALL PSMOOTH(1)
   10 CONTINUE
C
      DO 1 K=1,4
      DO 1 J=2,JL1
      DO 1 I=3,IL2
C
      RHS(I,J,K)=RHS(I,J,K)+SPECT(I,J)*(MP*(TEMP(I+2,J,K)
     - -4.*TEMP(I+1,J,K)+6.*TEMP(I,J,K)-4.*TEMP(I-1,J,K)
     - +TEMP(I-2,J,K))-MP*PSMU(I,J)*(TEMP(I+1,J,K)
     - -2.*TEMP(I,J,K)+TEMP(I-1,J,K)))
C
    1 CONTINUE
C
C ---------------  I = 2 ------------------------------
C
      DO 2 K=1,4
      DO 2 J=2,JL1
C
      RHS(2,J,K)=RHS(2,J,K)+SPECT(2,J)*(MP*(TEMP(4,J,K)
     -  -3.*TEMP(3,J,K)+3.*TEMP(2,J,K)-TEMP(1,J,K))
     -  -MP*PSMU(2,J)*(TEMP(3,J,K)-2.*TEMP(2,J,K
     -  +TEMP(1,J,K)))
C
    2 CONTINUE
C
C ---------------  I = IL-1 ---------------------------
C
      DO 3 K=1,4
      DO 3 J=2,JL1
C
      RHS(IL1,J,K)=RHS(IL1,J,K)+SPECT(IL1,J)*(MP*(TEMP(IL3,J,K)
     -  -3.*TEMP(IL2,J,K)+3.*TEMP(IL1,J,K)-TEMP(IL,J,K))
     -  -MP*PSMU(IL1,J)*(TEMP(IL,J,K)-2.*TEMP(IL1,J,K)
     -  +TEMP(IL2,J,K)))
C
    3 CONTINUE
C
      RETURN
      END
C
C**********************************************************
C
C SUBROUTINE "DAMPEY" IMPLEMENTS  FOURTH-ORDER EXPLICIT
C DAMPING  IN ETA-DIRECTION.
C
      SUBROUTINE DAMPEY (M,MP)
      PARAMETER (IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
```

```
     -           XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     -  XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      IF(MP.LT.1.0E-05) GO TO 10
      CALL PSMOOTH(2)
   10 CONTINUE
C
      DO 1 K=1,4
      DO 1 J=3,JL2
      DO 1 I=2,IL1
C
      RHS(I,J,K)=RHS(I,J,K)+SPECT(I,J)*(MP*(TEMP(I,J+2,K)
     - -4.*TEMP(I,J+1,K)+6.*TEMP(I,J,K)-4.*TEMP(I,J-1,K)
     - +TEMP(I,J-2,K))-MP*PSMU(I,J)*(TEMP(I,J+1,K)
     -  -2.*TEMP(I,J,K)+TEMP(I,J-1,K)))
C
    1 CONTINUE
C
C ---------------  J = 2 ------------------------------
C
      DO 2 K=1,4
      DO 2 I=2,IL1
C
      RHS(I,2,K)=RHS(I,2,K)+SPECT(I,2)*(MP*(TEMP(I,4,K)
     -  -3.*TEMP(I,3,K)+3.*TEMP(I,2,K)-TEMP(I,1,K))
     -  -MP*PSMU(I,2)*(TEMP(I,3,K)-2.*TEMP(I,2,K)
     -  +TEMP(I,1,K)))
C
    2 CONTINUE
C
C ---------------  J = JL-1 ---------------------------
C
      DO 3 K=1,4
      DO 3 I=2,IL1
C
      RHS(I,JL1,K)=RHS(I,JL1,K)+SPECT(I,JL1)*(MP*(TEMP(I,JL3,K)
     -  -3.*TEMP(I,JL2,K)+3.*TEMP(I,JL1,K)-TEMP(I,JL,K))
     -  -MP*PSMU(I,JL1)*(TEMP(I,JL,K)-2.*TEMP(I,JL1,K)
     -  +TEMP(I,JL2,K)))
C
    3 CONTINUE
C
      RETURN
      END
C
C**********************************************************
C
C SUBROUTINE "RHSV"
C COMPUTES THE RHS-VECTOR FOR STEP #1
C I.E. RHS=DF/DXI+DG/DETA-DV1/DXI-DV2/DXI-DW1/DETA-DW2/DETA
C
      SUBROUTINE RHSV
```

```
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TM,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
C
C     ------- COMPUTE RHS-VECTOR ----------------
C
C     ------------------------------------------------------------
C           PART #1.. INVISCID TERMS : DF/DXI+DG/DETA
C     ------------------------------------------------------------
C     ********************   DF / DXI **********************
C
      DO 1 J=2,JL1
C
      DO 1 I=1,IL
      DX1=YETA(I,J)*U(I,J)-XETA(I,J)*V(I,J)
      DX2=(1.+GAM1I)*P(I,J)+.5*(U(I,J)**2+V(I,J)**2)
     -     *RHO(I,J)
C
      TEMP(I,J,1)=RHO(I,J)*DX1
      TEMP(I,J,2)=RHO(I,J)*U(I,J)*DX1+YETA(I,J)*P(I,J)
      TEMP(I,J,3)=RHO(I,J)*V(I,J)*DX1-XETA(I,J)*P(I,J)
      TEMP(I,J,4)=DX1*DX2
    1 CONTINUE
C
      DO 3 K=1,4
      DO 3 J=2,JL1
      DO 3 I=2,IL1
      RHS(I,J,K)=.5*(TEMP(I+1,J,K)-TEMP(I-1,J,K))
    3 CONTINUE
C
C     ********************  DG / DETA  ***************************
C
      DO 4 J=1,JL
C
      DO 4 I=2,IL1
      DX1=XXI(I,J)*V(I,J)-YXI(I,J)*U(I,J)
      DX3=(1.+GAM1I)*P(I,J)+.5*(U(I,J)**2
     -     +V(I,J)**2)*RHO(I,J)
C
      TEMP(I,J,1)=RHO(I,J)*DX1
      TEMP(I,J,2)=RHO(I,J)*U(I,J)*DX1-YXI(I,J)*P(I,J)
      TEMP(I,J,3)=RHO(I,J)*V(I,J)*DX1+XXI(I,J)*P(I,J)
      TEMP(I,J,4)=DX1*DX3
C
    4 CONTINUE
C
      DO 5 K=1,4
      DO 5 J=2,JL1
      DO 5 I=2,IL1
      RHS(I,J,K)=RHS(I,J,K)+.5*(TEMP(I,J+1,K)-TEMP(I,J-1,K))
    5 CONTINUE
C
C     ------------------------------------------------------------
C       PART #2 .  VISCOUS TERMS : DV1/DXI+DV2/DXI+DW1/DETA+DW2/DETA
C     ------------------------------------------------------------
C
C     ******************** DV2 /DXI ************************
      DO 6 J=2,JL1
C
      DO 7 I=1,IL
      DX1=XMU(I,J)+EDDY(I,J)
      DX2=XXI(I,J)*XETA(I,J)
      DX3=YXI(I,J)*YETA(I,J)
      DX4=XXI(I,J)*YETA(I,J)
      DX5=XETA(I,J)*YXI(I,J)
      DX6=-DX1*(DX2+XLM*DX3)
      DX7=DX1*(DX5+XL*DX4)
      DX8=DX1*(DX4+XL*DX5)
      DX9=-DX1*(XLM*DX2+DX3)
      DX10=-GAM1I*(XMU(I,J)*PRI+EDDY(I,J)*PRTI)
     -     *(DX2+DX3)
C
      DX1=REI/XJAC(I,J)
      DX2=.5*(U(I,J+1)-U(I,J-1))
      DX3=.5*(V(I,J+1)-V(I,J-1)
      DX4=.5*GAM*(P(I,J+1)/RHO(I,J+1)-
     -     P(I,J-1)/RHO(I,J-1))
C
      TEMP(I,J,1)=0.0
      TEMP(I,J,2)=DX1*(DX6*DX2+DX7*DX3)
      TEMP(I,J,3)=DX1*(DX8*DX2+DX9*DX3)
      TEMP(I,J,4)=TEMP(I,J,2)*U(I,J)+TEMP(I,J,3)*V(I,J)
     -     +DX1*DX10*DX4
    7 CONTINUE
C
    6 CONTINUE
C
      DO 9 K=1,4
      DO 9 J=2,JL1
      DO 9 I=2,IL1
      RHS(I,J,K)=RHS(I,J,K)-.5*(TEMP(I+1,J,K)-TEMP(I-1,J,K))
    9 CONTINUE
C
C     ******************** DW1 / DETA ***************************
C
      DO 10 J=1,JL
C
      DO 11 I=2,IL1
      DX1=XMU(I,J)+EDDY(I,J)
      DX2=XXI(I,J)*XETA(I,J)
      DX3=YXI(I,J)*YETA(I,J)
      DX4=XXI(I,J)*YETA(I,J)
      DX5=XETA(I,J)*YXI(I,J)
      DX6=-DX1*(DX2+XLM*DX3)
      DX7=DX1*(DX5+XL*DX4
      DX8=DX1*(DX4+XL*DX5
      DX9=-DX1*(XLM*DX2+DX3
      DX10=-GAM1I*(XMU(I,J)*PRI+EDDY(I,J)*PRTI)
     -     *(DX2+DX3)
C
      DX1=REI/XJAC(I,J)
```

```
      DX2=.5*(U(I+1,J)-U(I-1,J))
      DX3=.5*(V(I+1,J)-V(I-1,J))
      DX4=.5*GAM*(P(I+1,J)/RHO(I+1,J)-
     -        P(I-1,J)/RHO(I-1,J))
C
      TEMP(I,J,1)=0.0
      TEMP(I,J,2)=DX1*(DX6*DX2+DX8*DX3)
      TEMP(I,J,3)=DX1*(DX7*DX2+DX9*DX3)
      TEMP(I,J,4)=TEMP(I,J,2)*U(I,J)+TEMP(I,J,3)*V(I,J)
     -           +DX1*DX10*DX4
   11 CONTINUE
C
   10 CONTINUE
C
      DO 13 K=1,4
      DO 13 J=2,JL1
      DO 13 I=2,IL1
      RHS(I,J,K)=RHS(I,J,K)-.5*(TEMP(I,J+1,K)-TEMP(I,J-1,K))
   13 CONTINUE
C
C ****************  DV1 / DXI ***********************
C
      DO 14 J=2,JL1
C
      DO 15 I=2,IL
      DX1=.5*(XMU(I,J)+XMU(I-1,J)+EDDY(I,J)+EDDY(I-1,J))
      DX2=.5*(XETA(I,J)+XETA(I-1,J))
      DX3=.5*(YETA(I,J)+YETA(I-1,J))
      DX4=DX2**2
      DX5=DX3**2
      DX6=DX1*(DX4+XLM*DX5)
      DX7=-XL1*DX1*DX2*DX3
      DX8=DX1*(XLM*DX4+DX5)
      DX9=GAM1I*.5*(PRI*(XMU(I,J)+XMU(I-1,J))+
     -      PRTI*(EDDY(I,J)+EDDY(I-1,J))*(DX4+DX5)
C
      DX1=REI/(.5*(XJAC(I,J)+XJAC(I-1,J)))
      DX2=U(I,J)-U(I-1,J)
      DX3=V(I,J)-V(I-1,J)
      DX4=GAM*(P(I,J)/RHO(I,J)-P(I-1,J)/RHO(I-1,J))
C
      TEMP(I,J,1)=0.0
      TEMP(I,J,2)=DX1*(DX6*DX2+DX7*DX3)
      TEMP(I,J,3)=DX1*(DX7*DX2+DX8*DX3)
      TEMP(I,J,4)=TEMP(I,J,2)*.5*(U(I,J)+U(I-1,J))
     -           +TEMP(I,J,3)*.5*(V(I,J)+V(I-1,J))
     -           +DX1*DX9*DX4
   15 CONTINUE
C
   14 CONTINUE
C
      DO 17 K=1,4
      DO 17 J=2,JL1
      DO 17 I=2,IL1
      RHS(I,J,K)=RHS(I,J,K)-(TEMP(I+1,J,K)-TEMP(I,J,K))
   17 CONTINUE
C
C ****************  DV2 / DETA ***********************
C
      DO 18 J=2,JL
C
```

```
      DO 19 I=2,IL1
      DX1=.5*(XMU(I,J)+XMU(I,J-1)+EDDY(I,J)+EDDY(I,J-1))
      DX2=.5*(XXI(I,J)+XXI(I,J-1))
      DX3=.5*(YXI(I,J)+YXI(I,J-1))
      DX4=DX2**2
      DX5=DX3**2
      DX6=DX1*(DX4+XLM*DX5)
      DX7=-XL1*DX1*DX2*DX3
      DX8=DX1*(XLM*DX4+DX5)
      DX9=GAM1I*.5*(PRI*(XMU(I,J)+XMU(I,J-1))+
     -      PRTI*(EDDY(I,J)+EDDY(I,J-1)))*(DX4+DX5)
C
      DX1=REI/(.5*(XJAC(I,J)+XJAC(I,J-1)))
      DX2=U(I,J)-U(I,J-1)
      DX3=V(I,J)-V(I,J-1)
      DX4=GAM*(P(I,J)/RHO(I,J)-P(I,J-1)/RHO(I,J-1))
C
      TEMP(I,J,1)=0.0
      TEMP(I,J,2)=DX1*(DX6*DX2+DX7*DX3)
      TEMP(I,J,3)=DX1*(DX7*DX2+DX8*DX3)
      TEMP(I,J,4)=TEMP(I,J,2)*.5*(U(I,J)+U(I,J-1))
     -           +TEMP(I,J,3)*.5*(V(I,J)+V(I,J-1))
     -           +DX1*DX9*DX4
   19 CONTINUE
C
   18 CONTINUE
C
      DO 21 K=1,4
      DO 21 J=2,JL1
      DO 21 I=2,IL1
      RHS(I,J,K)=RHS(I,J,K)-(TEMP(I,J+1,K)-TEMP(I,J,K))
   21 CONTINUE
C
      RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE 'CMATA', COMPUTES JACOBIAN MATRIX A= DF/DQ
      SUBROUTINE CMATA
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TW,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      DO 2 J=2,JL1
C
      DO 1 I=1,IL
      DX1=1.0/XJAC(I,J)
      DX2=YETA(I,J)*U(I,J)-XETA(I,J)*V(I,J)
      DX3=.5*GAM1*(U(I,J)**2+V(I,J)**2
```

```
      DX4=GAM*(P(I,J)*GAM1I/RHO(I,J)+.5*(U(I,J)**2
     -        +V(I,J)**2))
C
      A(I,J,1,1)=0.
      A(I,J,1,2)=-DX1*YETA(I,J)
      A(I,J,1,3)=-DX1*XETA(I,J)
      A(I,J,1,4)=0.
C
      A(I,J,2,1)=DX1*(YETA(I,J)*DX3-DX2*U(I,J))
      A(I,J,2,2)=DX1*(DX2-GAM2*YETA(I,J)*U(I,J))
      A(I,J,2,3)=-DX1*(XETA(I,J)*U(I,J)+GAM1*YETA(I,J)*V(I,J))
      A(I,J,2,4)=DX1*GAM1*YETA(I,J)
C
      A(I,J,3,1)=-DX1*(DX3*XETA(I,J)+DX2*V(I,J))
      A(I,J,3,2)=DX1*(GAM1*XETA(I,J)*U(I,J)+YETA(I,J)*V(I,J))
      A(I,J,3,3)=DX1*(DX2+GAM2*XETA(I,J)*V(I,J))
      A(I,J,3,4)=-DX1*GAM1*XETA(I,J)
C
      A(I,J,4,1)=DX1*DX2*(2.0*DX3-DX4)
      A(I,J,4,2)=DX1*(YETA(I,J)*(DX4-DX3)-GAM1*DX2*U(I,J))
      A(I,J,4,3)=-DX1*(XETA(I,J)*(DX4-DX3)+GAM1*DX2*V(I,J))
      A(I,J,4,4)=DX1*GAM*DX2
C
    1 CONTINUE
    2 CONTINUE
C
      RETURN
      END
C
C**********************************************************************
C
C SUBROUTINE 'CMATB', COMPUTES JACOBIAN MATRIX A= DG/DQ
      SUBROUTINE CMATB
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TM,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      DO 2 J=1,JL
C
      DO 1 I=2,IL1
C
      DX1=1.0/XJAC(I,J)
      DX3=.5*GAM1*(U(I,J)**2+V(I,J)**2)
      DX2=XXI(I,J)*V(I,J)-YXI(I,J)*U(I,J)
      DX4=GAM*(GAM1I*P(I,J)/RHO(I,J)+.5*(U(I,J)**2+
     -        V(I,J)**2))
C
      A(I,J,1,1)=0.
      A(I,J,1,2)=-DX1*YXI(I,J)
      A(I,J,1,3)=DX1*XXI(I,J)
      A(I,J,1,4)=0.
C
      A(I,J,2,1)=-DX1*(YXI(I,J)*DX3+U(I,J)*DX2)
      A(I,J,2,2)=DX1*(DX2+GAM2*YXI(I,J)*U(I,J))
      A(I,J,2,3)=DX1*(GAM1*YXI(I,J)*V(I,J)+XXI(I,J)*U(I,J))
      A(I,J,2,4)=-DX1*GAM1*YXI(I,J)
C
      A(I,J,3,1)=DX1*(XXI(I,J)*DX3-V(I,J)*DX2)
      A(I,J,3,2)=-DX1*(GAM1*XXI(I,J)*U(I,J)+YXI(I,J)*V(I,J))
      A(I,J,3,3)=DX1*(DX2-GAM2*XXI(I,J)*V(I,J))
      A(I,J,3,4)=DX1*GAM1*XXI(I,J)
C
      A(I,J,4,1)=DX1*DX2*(2.0*DX3-DX4)
      A(I,J,4,2)=-DX1*(YXI(I,J)*(DX4-DX3)+GAM1*U(I,J)*DX2)
      A(I,J,4,3)=DX1*(XXI(I,J)*(DX4-DX3)-GAM1*V(I,J)*DX2)
      A(I,J,4,4)=DX1*GAM*DX2
C
    1 CONTINUE
    2 CONTINUE
C
      RETURN
      END
C
C**********************************************************************
C
C SUBROUTINE 'CMATR', COMPUTES JACOBIAN MATRIX R= DV1/DQXI
      SUBROUTINE CMATR(I,R)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TM,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      DIMENSION R(IM,4,4)
C
      DO 2 J=2,JL1
C
      DX6=XETA(I,J)**2
      DX7=YETA(I,J)**2
      DX8=XMU(I,J)+EDDY(I,J)
      DX1=REI/(RHO(I,J)*XJAC(I,J)**2)
      DX2=(DX6+XLM*DX7)*DX8
      DX3=-XL1*DX8*XETA(I,J)*YETA(I,J)
      DX4=(XLM*DX6+DX7)*DX8
      DX5=GAM*(XMU(I,J)*PRI+EDDY(I,J)*PRTI)*(DX6+DX7)
C
      R(J,1,1)=0.
      R(J,1,2)=0.
      R(J,1,3)=0.
      R(J,1,4)=0.
C
      R(J,2,1)=-DX1*(DX2*U(I,J)+DX3*V(I,J))
      R(J,2,2)=DX1*DX2
      R(J,2,3)=DX1*DX3
```

```
      R(J,2,4)=0.
C
      R(J,3,1)=-DX1*(DX3*U(I,J)+DX4*V(I,J))
      R(J,3,2)=DX1*DX3
      R(J,3,3)=DX1*DX4
      R(J,3,4)=0.
C
      R(J,4,1)=-DX1*(DX2*U(I,J)**2+2.*DX3*U(I,J)*V(I,J)
     -          +DX4*V(I,J)**2-DX5*(.5*(U(I,J)**2+V(I,J)**2)
     -          -P(I,J)*GAM1I/RHO(I,J)))
      R(J,4,2)=DX1*(U(I,J)*(DX2-DX5)+DX3*V(I,J))
      R(J,4,3)=DX1*(DX3*U(I,J)+V(I,J)*(DX4-DX5))
      R(J,4,4)=DX1*DX5
C
    2 CONTINUE
C
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE 'CMATS', COMPUTES JACOBIAN MATRIX S= DN2/DQETA
      SUBROUTINE CMATS(J,R)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SIMA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      DIMENSION R(IM,4,4)
C
      DO 1 I=2,IL1
C
      DX6=XXI(I,J)**2
      DX7=YXI(I,J)**2
      DX8=XMU(I,J)+EDDY(I,J)
C
      DX1=REI/(RHO(I,J)*XJAC(I,J)**2)
      DX2=(DX6+XLM*DX7)*DX8
      DX3=-XL1*DX8*XXI(I,J)*YXI(I,J)
      DX4=(XLM*DX6-DX7)*DX8
      DX5=(XMU(I,J)*PRI+EDDY(I,J)*PRTI)*(DX6
     -          +DX7)*GAM
C
      R(I,1,1)=0.
      R(I,1,2)=0.
      R(I,1,3)=0.
      R(I,1,4)=0.
C
      R(I,2,1)=-DX1*(DX2*U(I,J)+DX3*V(I,J))
      R(I,2,2)=DX1*DX2
      R(I,2,3)=DX1*DX3
      R(I,2,4)=0.
C
      R(I,3,1)=-DX1*(DX3*U(I,J)+DX4*V(I,J))
      R(I,3,2)=DX1*DX3
      R(I,3,3)=DX1*DX4
      R(I,3,4)=0.
C
      R(I,4,1)=-DX1*(DX2*U(I,J)**2+2.*DX3*U(I,J)*V(I,J)
     -          +DX4*V(I,J)**2-DX5*(.5*(U(I,J)**2+V(I,J)**2)
     -          -GAM1I*P(I,J)/RHO(I,J)))
      R(I,4,2)=DX1*(U(I,J)*(DX2-DX5)+DX3*V(I,J))
      R(I,4,3)=DX1*(U(I,J)*DX3+V(I,J)*(DX4-DX5))
      R(I,4,4)=DX1*DX5
C
    1 CONTINUE
C
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE "CXMU", COMPUTES MOLECULAR DYNAMIC VISCOSITY USING
C SUTHERLAND'S FORMULA
      SUBROUTINE CXMU(ICHECK)
C
      PARAMETER(IM=299,JM=100
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SIMA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
C
      DO 1 J=1,JL
      DO 1 I=1,IL
        T=GXM*P(I,J)/RHO(I,J)
        XMU(I,J)=SQRT(T**3)*S1P/(T+S1)
    1 CONTINUE
C
  500 FORMAT (2X,'T,P,RHO = ',3F10.4,' AT I,J ',2I4)
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE "CONVRT", COMPUTES RHO/J, RHO*U/J,ETC.  USING PRIMITIVE
C VARIABLES RHO, U, V AND P.
      SUBROUTINE CONVRT
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SIMA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
```

```
      DO 1 J=1,JL
      DO 1 I=1,IL
        TEMP(I,J,1)=XJAC(I,J)*RHO(I,J)
        TEMP(I,J,2)=TEMP(I,J,1)*U(I,J)
        TEMP(I,J,3)=TEMP(I,J,1)*V(I,J)
        TEMP(I,J,4)=XJAC(I,J)*(GAM1I*P(I,J)+
     -    .5*RHO(I,J)*(U(I,J)**2+V(I,J)**2) )
    1 CONTINUE
C
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE "UCONVRT", COMPUTES PRIMITIVE VARIABLES RHO, U, V AND P
C USING RHO*J, RHO*U/J, ETC.
      SUBROUTINE UCONVRT
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/NORM/ UNORM,VNORM
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      UNORM = 0.0
      VNORM = 0.0
      DO 1 J=1,JL
      DO 1 I=1,IL
C
      ULAST = U(I,J)
      VLAST = V(I,J)
      RHO(I,J)=TEMP(I,J,1)/XJAC(I,J)
      RR=1./TEMP(I,J,1)
      U(I,J)=TEMP(I,J,2)*RR
      V(I,J)=TEMP(I,J,3)*RR
      P(I,J)=GAM1*RHO(I,J)*(TEMP(I,J,4)*RR-
     -   .5*(U(I,J)**2+V(I,J)**2))
      UNORM = UNORM + ( ULAST - U(I,J) )**2
      VNORM = VNORM + ( VLAST - V(I,J) )**2
C
    1 CONTINUE
      UNORM = SQRT(UNORM / FLOAT((IL-1)*(JL-1)))
      VNORM = SQRT(VNORM / FLOAT((IL-1)*(JL-1)))
C
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE "CONVRT1" FORMS CONSERVATION VARIABLES USING
C THE AVAILABLE PRIMITIVE VARIABLES RHO,U,V,P
      SUBROUTINE CONVRT1

      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     -          XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     -  XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      DO 1 J=1,JL
      DO 1 I=1,IL
        TEMP(I,J,1)=RHO(I,J)
        TEMP(I,J,2)=RHO(I,J)*U(I,J)
        TEMP(I,J,3)=RHO(I,J)*V(I,J)
        TEMP(I,J,4)=GAM1I*P(I,J)+.5*RHO(I,J)*(U(I,J)**2
     -    +V(I,J)**2)
    1 CONTINUE
C
      RETURN
      END
C
C*************************************************************
C
C SUBROUTINE "MONITR", COMPUTES INFORMATION REQUIRED DURING
C TIME INTEGRATION. THIS INCLUDES:
C    A) CONVERGENCE CRITERIA
C    B) AIRFOIL LIFT AND DRAG COEFFICIENTS
      SUBROUTINE MONITR (NSTEP,TAU,DRMAX,RMSDR)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/NORM/ UNORM,VNORM
C
C ---------------------------------------------------------------
C COMPUTE AIRFOIL LIFT AND DRAG COEFFICIENTS
C ---------------------------------------------------------------
C
      CALL LIFT (CLIFT,CDRAG)
C
C ---------- OUTPUT ----------
      WRITE(6,200) NSTEP,TAU,DRMAX,RMSDR,UNORM,VNORM,CLIFT,CDRAG
      CP = P(ITEU,1) - 1./GXM
      XN=FLOAT(NSTEP)
      XLOGMX=ALOG10(DRMAX)
      XLOGRM=ALOG10(RMSDR)
      WRITE(3,201) XN,TAU,XLOGMX,XLOGRM,CLIFT,CDRAG ,UNORM,VNORM,
     +             RMSDR,CP
C
C ---------- FORMATS ----------
```

```
  200 FORMAT(1X,'T =',I5,' TAU=',E10.3,
     -  4E12.5,' CL=',E12.5,' CD=',E12.6)
  201 FORMAT(10E13.5)
C
      RETURN
      END
C
C ***************************************************************
C
C SUBROUTINE "CONVRG"
      SUBROUTINE CONVRG (DRMAX,RMSDR)

      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     -        XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     -XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON/TIMEH/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
C
      SUMR=0.
      DRMAX=0.
      IL1=IL-1
      JL1=JL-1
      FN=FLOAT((IL-2)*(JL-2))
C
      DO 1 I=2,IL1
      DO 1 J=2,JL1
        DR=ABS(RHS(I,J,1))
        DRMAX=AMAX1(DRMAX,DR)
        SUMR=SUMR+DR*DR
    1 CONTINUE
      RMSDR=SQRT(SUMR/FN)
      RETURN
      END

C
C ***************************************************************
C
C SUBROUTINE "BTRIDX" SOLVES BLOCK-TRIDIAGONAL SYSTEMS
C FOR THE XI - SWEEP
      SUBROUTINE BTRIDX
C
      PARAMETER(IM=299,JM=100)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON/MISC/JL1,JL2,JL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PR1,PRT1,REI,XL,XL1,XLM,PI
      COMMON/TIMEH/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DAMP/ ME,MI,MPE,MPI,ISPECT
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      COMMON/BTRID/ AA(IM,4,4),BB(IM,4,4),CC(IM,4,4),
     - HRK(IM,4,4),L11(IM),L21(IM),L22(IM),L31(IM),
     - L32(IM),L33(IM),L41(IM),L42(IM),
     - L43(IM),L44(IM),U12(IM),U13(IM),U14(IM),
     - U23(IM),U24(IM),U34(IM)
C
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,
     -     U12,U13,U14,U23,U24,U34
C
C ---------------------------------------------------------------
C                 PART 1.   FORWARD  SWEEP
C ---------------------------------------------------------------
C
C  PREPARE FOR FIRST ITERATION
      CALL FILLMX(2)
C
      DO 1 I=2,IL1
C
      IF (I.EQ.2) GO TO 100
C
      DO 2 L=1,4
      DO 2 J=2,JL1
C
      RHS(I,J,L)=RHS(I,J,L)-BB(J,L,1)*RHS(I-1,J,1)
     1           -BB(J,L,2)*RHS(I-1,J,2)
     2           -BB(J,L,3)*RHS(I-1,J,3)
     3           -BB(J,L,4)*RHS(I-1,J,4)
C
      AA(J,L,1)=AA(J,L,1)-BB(J,L,1)*A(I-1,J,1,1)
     1          -BB(J,L,2)*A(I-1,J,2,1)
     2          -BB(J,L,3)*A(I-1,J,3,1)
     3          -BB(J,L,4)*A(I-1,J,4,1)
C
      AA(J,L,2)=AA(J,L,2)-BB(J,L,1)*A(I-1,J,1,2)
     1          -BB(J,L,2)*A(I-1,J,2,2)
     2          -BB(J,L,3)*A(I-1,J,3,2)
     3          -BB(J,L,4)*A(I-1,J,4,2)
C
      AA(J,L,3)=AA(J,L,3)-BB(J,L,1)*A(I-1,J,1,3)
     1          -BB(J,L,2)*A(I-1,J,2,3)
     2          -BB(J,L,3)*A(I-1,J,3,3)
     3          -BB(J,L,4)*A(I-1,J,4,3)
C
      AA(J,L,4)=AA(J,L,4)-BB(J,L,1)*A(I-1,J,1,4)
     1          -BB(J,L,2)*A(I-1,J,2,4)
     2          -BB(J,L,3)*A(I-1,J,3,4)
     3          -BB(J,L,4)*A(I-1,J,4,4)
C
C
    2 CONTINUE
C
  100 CONTINUE
C
C ------- USE CROUT'S METHOD TO SOLVE 4 X 4 SYSTEMS OF EQUATIONS -------
C
      DO 3 J=2,JL1
C
      L11(J)=1./AA(J,1,1)
      U12(J)=AA(J,1,2)*L11(J)
      U13(J)=AA(J,1,3)*L11(J)
      U14(J)=AA(J,1,4)*L11(J)
      L21(J)=AA(J,2,1)
      L22(J)=1./(AA(J,2,2)-L21(J)*U12(J))
      U23(J)=L22(J)*(AA(J,2,3)-L21(J)*U13(J))
      U24(J)=L22(J)*(AA(J,2,4)-L21(J)*U14(J
```

```
      L31(J)=AA(J,3,1)
      L32(J)=AA(J,3,2)-L31(J)*U12(J)
      L33(J)=1./(AA(J,3,3)-L31(J)*U13(J)-L32(J)*U23(J))
      U34(J)=L33(J)*(AA(J,3,4)-L31(J)*U14(J)
     1       -L32(J)*U24(J))
      L41(J)=AA(J,4,1)
      L42(J)=AA(J,4,2)-L41(J)*U12(J)
      L43(J)=AA(J,4,3)-L41(J)*U13(J)-L42(J)*U23(J)
      L44(J)=1./(AA(J,4,4)-L41(J)*U14(J)
     1       -L42(J)*U24(J)-L43(J)*U34(J) )
C
C  SOLVE FOR INTERMEDIATE VECTOR
C
      DX1=L11(J)*RHS(I,J,1)
      DX2=L22(J)*(RHS(I,J,2)-L21(J)*DX1)
      DX3=L33(J)*(RHS(I,J,3)-L31(J)*DX1
     1       -L32(J)*DX2 )
      DX4=L44(J)*(RHS(I,J,4)-L41(J)*DX1
     1       -L42(J)*DX2-L43(J)*DX3 )
C
      RHS(I,J,4)=DX4
      RHS(I,J,3)=DX3-U34(J)*DX4
      RHS(I,J,2)=DX2-U23(J)*RHS(I,J,3)
     1          -U24(J)*DX4
      RHS(I,J,1)=DX1-U12(J)*RHS(I,J,2)
     1          -U13(J)*RHS(I,J,3)-U14(J)*DX4
C
    3 CONTINUE
C
      IF (I.EQ.IL1) GO TO 1
C
C ------------  COMPUTE  GAMMA(I,J) (SEE NOTES FOR DEF.) ------------
C
      DO 4 M=1,4
      DO 4 J=2,JL1
C
      DX1=L11(J)*CC(J,1,M)
      DX2=L22(J)*(CC(J,2,M)-L21(J)*DX1)
      DX3=L33(J)*(CC(J,3,N)-L31(J)*DX1
     1       -L32(J)*DX2)
      DX4=L44(J)*(CC(J,4,M)-L41(J)*DX1
     1       -L42(J)*DX2-L43(J)*DX3)
C
      WRK(J,4,M)=DX4
      WRK(J,3,M)=DX3-U34(J)*DX4
      WRK(J,2,M)=DX2-U23(J)*WRK(J,3,M)
     1          -U24(J)*DX4
      WRK(J,1,M)=DX1-U12(J)*WRK(J,2,M)
     1          -U13(J)*WRK(J,3,M)-U14(J)*DX4
C
    4 CONTINUE
C
C ---------  COMPUTE AA,BB,CC FOR NEXT ITERATION ---------
C
      CALL FILLMX(I+1)
C
C --------  SAVE GAMMA(I,J)'S ON JACOBIAN MATRIX A --------
C
      DO 5 L=1,4
      DO 5 M=1,4
      DO 5 J=2,JL1
        A(I,J,L,M)=WRK(J,L,M)
    5 CONTINUE
C
    1 CONTINUE
C
C ================================================================
C                PART 2.     BACKWARD  SWEEP
C ================================================================
C
      IIE=IL1-2
      DO 6 II=1,IIE
      I=IL1-II
C
      DO 7 L=1,4
      DO 7 J=2,JL1
C
      RHS(I,J,L)=RHS(I,J,L)-A(I,J,L,1)*RHS(I+1,J,1)
     1          -A(I,J,L,2)*RHS(I+1,J,2)
     2          -A(I,J,L,3)*RHS(I+1,J,3)
     3          -A(I,J,L,4)*RHS(I+1,J,4)
C
    7 CONTINUE
    6 CONTINUE
C
      RETURN
      END
C
C*****************************************************************
C
C SUBROUTINE "FILLMX" FORMS 4X4 BLOCKS AA, BB, CC
C REQUIRED IN XI - SWEEP (INCLUDES IMPLICIT DAMPING)
      SUBROUTINE FILLMX (I)
C
      PARAMETER(IM=299,JM=100)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DAMP/ WE,WI,WPE,WPI,ISPECT
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/BTRID/ AA(IM,4,4),BB(IM,4,4),CC(IM,4,4),
     - WRK(IM,4,4),L11(IM),L21(IM),L22(IM),L31(IM),
     - L32(IM),L33(IM),L41(IM),L42(IM),
     - L43(IM),L44(IM),U12(IM),U13(IM),U14(IM),
     - U23(IM),U24(IM),U34(IM)
      COMMON/CMAT/ RR(IM,4,4),RP1(IM,4,4),RM1(IM,4,4)
C
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,
     -     U12,U13,U14,U23,U24,U34
C
C COMPUTE VISCOUS JACOBIAN MATRIX 'R'
      IP1=I+1
      IM1=I-1
      IF ( I .GT. 2 ) GOTO 100
      CALL CMATR(I,RR)
```

```
      CALL CMATR(IM1,RM1)
      GOTO 200
  100 DO 11 M=1,4
      DO 11 L=1,4
      DO 11 J=2,JL1
      RM1(J,L,M) = RR(J,L,M)
      RR(J,L,M) = RP1(J,L,M)
   11 CONTINUE
C
  200 CONTINUE
      CALL CMATR(IP1,RP1)

      DO 1 L=1,4
      DO 2 M=1,4
      DO 2 J=2,JL1
        AA(J,L,M)=2.*RR(J,L,M)
        BB(J,L,M)=-.5*A(I-1,J,L,M)+RM1(J,L,M)
        CC(J,L,M)=.5*A(I+1,J,L,M)-RP1(J,L,M)
    2 CONTINUE
C
C ------------  ADD IMPLICIT DAMPING ------------
C
      DO 3 J=2,JL1
      WISP=SPECT(I,J)*(WI+WPI*PSMU(I,J))
      AA(J,L,L)=AA(J,L,L)+2.*WISP/XJAC(I,J
      BB(J,L,L)=BB(J,L,L)-WISP/XJAC(I-1,J
      CC(J,L,L)=CC(J,L,L)-WISP/XJAC(I+1,J
    3 CONTINUE
C
C ------------  MULTIPLY  BY DTAU(I,J) ------------
      DO 4 M=1,4
      DO 4 J=2,JL1
      AA(J,L,M)=DTAU(I,J)*AA(J,L,M)
      BB(J,L,M)=DTAU(I,J)*BB(J,L,M)
      CC(J,L,M)=DTAU(I,J)*CC(J,L,M)
    4 CONTINUE
C
C ------------ ADD IDENTITY MATRIX TO  AA (DIAGONAL) ------
      DO 5 J=2,JL1
        AA(J,L,L)=AA(J,L,L)+1.0
    5 CONTINUE
C
    1 CONTINUE
C
      RETURN
      END

C
C*****************************************************************
C
C SUBROUTINE "BTRIDY" SOLVES BLOCK-TRIDIAGONAL SYSTEMS
C FOR THE ETA - SWEEP
      SUBROUTINE BTRIDY
C
      PARAMETER(IM=299,JM=100)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,SIP,
     -  PINF,SINA,COSA,PR1,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DAMP/ WE,WI,WPE,WPI,ISPECT
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      COMMON/BTRID/ AA(IM,4,4),BB(IM,4,4),CC(IM,4,4),
     - WRK(IM,4,4),L11(IM),L21(IM),L22(IM),L31(IM),
     - L32(IM),L33(IM),L41(IM),L42(IM),
     - L43(IM),L44(IM),U12(IM),U13(IM),U14(IM),
     - U23(IM),U24(IM),U34(IM)
C
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,
     -     U12,U13,U14,U23,U24,U34
C
C ------------------------------------------------------------
C                 PART 1.   FORWARD  SWEEP
C ------------------------------------------------------------
C
C PREPARE FOR FIRST ITERATION
      CALL FILLMY(2)
C
      DO 1 J=2,JL1
C
      IF (J.EQ.2) GO TO 100
C
      DO 2 L=1,4
      DO 2 I=2,IL1
C
      RHS(I,J,L)=RHS(I,J,L)-BB(I,L,1)*RHS(I,J-1,1)
     1           -BB(I,L,2)*RHS(I,J-1,2)
     2           -BB(I,L,3)*RHS(I,J-1,3)
     3           -BB(I,L,4)*RHS(I,J-1,4)
C
      AA(I,L,1)=AA(I,L,1)-BB(I,L,1)*A(I,J-1,1,1)
     1          -BB(I,L,2)*A(I,J-1,2,1)
     2          -BB(I,L,3)*A(I,J-1,3,1)
     3          -BB(I,L,4)*A(I,J-1,4,1)
C
      AA(I,L,2)=AA(I,L,2)-BB(I,L,1)*A(I,J-1,1,2)
     1          -BB(I,L,2)*A(I,J-1,2,2)
     2          -BB(I,L,3)*A(I,J-1,3,2)
     3          -BB(I,L,4)*A(I,J-1,4,2)
C
      AA(I,L,3)=AA(I,L,3)-BB(I,L,1)*A(I,J-1,1,3)
     1          -BB(I,L,2)*A(I,J-1,2,3)
     2          -BB(I,L,3)*A(I,J-1,3,3)
     3          -BB(I,L,4)*A(I,J-1,4,3)
C
      AA(I,L,4)=AA(I,L,4)-BB(I,L,1)*A(I,J-1,1,4)
     1          -BB(I,L,2)*A(I,J-1,2,4)
     2          -BB(I,L,3)*A(I,J-1,3,4)
     3          -BB(I,L,4)*A(I,J-1,4,4)
C
    2 CONTINUE
C
  100 CONTINUE
C
C ------------ USE CROUT'S METHOD TO SOLVE 4 X 4 SYSTEMS OF EQUATIONS ------------
C
      DO 3 I=2,IL1
```

```
C
      L11(I)=1./AA(I,1,1)
      U12(I)=AA(I,1,2)*L11(I)
      U13(I)=AA(I,1,3)*L11(I)
      U14(I)=AA(I,1,4)*L11(I)
      L21(I)=AA(I,2,1)
      L22(I)=1./(AA(I,2,2)-L21(I)*U12(I))
      U23(I)=L22(I)*(AA(I,2,3)-L21(I)*U13(I))
      U24(I)=L22(I)*(AA(I,2,4)-L21(I)*U14(I))
      L31(I)=AA(I,3,1)
      L32(I)=AA(I,3,2)-L31(I)*U12(I)
      L33(I)=1./(AA(I,3,3)-L31(I)*U13(I)-L32(I)*U23(I))
      U34(I)=L33(I)*(AA(I,3,4)-L31(I)*U14(I)
     1        -L32(I)*U24(I))
      L41(I)=AA(I,4,1)
      L42(I)=AA(I,4,2)-L41(I)*U12(I)
      L43(I)=AA(I,4,3)-L41(I)*U13(I)-L42(I)*U23(I)
      L44(I)=1./(AA(I,4,4)-L41(I)*U14(I)
     1        -L42(I)*U24(I)-L43(I)*U34(I) )
C
C  SOLVE FOR INTERMEDIATE VECTOR
C
      DX1=L11(I)*RHS(I,J,1)
      DX2=L22(I)*(RHS(I,J,2)-L21(I)*DX1)
      DX3=L33(I)*(RHS(I,J,3)-L31(I)*DX1
     1        -L32(I)*DX2 )
      DX4=L44(I)*(RHS(I,J,4)-L41(I)*DX1
     1        -L42(I)*DX2-L43(I)*DX3 )
C
      RHS(I,J,4)=DX4
      RHS(I,J,3)=DX3-U34(I)*DX4
      RHS(I,J,2)=DX2-U23(I)*RHS(I,J,3)
     1           -U24(I)*DX4
      RHS(I,J,1)=DX1-U12(I)*RHS(I,J,2)
     1           -U13(I)*RHS(I,J,3)-U14(I)*DX4
C
    3 CONTINUE
C
      IF (J EQ.JL1) GO TO 1
C
C ------------  COMPUTE  GAMMA(I,J) (SEE NOTES FOR DEF.) ------------
C
      DO 4 M=1,4
      DO 4 I=2,IL1
C
      DX1=L11(I)*CC(I,1,M)
      DX2=L22(I)*(CC(I,2,M)-L21(I)*DX1)
      DX3=L33(I)*(CC(I,3,M)-L31(I)*DX1
     1        -L32(I)*DX2)
      DX4=L44(I)*(CC(I,4,M)-L41(I)*DX1
     1        -L42(I)*DX2-L43(I)*DX3)
C
      WRK(I,4,M)=DX4
      WRK(I,3,M)=DX3-U34(I)*DX4
      WRK(I,2,M)=DX2-U23(I)*WRK(I,3,M)
     1           -U24(I)*DX4
      WRK(I,1,M)=DX1-U12(I)*WRK(I,2,M)
     1           -U13(I)*WRK(I,3,M)-U14(I)*DX4
C
    4 CONTINUE
C
C ------------  COMPUTE AA,BB,CC FOR NEXT ITERATION ------------
C
      CALL FILLMY(J+1)
C
C ------------ SAVE GAMMA(I,J)'S ON JACOBIAN MATRIX A ------------
C
      DO 5 L=1,4
      DO 5 M=1,4
      DO 5 I=2,IL1
        A(I,J,L,M)=WRK(I,L,M)
    5 CONTINUE
C
    1 CONTINUE
C
C-----------------------------------------------------------------------
C                PART 2.     BACKWARD  SWEEP
C-----------------------------------------------------------------------
C
      JJE=JL1-2
      DO 6 JJ=1,JJE
      J=JL1-JJ
C
      DO 7 L=1,4
      DO 7 I=2,IL1
C
      RHS(I,J,L)=RHS(I,J,L)-A(I,J,L,1)*RHS(I,J+1,1)
     1           -A(I,J,L,2)*RHS(I,J+1,2)
     2           -A(I,J,L,3)*RHS(I,J+1,3)
     3           -A(I,J,L,4)*RHS(I,J+1,4)
C
    7 CONTINUE
C
    6 CONTINUE
C
      RETURN
      END
C
C*****************************************************************
C
C SUBROUTINE "FILLMY" FORMS 4X4 BLOCKS AA, BB, CC
C REQUIRED IN ETA - SWEEP (INCLUDES IMPLICIT DAMPING)
      SUBROUTINE FILLMY (J)
C
      PARAMETER(IM=299,JM=100)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM,
      COMMON /MATRIX/ A(IM,JM,4,4),RHS(IM,JM,4)
      COMMON /PI/ IL,JL,ILE,ITEL,ITEU
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM1I,GKM,S1P,
     -  PINF,SINA,COSA,PRI,PRII,REI,XL,XL1,XLM,PI
      COMMON/TIMEM/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DAMP/ ME,MI,MPE,MPI,ISPECT
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
      COMMON/BTRID/ AA(IM,4,4),BB(IM,4,4),CC(IM,4,4),
     - WRK(IM,4,4),L11(IM),L21(IM),L22(IM),L31(IM),
     - L32(IM),L33(IM),L41(IM),L42(IM),
     - L43(IM),L44(IM),U12(IM),U13(IM),U14(IM),
     - U23(IM),U24(IM),U34(IM)
```

```
      COMMON/CMAT/ RR(IM,4,4),RP1(IM,4,4),RM1(IM,4,4)
C
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,
     -     U12,U13,U14,U23,U24,U34
C
C COMPUTE VISCOUS JACOBIAN MATRIX 'R'
      JP1=J+1
      JM1=J-1
      IF ( J .GT. 2 ) GOTO 100
      CALL CMATS(J,RR)
      CALL CMATS(JM1,RM1)
      GOTO 200
  100 DO 11 M=1,4
      DO 11 L=1,4
      DO 11 I=2,IL1
      RM1(I,L,M) = RR(I,L,M)
      RR(I,L,M) = RP1(I,L,M)
   11 CONTINUE
C
  200 CONTINUE
      CALL CMATS(JP1,RP1)

      DO 1 L=1,4
C
      DO 2 M=1,4
      DO 2 I=2,IL1
        AA(I,L,M)=2.*RR(I,L,M)
        BB(I,L,M)=-(.5*A(I,J-1,L,M)+RM1(I,L,M))
        CC(I,L,M)=.5*A(I,J+1,L,M)-RP1(I,L,M)
    2 CONTINUE
C
C ------------  ADD IMPLICIT DAMPING ------------
C
      DO 3 I=2,IL1
      WISP=SPECT(I,J)*(WI+WPI*PSMU(I,J))
      AA(I,L,L)=AA(I,L,L)+2.*WISP/XJAC(I,J)
      BB(I,L,L)=BB(I,L,L)-WISP/XJAC(I,J-1)
      CC(I,L,L)=CC(I,L,L)-WISP/XJAC(I,J+1)
    3 CONTINUE
C
C ------------  MULTIPLY  BY DTAU(I,J) ------------
      DO 4 M=1,4
      DO 4 I=2,IL1
      AA(I,L,M)=DTAU(I,J)*AA(I,L,M)
      BB(I,L,M)=DTAU(I,J)*BB(I,L,M)
      CC(I,L,M)=DTAU(I,J)*CC(I,L,M)
    4 CONTINUE
C
C ------------ ADD IDENTITY MATRIX TO  AA (DIAGONAL) -----
      DO 5 I=2,IL1
      AA(I,L,L)=AA(I,L,L)+1.0
    5 CONTINUE
C
    1 CONTINUE
C
      RETURN
      END

C
C ================================================================
C
```

```
C SUBROUTINE "BNDRY" IMPLEMENTS BOUNDARY CONDITIONS
C CASE : VISCOUS FLOW, AIRFOIL C-GRID
      SUBROUTINE BNDRY
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TW,S1,ALFA,AMGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAMI1,GKM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      COMMON/BND/ NBND,XMREF,TREF,TINF,CV,CR,CT,DEL,L1,L2,L3,L4
C
C ---------------------------------------------------------------
C DOWNSTREAM BOUNDARIES (XI=0.,XIMAX) : SUBSONIC OUTFLOW
C PEXIT SPECIFIED, EXTRAPOLATION FOR RHO,U,V.
C ---------------------------------------------------------------
      DO 12 J=2,JL1
      U(1,J)=U(2,J)
      V(1,J)=V(2,J)
      P(1,J)=PINF
      RHO(1,J)=RHO(2,J)
      EDDY(1,J)=EDDY(2,J)
   12 CONTINUE
C
      DO 22 J=2,JL1
      U(IL,J)=U(IL1,J)
      V(IL,J)=V(IL1,J)
      P(IL,J)=PINF
      EDDY(IL,J)=EDDY(IL1,J)
      RHO(IL,J)=RHO(IL1,J)
   22 CONTINUE
C
C ---------------------------------------------------------------
C ETA = ETAMAX, FREESTREAM VALUES SPECIFIED
C ---------------------------------------------------------------
      DO 25 I=1,IL
      U(I,JL)=COSA
      V(I,JL)=SINA
      P(I,JL)=PINF
      RHO(I,JL)=1.0
   25 CONTINUE
C
C ---------------------------------------------------------------
C AIRFOIL SURFACE AND MAKE CUT
C ---------------------------------------------------------------
C MAKE-CUT : AVERAGED EXTRAPOLATION
      DO 3 I=1,ITEL
      K=IL-I+1
      U(I,1)=.9*(U(I,2)+U(K,2))
      V(I,1)=.5*(V(I,2)+V(K,2))
      P(I,1)=.5*(P(I,2)+P(K,2))
      RHO(I,1)=.5*(RHO(I,2)+RHO(K,2))
      U(K,1)=U(I,1)
      V(K,1)=V(I,1)
      P(K,1)=P(I,1)
```

```
      RHO(K,1)=RHO(I,1)
C NEW BOUNDARY CONDITIONS FOR SYMETRICAL AIRFOIL IN THE WAKE AT ALFA = 0
C     V(I,1) = 0.0
C     V(K,1) = 0.0
    3 CONTINUE
C
      EDDY(1,1)=EDDY(2,1)
      EDDY(IL,1)=EDDY(1,1)
C
C AIRFOIL SURFACE: U=V=0.0; DT/DETA=0.(ADIABATIC WALL);
C                  DP/DETA=0.0
      DO 4 I=ITELP1,ITEUM1
      U(I,1)=0.0
      V(I,1)=0.0
      P(I,1)=P(I,2)
      DXI=GXM*(-P(I,3)/RHO(I,3)+4.*P(I,2)/RHO(I,2))/3.0
      RHO(I,1)=GXM*P(I,1)/DXI
    4 CONTINUE

      IF ( NBND .EQ. 0 ) GOTO 7
C
C BOUNDARY CONDITIONS FOR SUCTION ON UPPER SURFACE
C DX1=THETA

      XMR  = XMREF/XM1
      TRS  = SQRT( TREF/TINF )
      VIN  = CV * XMR * TRS
      AIN  = ABS( X(L3,1) - X(L4,1) )
      AOUT = ABS( X(L2,1) - X(L1,1) )
      DX2 = 0.0
      DX4 = 0.0
      DX5 = 0.0

      DO 5 I=L3,L4
        DX1 = ( Y(I+1,1) - Y(I-1,1) ) / ( X(I+1,1) - X(I-1,1) )
        DX1 = ATAN(DX1)
        IF ( NBND .EQ. 3 ) GOTO 17
        U(I,1) = VIN * SIN(DX1)
        V(I,1) = -VIN * COS(DX1)
 17     DX2 = DX2 + RHO(I,1)
C       DX3=GXM*(-P(I,3)/RHO(I,3)+4.*P(I,2)/RHO(I,2))/3.
C       DX4 = DX4 + DX3
C       DX5 = DX5 + P(I,1)
    5 CONTINUE
C
      XU = ABS( FLOAT( L4-L3+1 ) )
C     TAVG = DX3 / XU
C     PAVG = DX5 / XU
      RAVG = DX2 / XU
C     MDOT = RAVG * AIN * VIN
      VOUT = AIN/(AOUT*SIN(DEL))* VIN / CR
C
C  BOUNDARY FOR LOWER SURFACE EJECTION
      IF ( NBND .EQ. 2 ) GOTO 7
C
      DO 6 I=L1,L2
        RHO(I,1) = RAVG*CR
C       P(I,1) = CR*CT*PAVG
C       T = CT * TAVG
        U(I,1) = VOUT * COS(DEL)
        V(I,1) = -VOUT * SIN(DEL)
    6 CONTINUE

C ------- TRAILING EDGE ----------------------
    7 U(ITEL,1)=0.0
      V(ITEL,1)=0.0
      U(ITEU,1)=0.0
      V(ITEU,1)=0.0
      P(ITEL,1)=P(ITEL,2)
      P(ITEU,1)=P(ITEU,2)
      T=GXM*(-P(ITEL,3)/RHO(ITEL,3)+4.*P(ITEL,2)/RHO(ITEL,2))/3.
      RHO(ITEL,1)=GXM*P(ITEL,1)/T
      T=GXM*(-P(ITEU,3)/RHO(ITEU,3)+4.*P(ITEU,2)/RHO(ITEU,2))/3.
      RHO(ITEU,1)=GXM*P(ITEU,1)/T
C
      RETURN
      END

C
C ***************************************************************
C
C SUBROUTINE 'TURB' EVALUATES THE TURBULENT EDDY VISCOSITY
C USING THE ALGEBRAIC MODEL OF BALDWIN & LOMAX AND THE
C TRANSITION MODEL OF DHAWAN & NARASIMHA.
C LIST OF SYMBOLS
C EDDY : TURBULENT EDDY VISCOSITY
C XMUTI : EDDY VISCOSITY FOR INNER REGION
C XMUTO : EDDY VISCOSITY FOR OUTER REGION
C F : FUNCTION IN OUTER FORMULA
C GAMMA : TRANSITION FACTOR
C TAUW : SHEAR STRESS AT THE WALL
C JLAST : SPECIFIES THE EXTENT IN THE ETA-DIRECTION TO WHICH
C         THE EDDY VISCOSITY IS COMPUTED.
C IPLOT:
C       = 0 NO PRINTED OUTPUT IS PROVIDED
C       = 1 PRINTED OUTPUT IS PROVIDED
C
      SUBROUTINE TURB (IPLOT,IPLOT1)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TM,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TURB1/ITRSU,ITRSL,IMKST,IMKE,JLAST,
     -             XRELAX
      COMMON/DUMMY/ TEMP(IM,JM,6)
      COMMON/TURB2/XMUTI(JM),XMUTO(JM),D(JM),N(2*JM),F(2*JM),
     -      GAMMA(IM)
C
C SPECIFY CONSTANT PARAMETERS
      VK=0.40
      APLUS=26.0
      CKLEB=.30
      CLAU=.01680
      CCP=1.60
C MAKE CONSTANTS
```

Appendix G: Hyperbolic Grid Generating Code, Fortran Listing

The attached fortran listing is the original code used to generate the grids in this work as supplied by the Air Force Wright Aerodynamics Laboratories, Flight Dynamics Laboratory, Computational Fluid Dynamics Group. The code was developed jointly by Dr. Timothy Barth and Dr. Don Kinsey an described in reference (19). The code was run on the Cyber computer.

```
      PROGRAM MAIN
C
C          THIS PROGRAM GENERATES HYPERBOLIC C-MESHES ABOUT AIRFOILS
C          WRITTEN BY TIM BARTH
C
      OPEN(6,FILE='OUTPUT')
      OPEN(5,FILE='INPUT')
       CALL STARTE
       CALL FOIL
       CALL PARAME
       CALL STRETC
       CALL MAKE
       CALL GRIDGE
       STOP
       END
C=======================================================================
C
      SUBROUTINE STARTE
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
       COMMON /UNITS / BODYUN,GRIDUN,DOCUNI
       COMMON /FNAMES/ BODYFI,GRIDFI,DOCFIL
       COMMON /STRET/ DS(100),CNTRLP(100),NBP(100),ARCLEN(301)
       CHARACTER*80 BODYFI,GRIDFI,DOCFIL
       INTEGER BODYUN,GRIDUN,DOCUNI
C
C                  SET UP DOCUMENTATION FILE
C
C
 1000    FORMAT(A)
 10      CONTINUE
         WRITE(6,*)' INPUT DOC FILE NAME '
         READ(5,1000)DOCFIL
         DOCUNI  = 11
       OPEN(11,FILE=DOCFIL)
 20      CONTINUE
         WRITE(6,*)' INPUT GRID FILE NAME  '
         READ(5,1000)GRIDFI
         GRIDUN =  9
        OPEN(9,FILE=GRIDFI)
         RETURN
         END
C=======================================================================
C
      SUBROUTINE FOIL
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
       COMMON /UNITS / BODYUN,GRIDUN,DOCUNI
       COMMON /FNAMES/ BODYFI,GRIDFI,DOCFIL
       COMMON /STRET/ DS(100),CNTRLP(100),NBP(100),ARCLEN(301)
       CHARACTER*80 BODYFI,GRIDFI,DOCFIL
       CHARACTER*1 ANSWER
       INTEGER BODYUN,GRIDUN,DOCUNI
C
C                  SET UP FILE
C
C
 1000  FORMAT(A)
       WRITE(6,*)' DO YOU HAVE BODY COORDINATES ON FILE (Y/N)?'
       READ(5,1000)ANSWER
       IF(ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y')THEN
C
C
         WRITE(6,*)' INPUT BODY FILE NAME '
         READ(5,1000)BODYFI
         BODYUN = 10
       OPEN(10,FILE=BODYFI)
  5      CONTINUE
         KOUNT = 0
 10      CONTINUE
         READ(BODYUN,*,END=100)X(KOUNT+1),Y(KOUNT+1)
         KOUNT=KOUNT+1
         GO TO 10
 100     CONTINUE
         NBODY = KOUNT
         WRITE(DOCUNI,2000)BODYFI,NBODY
         WRITE(6,2000)BODYFI,NBODY
 2000    FORMAT(' BODY FILE WAS USER INPUT ',A7,' WITH',I5,' POINTS')
         CLOSE (BODYUN
C
       ELSE
         WRITE(6,*)' **********NACA00XX AIRFOIL**********'
         WRITE(6,*)' **********CLOSED TE NORMALIZED**********'
         WRITE(6,*)' INPUT XX'
         READ(5,*)TMAX
         WRITE(DOCUNI,*)' NACA00XX AIRFOIL: XX = ',TMAX
         TMAX = TMAX/100.
         NPTS = 300
         NIMALF = NPTS/2 + 1
         NIMM   = NIMALF - 1
         XINCPT = 1.0089304611365
         DELX   = 1./FLOAT(NIMM)
         XX     = 1. + DELX
         DO 20 I=1,NIMM
         XX = XX -DELX
         X1 = .5*XINCPT*(1.0 - COS(3.14159265358*XX))
         X(I) = X1
         Y(I) = -5.*TMAX*(0.2969*SQRT(X1) - .126*X1-0.3516*X1*X1
     $          + 0.2843*X1**3 -0.1015*X1**4)
 20      CONTINUE
         DO 21 I=1,NIMM
         II = NIMALF + I
         IR = NIMALF - I
         X(II) = X(IR)
         Y(II) = -Y(IR)
 21      CONTINUE
         NPTS = NPTS + 1
         DO 22 I=1,NPTS
         X(I) = X(I)/XINCPT
         Y(I) = Y(I)/XINCPT
 22      CONTINUE
         NBODY = NPTS
       ENDIF
       RETURN
       END
C=======================================================================
C
```

```
      SUBROUTINE PARAME
C
C-----------------------------------------------------------------------
      COMMON BASE JMAX,KMAX,FI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON VARS X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(302)
      COMMON /UNITS / BODYUN,GRIDUN,DOCUNI
      COMMON /FNAMES, BODYFI,GRIDFI,DOCFIL
      COMMON /STRET, DS(100),CNTRLP(100),NBP(100),ARCLEN(301)
      DIMENSION XS(100)
      CHARACTER*5 LOWER,UPPER,XXXXX
      CHARACTER*80 BODYFI,GRIDFI,DOCFIL
      INTEGER BODYUN,GRIDUN,DOCUNI
      LOWER = 'LOWER'
      UPPER = 'UPPER'
C
C
C           FIND LE
C
      XMIN =1000.
      DO 9876 J=1,NBODY
       IF(X(J).LT.XMIN)THEN
         XMIN = X(J)
         JMIN = J
       ENDIF
 9876 CONTINUE
      JLE = JMIN
C
C
      NCLUST = 0
      WRITE(6,*)'*****AIRFOIL LOWER SURFACE*****'
   10 CONTINUE
      WRITE(6,*)' INPUT X LOCATION, MIN ARCLENGTH SPACING,',
     >     ' RETURN TO CONTINUE'
      READ(5,*,END=20)XS(NCLUST+1),DS(NCLUST+1)
      NCLUST = NCLUST+1
      GO TO 10
   20 REWIND 5
C
C       BRUTE FORCE  BUBBLE SORT
C
      DO9875 M=1,NCLUST
      DO9874 N=1,NCLUST
      IF(XS(M).LT.XS(N))THEN
         TEMP1 = XS(M)
         TEMP2 = DS(M)
         XS(M) = XS(N)
         DS(M) = DS(N)
         XS(N) = TEMP1
         DS(N) = TEMP2
      ENDIF
 9874 CONTINUE
9875 CONTINUE
C
C           FIND NEAREST BODY POINT
C
      DO9873 N=1,NCLUST
      RMAX = 1.E6
        DO9872 J=1,JLE
         RR = ABS(XS(N)-X(J))
         IF(RR.LT.RMAX)THEN
            RMAX = RR
            JJ = J
         ENDIF
 9872   CONTINUE
        CNTRLP(N) = JJ
 9873 CONTINUE
      NLOWER = NCLUST
      WRITE(6,*)'*****AIRFOIL UPPER SURFACE*****'
   30 CONTINUE
      WRITE(6,*)' INPUT X LOCATION, MIN ARCLENGTH SPACING,',
     >     ' RETURN TO CONTINUE'
      REWIND 5
      READ(5,*,END=40)XS(NCLUST+1),DS(NCLUST+1)
      NCLUST = NCLUST+1
      GO TO 30
   40 REWIND 5
C
C       BRUTE FORCE  BUBBLE SORT
C
      DO9871 M=NLOWER+1,NCLUST
      DO9870 N=NLOWER+1,NCLUST
      IF(XS(M).GT.XS(N))THEN
         TEMP1 = XS(M)
         TEMP2 = DS(M)
         XS(M) = XS(N)
         DS(M) = DS(N)
         XS(N) = TEMP1
         DS(N) = TEMP2
      ENDIF
 9870 CONTINUE
 9871 CONTINUE
C
C           FIND NEAREST BODY POINT
C
      DO9869 N=NLOWER+1,NCLUST
      RMAX = 1.E6
        DO9868 J=JLE,NBODY
         RR = ABS(XS(N)-X(J))
         IF(RR.LT.RMAX)THEN
            RMAX = RR
            JJ = J
         ENDIF
 9868   CONTINUE
        CNTRLP(N) = JJ
 9869 CONTINUE
C
C
      ARCLEN(1) = 0
      DO9866 I=1,NBODY-1
        ARCLEN(I+1) = ARCLEN(I) +
     &       SQRT((X(I)-X(I+1))**2 + (Y(I)-Y(I+1))**2)
 9866 CONTINUE
C
C
      WRITE(DOCUNI,*)' ******** CLUSTERINGS ********'
      DO9865 I=2,NCLUST
        IB1 = CNTRLP(I)
        IB2 = CNTRLP(I-1)
        XXXXX = UPPER
        IF(I.LE.NLOWER)XXXXX = LOWER
        ALENGT = ARCLEN(IB1)-ARCLEN(IB2)
        WRITE(6,2000)X(IB2),X(IB1),ALENGT,XXXXX
```

```
2000    FORMAT(' INPUT NUMBER OF POINTS FOR THE INTERVAL X = ',
     >         F12.6,'  TO   X = ',F12.6,/,
     >         ' ARCLENGTH =',F12.6,' ON THE ',A,'  SURFACE')
        READ(5,*)NBP(I-1)
        WRITE(DOCUNI,3000)X(IB2),X(IB1),NBP(I-1),XXXXX,DS(I),
     >                    DS,I-1
3000    FORMAT(' NUMBER OF POINTS FOR THE INTERVAL X = ',
     >         F12.6,'  TO   X = ',F12.6,',' IS ',I5,
     >         ' ON THE ',A,'  SURFACE',/,
     >         ' MIN SPACING ON ENDS ARE',F14.7,1X,F14.7)
 9865 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE MAKE
C
C-----------------------------------------------------------------------
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SNUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
      COMMON /UNITS/ BODYUN,GRIDUN,DOCUNI
      COMMON /FNAMES/ BODYFI,GRIDFI,DOCFIL
      COMMON  STRET  DS(100),CNTRLP(100),NBP(100),ARCLEN(301)
      CHARACTER*1 ANSWER
      CHARACTER*80 BODYFI,GRIDFI,DOCFIL
      INTEGER BODYUN,GRIDUN,DOCUNI
      COMMON /BOD/ XINTER(301),YINTER(301),IMAX
      COMMON /SOBO/SOB
      DIMENSION KNAKE(301),YNAKE(301),XX(10),YY(10)
C
C
C
      WRITE(6,*)'=====OUTER DOMAIN LIMITS====='
      WRITE(6,*)' INPUT DISTANCE TO OUTER BOUNDARY'
      READ(5,*)SOB
      WRITE(DOCUNI,*)' DISTANCE TO OUTER BOUNDARY = ',SOB
      WRITE(6,*)' INPUT NUMBER OF POINTS IN THE WAKE REGION'
      READ(5,*)NWAKE
      WRITE(DOCUNI,*)' NWAKE = ',NWAKE
C
C       SLOPE ESTIMATION
C
      THETA1 = ATAN2(YINTER(IMAX)-YINTER(IMAX-1),
     >               XINTER(IMAX)-XINTER(IMAX-1))
      THETA2 = ATAN2(YINTER(1)-YINTER(2),
     >               XINTER(1)-XINTER(2))
       SLOPE = TAN(.5*(THETA1+THETA2))
      WRITE(6,*)' TRAILING EDGE SLOPE = ',SLOPE
      WRITE(6,*)
     >    ' DO YOU WISH TO CHANGE SLOPE ESTIMATION <Y/N>?  '
      READ(5,'(A1)')ANSWER
      IF(ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'Y')THEN
       WRITE(6,*)' INPUT NEW SLOPE='
       READ(5,*)SLOPE
      ENDIF
      YOPRIM = SLOPE
      Y1PRIM = 0.00000
C
C       START CUBIC PATCH AT X/C=1.05. END CUBIC AT X/C=1.5.
C
      XO = 1.010
      YO = YINTER(1) + YOPRIM*(XO-XINTER(1))
      X1 = 1.2
      Y1 = 2.*YO-YINTER(1)
C     (YINTER(1) + YOPRIM*(XO-XINTER(1)))/2.
      WRITE(6,*)' XO,YO,X1,Y1 =',XO,YO,X1,Y1
C
C       FIND COEFFICIENTS A,B,C,D FOR POLYNOMIAL Y = A X**3 + B X**2 + C X + D
C
      T1 = XO    - X1
      T2 = XO**2 - X1**2
      T3 = XO**3 - X1**3
      T4 = YOPRIM - Y1PRIM
      T5 = YO - Y1
      TL1 = 3.*T2
      TL2 = 2.*T1
      TL3 = 3.*T1*X1**2 - T3
      TL4 = 2.*X1*T1 - T2
      TL5 = Y1PRIM*T1 - T5
      D1  = TL1*TL4 - TL2*TL3
      IF( ABS(D1) .LT. 1.E-26)THEN
          WRITE(6,*)' CAUTION MAKE CALCULATION D1 IN ERROR '
          D1 = 1.E-26*SIGN(1.,D1)
      ENDIF
      A   = (T4*TL4 - TL2*TL5)/D1
      D2 = TL4
      IF  ABS(D2  .LT. 1.E-26)THEN
          WRITE(6,*)' CAUTION MAKE CALCULATION D2 IN ERROR '
          D2 = 1.E-26*SIGN(1.,D2)
      ENDIF
      B   = (TL5 - A*TL3)/D2
      D3 = T1
      IF( ABS(D3) .LT. 1.E-26)THEN
          WRITE(6,*)' CAUTION MAKE CALCULATION D3 IN ERROR '
          D3 = 1.E-26*SIGN(1.,D3)
      ENDIF
      C = (T5 - A*T3 - B*T2)/D3
      D = Y1 - A*X1**3 - B*X1**2 - C*X1
      WRITE(6,*)' A,B,C,D = ',A,B,C,D
     EPS = EPSIL(SOB,0.0,DS(1),NWAKE,.00001,40,1)
     XWAKE(1) = XINTER(1)
     YWAKE(1) =(YINTER(1)+ YINTER(IMAX))*0.50
     DO 10 K=2,NWAKE
     XWAKE(K) = XWAKE(K-1) + DS(1)*(1.+EPS)**(K-2)
     IF(XWAKE(K) .LE. XO)THEN
        YWAKE(K) = YWAKE(1) + YOPRIM*(XWAKE(K)-XWAKE(1))
     ELSEIF( XWAKE(K) .GT. X1)THEN
        YWAKE(K) = Y1
     ELSE
        Z = XWAKE(K)
        YWAKE(K) = A*Z**3 + B*Z**2 + C*Z + D
     ENDIF
      WRITE(6,*)K,XWAKE(K),YWAKE(K)
  10 CONTINUE
C
C
C       PIECE EVERYTHING TOGETHER
C
      DO9864 I=1,NWAKE
        X I = XWAKE(NWAKE-I+1)
        Y I = YWAKE(NWAKE-I+1)
```

```
 9864 CONTINUE
      DO9863 I=NMAKE+1,NMAKE+1+IMAX
        X(I)  = XINTER(I - NMAKE + 1
        Y(I)  = YINTER(I - NMAKE + 1
 9863 CONTINUE
      DO9862 I=NMAKE+1+IMAX-2,NMAKE+1+IMAX+NMAKE-2
        X(I) = XMAKE(I-(NMAKE+1+IMAX-3)
        Y(I) = YMAKE(I-(NMAKE+1+IMAX-3)
 9862 CONTINUE
      JMAX = 2*NMAKE+IMAX-2
      JTAIL1 = NMAKE
      JTAIL2 = JMAX-JTAIL1+1
      WRITE(6,*) ' JMAX   = ',JMAX
      WRITE(6,*) ' JTAIL1 = ',JTAIL1
      WRITE(6,*) ' JTAIL2 = ',JTAIL2
      WRITE(DOCUN1,*) ' JMAX   = ',JMAX
      WRITE(DOCUN1,*) ' JTAIL1 = ',JTAIL1
      WRITE(DOCUN1,*) ' JTAIL2 = ',JTAIL2
C
C
      END
C=======================================================================
      FUNCTION EPSIL FMX,FMIN,DFM,NPT,EPCC,ICC,    NCALL
C=======================================================================
C
C        THIS SUBROUTINE APPLIES A NEWTON-RAPHSON ROOT-FINDING
C        TECHNIQUE TO FIND A VALUE OF EPSILON FOR A PARTICULAR USE
C        OF THE EXPONENTIAL STRECHING TRANSFORMATION.
C
C   FMX IS TOTAL ARC LENGTH ALONG COORDINATE
C   FMIN IS STARTING VALUE OF ARC LENGTH   SUCH AS 0.0
C   DFM IS SPECIFIED INITIAL INCREMENT OF ARC LENGTH
C   NPT IS NUMBER OF POINTS ALONG COORDINATE
C   EPCC IS ITERATIVE ERROR BOUND, E.G.C  0.00002
C   ICC IS MAXIMUM NUMBER OF ITERATIONS
C   NCALL    IF NCALL=1 INITIAL GUESS FOR EPS IS USED
C            IF NCALL .GT. 1, PREVIOUS EPS USED AS INITIAL GUESS
C
C
      FMXL=FMX
      FMINL=FMIN
      DFML=DFM
      EPCCL=EPCC
      ICCL=ICC
C
C
      FNPTM2=NPT-2
      IF(NCALL.EQ.1   EPS=((FMXL/DFML)**(1.0/FNPTM2))-1.0
C
      DO 3 NIT=1,ICCL
      EP1=EPS+1.0
      EP1TN=EP1**FNPTM2
      REPS=1.0/EPS
      DFMOE=DFML*REPS
      F=FMXL-FMINL-DFMOE*(EP1TN-EP1-1.0)
      IF(ABS(F).LT.EPCCL)  GO TO 4
      DFMOE2=DFMOE*REPS
      FPN=DFMOE2*(1.0+EP1TN*(EPS*FNPTM2-1.0))
      DD = F/FPN
-----------------------------------------------------------------------
      EPS=EPS+F/FPN
3     CONTINUE
C
5     EPSIL=EPS
      WRITE(1,100)
      RETURN
C
4     EPSIL=EPS
      RETURN
C
100   FORMAT(/42H EXCEEDED MAX. NO. OF ITERATIONS IN EPSIL
101   FORMAT(7H EPSIL=,F12.5,5X,7H AND F=,F12.5,5X,7H AFTER ,I3,
     1  12H ITERATIONS.)
C
      END
C=======================================================================
C
C      SUBROUTINE STRETC
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XPI(301),XETA(301),YKI(301),
     1 XETA(301),R(301),ZZ(301),SCALE(10),SF(301,2),VOL(301)
      COMMON/UNITS/BODYUN,GRIDUN,DOCUN
      COMMON/FNAMES/BODFIL,GRIDFIL,DOCFIL
      COMMON/STRET/DS(10),CNTRLP(10),NBP(10),ARCLEN(301)
      CHARACTER*1 ANSWER
      CHARACTER*80 BODFIL,GRIDFIL,DOCFIL
      INTEGER BODYUN,GRIDUN,DOCUN
      COMMON/BOD/XINTER(301),YINTER(301),IMAX
      DIMENSION S(301),ARCDIS(301)
C
C
      S(1)=0
      NL = 1
      DO9861 N=1,NCLUST-1
         NODE1 = CNTRLP(N+1)
         NODE0 = CNTRLP(N)
         ALENGT = ARCLEN(NODE1)-ARCLEN(NODE0)
         DS0= DS(N)/ALENGT
         DS1= DS(N+1)/ALENGT
         CALL CLUST(ARCDIS,DS0,DS1,NL,NL+NBP(N),1)
         DO9860 I=NL+1,NL+NBP(N)
         S(I) = ARCLEN(NODE0) + ALENGT*ARCDIS(I)
 9860    CONTINUE
         NL = NL+NBP(N)
 9861 CONTINUE
      IMIN = 1
      IMAX = NL
      WRITE(6,*) ' DO YOU WISH TENSIONED SPLINE FITS (Y/N)?'
      READ(5,1000)ANSWER
1000  FORMAT(A)
      IF(ANSWER .EQ. 'N' .OR. ANSWER .EQ. 'n')THEN
      CALL CSPLIN(S,XINTER,ARCLEN,X,IMIN,IMAX,1,NBODY)
      CALL CSPLIN S,YINTER,ARCLEN,Y,IMIN,IMAX,1,NBODY
      ELSE
      WRITE(6,*) ' INPUT TENSION '
      READ(5,*)  SIGMA
      CALL  TSPLIN S,XINTER,ARCLEN,X,NBODY,IMAX,SIGMA
      CALL  TSPLIN S,YINTER,ARCLEN,Y,NBODY,IMAX,SIGMA
      ENDIF
```

```
C
C
C
      RETURN
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE CLUST(Y,DS0,DS1,JMIN,JMAX)
C
C-----------------------------------------------------------------------
      DIMENSION Y(301)
      DEL = 1./(JMAX-JMIN)
      S0=DEL/DS0
      S1=DEL/DS1
      CALL CLUST2(Y,S0,S1,JMIN,JMAX)
C
C           ONE STEP CORRECTOR
C
      ALPHA0 = (Y(JMIN+1)-Y(JMIN))/DS0
      ALPHA1 = (Y(JMAX)-Y(JMAX-1))/DS1
      S0=ALPHA0*DEL/DS0
      S1=ALPHA1*DEL/DS1
      CALL CLUST2(Y,S0,S1,JMIN,JMAX)
      RETURN
      END
C*********************************************************************
C
      SUBROUTINE CLUST2(Y,S0,S1,JMIN,JMAX)
C
C*********************************************************************
C
C   VINOKUR'S END POINT CLUSTERING FUNCITON
C
C   SHOULD BE RUN DOUBLE PRECISION ON VAX OR IBM
C
C   Y(J) IS STRETCED FUNCTION BETWEEN LIMITS OF 0 AND 1
C            I.E. IT IS NORMALIZED AND MUST BE RESCALED
C
CDOUBLE      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1)
      DATA EPS/.001/
      JM1=JMAX-1
      JP1=JMIN+1
      DEL=1./(JMAX-JMIN)
      B=SQRT(S0*S1)
      A=B/S1
      OA=1./A
      Y(JMAX)=1.0
      Y(JMIN)=0.0
      IF(B.GT.1.+EPS)GO TO 10
      IF(B.GT.1.-EPS)GO TO 20
      DZ=ASINHF(B)
      COSDZ=COS(DZ)
      CSCDZ=1./SQRT(1.-COSDZ*COSDZ)
      WO=CSCDZ*(COSDZ-OA)
      ZO=ATAN(WO)
      DW=CSCDZ*(A-COSDZ)-WO
      ODW=1./DW
      DO  1 J=JP1,JM1
    1 Y(J)=ODW*(TAN(DZ*(J-JMIN)*DEL+ZO)-WO)
      RETURN
C
   10 DZ=ASINHF(B)
      COSHDZ=COSH(DZ)
      OMACDZ=1.-A*COSHDZ
      ASINDZ=A*SQRT(COSHDZ*COSHDZ-1.)
      DO 11 J=JP1,JM1
      U=TANH(DZ*(J-JMIN)*DEL)
   11 Y(J)=U/(ASINDZ+OMACDZ*U)
      RETURN
   20 TWOBMO=2.*(B-1.)
      ONEMA=1.-A
      DO 21 J=JP1,JM1
      X=(J-JMIN)*DEL
      U=X*(1.+TWOBMO*(X-.5)*(1.-X))
   21 Y(J)=U/(A+ONEMA*U)
      RETURN
      END
C*********************************************************************
      FUNCTION ASINHF(U)
CDOUBLE      IMPLICIT REAL*8(A-H,O-Z)
      DATA A1,A2,A3,A4,A5/-.15,.0573214285714,-.0249072948 78,.0077424660
     1899,-.0010794122691/
      DATA B0,B1,B2,B3,B4/-.020617693089 2,.2490272170591,1.9496443322775
     1,-2.6294547252,1.8.5679591096315
      DATA U1,U2/2.782968117860 3,35.0539798452776/
      IF(U-U1)1,1,2
    1 UB=U-1.
      ASINHF=SQRT(6.*UB)*(((((A5*UB+A4)*UB+A3)*UB+A2)*UB+A1)*UB+1.)
      RETURN
CDOUBLE    2 V=DLOG(U)
    2 V=LOG(U)
      W=1./U-1./U2
CDOUBLE      ASINHF=V+DLOG(2.*V)*(1.+1./V)+(((B4*W+B3)*W+B2)*W+B1)*W+B0
      ASINHF=V+LOG(2.*V)*(1.+1./V)+(((B4*W+B3)*W+B2)*W+B1)*W+B0
      RETURN
      END
C*********************************************************************
      FUNCTION ASINHF(U)
CDOUBLE      IMPLICIT REAL*8(A-H,O-Z)
      DATA A1,A2,A3,A4,A5/.15,-.0573214285714,.0489742834696,
     1-.0533377753213,.0758451335824/
      DATA B3,B4,B5,B6/-2.644934068482,6.7947319658321,
     1-13.2055008110734,11.7260952338351/
      DATA U1,PI/.2693897165164,3.1415926535898 1/
      IF(U.LE.U1)GO TO 1
      UB=1.-U
      ASINHF=SQRT(6.*UB)*(((((A5*UB+A4)*UB+A3)*UB+A2)*UB+A1)*UB+1.)
      RETURN
    1 ASINHF=PI*(((((B6*U+B5)*U+B4)*U+B3)*U+1.)*U-1.)*U+1.
      RETURN
      END
      SUBROUTINE CSPLIN(XX,YY,X,Y,N1,N2,J1,J2)
      DIMENSION XX(1),YY(1),X(1),Y(1)
      COMMON/SCRACH/A(301),B(301),C(301),D(301),F(301),H(301)
C
C           CUBIC SPLINE INTERPOLATION
C  X,Y ARRAYS ARE TO BE INTERPOLATED
C  YY ARE FOUND INTERPOLATES CORRESPONDING TO XX
C  J1..J2. ARE INDICE LIMITS ON X,Y
C  N1, N2 ARE INDICE LIMITS ON XX : ALSO YY
C
C     FIRST FIND DERIVATIVE LIKE TERMS THAT ARE COEFFICIENTS
```

```
      JA = J1 + 1
      JB = J2-1
      DO 10 J=JA,J2
      H(J  ) = X(J  ) - X(J-1)
   10 D(J  ) = Y(J  ) - Y(J-1)/H(J)
      DO 12 J=JA,JB
      A(J) = H(J+1)
      B(J) = 2.*(H(J) + H(J+1))
   12 F(J) = 3.*(H(J)*D(J+1) + H(J+1)*D(J))
      B(J1) = 2.
      H(J1) = 1.
      F(J1) = 3.*D(JA)
      A(J2) = 1.
      B(J2) = 2.
      F(J2) = 3.*D(J2)
      CALL TRIB(A,B,H,C,F,J1,J2,1)
C
C  INTERPOLATION . X(J) ARRAY MUST BE MONOTONE
      FUDGE=1.E-6*ABS(X(J2)-X(J1))
      J = J1
      I = J1 - 1
      DO 20 N=N1,N2
   28 IF(XX(N)-FUDGE.LE.XX(N).AND.XX(N)+FUDGE.GE.XX(N)) GO TO 30
      IF(XX(N) - XX(N-1))22,22,24
   22 J = J-1
      I = I-1
      IF(I.GT. J2) GO TO 27
      GO TO 28
   24 J = J+1
      I = I+1
      IF(J.LT. J1) GO TO 27
      GO TO 28
   27 WRITE(6,600)
  600 FORMAT(55HOFATAL ERROR IN CSPLIN.  VALUE AT WHICH WE ARE SUPPOSED,
     1 52H TO INTERPOLATE IS OUTSIDE OF GIVEN TABLE OF VALUES.)
      STOP
   30 T = (XX(N) - X(J))/H(I)
      TT = T - 1
      YY(N) = T*Y(I) + TT*(Y(J) + H(I)*T*TT*((F(J) - D(I))*TT
     1 - (F(I) - D(I))*T))
   20 CONTINUE
      RETURN
      END
      SUBROUTINE TRIB(A,B,C,X,F,NL,NU,JINC)
      DIMENSION A(2),B(2),C(2),X(2),F(2)
C
C        THIS SUBROUTINE SOLVES A TRI-DIAGONAL SYSTEM OF LINEAR
C        EQUATIONS.
C
      X(NL)=C(NL)/B(NL)
      F(NL)=F(NL)/B(NL)
      NLP1=NL+JINC
      DO 1 J=NLP1,NU,JINC
      JM=J-JINC
      T=1./(B(J)-A(J)*X(JM))
      X(J)=C(J)*T
    1 F(J)=(F(J)-A(J)*F(JM))*T
      NUPNL=NU+NL
      DO 2 J1=NLP1,NU,JINC
      J=NUPNL-J1
      JP=J+JINC
    2 F(J)=F(J)-X(J)*F(JP)
      RETURN
      END
C*********************************************************************
C
      SUBROUTINE TSPLIN(XX,YY,X,Y,N,NN,SIGMA)
C
C*********************************************************************
C
C        DRIVER FOR TENSION SPLINE
C
      DIMENSION X(301),Y(301),A(301),B(301),C(301),R(301),
     1    FDP(301)
      DIMENSION XX(301),YY(301)
      CALL SPLINE(N,X,Y,FDP,SIGMA)
      CALL VALSPE(N,X,Y,FDP,NN,XX,YY,SIGMA)
      RETURN
      END
C*********************************************************************
C
      SUBROUTINE SPLINE(N,X,Y,FDP,SIGMA)
C
C*********************************************************************
      DIMENSION X(301),Y(301),A(301),B(301),C(301),R(301),
     1    FDP(301)
C
C          ALAMBD = 1     .....  CANTELEVER END CONDITIONS
C          ALAMBD = 0     .....  PARABOLIC RUNOFF
C
      ALAMBD = 0.
      NM2 = N -2
      NM1 = N -1
      DO 1 I=2,NM1
      DXP = X(I+1) - X(I)
      DXM = X(I)   - X(I-1)
      DYP = Y(I+1) - Y(I)
      DYM = Y(I)   - Y(I-1)
      SINHP = SINH(SIGMA*DXP)
      COSHP = COSH(SIGMA*DXP)
      SINHM = SINH(SIGMA*DXM)
      COSHM = COSH(SIGMA*DXM)
      IF(ABS(DXM).LT. 1.E-8)THEN
        WRITE(6,*)' ERROR...DXM=0.0...I=',I
      ENDIF
      IF(ABS(DXP).LT. 1.E-8)THEN
        WRITE(6,*)' ERROR...DXP=0.0...I=',I
      ENDIF
      A(I) = 1./DXM - SIGMA/SINHM
      B(I) = SIGMA*COSHM/SINHM - 1./DXM + SIGMA*COSHP/SINHP -1./DXP
      C(I) = 1./DXP  - SIGMA/SINHP
      R(I) = SIGMA*SIGMA*(DYP/DXP - DYM/DXM)
    1 CONTINUE
      B(2) = B(2) + ALAMBD*(X(2)-X(1))
      B(NM1) = B(NM1) + ALAMBD*C(NM1)
      DO 2 I=3,NM1
      IF(ABS(B(I-1)).LT.1.E-9)THEN
       WRITE(6,*)'CAUTION.. NEAR ZERO DIAGONAL.. TSPLINE CONTINUES'
       B(I-1)=SIGN(1.,B(I-1))*1.E-10
      ENDIF
      T = A(I)/B(I-1)
      B(I) = B(I) - T*C(I-1)
```

```
      R(I) = R(I) - T*R(I-1)
    2 CONTINUE
      FDP(NM1) = R(NM1)/B(NM1)
      DO 3 I=2,NM2
      NM1 = N-I
      FDP(NM1) =(R(NM1)-C(NM1)*FDP(NM1+1))/B(NM1)
    3 CONTINUE
      FDP(1) = ALAMBD*(FDP(2) -SIGMA*SIGMA*Y(2))
      FDP(N) = ALAMBD*(FDP(NM1) - SIGMA*SIGMA*Y(NM1))
      RETURN
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE VALSPE(N,X,Y,FDP,MN,XX,YY,SIGMA)
C
C-----------------------------------------------------------------------
      DIMENSION X(301),Y(301),FDP(301)
      DIMENSION XX(301),YY(301)
      NM1 = N - 1
      DO 50 J=1,MN
      DO 1 I=1,NM1
      IF(XX(J) .LE. X(I+1))GO TO 10
    1 CONTINUE
   10 DXM = XX(J) - X(I)
      DXP = X(I+1) - XX(J)
      DEL = X(I+1) - X(I)
      SINHPX= SINH(SIGMA*DXP)
      SINHMX= SINH(SIGMA*DEL)
      IF(ABS(SINHMX) .LT. 1.E-10)THEN
        WRITE(6,*)' CAUTION IN SPLINE....SINHMX = ',SINHMX
        SINHMX = 1.E-10*SIGN(1.,SINHMX)
      ENDIF
      SINHMX= SINH(SIGMA*DXM)
      SIG2IN= 1./SIGMA/SIGMA
      YY(J) = FDP(I)*SIG2IN*SINHPX/SINHMX
     $      + Y(I) -FDP(I)*SIG2IN*DXP/DEL
     $      + FDP(I+1)*SIG2IN*SINHMX/SINHMX
     $      + Y(I+1) - FDP(I+1)*SIG2IN*DXM/DEL
   50 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE GRIDGE
C
C-----------------------------------------------------------------------
C
C             DEVELOPED BY  TIM BARTH     X-6417
C             NASA-AMES...WRIGHT-PATTERSEN AFHAL
C             TASK 933
C
C
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XI(301),XETA(301),YXI(301),
     1 YETA(301),RI(301,100),SCALE(100),SR(301,2),VOL(301)
      COMMON/STORE/ XX(301,100) ,YY(301,100
      COMMON /ORDER/ALPHA,ALPHAM,KM1,KM2
      COMMON /UNITS / BODYUN,GRIDUN,DOCUNI
      COMMON /FNAMES/ BODYFI,GRIDFI,DOCFIL
      COMMON /LAX/RELAX
      INTEGER BODYUN,GRIDUN,DOCUNI
      CHARACTER*80 BODY,GRID,INPUT
C
C             OPEN VARIOUS FILES FOR INPUT,OUTPUT,ETC
C
C             INITIALIZE
C
      CALL INITIA
C
C             NOTE: SINCE THIS IS A MARCHING SCHEME, POINTS
C             NEED ONLY BE STORED FROM THE PREVIOUS K LEVEL.
C             THIS WOULD MEAN THAT THEY WOULD BE WRITTEN OUT
C             AFTER EVERY CALL TO STEP.....
C
C             BUT TO MAKE IT EASY TO USE THE BUNING GRAPHICS
C             PACKAGE, I WILL STORED THEM ALL AND WRITE
C             THEM OUT AT THE END
C
C             START BY LOADING BODY SURFACE POINTS INTO
C             XX,YY WHICH WILL STORE THE GRID AS WE MARCH OUT
C
      DO 5 J=1,JMAX
      XX(J,1) = X(J)
      YY(J,1) = Y(J)
      WRITE(GRIDUN,700)XX(J,1),YY(J,1)
    5 CONTINUE
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C%%%%%%%%%%%%% MAIN LOOP TO MARCH GRID TO OUTER BOUNDARY %%%%%%%%%%%%%
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      DO 10 K=2,KMAX
C
C             ADVANCE GRID FROM K ----> K+1
C
      CALL STEP(K)
C
C             STORE SURFACE GRID PTS (MAY WISH TO WRITE OUT POINTS HERE)
C
      DO 7 J=1,JMAX
      XX(J,K) = X(J)
      YY(J,K) = Y(J)
C  THIS WRITE PROVIDES DATA IN THE FORM THE NAVIER-STOKES SOLVER NEEDS.
      WRITE(GRIDUN,700) XX(J,K),YY(J,K)
    7 CONTINUE
  700 FORMAT(2E15.9)
C
      WRITE(6,*)'.....FINISHED LEVEL',K,'  ALPHA=',ALPHA
C
   10 CONTINUE
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C%%%%%%%%%%%% END OF MAIN LOOP TO MARCH GRID TO OUTER BOUNDARY %%%%%%%%
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
C             O.K. WE'RE DONE...LET'S WRITE OUT THAT GRID
C
C     WRITE(GRIDUN)JMAX,KMAX
C     WRITE(GRIDUN,250)((XX(J,K),J=1,JMAX),K=1,KMAX)
```

```
C     *          ,(YY(J,K ,J=1,JMAX),K=1,KMAX)
  250 FORMAT(2X,E20.10,2X,E20.10)
C
C                MASTA LA VISTA
C
      STOP
 1111 FORMAT(A)
      END
C=======================================================================
C
      SUBROUTINE SARC
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301,),Y(301,),XXI(301,),XETA(301,),YXI(301,),
     1 YETA(301,),R(301,100),SCALE(100,),SR(301,2),VOL(301,)
C
C          INTEGRATE ARC LENGTH AROUND BODY
C
      S = 0.
      DO 10 J=1,JMAX-1
      JP = J+1
      S=S+SQRT((X(JP)-X(J))**2 + (Y(JP)-Y(J))**2)
   10 CONTINUE
      SBODY= S
      RETURN
      END
C=======================================================================
C
      SUBROUTINE INITIA
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301,),Y(301,),XXI(301,),XETA(301,),YXI(301,),
     1 YETA(301,),R(301,100),SCALE(100,),SR(301,2),VOL(301,)
      COMMON /ORDER/ALPHA,ALPHAM,KM1,KM2
      COMMON /DISS/SMUIM,SMUIMV
      COMMON /SCALFT/ DSETA
      COMMON /SOBO/ SETAMX
      COMMON /UNITS / BODYUN,GRIDUN,DOCUNI
      COMMON /FNAMES/ BODYFI,GRIDFI,DOCFIL
      LOGICAL VERBOS
      CHARACTER*1 ANSWER
      INTEGER BODYUN,GRIDUN,DOCUNI
C
C
      VERBOS = .TRUE.
      PI = 4.*ATAN(1.)
C
C     JMAX     :       NUMBER OF POINTS IN THE STREAMWISE DIRECTION
C
C     KMAX     :       NUMBER OF POINTS IN THE NORMAL DIRECTION
C                      (NOTE: K=1 IS THE BODY SURFACE
C
C     DSETA    :       APPROXIMATE CELL HEIGHT FOR K=1 TO K=2
C
C     SETAMX   :       APPROXIMATE DISTANCE TO OUTER BOUNDARY
C                      (NOTE: THIS IS "EXACT" IF THE BODY IS A CIRCLE.
C                      I'VE NEVER ACTUALLY CHECKED IF THIS IS TRUE
C
C     SCKMAX :         VALUE OF SCALING FUNCTION AT KMAX "SCALE(KMAX)"
C                      SCALE IS THE FUNCTION THAT TRANSITIONS FROM
C                      STREAMWISE CLUSTERED VOLUMES TO EQUAL-
C                      SPACED VOLUMES AS MENTIONED IN THE
C                      ORIGINAL PAPER BY STEGER AND CHAUSSEE
C
C     SMU      :       COEFFICIENT FOR EXPLICIT DISSIPATION
C
C     SMUIM    :       COEFFICIENT FOR IMPLICIT DISSIPATION
C
C     ALPHAM   :       MAXIMUM ALPHA FOR MARCHING INTEGRATION
C
C                      ================================================
C                      *                                              *
C                      *  ALPHA = 0           EULER EXPLICIT          *
C                      *                                              *
C                      *  ALPHA = 1/2         TRAP RULE               *
C                      *                                              *
C                      *  ALPHA = 1           EULER IMPLICIT          *
C                      *                                              *
C                      ================================================
C
C     KM1      :       INNER VALUE OF K FOR VARIABLE ALPHA INTEGRATION
C                        EXPLAINED BELOW
C
C     KM2      :       OUTER VALUE OF K FOR VARIABLE ALPHA INTEGRATION
C                        (EXPLAINED BELOW
C
C
C     THE ALPHA INTEGRATION SCHEME STARTS OUT AT THE BODY WITH
C     ALPHA=0 (EULER IMPLICIT) AND LINEARLY INCREASES TO ALPHAM AT
C     K=KM1.  FROM KM1 TO KM2 ALPHA REMAINS CONSTANT AT ALHAM.
C     FROM KM2 TO KMAX,  ALPHA LINEARLY DECREASES BACK TO 0 AT KMAX
C
C     THIS VARIABLE ALPHA IS INCORPORATED TO ALLOW FOR ALPHA INTEGRATION
C     GREATER THAN ONE.... FOR ALPHA GREATER THAT ONE THE SCHEME
C     BECOMES VERY TEMPORIALLY-SPACIALLY DISSIPATIVE
C     (IF ETA IS VIEWED AS A TIME-LIKE  DIRECTION,
C     AND AS A RESULT GIVES VERY SMOOTH
C     GRIDS WITH A SMALL DEGRADATION IN ORTHOGONALITY, INVERSE JACOBIAN ...ETC
C
C     SMU,SMUIM ARE SCALED LINEARLY FROM ZERO AT K=1 TO SMU,SMUIM AT
C     KM1 AND REMAIN CONSTANT FORM KM1 TO KMAX
C
C
C               READ INPUTS
C
      WRITE(6,*)' INPUT KMAX,NORMAL WALL SPACING'
      READ(5,*)KMAX,DSETA
      WRITE('DOCUN1',*) ' KMAX,NORMAL WALL SPACING',KMAX,DSETA
      SCKMAX = .98
      SMU    = .005
      SMUIM  = .005
      ALPHAM = 2.5
      KM1 = .25*KMAX
      KM2 = .75*KMAX
C
```

```
C               ECHO INPUTS
C
  2    CONTINUE
       WRITE(6,1000)JMAX,KMAX,DSETA,SETAMX,SCKMAX,
      $             SMU,SMUIM,ALPHAM,KM1,KM2
 1000 FORMAT(' JMAX=',I4,'  KMAX=',I4,'  DSETA=',E12.6,
      $       ' SETAMX=',F12.6,' SCKMAX=',F12.6,/,
      &       ' SMU=',F14.7,' SMUIM=',F14.7,' ALPHAM=',F14.7,
      $       ' AT KM1=',I5,' KM2=',I5)
       WRITE(6,*)' DO YOU WISH TO CHANGE THE ABOVE DEFAULTS? <Y/N>'
       READ(5,2222)ANSWER
 2222  FORMAT(A)
       IF(ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'Y')THEN
        WRITE(6,*)' CHANGE THE DEFAULT VALUES IN SUBROUTINE INITIA.'
       STOP
       ENDIF
C
CCC                READ IN BODY COORDINATES
CCC
CCC                FIND EPS FOR OUTER BOUNDARY APPROX
CCC                USES AN EXPONENTIAL STRETCING IN THE NORMAL DIRECTION
CCC
CCC                EPSIL FIND THE COEFFICIENT FOR THE STRETCING GIVEN
CCC                THE WALL SPACING  AND THE OUTER BOUNDARY DISTANCE (APPROX)
C
       EPS=EPSIL(SETAMX,0.0,DSETA,KMAX,0.004,40,1)
C
CCC                DISTRIBUTE RADIAL POINTS
C
       VERBOS = .FALSE.
       IF(VERBOS)WRITE(6,*)'  INITIAL RADIAL SPACING'
       DO 15 J=1,JMAX
       R(J,1) = 0.00
   15  CONTINUE
       DO 16 K=2,KMAX
       DO 16 J=1,JMAX
       R(J,K)=R(J,K-1)+DSETA*(1.+EPS)**(K-2)
       IF(VERBOS .AND. J.EQ.1)WRITE(6,*)K,R(1,K)
   16  CONTINUE
C
CC           SET SCALING FUNCTION   SCKMAX = SCALE(KMAX)
C
       SCALE(1)=1.
       ESCAL = 1. - EXP(ALOG(SCKMAX)/FLOAT(KMAX-2))
       DO 17 K=2,KMAX
       SCALE(K) = ( 1. - ESCAL)**(K-2)
   17  CONTINUE
C
       RETURN
       END
C-----------------------------------------------------------------------
C
       SUBROUTINE METRIC
C
C-----------------------------------------------------------------------
       COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
       COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
      1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
       DO 10 J=2,JMAX-1
       JP=J+1
       JR=J-1
       XXI(J)=(X(JP)-X(JR))*0.5
       YXI(J)=(Y(JP)-Y(JR))*0.5
       XXYY=XXI(J)**2+YXI(J)**2
C
CCC          NOTE: ETA METRICS ARE OBTAINED FROM DIFFERENTIAL
CCC                EQUATIONS AND ARE NONLINEAR AS SUCH
CCC
CCC          FROM THE CALLING SEQUENCE USED...
CCC          VOLUMES FROM THE UNKNOWN K LEVEL ARE BEING
CCC          USED (THIS IS EXPERIMENTALLY WORKS WELL)
CCC          .... THIS WHEN USED WITH ALPHA > 1
CCC         SEEMS TO HELP OUT IN REGIONS OF CONCAVE CURVATURE
C
       XETA(J)=-YXI(J)*VOL(J)/XXYY
       YETA(J)= XXI(J)*VOL(J)/XXYY
  10   CONTINUE
       RETURN
       END
C-----------------------------------------------------------------------
C
       SUBROUTINE STEP(K)
C
C-----------------------------------------------------------------------
       COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
       COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
      1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
       COMMON/ORDER/ALPHA,ALPHAM,KM1,KM2
       COMMON/DISS/SMUIM,SMUIMV
       DIMENSION B(301,2,2),C(301,2,2),D(301,2,2),F(301,2)
       DIMENSION A(301,2,2),E(301,2,2)
C
CCC             SET UP ALPHA VARIATION
C
       IF(K .LE.KM1)THEN
         SC1 =  FLOAT(K-2)/FLOAT(KM1-2)
         SC2=SC1
       ELSEIF(K .GE.KM2)THEN
         SC1 =  1. - FLOAT(K-KM2)/FLOAT(KMAX-KM2)
         SC2=  1.
       ELSE
         SC1 =  1.
         SC2=SC1
       ENDIF
C
C
C
       ALPHA = 1. + (ALPHAM - 1.)*SC1
C
CCC       SCALE DISSIPATION
C
       SMUV = SC2*SMU
       SMUIMV = SC2*SMUIM
C
C
       CALL VOLUME(K)
       CALL METRIC
       CALL RMS(K)
C
C          INVERSION IN ETA DIRECTION
```

```
C           FILL A,B,C,F ARRAYS FOR BLOCK TRIDIAGONAL
C
      CALL FILTRY(K,A,B,C,D,E,F)
C
      CALL BPENTA(1,JMAX,A,B,C,D,E,F)
C
C           CALCULATE NEW X,Y COORDINATES
C
      DO 5 J=1,JMAX
      X(J) = X(J) +   F(J,1)
      Y(J) = Y(J) +   F(J,2)
 5    CONTINUE
      RETURN
      END
C=======================================================================
C
      SUBROUTINE FILTRY(K,A,B,C,D,E,F)
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
      DIMENSION B(301,2,2),C(301,2,2),D(301,2,2),F(301,2)
      DIMENSION A(301,2,2),E(301,2,2)
      COMMON /DISS/SMUIM,SMUIMV
      COMMON /ORDER/ ALPHA,ALPHAM,KM1,KM2
      DIMENSION AM(2,2)
C
C           ================================================
C           =      FILL TRIDIAGONAL MATRICES IN ETA-DIRECTION  =
C           ================================================
C
      DO 5 N=1,2
      DO 5 M=1,2
      DO 5 J=1,JMAX
      A(J,N,M) = 0.
      E(J,N,M) = 0.
 5    CONTINUE
      DO 10 J=2,JMAX-1
C
C           METRIC TERMS IN B.-1A
C
      R1=XETA(J)
      R2=YETA(J)
      R3=XXI(J)
      R4=YXI(J)
      RMET=1./(R3**2+R4**2)
      R1=R1*RMET
      R2=R2*RMET
C
      CALL GMATRX(AM,J,K,R1,R2,R3,R4)
C
C           LOAD B -1A INTO BANDS (ALLOW FOR GENERAL INTEGRATION..ALPHA)
C
      C(J,1,1) = 1.
      C(J,2,2) = 1.
      C(J,1,2) = 0.
      C(J,2,1) = 0.
      DO 8 M=1,2
      DO 8 N=1,2
      B(J,N,M) = -.5*ALPHA*AM(N,M)
      D(J,N,M) =  .5*ALPHA*AM(N,M)
 8    CONTINUE
      DO 10 N=1,2
      F(J,N)=SR(J,N)
10    CONTINUE
C
C             FULL IMPLICIT 4TH ORDER DISSIPATION
C
      J=2
      DO 12 N=1,2
      A(J,N,N)=A(J,N,N)+0.*SMUIMV
      B(J,N,N)=B(J,N,N)-2.*SMUIMV
      C(J,N,N)=C(J,N,N)+5.*SMUIMV
      D(J,N,N)=D(J,N,N)-4.*SMUIMV
      E(J,N,N)=E(J,N,N)+1.*SMUIMV
12    CONTINUE
      J=JMAX-1
      DO 14 N=1,2
      A(J,N,N)=A(J,N,N)+1.*SMUIMV
      B(J,N,N)=B(J,N,N)-4.*SMUIMV
      C(J,N,N)=C(J,N,N)+5.*SMUIMV
      D(J,N,N)=D(J,N,N)-2.*SMUIMV
      E(J,N,N)=E(J,N,N)+0.*SMUIMV
14    CONTINUE
      DO 15 N=1,2
      DO 15 J=3,JMAX-2
      A(J,N,N)=A(J,N,N)+1.*SMUIMV
      B(J,N,N)=B(J,N,N)-4.*SMUIMV
      C(J,N,N)=C(J,N,N)+6.*SMUIMV
      D(J,N,N)=D(J,N,N)-4.*SMUIMV
      E(J,N,N)=E(J,N,N)+1.*SMUIMV
15    CONTINUE
C
C             IMPLICIT C-GRID B-C AT K=1,KMAX
C
      DO 20 N=1,2
      DO 20 M=1,2
      J=1
      A(J,N,M)=0.
      B(J,N,M)=0.
      C(J,N,M)=0.
      D(J,N,M)=0.
      E(J,N,M)=0.
      F(J,N)=0.
      J=JMAX
      A(J,N,M)=0.
      B(J,N,M)=0.
      C(J,N,M)=0.
      D(J,N,M)=0.
      E(J,N,M)=0.
      F(J,N)=0.
20    CONTINUE
      J=1
      C(J,1,1)= 1.
      C(J,2,2)= 1.
      D(J,2,2)= -2
      E(J,2,2)=  1
      J=JMAX
      C(J,1,1)= 1.
      C(J,2,2)= 1.
      B(J,2,2) = -2
```

```
      A(J,2,2)=   1.
      RETURN
      END
C=======================================================================
      SUBROUTINE VOLUME(K)
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
      COMMON /ORDER/ALPHA,ALPHAM,KM1,KM2
      CALL SARC
      DO 5 J=2,JMAX-1
      JP = J+1
      JR = J-1
      DS = .5*( R(J-1,K)-R(J-1,K-1) + R(J+1,K)-R(J+1,K-1) )
      DL = SQRT( (.5*(X(JP)-X(JR)))**2 + (.5*(Y(JP)-Y(JR)))**2)
      VOL(J)=DS*(SCALE(K)*DL +(1.-SCALE(K))*SBODY/FLOAT(JMAX-1))
5     CONTINUE
      RETURN
      END
C=======================================================================
      SUBROUTINE RHS(K)
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
      COMMON /ORDER/ALPHA,ALPHAM,KM1,KM2
C
C      CALCULATE FORCING FUNCTION; CONSISTS OF NEW AND OLD
C      VOLUMES PLUS VARIABLES EVALUATED AT THE KTH LEVEL.
C
      DO 5 J=2,JMAX-1
      JP = J+1
      JR = J-1
C
      R1=XXI(J)
      R2=YXI(J)
      R3=XETA(J)
      R4=YETA(J)
      RMET=1./(R1**2+R2**2)
      SR(J,1)=-R2*(ALPHA*VOL(J)+(1.-ALPHA)*(R1*R4-R2*R3))*RMET
      SR(J,2)= R1*(ALPHA*VOL(J)+(1.-ALPHA)*(R1*R4-R2*R3))*RMET
C
C             NUMERICAL DISSIPATION
C
      IF(J .EQ.2)THEN
        JPP=J+2
        SMX=-SMUV*(-2.*X(JR)+5.*X(J)-4.*X(JP)+X(JPP))
        SMY=-SMUV*(-2.*Y(JR)+5.*Y(J)-4.*Y(JP)+Y(JPP))
      ELSEIF(J .EQ. JMAX-1)THEN
        JRR=J-2
        SMX=-SMUV*(-2.*X(JP)+5.*X(J)-4.*X(JR)+X(JRR))
        SMY=-SMUV*(-2.*Y(JP)+5.*Y(J)-4.*Y(JR)+Y(JRR))
      ELSE
        JPP=J+2
        JRR=J-2
        SMX=-SMUV*(X(JRR)+6.*X(J)+X(JPP)-4.*(X(JP)+X(JR)))
        SMY=-SMUV*(Y(JRR)+6.*Y(J)+Y(JPP)-4.*(Y(JP)+Y(JR)))
      ENDIF
      SR(J,1)=SR(J,1)+SMX
      SR(J,2)=SR(J,2)+SMY
C
C
5     CONTINUE
      RETURN
      END
C=======================================================================
C
C
      SUBROUTINE GMATRX(A,J,K,R1,R2,R3,R4)
C
C=======================================================================
      COMMON/BASE/JMAX,KMAX,PI,SBODY,SMU,SMUV,NBODY,NCLUST
      COMMON/VARS/X(301),Y(301),XXI(301),XETA(301),YXI(301),
     1 YETA(301),R(301,100),SCALE(100),SR(301,2),VOL(301)
      DIMENSION A(2,2)
C
C             FILL BINV=A
C
      A(1,1)=R1*R3-R2*R4
      A(1,2)=R2*R3+R4*R1
      A(2,1)=R4*R1+R3*R2
      A(2,2)=R4*R2-R3*R1
      RETURN
      END
C=======================================================================
C
      FUNCTION EPSIL(FMX,FMIN,DFM,NPT,FPCC,ICC,    NCALL)
C
C=======================================================================
C
C        THIS SUBROUTINE APPLIES A NEWTON-RAPHSON ROOT-FINDING
C        TECHNIQUE TO FIND A VALUE OF EPSILON FOR A PARTICULAR USE
C        OF THE EXPONENTIAL STRECHING TRANSFORMATION.
C
C     FMX IS TOTAL ARC LENGTH ALONG COORDINATE
C     FMIN IS STARTING VALUE OF ARC LENGTH   SUCH AS 0.0
C     DFM IS SPECIFIED INITIAL INCREMENT OF ARC LENGTH
C     NPT IS NUMBER OF POINTS ALONG COORDINATE
C     FPCC IS ITERATIVE ERROR BOUND, E.G.O( 0.00002)
C     ICC IS MAXIMUM NUMBER OF ITERATIONS
C     NCALL     IF NCALL=1 INITIAL GUESS FOR EPS IS USED
C               IF NCALL .GT. 1, PREVIOUS EPS USED AS INITIAL GUESS
C
C
      FMXL=FMX
      FMINL=FMIN
      DFML=DFM
      FPCCL=FPCC
      ICCL=ICC
C
C
      FNPTM2=FLOAT(NPT-2)
      IF(NCALL.EQ.1)  EPS=(FMXL/DFML)**(1.0/FNPTM2)-1.0
C
      DO 3 NIT=1,ICCL
      EP1=EPS+1.0
      EP1TN=EP1**FNPTM2
      REPS=1.0/EPS
```

```
      DFMOE=DFML*REPS
      F=FMXL-FMINL-DFMOE*(EP1TM=EP1-1.0
      IF(ABS F..LT.FPCCL    GO TO 4
      DFMOE2=DFMOE*REPS
      FPN=DFMOE2='1.0+EP1TM*(EPS*FNPTM2-1.0
      EPS=EPS+F/FPN
3     CONTINUE
C
5     EPSIL=EPS
      WRITE(6,100)
      RETURN
C
4     EPSIL=EPS
C      WRITE(6,101   EPSIL,F,NIT
      RETURN
C
100   FORMAT(/42H EXCEEDED MAX. NO. OF ITERATIONS IN EPSIL.)
101   FORMAT(/7H EPSIL=,F12.5,5X,7H AND F=,F12.5,5X,7H AFTER ,I3,
     *  12H ITERATIONS.)
C
      END
C=====================================================================
C
      SUBROUTINE BPENTA IL,IU,A,B,C,D,E,F
C
C=====================================================================
      DIMENSION A(301,2,2),B(301,2,2),C(301,2,2),
     +                  D(301,2,2),E(301,2,2),F(301,2,
      DIMENSION G(2,2),H(2,2),U(301,2,2),V(301,2,2),
      COMMON/LUD/ L11,L21,L22,U12,V1,V2
      REAL L11,L21,L22
C
C
C           2X2 BLOCK PENTADIAGONAL SOLVER
C           WRITTEN BY TIM BARTH, NASA-AMES
C
C
      IS = IL + 2
      I  = IL
C
      DO 10 N=1,2
      DO 10 M=1,2
      G(N,M) = C(I,N,M)
10    CONTINUE
      CALL LUDEC(G)
      D1 = V1*F(I,1)
      D2 = V2*(F(I,2  - L21*D1)
      F(I,2  = D2
      F(I,1) = D1 - U12*F(I,2
      DO 11 M=1,2
      D1 = V1*E(I,1,M)
      D2 = V2*(E(I,2,M) - L21*D1)
      V(I,2,M) = D2
      V(I,1,M) = D1 - U12*V(I,2,M)
      D1 = V1*D(I,1,M
      D2 = V2*(D(I,2,M  - L21*D1)
      U(I,2,M) = D2
      U(I,1,M) = D1 - U12*U(I,2,M
11    CONTINUE
C
C
      I = IL + 1
      L = I  - 1
C
C
      DO 20 N=1,2
      DO 20 M=1,2
      G(N,M) = C(I,N,M) - B(I,N,1)*U(L,1,M) - B(I,N,2)*U(L,2,M)
20    CONTINUE
      CALL LUDEC(G)
      DO 21 M=1,2
      F(I,M) = F(I,M) - B(I,M,1)*F(L,1  - B(I,M,2)*F(L,2)
21    CONTINUE
      D1 = V1*F(I,1)
      D2 = V2*(F(I,2  - L21*D1)
      F(I,2) = D2
      F(I,1  = D1 - U12*F(I,2
      DO 22 N=1,2
      DO 22 M=1,2
      H(N,M) = D(I,N,M  - B(I,N,1)*V(L,1,M) - B(I,N,2)*V(L,2,M)
22    CONTINUE
      DO 25 M=1,2
      D1 = V1*E(I,1,M
      D2 = V2*(E(I,2,M  - L21*D1
      V(I,2,M  = D2
      V(I,1,M) = D1 - U12*V(I,2,M
      D1 = V1*H(1,M
      D2 = V2*(H(2,M  - L21*D1
      U(I,2,M  = D2
      U(I,1,M) = D1 - U12*U(I,2,M)
25    CONTINUE
C
C
      DO 50 I=IS,IU
      L = I - 1
      LL= I - 2
      DO 30 N=1,2
      DO 30 M=1,2
      B(I,N,M  = B(I,N,M  - A(I,N,1)*U(LL,1,M  - A(I,N,2)*U(LL,2,M
30    CONTINUE
      DO 35 N=1,2
      DO 35 M=1,2
      G(N,M) = C(I,N,M) - A(I,N,1)*V(LL,1,M) - A(I,N,2)*V(LL,2,M)
     *                  - B(I,N,1)*U(L ,1,M) - B(I,N,2)*U(L ,2,M)
35    CONTINUE
      CALL LUDEC(G)
      DO 37 M=1,2
      F(I,M  = F(I,M  - B(I,M,1)*F(L,1  - B(I,M,2)*F(L,2
     *                - A(I,M,1)*F(LL,1 - A(I,M,2)*F(LL,2
37    CONTINUE
      D1 = V1*F(I,1)
      D2 = V2*(F(I,2  - L21*D1)
      F(I,2) = D2
      F(I,1) = D1 - U12*F(I,2)
      IF(I .EQ. IU)GO TO 80
      DO 42 N=1,2
      DO 42 M=1,2
      H(N,M  = D(I,N,M  - B(I,N,1)*V(L,1,M) - B(I,N,2)*V(L,2,M
42    CONTINUE
      DO 45 M=1,2
      D1 = V1*H(1,M
      D2 = V2*(H(2,M  - L21*D1)
```

```
      U(I,2,M) = D2
      U(I,1,M) = D1 - U12*U(I,2,M)
45    CONTINUE
      IF(I.EQ.IU-1)GO TO 80
      DO 46 M=1,2
      D1 = V1*E(I,1,M)
      D2 = V2*E(I,2,M) - L21*D1
      V(I,2,M) = D2
      V(I,1,M) = D1 - U12*V(I,2,M)
46    CONTINUE
80    CONTINUE
50    CONTINUE
C
C     BACK SWEEP
C
      I = IU - 1
      DO 60 M=1,2
      F(I,M) = F(I,M) - U(I,M,1)*F(I+1,1) - U(I,M,2)*F(I+1,2)
60    CONTINUE
      DO 65 I = IU-2,IL,-1
      DO 65 M=1,2
      F(I,M) = F(I,M) - U(I,M,1)*F(I+1,1) - U(I,M,2)*F(I+1,2)
     *                - V(I,M,1)*F(I+2,1) - V(I,M,2)*F(I+2,2)
65    CONTINUE
      RETURN
      END
      SUBROUTINE LUDEC(A)
      DIMENSION A(2,2)
      COMMON/LUD/ L11,L21,L22,U12,V1,V2
      REAL L11,L21,L22
C  SUBROUTINE COMPUTES L-U DECOMPOSITION ELEMENTS
      L11 = A(1,1)
      V1 = 1./L11
      U12 =  V1*A(1,2)
      L21 = A(2,1)
      L22 = A(2,2)  - L21*U12
      V2 = 1./L22
      RETURN
      END
```

```
      CWAKE=0.058
      CKLEBH=0.53
      CSCALE=7.5
      CWK = .25
      EDYMAX = 5000.
C
      RE2=SQRT(RE)
      ITRSU1=ITRSU+1
      F(1)=0.0
      D(1)=0.0
      XMUTI(1)=0.0
      W(1)=999.
      XMUTO(1)=999.
      IL1=IL-1
      ITEUP1=ITEU+1
      IWKST1=IWKST-1
      IWKSTL=IL-IWKST+1
      XRLXW=.5*XRELAX
      IWKEP1=IWKE+1
      IWKEL=IL-IWKE+1
C
C ------------------------------------------------------------
C
C     AIRFOIL UPPER SURFACE
C ------------------------------------------------------------
      DO 1 I=ITRSU1,ITEU
C
      UETA=U(I,2)-U(I,1)
      VETA=V(I,2)-V(I,1)
      XE=X(I,2)-X(I,1)
      YE=Y(I,2)-Y(I,1)
      TAUW=ABS(XMU(I,1)*((UETA*XXI(I,1)+VETA*YXI(I,1))/
     -     (XXI(I,1)*YE-XE*YXI(I,1)))
C EVALUATE VANDRIEST DAMPING FACTOR
      DO 2 J=2,JLAST
      YPLUS=RE2*SQRT(RHO(I,1)*TAUW)*S(I,J)/XMU(I,1)
      D(J)=1.-EXP(-YPLUS/APLUS)
      IF (D(J).LT.0.9999) GO TO 2
      JD=J+1
      GO TO 100
    2 CONTINUE
      GO TO 200
  100 DO 3 J=JD,JLAST
    3 D(J)=1.0
C EVALUATE FOUTER AND XMUTI
  200 DO 4 J=2,JLAST
      UXI=.5*(U(I+1,J)-U(I-1,J))
      VXI=.5*(V(I+1,J)-V(I-1,J))
      UETA=.5*(U(I,J+1)-U(I,J-1))
      VETA=.5*(V(I,J+1)-V(I,J-1))
      W(J)=-(XXI(I,J)*UETA-XETA(I,J)*UXI-YETA(I,J)*VXI
     -     +YXI(I,J)*VETA)/XJAC(I,J)
      F(J)=S(I,J)*ABS(W(J))*D(J)
      XMUTI(J)=RE*RHO(I,J)*ABS(W(J))*((VK*S(I,J)*D(J))**2)
     -         *GAMMA(I)
    4 CONTINUE
C DETERMINE FMAX,YMAX
      FMAX=0.0
      DO 5 J=2,JLAST
      IF(F(J).LE.FMAX) GO TO 5
      FMAX=F(J)
      JMAX=J
-----------------------------------------------------------------
    5 CONTINUE
      YMAX = S(I,JMAX)
      IF ( JMAX .EQ. JLAST ) GOTO 6
C USE QUADRATIC INTERPOLATION TO COMPUTE FMAX,YMAX
      JMXM=JMAX-1
      JMXP=JMAX+1
      F1=(F(JMXM)-FMAX)/(S(I,JMXM)-S(I,JMAX))
      F2=(FMAX-F(JMXP))/(S(I,JMAX)-S(I,JMXP))
      F3=(F1-F2)/(S(I,JMXM)-S(I,JMXP))
      YMAX=.5*(S(I,JMAX)+S(I,JMXM)-F1/F3)
      FMAX=F(JMXM)+F1*(YMAX-S(I,JMXM))+F3*(YMAX-S(I,JMXM))
     -     *(YMAX-S(I,JMAX))
C COMPUTE 'FWAKE'
  6   FWAKE=YMAX*FMAX
C
C ADDED
C COMPUTE UDIF
      UDIF = 0.
      IFLAG = 0
      DO 138 J=2, JLAST
        DX1 = ABS(U(I,J)) + ABS(V(I,J))
        UDIF = AMAX1(UDIF,DX1)
  138 CONTINUE
      FWAKE1 = CWK*YMAX*(UDIF*UDIF)/FMAX
      IF ( FWAKE1 .LT. FWAKE ) IFLAG =1
      FWAKE = AMIN1(FWAKE,FWAKE1)

C COMPUTE XMUTO
  400 DO 7 J=2,JLAST
        FKLEB=1./(1.+5.5*(CKLEB*S(I,J)/YMAX)**6)
        XMUTO(J)=RE*CCP*CLAU*RHO(I,J)*FWAKE*FKLEB*GAMMA(I)
        XMUTO(J) = AMIN1(XMUTO(J),EDYMAX)
    7 CONTINUE
C COMPUTE EDDY(I,J)
      DO 8 J=2,JLAST
      IF(XMUTI(J).LT.XMUTO(J)) GO TO 8
      JTRAN=J
      GO TO 500
    8 CONTINUE
  500 JTRM1=JTRAN-1
      DO 9 J=2,JTRM1
    9 EDDY(I,J)=XMUTI(J)
      DO 10 J=JTRAN,JLAST
   10 EDDY(I,J)=XMUTO(J)
C
      IF (IPLOT.EQ.0) GO TO 1
C OUTPUT
      Y2PLUS=RE2*SQRT(RHO(I,1)*TAUW)*S(I,2)/XMU(I,1
      WRITE (6,202) I,YMAX,FMAX,FWAKE,JMAX,JTRAN,Y2PLUS
      IF(IPLOT1.EQ.0) GO TO 1
      WRITE (6,203)
      DO 11 J=2,JLAST
   11 WRITE(6,204) J,S(I,J),F(J),W(J),D(J),XMUTI(J),XMUTO(J),IFLAG
C
    1 CONTINUE
C
C ------------------------------------------------------------
C
C     AIRFOIL LOWER SURFACE
C ------------------------------------------------------------
      KLAST=ITRSL-ITEL+1
      DO 12 K=2,KLAST
```

```
      I=ITRSL-K+1
C
      UETA=U(I,2)-U(I,1)
      VETA=V(I,2)-V(I,1)
      XE=X(I,2)-X(I,1)
      YE=Y(I,2)-Y(I,1)
      TAUM=ABS(XMU(I,1)*(UETA*XXI(I,1)+VETA*YXI(I,1))/
     -        (XXI(I,1)*YE-XE*YXI(I,1)))
C EVALUATE VANDRIEST DAMPING FACTOR
      DO 13 J=2,JLAST
      YPLUS=RE2*SQRT(RHO(I,1)*TAUM)*S(I,J)/XMU(I,1)
      D(J)=1.-EXP(-YPLUS/APLUS)
      IF (D(J).LT.0.9999) GO TO 13
      JD=J+1
      GO TO 600
   13 CONTINUE
      GO TO 700
  600 DO 14 J=JD,JLAST
   14 D(J)=1.0
C EVALUATE FOUTER AND XMUTI
  700 DO 15 J=2,JLAST
      UXI=.5*(U(I+1,J)-U(I-1,J))
      VXI=.5*(V(I+1,J)-V(I-1,J))
      UETA=.5*(U(I,J+1)-U(I,J-1))
      VETA=.5*(V(I,J+1)-V(I,J-1))
      W(J)=-(XXI(I,J)*UETA-XETA(I,J)*UXI-YETA(I,J)*VXI
     -      +YXI(I,J)*VETA)/XJAC(I,J)
      F(J)=S(I,J)*ABS(W(J))*D(J)
      XMUTI(J)=RE*RHO(I,J)*ABS(W(J))*((VK*S(I,J)*D(J))**2)
     -        *GAMMA(I)
   15 CONTINUE
C DETERMINE FMAX,YMAX
      FMAX=0.0
C ---------------------------------------
      DO 16 J=2,JLAST
      IF(F(J).LE.FMAX) GO TO 16
      FMAX=F(J)
      JMAX=J
   16 CONTINUE
      YMAX=S(I,JMAX)
      IF ( JMAX .EQ. JLAST) GOTO 17
C USE QUADRATIC INTERPOLATION TO COMPUTE FMAX,YMAX
      JMXM=JMAX-1
      JMXP=JMAX+1
      F1=(F(JMXM)-FMAX)/(S(I,JMXM)-S(I,JMAX))
      F2=(FMAX-F(JMXP))/(S(I,JMAX)-S(I,JMXP))
      F3=(F1-F2)/(S(I,JMXM)-S(I,JMXP))
      YMAX=.5*(S(I,JMAX)+S(I,JMXM)-F1/F3)
      FMAX=F(JMXM)+F1*(YMAX-S(I,JMXM))+F3*(YMAX-S(I,JMXM))
     -      *(YMAX-S(I,JMAX))
C COMPUTE 'FMAKE'
   17 FMAKE=YMAX*FMAX
C
C ADDED
C COMPUTE UDIF
      UDIF = 0.
      IFLAG = 0
      DO 139 J=2, JLAST
        DX1 = ABS(U(I,J)) + ABS(V(I,J))
        UDIF = AMAX1(UDIF,DX1)
  139 CONTINUE
-------------------------------------------
      FMAKE1 = CMK*YMAX*(UDIF*UDIF)/FMAX
      IF ( FMAKE1 .LT. FMAKE ) IFLAG =1
      FMAKE = AMIN1(FMAKE,FMAKE1)

C COMPUTE XMUTO
  900 DO 18 J=2,JLAST
        FKLEB=1./(1.+5.5*(CKLEB*S(I,J)/YMAX)**6)
        XMUTO(J)=RE*CCP*CLAU*RHO(I,J)*FMAKE*FKLEB*GAMMA(I)
        XMUTO(J) = AMIN1(XMUTO(J),EDYMAX)
   18 CONTINUE
C COMPUTE EDDY(I,J)
      DO 19 J=2,JLAST
      IF(XMUTI(J).LT.XMUTO(J)) GO TO 19
      JTRAN=J
      GO TO1000
   19 CONTINUE
 1000 JTRM1=JTRAN-1
      DO 20 J=2,JTRM1
   20 EDDY(I,J)=XMUTI(J)
      DO 21 J=JTRAN,JLAST
   21 EDDY(I,J)=XMUTO(J)
C
      IF (IPLOT.EQ.0) GO TO 12
C OUTPUT
      Y2PLUS=RE2*SQRT(RHO(I,1)*TAUM)*S(I,2)/XMU(I,1)
      WRITE (6,202)I,YMAX,FMAX,FMAKE,JMAX,JTRAN,Y2PLUS,IFLAG
      IF(IPLOT1.EQ.0) GO TO 12
      WRITE (6,203)
      DO 22 J=2,JLAST
   22 WRITE(6,204) J,S(I,J),F(J),W(J),D(J),XMUTI(J),XMUTO(J)
C
   12 CONTINUE
C
C ---------------------------------------------------------------
C                    MAKE
C ---------------------------------------------------------------
      DO 30 I=IMKST,IMKE
      ILM=IL-I+1
C DETERMINE UMAX,UMIN,MAKE CENTERLINE(YMKC)
      UMAX=0.0
      UMIN=9999.
      DO 31 J=1,JLAST
      VELU=SQRT(U(I,J)**2+V(I,J)**2)
      IF(VELU.GT.UMAX) UMAX=VELU
      VELL=SQRT(U(ILM,J)**2+V(ILM,J)**2)
      IF(VELL.GT.UMAX) UMAX=VELL
      IF(VELU.GE.UMIN) GO TO 32
      UMIN=VELU
      JMIN=J
      IMIN=I
   32 IF(VELL.GE.UMIN) GO TO 31
      UMIN=VELL
      JMIN=J
      IMIN=ILM
   31 CONTINUE
      UDIF=UMAX-UMIN
      IF(IMIN.GE.ITEU) SIGN=1.
      IF(IMIN.LT.ITEU) SIGN=-1.
      YMKC=S(IMIN,JMIN)*SIGN
C ---------COMPUTE FOUTER ------------------------
C  UPPER PART
```

```
      DO 33 J=2,JLAST
      JJ=JL+J
      UXI=.5*(U(I+1,J)-U(I-1,J))
      VXI=.5*(V(I+1,J)-V(I-1,J))
      UETA=.5*(U(I,J+1)-U(I,J-1))
      VETA=.5*(V(I,J+1)-V(I,J-1))
      W(JJ)=-(XXI(I,J)*UETA-XETA(I,J)*UXI-YETA(I,J)*VXI
     -     +YXI(I,J)*VETA)/XJAC(I,J)
      YMK=S(I,J)-YMKC
      F(JJ)=ABS(YMK*W(JJ))
   33 CONTINUE
C  LOWER PART
      DO 34 J=2,JLAST
      JJ=JL-J+1
      UXI=.5*(U(ILW+1,J)-U(ILW-1,J))
      VXI=.5*(V(ILW+1,J)-V(ILW-1,J))
      UETA=.5*(U(ILW,J+1)-U(ILW,J-1))
      VETA=.5*(V(ILW,J+1)-V(ILW,J-1))
      W(JJ)=-(XXI(ILW,J)*UETA-XETA(ILW,J)*UXI-YETA(ILW,J)*VXI
     -     +YXI(ILW,J)*VETA)/XJAC(ILW,J)
      YMK=S(ILW,J)+YMKC
      F(JJ)=ABS(YMK*W(JJ))
   34 CONTINUE
C MAKE-CUT POINTS
      UXI=.5*(U(I+1,1)-U(I-1,1))
      VXI=.5*(V(I+1,1)-V(I-1,1))
      UETA=.5*(U(I,2)-U(ILW,2))
      VETA=.5*(V(I,2)-V(ILW,2))
      XE=.5*(X(I,2)-X(ILW,2))
      YE=.5*(Y(I,2)-Y(ILW,2))
      XJ=XXI(I,1)*YE-XE*YXI(I,1)
      W(JL)=-(XXI(I,1)*UETA-XE*UXI-YE*VXI
     -     +YXI(I,1)*VETA)/XJ
      YMK=-YMKC
      F(JL)=ABS(YMK*W(JL))
      W(JL+1)=W(JL)
      F(JL+1)=F(JL)
C COMPUTE FMAX,YMAX
      FMAX=0.
      JJS=JL-JLAST+1
      JJE=JL+JLAST
      DO 341 JJ=JJS,JJE
      IF(F(JJ).LE.FMAX) GO TO 341
      FMAX=F(JJ)
      IF(JJ.GT.JL) GO TO 342
      JMAX=JL-JJ+1
      IMAX=ILW
      YMAX=S(IMAX,JMAX)+YMKC
      GO TO 341
  342 JMAX=JJ-JL
      IMAX=1
      YMAX=S(IMAX,JMAX)-YMKC
  341 CONTINUE
C
C COMPUTE FMAKE
      FMAKE=ABS(YMAX)*(UDIF**2)/FMAX
C COMPUTE EDDY(I,J)
   37 DO 38 J=1,JLAST
      YMK=ABS(S(I,J)-YMKC)
      FKLEB=1./(1.+5.5*(CKLEB*YMK/ABS(YMAX))**6)
   38 EDDY(I,J)=RE*CMAKE*RHO(I,J)*FMAKE*FKLEB
      DO 39 J=1,JLAST
      YMK=ABS(S(ILW,J)+YMKC)
      FKLEB=1./(1.+5.5*(CKLEB*YMK/ABS(YMAX))**6)
   39 EDDY(ILW,J)=RE*CMAKE*RHO(ILW,J)*FMAKE*FKLEB
C OUTPUT
      IF(IPLOT.EQ.0) GO TO 30
      WRITE(6,209)X(IMIN,JMIN),IMAX,JMAX,YMAX,FMAX,FMAKE,YMKC,IMIN,JMIN,
     -    UDIF
      IF(IPLOT1.EQ.0) GO TO 30
      KLAST=2*JLAST
      WRITE (6,210)
      DO 42 K=1,KLAST
      JJ=JL+JLAST-K+1
      J=JJ-JL
      IF(JJ.LE.JL) J=JL-JJ+1
      II=I
      IF(JJ.LE.JL) II=ILW
      WRITE(6,204) I,S(II,J),U(II,J),W(JJ),F(JJ),EDDY(II,J)
   42 CONTINUE
C
   30 CONTINUE
C
C ------------------------------------------------------------------
C
C        NEAR-WAKE
C ------------------------------------------------------------------
      DO 40 I=ITEUP1,IMKST1
      ILW=IL-I+1
      XL=CSCALE*(SS1(I)-SS1(ITEU))/XRLXH)/XRELAX
      FACTOR=TANH(XL)
      DO 40 J=1,JLAST
      EDDY(I,J)=.5*((EDDY(ITEU,J)+EDDY(IMKST,J))+FACTOR*
     -           (EDDY(IMKST,J)-EDDY(ITEU,J)))
      EDDY(ILW,J)=.5*((EDDY(ITEL,J)+EDDY(IMKSTL,J))+FACTOR*
     -           (EDDY(IMKSTL,J)-EDDY(ITEL,J)))
   40 CONTINUE
C
C
C ------------------------------------------------------------------
C
C  REGION OF CONSTANT EDDY VISCOSITY (VERY FAR FROM AIRFOIL)
C ------------------------------------------------------------------
      IF(IMKE.GE.IL1) GO TO 3000
      DO 41 I=IMKEP1,IL1
      ILW=IL-I+1
      DO 41 J=1,JLAST
      EDDY(I,J)=EDDY(IMKE,J)
      EDDY(ILW,J)=EDDY(IMKEL,J)
   41 CONTINUE
C ------------------------------------------------------------------
C******* FORMATS **********************
  202 FORMAT (1X,'I=',I4,2X,'YMAX=',F11.4,2X,'FMAX=',F11.4,2X,
     -'FMAKE=',F11.4,2X,'JMAX=',I3,2X,'JTRAN=',I3,2X,'Y2PLUS=',F11.4,
     -12)
  203 FORMAT(4X,'J',6X,'S(I,J)',6X,'F(J)',6X,'VORT.',6X,'DAMP.',
     -6X,'XMUTI',6X,'XMUTO')
  204 FORMAT(2X,I3,6(2X,E10.4))
  209 FORMAT(2X,'X/C=',F7.4,2X,'IMAX=',I4,2X,'JMAX=',I4,2X,'YMAX=',
     -F7.4,2X,'FMAX=',F7.4,2X,'FMAKE=',F7.4,2X,
     -'YMKC=',F7.4,2X,'IMIN=',I4,2X,'JMIN=',I4,2X,'UDIF=',F7.4
  210 FORMAT(4X,'J',6X,'S(I,J)',6X,'U(I,J)',6X,'VORT',6X,'F(J)',
     -    6X,'EDDY')
 3000 RETURN
```

```
      END

C
C ***************************************************************
C
C SUBROUTINE "LIFT" COMPUTES AIRFOIL LIFT AND DRAG COEFFICIENTS
C VISCOUS CASE
      SUBROUTINE LIFT (CL,CD)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/DX/ DX1,DX2,DX3,DX4,DX5,DX6,DX7,DX8,DX9,DX10
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLN,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      FX = 0.0
      FY = 0.0
C
      DO 1 I=ITEL,ITEUM1
      XXIK=X(I+1,1)-X(I,1)
      YXIK=Y(I+1,1)-Y(I,1)
      UETA=.5*(U(I+1,2)-U(I+1,1)+U(I,2)-U(I,1))
      VETA=.5*(V(I+1,2)-V(I+1,1)+V(I,2)-V(I,1))
      XETK=.5*(XETA(I+1,1)+XETA(I,1))
      YETK=.5*(YETA(I+1,1)+YETA(I,1))
      PK=.5*(P(I+1,1)+P(I,1))
      XMUK=.5*(XMU(I+1,1)+XMU(I,1))
      UXI=U(I+1,1)-U(I,1)
      VXI=V(I+1,1)-V(I,1)
      XJK=1./(XXIK*YETK-XETK*YXIK)
      UX=XJK*(UXI*YETK-UETA*YXIK)
      VX=XJK*(VXI*YETK-VETA*YXIK)
      UY=XJK*(-UXI*XETK+UETA*XXIK)
      VY=XJK*(-VXI*XETK+VETA*XXIK)
      TXY=REI*XMUK*(UY+VX)
      TXX=-PK+REI*XMUK*(XL*UX+XL*VY)
      TYY=-PK+REI*XMUK*(XL*VY+XL*UX)
      DX1=-YXIK*TXX+XXIK*TXY
      DX2=-YXIK*TXY+XXIK*TYY
      UA = .5*(U(I+1,1)+U(I,1))
      VA = .5*(V(I+1,1)+V(I,1))
      RA = .5*(RHO(I+1,1)+RHO(I,1))
      DX3=-RA*(-UA*UA*YXIK + UA*VA*XXIK)
      DX4=-RA*(-UA*VA*YXIK + VA*VA*XXIK)

C
      FX=FX+DX1+DX3
      FY=FY+DX2+DX4
    1 CONTINUE
C
      FLIFT=-FX*SINA+FY*COSA
      FDRAG=FX*COSA+FY*SINA
      CL=2.*FLIFT
      CD=2.*FDRAG
      RETURN
      END

C
C ***************************************************************
C
C SUBROUTINE 'OUTPUT' PRINTS OUT FLOW FIELD INFORMATION
      SUBROUTINE OUTPUT
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM)
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SS1(IM)
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XM1,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAM11,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLN,PI
      COMMON/DUMMY/ TEMP(IM,JM,4)
C
      XM12=XM1**2
C ------ UPPER SURFACE ----------
      WRITE(6,204)
      DO 1 I=ILE,ITEU
      WRITE (6,200) I
      WRITE (6,202)
      DO 1 J=1,JL
      T=GAM*XM12*P(I,J)/RHO(I,J)
      WRITE (6,203) J,X(I,J),Y(I,J),U(I,J),V(I,J),P(I,J),
     1RHO(I,J),T,EDDY(I,J)
    1 CONTINUE
C ------ LOWER SURFACE ----------
      WRITE (6,205)
      KLAST=ILE-ITEL
      DO 2 K=1,KLAST
      I=ILE-K
      WRITE(6,200) I
      WRITE(6,202)
      DO 2 J=1,JL
      T=GAM*XM12*P(I,J)/RHO(I,J)
      WRITE(6,203)J,X(I,J),Y(I,J),U(I,J),V(I,J),P(I,J),
     -      RHO(I,J),T,EDDY(I,J)
    2 CONTINUE
C ------ MAKE REGION ----------
      WRITE(6,206)
      ITEUP1=ITEU+1
      DO 3 I=ITEUP1,IL
      K=IL-I+1
      WRITE(6,200) I
      WRITE(6,202)
      DO 4 KJ=1,JL
      J=JL-KJ+1
      T=GAM*XM12*P(I,J)/RHO(I,J)
      WRITE(6,203)J,X(I,J),Y(I,J),U(I,J),V(I,J),P(I,J),
     -     RHO(I,J),T,EDDY(I,J)
    4 CONTINUE
      DO 5 J=2,JL
      T=GAM*XM12*P(K,J)/RHO(K,J)
      WRITE(6,203)J,X(K,J),Y(K,J),U(K,J),V(K,J),P(K,J),
     -       RHO(K,J),T,EDDY(K,J)
```

```
    5 CONTINUE
    3 CONTINUE
C ******   FORMATS   ********************************
  200 FORMAT (40X,'*** COLUMN',I4,2X,'***'
  202 FORMAT (3X,'J',6X,'X',I,J',6X,'Y',I,J',6X,'U',I,J)',6X,
     1'V',I,J',6X,'P',I,J',6X,'RHO',I,J',6X,'T',I,J',6X,'EDDY',
  203 FORMAT (2X,I3,8,2X,F10.4)
  204 FORMAT(1H1,40X,'------- UPPER SURFACE -------')
  205 FORMAT(1H1,40X,'------- LOWER SURFACE -------')
  206 FORMAT(1H1,40X,'------- MAKE REGION -------')
      RETURN
      END

C
C*****************************************************************
C
C SUBROUTINE "TRANSI" EVALUATES THE TRANSITION FACTOR FOR
C THE TURBULENCE MODEL
      SUBROUTINE TRANSI
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XMI,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAMII,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TURB1/ITRSU,ITRSL,IWKST,INKE,JLAST,
     -       XRELAX
      COMMON/DUMMY/ TEMP(IM,JM,4)
      COMMON/TURB2/XMUTI(JM),XMUTO(JM),D(JM),W(2*JM),F(2*JM),
     -      GAMMA(IM)
C
      DO 1 I=1,IL
      GAMMA(I)=1.0
    1 CONTINUE
C
C SET EDDY=0.0 IN LAMINAR REGION
      JLAST1=JLAST+1
      DO 2 J=1,JLAST
      DO 2 I=ITRSL,ITRSU
      EDDY(I,J)=0.0
    2 CONTINUE
      DO 3 J=JLAST1,JL
      DO 3 I=1,IL
      EDDY(I,J)=0.0
    3 CONTINUE
      DO 4 I=ITEL,ITEU
    4 EDDY(I,1)=0.0
C
      RETURN
      END

C
C*****************************************************************
C
C SUBROUTINE 'PSMOOTH' COMPUTES MACCORMACK'S PRESSURE DAMPING
C COEFFICIENT
      SUBROUTINE PSMOOTH(IXORY)
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XMI,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAMII,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEW/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
C
      IF(IXORY.EQ.2) GO TO 10
C ------------------ XXI DIRECTION ------------------
      DO 2 J=2,JL1
      DO 2 I=2,IL1
      PSMU(I,J)=ABS( (P(I+1,J)-2.*P(I,J)+P(I-1,J))/
     1  (P(I+1,J)+2.*P(I,J)+P(I-1,J)))
    2 CONTINUE
      GO TO 20
C
   10 CONTINUE
C ------------------ ETA-DIRECTION ------------------
      DO 3 J=2,JL1
      DO 3 I=2,IL1
      PSMU(I,J)=ABS( (P(I,J+1)-2.*P(I,J)+P(I,J-1))/
     1  (P(I,J+1)+2.*P(I,J)+P(I,J-1)))
    3 CONTINUE
C
   20 RETURN
      END

C
C*****************************************************************
C
C SUBROUTINE 'SPECR' COMPUTES SCALING FACTOR FOR DAMPING
C COEFFICIENT
      SUBROUTINE SPECR
C
      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM),
     1XMU(IM,JM),EDDY(IM,JM
      COMMON /GRID/ X(IM,JM),Y(IM,JM),XXI(IM,JM),YXI(IM,JM),
     1XETA(IM,JM),YETA(IM,JM),XJAC(IM,JM),S(IM,JM),SSI(IM
      COMMON /P1/ IL,JL,ILE,ITEL,ITEU
      COMMON /P2/ GAM,PR,PRT,XMI,RE,TH,S1,ALFA,ANGLE
      COMMON/MISC/IL1,IL2,IL3,JL1,JL2,JL3,ILEP1,ILEM1,
     -  ITELP1,ITELM1,ITEUP1,ITEUM1,GAM1,GAM2,GAMII,GXM,S1P,
     -  PINF,SINA,COSA,PRI,PRTI,REI,XL,XL1,XLM,PI
      COMMON/TIMEW/ DTAU(IM,JM),CFL,DTMIN,DTMAX,BETA,DTVIS
      COMMON/SPECTR/ SPECT(IM,JM),PSMU(IM,JM)
C
      DO 2 J=1,JL
      DO 2 I=1,IL
      SPECT(I,J)=ABS(YETA(I,J)*U(I,J)-XETA(I,J)*V(I,J))
     1 +SQRT(GAM*P(I,J)/RHO(I,J)*(XETA(I,J)**2+
     2 YETA(I,J)**2)*SQRT(XXI(I,J)**2+YXI(I,J)**2)
     3 +ABS(XXI(I,J)*V(I,J)-YXI(I,J)*U(I,J)
```

```
    2 CONTINUE
      RETURN
      END
/EOF
```

Appendix H: Data Reduction 1, Fortran Listing

The attached fortran listing is a data reduction code developed by the author with portions based on a code developed by Dr. Visbal (38). It was used to read in a solution output from the Navier-Stokes code (u, v, p and ρ) and produce clipped data for plotting purposes. It calculated Mach number (M), streamlines (ψ), vorticity (ω) and pressure coefficients (C_p) at each clipped point. In addition it computed skin friction (C_f) , C_p and separation points on the airfoil surface and determined the mass flux, viscous and pressure components of lift and drag. It also determined the minimum and maximum values for u, v, M, ψ, ω, C_p and C_f in the flow field. This reduction code was run on the Cray XMP computer.

```
C PROGRAM TO REDISTRIBUTE GRID LINES IN ETA-DIRECTION
C
      PROGRAM REGRID
      PARAMETER (IM=199,JM=100
      DIMENSION X(IM,JM),Y(IM,JM),S(JM),SREF(JM),SN(JM
     1          XN(IM,JM),YN(IM,JM)
C
      OPEN(UNIT=1,FILE='GRID',STATUS='OLD')
      OPEN(UNIT=2,FILE='NGRID',STATUS='NEW')
      OPEN(UNIT=3,FILE='GDATA',STATUS='OLD')
      REWIND 1
      REWIND 2
      REWIND 3
C
      IL=IM
      JE=JM
      JLN=JM
      JE=JM
C
      JLN1=JLN-1
C
      REWIND 1
      DO 1 I=1,IL
      DO 1 J=1,JE
    1 READ (1,1000)X(I,J),Y(I,J)
 1000 FORMAT(2E15.8
 1001 FORMAT(E16.9
C
      DO 2 J=1,JLN
    2 READ(5,1001) SREF(J)
C
      DO 3 I=1,IL
C
      S(1)=0.
      DO 4 J=2,JE
      DS=SQRT((X(I,J)-X(I,J-1))**2
     1 +(Y(I,J)-Y(I,J-1))**2)
      S(J)=S(J-1)+DS
    4 CONTINUE
C
      DO 5 J=1,JLN
      SN(J)=SREF(J)*S(JE)/SREF(JLN)
    5 CONTINUE
C
C INTERPOLATE IN ETA-DIRECTION
      XN(I,1)=X(I,1)
      YN(I,1)=Y(I,1)
      XN(I,JLN)=X(I,JE)
      YN(I,JLN)=Y(I,JE)
C
      K=2
      DO 6 J=2,JLN1
   20 IF(SN(J).LE.S(K)) GO TO 10
      K=K+1
      GO TO 20
   10 DX=X(I,K)-X(I,K-1)
      DY=Y(I,K)-Y(I,K-1)
      RS=(SN(J)-S(K-1))/(S(K)-S(K-1))
      XN(I,J)=X(I,K-1)+RS*DX
      YN(I,J)=Y(I,K-1)+RS*DY
    6 CONTINUE
C
    3 CONTINUE
C
      REWIND 2
      DO 7 I=1,IL
      DO 7 J=1,JLN
      WRITE(2,1000) XN(I,J),YN(I,J)
    7 CONTINUE
C
      STOP
      END
```

```
      PROGRAM STREAM
C
C THIS PROGRAM WAS WRITTEN BY CAPT PAUL D. BOYLES IN SUPPORT OF
C MY THESIS WORK AT AFIT, AUG 1988
C
C THIS PROGRAM READS IN A SET OF DATA X,Y,P,RHO AND PARAMETERS AS
C DEFINED BELOW, AND COMPUTE PSI, H , MACH#, CP AND CF VS X.
C IT ALSO COMPUTES LIFT AND DRAG IN COMPONENT FORM
C THE DATA IS PRINTED AT POINTS DEFINED BY THE PARAMETERS FOR MINIMAL
C COST IN USING THE PRINT SUBROUTINES.

      PARAMETER(IM=299,JM=100)
      COMMON /FLOWV/ U(IM,JM),V(IM,JM),P(IM,JM),RHO(IM,JM)
      COMMON/GRID/X(IM,JM),Y(IM,JM),CF(IM),ISTON(JM)
      COMMON/STRM/PSI(IM,JM),H(IM,JM),XMAC(IM,JM)
C
C     IL = MAX # POINTS I GRID
C     JL = MAX # POINTS J GRID
C     ILE = I POINT AT LEADING EDGE
C     ITEU = I POINT AT THE TRAILING EDGE ON THE UPPER SURFACE
C     ITEL = I POINT AT THE TRAILING EDGE ON THE LOWER SURFACE
C     XSTEP = MIN DELTA X STEP SIZE FOR NEXT DATA POINT OUTPUT
C     YSTEP = MIN DELTA Y STEP SIZE FOR NEXT DATA POINT OUTPUT
C     XMAX  = MAXIMUN X BOUNDARY DEFINED BY THE PROPOSED PLOT
C     XMIN  = MINIMUM X BOUNDARY DEFINED BY THE PROPOSED PLOT
C     YMAX  = MAXIMUN Y BOUNDARY DEFINED BY THE PROPOSED PLOT
C     YMIN  = MINIMUM Y BOUNDARY DEFINED BY THE PROPOSED PLOT
C     XM1 = MACH #
C
C INPUT DATA
      OPEN(UNIT= 1,FILE='TDATA',STATUS='OLD')
      OPEN(UNIT= 2,FILE='STREAM',STATUS='OLD')
      OPEN(UNIT= 4,FILE='PSIAH',STATUS='NEW')
      OPEN(UNIT=11,FILE='UV',STATUS='NEW')
      OPEN(UNIT= 7,FILE='VEL',STATUS='NEW')
      OPEN(UNIT= 3,FILE='CPCF',STATUS='NEW')
      OPEN(UNIT= 6,FILE='INFO',STATUS='NEW')
      REWIND 1
      READ (1,510) IL
      READ (1,510) JL
      READ (1,510) ILE
      READ (1,510) ITEL
      READ (1,510) ITEU
      READ (1,510) L1
      READ (1,510) L2
      READ (1,510) L3
      READ (1,510) L4
C IBODY=0 FOR NO BODY CONTOUR, =1 FOR BODY
      READ (1,520) XM1
      READ (1,520) ALPHA
      READ (1,520) RE
      READ (1,520) CV
      READ (1,520) DEL
      READ (1,510) IBODY
      READ (1,520) XSTEP
      READ (1,520) YSTEP
      READ (1,520) XMAX
      READ (1,520) XMIN
      READ (1,520) YMAX
      READ (1,520) YMIN
      READ (1,510) INUM
      READ (1,510) JNUM
      DO 3 I = 1,INUM
    3   READ(1,510) ISTON(I)
      WRITE (6,540) IL,JL,ILE,ITEL,ITEU,IBODY
      WRITE (6,541) JNUM,INUM,'ISTON I ',I=1,INUM)
      WRITE (6,550) XM1,RE,ALPHA
      WRITE (6,560) XSTEP,YSTEP,XMAX,XMIN,YMAX,YMIN
      WRITE (6,559) L1,L2,L3,L4,CV,DEL
C
  540 FORMAT(2X,6I5)
  541 FORMAT 2X,10I5
  550 FORMAT  .2X,'MACH # =',F8.3,'  RE =',F15.3,'  ALPHA =',F7.2
  560 FORMAT  .2X,2F10.6,4F8.2
  559 FORMAT  .2),'L1,L2,L3,L4= ',4I4,'  CV =',F9.5,'  DEL =',F7.2
C
      GAM=1.4
      JLM1=JL-1
      GXM=GAM*XM1*XM1
      PI = 3.14159
      ALPHA = ALPHA*PI/180.
      SINA = SIN(ALPHA)
      COSA = COS(ALPHA)
      CPMAX=0.
      CFMAX=0.
      CPMIN=1.0E+50
      CFMIN=1.0E+50
      PINF=1./GXM
      PINFI=1./PINF
      GXMI=1./GXM
      REI = 1./RE
      HMIN=1.0E+50
      HMAX=0.
      PSIMAX = 0.
      PSIMIN = 1.0E+50
      XMMAX=0.0
      UMIN=1.0E+50
      UMAX=0.0
      VMIN=1.0E+50
      VMAX=0.0
      UCONST = 1.0 * COS(ALPHA)
      VCONST = 1.0 * SIN(ALPHA)
      RHOREF = 1.0
C
C READ FLOW
      REWIND 2
C

      DO 444 J=1,JL
      DO 444 I=1,IL
      READ(2,500) X(I,J),Y(I,J),U(I,J),V(I,J),P(I,J),RHO(I,J)
  444 CONTINUE
C
C WRITE VELOCITIES TO X STATIONS IN THE WAKE
C
      DO 66 J =JNUM,1,-1
   66   WRITE 7,576  'U'ISTON(I),J=1,INUM),Y ITEU+1,J
      DO 55 J =1,JNUM
   55   WRITE 7,576   U IL-ISTON I ,J ,I=1,INUM ,Y ITEL-1,J
      WRITE 7,577   / I,I=1,INUM
```

```
  577 FORMAT(16X,1DE16.8)
C COMPUTE LIFT AND DRAG
C
      FXB = 0.0
      FXM = 0.0
      FYB = 0.0
      FYM = 0.0
      FXP = 0.0
      FYP = 0.0
      S1  = .42
      S1P = 1. + S1
      XLM = 4./3.
      XL  = -2./3
C
      DO 1 I=ITEL,ITEU
      IF (I .EQ. ITEU) GOTO 136
      K = I+1
      XETA = .5*(-X(I,3) + 4.*X(I,2) - 3.*X(I,1))
      XETAP1 = .5*(-X(K,3) + 4.*X(K,2) - 3.*X(K,1))
      YETA = .5*(-Y(I,3) + 4.*Y(I,2) - 3.*Y(I,1))
      YETAP1 = .5*(-Y(K,3) + 4.*Y(K,2) - 3.*Y(K,1))
      XXIK=X(I+1,1)-X(I,1)
      YXIK=Y(I+1,1)-Y(I,1)
      UETA=.5*(U(I+1,2)-U(I+1,1)+U(I,2)-U(I,1))
      VETA=.5*(V(I+1,2)-V(I+1,1)+V(I,2)-V(I,1))
      XETK=.5*(XETAP1+XETA)
      YETK=.5*(YETAP1+YETA)
      PK=.5*(P(I+1,1)+P(I,1))
      TP1 = GXM*P(K,1)/RHO(K,1)
      T = GXM*P(I,1)/RHO(I,1)
      XMU = SQRT(T**3)*S1P/(T+S1)
      XMUP1 = SQRT(TP1**3)*S1P/(TP1+S1)
      XMUK=.5*(XMUP1+XMU)
      UXI=U(I+1,1)-U(I,1)
      VXI=V(I+1,1)-V(I,1)
      XJK=1./(XXIK*YETK-XETK*YXIK)
      UX=XJK*(UXI*YETK-UETA*YXIK)
      VX=XJK*(VXI*YETK-VETA*YXIK)
      UY=XJK*(-UXI*XETK+UETA*XXIK)
      VY=XJK*(-VXI*XETK+VETA*XXIK)
      TXY=REI*XMUK*(UY+VX)
      TXX=REI*XMUK*(XLM*UX+XL*VY)
      TYY=REI*XMUK*(XLM*VY+XL*UX)
      DX1=-YXIK*TXX+XXIK*TXY
      DX2=-YXIK*TXY+XXIK*TYY
      TXP=-PK
      TYP=-PK
      DX5=-YXIK*TXP
      DX6= XXIK*TYP
      UA = .5*(U(I+1,1)+U(I,1))
      VA = .5*(V(I+1,1)+V(I,1))
      RA = .5*(RHO(I+1,1)+RHO(I,1))
      DX3=-RA*(-UA*UA*YXIK + UA*VA*XXIK)
      DX4=-RA*(-UA*VA*YXIK + VA*VA*XXIK)

C
      FXB =FXB + DX1
      FXM =FXM + DX3
      FYB =FYB + DX2
      FYM =FYM + DX4
      FXP =FXP + DX5
      FYP =FYP + DX6
C CALCULATION OF VORTICITY H(I,1) ON THE SURFACE
  136 XXI = .5*( X(I+1,1) - X(I-1,1) )
      YXI = .5*( Y(I+1,1) - Y(I-1,1) )
      XJC=1./(XXI *YETA-XETA*YXI )
      UX=XJC*(UXI*YETA-UETA*YXI)
      VX=XJC*(VXI*YETA-VETA*YXI)
      UY=XJC*(-UXI*XETA+UETA*XXI)
      VY=XJC*(-VXI*XETA+VETA*XXI)
      BETA = XXI*XXI + YXI*YXI
      TXY  = REI*XMU*(UY+VX)
      SIGMA = REI*XMU*(UX-VY)
      CPI  =2*(-XXI*YXI*SIGMA + ( XXI*XXI - YXI*YXI )*TXY )/BETA

    1 CONTINUE
C
      FLIFTB = -FXB *SINA + FYB*COSA
      FDRAGB =  FXB *COSA + FYB*SINA
      CLB = 2.*FLIFTB
      CDB = 2.*FDRAGB
      FLIFTM = -FXM *SINA + FYM*COSA
      FDRAGM =  FXM *COSA + FYM*SINA
      CLM = 2.*FLIFTM
      CDM = 2.*FDRAGM
      FLIFTP = -FXP *SINA + FYP*COSA
      FDRAGP =  FXP *COSA + FYP*SINA
      CLP = 2.*FLIFTP
      CDP = 2.*FDRAGP

C COMPUTE MACH NO.
C
      DO 2 J=1,JL
        DO 21 I=1,IL
          XMAC(I,J) =XMI*SQRT((U(I,J)**2+V(I,J)**2)/(GXM*P(I,J
     -        )/RHO(I,J)))
          P(I,J) =2.* P(I,J)-GXMI
   21   CONTINUE
    2 CONTINUE
C
C     ESTABLISH PSI VALUES ALONG CUTLINE AND ON SURFACE
C
      PSI(ILE,1) = 0.
      DO 200 I=ILE ,ITEU
       YH = Y(I,3) - Y(I,1)
       XH = X(I,3) - X(I,1)
       IF ( ABS(YH) .LT. 1.E-10) YH = 1.E-10
       IF ( ABS(XH) .LT. 1.E-10) XH = 1.E-10
       UY = ( -U(I,3) + 4.*U(I,2) - 3.*U(I,1) ) / YH
       VX = ( -V(I,3) + 4.*V(I,2) - 3.*V(I,1) ) / XH
       H(I,1) = VX - UY
        VUP = -.5*(V(I+1,1) + V(I,1))
        UUP =  .5*(U(I+1,1) + U(I,1))
        RHOUP = .5*( RHO(I+1,1) + RHO(I,1) ) / RHOREF
        DXU = X(I+1,1)-X(I,1)
        DYU = Y(I+1,1)-Y(I,1)
        DPSIU = (VUP*DXU - UUP*DYU)* RHOUP
        PSI(I+1,1) = PSI(I,1) + DPSIU
  200 CONTINUE
```

```
      YKH= Y(K,3) - Y(K,1)
      XKH= X(K,3) - X(K,1)
      IF ( ABS(YKH) .LT. 1.E-10) YKH = 1.E-10
      IF ( ABS(XKH) .LT. 1.E-10) XKH = 1.E-10
      UKY= ( - U(K,3) + 4.*U(K,2) - 3.*U(K,1)    ) / YKH
      VKX= ( - V(K,3) + 4.*V(K,2) - 3.*V(K,1)    ) / XKH
      W(K,1) = VKX - UKY
       ULO =  .5*(U(K-1,1) + U(K,1))
       VLO =  .5*(V(K-1,1) + V(K,1))
       RHOLO = .5*( RHO(K-1,1) + RHO(K,1) ) / RHOREF
       DXL = (X(K-1,1)-X(K,1))
       DYL = (Y(K-1,1)-Y(K,1))
       DPSIL = (VLO*DXL - ULO*DYL)* RHOLO
       PSI(K-1,1) = PSI(K,1) + DPSIL
201 CONTINUE
C
      J = 1
      K = ITEL-1
      DO 325 I=ITEU+1,IL-1
        VUP =  - .5*(V(I-1,J) + V(I,J))
        VLO =  - .5*(V(K+1,J) + V(K,J))
        DXL = (X(K,1)-X(K+1,1))
        DXU = (X(I,1)-X(I-1,1))
        RHOUP = .5*( RHO(I-1,J) + RHO(I,J)   ) / RHOREF
        RHOLO = .5*( RHO(K+1,J) + RHO(K,J)   ) / RHOREF
        DPSIU = VUP*RHOUP*DXU
        DPSIL = VLO*RHOLO*DXL
C  INTEGRATE ALONG UPPER CUT
        PSI(I,J) = PSI(I-1,J) + DPSIU
        PSI(K,J) = PSI(K+1,J) + DPSIL
        W(1,J) = (U(I,2) - U(K,2)) / (Y(I,2) - Y(K,2))
     +          -(V(I+1,1) - V(I-1,1)) / (X(I+1,1) - X(I-1,1))
        W(K,J) = (U(I,2) - U(K,2)) / (Y(I,2) - Y(K,2))
     +          -(V(K+1,1) - V(K-1,1)) / (X(K+1,1) - X(K-1,1))
        K = K -1
325 CONTINUE
C
C
C  DEFINE DELTA PSI THROUGHOUT FLOWFIELD
C
      DO 324 J=2,JL-1
      DO 224 I=2,IL-1
        UA   = .5*( U(I,J-1) + U(I,J) )
        VA   = .5*( V(I,J-1) + V(I,J) )
        RHOA = .5*( RHO(I,J-1) + RHO(I,J)   ) / RHOREF
        XXI  = .5*( X(I+1,J) - X(I-1,J) )
        YXI  = .5*( Y(I+1,J) - Y(I-1,J) )
        XETA = .5*( X(I,J+1) - X(I,J-1) )
        YETA = .5*( Y(I,J+1) - Y(I,J-1) )
        XJAC =   XXI*YETA - XETA*YXI
        IF ( ABS(XJAC).LE. 1.0E-7) WRITE(6,530) I,J,XJAC,XXI,YXI,
     +                                          XETA,YETA
        XJAC = 1. / XJAC
        UXI  = .5*( U(I+1,J) - U(I-1,J) )
        VXI  = .5*( V(I+1,J) - V(I-1,J) )
        UETA = .5*( U(I,J+1) - U(I,J-1) )
        VETA = .5*( V(I,J+1) - V(I,J-1) )
        W(I,J)= XJAC*( VXI*YETA - VETA*YXI + UXI*XETA - UETA*XXI )
        DPSI =   UA*YETA - VA*XETA )*RHOA
        PSI(I,J) = PSI(I,J-1) + DPSI
224 CONTINUE
324 CONTINUE
C
      DO 225 J=1,JL-1
        W(IL,J    ) = W(IL-1,J)
        PSI(IL,J) = PSI(IL-1,J)
        W(1,J    ) = W(2,J)
        PSI(1,J) = PSI(2,J)
225 CONTINUE
C
      DO 226 I=1,IL
        W(I,JL    ) = W(I,JL-1)
        PSI(I,JL) = PSI(I,JL-1)
226 CONTINUE
C
530 FORMAT (2X,'THE JACOBIAN IS SMALL ',2I5,5E16.8
C FIND CPMAX,CPMIN,WMAX,WMIN,XMMAX
      DO 227 I=1,IL
      DO 227 J=1,JL
        CPMAX=AMAX1(CPMAX,P(I,J))
        CPMIN=AMIN1(CPMIN,P(I,J))
        XMMAX=AMAX1(XMMAX,XMACH(I,J))
        UMAX=AMAX1(UMAX,U(I,J))
        UMIN=AMIN1(UMIN,U(I,J))
        VMAX=AMAX1(VMAX,V(I,J))
        VMIN=AMIN1(VMIN,V(I,J))
       IF (   J .EQ. 1   GOTO 227
        WMIN=AMIN1(WMIN,W(I,J))
        WMAX=AMAX1(WMAX,W(I,J))
        PSIMAX = AMAX1(PSIMAX,PSI(I,J))
        PSIMIN = AMIN1(PSIMIN,PSI(I,J))
227 CONTINUE
C
      WRITE(6,282) UMAX,VMAX
      WRITE(6,383) UMIN,VMIN
      WRITE(6,110)CPMIN,CPMAX,XMMAX
282 FORMAT(/,2X,'UMAX=',E16.9,5X,'VMAX=',E16.9
383 FORMAT(/,2X,'UMIN=',E16.9,5X,'VMIN=',E16.9
C
C OUTPUT
C
      REWIND 6
      REWIND 11
C OUTPUT BODY
      IF(IBODY.EQ.0)GO TO 3331
      IC=0
C
      DO 333 I=ITEL,ITEU
        WRITE  6,2000  X(I,1),Y(I,1)
        WRITE 11,2000  X(I,1),Y(I,1)
        IC=IC+1
333 CONTINUE
      WRITE(6,334) IC
334 FORMAT(/,2X,'NO. OF BODY POINTS = ',I5)
C
C
3331 CONTINUE
C
C
      ID=0
      IDI = 0
      JJ = 1
```

```
      IF ( Y(1,J) .GT. YMAX .OR. Y(1,J) .LT. YMIN ) GOTO 400
      IF ( J .EQ. 1 .OR. J .EQ. 2 ) GOTO 41
      IF ( ABS( Y(1,J)-Y(1,JJ) ) .LT. YSTEP ) GOTO 400
 41   IDI = IDI + 1
      JJ = J
      JDI = 0
      II= 1
      DO 4 I=1,IL
        IF ( X(I,1) .GT. XMAX .OR. X(I,1) .LT. XMIN ) GOTO 4
        IF (I .EQ. ILE .OR. I .EQ. ITEL .OR. I .EQ. ITEU) GOTO 42
        IF ( ABS( X(I,1) - X(II,1) ) .LT. XSTEP ) GOTO 4
 42     II = I
        JDI=JDI+1
        WRITE(4 ,2000) X(I,J),Y(I,J),PSI(I,J),W(I,J),XMAC(I,J)
        WRITE(11,2000) X(I,J),Y(I,J),U(I,J) ,V(I,J),P(I,J)
        ID=ID+1
 4    CONTINUE
400 CONTINUE
C
      WRITE(6,45) ID
 45   FORMAT(/,2X,'NO. OF PSI,W,XMAC,P,U,V POINTS =',1X,I5)
      WRITE(6,44) IDI,JDI
 44   FORMAT(2X,/,2X,'OUT=',I5,2X,'IN=',I5,/)
C
C
      REWIND 3
      CFL = 0.0
      ISEP = 0
      SIGN = -2.0
      DO 5 I=ITEL,ITEU
C  CALCULATE LOCAL SKIN FRICTION
        IF ( I .GE. ITEL .AND. I .LT. ILE ) CF(I) = -CF(I)
        WRITE(3,2000)X(I,1),P(I,1),CF(I)
        CFMAX=AMAX1(CFMAX,CF(I))
        CFMIN=AMIN1(CFMIN,CF(I))
        CPMAX=AMAX1(CPMAX,P(I,1))
        CPMIN=AMIN1(CPMIN,P(I,1))

        IF ( ( CFL .LT. 0.0 .AND. CF(I) .GE. 0.0 ) .OR.
     +       ( CFL .GT. 0.0 .AND. CF(I) .LE. 0.0 ) ) THEN
          IF ( I .GT. ILE ) WRITE(6,561) X(I-1,1),X(I,1)
          IF ( I .LT. ILE ) WRITE(6,562) X(I,1),X(I-1,1)
          XSEP = -CFL*(X(I-1,1)-X(I,1))/(CFL-CF(I)) + X(I-1,1)
          WRITE(6,564)XSEP
          ISEP = 1
        END IF
        CFL = CF(I)
 5    CONTINUE
C
      IF ( ISEP .EQ. 0 ) WRITE(6,563)
C
 561  FORMAT(2X,'SEPERATION ON THE UPPER SURFACE OCCURS BETWEEN ',F8.4,
     +       ' AND ',F8.4)
 562  FORMAT(2X,'SEPERATION ON THE LOWER SURFACE OCCURS BETWEEN ',F8.4,
     +       ' AND ',F8.4)
 563  FORMAT(2X,'SEPERATION DID NOT OCCUR ON THE SURFACE',/)
 564  FORMAT(2X,'BY INTERPOLATION, SEPERATION OCCURS AT X   = ',F8.4,//)

C
      WRITE(6,105)CPMIN,CPMAX
      WRITE(6,106)CFMIN,CFMAX
      WRITE(6,120) WMIN,WMAX
      WRITE(6,130) PSIMIN,PSIMAX
      WRITE(6,533) CLB+CLM+CLP,CLP,CLB,CLM
      WRITE(6,531) CDB+CDM+CDP,CDP,CDB,CDM
C
 110  FORMAT(2X,"CPMIN=",F12.5,2X,"CPMAX=",F12.5,2X,
     - "MAX. MACH NO.=",F12.5)
 105  FORMAT(2X,"SURFACE CPMIN=",F12.5,2X,"CPMAX=",F12.5,2X,/)
 106  FORMAT(2X,"SURFACE CFMIN=",F12.5,2X,"CFMAX=",F12.5,2X,/)
 120  FORMAT(2X,"MIN. VORT.=",E16.9,2X,"MAX. VORT.=",E16.9,/)
 130  FORMAT(2X,'MIN. PSI. =',E16.9,2X,'MAX. PSI. =',E16.9,//)
 533  FORMAT(2X,'CL =',F13.6,2X, ,2X,'CL DUE TO PRESSURE, BODY'
     +       ,' MASS ',3F13.6,/)
 531  FORMAT(2X,'CD =',F13.6,2X, ,2X,'CD DUE TO PRESSURE, BODY'
     +       ,' MASS ',3F13.6)
 500  FORMAT (1X,6E16.9
 510  FORMAT (17
 520  FORMAT (F16.10)
C
1000  FORMAT (7E16.8)
2000  FORMAT (7E16.8)
C
      CLOSE(1)
      CLOSE(2)
      CLOSE(7)
      CLOSE(11)
      CLOSE(4)
      CLOSE(6)
      CLOSE(3)
      STOP
      END
```

Appendix I: Data Reduction Program 2, Fortran Listing

The attached fortran listing is a data reduction code developed by the author. It was used to read in a solution output from the Navier-Stokes code (N, C_l, C_d and u, v, and ρ time step residuals) and calculated average , average maximum and average minimum values and average frequency for C_l, C_d, and the residuals. In addition it produced plotting files for N vs maximum C_l and C_d and N vs minimum C_l and C_d. This reduction code was run on the Cyber computer.

```
      PROGRAM REDCL
C
C THIS PROGRAM WAS WRITTEN BY CAPT PAUL D. BOYLES IN SUPPORT OF
C MY THESIS WORK AT AFIT, OCT 1988
C
C THIS PROGRAM READS IN A SET OF DATA N,CL,CD,VRMS,URMS,RES
C THE AVERAGE VALUES ARE THEN COMPUTED OVER TIME AND THE MIN MAX VALUES

      PARAMETER(IM=20000,NM=1000,JM=4)
      COMMON /FLOWV/ N(NM,JM),CLMAX(NM),CLMIN(NM),CDMAX(NM),
     +                          CDMIN(NM)
C
C     NL = MAX # POINTS I GRID
C     NS = INITIAL NSTEP TO BEGIN ANALYSIS
C     NF = FINAL NSTEP FOR ANALYSIS
C     YMAX  = MAXIMUM Y BOUNDARY DEFINED BY THE PROPOSED PLOT
C     YMIN  = MINIMUM Y BOUNDARY DEFINED BY THE PROPOSED PLOT
C     XM1 = MACH #
C
C INPUT DATA
      OPEN(UNIT= 1,FILE='REDDAT',STATUS='OLD')
      OPEN(UNIT= 2,FILE='INFO' ,STATUS='OLD')
      OPEN(UNIT= 3,FILE='STUF',STATUS='UNKNOWN')
      OPEN(UNIT= 4,FILE='CLDAT' ,STATUS='UNKNOWN')
      OPEN(UNIT= 5,FILE='CDDAT' ,STATUS='UNKNOWN')
      OPEN(UNIT= 6,FILE='CLCD' ,STATUS='UNKNOWN')
      REWIND 1
      READ (1,510) NS
      READ (1,510) NF
      READ (1,520) XM1
      READ (1,520) ALPHA
      READ (1,520) RE
      READ (1,520) CV
      READ (1,520) DEL
      WRITE (3,540) NS,NF
      WRITE (3,550) XM1,RE,ALPHA
      WRITE (3,559) CV,DEL
C
  510 FORMAT(I7)
  520 FORMAT(F12.5)
  540 FORMAT(/,2X,2I7)
  550 FORMAT(/,2X,'MACH # =',F8.3,'  RE =',F15.3,'  ALPHA =',F7.2)
  559 FORMAT(/,2X,'  CV =',F9.5,'  DEL =',F7.2)
C
      REWIND 2
C
      NL = 0
      CLL = 0
      CDL = 0
      CLXL = 0
      CDXL = 0
      NCLMX = 0
      NCLMN = 0
      NCDMX = 0
      NCDMN = 0
      CLCDA = 0
      CLA = 0
      CDA = 0
      CPA = 0
      UA  = 0
      VA  = 0
      RA  = 0

  500 FORMAT (10E13.5
      DO 444 I=1,IM
        READ(2,500) RT,TAU,XLOG,YLOG,CL,CD,UNORM,VNORM,RES,CP
        NT = RT
        IF ( NT .LT. NS ) THEN
          CLL = CL
          CDL = CD
          NTL = NT
          GOTO 444
        END IF
        NL = NL + 1
        CLX = CLL-CL
        CDX = CDL-CD
        WRITE(6,500)RT,CL,CD,CL/CD
        IF ( CLX*CLXL .LE. 0.0 .AND. CLX .GT. 0.0 ) THEN
          NCLMX  = NCLMX + 1
          CLMAX(NCLMX) = CLL
          N(NCLMX,1) = NTL
        END IF

        IF ( CLX*CLXL .LE. 0.0 .AND. CLX .LT. 0.0 ) THEN
          NCLMN  = NCLMN + 1
          CLMIN(NCLMN) = CLL
          N(NCLMN,2) = NTL
        END IF
        IF ( CDX*CDXL .LE. 0.0 .AND. CDX .GT. 0.0 ) THEN
          NCDMX  = NCDMX + 1
          CDMAX(NCDMX) = CDL
          N(NCDMX,3) = NTL
        END IF

        IF ( CDX*CDXL .LE. 0.0 .AND. CDX .LT. 0.0 ) THEN
          NCDMN  = NCDMN + 1
          CDMIN(NCDMN) = CDL
          N(NCDMN,4) = NTL
        END IF
        CLL = CL
        CDL = CD
        NTL = NT
        CLXL = CLX
        CDXL = CDX
        CLA = CL + CLA
        CLCDA = CL/CD + CLCDA
        CDA = CD + CDA
        CPA = CP + CPA
        UA = UNORM + UA
        VA = VNORM + VA
        RA = RES + RA

        IF ( NT .GE. NF ) GOTO 100
  444   CONTINUE

  100   CLA = CLA / NL
        CDA = CDA / NL
        CLCDA = CLCDA / NL
        CPA = CPA / NL
        UA  = UA  / NL
```

```
      VA  = VA  / NL
      RA  = RA  / NL
      WRITE(3,561) CLA,CDA,CPA
      WRITE(3,566) CL/CDA,CLA/CDA
      WRITE(3,562) UA,VA,RA
      WRITE(3,*) NL,NT,NS,NF
      CLMAXA = 0
      CLMINA = 0
      CDMAXA = 0
      CDMINA = 0
      NL = NS
      DO 200 I=1,NCLMX
        CLMAXA = CLMAXA + CLMAX(I)
        WRITE(4,560) N(1,1),CLMAX(I),N(1,1)-NL
200     NL = N(1,1)
        NL = NS
      DO 210 I=1,NCLMN
        CLMINA = CLMINA + CLMIN(I)
        WRITE(4,560) N(1,2),CLMIN(I),N(1,2)-NL
210     NL = N(1,2)
        NL = NS
      DO 220 I=1,NCDMX
        CDMAXA = CDMAXA + CDMAX(I)
        WRITE(5,560) N(1,3),CDMAX(I),N(1,3)-NL
220     NL = N(1,3)
        NL = NS
      DO 230 I=1,NCDMN
        CDMINA = CDMINA + CDMIN(I)
        WRITE(5,560) N(1,4),CDMIN(I),N(1,4)-NL
230     NL = N(1,4)

      IF ( NCLMX .EQ. 0 .OR. NCLMN .EQ. 0 ) THEN
        WRITE(3,*) 'SOLUTION IS NOT PERIODIC'
        GOTO 150
      END IF
      CLMAXA = CLMAXA/NCLMX
      CLMINA = CLMINA/NCLMN
      CDMAXA = CDMAXA/NCDMX
      CDMINA = CDMINA/NCDMN

      WRITE(3,563) CLMAXA,CLMINA,NCLMX,NCLMN
      WRITE(3,564) CDMAXA,CDMINA,NCDMX,NCDMN
150   CONTINUE
560   FORMAT(I7,F13.5,I7)
561   FORMAT(' ',2X,'CL AVG = ',F9.6,'  CD AVG = ',F9.6,'  CP AVG = ',F9.6
566   FORMAT(' ',2X,'CL/CD AVG = ',F9.6,'  CL AVG/CD AVG = ',F9.6,
562   FORMAT(' ',2X,'URMS A = ',E9.4,'  VRMS A = ',E9.4,'  DR/DT = ',E9.4
563   FORMAT(' ',2X,'CLMAX AVG = ',F9.6,'  CLMIN AVG = ',F9.6,
     +          '  NCL = ',2I5)
564   FORMAT(' ',2X,'CDMAX AVG = ',F9.6,'  CDMIN AVG = ',F9.6,
     +          '  NCD = ',2I5)

      END
```

Appendix J: Geometric Progression Program, Fortran Listing

The attached code is the original code used in this work as supplied by Dr. Miguel Visbal (37). This code generates a geometric spacing in the η direction of the computational domain. A line of constant η can be specified at which the spacing becomes uniform. This allowed the geometric progression to be controlled when the outer boundary was large. For the original grids used in this work, step sizes in the η direction at the outer boundary, 20 cord lengths from the airfoil surface, were 2.7 cord lengths. With the use of this code and the redistribution code (appendix k), the spacing for the same grid became a constant step size of .6 cord lengths from a distance of 5 cord lengths to 20. This code was run on the Cyber computer.

```
C PROGRAM TO REDISTRIBUTE GRID LINES IN ETA-DIRECTION
C
      PROGRAM REGRID
      PARAMETER (IM=199,JM=100)
      DIMENSION X(IM,JM),Y(IM,JM),S(JM),SREF(JM),SN(JM)
     1         XN(IM,JM),YN(IM,JM)
C
      OPEN(UNIT=1,FILE='GRID',STATUS='OLD')
      OPEN(UNIT=2,FILE='NGRID',STATUS='NEW')
      OPEN(UNIT=3,FILE='GDATA',STATUS='OLD')
      REWIND 1
      REWIND 2
      REWIND 3
C
      IL=IM
      JL=JM
      JLN=JM
      JE=JM
C
      JLN1=JLN-1
C
      REWIND 1
      DO 1 I=1,IL
      DO 1 J=1,JL
    1 READ (1,1000)X(I,J),Y(I,J)
 1000 FORMAT(2E15.8)
 1001 FORMAT(E16.9)
C
      DO 2 J=1,JLN
    2 READ(5,1001) SREF(J)
C
      DO 3 I=1,IL
C
      S(1)=0.
      DO 4 J=2,JE
      DS=SQRT((X(I,J)-X(I,J-1))**2
     1 +(Y(I,J)-Y(I,J-1))**2)
      S(J)=S(J-1)+DS
    4 CONTINUE
C
      DO 5 J=1,JLN
      SN(J)=SREF(J)*S(JE)/SREF(JLN)
    5 CONTINUE
C
C INTERPOLATE IN ETA-DIRECTION
      XN(I,1)=X(I,1)
      YN(I,1)=Y(I,1)
      XN(I,JLN)=X(I,JE)
      YN(I,JLN)=Y(I,JE)
C
      K=2
      DO 6 J=2,JLN1
   20 IF(SN(J).LE.S(K)) GO TO 10
      K=K+1
      GO TO 20
   10 DX=X(I,K)-X(I,K-1)
      DY=Y(I,K)-Y(I,K-1)
      RS=(SN(J)-S(K-1))/(S(K)-S(K-1))
      XN(I,J)=X(I,K-1)+RS*DX
      YN(I,J)=Y(I,K-1)+RS*DY
    6 CONTINUE
C
    3 CONTINUE
C
      REWIND 2
      DO 7 I=1,IL
      DO 7 J=1,JLN
      WRITE(2,1000) XN(I,J),YN(I,J)
    7 CONTINUE
C
      STOP
      END
```

Appendix K: Grid Redistribution Program, Fortran Listing

The attached code is a modified version of the original code supplied by Dr. Miguel Visbal (37). This code reads in a grid produced by the hyperbolic grid generator (appendix G) and the spacing geometry as output from the geometric progression program (appendix J). It linearly interpolates the points in the original grid and redistributes the spacing in the η direction to match the new spacing. A new grid is produced with better spacing characteristics at the outer boundary. This code was run on the Cray XMP.

```
C PROGRAM "STRETCH1" DETERMINES THE CORRECT VALUES FOR
C THE STRETCHING PARAMETERS REQUIRED IN THE MESH GENERATION
C CODE  FOR THE DISTRIBUTION OF THE ETA-LINES
C THIS CASE: EXPONENTIAL DISTRIBUTION OF THE ETA-LINES FROM
C ETA=0 TO ETA=ETA1 , AND UNIFORM  FROM ETA=ETA1 TO ETA=1.
C
      PROGRAM STRECH (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE1)
C
      DIMENSION S(399)
C INPUT DATA
      JL=100
      J0=90
      JC1=2
      SMAX=20.0
      DS0=.0001
      INVERS=0
      C1=10.0
      SIGN=+1.
      X0=0.0
C
      JL1=JL-1
      J0M1=J0-1
      JC1M1=JC1-1
      DE=1.0/FLOAT(JL1)
      ETA1=DE*FLOAT(J0M1)
      ETA1C=DE*FLOAT(JC1M1)
C
C
      N=0
   10 N=N+1
      IF (N.GT.50) GO TO 20
      EC1=EXP(C1)
      X1=EXP(C1*ETA1C/ETA1)
      F=ETA1*(EC1-1.0-C1*EC1*(1.0-ETA1
      G1=SMAX*ETA1*(X1-1.0)
      F1=F*DS0-G1
      DFDC1=F-ETA1*EC1*(1.0-ETA1)
      DG1DC1=SMAX*ETA1C*X1
      DF1DC1=DS0*DFDC1-DG1DC1
      XJAC=DF1DC1
      DC1=-F1/XJAC
      C1=C1+DC1
      IF(ABS(DC1/C1).GT.0.0001) GO TO 10
      EC1=EXP(C1)
      F=ETA1*(EC1-1.0-C1*EC1*(1.0-ETA1)
C
C COMPUTATION OF S(J)
      S(1)=0.0
      DO 400 J=2,J0M1
      ETA=DE*(J-1)
  400 S(J)=S(1)+ETA1*SMAX*(EXP(C1*ETA/ETA1)-1.0)/F
      DO 401 J=J0,JL
      ETA=DE*(J-1)
  401 S(J)=S(1)+SMAX*(ETA1*(EC1-1.0)+C1
     1*EC1*(ETA-ETA1))/F
  300 CONTINUE
C
C COMPUTE STRETCHING FACTOR
      SF=EXP(C1/FLOAT(J0M1)
C
C OUTPUT
      WRITE (6,200) JL,J0,JC1,SMAX,DS0,C1
      WRITE (6,204) SF
      DO 1 J=1,JL
      S(J)=X0+SIGN*S(J)
      IF (J.GE.2) DS=S(J)-S(J-1)
    1 WRITE (6,201) J,S(J),DS
C CREATE OUTPUT FILE
      REWIND 1
      IF (INVERS.EQ.1) GO TO 500
      DO 2 J=1,JL
    2 WRITE (1,1000) S(J)
      GO TO 20
  500 DO 3 L=1,JL
      J=JL-L+1
    3 WRITE (1,1000) S(J)
C FORMATS
  200 FORMAT (2X,"JL=",I3,2X,"J0=",I3,2X,"JC1=",I3,2X,
     1"SMAX=",F12.6,2X,"DS0=",E12.5,2X,"C1=",F8.6)
  201 FORMAT (10X,I3,5X,"S(J)=",E12.5,5X,"DS=",E12.5)
  204 FORMAT (20X,"STRETCHING FACTOR=",F12.5)
 1000 FORMAT E16.9
C
   20 STOP
      END
```

Bibliography

1. Abbott, Ira H. and von Doenhoff, Albert E. Theory of Wing Sections. New York: McGraw-Hill Book Co., 1949.

2. Anderson, John C. and others. Computational Fluid Mechanics and Heat Transfer. New York: Hemisphere Publishing Corp., 1984.

3. Ashley, Holt and Landahl, Marten. Aerodynamics of Wings and Bodies. New York: Dover Publications Inc., 1965.

4. Beam, Richard M. and Warming, R. F. "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", AIAA 3rd Computational FLUID Dynamics Conference. June 1987.

5. Bearman, P. W. "Vortex Shedding From Oscillating Bluff Bodies", Annual Review In Fluid Mechanics. 199-221. (1984).

6. Beran, Philip S. Lecture notes from AERO 521, Dynamics of Viscous Flow. School of Engineering, Air Force Institute of Technology, WPAFB OH, June-August 1988.

7. -----. Professor Air Force Institute of Technology. Private Communication. AFIT/ENY, WPAFB OH, July-November 1988.

8. Baldwin, B. S. and Lomax H. "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows", AIAA 16th Aerospace Sciences Meeting. January 1978. AIAA-78-257.

9. Cebeci, Tuncer and Smith, A. M. O. Analysis of Turbulent Boundary Layers. Orlando: Academic Press Inc, 1974.

10. Cebeci, T. "The Calculation of Flow Over Iced Airfoils", AIAA 26th Aerospace Sciences Meeting. January 1988. AIAA-88-0112.

11. Emanuel, George. Gas Dynamics: Theory and Applications. New York: AIAA Inc., 1986.

12. Fant, A. Lecture Notes for AERO 538, Dynamics of Compressible Flow. School of Engineering, Air Force Institute of Technology, October-December 1988.

13. Halim, Ahmad A. M. "A Global Marching Technique for the Prediction of Separated Flows over Arbitrary Airfoils", AIAA Journal, Vol. 25, No. 9, Septemper 1987.

14. Huband, Gary W. Numerical Study of Supersonic Flows Using Different Techniques. MS Thesis, AFIT/GA/AA/86D-8. Air Force Institute of Technology, WPAFB OH, December 1986.

15. Hughes, William F. and Gaylord, Eber W. Basic Equations of Engineering Science. Schaum's Outline Series. New York: McGraw-Hill Book Co., 1964.

16. Jochum, Keith B. Numerical Study of High Speed Viscous Flows. MS Thesis, AFIT/GAE/AA/86D-4. Air Force Institute of Technology, WPAFB OH, December 1986.

17. Karamcheti, Krishnamurty. Principles of Ideal-Fluid Aerodynamics. Malabar: Robert E, Kreiger Publishing Co., 1966.

18. King, P. Class lecture notes from AERO 827, Turbulent Flow. School of Engineering, Air Force Institute Of Technology, WPAFB OH, January-March 1988.

19. Kinsey, Don W. and Barth, Timothy J. "Description of a Hyperbolic Grid Generating Procedure For Arbitrary Two-Dimensional Bodies", AFWAL-TM-84-FIMM. WPAFB OH, July 1984.

20. Kohlman, David L. Introduction to V/STOL Airplanes. Ames: Iowa State University Press, 1981.

21. Kreyzig, Erwin. Advanced Engineering Mathematics (Fifth Edition). New York: John Wiley And Sons, 1983.

22. Kueth, Arnold M. and Chow, Chuen-Yen. Foundations of Aerodynamics (Third Edition). New York: John Wiley And Sons, 1976.

23. McCormick, Barnes W. jr. Aerodynamics of V/STOL Flight. Orlando: Academic Press Inc., 1967.

24. NATO. Experimental Data Base for Computer Program Assessment. AGARD-AR-138. London: Technical Editing and Reproduction Ltd.

25. Peyret, Roger and Thomas, Taylor D. Computational Methods for Fluid Flow. New York: Springer-Verlag, 1983.

26. Pope, Alan. Basic Wing and Airfoil Theory. New York: McGraw-Hill Book Co., 1951.

27. Rizzeta, Don. Aerospace Engineer. Private Communication. AFWAL/FIMM, WPAFB OH, July-November 1988.

28. Rodenbaugh, Bill. "The Influential Variables of V/STOL Propulsion", Proceedings of the First National V/STOL Aircraft Symposium. 1-29. WPAFB OH, November 1965.

29. Rubin, S. G. and Reddy, D. R. "Global PNS Solutions and Laminar and Turbulent Flow", AIAA-83-1911.

30. Robinson, A. and Laurman, M. A. Wing Theory. Cambridge: Cambridge University Press, 1956.

31. Schlichting, Hermann. Boundary Layer-Theory (Seventh Edition). New York: McGraw-Hill Book Co., 1979.

32. Shang, Joeseph A. Chief Computational Fluids Group. Discussions. AFWAL/FIMM, WPAFB OH, July-November 1988.

33. Steiger, Joseph L. and Chaussee, Denny S. "Generation of Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations", SIAM J. Sci. Stat. Comput. Vol 4. December 1980.

34. Visbal, Miguel R. Aerospace Engineer. Private Communication. AFWAL/FIMM, WPAFB OH, July-November 1988.

35. -----. "Calculation of Viscous Transonic Flows About a Super Critical Airfoil. AFWAL-TR-86-3013. WPAFB OH, July 1983.

36. -----. Original Code for Beam and Warming Implicit Scheme. August 1985.

37. -----. Original Code for Grid Redistribution. 1988.

38. -----. Original Code for Streamline and Vorticity Calculations. 1988.

39. -----. Prepared Notes on Lift and Drag Calculations. July-November 1988.

40. Warming, R. F. and Beam, Richard M. "On the Construction and Application of Implicit Factored Schemes for Conservation Laws", SIAM-AMS-Proceedings Volume 11. 85-129. 1978.

41. White, Frank M. Viscous Fluid Flow. New York: McGraw-Hill Book Co., 1974.

Vita

Captain Paul D. Boyles, Jr. [redacted] attended High Point College in High Point, North Carolina, from which he received the degree of Bachelor of Science in Mathematics with a minor in Chemistry in May 1982. Upon graduation, he entered the Air Force Officer's Training School and received his commission in August 1982. He then entered the Air Force Institute of Technology Undergraduate Engineering conversion degree program at Parks Air College of St. Louis University in Cahokia, Illinois. He graduated *cum laude* with the degree of Bachelor of Science in Aerospace Engineering in July 1984. He then served as a staff officer for Headquarters Tactical Air Command, Deputy Chief of Staff, Requirements at Langley Air Force Base, Virginia, until entering the School of Engineering, Air Force Institute of Technology, in June, 1987.

[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

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 Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/AA/88D-02	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (*If applicable*) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
NAVIER-STOKES SOLUTION FOR A NACA 0012 AIRFOIL (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Paul D. Boyles, Jr., B. S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988 December	186

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Computational Fluid Dynamics, Fluid Dynamics, Navier-Stokes equations, Mass Transfer, Fan
20	04		

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

Thesis Chairman: Ahmad Halim
Associate Professor of Aerospace Engineering

(see reverse side)

12 Jan 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	
22a. NAME OF RESPONSIBLE INDIVIDUAL [illegible]	22b. TELEPHONE (*Include Area Code*) (513) 255-3030
	22c. OFFICE SYMBOL AFIT/ENA

DD Form 1473, JUN 86 *Previous editions are obsolete*

SECURITY CLASSIFICATION OF THIS PAGE

STOL aircraft use a variety of mechanisms to augment lift. Small fans with vectored exhaust imbedded in an aircraft wing could increase lift and reduce drag. The aim of this thesis is to investigate the two-dimensional effect of a small "fan in wing" on the flow field and on the lift and drag behavior of a NACA 0012 airfoil.

Numerical solutions are obtained for a Mach number of 0.3 and a Reynolds number of one million. The parameters examined are angle of attack, fan ejection angle and suction velocity. The numerical code used is based on the Beam-Warming implicit factorization algorithm for solving the two-dimensional mass-averaged compressible Navier-Stokes equations for viscous, unsteady flows.