NW

N84-13491 #

A FULLY VECTORIZED NUMERICAL SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

By

NISHEETH PATEL

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in the College of Engineering

Mississippi State, Mississippi

December 1983

A FULLY VECTORIZED NUMERICAL SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

By

NISHEETH PATEL

[illegible]
Professor of Aerospace Engineering (Chairman of Committee and Dissertation Director

[illegible]
Professor of Mathematics (Member of Committee)

[illegible]
Professor of Aerospace Engineering (Member of Committee)

[illegible]
Professor of Aerospace Engineering (Member of Committee)

Keith Koenig
Associate Professor of Aerospace Engineering (Member of Committee)

[illegible]
Professor and Head of the Department of Aerospace Engineering

[illegible]
Dean of the College of Engineering

[illegible]
Vice President for Graduate Studies and Research

December 1983

To

Sadhna and Timmy, longtime cherished friends

— With my sincere appreciation for their strong support and encouragement

ACKNOWLEDGEMENTS

Thanks to Dr. Joe F. Thompson, my advisor, for his valuable guidance and assistance.

Thanks to Mr. Charles B. Cliett, Head of the Department of Aerospace Engineering, for his moral support.

Thanks to the members of the graduate committee for their time in reviewing this dissertation.

Thanks to Mrs. Rita Curry for a most effectively handled typing job.

Thanks to my firends, who contributed toward my learning. They are too numerous to mention but they know who they are.

Thanks to my parents for their effort, encouragement and understanding.

Thanks to the NASA Langley Research Center who supported this research through NASA Grant NSG-25-001-055.

Nisheeth Patel
Mississippi State University
November 1983

ABSTRACT

Nisheeth Patel, Doctor of Philosophy, 1983

Major: Engineering, Department of Aerospace Engineering

Title of Dissertation: A Fully Vectorized Numerical Solution of the Incompressible Navier-Stokes Equations

Directed by: Joe F. Thompson

Pages in Disseration: 160 Words in Abstract: 255

Abstract

A vectorizable algorithm is presented for the implicit finite difference solution of the incompressible Navier-Stokes equations in general curvilinear coordinates. The unsteady Reynolds averaged Navier-Stokes equations solved are in two-dimension and non-conservative primitive variable form. A two-layer algebraic eddy viscosity turbulence model is used to incorporate the effects of turbulence. Two momentum equations and a Poisson pressure equation, which is obtained by taking the divergence of the momentum equations and satisfying the continuity equation, are solved simultaneously at each time step. An elliptic grid generation approach is used to generate a boundary-conforming coordinate system about an airfoil. The governing equations are expressed in terms of the curvilinear coordinates and are solved on a uniform rectangular computational domain. A checkerboard SOR, which can effectively utilize the computer architectural concept of vector processing, is used for iterative solution of the governing equations. The method is applied to the cases of an 18% thick NACA 66_3018 airfoil at zero degree angle of attack for chord Reynolds number range of 1000-40,000.

The effects of various boundary-conforming coordinate systems, artificial viscosities, smoothers, down-stream boundary conditions, initial guesses and number of iterations during the acceleration phase on the solution of the flow field are studied. Numerical results are given in terms of surface pressure distributions and velocity vector fields at selected times. Computed steady-state results are compared with experimental data. On the CDC CYBER-203 computer, the algorithm demonstrated a factor of about 11 improvement over a CDC-7600 scalar version of the code.

TABLE OF CONTENTS

Page

NOMENCLATURE

Symbols

C_p	Pressure Coefficient
D	Divergence of Velocity Vector
$\underset{\sim}{g}$	Body-Force Vector
i,j	Cartesian Unit Vectors
IL, JL	Number of ξ and η-lines in Transformed Plane
J	Jacobian of Coordinate Transformation
ℓ	Characteristic length
$\underset{\sim}{n}$	Unit Normal
p	Pressure
P,Q	Coordinate Control Functions
Re	Reynolds number
t	Time
u	x-component of velocity
v	y-cinoibebt of velocity
$\underset{\sim}{V}$	Velocity vector
$\underset{\sim}{U}_\infty$	Free-Stream Velocity
x,y	Cartesian Coordinates
ξ,η	Curvilinear Coordinates
$\alpha,\beta,\gamma,\sigma,\tau$	Coordinate Transformation Parameters
ψ	Angle of Attack
Ω	Artificial Viscosity Proportionality Factor
ϕ	Parameter to limit the magnitude of Artificial Viscosity
ϵ	Eddy viscosity
μ	Viscosity
μ_a	Artificial Viscosity

ζ	Density
Σ_{ij}	Stress Tensor
ω	Vorticity
κ	Acceleration Parameter
δ_{ij}	Kroncker Delta
$\underset{\sim}{\nabla}$	Del-Operator $\underset{\sim}{\nabla} = \underset{\sim}{i} \frac{\partial}{\partial x} + j \frac{\partial}{\partial y}$
Δ	Small Increment

Superscripts

b	Values at Black Nodes
r	Values at Red Nodes
'	Denotes First Derivaitve
"	Denotes Second Derivative

Subscripts

i,j	Evaluation at Grid Point $\xi = i$, $\eta = j$
L	Denotes Lower Down-Stream Boundary
max	Denotes Maximum
min	Denotes Minimum
NWS,NWE	Denotes Trailing i - Index Value on Lower and Upper Surfaces of Airfoil
u	Denotes Upper Down-Stream Boundary
t,x,y,ξ,η	Denotes First Partial Differentiation
$xx,yy,\xi\xi,\eta\eta$	Denotes Second Partial Differentiation
ξ,η	Denotes Cross Partial Differentiation
∞	Denotes Infinity Condition
w	Denotes Wall Condition

LIST OF FIGURES

Chapter I

INTRODUCTION

Computational Fluid Dynamics (CFD) has made a significant contribution in the recent development of aerospace vehicles. Practical aerodynamics, which controls the design of flight vehicles, is essentially about complex flow at high Reynolds number past arbitrary configurations. The governing equations, which describe physical features of such a flow, are non-linear partial differential equations - the Navier-Stokes equations. Simplification of these governing equations will limit the application. In the past, experimental fluid dynamics has played an important role, however, with the breakthrough in solving non-linear partial differential equations and high speed computation, CFD has risen to complement the role of experimental fluid dynamics.

Perhaps the foundation of modern fluid dynamics was laid by Prandtl when he first presented boundary layer theory in 1904. However, it was not recognized until 1920 when Prandtl presented insight on separation. A review of classical fluid dynamics has been presented by Goldstein [1]. The foundation stone for CFD was probably laid by Courant, Friedricks and Lewy [2] with the introduction of the numerical stability condition for the solution of hyperbolic equations, known as CFL condition. In the 1960's the finite difference methods, such as Lax -Wendroff [3] type and MacCormack's [4] explicit methods, were developed for solving the Euler equations in conservation law form. Since then, with rapid increase in computer speed and computer memory, CFD has developed sufficiently to become established as a discipline.

Several recent surveys on CFD development and future of CFD have been presented in references [5], [6], [7], and [8].

Although in most cases CFD does offer the potential of obtaining complete information about complex flows without experimentation, it has its own difficulties. The essential areas to be considered before solving the Navier-Stokes equations are grid generation, algorithm, turbulence model, and computer. Significant improvement in any of these areas will enhance computational efficiency and accuracy of the solution.

Many cases of practical interest contain an arbitrary domain. Since the boundary is not aligned with the grid when using a cartesian coordinate for an arbitrary region, it requires the use of interpolation formulas near the boundary. The imposition of boundary conditions with a complicated computational region having irregular boundaries is a primary difficulty with the cartesian coordinate system. Moreover, the Navier-Stokes equations and their boundary conditions are such that the viscous effects are confined to a very thin region immediately adjacent to the solid boundaries. Although the region is quite thin it produces considerable effects on the total solution of the flow field. In addition, the stability conditions, iterative convergence and truncation errors of the numerical algorithm employed may be adversely affected [9]. Since a cartesian grid has limited applications, the recent trend has been to use a boundary-conforming coordinate system. The boundary-conforming coordinate systems are defined as those which possess constant coordinate lines coincident with all boundaries of the physical plane, which in turn correspond to a rectangular grid with square cells in the transformed plane. The governing continuum equations are derived on the rectangular grid in the transformed plane. An elliptic grid generating system developed by Thompson, et. al., [10] is capable of generating a boundary-conforming coordinate system. The main attraction of this approach is flexibility, automation and a moderate degree of control by the user. The recent surveys on grid generation techniques and applications are presented by Thompson in references [11], [12], and [13].

In the past decade, numerical algorithms used in simulation of fluid flows have improved substantially. Explicit algorithms are simple. However, restriction on the time step imposed by stability considerations is a main disadvantage of these schemes. Increased interest in implicit schemes led to the development and use of efficient algorithms such as those due to Briley and McDonald [14], Beam and Warming [15], MacCormack's rapid solver [16], and the hybrid MacCormack's scheme [17]. The mathematical reviews of these developments are presented by Lomax in references [18] and [7].

Practical computations involve numerical simulation of turbulence to provide more understanding of physical phenomena. Several basic algebraic, one-equation and two-equation models have been developed and used to analyze turbulent flows. A comprehensive review on turbulence modelling has been presented by Marvin in reference [19].

In the past decade computer speeds and computer memories have increased at a significant rate. The development of supercomputers, with memory measured in million words and calculation rate in mflops (million floating point operations per second), such as the ILLIAC-IV (16 million words, 25 mflops), CYBER-203 (1 million words, 20 mflops), CRAY-1 (1 million word, 30 mflops), CYBER-205 (4 million words, 80 mflops) and CRAY-1S (4 million words, 30 mflops) have dramatically increased the capabilities of CFD and reduced the cost of computation. For example, the CRAY-1S can perform 100,000 calculations for less than a penny. The CRAY-2, which is still in the design stage, will approximately double the speed of the CRAY-1S with possible further reduction in calculation cost. The supercomputers are used in areas such as aeronautical engineering, nuclear research, weather forecasting and other military and civilian applications. The CYBER-200 and CRAY-1 series computers are vector processors and use pipe-line architecture to increase the calculation rate. Levine presented an introduction to supercomputers and their architecture in reference [20], and technical information can be found in reference [21]. A computational algorithm developed with the supercomputer architecture in mind can effectively use its computational capabilities and hence reduce the run costs significantly.

The present study will be focused in the areas of algorithm development and the use of a supercomputer for the numerical solution of incompressible Navier-Stokes equations. The areas of grid generation and turbulence modelling will be addressed as essential elements required for simulation of the flow. The information about grid generation techniques and the turbulence model used in this study can be found in appropriate references. However they will be presented in detail for completeness.

1.1 Review of Previous Investigations

The unsteady and steady incompressible Navier-Stokes equations have been successfully used by many researchers to simulate the flow field of different characteristics. The basic formulations used in most work are velocity-vorticity, streamfunction-vorticity, streamfunction-biharmonics and primitive variables. Cebeci has reviewed the last three formulations in reference [22].

The viscous incompressible flow past an airfoil has been subjected to several numerical attacks in the past decade. The pioneering computations of laminar, incompressible two-dimensional flows about an airfoil are summarized in [23] and [24]. Hodge [25] used the optimized boundary conforming coordinate system for the laminar flow. In recent years, turbulent flows have been of increasing interest to researchers. Sugavanam [26] obtained the solution for flow past a Joukowski airfoil using velocity-vorticity formulation. Hegna [27] used primite variables for flow past a NACA 0012 airfoil. Bernard [28] employed the Approximate Factorization technique for NACA-66_3-0018 airfoil section. Moitra [29] simulated three dimensional turbulent flow around an airfoil. Lately, Freeman [30] used an adaptive grid approach for dynamic coupling of the grid and flow field solution. Experimental investigations have been reported by Mueller in reference [31].

1.2 Research Objectives and Outline

The objective of this effort is to develop a vectorized computer code for viscous turbulent, two-dimensional incompressible flow past an airfoil using an implicit finite-differencing scheme (Backward-

Time, Central-Space). On the pipeline computers, such as the CYBER-200 series, it is desirable to work with very long vectors for efficient use of its vector processing capabilities. Explicit methods are simple and can be easily vectorized since the entire grid can be considered as a long vector. However, the major disadvantage of explicit schemes lies in the time step restrictions imposed by stability considerations. Implicit schemes are frequently unconditionally stable and usually employ an iterative method such as SOR (Successive Over Relaxation). Since application of SOR on a vector machine results in either the inefficient use of its vector processing abilities or the necessity to shift to slower scalar operations, the checkerboard SOR algorithm (Chapter VII) will be used for the iterative solution of the governing equations on a vector processor. Also the effects of the algorithm, smoothers, grid, various forms of articifial viscosity and some boundary conditions on the solution will be investigated for the specific problem under study. The comparisons will be made with the available experimental results in Chapter VIII.

The present research effort is carried out in the following frame-work. The governing equations are the two-dimensional, incompressible Reynolds-averaged Navier-Stokes equations, written in non-conservative form in terms of primitive variables. A Poisson equation for the pressure is obtained by taking the divergence of the momentum equations. The two-layer algebraic turbulence model of Baldwin and Lomax [32] is used to calculate the eddy viscosity for the Reynolds-averaged equations. Dirichlet boundary conditions are imposed on the freestream boundary. On the downstream boundary, extra-poltation boundary conditions on the velocity and Dirichlet boundary condition on the pressure are imposed. The boundary conditions imposed on the airfoil surface are obtained employing no-slip conditions for the velocity and by setting the normal derivative of the pressure equal to zero on the boundaries. A linear gradual start is used to accelerate the flow from rest to its final freestream velocity. An implicit finite-differencing scheme, obtained using backward-time central-space approximations for the governing equations in the transformed plane is used to obtain flow field solutions. At each time step, the three governing equations are solved simultaneously, for u, v, and p.

CHAPTER II

THE BOUNDARY-CONFORMING CURVILINEAR COORDINATE SYSTEM

2.1 The Boundary-Conforming Concept

A coordinate system can have a significant influence on the numerical solution of hosted partial differential equations. For many cases of practical interest, the irregularities present in the boundary geometry will limit the use of the Cartesian coordinate system in finite difference flow field simulation. A cartesian coordinate system under such circumstances will require interpolation near the body boundary to implement the boundary conditions. In the boundary conforming coordinate system grid lines coincide with the body boundary thus yielding a degree of simplicity in the implementation of the boundary conditions. Also, the use of boundary conforming grids in the solution of partial differential equations in domains surrounding arbitrary geometrical boundary shapes will give a well-ordered system of algebraic difference equations compatible with the algorithms which can efficiently use the vector-processing computers. Various possible approaches such as conformal mapping, transfinite mapping, algebraic and elliptic equations have been successfully employed to generate body-conforming curvilinear coordinate systems. A comprehensive survey to these techniques and applications have been given by Thompson in references [11] and [13] and by Thompson, Warsi and Mastin in reference [12].

The feasible and systematic way of generating an appropriate body-conforming coordinate system should consider constraints that are needed for a given problem. Preferably, the grid should be generated in an automatic manner with the elements of control within the mesh generation process. Also the user should have an acceptable degree of control over grid smoothness, skewness and stretching. Hosted algorithms are usually sensitive to the grid smoothness, skewness and stretching and general reasons for this effect include the following: The coefficients of the transformed partial differential equations depend on the derivatives of the functions defining the coordinate system thus smoothness of the grid will have considerable effects on the accuracy of the solution. The local truncation error increases with departure from orthogonality. Also, the use of algebraic turbulence models demand near-orthogonality at the boundary for consistent modelling of turbulent flows. For a fixed number of grid points, the clustering of grid points in the region of large gradient should reduce the error and improve solutions. Moreover, some algorithms require a grid generating procedure that can be dynamically coupled to the physical solution properties to enhance accuracy and efficiency of the numerical results.

The grid generated using conformal mapping techniques have been used by several investigators. The main advantage is that it allows greater control by the user. The main disadvantage of this method is the lack of flexibility and automation. An elliptic system can generate a grid in an automatic manner with a moderate degree of control by the user. It can be extended to three-dimensions and a .ptive grid procedures.

2.2 Elliptic Systems

An elliptic grid generating system proposed by Thompson, Thames and Mastin [10] and successfully used by many investigators is capable of generating a suitable grid for the present study. The elliptic grid generating system is less susceptible to grid overlapping and can be subjected to a variety of grid control procedures to obtain desired grid characteristics as discussed in the previous section.

Numerical grid generation usually involves transformation of the physical domain of interest into a geometrically simple computational domain, such as a single rectangular domain. The solution of grid generation equations in the computational domain produces the corresponding grid in the physical domain. The physical space defined by Cartesian coordinates x and y is mapped onto the computational space through the mapping functions

$$\xi = \xi(x,y) \tag{2.1}$$

$$\eta = \eta(x,y) \tag{2.2}$$

by making the inner, outer, lower downstream and upper downstream boundaries coincide with $\eta = \eta_{min}$, $\eta = \eta_{max}$, $\xi = \xi_{min}$ and $\xi = \xi_{max}$, respectively. The extremum principle ensures occurence of extrema only on the boundaries and hence overlapping of grid lines can be avoided.

The topological correspondence in a C-type grid about a 2-D airfoil may be better understood with the help of Figure 1. The boundary $\eta = \eta_{min}$ is mapped onto the inner boundary $\Gamma_4 - \Gamma_1 - \Gamma_5$ containing the branch cuts, and the airfoil $\xi = \xi_{min}$ and $\xi = \xi_{max}$ correspond to the downstream section Γ_{3L} and Γ_{3u} respectively, ξ increasing clockwise around the airfoil. The η = constant family of lines form open curves resembling the letter C. The $\eta = \eta_{max}$ boundary is mapped on to the outer freestream boundary Γ_2.

The elliptic grid generation method of Thompson et.al. [10] permits any desired distribution of ξ and η on the boundaries. The inherent smoothness of solutions of elliptic systems is well recognized, and hence they are less adaptable to propagation of boundary slope discontinuities into the field. The choice of the elliptic system is further reinforced by the ability of the inhomogenous terms in Poisson's equation to control coordinate line spacing with respect to a curve or a point within the field. The chosen grid generating system has the following form:

$$\xi_{xx} + \xi_{yy} = \frac{\alpha}{J^2} P(\xi,\eta) \qquad (2.3)$$

$$\eta_{xx} + \eta_{yy} = \frac{\gamma}{J^2} Q(\xi,\eta) \qquad (2.4)$$

A desired form of the control functions P and Q makes it possible to concentrate lines in regions of the field. An interchange of dependent and independent variables enables one to perform all computation in the transformed field (Appendix A). The generating system in the transformed field becomes:

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -(\alpha x_{\xi} P + \gamma x_{\eta} Q) \qquad (2.5)$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -(\alpha y_{\xi} P + \gamma y_{\eta} Q) \qquad (2.6)$$

The transformed equations are solved in the rectangular $\xi\eta$- plane and the Dirichlet boundary conditions are specified for x and y by the known shape of boundaries. The coefficients of equations (2.3) - (2.6) are functions of the transformation and are defined by

$$\alpha = x_{\eta}^2 + y_{\eta}^2 \qquad (2.7)$$

$$\beta = x_{\xi} x_{\eta} + y_{\xi} y_{\eta} \qquad (2.8)$$

$$\gamma = x_{\xi}^2 + y_{\xi}^2 \qquad (2.9)$$

$$J = x_\xi y_\eta - x_\eta y_\xi \qquad (2.10)$$

where J is the Jacobian of the transformation.

A great deal of simplification in computation results if integer values are assigned to ξ and η and increments $\Delta\xi$ and $\Delta\eta$ are chosen to unity. This gives rise to uniform spacing in the transformed plane, with no loss of generality since these increments cancel from the equation anyway.

The generating system of equations (2.5 - 2.6) is represented by second-order central finite difference approximations in the transformed plane. The quantities $\Delta\xi$ and $\Delta\eta$ disappear by cancellation in all difference equations. Equations (2.5) and (2.6) are solved by the point successive over-relaxation (SOR), [33], scheme after control functions P and Q have been specified.

2.3 Control Functions

The inhomogenous terms (P & Q) in equations (2.3) and (2.4) can be automatically chosen to obtain control of spacing, orthogonality and stretching.

2.3.1 Thompson et.al. Approach

Thompson's approach consists of determining a correspondance between η values and the radius of concentric circles distributed between two circles with radius r_1 and r_2, one circumscribing the airfoil and the other tangential to the outer boundary respectively. Applying the coordinate generating equations (2.5 - 2.6) to the radii r_1 and r_2, while noting that $\eta = 1$ on the airfoil and $\eta = JL$ on the outer boundary, results in the following expression for Q:

$$Q(\xi,\eta) = -\left[\frac{r''(\eta)}{r'(\eta)} - \frac{r'(\eta)}{r(\eta)}\right] \qquad (2.11)$$

where $r(\eta)$ is a function of the hyperbolic tangent [34]. The effect of Q is to place a line corresponding to $\eta = k$ at a distance proportional to the $r_k - r_1$ from the body surface. Usually, to ensure proper resolution of the boundary layer the first line away from the boundary is placed at an approximate distance of one percent of the Blassius flat plate boundary layer thickness from the body, i.e.

$$r(\eta=2) - r_1 = 0.01\left(\frac{5}{\sqrt{Re}}\right) \qquad (2.12)$$

In general, for the above approach, the control functions are determined from specified line distribution and the control functions have direct control over line spacing in the field (fig. 2a).

2.3.2 Sorenson's Approach:

Sorenson [35] determines the inhomogenous terms P and Q to control the spacing between mesh points, along mesh lines intersection the boundaries and the angles with which mesh lines intersect the boundaries (fig. 2b). P and Q are defined in terms of four new variables. In particular, for $1 \leq \eta \leq \eta_{max}$ they are

$$P(\xi,\eta) = p(\xi)e^{-a(\eta-1)} + r(\xi)e^{-c(\eta_{max}-\eta)} \qquad (2.13)$$

$$Q(\xi,\eta) = q(\xi)e^{-b(\eta-1)} + s(\xi)e^{-d(\eta_{max}-\eta)} \qquad (2.14)$$

where a,b,c and d are positive constants and in particular a,b,c and d were set equal to 0.55. Note that terms p,q,r and s appearing in the above equations are functions of ξ only. The four more unknowns p,q,r and s introduced in equations (2.5) and (2.6) through equations (2.13) and (2.14) require four equations. At the inner boundary (= 1), the coefficient of r and s in equations (2.13) and (2.14) becomes very small and hence the wnd terms on the RHS of these equations can be dropped so that

$$P(\xi,1) = p(\xi) \qquad (2.15)$$

$$Q(\xi,1) = q(\xi) \qquad (2.16)$$

Similary at the outer boundary ($\eta = \eta_{max}$)

$$P(\xi, \eta_{max}) = r(\xi) \qquad (2.17)$$

$$Q(\xi, \eta_{max}) = s(\xi) \qquad (2.18)$$

Substituting equations (2.15) - 2.18) in equations (2.5 - 2.6) we obtain

$$p(\xi) = \left[\frac{R_1 y_\eta - R_2 x_\eta}{\alpha J}\right]_{\eta=1} \qquad (2.19)$$

$$q(\xi) = \left[\frac{-R_1 y_\xi + R_2 x_\xi}{\gamma J}\right]_{\eta=1} \qquad (2.20)$$

$$r(\xi) = \left[\frac{R_3 y_\eta - R_4 x_\eta}{\alpha J}\right]_{\eta=\eta_{max}} \qquad (2.21)$$

$$s(\xi) = \left[\frac{-R_3 y_\xi + R_4 x_\xi}{\gamma J}\right]_{\eta=\eta_{max}} \qquad (2.22)$$

where

$$R_1 = [-(\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta})]_{\eta=1} \tag{2.23}$$

$$R_2 = [-[\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta})]_{\eta=1} \tag{2.24}$$

$$R_3 = [-(\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta})]_{\eta=\eta_{max}} \tag{2.25}$$

$$R_4 = [-(\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta})]_{\eta=\eta_{max}} \tag{2.26}$$

Equations (2.19 - 2. 22) involve the derivatives at the inner and outer boundaries. At this point, if we assume that information about all these derivatives at the boundaries is readily available, we can compute the control functions P and Q using equations (2.13 - 2.14) for given values of ξ and η in the field.

The geometric constraints imposed by Sorenson will be used to define values of some derivatives at the inner and outer boundaries. The first requirement is that the spacing along ξ = constant lines between an inner boundary node at $\eta = 1$ and the corresponding next grid node at $\eta = 2$ is specified by the user. Let this desired spacing in the physical plane be denoted by $\Delta s|_{\eta=1}$ so that we have

$$\Delta s|_{\eta=1} = [(\Delta x)^2 + (\Delta y)^2]^{\frac{1}{2}}_{\eta=1} \tag{2.27}$$

in the limit Δx and Δy approach zero

$$ds|_{\eta=1} = [(dx)^2 + (dy)^2]^{\frac{1}{2}}_{\eta=1} \tag{2.28}$$

and transforming using the chain rule

$$ds|_{\eta=1} = [(x_\xi d\xi + x_\eta d\eta)^2 + (y_\xi d\xi + y_\eta d\eta)^2]^{\frac{1}{2}}_{\eta=1} \tag{2.29}$$

for small distance ds along ξ = constant

$$ds|_{\eta=1} = [(x_\eta^2 + y_\eta^2)^{1/2} d\eta]_{\eta=1} \tag{2.30}$$

or

$$s_\eta|_{\eta=1} = [x_\eta^2 + y_\eta^2]^{1/2}_{\eta=1} \tag{2.31}$$

The second requirement is that the angle θ of the intersection between the inner boundary and the ξ = constant line is specified by the user. By using the definition of the dot product

$$[\nabla\xi \cdot \nabla\eta]_{\eta=1} = [|\nabla\xi|\ |\nabla\eta|\ \cos\theta]_{\eta=1} \tag{2.32}$$

or

$$[\xi_x\eta_x + \xi_y\eta_y]_{\eta=1} = [(\xi_x^2 + \xi_y^2)^{1/2}(\eta_x^2 + \eta_y^2)^{1/2}\cos\theta]_{\eta=1} \tag{2.33}$$

Using relations given in Appendix A and equation (2.31) in (2.33) after some algebra we obtain

$$x_\eta\Big|_{\eta=1} = \left[\frac{s_\eta(-x_\xi\cos\theta - y_\xi\sin\theta)}{(x_\xi^2 + y_\xi^2)^{1/2}}\right]_{\eta=1} \tag{2.34}$$

$$y_\eta\Big|_{\eta=1} = \left[\frac{s_\eta(-y_\xi\cos\theta + x_\xi\sin\theta)}{(x_\xi^2 + y_\xi^2)^{1/2}}\right]_{\eta=1} \tag{2.35}$$

Thus, the values of derivatives x_η and y_η at the inner boundary can be fixed by the user. Similar expressions can be obtained to fix the values of derivatives x and y at the outer boundary.

After the values of x and y at the boundaries are specified, an iterative solution of grid generating equations (2.5 - 2.6) require computation of the forcing functions P and Q in the field, which in turn, require information about derivative x_ξ, x_η, $x_{\xi\xi}$, $x_{\xi\eta}$, $x_{\eta\eta}$, y_ξ, y_η, $y_{\xi\xi}$, $y_{\xi\eta}$ and $y_{\eta\eta}$ at the inner and outer boundaries. The desired values of spacing and angle at the boundaries, supplied as an input, will fix the values of x_η and y_η as discussed before. Also at the inner and outer boundaries η = constant, and hence values of x_ξ, y_ξ, $x_{\xi\xi}$ and $y_{\xi\xi}$ are fixed. Derivatives $x_{\xi\eta}$ and $y_{\xi\eta}$ can be computed by differencing x and y with respect to ξ and are fixed at all iteration levels. However computation of $x_{\eta\eta}$ and $y_{\eta\eta}$ at the inner and outer boundaries will require the use of one sided differencing schemes. These one sided finite difference approximations will require information about x and y at more than one point off the boundaries. As the values of x and y in the field will change with every iteration, the only derivatives that change with it are $x_{\eta\eta}$ and $y_{\eta\eta}$. Thus, at each iteration level, the control functions P and Q change through $x_{\eta\eta}$ and $y_{\eta\eta}$. The control over mesh spacing and angles in the field, introduced by equations (2.13) and (2.14), decays with the increase in values of $(\eta-1)$ and $(\eta_{max}-\eta)$. The four control functions a,b,c and d in these equations determine the rate of exponential decay. It is interesting to note here that Sorenson's method is not overspecified. Since control functions are to be determined we can specify additional boundary conditions. An iterative method such as SOR can be used to solve the system of governing equations.

In the present investigation grids generated using Thompson's and Sorenson's techniques were employed.

Chapter III

THE EQUATIONS OF MOTION

The governing equations for a high Reynolds number incompressible flow field are the conservation equations for momentum and mass known as the Navier-Stokes equations. The governing equations in the present study are the time-dependent, incompressible, two-dimensional, Reynolds-averaged Navier-Stokes equations formulated in terms of the primitive variables. The pressure equation solved is the Poisson equation, derived by taking the divergence of the momentum equations. The Eulerian method is usually employed in computational fluid dynamics. This method involves a fixed control volume that is specified relative to a given coordinate system. Properties of the fluid are then specified as functions of both space and time. The conservation equations are approached using this methodology.

3.1 Conservation of Momentum

For a given system, Newton's Second Law states that the rate of change of momentum is equal to the sum of the external forces acting on it. For an arbitrary material volume V, this law can be written as:

$$\iiint\limits_V \frac{\partial}{\partial t}(\rho u_i)dV + \iint\limits_S (\rho u_i)u_j n_j ds$$

$$= \iint\limits_S (\Sigma_{ij} - p\delta_{ij})n_j ds + \iiint\limits_V \rho g_i dV \qquad (3.1)$$

The index "i" denotes any of the three cartesian coordinate directions x_1, x_2, x_3, and the Einstein summation convention has been used for the index "j". The dimensional variables are:

ρ =	density
u_i =	velocity
s =	material surface
n_i =	unit vector, normal to s
Σ_{ij} =	shear stress tensor
p =	pressure
δ_{ij} =	Kroncker delta
g_i =	body-force acceleration

The divergence theorem transforms Eq. (3.1) to

$$\iiint_V \frac{\partial}{\partial t}(\rho u_i)dV + \iiint_V \frac{\partial}{\partial x_j}(\rho u_i u_j)dV$$

$$= \iiint_V \left(\frac{\partial}{\partial x_j}\Sigma_{ij} - \frac{\partial p}{\partial x_i} + \rho g_i\right)dV \tag{3.2}$$

Since this equation is valid for any arbitrary volume V; when the integrands are continuous, the equation is

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = \frac{\partial}{\partial x_j}\Sigma_{ij} - \frac{\partial p}{\partial x_i} + \rho g_i \tag{3.3}$$

For an incompressible flow density ρ is constant, so Eq. (3.3) becomes

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = \frac{1}{\rho}\left(\frac{\partial}{\partial x_j}\Sigma_{ij} - \frac{\partial p}{\partial x_i}\right) + g_i \tag{3.4}$$

and Stokes' hypothesis gives the shear stress as

$$\Sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.5}$$

where μ is the viscosity of the fluid.

Equations (3.4) and (3.5) are normalized by defining the following dimensionless quantities.

$$\hat{u}_i = u_i/U_\infty \tag{3.6}$$

$$\hat{x}_i = x_i/\ell \tag{3.7}$$

$$\hat{t} = tU_\infty/\ell \tag{3.8}$$

$$\hat{\mu} = \mu/\mu_\infty \tag{3.9}$$

$$\hat{p} = p/(\rho U_\infty^2) \tag{3.10}$$

$$\hat{g}_i = \frac{g_i \ell}{U_\infty^2} \tag{3.11}$$

The reference quantities are

U_∞ = freestream velocity

ℓ = characteristic length (airfoil chord for this case)

μ_∞ = freestream viscosity

Substituting equations (3.6) - (3.11) into equation (3.4) yields the normalized, time dependent, incompressible Navier-Stokes equations

$$\frac{\partial \hat{u}_i}{\partial \hat{t}} + \frac{\partial}{\partial \hat{x}_j}(\hat{u}_i \hat{u}_j) = -\frac{\partial \hat{p}}{\partial \hat{x}_i} + \frac{1}{Re}\frac{\partial}{\partial \hat{x}_j}\left[\hat{\mu}\left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} + \frac{\partial \hat{u}_j}{\partial \hat{x}_i}\right)\right] + g_i \tag{3.12}$$

where Re is Reynolds number given by

$$Re = \rho U_\infty \ell / \mu_\infty \qquad (3.13)$$

Usually, viscosity is taken as constant for incompressible, non-conducting flow. However, in this study it is retained as a variable to facilitate the implementation of an algebraic model for turbulence.

3.2 Conservation of Mass

For a given system in which matter is neither created or destroyed the law of mass conservation (continuity) can be written as

$$\iiint_V \frac{\partial \rho}{\partial t} dv + \iint_S \rho u_j n_j ds = 0 \qquad (3.14)$$

Applying the divergence theorem and eliminating the volume integrals as before, Eq. (3.14) reduces to

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j) = 0 \qquad (3.15)$$

which for incompressible fluids is

$$D \equiv \frac{\partial u_j}{\partial x_j} = 0 \qquad (3.16)$$

where D is the divergence of the velocity vector. Equation (3.16) remains unchanged by the introduction of non-dimensional variables.

3.3 The Pressure Equation

The incompressiblity constraint eliminates the equation of state, which relates pressure, density and temperature. Hence, the real difficulty in the calculation of the velocity field for incompressible flow lies in the unknown pressure field. The pressure gradient forms a part of the source term of the momentum Eq. (3.12). Yet there is no obvious equation for obtaining pressure. The pressure field is indirectly specified via the continuity equation. When the correct pressure field is substituted into the momentum equations, the resulting velocity field satisfies the continuity equation.

To obtain a Poisson equation for pressure a divergence operation is performed on the momentum Eq. (3.12).

$$\frac{\partial^2 p}{\partial x_i^2} = -\rho \frac{\partial^2 u_i}{\partial t \partial x_i} - \rho \frac{\partial^2 (u_i u_j)}{\partial x_i \partial x_j} + \frac{\partial^2 \Sigma_{ij}}{\partial x_i \partial x_j} + \rho \frac{\partial^2 g_i}{\partial t \partial x_i} \tag{3.17}$$

Substituting Eqs. (3.5) and (3.16) into (3.17) leads to

$$\nabla^2 p = -\rho \frac{\partial D}{\partial t} - \rho \frac{\partial u_j}{\partial x_i} \frac{\partial u_i}{\partial x_j} + \frac{\partial^2 \mu}{\partial x_i \partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + 2 \frac{\partial \mu}{\partial x_j} \nabla^2 u_j \tag{3.18}$$

where it is assumed that $D = 0$, but $\frac{\partial D}{\partial t} \neq 0$ and that $\frac{\partial g_i}{\partial x_i} = 0$, i.e., the body-force acceleration is applied uniformly to the entire field.

In deriving Eq. (3.18) D can be extracted and set equal to zero and thus $\frac{\partial D}{\partial t}$ will be zero; however, due to computer round-off error $\frac{\partial D}{\partial t}$ is expected to retain an appreciable value. Therefore the derivative

$\frac{\partial D}{\partial t}$ serves as a corrective term to adjust the pressure in an effort to satisfythe continuity equation, as suggested by Hirt and Harlow [36].

3.4 Normalized Governing Equations in Two Dimensions

From this point on, all variables used will be non-dimensional, and the circumflex (^) will be dropped from the notation. Carrying out the indicated summations and identifying u_1, u_2, x_1 and x_2 with u, v, x and y respectively yields the two-dimensional governing equations.

It should be noted the momentum Eq. (3.12) is written in conservative form. As shown by Roache[9], this conservative form allows the finite-difference equations to preserve the Gauss divergence property of the continuum equations. Also, the Rankine-Hugonoit shock relations were derived using the conservative form. Thus, shock jump conditions are automatically satisfied since the conservative variables are continuous across the shock and need no special treatment because of discontinuitites. Since the flow under investigation in this research contains no such discontinuities a further simplification can be obtained using the non-conservative form. The non-conservative form of Eq. (3.12) is obtained by expanding the convective derivatives and using the continuity equation (3.16).

$$(u^2)_x + (uv)_y = uu_x + vu_y \qquad (3.19)$$

$$(v^2)_y + (uv)_x = vv_y + uv_x \qquad (3.20)$$

Thus, in cartesian notation the governing equations in non-conservative form and two-dimensions are

$$u_t + uu_x + vu_y = -p_x + \frac{1}{Re}[\mu\nabla^2 u + 2\mu_x u_x + \mu_y(u_y + v_x)] + g_1 \quad (3.21)$$

$$v_t + uv_x + vv_y = -p_y + \frac{1}{Re}[\mu\nabla^2 v + 2\mu_y v_y + \mu_x(u_y + v_x)] + g_2 \quad (3.22)$$

$$\nabla^2 p = -D_t - (u_x^2 + 2u_y v_x + v_y^2) + \frac{2}{Re}[\mu_x \nabla^2 u + \mu_y \nabla^2 v + \mu_{xx} u_x + \mu_{xy}(u_y + v_x) + \mu_{yy} v_y] \quad (3.23)$$

3.5 The Transformed Equations

Equations (3.21), (3.22) and (3.23) in the physical xy plane are transformed into the ξη-plane using the definitions and relations given in Appendix A. The individual components of the transformed equations which are valid on a rectangular field (or a combination of rectangular fields) in the ξη-plane are

$$[u_t]_{x,y} = [u_t]_{\xi,\eta} - (u_x x_t + u_y y_t) = [u_t]_{\xi,\eta} \quad (3.24)$$

$$[v_t]_{x,y} = [v_t]_{\xi,\eta} - (v_x x_t + v_y v_t) = [v_t]_{\xi,\eta} \quad (3.25)$$

$$u_x = (y_\eta u_\xi - y_\xi u_\eta)/J \quad (3.26)$$

$$u_y = (x_\xi u_\eta - x_\eta u_\xi)/J \quad (3.27)$$

$$v_x = (y_\eta v_\xi - y_\xi v_\eta)/J \quad (3.28)$$

$$v_y = (x_\xi v_\eta - x_\eta v_\xi)/J \quad (3.29)$$

$$p_x = (y_\eta p_\xi - y_\xi p_\eta)/J \quad (3.30)$$

$$p_y = (x_\xi p_\eta - x_\eta p_\xi)/J \tag{3.31}$$

$$\nabla^2 u = (\alpha u_{\xi\xi} - 2\beta u_{\xi\eta} + \gamma u_{\eta\eta} + \sigma u_\eta + \tau u_\xi)/J^2 \tag{3.32}$$

$$\nabla^2 v = (\alpha v_{\xi\xi} - 2\beta v_{\xi\eta} + \gamma v_{\eta\eta} + \sigma v_\eta + \tau v_\xi)/J^2 \tag{3.33}$$

$$\mu_x = (y_\eta \mu_\xi - y_\xi \mu_\eta)/J \tag{3.34}$$

$$\mu_y = (x_\xi \mu_\eta - x_\eta \mu_\xi)/J \tag{3.35}$$

$$[g_1]_{x,y} = [g_1]_{\xi,\eta} - (g_{1_x} x_t + g_{1_y} y_t) = [g_1]_{\xi,\eta} \tag{3.36}$$

$$[g_2]_{x,y} = [g_2]_{\xi,\eta} - (g_{2_x} x_t + g_{2_y} y_t) = [g_2]_{\xi,\eta} \tag{3.37}$$

$$\nabla^2 p = (\alpha p_{\xi\xi} - 2\beta p_{\xi\eta} + \gamma p_{\eta\eta} + \sigma p_\eta + \tau p_\xi)/J^2 \tag{3.38}$$

$$[D_t]_{x,y} = [D_t]_{\xi,\eta} - (D_x x_t + D_y y_t) = [D_t]_{\xi,\eta} \tag{3.39}$$

$$\begin{aligned}\mu_{xx} = [&y_\eta^2 \mu_{\xi\xi} - 2y_\xi y_\eta \mu_{\xi\eta} + y_\xi^2 \mu_{\eta\eta} - (y_\eta^2 y_{\xi\xi} - 2y_\xi y_\eta y_{\xi\eta} \\ &+ y_\xi^2 y_{\eta\eta})\mu_y - (y_\eta^2 x_{\xi\xi} - 2y_\xi y_\eta x_{\xi\eta} + y_\xi^2 x_{\eta\eta})\mu_x]/J^2\end{aligned} \tag{3.40}$$

$$\begin{aligned}\mu_{xy} = \{&(x_\xi y_\eta + x_\eta y_\xi)\mu_{\xi\eta} - x_\xi y_\xi \mu_{\eta\eta} - x_\eta y_\eta \mu_{\xi\xi} \\ &+ [x_\eta y_\eta x_{\xi\xi} + x_\xi y_\xi x_{\eta\eta} - (x_\eta y_\xi + x_\xi y_\eta) x_{\xi\eta}]\mu_x \\ &+ [x_\eta y_\eta y_{\xi\xi} + x_\xi y_\xi y_{\eta\eta} - (x_\eta y_\xi + x_\xi y_\eta) y_{\xi\eta}]\mu_y\}/J^2\end{aligned} \tag{3.41}$$

$$\mu_{yy} = [x_\eta^2\mu_{\xi\xi} - 2x_\xi x_\eta\mu_{\xi\eta} + x_\xi^2\mu_{\eta\eta} - (x_\eta^2 y_{\xi\xi} - 2x_\xi x_\eta y_{\xi\eta}$$
$$+ x_\xi^2 y_{\eta\eta})\mu_y - (x_\eta^2 x_{\xi\xi} - 2x_\xi x_\eta x_{\xi\eta} + x_\xi^2 x_{\eta\eta})\mu_x]/J^2 \qquad (3.42)$$

where α, β, γ and J are defined in equations (2.7) - (2.10) and σ and τ are given by

$$\sigma = [-y_\xi(\alpha P x_\xi + \gamma Q x_\eta) + x_\xi(\alpha P y_\xi + \gamma Q y_\eta)]/J \qquad (3.43)$$

$$\tau = [y_\eta(\alpha P x_\xi + \gamma Q x_\eta) - x_\eta(\alpha P y_\xi + \gamma Q y_\eta)]/J \qquad (3.44)$$

The discretization of the transformed versions of equations (3.21), (3.22) and (3.23) and the numerical procedures used to obtain their solutions are discussed in Chapter V.

3.6 Turbulence Model

Since the flow fields of interest are turbulent, the solution of the Navier-Stokes equations must take into account the effects of the random fluctuations of the dependent variables inherent to turbulent flows. The turbulent nature characteristics of these flows can be accounted for in the numerical soltuion by a variety of eddy viscosity models ranging from locally dependent algebraic models to the more complex higher order closure models. A paper by Marvin [19] provides a comprehensive survey of turbulence models generally employed in computation of external aerodynamics flows of practical interests. To date no single turbulence model has emerged that can be applied to the variety of flows encountered in computational aerodynamics. Also, the use of higher closure models will not necessarily give more accurate solution. Therefore 't was decided to use the locally dependent eddy viscosity model. The turbulence model used in this research is an extension of the Cebeci-algebraic viscoisty model [37] as modified and reported by Baldwin-Lomax [32]. In this model distribution of vorticity is used to determine length scale which eliminates the somewhat uncertain process of finding the outer edge of the shear layer. The non-dimensional molecular coefficient of viscosity μ in the laminar Navier-Stokes equation is replaced by

$$\mu = 1 + \varepsilon \tag{3.45}$$

where ε is eddy viscosity. The boundary layer region on a body consists of two layers, the inner layer and outer layer. The inner layer of this model accounts primarily for the laminar sublayer adajcent to the wall, with the outer layer accounting for the remainder of the boundary layer region. In cartesian coordinates the expression for the modified inner model based on Prandtl's mixing length theory can be written as

$$(\varepsilon)_{\text{inner}} = \delta \ell^2 |\underset{\sim}{\omega}| \tag{3.46}$$

where ω is defined as vorticity

$$|\underset{\sim}{\omega}| = \left| \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \right| \tag{3.47}$$

The mixing length in this model is obtained from Van Driest's sublayer model, and is given as

$$\ell \equiv 0.4y[1 - e^{\frac{-y\sqrt{\rho_w \tau_w}}{26\mu_w}}] \tag{3.48}$$

where y is the normal distance from the wall.

The outer region eddy viscositymodel consists of a modified closure-type model defined by the equation

$$(\varepsilon)_{outer} = 0.0268\rho\, F_1 F_2(y) \qquad (3.49)$$

where $F_2(y)$ is the Klebanoff intermittency factor given by

$$F_2(y) = [1 + 5.5\,(\frac{0.3y}{y_{max}})^6]^{-1} \qquad (3.50)$$

and

$$F_1 = y_{max} F_{max} \qquad (3.51)$$

$$F(y) = y|\underset{\sim}{\omega}|\,[1 - e^{\frac{-y\sqrt{\rho_w \tau_w}}{26\mu_w}}] \qquad (3.52)$$

The quantity F_{max} is the maximum value of $F(y)$ that occurs in a profile and y_{max} is the value of y at which it occurs.

The eddy viscosity in the wake region is given by the equation (3.49) with F_1 and $F(y)$ defined as

$$F_1 = 0.25 y_{max} u^2_{dif}/F_{max} \qquad (3.53)$$

$$F(y) = y|\underset{\sim}{\omega}| \qquad (3.54)$$

where

$$u_{dif} = (\sqrt{u^2 + v^2})_{max} - (\sqrt{u^2 + v^2})_{min} \qquad (3.55)$$

For some cases under investigation, the boundary layer transition points were set by assuming that transition occurs at the minimum pressure points and for the other cases, the transition points were maximum airfoil thickness points on the airfoil surface. The above prescribed two-layer eddy-viscosity model was successfully used by Baldwin and Lomax [32] and other investigators to predict separated flows.

Chapter IV

BOUNDARY AND INITIAL CONDITIONS

Boundary and initial conditions must be defined in order to solve the governing partial differential equations of a given flow field. Since important features such as boundary layer arises from boundary conditions, these conditions must be carefully defined. The conservation equations for incompressible flow about an airfoil when formulated in terms of primitive variables require initial velocity and pressure distributions and either Neumann or Dirichlet boundary conditions for the velocity and pressure on the boundaries.

4.1 Initial Conditions

Since the governing equations contain time dependent terms, initial conditions must be specified for the solution to proceed. Initial values of velocities and pressure must be imposed over the field. The values of non-dimensional velocities and pressure were set to zero at a time $t = 0$. Once an initial case for the flowfield had numerically converged to a valid solution, each succeeding time step was initialized by using velocity and pressure distribution of the preceding time steps.

4.2 Free-stream and Downstream Boundary Conditions

Computationally, the free-stream boundaries are generally placed at a reasonable distance from a body such that uniform flow conditions remain undisturbed by the presence of the body. Velocities and pressure are completely specified at the free-stream boundary (Section Γ_2, in Fig. 1).

The flow is accelerated from zero to a desired final velocity using the body force terms g_1 and g_2 in equations (3.21) and (3.22). The values of the velocities on the free-stream boundary during the acceleration phase were determined in the following manner

$$u_\infty = \int_0^t g_1 dt \qquad 0 \leq t \leq 1 \tag{4.1}$$

$$v_\infty = \int_0^t g_2 dt \qquad 0 \leq t \leq 1 \tag{4.2}$$

$$P_\infty = 0 \tag{4.3}$$

where $g_1 = \cos\psi$ and $g_2 = \sin\psi$, where ψ is the angle of attack. For each time step of the acceleration phase, the velocities on the free-stream boundary are found using eqs. (4.1) - (4.2) and are held fixed for computation of solution for that time step. After the acceleration phase the free-stream velocities are

$$u_\infty = \cos\psi \tag{4.4}$$
$$v_\infty = \sin\psi$$

The body force terms g_1 and g_2 were set equal to zero after the acceleration phase.

The boundary (Γ_3) is placed a great distance downstream. For this case, no velocity gradient exist at the downstream boundary. The pressure at the downstream boundary was set equal to the free-stream pressure. These boundary conditions can be written as

$$u_x = 0 \tag{4.6}$$

$$v_x = 0 \tag{4.7}$$

$$p = 0 \tag{4.8}$$

Also the effects of different forms of downstream boundary conditions investigated will be presented in Chapter VII.

4.3 Body-Surface Boundary-Conditions

The airfoil surface is considered to be a no-slip and impermeable boundary. The no-slip and no-transpiration conditions at the airfoil surface can be written as

$$u = 0 \tag{4.9}$$

and

$$v = 0 \tag{4.10}$$

The pressure on the airfoil surface is unknown, but can be approximated using the normal pressure derivative in the following way. The momentum equations (3.21) and (3.22) are utilized to evaluate the normal derivative of the pressure. Due to no-slip no transpiration boundary condition at the surface, the transient and convective terms in the momentum equations drop out and we obtain

$$\frac{\partial p}{\partial n} = \underset{\sim}{n} \cdot \underset{\sim}{\nabla p} = \underset{\sim}{n} \cdot (\underset{\sim}{g} + \frac{1}{Re} \nabla^2 \underset{\sim}{u}) \tag{4.11}$$

where $\underset{\sim}{n}$ is a unit normal and $\underset{\sim}{g}$ is the body force vector.

Initial attempts to use the above pressure boundary condition led to computational divergence. The simplified version obtained by neglecting the viscous terms was used in the present study.

$$\underset{\sim}{n} \cdot \underset{\sim}{\nabla p} = \underset{\sim}{n} \cdot \underset{\sim}{g} \tag{4.12}$$

The presence of the body force vector influences the pressure boundary condition during the acceleration phase; however after the acceleration phase is over equation (4.12) reduces to the familiar form

$$\underset{\sim}{n} \cdot \underset{\sim}{\nabla p} = 0 \tag{4.13}$$

For the present case, the airfoil surface is represented by η = constant line. The direction normal $\underset{\sim}{n}$ can be given by $\underset{\sim}{\nabla\eta}$. Thus, equation (4.12) becomes

$$\underset{\sim}{\nabla\eta} \cdot \underset{\sim}{\nabla p} = \underset{\sim}{\nabla\eta} \cdot g \tag{4.14}$$

or

$$(\xi_x\eta_x + \xi_y\eta_y)P_\xi + (\eta_x^2 + \eta_y^2)P_\eta = \eta_x g_1 + \eta_y g_2 \tag{4.15}$$

Using Appendix A we obtain

$$P_\eta = \frac{1}{\gamma}\{\beta P_\xi + J(-y_\xi g_1 + x_\xi g_2)\} \tag{4.16}$$

The surface pressure can be evaluated using a one-sided finite-difference approximation for P_η. For the problem under consideration the boundary conditions at the airfoil surfaces are probably the most crucial.

4.4 Re-entrant Boundary Conditions

The re-entrant sections, Γ_4 and Γ_5 in figure 1 are not boundaries in the physical plane but represent points within the flow field. The branch cut is made between the trailing edge of the airfoil and the downstream boundary to eliminate discontinuity in the inner boundary in the transformed plane. The values of flow variables cannot be fixed at these boundaries but they should evolve as a part of the field solution. This insures the continuity in flow variables and their gradients across the cut.

4.5 Trailing-edge Boundary Conditions

In the transformed plane body surface is a continuous line; however, in the physical plane the trailing edge is a sharp point. The surface-normal vector $\underset{\sim}{\nabla\eta}$ is discontinuous at the trailing edge. This geometric discontinuity leads to unequal trailing edge pressure found using equation (4.16). The basic assumption that there be no unbalanced forces at the trailing edge would be violated. To avoid this problem, the trailing edge pressure was found by taking the average of the trailing edge pressure on the upper and lower airfoil surfaces (points (NWE,1) and (NWS,1) in Fig. 1). However it led to jump in the pressure at the trailing edge which is physically unrealistic phenomena. To obtain smooth pressure distribution at the trailing edge, the following extrapolates were found useful in the present study

$$P(NSW,1) = \tfrac{1}{2}(P(NWS + 1, 1) + P(NWE - 1, 1)) \tag{4.17}$$

$$P(NWE,1) = \tfrac{1}{2}(P(NWS + 1, 1) + P(NWE - 1, 1)) \tag{4.18}$$

Chapter V

SOLUTION ALGORITHM

5.1 Numerical Procedure

The governing equations are the two dimensional, time dependent Navier-Stokes equations in the non-conservative form. The Poisson equation for the pressure is obtained by taking the divergence of the momentum equations and utilizing the continuity equation. The two momentum equations and the Poisson pressure equation form a set of three governing equations for three flow field unknowns u, v and p.

These governing equations are solved in the transformed plane for each field node using a fully implicit finite-differencing algorithm. This implicit algorithm is obtained by means of backward-time and central-space differencing of derivatives in the transformed plane. The governing finite-difference equations in an implicit form are fully vectorized and solved simultaneously at each time step using a checkerboard matrix iterative technique (Chapter VI).

5.2 Finite-difference Approximations to Governing Equations

As discussed before in Section 3.5 the task of obtaining the transformed governing equations is straight forward and requires substitutions of the transformed expression for the derivatives (3.24 - 3.44) in the governing equations (3.21 - 3.23). The presentation of fully transformed governing equations has been avoided here for simplicity; however, this section will detail the specifc transformations that are pertinent to the final form of computational equations.

All spatial derivatives in the transformed equations are approximated by second-order-accurate central-difference expressions as follows:

$$\left.\frac{\partial f}{\partial \xi}\right|_{i,j} = \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta\xi} \qquad (5.1)$$

$$\left.\frac{\partial^2 f}{\partial \xi^2}\right|_{i,j} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta\xi^2} \qquad (5.2)$$

Similar finite-difference expressions are used to approximate η derivaitve. The second-order accurate expression for the cross-derivative is

$$\frac{\partial f}{\partial\xi\partial\eta} = \frac{f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1}}{4\Delta\xi\Delta\eta} \qquad (5.3)$$

The grid spacing $\Delta\xi$ and $\Delta\eta$ is chosen to be unity because of the construction of the mapping from the physical plane to the transformed plane.

As presented in section 4.2, the flow is accelerated from rest to the final desired free stream velocity. Hence, the temporal derivatives are represented by the first-order-accurate two-point backward-difference scheme at the first time step and by second-order accurate three-point backward-difference at all subsequent time steps. The expression for two-point backward difference is

$$\left.\frac{\partial f}{\partial t}\right|_{i,j}^{(n)} = \frac{f_{i,j}^{(n)} - f_{i,j}^{(n-1)}}{\Delta t} \qquad (5.4)$$

and for the three-point backward difference is

$$\left.\frac{\partial f}{\partial t}\right|_{i,j}^{(n)} = \frac{3f_{i,j}^{(n)} - 4f_{i,j}^{(n-1)} + f_{i,j}^{(n-2)}}{2\Delta t} \qquad (5.5)$$

where the superscript (n) indicates the time level.

To obtain the computational form of the governing difference equations for an iterative scheme, we must combine the diagonal terms (those with subscript i,j) of spatial derivatives with an appropriate temporal derivative term. As the central-difference approximations have been used for the spatial derivatives, the terms with subscript i,j will appear only due to tranformation of $\nabla^2(\ )$ in equations (3.21 - 3.23). For completeness, we transform $\nabla^2 u$ in the u momentum. From equation (3.32) or Appendix A we have

$$\nabla^2 u = \frac{1}{J^2}(\alpha u_{\xi\xi} - 2\beta u_{\xi\eta} + \gamma u_{\eta\eta} + \sigma u_{\eta} + \tau u_{\xi})$$

The finite-difference approximation of derivatives $u_{\xi\xi}$ and $u_{\eta\eta}$ will involve the diagonal terms and approximation of all other derivatives ill involve off-diagonal terms. Separation of diagonal and off diagonal terms gives

$$\nabla^2 u = (\nabla^2 u)_D + (\nabla^2 u)_{OD} \tag{5.6}$$

where

$$(\nabla^2 u)_D = -\frac{2}{J^2}(\alpha + \gamma)u_{i,j} \tag{5.7}$$

$$(\nabla^2 u)_{OD} = \frac{1}{J^2}[\alpha(u_{i+1,j} + u_{i-1,j}) + \gamma(u_{i,j+1} + u_{i,j-1})$$

$$-2\beta u_{\xi\eta} + \sigma u_{\eta} + \tau u_{\xi}] \tag{5.8}$$

Note that in the remainder of this section terms such as u_x, v_y, etc. will appear but are to be implicitly assumed to have been evaluated according to equations (3.24 - 3.44) or relations in Appendix A. Substituting equation (5.6) in the u momentum equation (3.21) and combining the diagonal term with the temperal term at time level n, we obtain the

ORIGINAL PAGE IS
OF POOR QUALITY

computational form of the difference equation.

$$\left[\frac{A}{\Delta t} + \frac{2\mu}{J^2 Re}(\alpha + \gamma)\right]u_{i,j}^{(n)} = B + (RHS)_1 \qquad (5.9)$$

where

$$(RHS)_1 = -uu_x - vu_y - p_x + \frac{1}{Re}[\mu(\nabla^2 u)_{OD}$$

$$+ 2\mu_x u_x + \mu_y(u_y + v_x)] + g_1 \qquad (5.10)$$

and

$$A = 1 \qquad (5.11)$$

$$B = \frac{u_{i,j}^{(n-1)}}{\Delta t} \qquad (5.12)$$

for two-point backward differencing, or

$$A = \frac{3}{2}$$

$$B = \frac{4u_{i,j}^{(n-1)} - u_{i,j}^{(n-2)}}{2\Delta t} \qquad (5.14)$$

for three-point backward differencing. The computational difference equation for the v momentum, derived in similar fashion is

$$\left[\frac{A}{\Delta t} + \frac{2\mu}{J^2 Re}(\alpha + \gamma)\right]v_{i,j}^{(n)} = C + (RHS)_2 \qquad (5.15)$$

where

$$(RHS)_2 = -uv_x - vv_y - p_y + \frac{1}{Re}[\mu(\nabla^2 v)_{OD}$$

$$+ 2\mu_y v_y + \mu_x(u_y + v_x)] + g_2 \qquad (5.16)$$

and

$$C = \frac{v_{i,j}^{(n-1)}}{\Delta t} \text{ for two-point backward differencing} \qquad (5.17)$$

or

$$C = \frac{4v_{i,j}^{(n-1)} - v_{i,j}^{(n-2)}}{2\Delta t} \text{ for three-point backward differencing} \qquad (5.18)$$

The computational form of the Poisson pressure equation can be obtained in a similar manner from equation (3.23). The term D_t represents the time derivative of the divergence of the velocity vector. It is assumed that the conservation of mass is satisfied at the most recent time level (i.e., $D^n = 0$); however values of the divergence at previous time levels have been retained as a corrective term. Thus

$$D_t = -\frac{D^{(n-1)}}{\Delta t} \text{ for two-point backward differencing} \tag{5.19}$$

$$D_t = \frac{-4D^{(n-1)} + D^{(n-2)}}{2\Delta t} \text{ for three-point backward differencing} \tag{5.20}$$

and the computational pressure equation takes the following form

$$\begin{aligned} \frac{2(\alpha + \gamma)p}{J^2} = (\nabla^2 p)_{OD} + D_t + u_x^2 + 2u_y v_x \\ + v_y^2 - \frac{2}{Re}[\mu_x \nabla^2 u + \mu_y \nabla^2 v \\ + \mu_{xx} u_x + \mu_{xy}(u_y + v_x) + \mu_{yy} v_y] \end{aligned} \tag{5.21}$$

where D_t can be approximated using either equation (5.19) or equation (5.20). The first approximation is first order accurate while the second approximation is second order accurate. Several computer runs were made with the two approximations for comparison. No significant difference were found between results obtained using the first order and second order accurate approximations. Also, for particular test runs, none of these two approximations was specifically responsible for decay or divergence of solution. The above tests were not entirely conclusive.

The goal is to find the steady state solution regardless of accuracy of the transient solution. Since the time derivative terms will hopefully disappear in the steady state and higher order approximations usually require more operations per mesh point, we used first-order two-point approximation for D_t in the pressure equation (5.21).

5.3 Finite-difference Approximations to Boundary Conditions

The downstream boundary condition equations (4.6 - 4.7) are transformed according to the relations in Appendix A. Equation (4.6) for the lower downstream boundary ($\xi=1$) takes the following form

$$u_x = \frac{1}{J}\{y_\eta u_\xi - y_\xi u_\eta\} = 0 \qquad (5.22)$$

$$u_\xi = (\frac{y_\xi}{y_\eta})u_\eta \qquad (5.23)$$

Using one sided three point forward differencing for u_ξ

$$u_{1,j} = \frac{1}{3}[4u_{2,j} - u_{3,j} - 2(\frac{y_\xi}{y_\eta})u_\eta] \qquad (5.24)$$

Similarly fc the upper-downstream boundary condition (ξ = IL) using three point backward differencing for u_ξ we obtain

$$u_{IL,j} = \frac{1}{3}[4u_{IL-1,j} - u_{IL-2,j} + 2(\frac{y_\xi}{y_\eta})u_\eta] \qquad (5.25)$$

In the above equations, derivative u_η is evaluated using central-difference approximations. Replacing velocity u by velocity v in equations (5.24) and (5.25) we can obtain expressions for equation (4.7).

Pressure values on the airfoil surface are determined using the Neumann boundary condition (4.16). Using a three point forward-difference approximation for p_η the expression for airfoil pressure is

$$P_{i,1} = \frac{1}{3} \{4P_{i,2} - P_{i,3} - \frac{2}{\gamma} [\beta P_\xi + J(g_2 x_\xi - g_1 y_\xi)]\} \quad (5.26)$$

5.4 The Re-entrant Boundary

The procedure for the evaluation of flow field variables (u, v and p) on the cut extending from the outflow section (fig. 1) deserves special attention. The two re-entrant sections Γ_4 and Γ_5 resulting from the cut are one and the same line in the physical plane. Thus corresponding points on the two re-entrant sections have the same x,y coordinates in the physical plane but different ξ values. The momentum and pressure equations on the re-entrant section can be solved assuming continuous derivatives across the cut. However, in this study, flow variables on the re-entrant section were found by averaging the corresponding values above and below the branch cut. As the grid spacing required at the branch cut to resolve the flow is very small in a C-type grid, the averaging gives almost the same values of the flow variables as those found solving the governing equations at the re-entrant section. An expression obtained using notations of fig. 1 for two nodes in the computational plane that correspond to the first node off the trailing edge in the physical plane is

$$f_{NWS-1,1} = f_{NWE+1,1} = \tfrac{1}{2}[f_{NWE+1,2} + f_{NWS-1,2}] \quad (5.27)$$

Similar relations were used to find the values of the velocities and pressure at nodes on the branch cut. This approach simplifies computation of flow variables at the re-entrant section without sacrificing accuracy and hopefully enhances the computational efficiency due to less involved operations.

5.5 Artificial Viscosity

Central-differencing schemes frequently display oscillations on a coarse grid. The present implicit scheme exhibits oscillatory behavior at high Reynolds number due to inaccuracies introduced by finite-differencing. Unless these extreme oscillations are damped out the numerical solution becomes useless. In most cases under investigation, the solution started diverging about time $t = 1.0$ without inclusion of artificial viscosity. The use of artificial diffusion was found necessary to obtain steady state solution. The pressure oscillations were responsible for fluctuations and discontinuities in the velocity field. One possible source of the pressure oscillations was the divergence of the velocity vector which is a part of the source term of Poisson pressure equation (3.23) and may have retained significant magnitude. The basic assumption to obtain the Poisson pressure equation was the preservation of the continuity at the most recent time level. Thus, significant deviation from the satisfaction of the continuity equation can contaminate the pressure field. Incorporation of artificial viscosity based on the divergence of velocity at the current time level can damp extreme oscillations. The modified non-dimensional viscosity coefficient in the momentum equations (3.21 - 3.22) is given by

$$\mu = 1 + \varepsilon + \mu_a \qquad (5.28)$$

where ε is eddy viscosity term discussed in section 2.6 and μ_a is artificial viscosity. One possible form of the artificial viscosity is

$$\mu_a = \Omega \mathrm{Re} J |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}| \qquad (5.29)$$

Note that this form of artificial viscosity has units of eddy viscosity and it has the advantage of being analytically zero. The term Ω can limit the effects of artificial viscosity and will be a constant or a variable derived from the flow characteristic. This particular form of artificial viscosity has a desired property of being proportional to $|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ and only becoming effective in regions where the divergence of velocity is significant. The fluid dynamics phenomena investigated with various form of artificial viscosity will be discussed in Chapter VII.

Strictly speaking, the computation of viscous derivatives for the momentum equations (3.21 - 3.22) and the pressure equation (3.23) should use the modified viscosity coefficient (eq. 5.28). However this approach lead to divergence of the solution and hence the viscous derivatives for the governing equations were computed using the viscosity coefficent given by equation (3.45). The artificial viscosity μ_a was incorporated in the viscosity coefficient μ of the momentum equations at every time step.

5.6 Smoothers

At early time stages, the solution may contain enough noise to excite oscillations. Nonlinear interaction will amplify these oscillations which in turn may destroy the solution. In such cases, we wish to filter out the unwanted oscillations from the solution. In most test runs, wavy divergence of velocity field was obtained in the direction of ξ = constant lines. If somehow, a smooth divergence of velocity field can be obtained, it can reduce the pressure oscillations. Two types of smoothers, one using the adjacent nodes in ξ = constant direction and the other using the four neighboring nodes were investigated. The latter gave better overall smoothing of divergence of velocity field. The expression for the smoother is

$$f_{i,j} = \frac{1}{2}[f_{i,j} + \frac{1}{4}(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1})] \quad (5.30)$$

On the other hand, pressure exhibited excessive oscillatory behavior in the direction of η = constant lines. Smoothing of the pressure itself, using equation (5.30), lead to incorrect pressure solution, but smoothing of the source term of the Poisson pressure equation (3.23) can smooth out the pressure field oscillations. Denoting the source term by S, with the assumption that all terms on the right hand side of equation (3.23) are lumped into S, we can smooth out the source term by using S instead of f in equation (5.30). The results obtained using divergence of velocity and source term smoothers were almost the same. As computer operations for the source term smoother are more involved and smoothing operation is usually required at every iteration it was not investigated further in the light of computational efficiency.

The divergence of velocity smoother, applied at every time-step, was employed to reduce the need of artificial viscosity.

Chapter VI

VECTOR PROCESSORS AND CHECKERBOARD SOR

In the last decade, significant progress has been made in the area of algorithms that are used for solving the governing flow field equations. Computer codes employed in many engineering applications still use large amounts of computer resources. The basic requirement is that the algorithm be efficient. In practical terms this means obtaining the solution with desired accuracy using the least amount of computer resources. Any improvement in the numerical scheme used can certainly enhance the efficiency. Some improved algorithms have been mentioned in Chapter 1 with appropriate references. Furthermore, in many cases the computers available play an important role in the development of efficient algorithms. Hence the computer achitecture such as serial, vector or parallel certainly dictate the basic requirement of algorithms. Frequently, the structure and size of computer memory and data mangement system can play a crucial role in the implementation of efficient algorithms.

6.1 Vector Processors

The advent of high performance sixth generation computers such as the CYBER-200 and CRAY-1 series, provides an important breakthrough for computationally demanding engineering problems. These supercomputers incorporate vector processing capabilities to provide the computational power required by large scale numerical simulation [21].

Vector processors are generally divided into two main classes of architecture: memory to memory (MM) and register-to-register (RR). Normally MM architecture vector processors operate at its highest level of performance when algorithms being processed have the following characteristics. Operand and result vectors are stored contiguously in memory i.e., successive elements of the vector must be stored in adjacent memory locations. The length of the vector is long. The example of MM architecture is the CYBER-200 series machines. RR achitecture usually involves some type of cache between main memory and functional units. The fundamental idea of cache organization is that by keeping most frequently accessed instructions and data in the fast cache memory, the cache is only a small fraction of the size of main memory. If the active portions of the program and data are placed in a fast cache memory, the average memory access time can be reduced considerably, thus reducing the total execution time of the program. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components. These types of vector processors usually achieve their highest level of performance when processing algorithms that satisfy the following requirements. Parallel execution of the functional units is maximized. The example of RR architecture is the CRAY-1 series machines.

The above-described two types of vector processors are called pipeline processors. Pipeline is a technique of decomposing a sequential process into subprocesses with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments. A pipeline can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. The name "pipeline" implies a flow of information analogous to an industrial assembly line. It is characteristic of pipelines that several computations can be in progress in distinct segments at the same time.

The CRAY-1 series are RR type pipeline machines which operate most efficiently on vectors which are of length 64 or a multiple of 64. The reason is that the vector registers hold 64 words which are sent to the pipeline. Thus in the CRAY-1 series machines the vector registers are limited to 64 elements and hence extremely long vector lengths will not necessarily enhance the computational efficiency. The CRAY-1 memory section normally consists of 16 banks of memory. The memory size can be as large as about 1 million words. Each word contains 44 data bits and 8 check bits. The control of data flow between the parallel functional units and hierarchically organized memories is of significant importance for algorithm efficiency.

The CYBER-200 series are MM type pipeline machines which operate more efficiently as the vector length increases. Each vector instruction involves a startup time, the time required to produce first result. Since startup time becomes relatively less important as the vector length increases, the vector operations become more efficient. Thus it is desirable to work with moderate to long vectors on the CYBER-200 series machines. The CYBER-203 has about 1 million words of primary memory with virtual memory architecture. Memory on this machine is called as pages, which are of small and large size. The small page is made up of 521 words of 64 data bits and the large pages are of 65,536 words. A user can have access to about seven large pages in primary memory at a time. The movement of data from secondary memory into primary memory involves moving of pages. This movement of pages in in and out of primary memory is called page fault and involves startup time and transmissio time. It is desirable to make most efficient use of data when it is in primary memory to avoid situations when the machine time spent on data management makes up a considerable part of the total time.

Thus performance on a vector processor can vary widely as a function of algorithm, implementation and data management.

6.2 Checkerboard SOR

As the computers discussed in the above section attain their highest level of performance when processing vectors, it is clearly desirable to search for methods that can take advantage of the vector operation capabilities without suffering significant loss in convergence rate compared to widely accepted methods for serial computers.

Generally the choice of an appropriate algorithm is dictated by whether the flow is subsonic, transonic or supersonic. Although it is the steady state solution that is generally sought one often uses the time dependent equations to reach steady state. An explicit algorithm which can be easily vectorized may have much slower convergence rate. With explicit methods entire two or three-dimensional grids can be considered as one long vector. On some machines this will lead to a high level of optimization. The solution of the three dimensional compressible Navier-Stokes equations obtained using vector processors have been published by several investigators. Smith et.al. [38] and Shang et. al. [39] solved these equations using an explicit scheme on the CDC STAR-100 and CRAY-1 computers respectively. For the 3-D problems solved using an explicit scheme, the vector lengths were restricted to the number of grid points in each 2-D plane due to efficient use of computer architecture in Shang's investigation and due to efficient data management for memory in Smith's study. Spradley et. al. [40] solved these 3-D equations using general interpolants methods (GIM) on the CDC STAR-100. He chose weight functions such that to produce explicit finite-difference type analog and used the vector lengths equal to the total number of grid points in a 3-D flow field.

Although long vectors available at each time step for explicit schemes may increase efficiency of some vector processors, the large number of time steps required to reach the steady state may adversely affect the overall performance of the algorithm. Furthermore, in many cases, one is only interested in obtaining the steady state solution as fast as possible without regard to the accuracy of the transient solution. The time step restriction imposed by stability consideration is a major disadvantage of explicit schemes. Hence there is increased interest in implicit schemes in recent years. Also, for implicit schemes, one frequently uses the time dependent equations, and fairly accurate steady state solution is reached with larger time steps. Although implicit methods are usually linearly unconditionally stable, however there exists time step restriction based on accuracy requirements. Also operations for an iterative relaxation procedures are more involved. The development of an efficient relaxation method is an important element for implicit algorithms.

The most widely used classical relaxation methods are generally not suitable for vector computers. The point accelerated successive relaxation (SOR) method [33], which is perhaps most frequently used, is reliable and very competitive for many problems. The convergence rate of point

SOR depends partly upon using updated values at adjacent points while solving for a given point.

Point SOR schemes can be efficiently implemented on a scalar machine. However, for vector processors, vectors must be stored and must be available for concurrent computer functions required for desired arithmetic operations. This requirement is very restrictive and the classical point SOR method is not suitable for vector processing in its original form. There are some possible ways of system ordering for solution of PDE using a rectangular grid on a vector machine. Sufficiently large vectors can be identified within the field or subfield by (a) associating vectors with alternate rows or columns (ZEBRA) or (b) associating vectors with alternate field points (red - black). Option (b) is a simple way of making point SOR suitable for vector processing. This modified SOR is usually referred to as checkerboard SOR or hopscotch method in case of parabolic problems.

Early work related to the hopscotch method was presented by Gordon [41] in 1965, many years before vector processors became available. His work was motivated by the favorable stability properties of the method. Gordon[41] described the original technique as "A non-symmetric difference equation" obtained using explicit and implicit finite difference schemes at alternate mesh points and showed that combined scheme was unconditionally stable. Scala et. al. [42] applied this technique to solve the Navier-Stokes equations about a circular cylinder using a cylindrical coordinate system on a serial computer. Gourlay, et. al., [43, 44] presented the original technique in a more general form and showed tnat the checkerboard method can be regarded as an Alternating

Direction Implicit (ADI) method with the coefficient matrix split in a special way. The fundamental idea of an ADI scheme is of splitting the problem into a series of simpler problems. Normally, each simpler problem corresponds to each space dimension and in many space dimension problems complexity increases considerably. The major advantage of the checkerboard algorithm is that it can be always decomposed into two simpler problems (two stage process) irrespective of the number of space dimensions.

For illustrative purpose, it is convenient to consider a simple model problem. Some detail for solving the Poisson equation using checkerboard-SOR will be presented. Let us consider the Poisson equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = S(x,y) \tag{6.1}$$

with simple Dirichlet boundary condition on the boundary and with the field subdivided in square cells to length h as shown in fig. 3. Using central finite-difference approximations at a mesh point (x_i, y_i) equation (6.1) can be written as

$$4f_{i,j} - f_{i+1,j} - f_{i-1,j} - f_{i,j+1} - f_{i,j-1} = -h^2 S_{i,j} \tag{6.2}$$

Instead of considering natural ordering of mesh point, i.e. sweeping rowwise, let us visualize the field mesh points as forming a red-black chess board. This red-black ordering can be defined as follows: Cell field mesh point (i,j) red if (i+j) is odd and point (i,j) black if (i+j) is even. Hence the red unknowns will be the set of all $f_{i,j}$ for which (i+j) is odd and similarly for the black unknowns when (i+j) is even. Applying the classical SOR to the red and black unknowns it can be shown that each classical SOR iteration can be split into two stages. The first or red stage consists of improving the red unknowns according to

$$f_{i,j}^{(n+1,r)} = \frac{\kappa}{4}(-h^2 S_{i,j}^{(n,r)} + f_{i+1,j}^{(n,b)} + f_{i-1,j}^{(n,b)} + f_{i,j+1}^{(n,b)} + f_{i,j-1}^{(n,b)})$$

$$+ (1 - \kappa) f_{i,j}^{(n,r)} \qquad (6.3)$$

and during the following second or black stage the black unknowns are improved according to

$$f_{i,j}^{(n+1,b)} = \frac{\kappa}{4}(-h^2 S_{i,j}^{(n,b)} + f_{i+1,j}^{(n+1,r)} + f_{i-1,j}^{(n+1,r)} + f_{i,j+1}^{(n+1,r)} + f_{i,j-1}^{(n+1,r)})$$

$$+ (1 - \kappa) f_{i,j}^{(n,b)} \qquad (6.4)$$

In equations (6.3) and (6.4) κ is a relaxation parameter used to accelerate convergence, superscripts n,r and b denote iteration level, red and black nodes respectively. During the red stage all red iterates are updated with the help of the adjacent black iterates and conversely in this particular case. Each state is inherently parallel in that all iterates of the same color can be updated simultaneously without changing those of the other color. Each term on the RHS of equations (6.3) and (6.4) can be represented as a vector, assuming scalar terms such as h, is a vector of desired length with each element of the vector having the same value. Let us assume κ to be constant for the present case. Thus equations (6.3) and (6.4) can take advantage of vector processing capabilities of a supercomputer. Each iteration is made up of two stages and the red and black states succeed one another, however the black stage must not start until the preceding red stage has completed and conversely.

For the Dirichlet boundary conditions, if any term on the RHS in equation (6.3) and (6.4) belongs to the boundary, then the corresponding term is understood to have the prescribed boundary value. In case of the Newmann boundary condition, redefinition of the boundary data can be easily incorporated after each stage or two stages depending upon the number of grid points on the boundary, type of vector processor or tradeoff between the scalar and vector operations.

It is worth noting that two-step Jacobi, which can use vector length equal to total number of nodes in the field, is also an attractive method for vector processor in which vector length is an important factor in the calculation rate and vector processor performance is at least twice its scalar performance. In many applications, the Jacobi method with acceleration parameter $\kappa = 1$ may not be able to compete with the checkerboard SOR method. Frequently, cyclic change of red and black stages may give better convergence rate for the hopscotch method.

There also exists a family of hopscotch methods such as line or zebra-like [44] and block methods [45, 46]. Some properties of hopscotch methods have been presented by Gourlay et. al. [47]. Greenberg [48] employed the approximate factorization scheme of Beam and Warming [15] in hopscotch form and other hopscotch methods to investigate fluid dynamics problems on a serial computer. South et. al. [49] used Checkerboard SOR, Zebra SOR and Checkerboard Leapfrog method for transonic flow calculations on the CDC-STAR-100 and CRAY-1 vector computers.

6.3 Implementation on the CYBER-203

As the checkerboard SOR is the heart of an implicit method used in the present study, it was decided first to implement the model problem on the CYBER-203 and then to employ the same basic features of the implementation for solving the Navier-Stokes equations. One possible way of approaching the model problem using the CYBER-200 FORTRAN language will be discussed in this section.

A close examination of the test problem indicates that the first task is to determine vectors of the red and black field variables from the arrays containing all field and boundary nodes of the same variables. Once the vectors of desired color are obtained, the arithmatic operations on these vectors are rather simple and can be performed using explicit vector instructions. An emphasis is made on the use of predefined vector functions and rich instruction set of the CYBER-200 FORTRAN compiler. The bit addressable memory, which allows the use of bit vectors is one of the important characterisitics of this machine.

The total number of elements in any array equals to the product of its dimension sizes. All elements of the array are stored contiguously in a memory. To find the location of an array element for a given array T(A,B) of a particular instance of subscript T(a,b), the formula a + A * (b-1) can be used. Thus an array can be thought of as a vector, and wherever required we will use word vector to represent an array in the remainder of this section. Each element of a bit vector requires storage of one bit in contrast to 64 bits required for each element of a single precision value vector (real or integer) on this machine. An element of a bit vector can take a value of either 1 or 0 representing logical operator truth and false respectively. All logical operations such as AND, NOT, etc. can be performed on the bit vectors. The logical operations are performed on either corresponding elements of two bit vector operands or a scalar operand paired with successive elements of a vector operand. This important feature allows us to generate a bit vector of desired structure or pattern which in turn can be used as a control vector in some very efficient built in functions. Also there are some functions which help to form an initial bit vector of some desired 0-1 pattern.

Some useful functions, which use bit vectors as control vectors will be briefly described, since they are an important part of the present implementation. Details of the builtin functions for the CYBER-200 FORTRAN compiler can be found in reference [50]. Bit vectors can be used as a control vector to select elements from a value vector. The CMPRS function deletes selected elements from a real or integer vector as dictated by a bit control vector. The MERG function merges the elements in two value vectors into a result vector under control of a bit vector. The function CTRL changes the values of selected elements in a result vector using the values in an argument vector under the control of a bit control vector.

For illustration, let us consider the model problem. As we would like to solve equation (6.3) in vector form, we must have vectors of all terms on the RHS of equation(6.3). At the beginning of the first or red stage array f(5,5) of all nodes (including field and boundary) is available. The task is now to obtain vectors of all terms involving f and S on the RHS of the equation. The procedure involves selecting and assembling all $f_{i,j}^{(n,r)}$ at the red field nodes from vector f of the field and boundary nodes. Similarly we must form vectors for $f_{i+1,j}^{(n,b)}$, $f_{i,j-1}^{(n,b)}$, $f_{i,j+1}^{(n,b)}$, $f_{i,j-1}^{(n,b)}$, which are located at the black nodes. Let us assume that all control bit vectors of desired pattern are stored in the memory and are available to facilitate the use of previously described functions. As mentioned before, these bit vectors can be easily formed using some buit-in functions and logical operation such as AND, NOT, OR, etc.

As shown in fig. 4, execution of function CMPRS will give us the vector for $f_{i,j}^{(n,r)}$ terms. Calling the CMPRS function with an appropriate control bit vector will give the vector for $f_{i+1,j}^{(n,b)}$ (fig. 5). Similarly we can obtain vectors for the remaining f terms using appropriate bit vectors. To obtain a vector for the source term $S_{i,j}^{(n,r)}$, we can use the same bit vector as in fig. 4, however the argument value vector will be S(5,5) instead of f(5,5). This completes the formation of all required vectors for solving equation (6.3) The equation involves scalar terms $\frac{\kappa}{4}$, $(1 - \kappa)$ and h^2. These scalar terms are assumed to be implicitly expanded to the necessary vector length, with each element having the same scalar value. The arithmetic operations involved are straight-forward and equation (6.3) can be solved for $f_{i,j}^{(n+1,r)}$ on the vector processor using explicit vector notations.

Before we go to the second or black stage (eq. 6.4) we must update array f(5,5) using vector $f_{i,j}^{(n+1,r)}$. One possible way to replace the old values of variable f at the red field nodes with the updated values, is to use the CTRL function. For the CTRL function, vector lengths of result, control bit and argument vector should be the same. The result vector in this case will be vector f and has the same vector length as of the control bit vector. However, the argument vector is about one half the length of vector f. Using the MERG function and a dummy value vector we can generate a vector of required length, having values of $f_{i,j}^{(n+1,r)}$ in the desired elements and the rest of the elements having the values of the dummy vector. The values of the elements of a dummy vector are insignificant and can be chosen arbitrarily. As shown in fig. 6, MERG merges dummy vector DM, having arbitrary value for all elements and vector $f_{i,j}^{(n+1,r)}$ into a result vector RS, with the help of a control vector. The merge stops when the result vector RS is full. Now having obtained vector RS of appropriate length and elements, the execution of the CTRL function with f as a result vector and RS as an argument vector under the control of a given bit vector will update the values of f at the red field nodes (fig. 7). This completes the implementation of the first or red stage (eq. 6.3) on the CYBER-203 computer using the bit control vector approach.

The implementation of the second or black stage (eq. 6.4) can be incorporated in the same framework using an updated array f and appropriate control bit vectors. Each iteration is made up of the two stages and the above iterative procedure for vector processing can be continued until desired accuracy is obtained.

It is obvious that vector algorithms require more storage than scalar algorithms. However, due to large memory (1 million, 64 bit words) and sharing same storage locations the increased storage requirement can be handled properly in many applications. Instead of using the bit control vector approach, the above can be implemented using integer index vectors which are incorporated in functions such as GATHR (gather and SCATR (scatter). In these functions, instead of a bit vector, an integer vector with appropriate index values is used as a control vector. The integer index vector approach was not investigated in the present study. It is also interesting to note that in many applications the bit control vectors of desired structure need to be generated once only and can be used many times in a computer code. Since each element of a bit vector corresponds to one bit only, the storage requirement for bit vectors are far less than conventional value vectors.

6.4 The Navier-Stokes Equations and Checkerboard SOR

The governing equations for incompressible flow about an airfoil are the Navier-Stokes equations. In the present study, equations (3.21) and (3.22) for the velocities and equation (3.23) for the pressure are solved simultaneously at each time step using the checkerboard SOR method. These transformed equations are somewhat complicated compared to the model problem and its implementation on the vector processor is more involved.

The transformed or computational plane is rectangular regardless of the shape of the physical plane. The field nodes in the 2-D transformed plane are represented in a checkerboard pattern so that each red grid point has four black neighbors and vice versa. Three unknowns, two velocities and the pressure are associated with each node. All terms on the RHS of equations (3.21), (3.22) and (3.23) are represented using appropriate finite-difference approximations as discussed in Chapter V. Thus all terms on the RHS of these difference equations can be represented in vector form as discussed for the test problem in the previous section. The storage of each term, including geometric coefficients, requires two vectors, each of about one-half the size of the entire field. Explicit vector instructions are employed to perform the arithmetic operations involved in the equations.

The transformed equations contain cross derivatives. The cross derivatives are evaluated using the central difference approximations. When solving the equations for a red field node, evaluation of cross derivatives involve red nodes in contradiction to the updated black nodes involved in evaluation of first and second derivatives. Although the cross derivatives are lagging by one stage per iteration in the present formulation, it did not show certain adverse effects on the convergence rate during numerical experimentation. Gourlay et. al. [51] has discussed handling of cross derivatives in some hopscotch methods.

It is desirable to use the checkerboard SOR with relaxation parameter varying from iteration to iteration instead of a constant relaxation parameter to accelerate convergence. The major unresolved problem concerning the checkerboard SOR is that of determination of sequence of optimum parameters which will produce the smallest number of iterations for a specified degree of convergence. For solution of the velocity equations using the classical SOR, the computation of sequence of acceleration parameter proposed by Thompson [52] and described in Appendix B produced nealy optimal iterative procedure in previous investigations [29, 30]. In many cases the values of computed acceleration parameters for the checkerboard SOR and theoretical optimal acceleration parameters for the classical SOR are comparable and have about the same range [53]. The typical values of acceleration parameters are less than one for the velocities and the pressure is not accelerated. The additional operations involved in computation of acceleration parameters is justified by producing faster convergence to the checkerboard SOR.

The sequence used for solving the three governing equations simultaneously may show some, if not impressive, improvement of the convergence rate. Out of several possible sequences, the sequence of solving the velocity for the red nodes than the pressure for the black nodes again the velocity for the black nodes and the pressure for the red nodes is found to have favorable convergence characterisitcs.

The details of all interesting features of the computer code and other studies will not be presented, partly due to the lack of space and partly because the outcome of some numerical experiments seems to be inconclusive. However, the present discussion shows that for the solution of large scientific problems on a vector computer, a consistent algorithm will always out perform an inconsistent algorithm implemented without considering the architecture of the computer.

Chapter VII

COMPUTATIONAL RESULTS

7.1 Coordinate Systems

Two different approaches discussed in Chapter II were used to generate "C" type coordinate systems for the NACA 66_3-018 airofil, which is symmetric and has maximum thickness ratio of 18%. The grid contains IL points on the ξ axis and JL points on the η axis, in particular, the values of IL and JL were set to 113 and 51 respectively for all coordinate systems. The major concern was to obtain accurate numerical resolution of the flow field about the airfoil. Since grid characteristics such as mesh spacing, smoothness and skewness can greatly affect the effectiveness of the hosted algorithm, it was decided to examine some effects of the grid characteristics on the flow field solution.

It is desirable to have much finer grid spacing in the regions of boundary layers containing relatively high velocity gradient because a relatively coarse grid can lead to significant truncation errors in the solution of the Navier-Stokes equations. The RHS of the Poisson pressure equation contains velocity gradient terms and hence errors in dominant velocity gradient terms can result in erroneous values of the pressure near the body surface. The wall pressure boundary condition equation uses one sided difference approximation so errors in the pressure field near the wall can lead to errors in the implementation of the boundary conditions. The algebraic eddy viscosity turbulence model used in this study involves velocity gradient term and accurate computation of velocity gradients is important for consistent turbulent modeling.

Whenever a grid in the physical plane is not smooth the transformation coefficients such as ξ_x, η_x, ξ_y and η_y can induce considerable numerical error in the solution caused by the nonuniform grid spacing. In some cases the grid skewness can also lead to numerical oscillations and inaccuracies. A detailed discussion about the effects of these grid characteristics on the solution can be found in references [11,34].

Three coordinate systems were used in the present effort. The first coordinate system CORD1 (fig. 8) was generated using Sorenson's [35] approach and was rather crude. This coordinate system was used for development, testing and debugging of the computer code in the early stages. Coordinate systems CORD2 (fig. 9) and CORD3 (fig. 10) were generated using Thompson's [10] approach and Sorenson's [35] approach respectively. The grid point distribution on the inner boundary was the same for these two coordinate systems. Also these two coordinate systems were extensively used for many numerical experiments and solutions to be presented in the remainder of this section.

As central-difference approximations used in this study are susceptible to numerical oscillations at higher Reynolds number, it was also decided to investiage effects of grid characteristics at lower Reynolds number to isolate oscillations caused by central-differences. Two coordinate systems CORD2 and CORD3, having the same grid point distribution on the inner boundary were tested for Reynolds number 1000. The solution is the trailing edge region had a dominant effect on the total flow field solution. Since the grid lines of CORD2 (fig. 9) are skewed in the trailing edge region, coordinate systems CORD3 (fig. 10) was generated with nearly orthogonal lines in the trailing edge region.

Better overall results were expected using coordinate system CORD3, however it turned out the other way. Numerical results obtained using coordinate system CORD2 were much better than those obtained using CORD3. It was thought that other grid parameters such as coordinate stretching functions and the rate of change of grid spacing would have significant effect on the solution [34, 54]. An exponential stretching function was used in Sorensen's approach while in Thompson's approach coordinate control function was derived using the hyperbolic tangent as the point distribution function. As mentioned in reference [34], the hyperbolic tangent is better than exponential and gives optimal truncation error. It should be noted here that the control function in Sorensen's approach controlled only spacing of the first line off the boundary and angle of inclination of ξ = constant lines with the boundary while the control function in Thompson's approach was able to control grid line distribution (fig. 2). The above case is not entirely conclusive, however it does show the importance of a proper coordinate system for the Navier-Stokes solution. For a given problem finding of an optimum coordinate system by trial and error method is expensive, so we decided to limit our experimentation with coordinate systems.

Some important parameters of two coordinate systems CORD2 and CORD3, which were used extensively in this study, are described below. For both grids the leading edge of the airfoil was located at (0,0) and the trailing edge was at (1,0). The y coordinates of the uppermost point at I = 113 was +5.09 (5 chord lengths) the lowermost point at I = 1 was at -5.09. The x coordinate of the forward most point was -4.09 and of the backwardmost point was 11.0 (11 chord lengths). The value of index i at the leading edge was 57. The lower-surface trailing edge and upper-surface trailing edge points were located at $i = 21$ and $i = 93$ respectively. The value of index i at the maximum airfoil thickness point on the lower and upper surface were 34 and 80 respectively, and the value of the x coordinate was 0.467. The term Δs denotes grid spacing between the first line off the boundary and boundary along ξ = constant lines. The values of Δs at the inner boundary for coordinate system CORD2 were 0.000046, 0.000055, 0.000010 and 0.000026 at $I = 1$, 21, 34 and 57 respectively. The minimum values of Δs at the inner boundary was 0.000001. The values of Δs at the outer boundary were 0.28, 0.35, 0.32 and 0.42 at $I = 1$, 21, 34, and 57 respectively. For the coordinate system CORD3 the values of Δs at the inner boundary were set to 0.00001 and at the outer boundary were set to half the chord length (i.e. 0.5). The angles of inclination with which ξ = constant lines intersect the inner and outer boundaries were approximately 90°. For the two coordinate system the grid point distribution at the leading and trailing edge in the boundary layer region is shown in Table 1. Also note that coordinate system CORD2 has a uniform spacing around the airfoil while CORD3 has closer spacing at high curvature regions e.g. leading edge.

7.2 Computational Procedure

In this section a computational procedure for solving the governing equations and some details of the comptuer code are described. An appropriate coordinate system for a given problem was generated using separate grid generation computer code on a scalar machine.

The values of x and y coordinates for a final grid were input to the CYBER-200 FORTRAN compiler code. Bit control vectors of desired pattern were generated and stored. Since all geometric coefficients can

can be efficiently computed using bit vector approach, all tranformation coefficients were computed using bit vector approach, all transformation coefficients were computed by the computer code in the vector mode instead of using values of coefficients supplied by other scalar codes. Second-order accurate central-difference formulas were used to compute transformation coefficients in the field. On the airfoil surface upper down-stream boundary and lower down-stream boundary the transformation derviatves were computed using second order accurate one-sided forward or backward differences. All geometric coefficients were separated for the red and black nodes and stored. For restarting the flow from previously obtained solutions, all required flow field variables and important parameters were read in. Before starting off a loop for time steps, all required bit control vectors for solution of the governing equations were generated and stored once and for all.

All calculations were performed with the time step $t = 0.01$. The free-stream boundarycondition on theouter boundary was applied at every time step. The gradua start consisted of 100 time steps during which the free-stream velocities and the body force terms were given by

$$u_\infty = t * g_1 \qquad \text{for } 0 < t < 1.0 \qquad (7.1)$$

$$v_\infty = t * g_2 \qquad (7.2)$$

$$g_1 = \cos \psi \qquad (7.3)$$

$$g_2 = \sin \psi \qquad (7.4)$$

after the gradual start

$$u_\infty = \cos \qquad (7.5)$$

$$\text{for } t > 1.0$$

$$v_\infty = \sin \qquad (7.6)$$

$$g_1 = 0 \qquad (7.7)$$

$$g_2 = 0 \qquad (7.8)$$

At every time step, two velocities and the pressure equations were solved simultaneously using checkerboard SOR. The sequence of solving the velocities for the red nodes, the pressure for the black nodes, the velocity for the black nodes and the pressure for the red nodes was used for each checkerboard iteration. The convergence criteria for each time step was established by the following procedure. The solution was either initially started or restarted from the previous time step. For the first time step first order two point backward-difference approximations were used for the time derivatives. The iteration continued until difference between the magnitude of each flow variable (u, v and p) at two successive iterations were less than 0.0001. In most cases, maximum number of checkerboard iterations were limited to 50. At every iteration acceleration parameters for the velocities were computed using equations given in Appendix B. The computation of the accelerated parameters is more involved and requires considerable arithmetic operations. A flag was set when the solution converged within 10% of the established convergence criteria. When this flag was set, the computation of accleration parameter was bypassed and iteration continued with the previously computed acceleration parameters to enhance the computational efficiency.

The Neumann pressure boundary conditions, the re-entrant condition and the downstream boundary condition was applied after every checkerboard iteration. The trailing-edge pressure was extrapolated after applying the pressure boundary condition. This completes the solution procedure involved at every iteration.

Once the solution converged within a given error norm or maximum number of iterations allowed were reached, the divergence of the velocity was computed at every time step. As soon as the divergence of the velocity was computed, it was smoothed out in most cases. Then the required turbulence was switched on to compute eddy viscosity. Then the desired artificial viscosity was computed at every time step to incorporate daming. A condition was established that artificial viscosity cannot be turned on unless turbulence was turned on. The above cycle was continued for the desired number of time steps.

7.3 Some Numerical Experiments

This section will present some numerical experiments carried out during the course of this study. It should be noted that all techniques described in this section were not tested thoroughly and some of them did not improve the solution or efficiency significantly. However many of the approaches attmpted, seem encouraging and may work well for other applications. The primary attention was focused upon the development of the efficient computer code and the computation of a reasonably accurate flow field solution using minimum computer resources.

It is true, that the use of an appropriate algorithm is generally much more crucial than coding techniques. However, an optimized code with proper algorithm can increase efficiency considerable. In this study once the algoritmh was settled upon, considerable time was spent on optimizing the code. There are very few loops in the code and they are generally unavoidable such as time step loop and iteration loop. Effort was concentrated to develop a code in the light of fundamental properties of the vector processor which allowed the use of explicit vector instructions. All routines in the code were analyzed using a timing package which prints out a histogram of CPU usage. The main effort was diverted to some possible restructuring of routines and comparing its efficiency based on CPU timing. Initially all routines in the code could be compiled with the highest level of optimization (B)) on the CYBER-200 FORTRAN compiler. To incorporate various approaches to be tested, as discussed in the following paragraphs, a few routines were forced to one lower level of optimization (BE). No attempts were made toward optimizing the memory and data mangaement procedures. Vectors of about 2720 length were employed in the present study. Since the performance of the CYBER-203 increases with increase in vector length an application involving a very large coordinate system can result in relatively greater speed-up. Optimization may involve some work; however, for large scale problems usually it does pay off.

Central differences used to approximate the spatial derivatives are easily succeptible to oscillations at higher Reynolds number. Computed solutions displayed large amplitude oscillations in the flow variables and destroyed the accuracy. The eddy viscosity model increases the molecular viscosity and thus lowers the Reynolds number of the flow. The switching on of the turbulence model appeared to damp some oscillations but it did not show any significant degree of control over large amplitude oscillations. All attempts to obtain the steady state solution for flows at Reynolds number 10,000, which were started from rest, were unsuccesful beyond time $t = 1.0$, even with inclusion of the turbulence model. One-sided differencing schemes may eliminate these nonlinear oscillations and may be vectorized from some simple regions. Hodge [25] used upwind differences for the first derivative terms, except the pressure gradient and velocity divergence terms, seemingly to avoid oscillations caused by central-difference. For "C" and "O" type coordinate systems, incorporation of the upwind-differences requrie checking the sign of contravariant velocity $(\xi_x u + \xi_y v)$ to incorporate appropriate indexing.

This condition may not allow efficient vectorization of one-sided differencing schemes for this study on the CYBER -203.

One possible way to damp out the oscillations is by the use of an artificial viscosity. The adverse pressure gradient in the trailing edge regions had the dominant effect on the solution and it was assumed to trigger the nonlinear oscillations. The amplitude of these oscillations were small initially and remained localized near the trailing edge regions for some time, but in absence of damping its amplitude started increasing. The oscillations propagated toward the leading edge with passage of time. The flow was rather stable till time $t = 0.5$, so in most cases it was decided to turn on damping at time $t = 0.51$, well before the oscillations started contaminating the solution. Several numerical experiments will be described before going into details of various forms of artifical viscosity.

The time derivative of the divergence of velocity D_t, appearing in equation (5.21) can be evaluated using either two point or three point backward difference approximations after the first time step. Several computer runs were made using both options. Error norms obtained using both cases for about the first one hundred time steps were almost the same indicating none was specifically responsible for divergence of the solution. The spatial derivative terms of equation (5.21) were evaluated using second order accurate difference approxiamtions. Thus the use of two-point first order accurate approximation for the time derivaitve may reduce the overall accuracy of the equation. Since no emphasis was placed on the transient solution, and it was assumed that the time derivative term disappears in the steadt state, first order accurate approximation was used for the time derivative term of eq. (5.21) in most computer runs. Although it did not show any noticable increase in computational efficiency, it is interesting to note here that the two point backward approximations involve less computer operation and storage than the three point approximations.

Since the implicit system of equations were solved at each time step by an iterative method, the previous time step solution was used as an initial guess for the next time step in all cases. For some cases a poor choice of initial guess may delay or destroy the convergence of the method. In an attempt to reduce the iterations by providing a good guess of the solution at the next time level an initial guess which was close to the desired solution was tried. The initial guess for the velocities on the field and the re-entrant boundary was found using the following relations during the acceleration phase.

$$u^{n}_{i,j} = u^{n-1}_{i,j} + \Delta t \cos \psi \qquad (7.9)$$

$$v^{n}_{i,j} = v^{n-1}_{i,j} + \Delta t \sin \psi \qquad (7.10)$$

Instead of improving the convergence, the solution started diverging. As the "C" type grid employed is coarse in the outer region and fine near the body and in the wake, the initial guess for the second attempt were found using the above relations only on the re-entrant section during the acceleration phase. Again no improvement was found and in both cases the solution started diverging approximately at time $t = 0.5$. For this study, it is not clear what should be the criteria to choose the initial guess in an effective manner and how it will accelerate the convergence.

It was thought that the downstream boundary conditions may help control the oscillations or allow the passage of oscillations, which originated in the trailing edge region. Instead of the downstream boundary conditions presented in section 4.2, the following downstream boundary conditions were attempted.

$$u = u_\infty \tag{7.11}$$

$$v = v_\infty \tag{7.12}$$

$$p = p_\infty \tag{7.13}$$

The implementation of the free-stream boundary conditions, section 4.2, on the downstream boundary did not improve the solution during the acceleration phase. In another attempt, the velocity boundary conditions were the same as the free-stream boundary condition on the downstream boundary, however the following pressure boundary condtion was used.

$$p_{\xi\xi} = 0 \tag{7.14}$$

Again, this boundary condition did not show any positive effect on the solution. Thus flow is perhaps much more sensitive to the outer and body surface boundary conditions with the downstream boundary condition having no significant influence on the solution.

One possible way to enhance the stability of a numerical solution is to filter out unwanted oscillations using filters or smoothers. The use of smoothers will not eliminate the source of high amplitude oscillations but will control them by spreading them over some region. Since in some cases wavy solutions with high amplitude of flow quantites such as divergence of velocity, pressure can lead to unrealistic solution, the use of smoothers can help control oscillations. The divergence of velocity showed a wavy field in the direction of ξ = constant lines. Two types of divergence of velocity smoothers were attempted, one using the neighboring nodes in ξ = constant lines direction and the other using four neighboring nodes in both directions. Smoothing obtained using four neighboring nodes was much better. It reduced the need of the artificial viscosity by some margin. The pressure field was wavy in the direction of η = constant lines. An attempt to smooth the pressure led to an incorrect solution. Next the source terms of the pressure equation (3.23) were smoothed out. The solutions obtained using this approach were encouraging. Since the pressure equation was solved using an iterative method, for consistent smoothing the smoother should be applied at every iteration in contrast to the divergence smoother which was applied at every time step. The source term smoother may be computationally inefficient due to computer operations and additional storage required. For some similar runs, results obtained using the source term smoothers were about the same as those obtained using the divergence smoother. Considering the above tests, the source term smoother was not a practical way of smoothing the pressure oscillation in the present study, and hence the divergence smoother was employed for most computations. For turbulent flows, the values of eddy viscosity, computed using the two-layer algebraic model, varied considerably in the boundary layer and wake region. Some forms of artificial viscosities, to be presented later, were based on the eddy viscosity. Most of them used unsmoothed values of the eddy viscosity, however, a few of them employed smoothed values of the eddy viscosity. Artificial viscosities, which employed smoothed values of the eddy viscosity did not show any increase of effectiveness over the artificial viscosities based on unsmoothed eddy viscosity.

Perhaps the most effective way of eliminating the flow field oscillations caused by central-differencing at higher Reynolds number is to use an aritificial viscosity. In this study it was assumed that the flow was turbulent when an artificial viscosity was switched on and hence the molecular viscosity $\mu = 1 + \varepsilon$ in the momentum equations was replaced by $\mu = 1 + \varepsilon + \mu_a$. Term μ_a denotes artificial viscosity and it increases the value of molecular viscosity. An artificial viscosity having uniform or constant value over the whole flow field will increase artificial diffusion everywhere in the field and is obviously not the solution of the problem. However, an artificial viscosity which is a function of some flow quantities having appreciable values in the region of extreme oscillations and negligible values everywhere else can effectively diffuse oscillations without changing characterisitcs of the original flow field considerably. Various forms of attempted artificial viscosities are listed below

$\Omega ReJ\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ $\quad \underset{\sim}{V} = \underset{\sim}{i}u + \underset{\sim}{j}v$

$\Omega = 0.0001 - \Omega = 1.0$

$\Omega ReJ\ |\underset{\sim}{\omega}|$ $\quad \omega = v_x - u_y$

$\Omega = 0.01 - \Omega = 1.0$

$\Omega ReJ\ |\underset{\sim}{\nabla}(V^2)|$ $\quad \Omega = 1.0$

$\Omega ReJ\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ $\quad \Omega = \varepsilon$

$\Omega ReJ\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ $\quad \Omega = V^2$

$\Omega ReJ\ \sqrt{|\underset{\sim}{\omega}|\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|}$ $\quad \Omega = 1.0$

$\Omega ReJ\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ $\quad \Omega = \{(\frac{\partial V^2}{\partial \xi})^2 + (\frac{\partial V^2}{\partial \eta})^2\}^{1/2}$

$\Omega ReJ \sqrt{\lvert\underset{\sim}{\omega}\rvert \, \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert}$	$\Omega = \{(\frac{\partial V^2}{\partial \xi})^2 + (\frac{\partial V^2}{\partial \eta})^2\}^{\frac{1}{2}}$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = (e^{\varepsilon} - 1.0)$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = \phi \, \varepsilon \, \lvert\underset{\sim}{\omega}\rvert \quad \phi = 1.0 - \phi = 10.0$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = (e^{\phi\varepsilon} - 1.0)\lvert\omega\rvert, \; \phi = 1.0 - \phi = 7.0$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = \phi \, \lvert\underset{\sim}{\omega}\rvert \quad \phi = 1.0 - \phi = 10.0$
$\Omega ReJ\Delta t \, \lvert\nabla^2 P\rvert$	$\Omega = 1.0$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = \phi \, \varepsilon \quad \phi = 10.0 - \phi = 1000.0$
$\Omega ReJ \lvert\underset{\sim}{\nabla} \cdot \underset{\sim}{V}\rvert$	$\Omega = \min(\phi \, \varepsilon, 1.0) \; \phi = 1.0 - \phi = 10.0$

Artificial viscosity was applied at every time step and results obtained using some of the above described forms of artificial viscosities will be presented in the next section.

7.4 Numerical Results

The following general procedure was established for numerical computation and it was common to many of the flow solutions attempted. A NACA 66_3018 airfoil section at zero angle of attack was considered for all computations. Coordinate systems with 113 (IL) grid points in ξ direction and 51 (JL) points in η direction were used. Gradual start was made up of 100 time steps with a time step size of 0.01. The previous time step solution was used as an initial guess for the next time step solution. The acceleration parameters were computed using the local velocities. The convergence criteria for the velocity and pressure were 10^{-4} and the maximum number of iterations at each time step were limited to 50. First order time differencing was used at the first time step and second order time-differencing scheme was used for all subsequent time steps. The flow was laminar till time $t = 0.5$. For turbulent flow, computation of eddy viscosity was turned on at time $t = 0.51$ and transition was assumed to occur at minimum pressure on the upper and lower airfoil surfaces. For solutions with an artificial viscosity, the artificial viscosity was turned on at time $t = 0.51$ and it was computed at every time step. Also, whenever artificial viscosity was switched on, turbulence was assumed to be turned on at the same time. Except for some initial cases, the trailing edge pressure was extrapolated and the divergence of velocity smoother was used for the flow simulation. Some exceptions to the above-described procedure will be mentioned at appropriate places in the following paragraphs.

The first type of coordinate system CORD1 considered in this study was generated using Sorenson's approach [35] and is shown in fig. 8.

The laminar flow past the airfoil was considered at Reynolds number of 10,000. The pressure distributions at time $t = 0.5$ and $t = 0.7$ are shown in fig. 11 and fig. 12 respectively. These pressure distributions indicate that the flow was well behaved until time $t = 0.7$. The solution was restarted from time $t = 0.5$ with the turbulence model switched on and transition occuring at the minimum pressure. The turbulent flow solution at time $t = 0.7$ was essentially the same as the laminar flow solution at the same time. At time $t = 0.8$ the pressure started oscillating at the trailing edge (fig. 13). With the passage of time, the solution diverged at time $t = 0.85$. To damp out the trailing edge oscillations in the turbulent flow, the transition was forced to occur at the maximum airfoil thickness points on both surfaces instead of minimum pressure points which were almost at the trailing edge. As shown in fig. 14 the amplitude of the pressure oscillations was reduced somewhat at $t = 0.8$, however the solution diverged at time $t = 0.89$. Again turbulent flow was restarted from time $t = 0.5$ with artificial viscosity added. The artificial viscosity was computed using $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, with $\Omega = 1.0$. Previously observed oscillation disappeared at time $t = 0.8$ [fig. 15] due to artificial diffusion. As expected, the pressure coefficient was going down with increase in time. Figure 16 shows the pressure distribution at time $t = 1.0$. With further increase in time, the solution diverged at $t = 1.22$. In pressure distributions for all stable solutions described so far, there was abrupt pressure rise at the trailing edge. To remedy this problem, the pressure at the trailing edge was extrapolated using eqs. (4.17-4.18). Fig. 17 shows the pressure distribuiton with the extrapolation for turbulent flow at $t = 0.7$. Another form of artificial viscosity, $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, where

$\Omega = \varepsilon$ was attempted with transition occuring at maximum airfoil thickness points. The solution started oscillating at $t = 1.0$ (fig. 18). Once again an artificial viscosity , $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = 1.0$, was attempted. However this time the artificial viscosity was computed at every iteration instead of every time step. Pressure distributions at time $t = 1.0$ and $t = 2.0$ are shown in fig. 19 and fig. 20 and the solution diverged at $t = 2.28$. Next an artificial viscosity $\Omega ReJ\sqrt{|\underset{\sim}{\omega}|\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|}$ with $\Omega = 1.0$ was considered. Fig. 21 and fig 22 show the pressure distribution at $t = 1.7$ and $t = 2.0$. At time $t = 1.6$ the solution already started oscillating in the trailing edge region. Since coordinate system CORD1 was rather crude, computations using CORD1 were stopped.

A second coordinate system CORD2 (fig. 9) was generated using Thompson's approach [10] with control functions involving hyperbolic tangent to control η-line spacing in the boundary layer region. The Reynolds number considered was 10,000. The pressure distributions and leading - edge and trailing-edge velocity vectors at time $t = 0.5$ and $t = 0.7$ for the laminar flow are shown in fig. 23 and fig. 24. The turbulent flow was restarted from time $t = 0.5$ with transition occuring at minimum pressure. Fig. 25 and fig. 26 show the pressure distribution and velocity vectors at time $t = 0.7$ and $t = 1.0$. The laminar and turbulent solution at time $t = 0.7$ were almost the same. An abrupt increase in the magnitude of velocity vectors at time $t = 1.0$ in the trailing edge region indicates the presence of nonlinear oscillations in the solution. The turbulent flow solution diverged at $t = 1.07$. The turbulent flow with transistion forced at the maximum airfoil thickness was considered next and the solution at time $t = 1.0$ is shown in fig. 27. No significant improvement in the solution was found. An artificial viscosity, $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = \varepsilon$ and transition occuring at maximum airfoil thickness was attempted. Comparing previous solutions with this solution at time $t = 1.0$ (fig. 28), no sufficient diffusion is the trailing edge region could be obtained. Perhaps, this was due to very small values of eddy viscosity which diluted the artificial viscosity. The solution diverged at time $t = 1.32$. An artificial viscosity $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = 1.0$ was considered. The turbulent flow solutions, using this artificial viscosity, at time $t = 1.0$ and $t = 2.0$ are shown in fig. 29 and fig. 30. The solution diverged at time $t = 2.18$. These solutions indicated nonlinear oscillations in the trailing edge region, which could not be eliminated using the above described forms of artificial viscosity. It was thought that these oscillations were caused by skewed grid lines in the trailing edge region (fig. 9) which ultimately destroyed the solution.

A third coordinate system CORD3 (fig. 10) was generated with nearly orthogonal lines in the trailing edge region using Sorensen's approach. Laminar flow solutions for Reynolds number 10,000 at time $t = 0.5$ and $t = 0.7$ are shown in fig. 31 and fig. 32. The flow was well behaved as expected. Turbulent flow solution with transition occuring at minimum pressure was attempted and fig. 33 shows the solution at $t = 0.9$. Velocity vectors at the trailing edge reversed their direction with unusually large magnitude. This phenomena also indicates the presence of oscillations in the solution. The solution diverged at time $t = 1.02$. In an attempt to damp out these oscillations, transition was enforced at maximum airfoil thickness points. Fig. 34 shows the pressure distribution and velocity vector plots at time

$t = 0.9$. Note the magnitude of velocity vectors at the trailing edge has reduced somewhat but not sufficiently. At this stage, the divergence of velocity smoother and two-point backward differencing scheme for $\frac{\partial D}{\partial t}$ term of the pressure equation were incorporated. The solution was started from rest and the pressure distribution and velocity vectors at time $t = 1.0$ are shown in fig. 35. The RHS smoother, as discussed in Section 5.6 was also considered. The solution at time $t = 1.0$ is shown in fig. 36. No significant differences in the solutions obtained using these two smoothers were found. Smoothing of the pressure gave invalid solutions. The smoothers were able to filter out some oscillations at the trailing edge. For all results presented hereafter, the divergence of velocity smoother was turned on and two-point backward differencing scheme for $\frac{\partial D}{\partial t}$ term were employed all the time. Attention was now focused on the artificial viscosity. Fig. 37 shows the solution obtained at $t = 1.0$ using an artificial viscosity $\Omega \mathrm{Re} J|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = \{(\frac{\partial V^2}{\partial \xi})^2 + (\frac{\partial V^2}{\partial \eta})^2\}^{\frac{1}{2}}$. Artificial viscosity with the same Ω, but of form $\Omega \mathrm{Re} J\sqrt{|\underset{\sim}{\omega}|\ |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|}$ was considered next. Solution (Fig. 38) obtained using the latter form was somewhat better. Solutions obtained at $t = 1.0$ using artificial viscosity $\Omega \mathrm{Re} J|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = \varepsilon|\underset{\sim}{\omega}|$ and $\Omega = 1.0$ are shown in fig. 39 and fig. 40 respectively. An artificial viscosity based on Laplacian of the pressure i.e. $\Omega \mathrm{Re} J \Delta t|\nabla^2 P|$ with $\Omega = 1.0$ was attempted and the solution at $t = 1.0$ is shown in fig. 41. Artificial viscosity $\Omega \mathrm{Re} J|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = [e^{\phi\varepsilon} - 1.0]|\underset{\sim}{\omega}|$ and $\phi = 4.0$ gave some interesting results [fig. 42]. The values of ϕ greater than 7.0 led to divergence of the solution for this particular case. The time history of solution obtained using artificial viscosity $\Omega \mathrm{Re} J|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$

with $\Omega = \phi|\underset{\sim}{\omega}|$ and $\phi = 1.0$ is shown in figs. 43 - 46. The solution was almost steady at time $t = 4.0$ (fig. 46). This Reynolds number 10,000 numerical solution was compared with Reynolds number 40,000 experimental solution [31] for qualitative purpose only. The discrepancies between the computational experiment results was thought due to unreasonably thick boundary layer and/or due to grid characteristics such as stretching function. An attempt was made to restart flow from time $t = 4.0$ without inclusion of the artificial viscosity, to obtain correct boundary layer. However, the solution diverged at time $t = 4.28$. Several values of ϕ, ranging from 0.05 to 0.9 were experimented with but the results were not encouraging. Several computer runs with artificial viscosity given by $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ and $\Omega = \phi\varepsilon$ for values of ϕ from 10 to 1000 were made but without certain improvement. At this point, it was decided to lower the Reynolds number to isolate oscillations caused by higher Reynolds number and to investigate the effects of the coordinate systems on the solution.

Coordinate system CORD3 was used for a Reynolds number 1000. The pressure distribtuion and velocity vectors plots for the laminar flow at time $t = 1.0$ and $t = 1.5$ are shown in fig. 47 and fig. 48. With increase in time the solution diverged at time 2.02 and the pressure distribution at time $t = 2.0$ is shown in fig. 49. Perhaps insufficient grid resolution in the boundary layer was responsible for the divergence of the solution.

Next, coordinate system CORD2 was considered for the laminar flow at a Reynold number of 1000. Time history of the flow at time $t = 1.2$, 2.0, 3.0 and 4.0 is shown in figs. 50 - 53. The flow characteristics were not changing with further increase in time and the number of iterations came down to 8. Hence the flow was considered steady at time t = 4.0. For qualitative comparison, the numerical solution is compared with the experimental solution at a Reynolds number 40,000. The discrepancies in the pressure distribution from the leading edge to the maximum airfoil thickness can be identified. It is interesting to note here that coordinate system CORD2 allowed to obtain the steady state solution while CORD3 did not. A coordinate control function which was found using the hyperbolic tangent as the point distribution function was used for coordinate system CORD2. It was noted in reference [34] that a hyperbolic tangent function gave optimum truncation error. With the same coordinate system, i.e., CORD2 an attempt was made to obtain Reynolds number 10,000 solution by restarting the laminar flow from Reynolds number 1000 solution at t = 4.0. The pressure distribution and velocity vectors for Reynolds number 10,000 at t = 5.0 is shown in fig. 54.

It was thought that the accuracy of the solution during the acceleration phase had significant effect on the total flow field solution. Hence coordinate system CORD2 was considered for Reynolds number 1000 laminar flow solution with increased number of iterations in initial stages. The error norm used was the same as before (10^{-4}), however, the maximum number of iterations for initial time steps were increased to satisfy the above convergence criteria exactly. The number of iterations started increasing from 10 at the first time step to 75 at time t = 1.0, about 100 at time t = 1.5, about 90 at t = 2.0 and about 60 at t = 2.5. Some test runs were made with increased number of iterations and maximum number of iterations fixed at 50 beyond time t = 3.0. No noticible difference between the solutions using 50 and increased iterations was found. Hence the maximum number of iterations beyond time t = 3.0 were again fixed to 50. The time history of the solution starting with time t = 1.0 till time t = 10.0 is shown in figs. 55 - 64 at an interval of 100 time steps. Note the difference in the pressure distribution at t = 1.0 between this case (fig. 55) and a case with fixed iteration (fig. 50). This difference in the pressure distribution becomes more obvious at time t = 2.0, 3.0 and 4.0. For the present approach the solution had not achieved steady state at t = 4.0, with number of iteration about 40. Note that starting at time t = 3.0 velocity vectors at the leading edge region start changing its angle of inclination gradually and becoming parallel to the airfoil surfaces. Also the magnitude of velocity vectors in the trailing edge region keep increasing with passage of time. No noticeable difference in the pressure distribution was found between the solution at time t = 9.0 and t = 10.0 and hence computations for Reynolds number 1000 were stopped at t = 10.0. The pressure distribution in the leading edge region (fig. 64) has improved considerably compared to the previous steady state solution (fig. 53). Also, the experimental results at Reynolds number 40,000 matched qualitatively better than previous approaches. The pressure distribution in the leading edge region was the major cause of discrepancies. A closer look at coordinate system CORD2 (fig. 9) shows that there is a sudden change in grid points spacing after approximately 9 points from the leading edge on both, upper and lower surfaces. Note that these points were placed by curvature of this surface.

Perhaps the redistribution of points in this region may help us to obtain correct pressure solution.

Finally, Reynolds number was increased to 40,000 and the flow was restarted from the Reynolds number 1000 laminar solution at $t = 10.0$. The maximum number of iterations were limited to 50. The laminar flow solution diverged at $t = 10.16$ indicating the presence of large amplitude oscillations at higher Reynolds number. An attempt was made to damp out oscillations with the turbulence turned on at $t = 10.01$ and transition occuring at minimum pressure points. Again, the solution diverged at $t = 10.17$. Next the turbulent flow solution using artificial viscosity $\Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$ with $\Omega = \phi|\underset{\sim}{\omega}|$ and $\phi = 1.0$ was considered. The use of the artificial viscosity allowed a steady solution to be obtained. Some minor oscillations in the trailing edge region were observed at about time $t = 16.0$. Hence the value of ϕ was increased to 10.0 after time $t = 16.5$. The surface pressure distribution and the leading and trailing edge velocity vector plots at time $t = 20.0$ are shown in fig. 65. The separation was found to occur at about 60% chord position on the upper surface and at about 64.1% chord position on the lower surface. The computed surface pressure distribution is compared with the experimental data. The use of artificial viscosity increased the thickness of the boundary layer. The surface grid point distribution was though to be responsible for the discrepancies between the computed and experimental surface pressure distribution in the leading edge region. The distribution of the divergence of the velocity and the total viscosity $(1 + \varepsilon + \mu_a)$, for the first 20 η = constant lines, at the leading and trailing edge is shown in Table 1. The maximum of the divergence of the velocity field occured at the second node point off

C-2

the wall at the leading edge and the value of the total viscosity at the same node was 1.11. Since the value of the eddy viscosity ε was zero at the leading edge, the value of artificial viscosity was 0.11 at that point. With increase in the value of J, the magnitude of the divergence started decreasing, however values of the aritifical viscosity was increasing till J = 10 and then it started decreasing. At the trailing edge the values of the divergence of the velocity were less compared to the values at the leading edge. However, the values of the artificial viscoisty at the trailing edge were larger than at the leading edge. The values were increasing with increase in J till J = 15 and then it started decreasing. The increase in the values of the artificial viscosity at the trailing edge was probably due to increase in the magnitude of vorticity and increase in the cell size.

7.5 Computer time

The computations were performed on the CDC, CYBER-203 computer at NASA Langley Research Center, Hampton, Virginia. For about 50 iterations per time step, average CPU time for these computations was observed to be 3.3 seconds/time step. This compares to 37.4 seconds/time step [30] for a similar serial code on the CDC 7600. A factor of 11.36 was observed improvement in speed. A coordinate system with 5763 (113 x 51) grid points was used in the present study, giving average computational rate of 1.145×10^{-5} seconds/iteration/grid point. Further increase in speed through data management optimization and additional code seems possible.

Chapter VIII

CONCLUSIONS

The prime motivation of this study was to develop a vectorizable algorithm for the implicit finite-difference solution of the incompressible Navier-Stokes equations in general curvlinear coordinates. The results indicate that it is economically feasible to obtain flow field soluiton past complex geometries. Much of the present effort was diverted to the numerical solution of the incompressible two-dimensional Reynolds averaged Navier-Stokes equations in nonconservative primitive variable formulation on the vector computer, especially to the development of a relaxation technique amenable to vector processing. The checkboard SOR relaxation technqiue and boundary-conforming coordinate system make the method efficient and versatile for a wide variety of configurations which could be addressed using a vector computer. The computer code was fully vectorized in the sense that all vectorizable loops were vectorized using explicit vector instructions and arithmetic operations were performed in a vector mode. The present computations on the CYBER-203 indicated a speed gain of about 11 over CDC-7600. The acceleration parameters, based on local velocities, were computed using the classical point SOR analysis and need to be studied in detail to make them optimal for the checkerboard SOR. The present implicit scheme is linearly unconditionally stable excluding the pressure terms. This scheme with central-difference approximations for the spatial derivatives, exhibited oscillatory behavior at the higher Reynolds number. Out of several smoothers attempted, the divergence of velocity smoother proved to be an effective way of filtering oscillations during the early stages of the solution. However, with passage of the time and increase amplitude of oscillations, the effectiveness of the smoother was lost and the accuracy of the solution was destroyed. Perhpas the use of an artificial viscosity was the most effective way of eliminating the flow field oscillations at higher Reynolds number for the present method. For solution at higher Reynolds number, restarting the flow from the steady state solution at lower Reynolds number was not particularly effective in controling the nonlinear oscillations. Thus initial conditions had little effect on the stability of the flow field solution. On the other hand, the accuracy of the solution during the accleration phase had significant influence on the steady solution. The down-stream boundary conditions showed little influence on the total flow field solution. Also it is not clear what type of initial guess for the checkerboard SOR can accelerate the convergence and reduce the number of iterations required for a given error norm. Computed results indicate that it is possible to obtain considerable speed-up using the present method. The effects of several coordinate systems on the numerical solution were studied. The importance of a proper coordinate line distribution to avoid grid induced errors and sensitiveness of the algorithm to the coordinate system were observed. In particular, the rate of change of line spacing in the boundary layer region was found to be more important than the grid line skewness at the boundary. Also, the grid point distribution on the body surface showed considerable influence on the solution. The turbulence model used in this study shows little control over nonlinear oscillations. Hence some compromise in the flow field solution was made by using an artificial viscosity. The use of an artificial viscosity usually made the boundary layer thicker than what it should be, however it did stabilize the flow solution. The effects of several forms of artificial viscosities on the solution were studied and compared for several test runs. Although steady state solutions were not attempted using each of them, the information about their relative merit can help to choose an appropriate form for particular application. Some ranges of a parameter, which was used to control the effect of various forms of airfoil viscosities, were obtained from numerical experiments. The use of various smoothers was found to reduce the need of an artificial viscosity by little margin.

In the course of the present study the coordinate systems played a crucial role. The need for an optimized coordinate system for the Navier-Stokes solutions became apparent. Inability to compute flow field solution on one coordinate system and the discrepancies between the computed results and experimental data on the other coordinate system was perhaps due to the deficiencies of the coordinate systems. An adaptive coordinate system, which adapts to flow field variables gradients in a numerical solution may effectively solve this problem. The dynamic coupling of the coordinate governing equations with the flow field governing equations to resolve developing gradients is perhaps the most promising approach to improve the overall outcome of the present computational procedure. An adaptive grid may eliminate the extreme oscillations encountered using a fixed grid and enhance the performance of the algorithm because fewer iterations will be required due to improved convergence. Since computer operations become more efficient with increase in vector length on the CYBER-203 an application involving large number of grid points can increase the relative speed gain. The CYBER-203 is a one-pipe machine while the CYBER-205 is two-pipe machine. Thus significant speed up in computational rate can be obtained by running the computer code on the CYBER-205. The primitive variable formulation used for the governing equations can be easily extended to three-dimensional problems. For large three-dimensional problems, the increased number of grid points saturate or nearly saturate the available memory. Hence some grid points must be held in secondary storage and they must be transmitted to and from the central memory. As overall execution time is a function of this memory transfer, the memory and data managment become much more important for three-dimensional problems. The computer time required for large scientific problem is generally so large that any increase in efficiency can represent substantial savings. In recent years, success in the development of high technology, such as very large scale integrated (VLSI) systems has revolutionized computer architecture. It seems possible to build a special purpose computer tailored for a special application. The software logic of a relaxation method, such as checkerboard SOR, can be implemented in hardware using VLSI elements. As point relaxation techniques are at the core of many systems of partial differential equations occuring in fluid dynamics, heat transfer, plasma dynamics, electrical network, semiconductor device modelling and structural analysis, the hardware implementation cost can be justified. Success in the development of a special purpose computer in the future will significantly reduce real time processing and cost of computing. The future of high-speed scientific computing, with increased emphasis on vector processing, seems to be quite promising.

APPENDIX A

VARIOUS RELATIONS AND DEFINITIONS IN THE TRANSFORMED PLANE

This appendix contains the relations and definitions necessary to transform the equations of motion and boundary conditions from the physical to the computational plane. All transformations are presented in fully non-conservative form. The two following definitions are applicable throughout this appendix:

$f(x,y,t)$ ≡ a scalar function with continuous first and second derivatives.

$\underset{\sim}{F}(x,y) = \underset{\sim}{i}\, F_1(x,y) + \underset{\sim}{j}\, F_2(x,y)$ ≡ a vector function with continuous first derivatives. $\underset{\sim}{i}$ and $\underset{\sim}{j}$ are Cartesian unit vectors.

Definitions of the Transformation

$$J = x_\xi y_\eta - x_\xi y_\eta \qquad (A.1)$$

$$\alpha = x_\eta^2 + y_\eta^2 \qquad (A.2)$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta \qquad (A.3)$$

$$\gamma = x_\xi^2 + y_\xi^2 \qquad (A.4)$$

$$Dx = \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \qquad (A.5)$$

$$Dy = \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \qquad (A.6)$$

$$\sigma = (y_\xi Dx - x_\xi Dy)/J \qquad (A.7)$$

$$\tau = (x_\eta Dy - y_\eta Dx)/J \qquad (A.8)$$

Transformation of Scalar Derivatives

$$f_x = (\partial f/\partial x)_{y,t} = (y_\eta f_\xi - y_\xi f_\eta)/J \tag{A.9}$$

$$f_y = (\partial f/\partial y)_{x,t} = (x_\xi f_\eta - x_\eta f_\xi)/J \tag{A.10}$$

$$\begin{aligned} f_{xx} = (\partial^2 f/\partial x^2)_{y,t} &= (y_\eta^2 f_{\xi\xi} - 2y_\xi y_\eta f_{\xi\eta} + y_\xi^2 f_{\eta\eta})/J^2 \\ &+ (y_\eta^2 y_{\xi\xi} - 2y_\xi y_\eta y_{\xi\eta} + y_\xi^2 y_{\eta\eta})(x_\eta f_\xi - x_\xi f_\eta)/J^3 \\ &+ (y_\eta^2 x_{\xi\xi} - 2y_\xi y_\eta x_{\xi\eta} + y_\xi^2 x_{\eta\eta})(y_\xi f_\eta - y_\eta f_\xi)/J^3 \end{aligned} \tag{A.11}$$

$$\begin{aligned} f_{yy} = (\partial^2 f/\partial y^2)x,t &= (x_\eta^2 f_{\xi\xi} - 2x_\xi x_\eta f_{\xi\eta} + x_\xi^2 f_{\eta\eta})J^2 \\ &+ (x_\eta^2 y_{\xi\xi} - 2x_\xi x_\eta y_{\xi\eta} + x_\xi^2 y_{\eta\eta})(x_\eta f_\xi - x_\xi f_\eta)/J^3 \\ &+ (x_\eta^2 x_{\xi\xi} - 2x_\xi x_\eta x_{\xi\eta} + x_\xi^2 x_{\eta\eta})(y_\xi f_\eta - y_\eta f_\xi)/J^3 \end{aligned} \tag{A.12}$$

$$\begin{aligned} f_{xy} = (\partial^2 f/\partial x \partial y)_t &= [(x_\xi y_\eta + x_\eta y_\xi) f_{\xi\eta} - x_\xi y_\xi f_{\eta\eta} \\ &- x_\eta y_\eta f_{\xi\xi}]/J^2 + [x_\eta y_\eta x_{\xi\xi} - (x_\xi y_\eta + x_\eta y_\xi) x_{\xi\eta} \\ &+ x_\xi y_\xi x_{\eta\eta}](y_\eta f_\xi - y_\xi f_\eta)/J^3 + [x_\eta y_\eta y_{\xi\xi} \\ &- (x_\xi y_\eta + x_\eta y_\xi) y_{\xi\eta} + x_\xi y_\xi y_{\eta\eta}](x_\xi f_\eta - x_\eta f_\xi)/J^3 \end{aligned} \tag{A.13}$$

$$f_t = (\partial f/\partial t)_{x,y} = (f_t)_{\xi,\eta} - (f_x x_t + f_y y_t) \tag{A.14}$$

Transformation of Vector Derivatives

Laplacian:

$$\begin{aligned} \nabla^2 f &= (\alpha f_{\xi\xi} - 2\beta f_{\xi\eta} + \gamma f_{\eta\eta})/J^2 \\ &- [(Dx)(f_x) + (Dy)(f_y)]/J^2 \end{aligned} \tag{A.15}$$

or

$$\nabla^2 f = (\alpha f_{\xi\xi} - 2\beta f_{\xi\eta} + \gamma f_{\eta\eta} + \sigma f_\eta + \tau f_\xi)/J^2 \qquad \text{(A.16)}$$

Gradient:

$$\underset{\sim}{\nabla} f = [(y_\eta f_\xi - y_\xi f_\eta)\underset{\sim}{i} + (x_\xi f_\eta - x_\eta f_\xi)\underset{\sim}{j}]/J \qquad \text{(A.17)}$$

Divergence:

$$\underset{\sim}{\nabla} \cdot \underset{\sim}{F} = [y_\eta (F_1)_\xi - y_\xi (F_1)_\eta + x_\xi (F_2)_\eta - x_\eta (F_2)_\xi]/J \qquad \text{(A.18)}$$

Unit Normal and Tangent Vectors

Normal to η-line:

$$\underset{\sim}{n}^{(\eta)} = \underset{\sim}{\nabla}\eta/|\underset{\sim}{\nabla}\eta| = (-y_\xi \underset{\sim}{i} + x_\xi \underset{\sim}{j})/\sqrt{\gamma} \qquad \text{(A.19)}$$

Normal to ξ-line:

$$\underset{\sim}{n}^{(\xi)} = \underset{\sim}{\nabla}\xi/|\underset{\sim}{\nabla}\xi| = (y_\eta \underset{\sim}{i} - x_\eta \underset{\sim}{j})/\sqrt{\alpha} \qquad \text{(A.20)}$$

Tangent to η-line:

$$\underset{\sim}{t}^{(\eta)} = \underset{\sim}{n}^{(\eta)} \times \underset{\sim}{k} = (x_\xi \underset{\sim}{i} + y_\xi \underset{\sim}{j})/\sqrt{\gamma} \qquad \text{(A.21)}$$

Tanget to ξ-line:

$$\underset{\sim}{t}^{(\xi)} = \underset{\sim}{n}^{(\xi)} \times \underset{\sim}{k} = -(x_\eta \underset{\sim}{i} + y_\eta \underset{\sim}{j})\sqrt{\alpha} \qquad \text{(A.22)}$$

Directional Derivatives

$$\partial f/\partial \underset{\sim}{n}^{(\eta)} = \underset{\sim}{n}^{(\eta)} \cdot \underset{\sim}{\nabla} f = (\gamma f_\eta - \beta f_\xi)/J\sqrt{\gamma} \qquad \text{(A.23)}$$

$$\partial f/\partial \underset{\sim}{t}^{(\eta)} = \underset{\sim}{t}^{(\eta)} \cdot \underset{\sim}{\nabla} f = f_\xi/\sqrt{\gamma} \qquad \text{(A.24)}$$

$$\partial f/\partial \underset{\sim}{n}^{(\xi)} = \underset{\sim}{n}^{(\xi)} \cdot \underset{\sim}{\nabla} f = (\alpha f_\xi - \beta f_\eta)/J\sqrt{\alpha} \qquad \text{(A.25)}$$

$$\partial f/\partial \underset{\sim}{t}^{(\xi)} = \underset{\sim}{t}^{(\xi)} \cdot \underset{\sim}{\nabla} f = - f_\eta/\sqrt{\alpha} \tag{A.26}$$

<u>Transformation Metrics</u>

$$\xi_x = \frac{y_\eta}{J} \tag{A.27}$$

$$\xi_y = - \frac{x_\eta}{J} \tag{A.28}$$

$$\eta_x = - \frac{y_\xi}{J} \tag{A.29}$$

$$\eta_y = \frac{x_\xi}{J} \tag{A.30}$$

APPENDIX B

The Navier-Stokes equations in non-conservative primitive variables formulation can be represented by the following general partial differential equation, neglecting the cross derivative terms

$$A_1 f_{\xi\xi} + A_2 f_{\eta\eta} + B_1 f_\xi + B_2 f_\eta + Cf + D = 0 \qquad \text{(B.1)}$$

where f denotes velocity u or v. For (IL-1) * (JL-1) simultaneous equations, the spectral radius ζ_j of Jacobi iteration is given by [52].

$$\zeta_j = \frac{1}{|C - 2(A_1 + A_2)|} \{ \sqrt{|4A_1^2 - B_1^2|} \cos(\frac{\pi}{IL-1})$$

$$+ \sqrt{|4A_2^2 - B_2^2|} \cos(\frac{\pi}{JL-1}) \} \qquad \text{(B.2)}$$

The optimal acceleration parameter κ for SOR iterations can be obtained using the following relations

$$\kappa = \frac{2}{1 + \sqrt{1 - \zeta_j^2}} \qquad \text{if } 4A_1^2 \geq B_1^2 \text{ and } 4A_2^2 \geq B_2^2 \qquad \text{(B.3)}$$

and

$$\kappa = \frac{}{1 + \sqrt{1 + \zeta_j^2}} \qquad \text{otherwise} \qquad \text{(B.4)}$$

In this study, when conditions for equation (B.3) were satisfied, instead of computing acceleration parameter using equation (B.3) which gives $\kappa > 1.0$, the acceleration parameters were set equal to 1. In other words, the momentum equations were solved using acceleration parameter less than or equal to 1.0.

The coefficients in equation (B.1) are defined as follows.

$$A_1 = \frac{\mu\alpha}{ReJ^2}$$

$$A_2 = \frac{\mu\gamma}{ReJ^2}$$

$$T_1 = \frac{\mu\sigma}{ReJ^2} + 2(x_\eta v - y_\eta u)$$

$$T_2 = \frac{\mu\tau}{ReJ^2} - 2(x_\xi v - y_\xi u)$$

For x momentum equation

$$B_1 = T_1 + \frac{1}{ReJ}(2\mu_x y_\eta - \mu_y x_\eta)$$

$$B_2 = T_2 + \frac{1}{ReJ}(-2\mu_x y_\xi + \mu_y x_\xi)$$

For y momentum equation

$$B_1 = T_1 + \frac{1}{ReJ}(-2\mu_y x_\eta + \mu_x y_\eta)$$

$$B_2 = T_2 + \frac{1}{ReJ}(2\mu_y x_\xi - \mu_x y_\xi)$$

$$C = -\frac{T}{\Delta t}$$

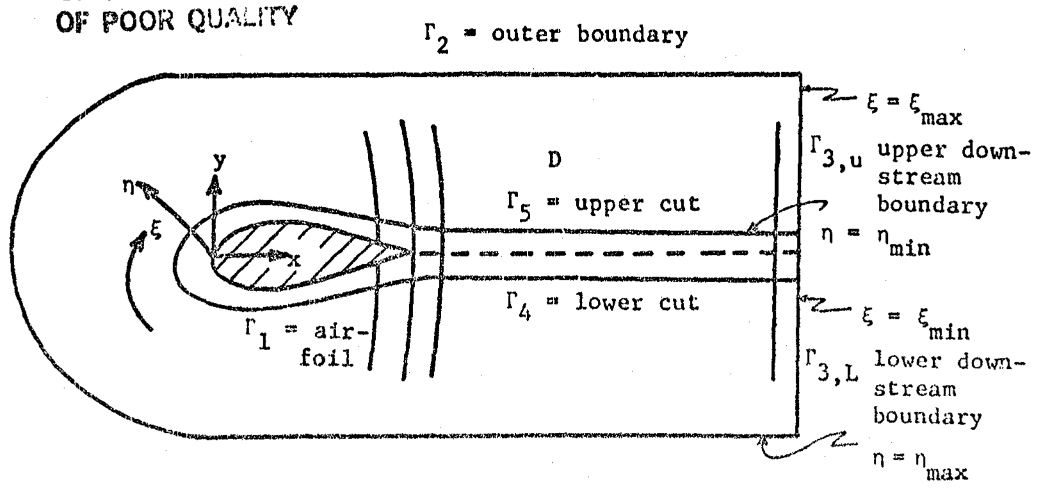
$T = 1$ for first-order time differencing

$T = \frac{3}{2}$ for second-order time differencing

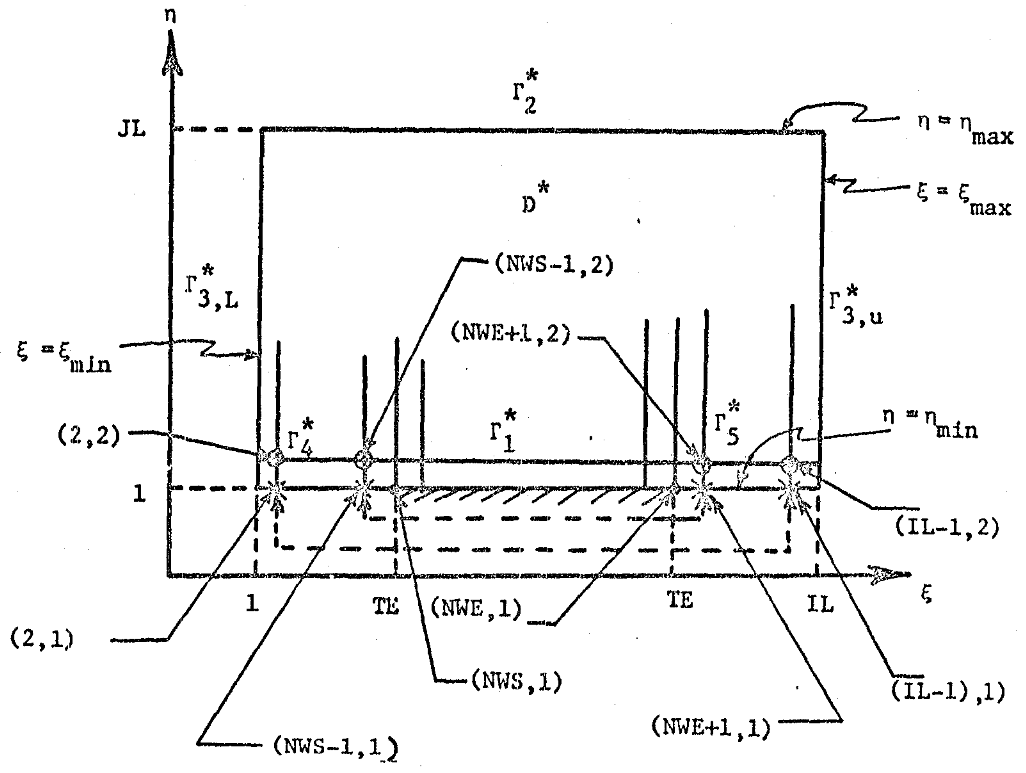
	Coordinate System CORD3		Coordinate System CORD2		Coordinate System CORD2 Re = 40,000 Solution at t = 20.0			
	I = 93	I = 57	I = 93	I = 57	I = 93		I = 57	
J	Arc length S	Arc length S	Arc length S	Arc length S	ABS($\nabla \cdot \underset{\sim}{V}$)	$1 + \varepsilon + \mu_a$	ABS($\nabla \cdot \underset{\sim}{V}$)	$1 + \varepsilon + \mu_a$
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0000116	0.0000116	0.0000559	0.0000026	1.13	3.75	43.22	1.07
3	0.0000269	0.0000269	0.0001273	0.0000061	1.54	5.78	50.96	1.11
4	0.0000470	0.0000470	0.000220	0.0000105	1.53	7.07	50.41	1.14
5	0.0000736	0.0000734	0.000341	0.0000163	1.15	6.79	50.25	1.18
6	0.000108	0.0000108	0.000498	0.0000239	1.44	10.18	49.37	1.23
7	0.000154	0.000153	0.000702	0.0000336	0.96	8.76	48.82	1.29
8	0.000215	0.000213	0.000965	0.000046	1.25	13.75	47.48	1.37
9	0.000295	0.000292	0.001309	0.000062	0.87	12.04	46.38	1.46
10	0.00040	0.000395	0.001756	0.000083	1.03	17.15	44.46	1.57
11	0.000538	0.000530	0.002337	0.000111	0.80	16.35	42.73	1.70
12	0.0007211	0.000706	0.003094	0.000147	0.78	19.17	40.15	1.84
13	0.000962	0.000935	0.004079	0.000193	0.70	20.44	37.72	2.01
14	0.001279	0.00123	0.005363	0.000253	0.54	18.50	34.46	2.19
15	0.001697	0.001621	0.997039	0.000331	0.58	22.48	31.37	2.38
16	0.00225	0.002121	0.00922	0.000433	0.33	14.86	27.53	2.56
17	0.00297	0.00276	0.01206	0.000565	0.43	20.52	23.90	2.73
18	0.00394	0.00359	0.01574	0.000737	0.10	5.83	19.67	2.81
19	0.00521	0.00464	0.02049	0.000962	0.23	13.22	15.74	2.84
20	0.00689	0.00599	0.02659	0.00125	0.11	7.55	11.39	2.67

Table 1. Coordinate Line Distribution and Divergence of Velocity and Artificial Viscosity Field at Leading and Trailing Edge.

ORIGINAL PAGE IS
OF POOR QUALITY

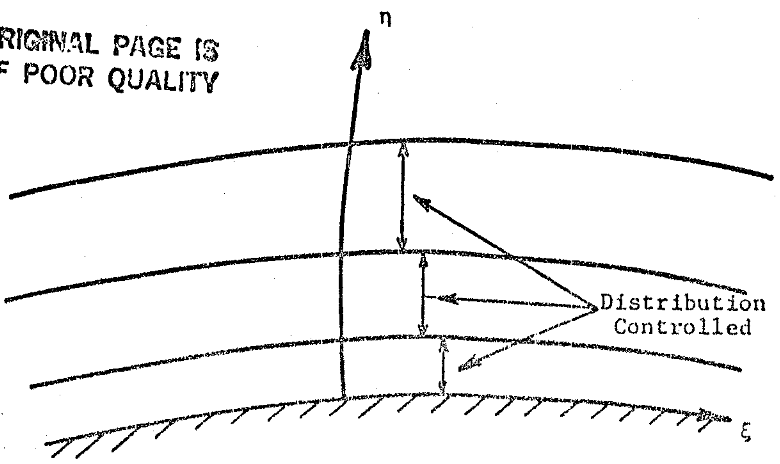


a. Physical Field

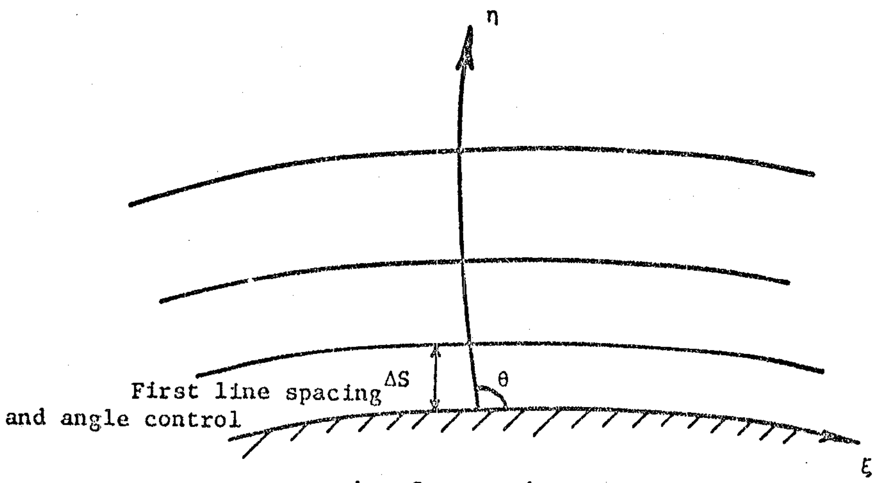


b. Transformed Field

Fig. 1 C-Type Coordinate System

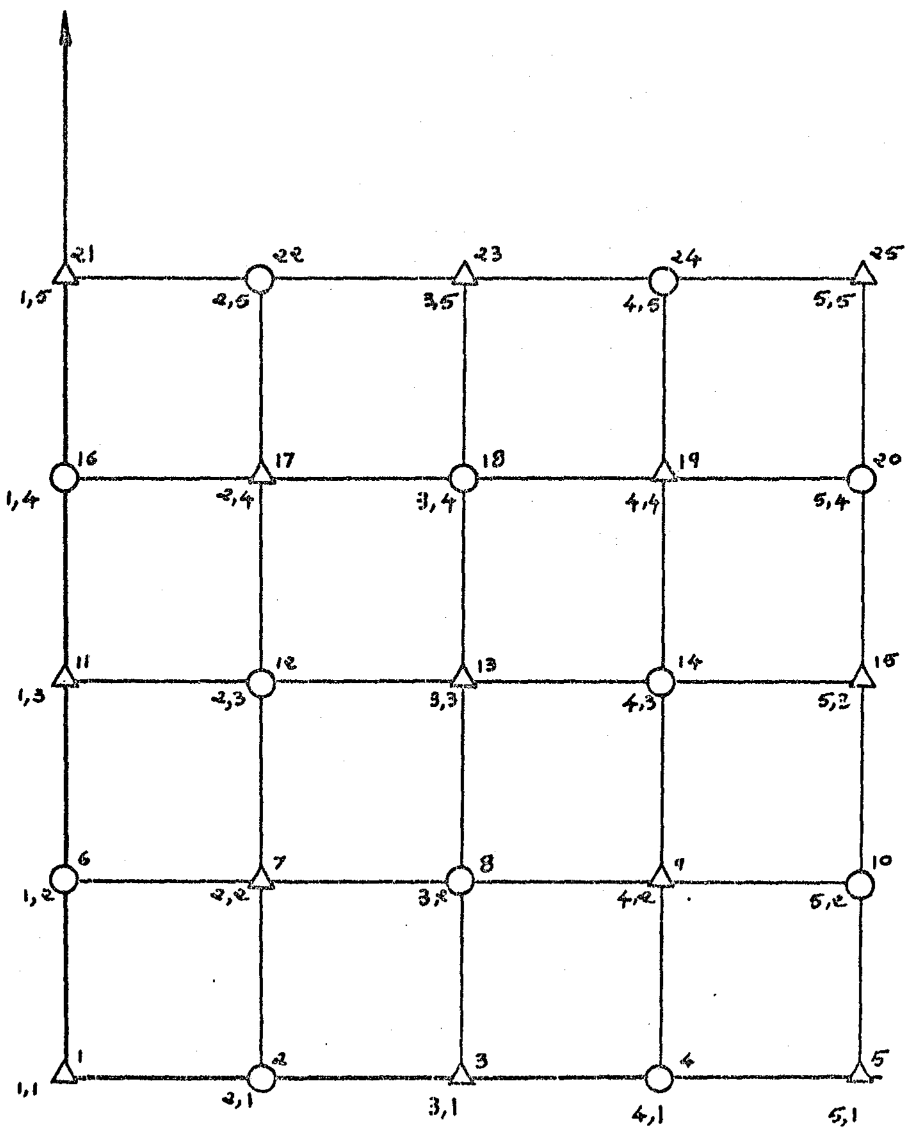


a. Thompson's Approach



b. Sorenson's Approach

Fig. 2 Coordinate Line Control



○ Red

△ Black

Fig. 3 Checkerboard Pattern

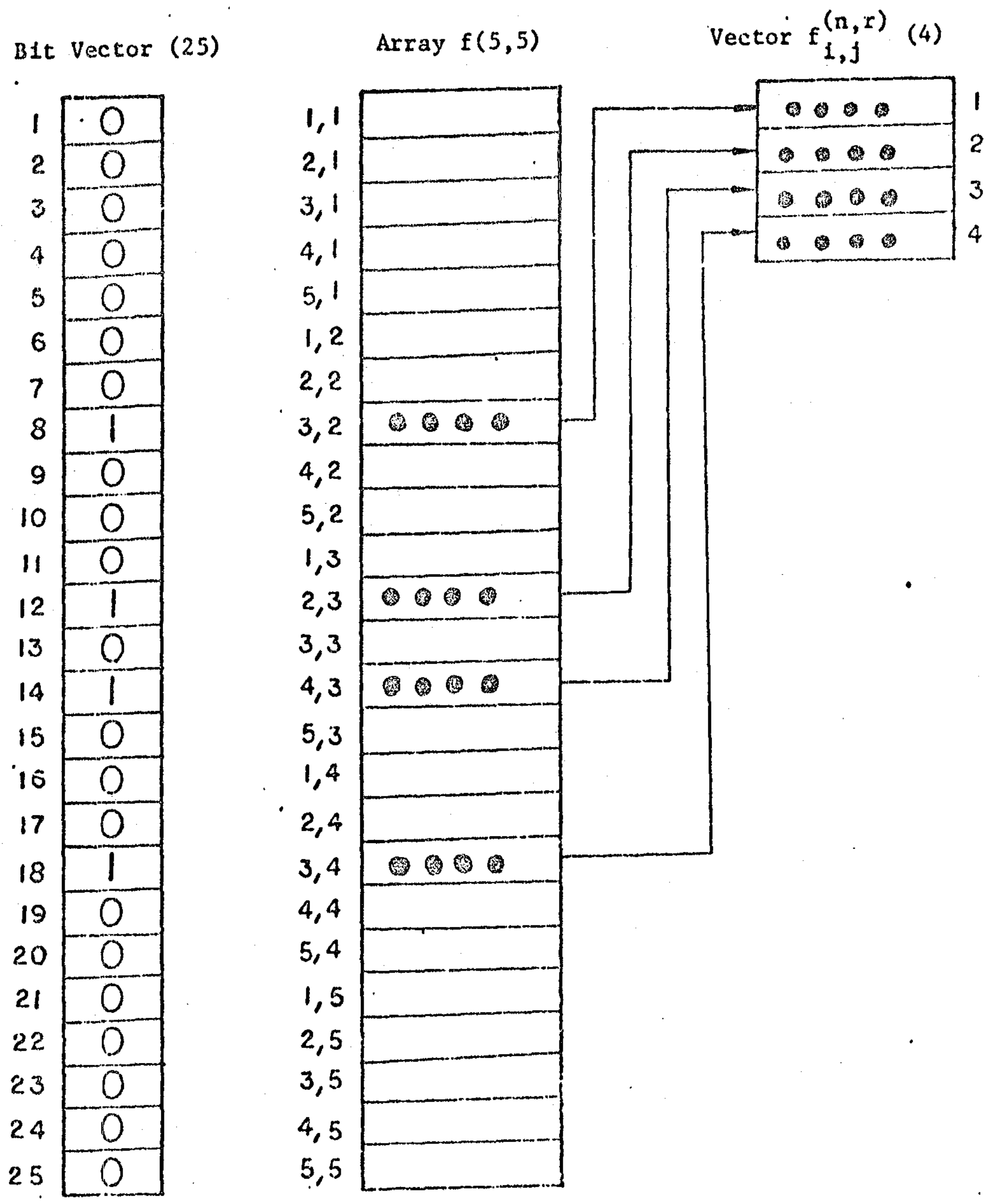


Fig. 4 Generation of $f_{i,j}^{(n,r)}$ Using the CMPRS Function

ORIGINAL PAGE IS
OF POOR QUALITY

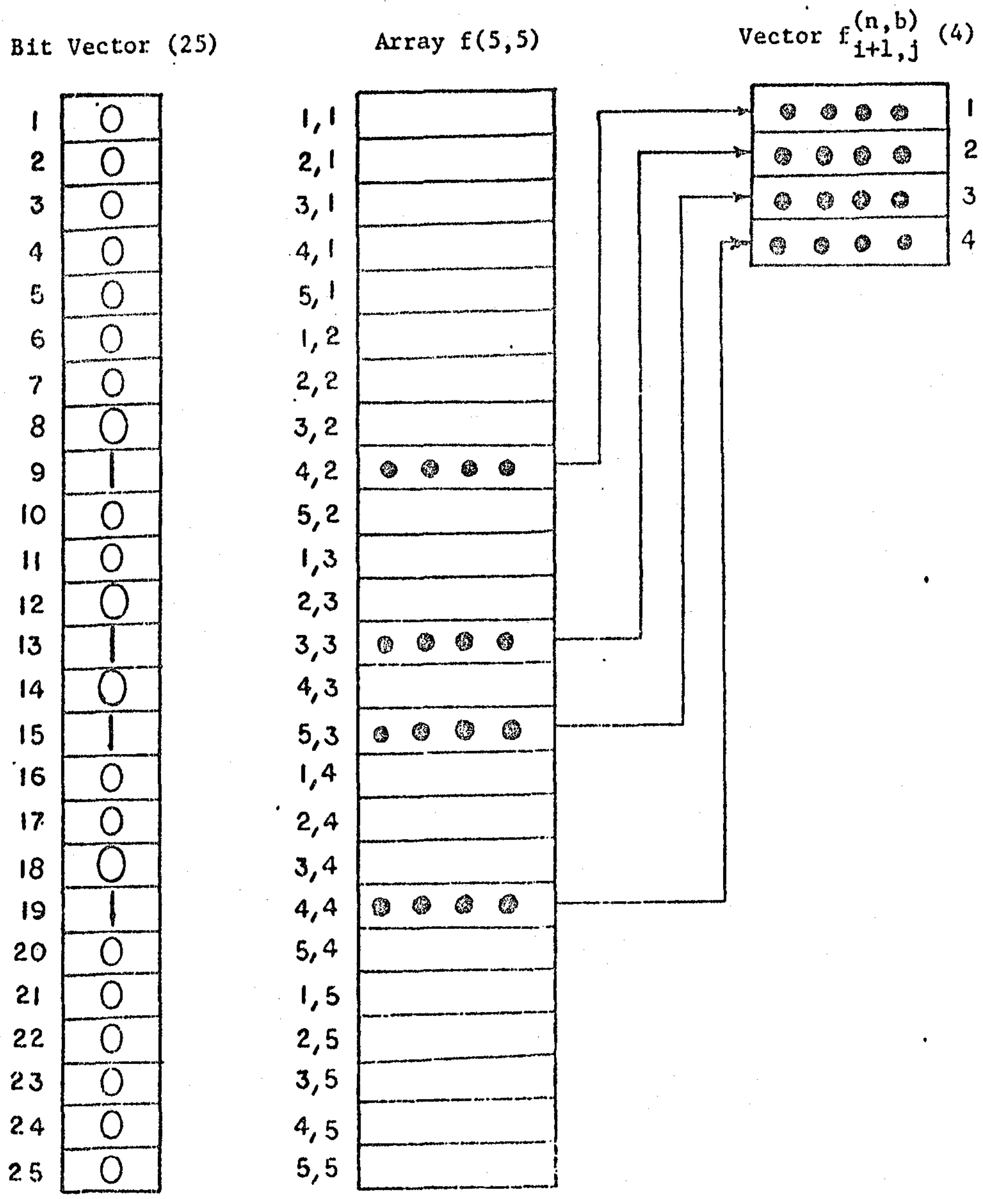


Fig. 5 Generation of $f_{i+1,j}^{(n,b)}$ Vector Using the CMPRS Function

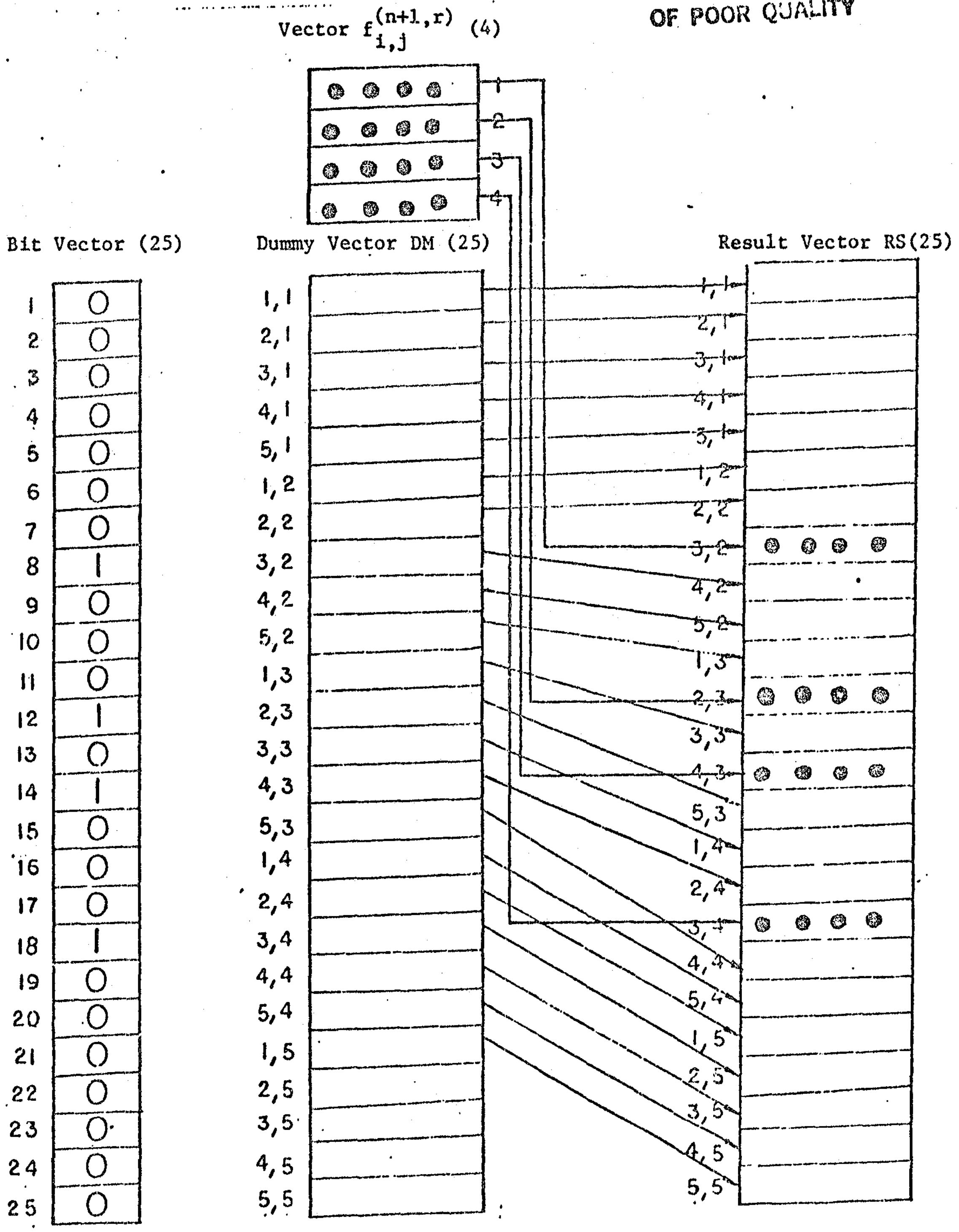


Fig. 6 Merging of Vectors $f_{i,j}^{(n+1,r)}$ and DM in vector RS Using the MERG Function

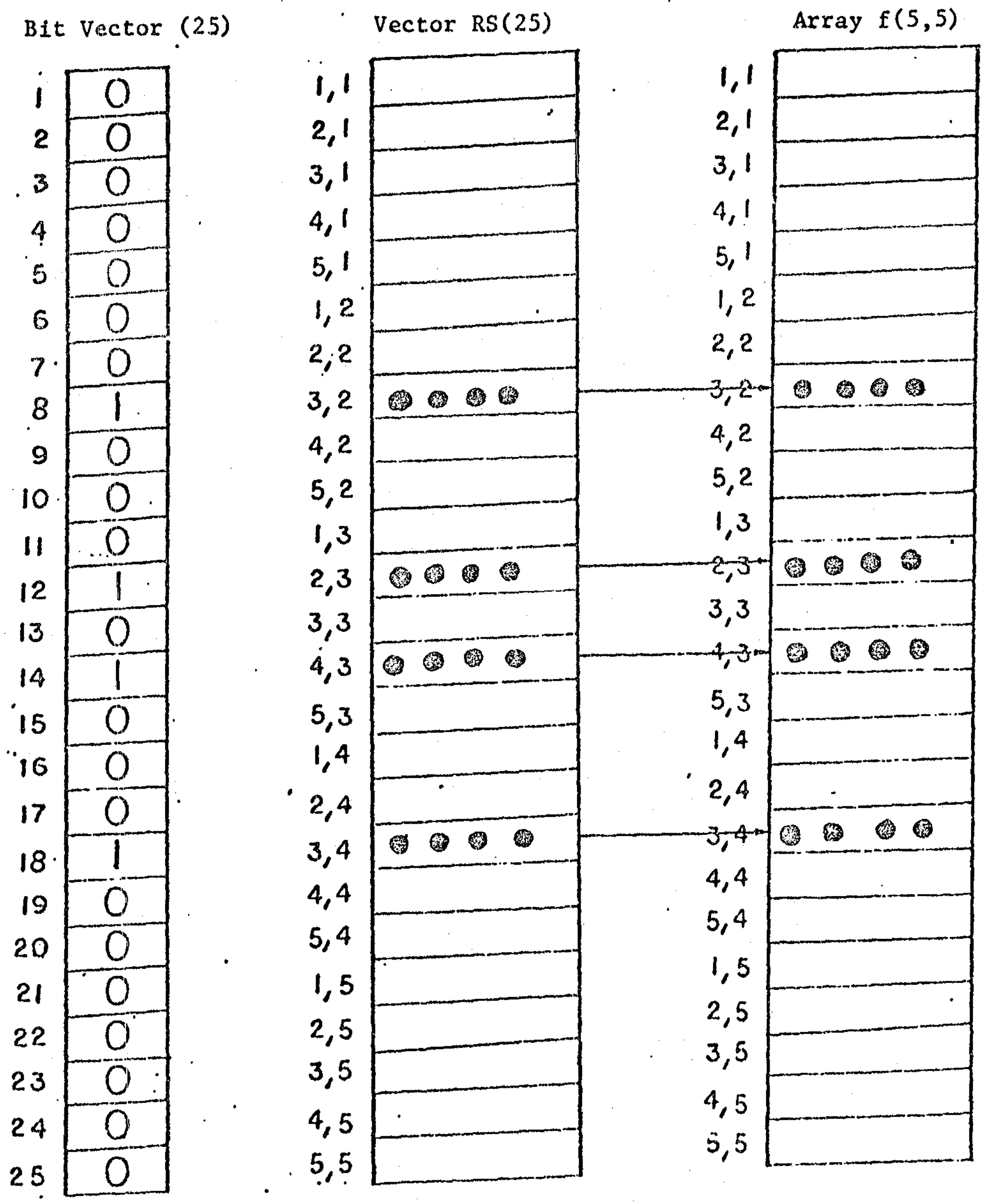
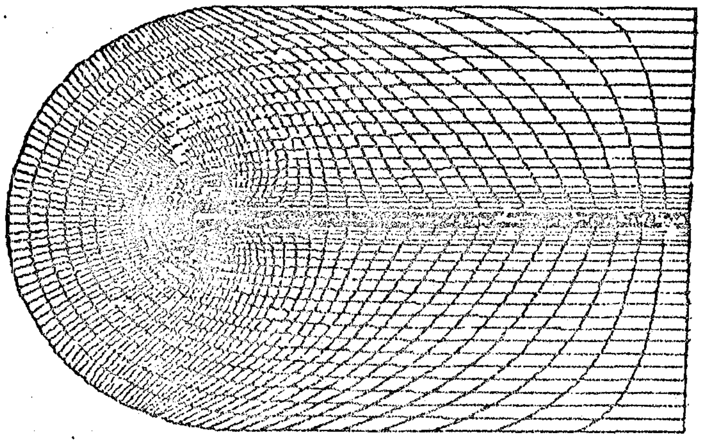
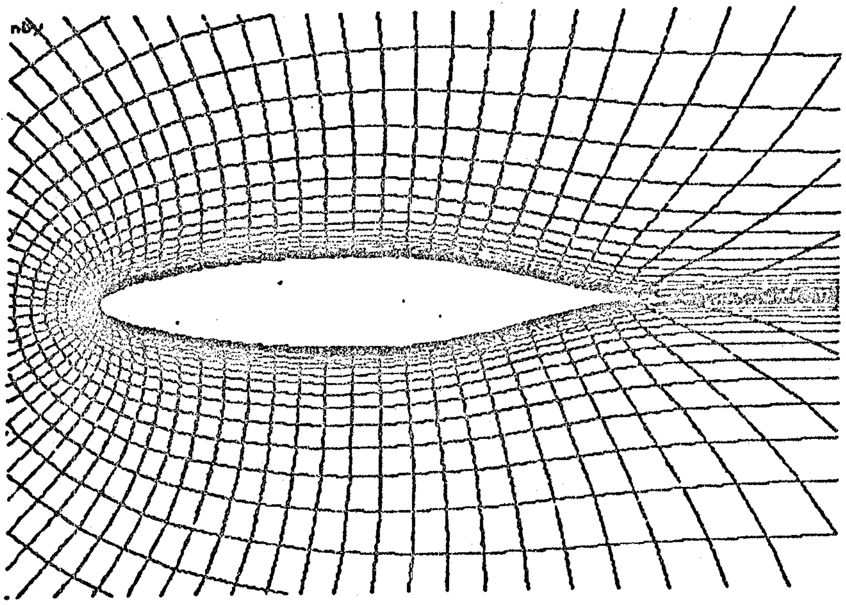


Fig. 7 Changing Values of Selected Elements Using the CTRL Function

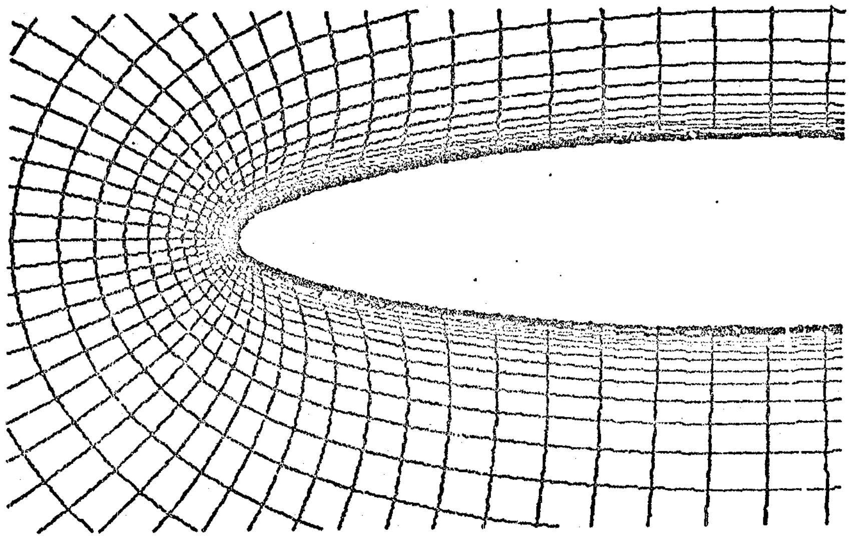


(a) Entire Grid

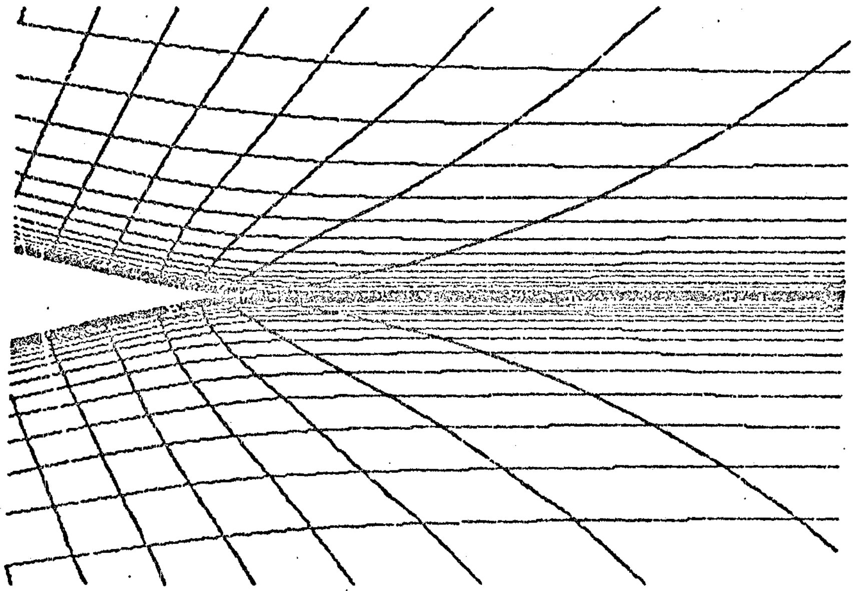


(b) Region Near Airfoil

Fig. 8 Coordinate System CORD1

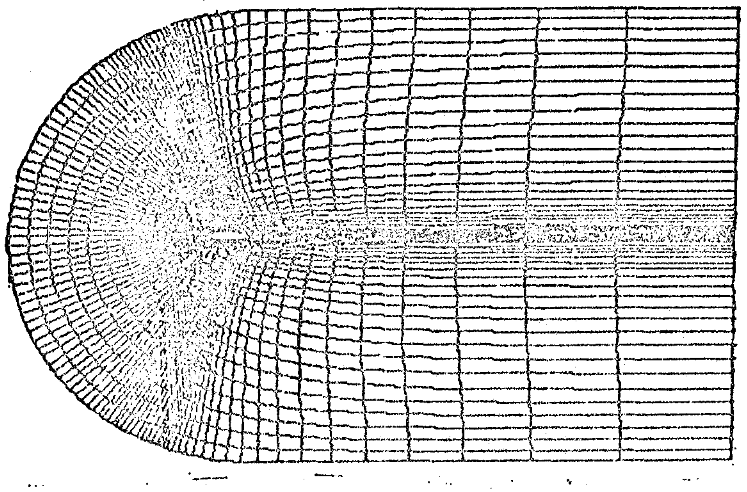


(c) Close-up of Leading Edge



(d) Close-up of Trailing Edge

Fig. 8 Coordinate System CORD1


(a) Entire Grid

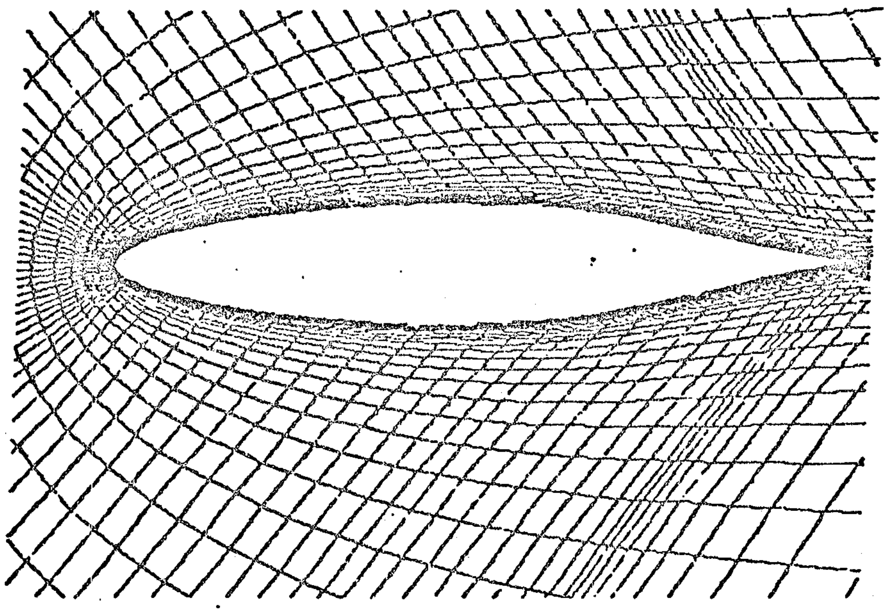
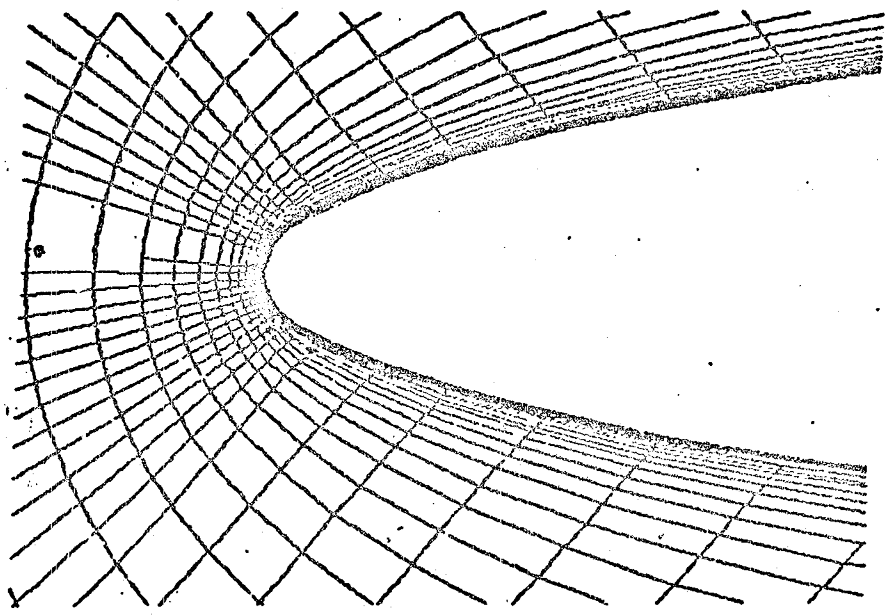
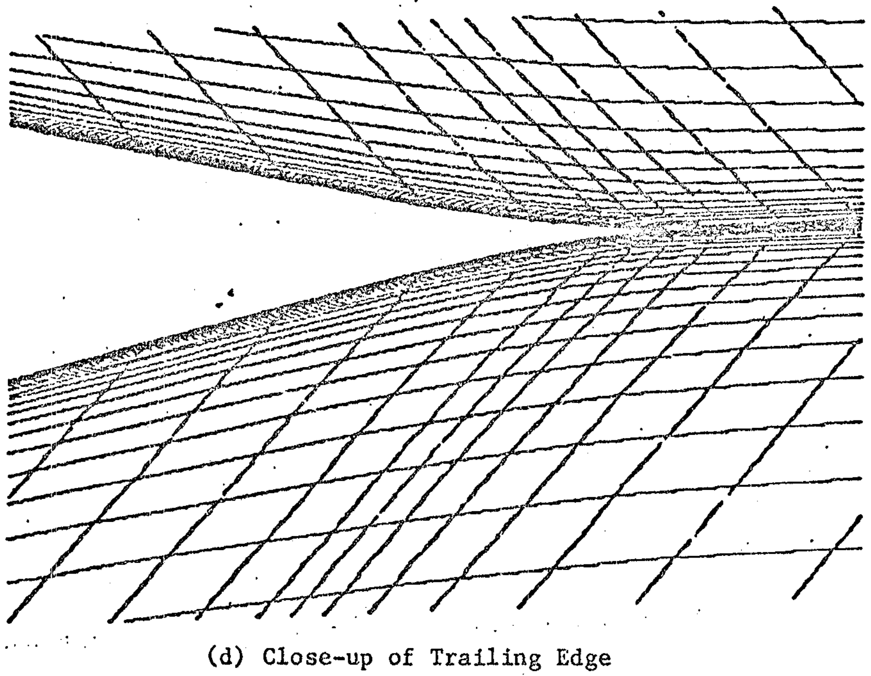

(b) Region Near Airfoil

Fig. 9 Coordinate System CORD2

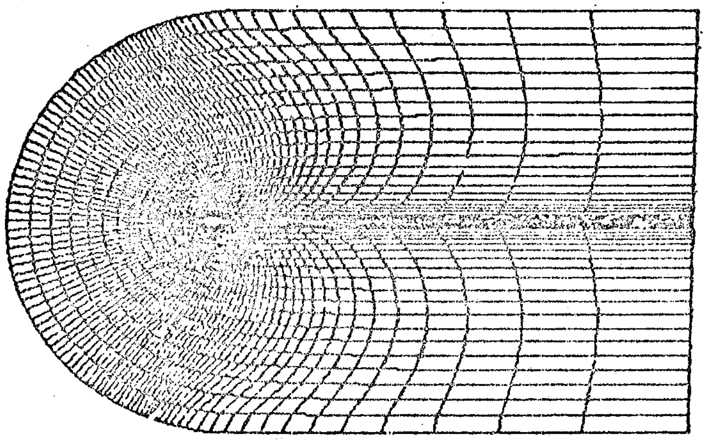


(c) Close-up of Leading Edge

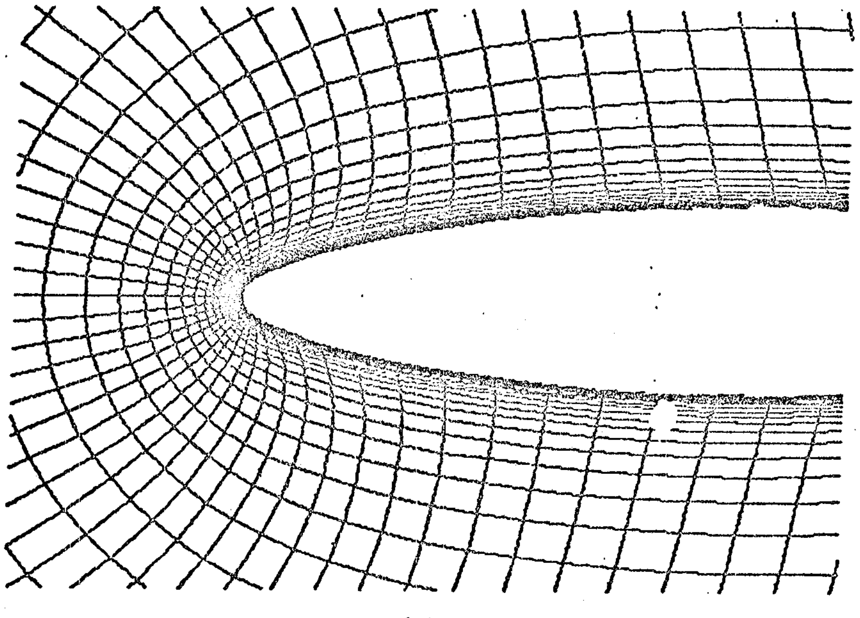


(d) Close-up of Trailing Edge

Fig. 9 Coordinate System CORD2

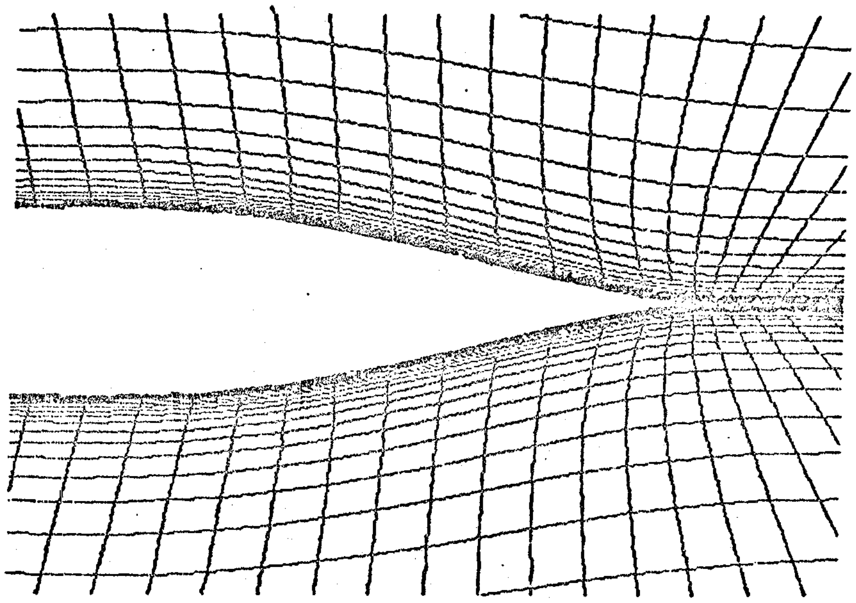


(a) Entire Grid

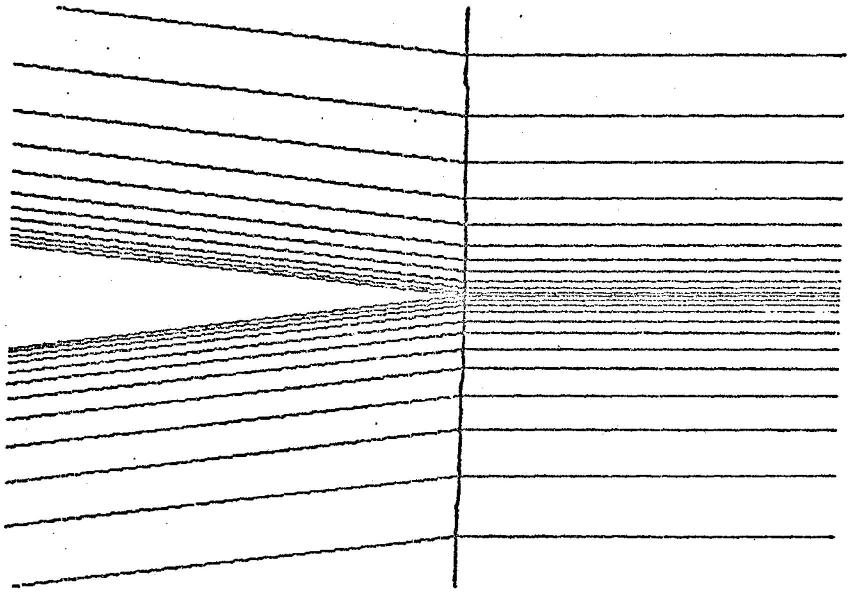


(b) Close-up of Leading Edge

Fig. 10 Coordinate System CORD 3



(c) Close-up of Trailing Edge



(d) Very Close View of Trailing Edge

Fig. 10 Coordinate System CORD3

ORIGINAL PAGE IS
OF POOR QUALITY

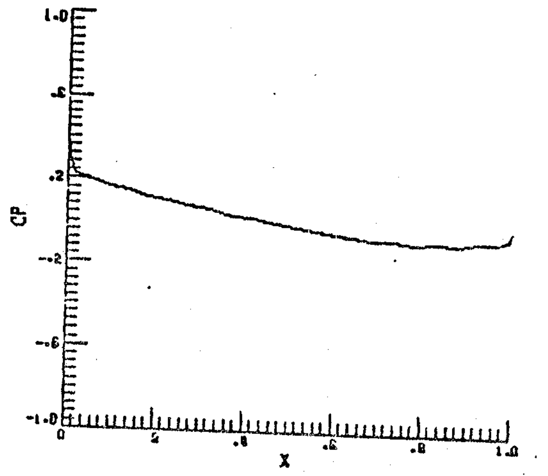


Fig. 11 Surface Pressure Distribution
Laminar Flow, t = 0.5, Re = 10,000

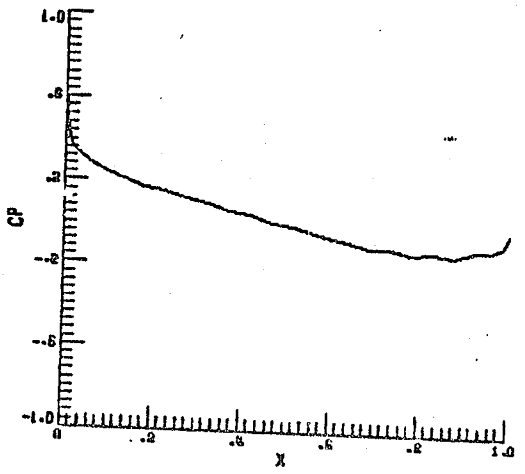


Fig. 12 Surface Pressure Distribution
Laminar Flow, t = 0.7, Re = 10,000

ORIGINAL PAGE IS
OF POOR QUALITY

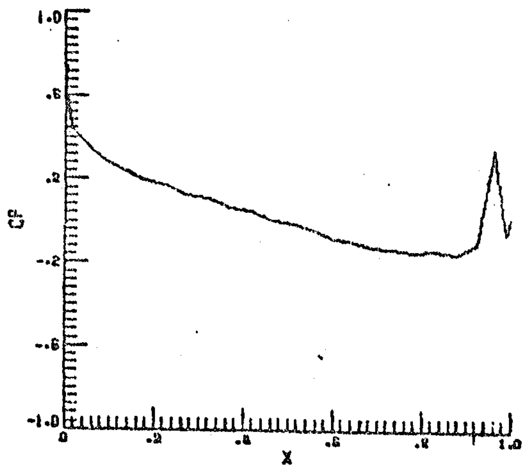


Fig. 13 Surface Pressure Distribution
Turbulent flow, t = 0.8, Re = 10,000

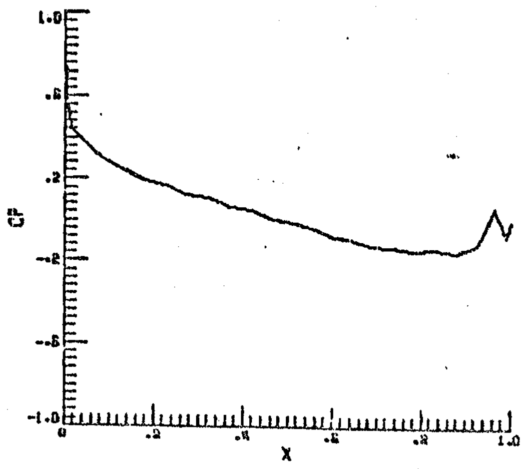


Fig. 14 Surface Pressure Distribution
Turbulent flow, t = 0.8, Re = 10,000
Transition at Maximum Airfoil Thickness
Points.

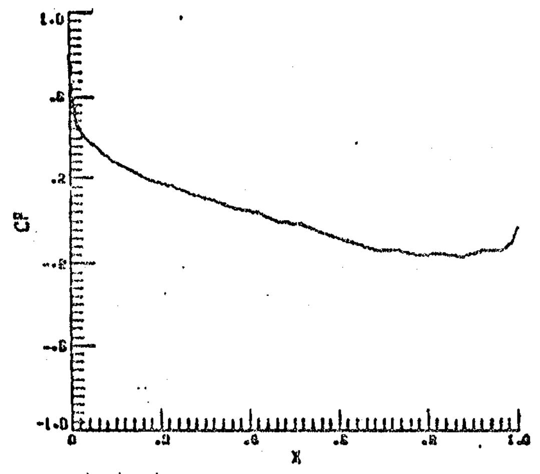


Fig. 15 Surface Pressure Distribution
Turbulent flow, $t = 0.8$, $Re = 10,000$
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$

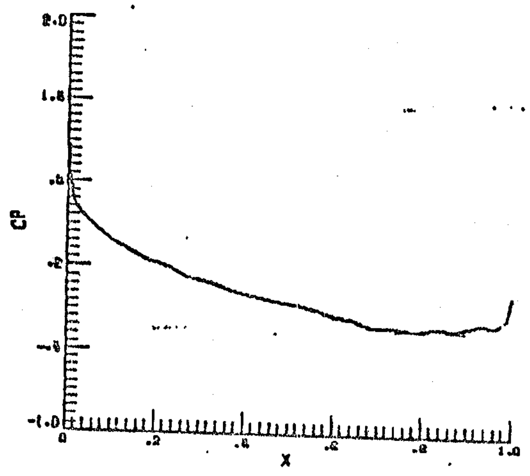


Fig. 16 Surface Pressure Distribution
Turbulent flow, $t = 1.0$, $Re = 10,000$
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$

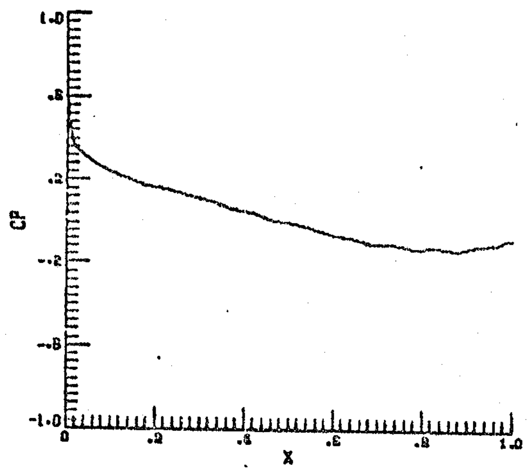


Fig. 17 Surface Pressure Distribution
Turbulent flow, t = 0.7, Re = 10,000
Extrapolated trailing edge pressure.

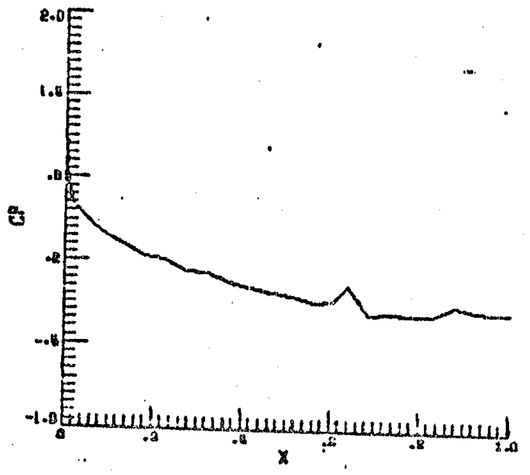


Fig. 18 Surface Pressure Distribution
Turbulent flow, t = 0.7, Re = 10,000
Extrapolated trailing edge pressure
$\mu_a = \Omega \mathrm{Re} J |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = \varepsilon$

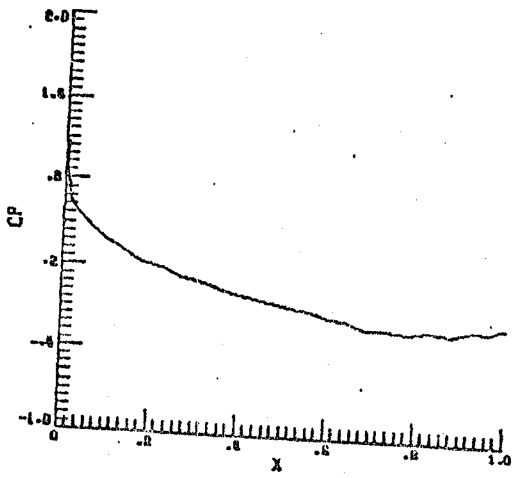


Fig. 19 Surface Pressure Distribution
Laminar Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$
μ_a applied every iteration

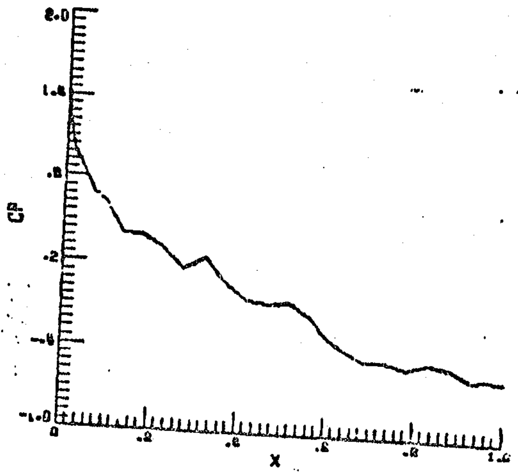


Fig. 20 Surface Pressure Distribution
Laminar Flow, t = 2.0, Re = 10,000
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$
μ_a applied every iteration

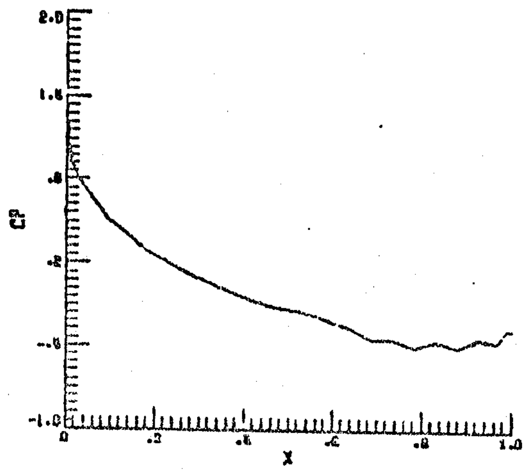


Fig. 21 Surface Pressure Distribution
Turbulent flow, t = 1.6 Re = 10,000
$\mu_a = \Omega \mathrm{Re} J \sqrt{|\underset{\sim}{\omega}|\ |\nabla \cdot \underset{\sim}{V}|}$, $\Omega = 1.0$

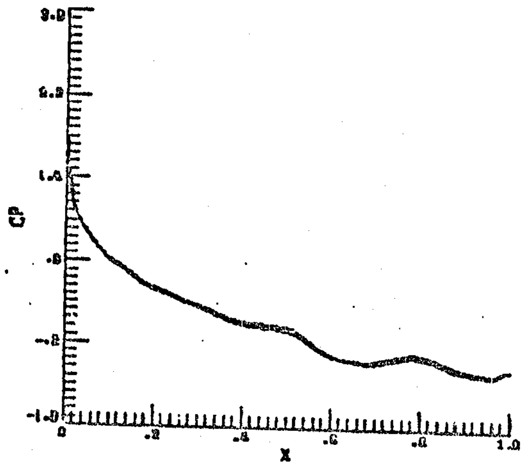
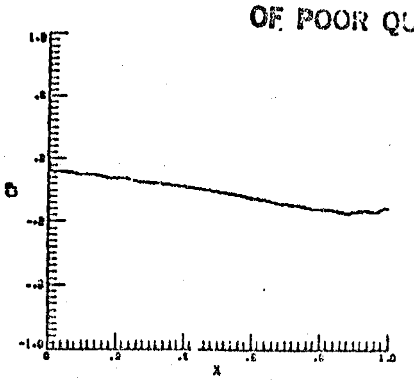


Fig. 22 Surface Pressure Distribution
Turbulent flow, t = 2.0, Re = 10,000
$\mu_a = \Omega \mathrm{Re} J \sqrt{|\underset{\sim}{\omega}|\ |\nabla \cdot \underset{\sim}{V}|}$, $\Omega = 1.0$

(a)



(b)

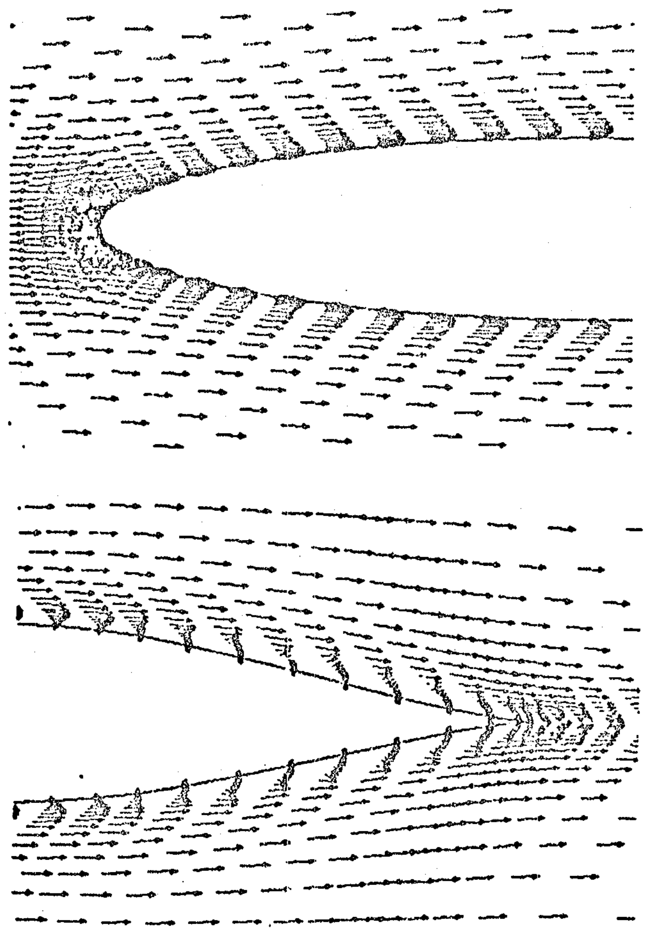
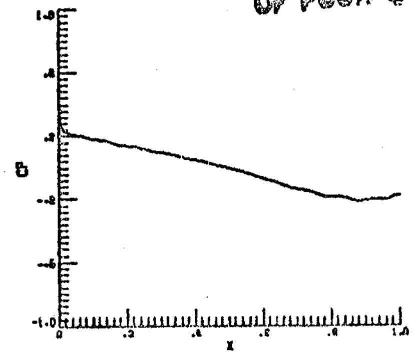


Fig. 23 (a) Surface Pressure Distribution
(b) Leading & Trailing Edge Velocity Vector Fields
Laminar Flow, t = 0.5, Re = 10,000

(a)



(b)

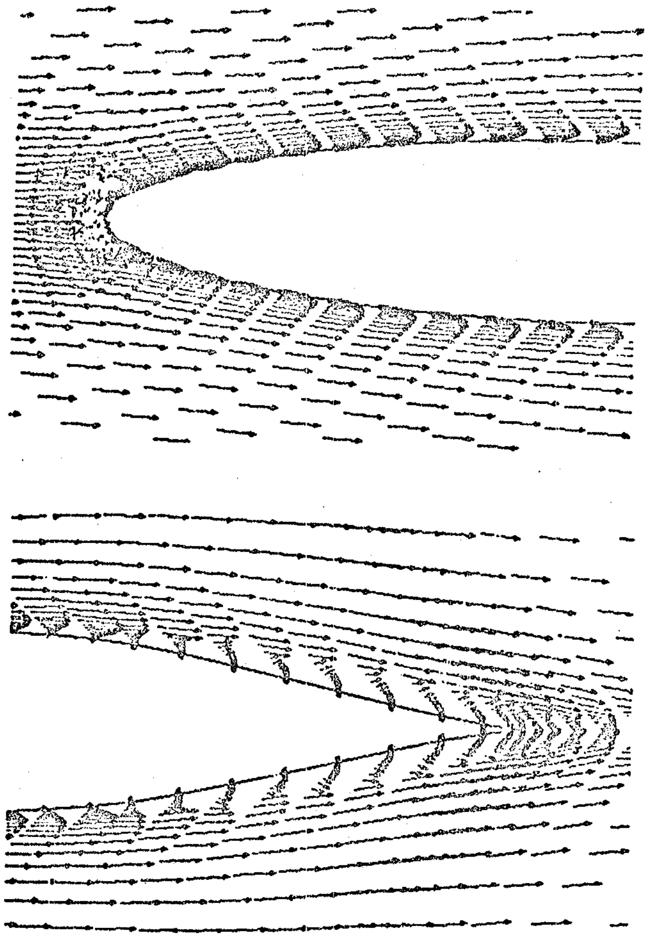
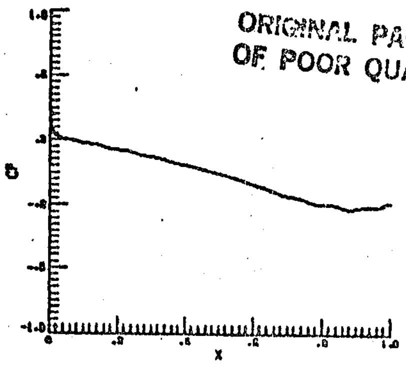


Fig. 24 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
laminar Flow, t = 0.7, Re = 10,000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

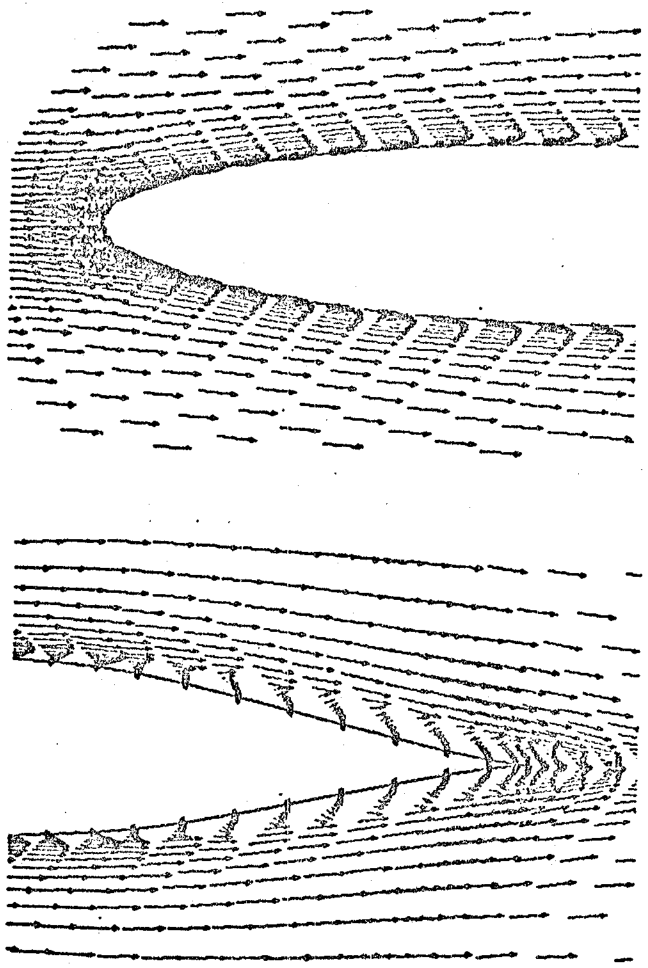
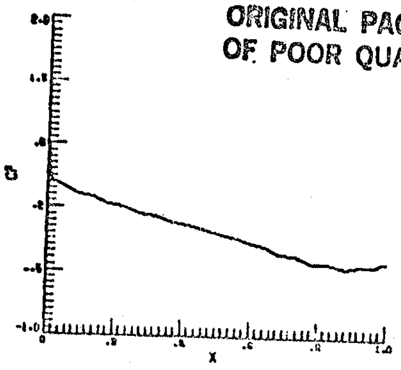


Fig. 25 . (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 0.7, Re = 10,000

(a)



(b)

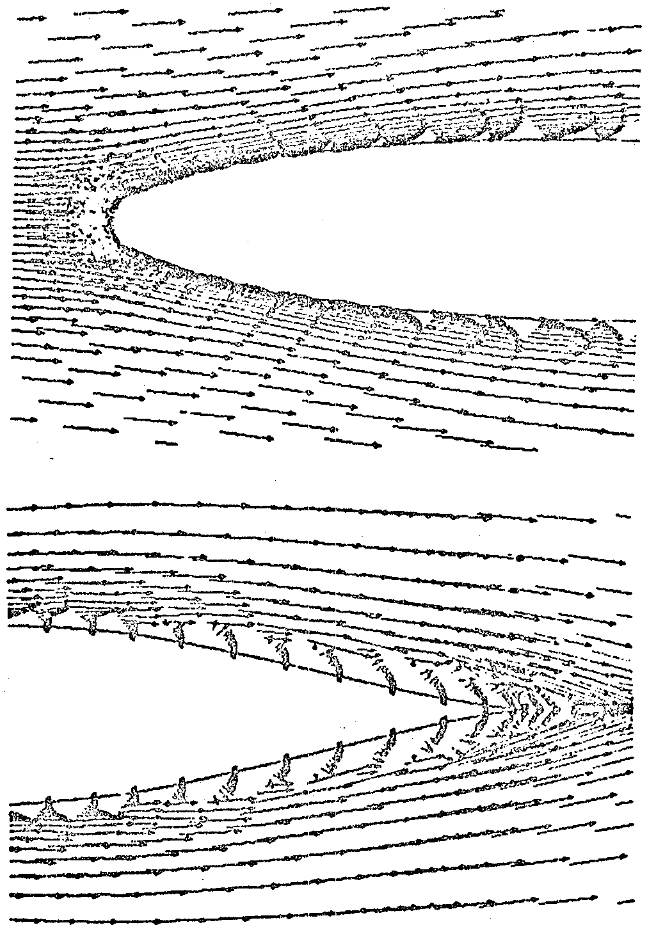
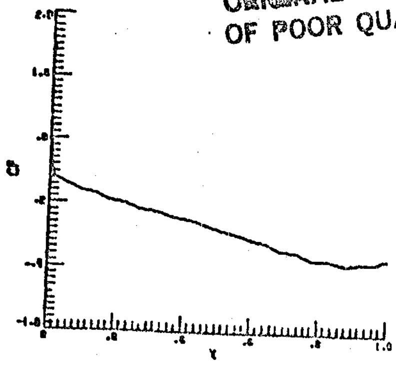


Fig. 26 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000

(a)



(b)

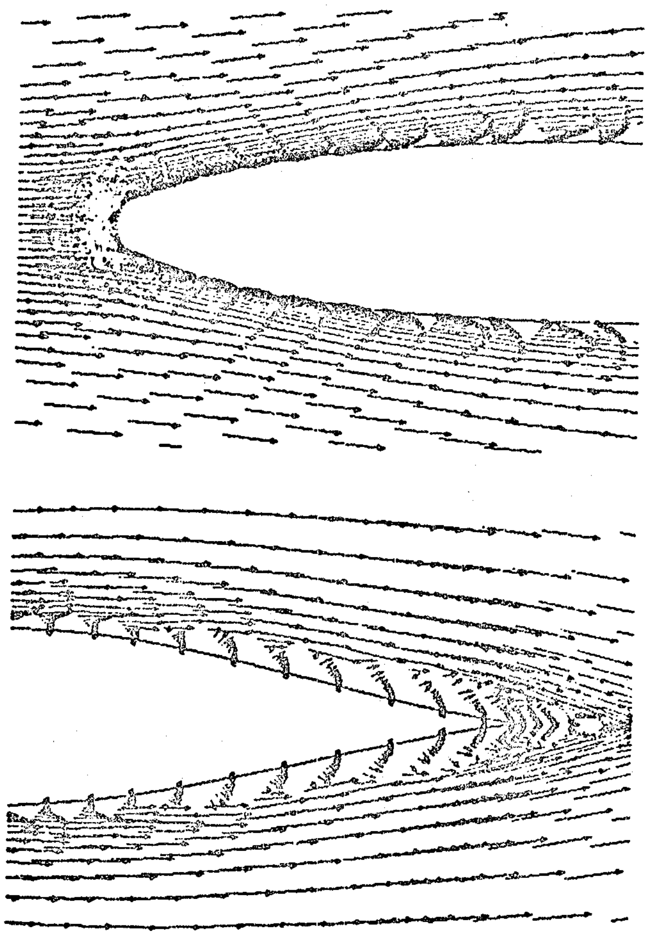
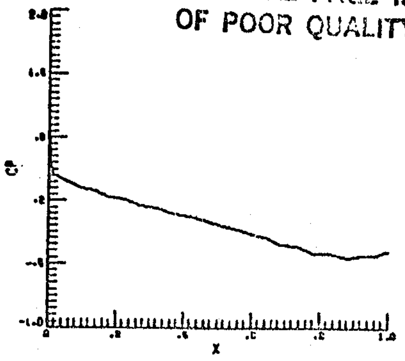


Fig. 27 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
Transition at Maximum Airfoil Thickness Points

(a)



(b)

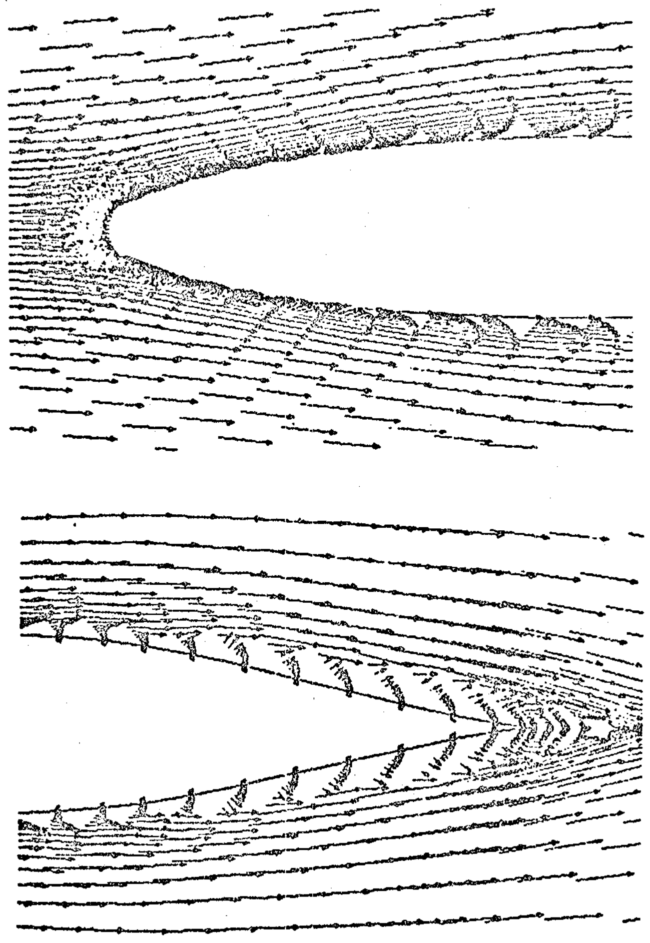
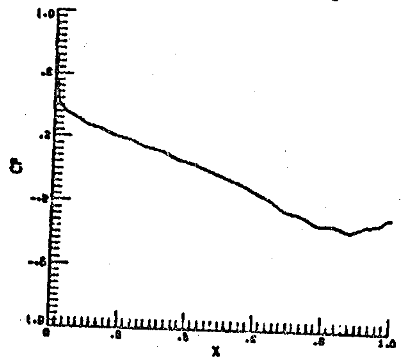


Fig. 28 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
Transition at Maximum Airfoil Thickness Points
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = \varepsilon$

(a)



(b)

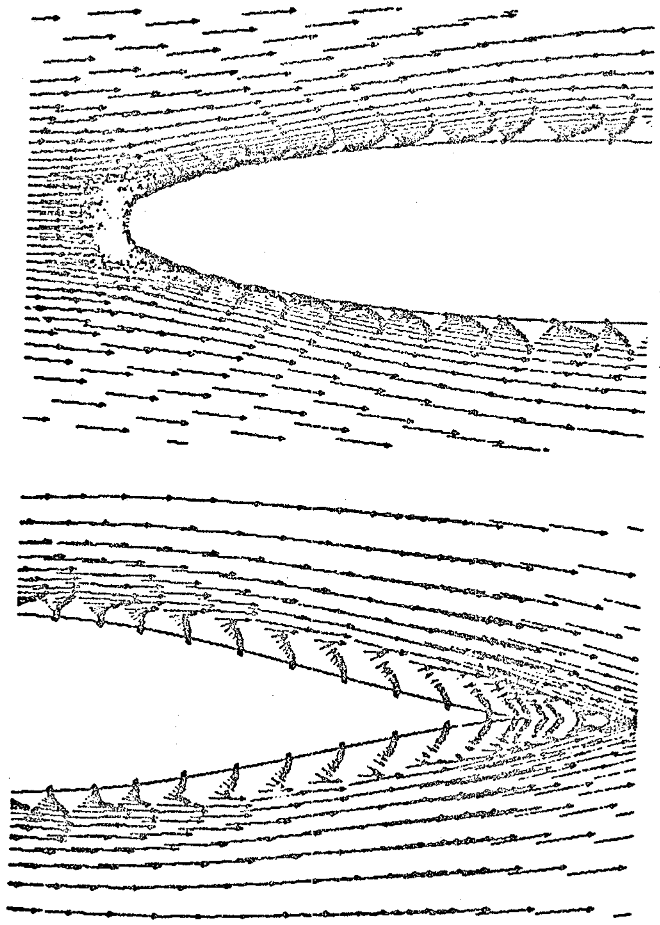
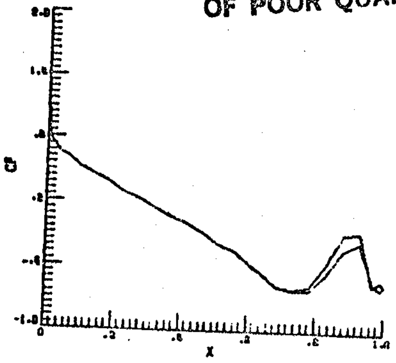


Fig. 29 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$

(a)



(b)

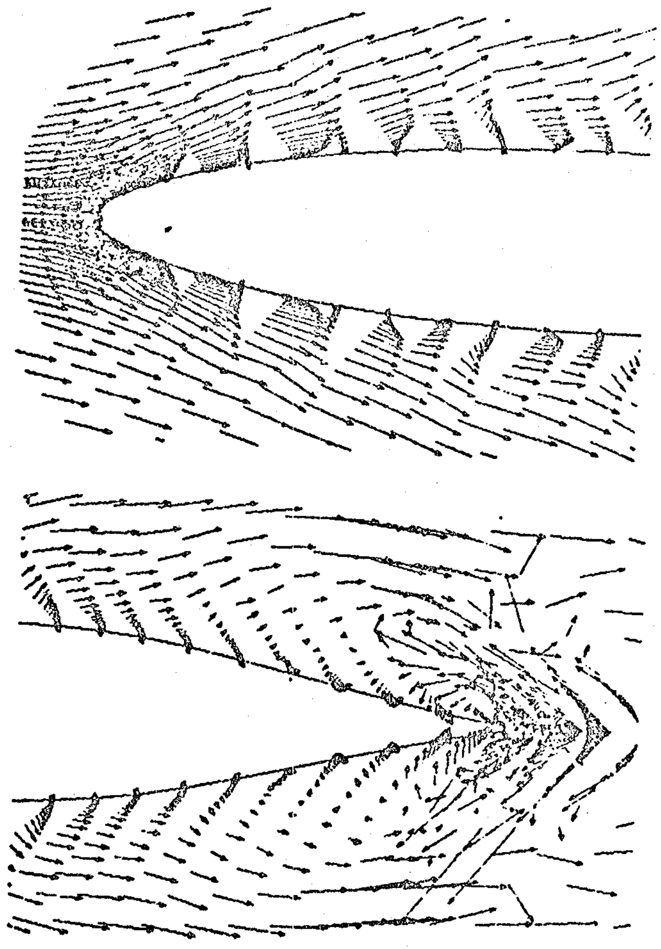
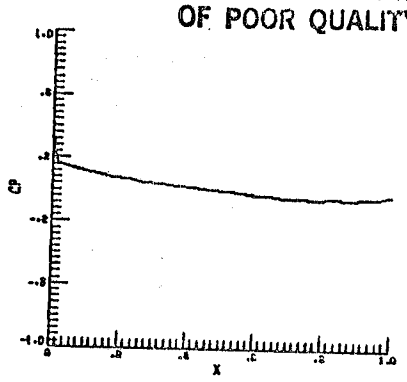


Fig. 30 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 2.0, Re = 10,000
$\mu_a = \Omega ReJ|\nabla \cdot V|$, $\Omega = 1.0$

(a)



(b)

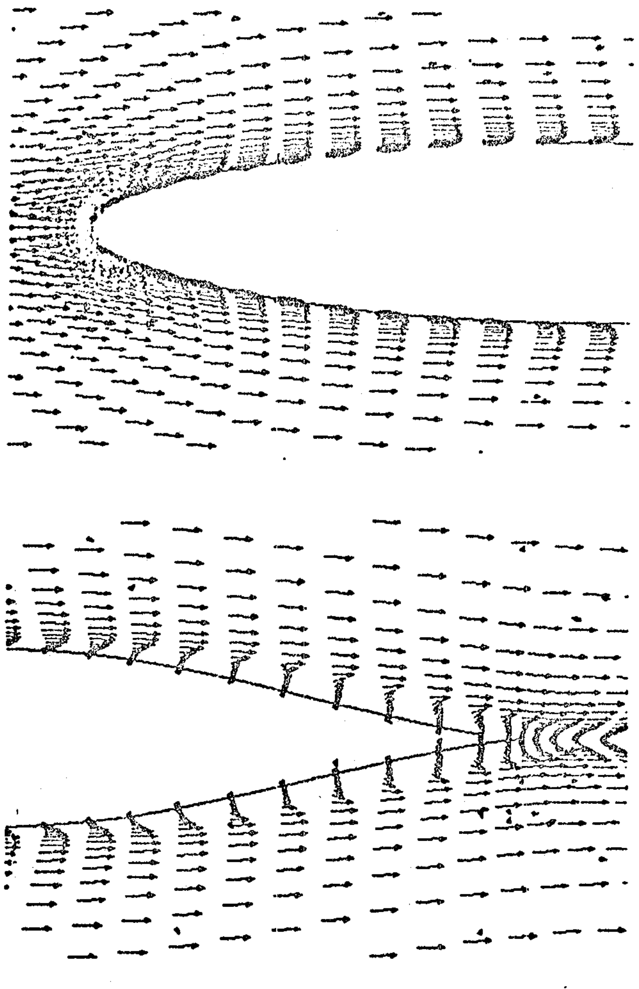
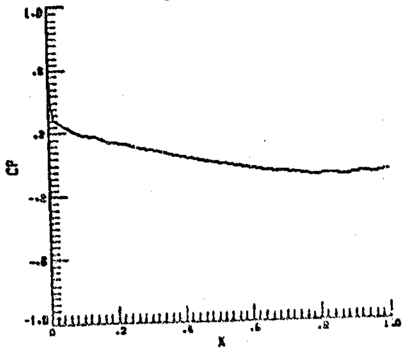


Fig. 31 (a) Surface Pressure Distributin
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 0.5, Re = 10,000

(a)



(b)

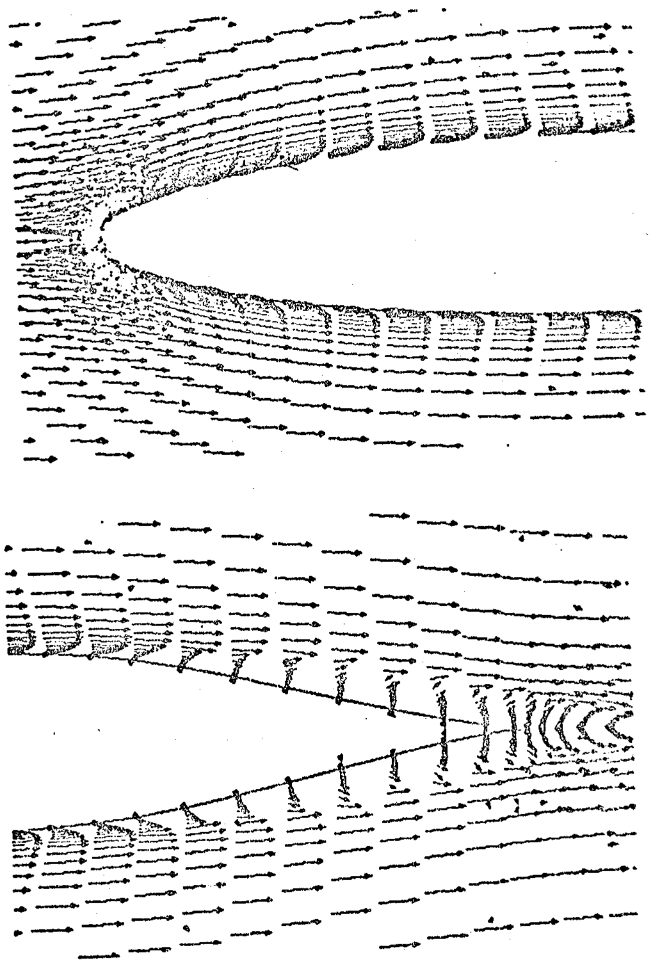
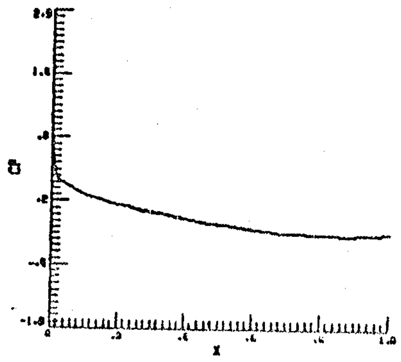


Fig. 32 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar flow, t = 0.7, Re = 10,000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

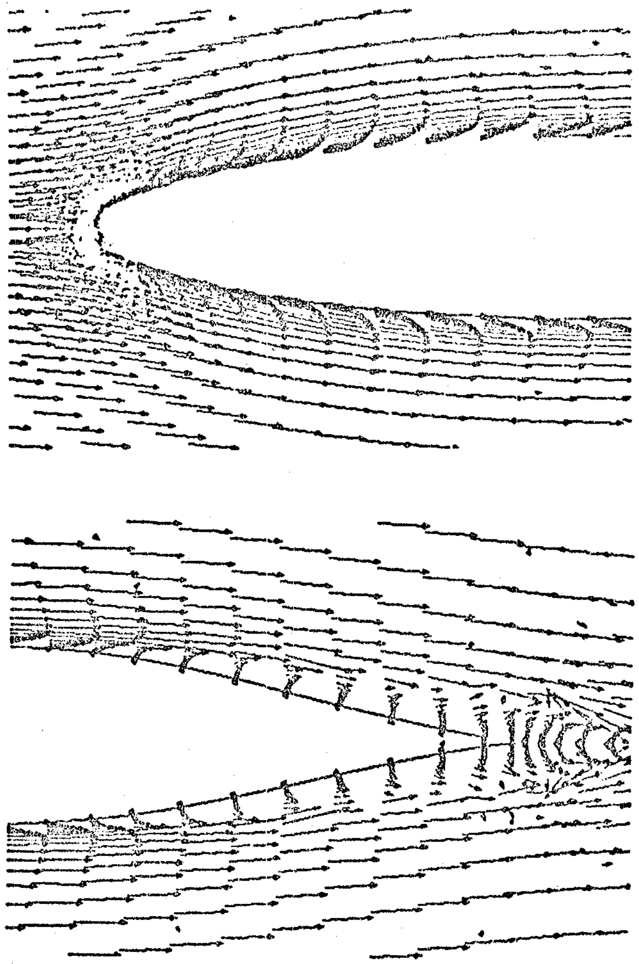
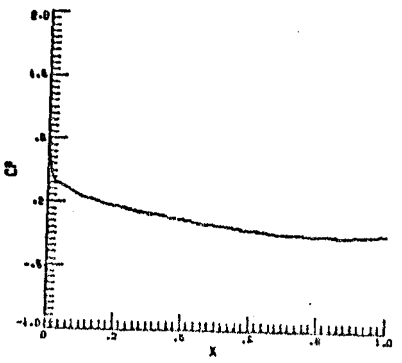


Fig. 33 (a) Surface Pressure Distribution
(b) Leading & Trailing Edge Velocity Vector Fields
Turbulent flow, t = 0.9, Re = 10,000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

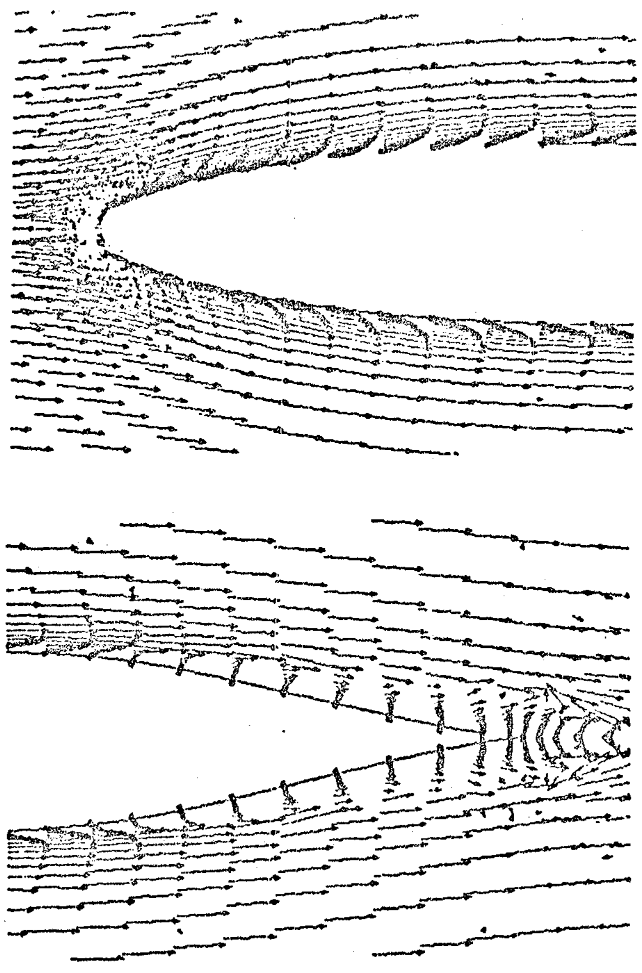
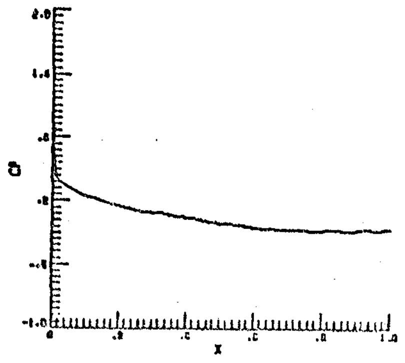
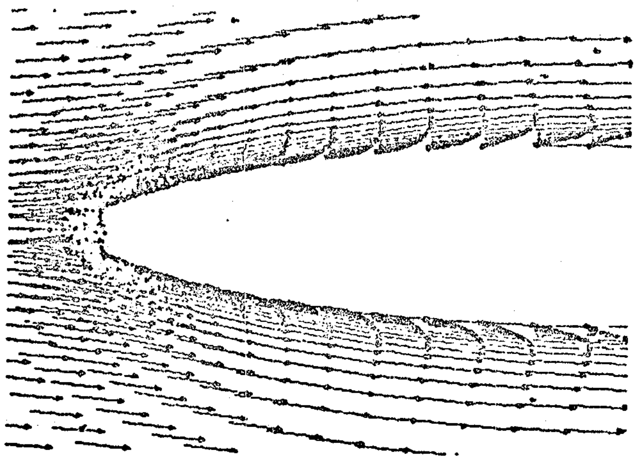


Fig. 34 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent flow, t = 0.9, Re = 10,000
Transition at Maximum Airfoil Thickness Points.

(a)





(b)

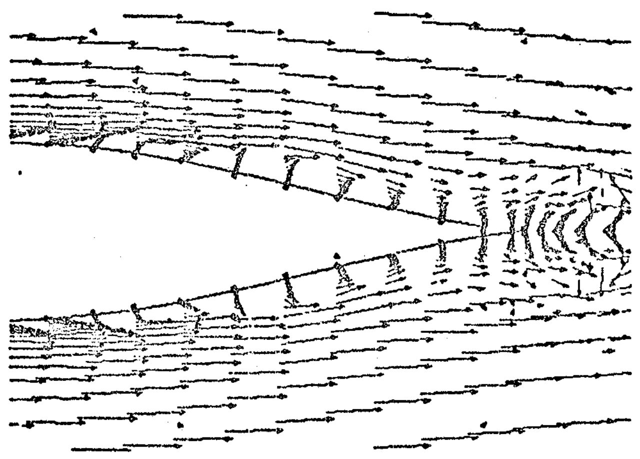
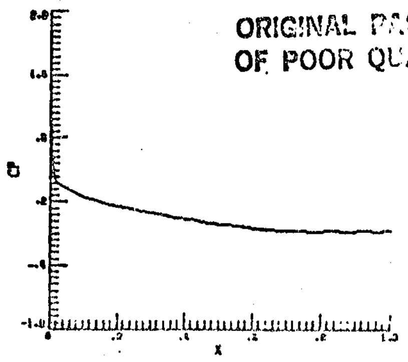


Fig. 35 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
Divergence of Velocity Smoothers,
$\frac{\partial D}{\partial t}$ 1st Order Accurate

(a)



(b)

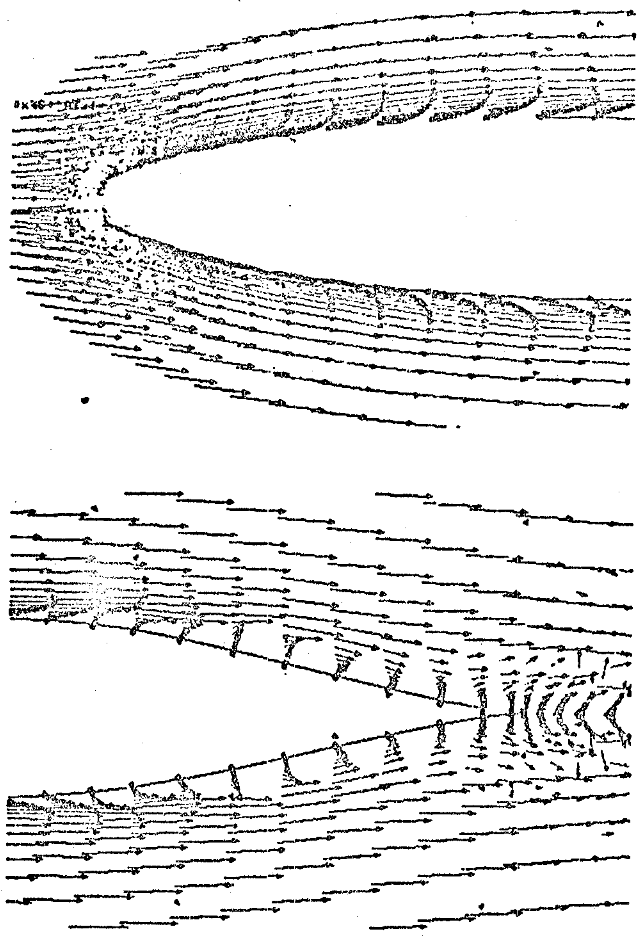
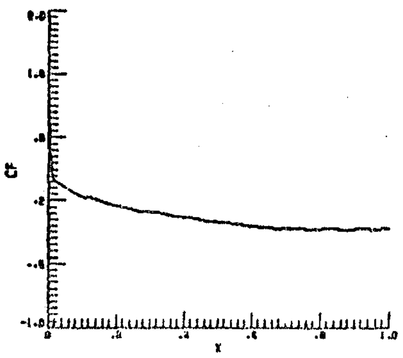
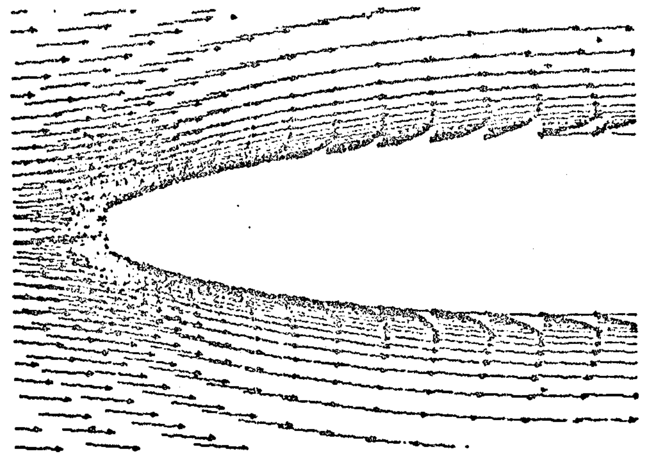


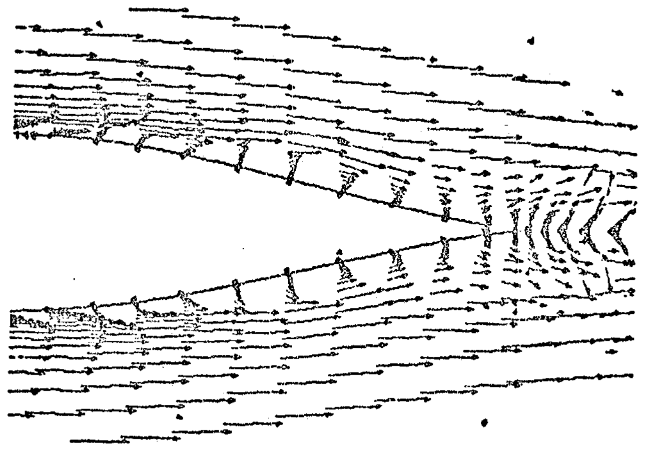
Fig. 36 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
RHS Smoother

(a)





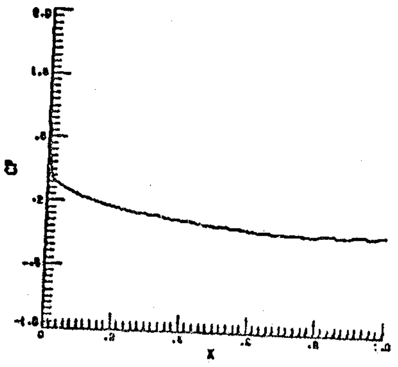
(b)



Fig, 37 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, $t = 1.0$, $Re = 10,000$
$\mu_a = ReJ|\nabla \cdot \underset{\sim}{V}|$, $\Omega = \{(\partial V^2/\partial \xi)^2 + (\partial V^2/\partial \eta)^2\}^{\frac{1}{2}}$

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

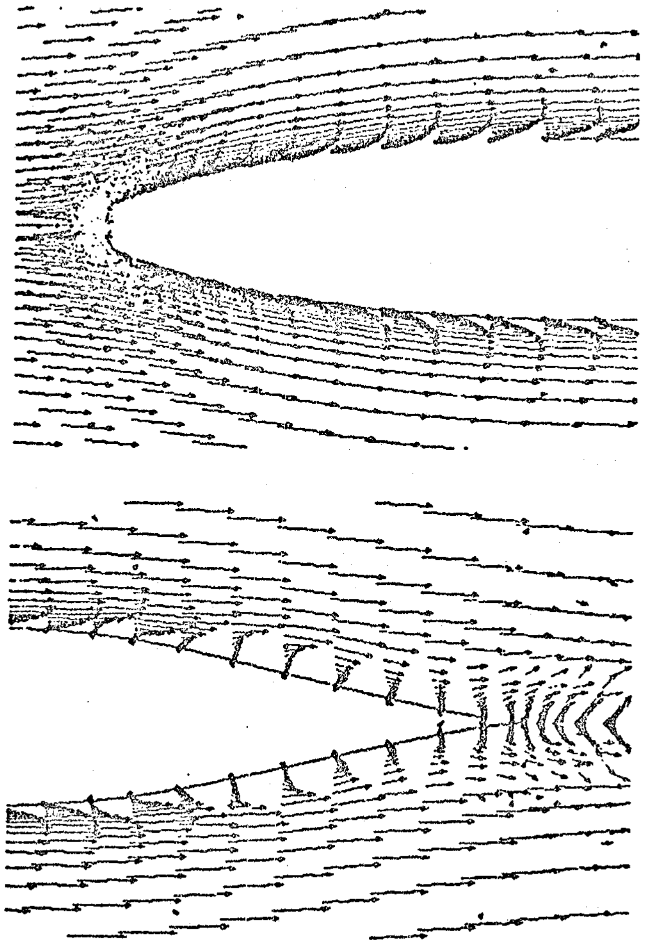
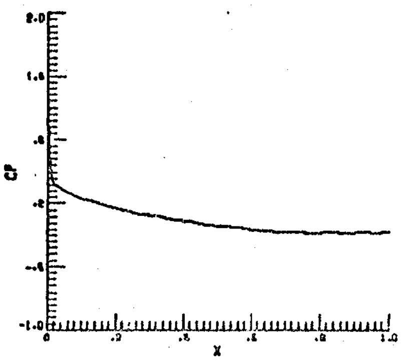


Fig. 38 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, $t = 1.0$, $Re = 10{,}000$
$\mu_a = \Omega ReJ\sqrt{|\omega|\ |\underset{\sim}{\nabla}\cdot\underset{\sim}{V}|}$, $\Omega = \{(\partial V^2/\partial\xi)^2 + (\partial V^2/\partial\eta)^2\}^{\frac{1}{2}}$

(a)



(b)

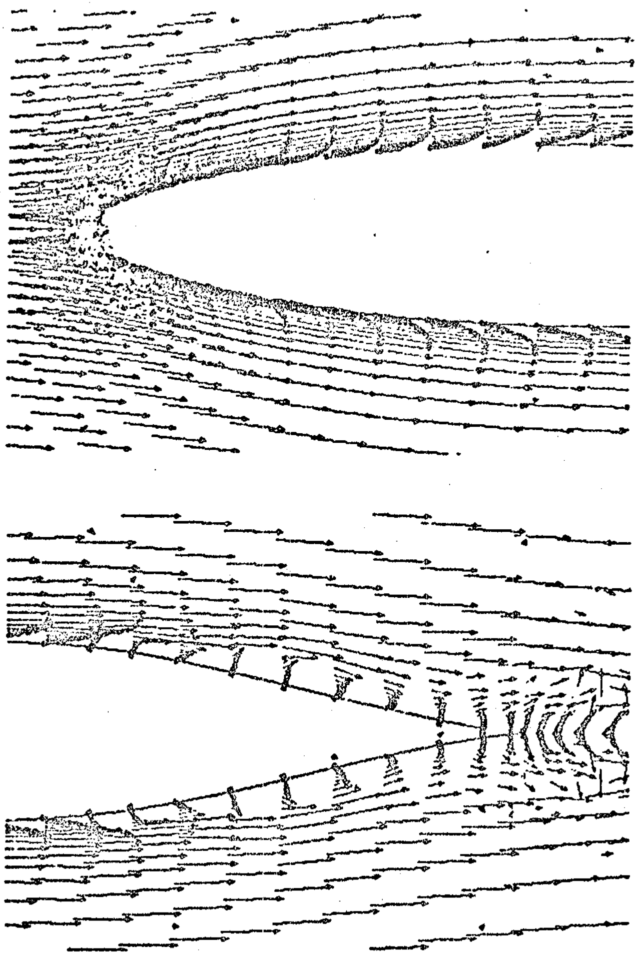
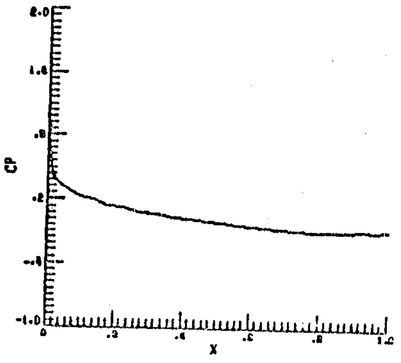


Fig. 39 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega \mathrm{Re} J |\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = \varepsilon|\underset{\sim}{\omega}|$

(a)



(b)

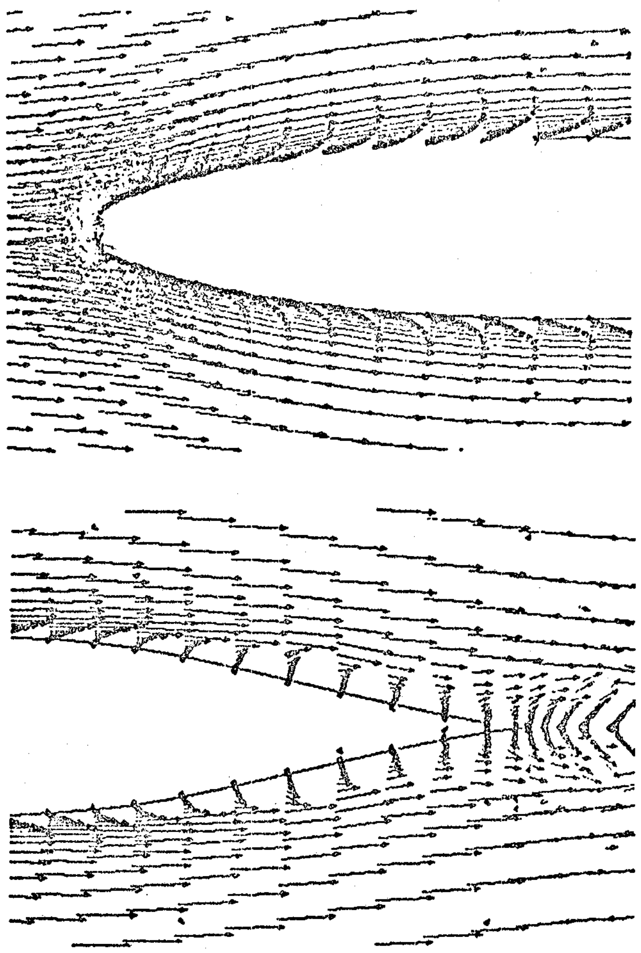
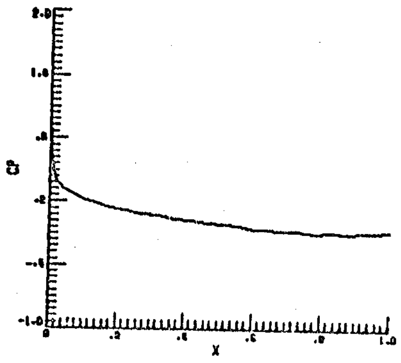


Fig. 40 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega \mathrm{ReJ}|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = 1.0$

(a)



(b)

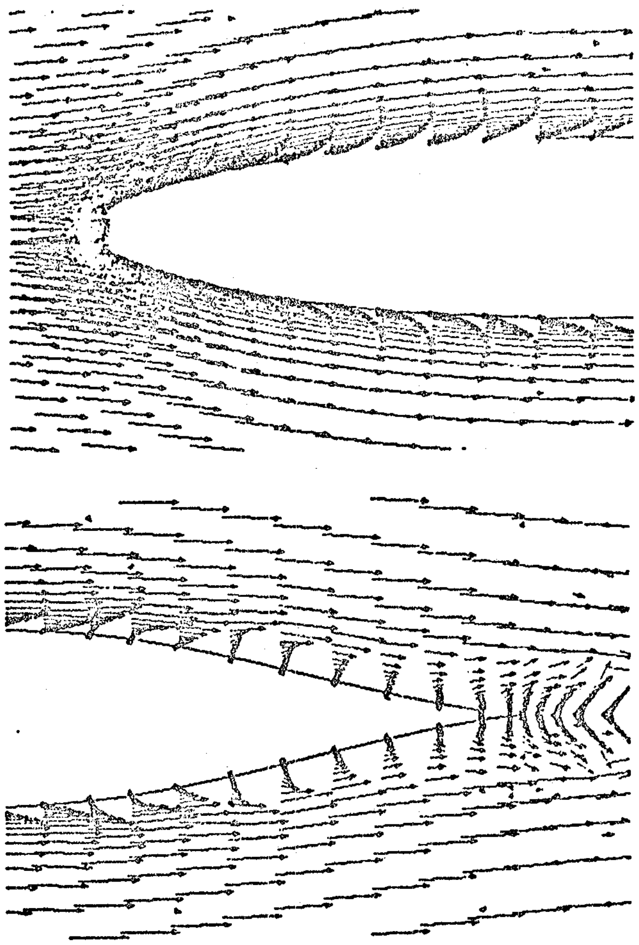
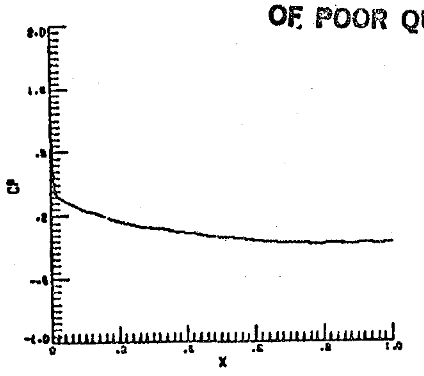


Fig. 41 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega ReJ\Delta t|\nabla^2 P|$, $\Omega = 1.0$

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

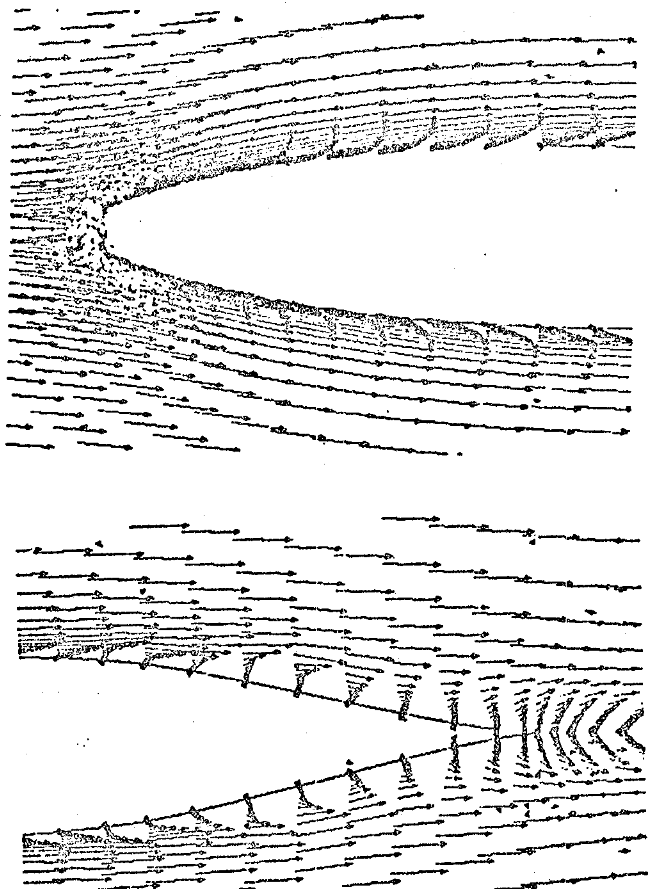
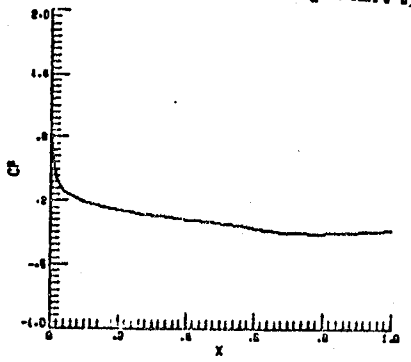


Fig. 42 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent flow, t = 1.0, Re = 10,000
Transition at Maximum Airfoil Thickness Points
$\mu_a = \Omega ReJ|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = [\ell^{\phi\varepsilon} - 1.0]|\underset{\sim}{\omega}|$, $\phi = 4.0$

(a)



(b)

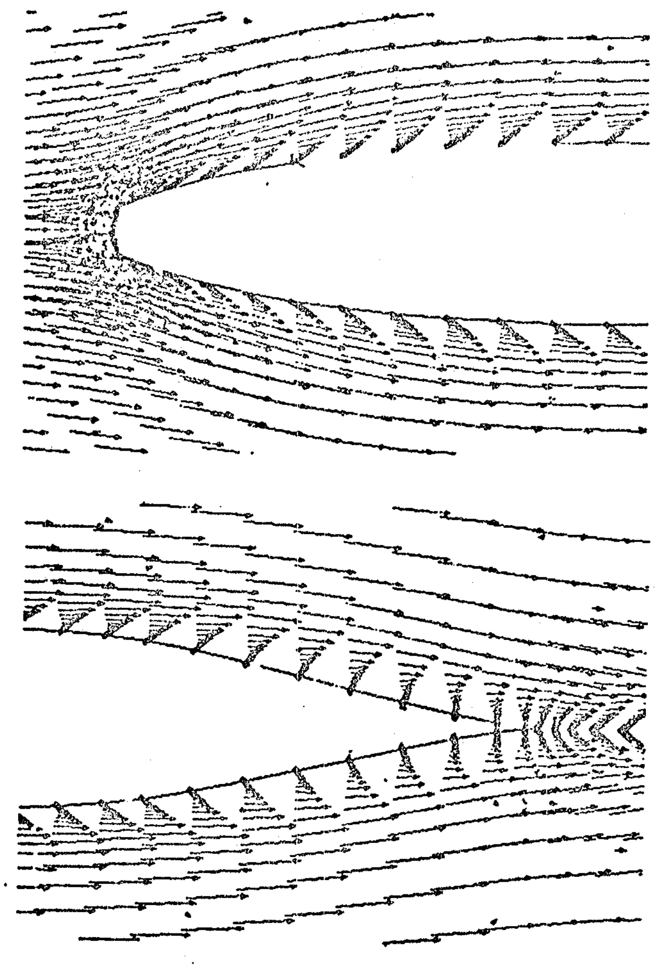
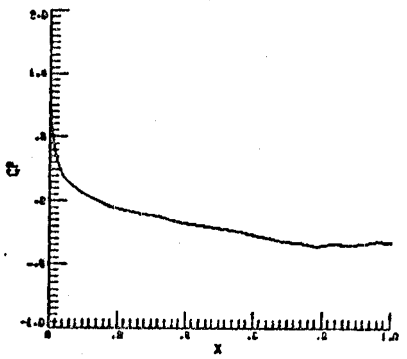


Fig. 43 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 1.0, Re = 10,000
$\mu_a = \Omega \mathrm{ReJ} |\nabla \cdot \underset{\sim}{V}|$, $\Omega = \phi|\underset{\sim}{\omega}|$, $\phi = 1.0$

(a)



(b)

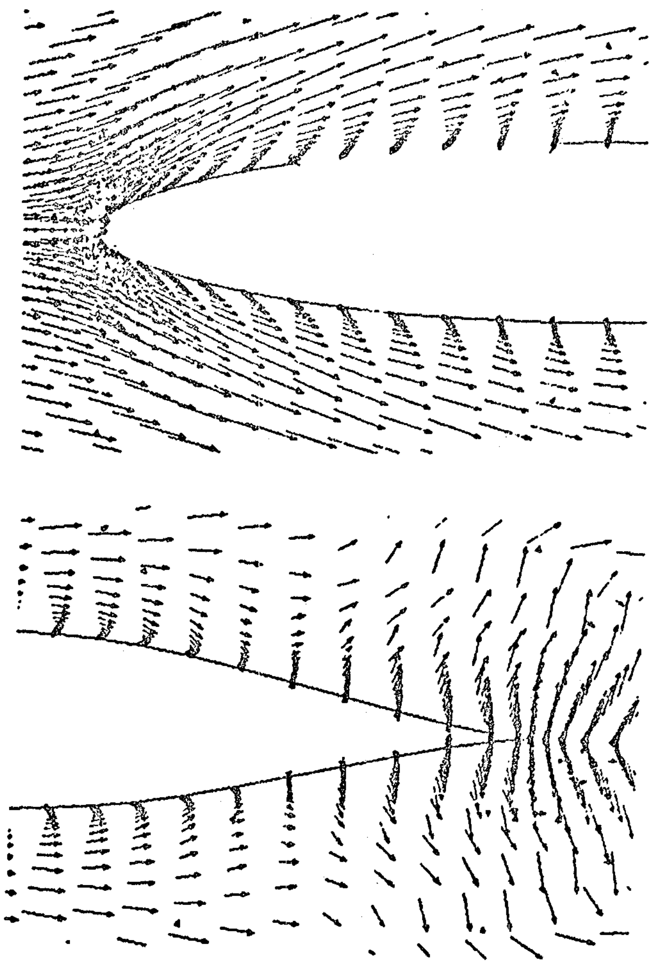
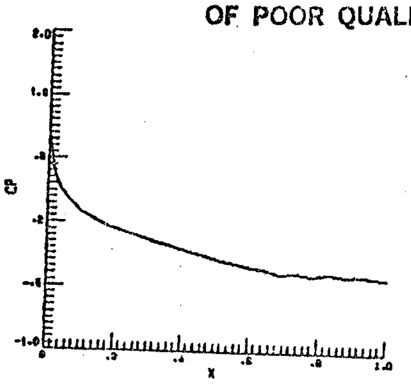


Fig. 44 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, t = 2.0, Re = 10,000
$\mu_a = \Omega \mathrm{Re} J |\nabla \cdot \underset{\sim}{V}|$, $\Omega = \phi|\underset{\sim}{\omega}|$, $\phi = 1.0$

(a)



(b)

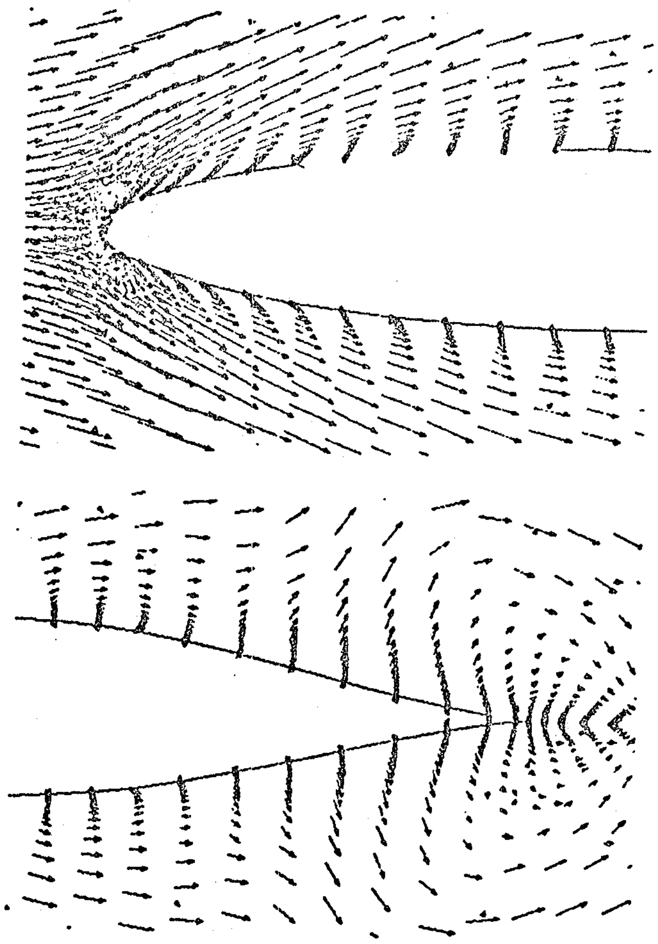
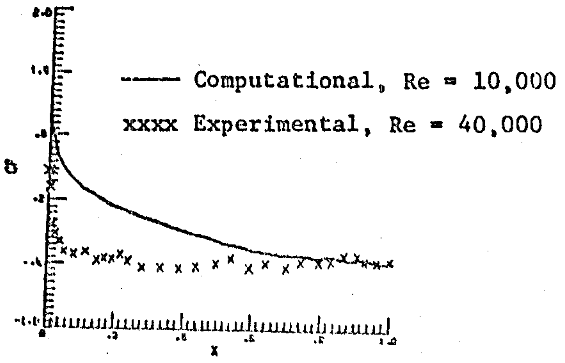


Fig. 45 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent Flow, $t = 3.0$, $Re = 10,000$
$\mu_a = \Omega ReJ|\nabla \cdot \underset{\sim}{V}|$, $\Omega = \phi|\underset{\sim}{\omega}|$, $\phi = 1.0$

(a)



(b)

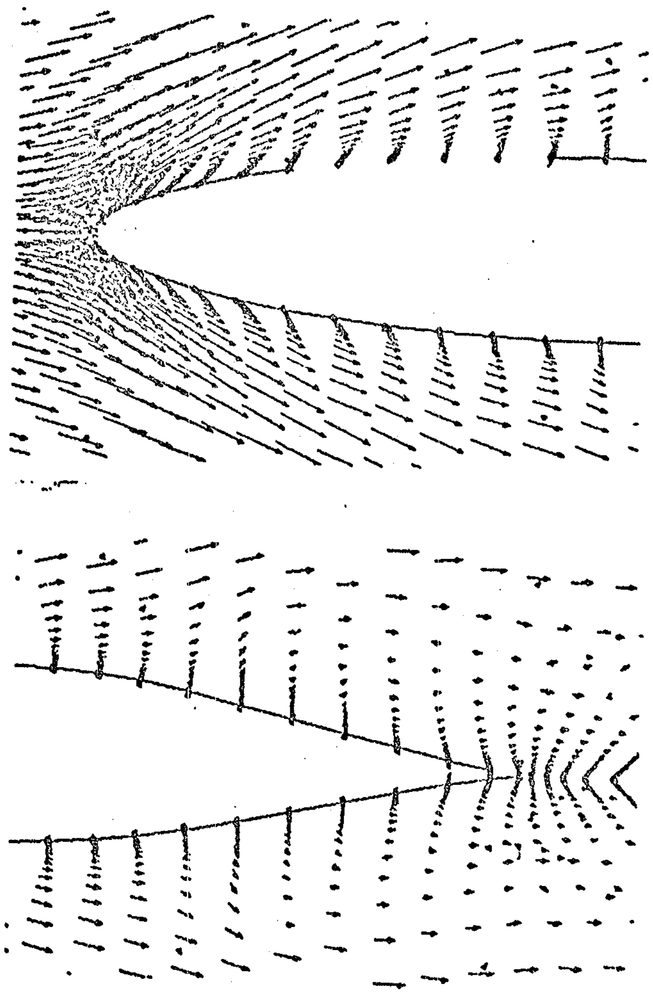
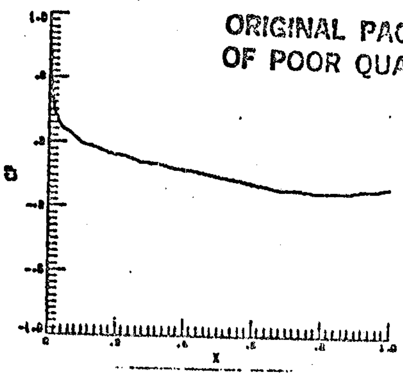


Fig. 46 (a) Surface Pressure Distribtuion
(b) Leading & Trailing-edge Velocity Vector Fields
Turbulent flow, t = 4.0, Re = 10,000
$\mu_a = \Omega \mathrm{ReJ} |\nabla \cdot \underset{\sim}{V}|$, $\Omega = \phi |\underset{\sim}{\omega}|$, $\phi = 1.0$

(a)



(b)

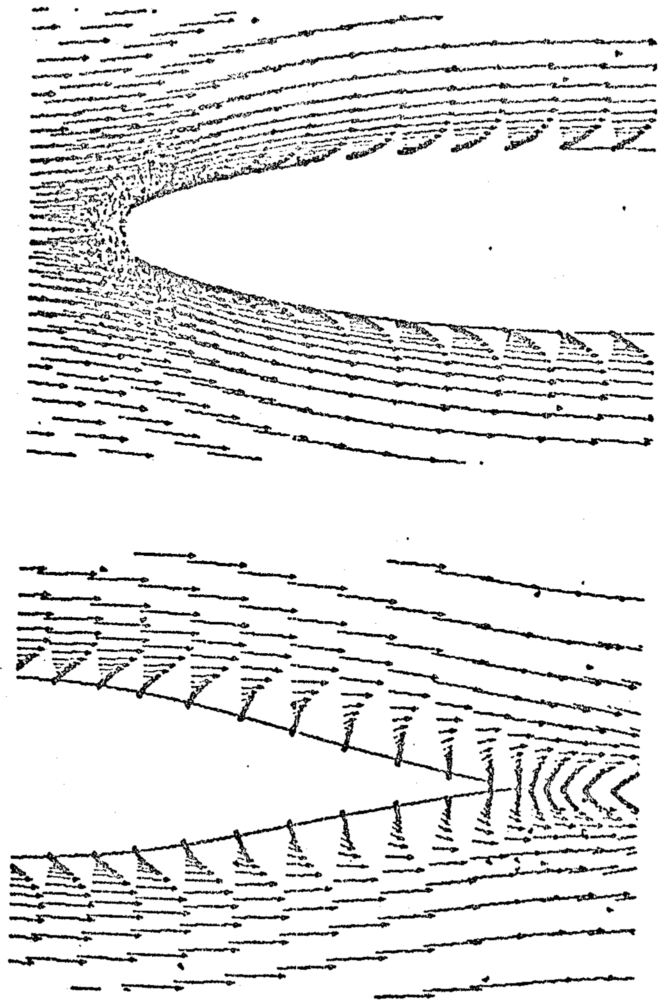
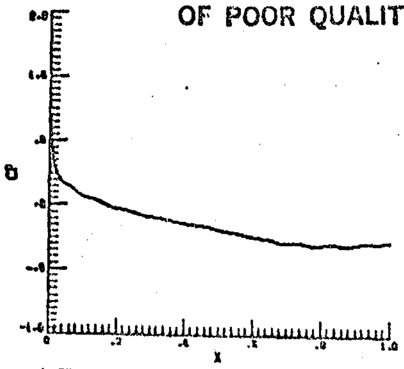
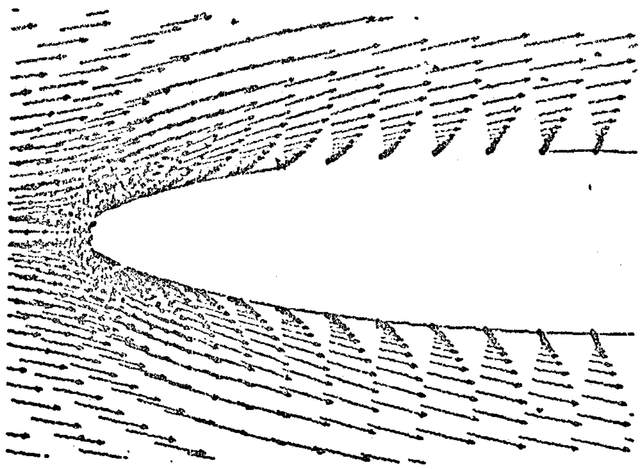


Fig. 47 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar flow, t = 1.0, Re = 1000

(a)



(b)



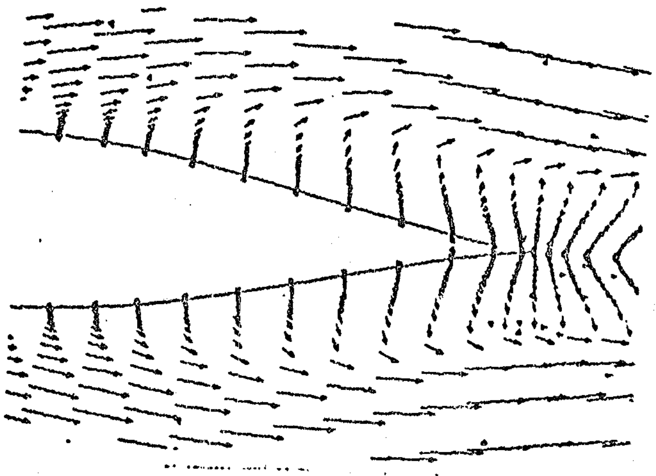


Fig. 48 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Vector Velocity Plots
Laminar Flow, t = 1.5, Re = 1000

ORIGINAL PAGE IS
OF POOR QUALITY

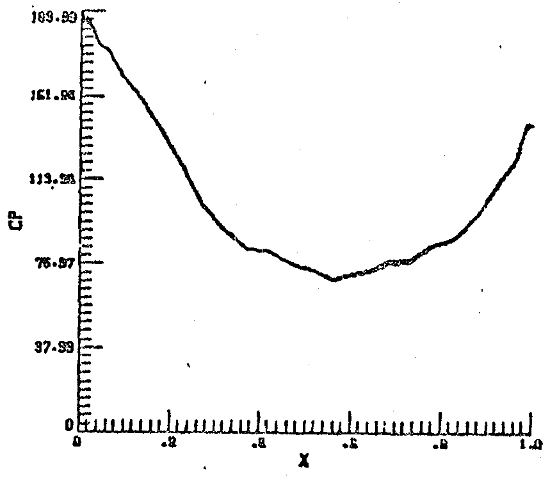
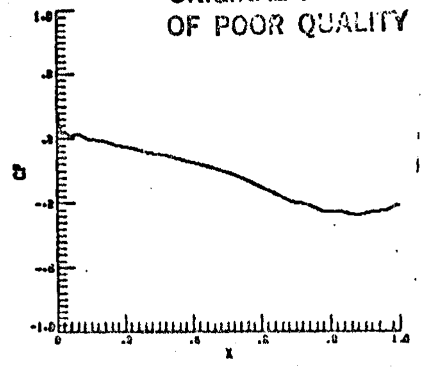
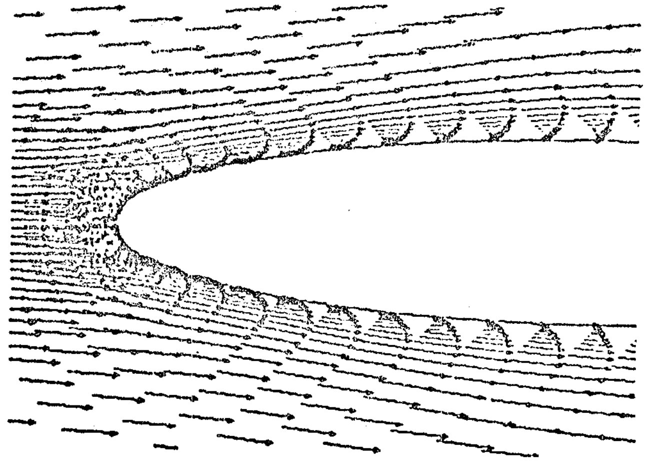


Fig. 49 Surface Pressure Distribution
Laminar Flow, t = 2.0, Re = 1000

(a)



(b)



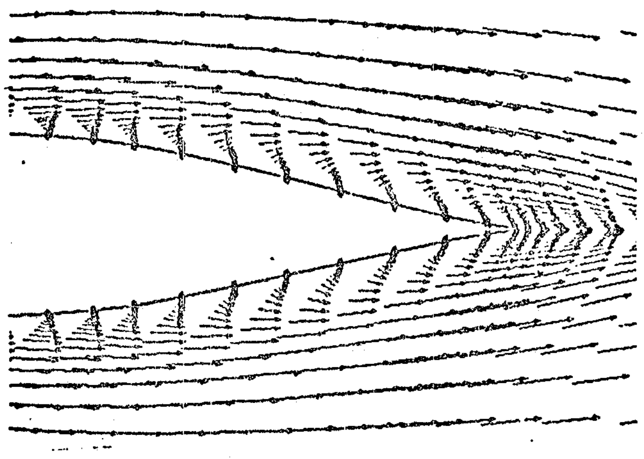
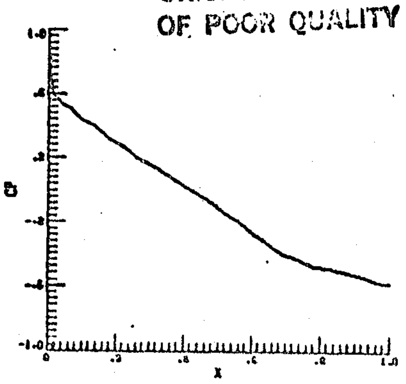


Fig. 50 (a) Surface Pressure Distribtuion
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 1.0, Re = 1000

(a)



(b)

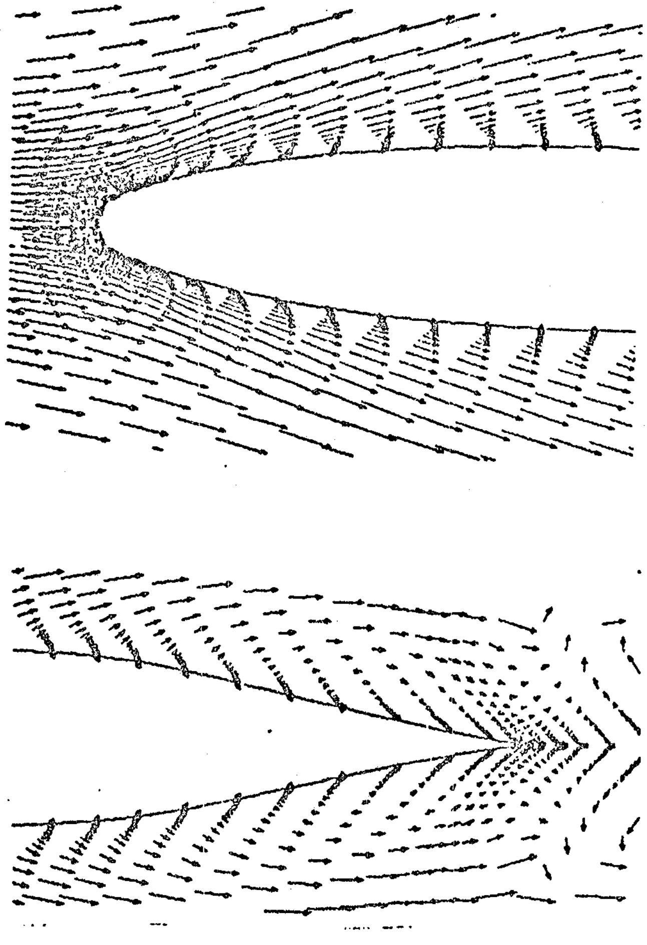
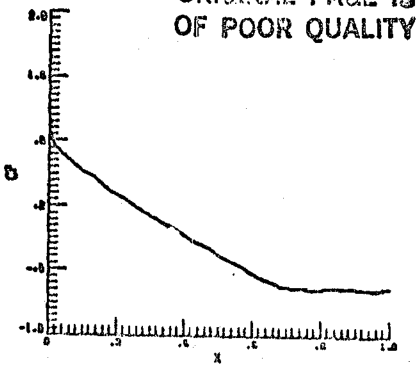


Fig. 51 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 2.0, Re = 1000

(a)



(b)

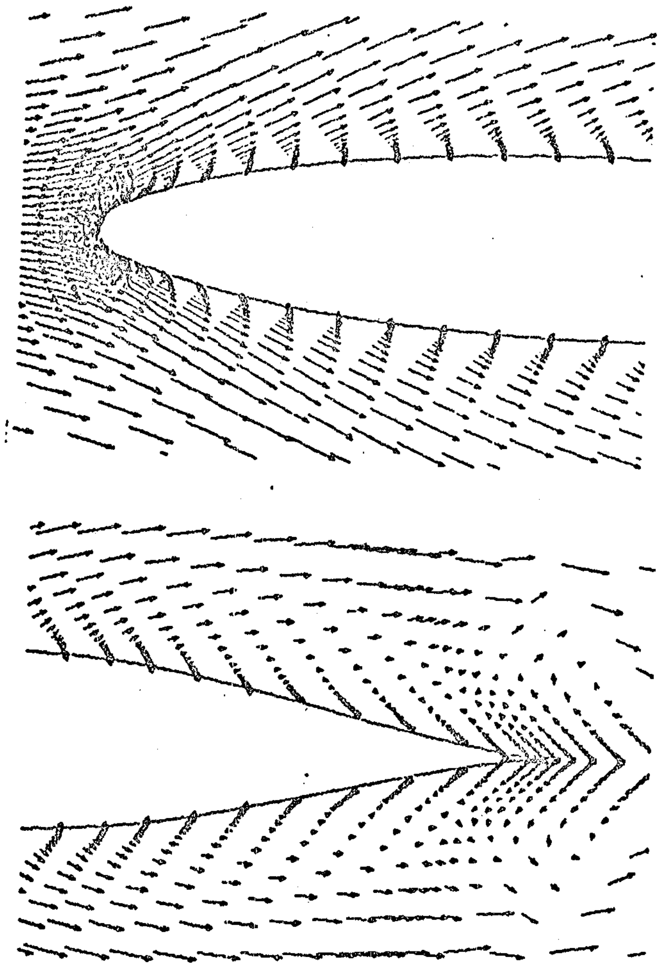
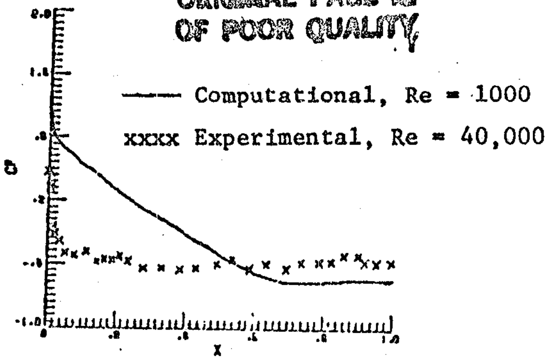
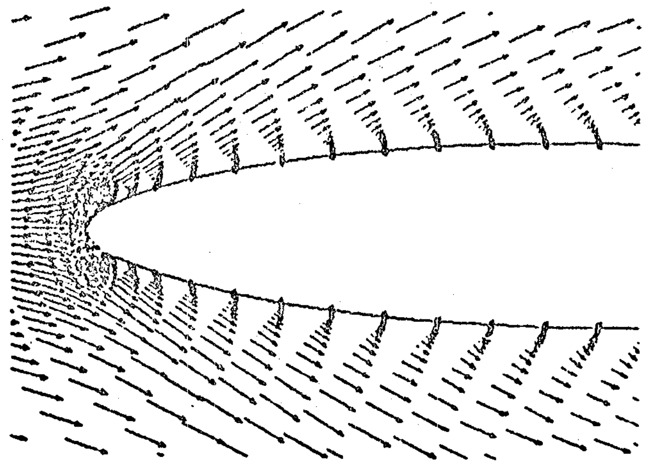


Fig. 52 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Veloctiy Vector Fields
Laminar Flow, t = 3.0, Re = 1000

(a)





(b)

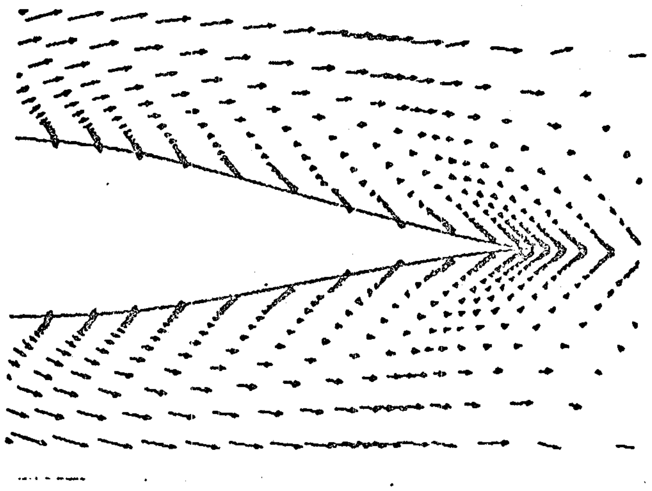
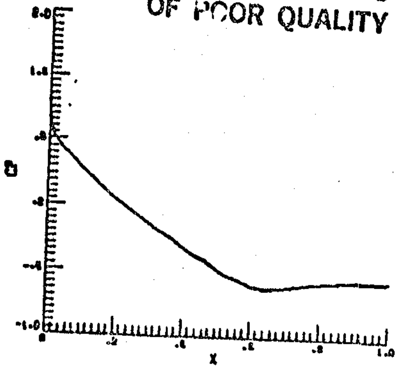
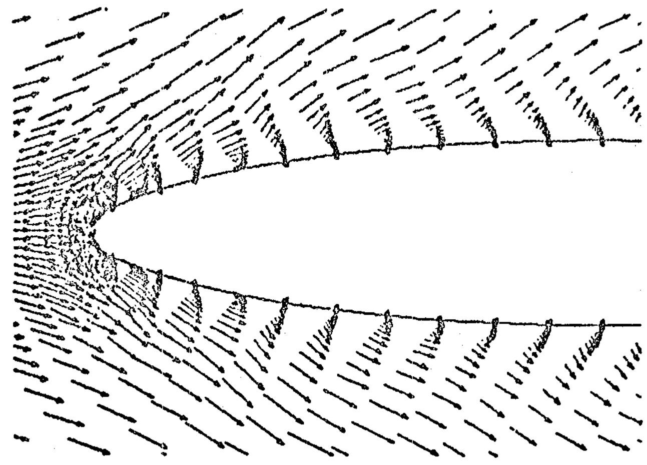


Fig. 53 (a) Surface Pressure Distribution .
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 4.0, Re = 1000

(a)



(b)



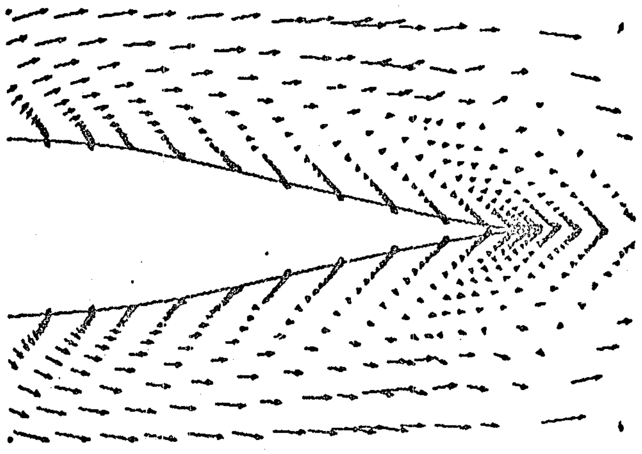
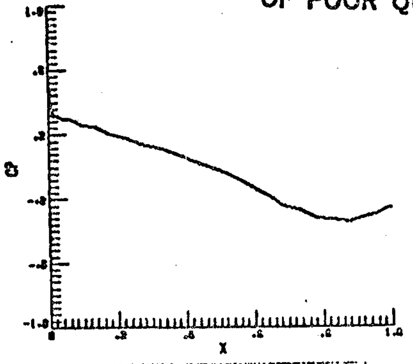


Fig. 54 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 5.0, Re = 10,000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

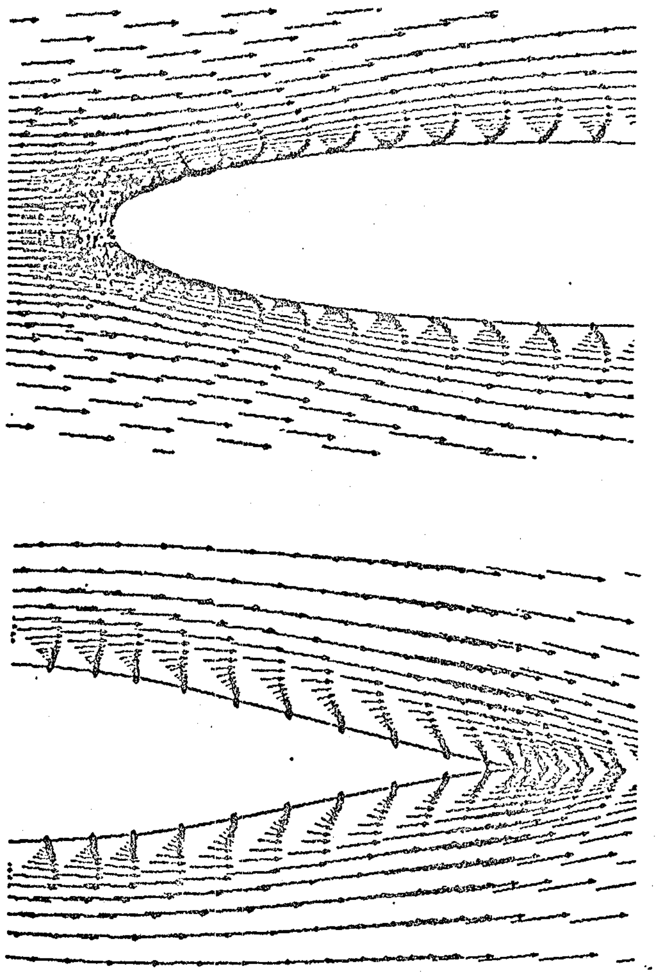
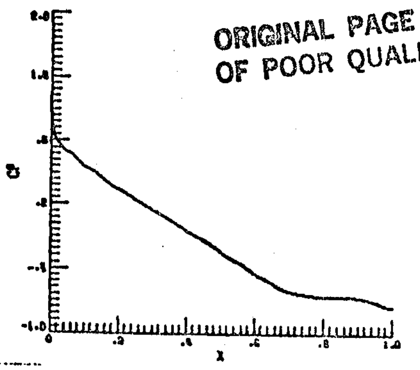


Fig. 55 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t́ = 1.0, Re = 1000

(a)



(b)

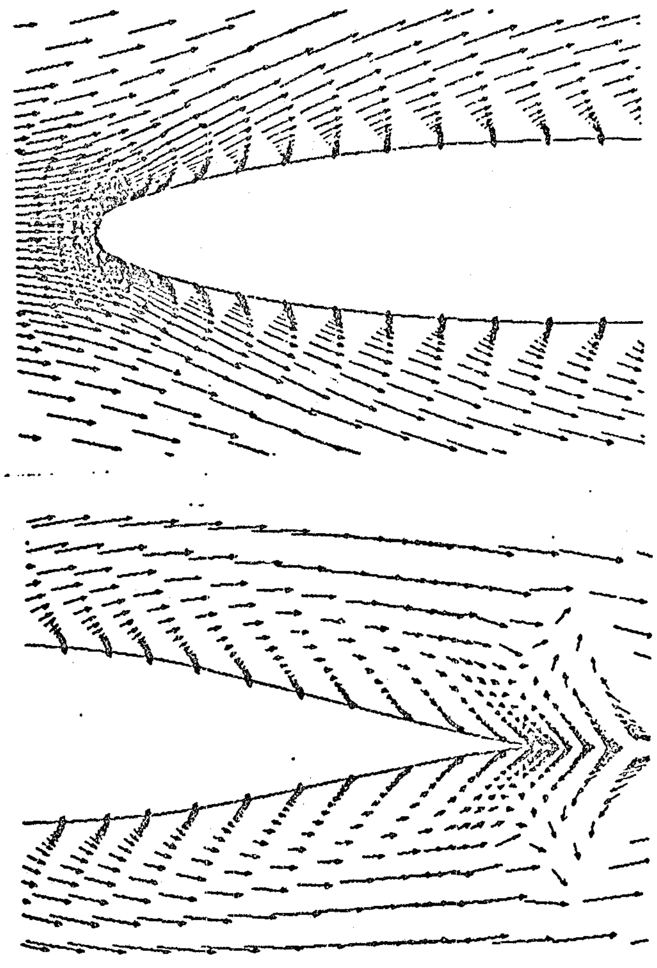
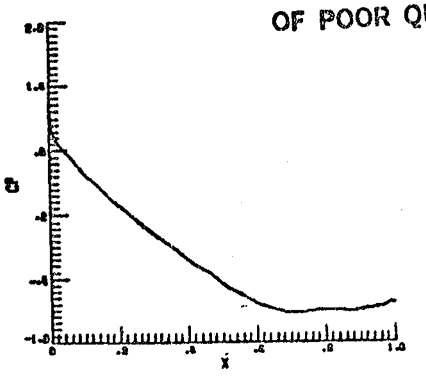


Fig. 56 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector
Laminar Flow, t = 2.0, Re = 1000.

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

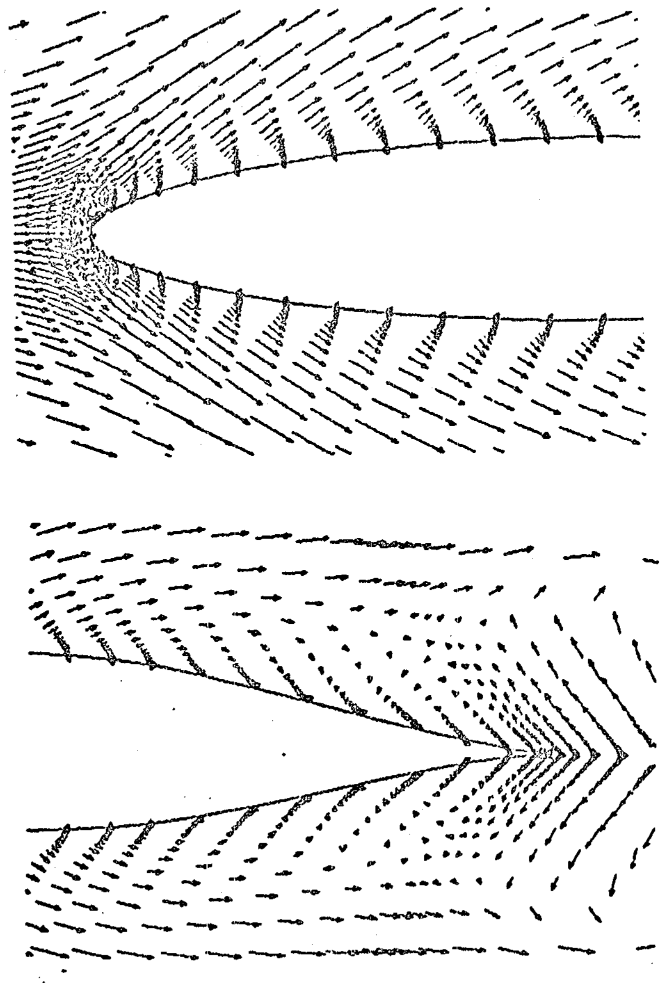
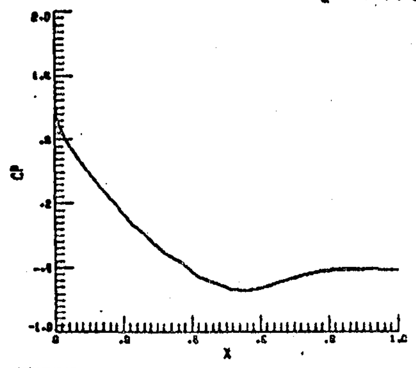


Fig. 57 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 3.0, Re = 1000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

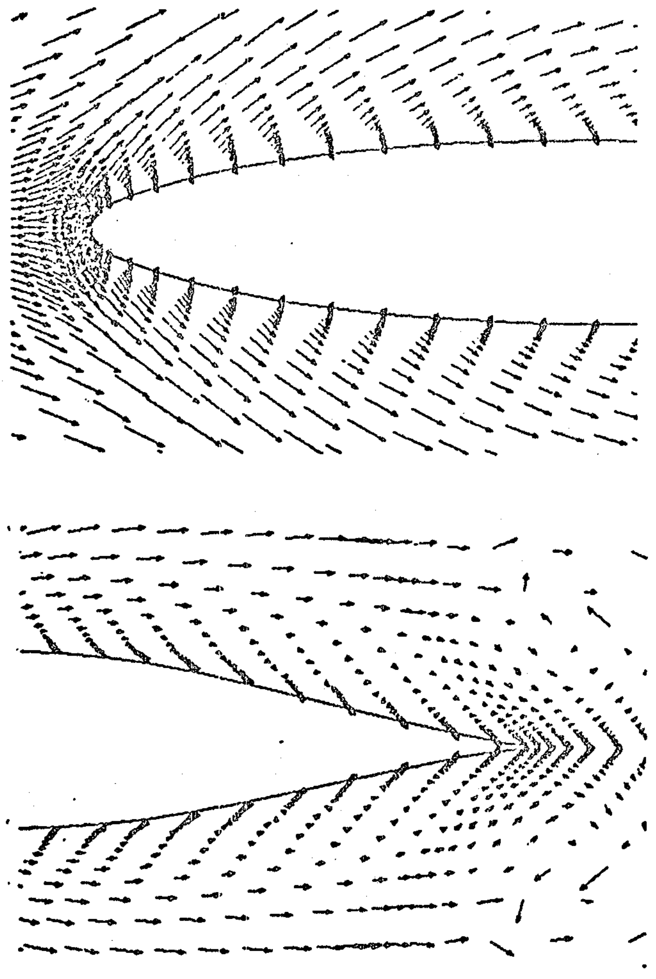
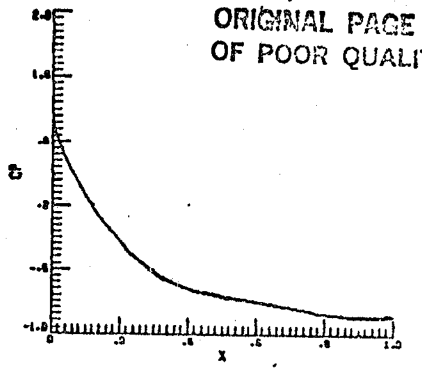


Fig. 58 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t' = 4.0, Re = 1000

(a)



(b)

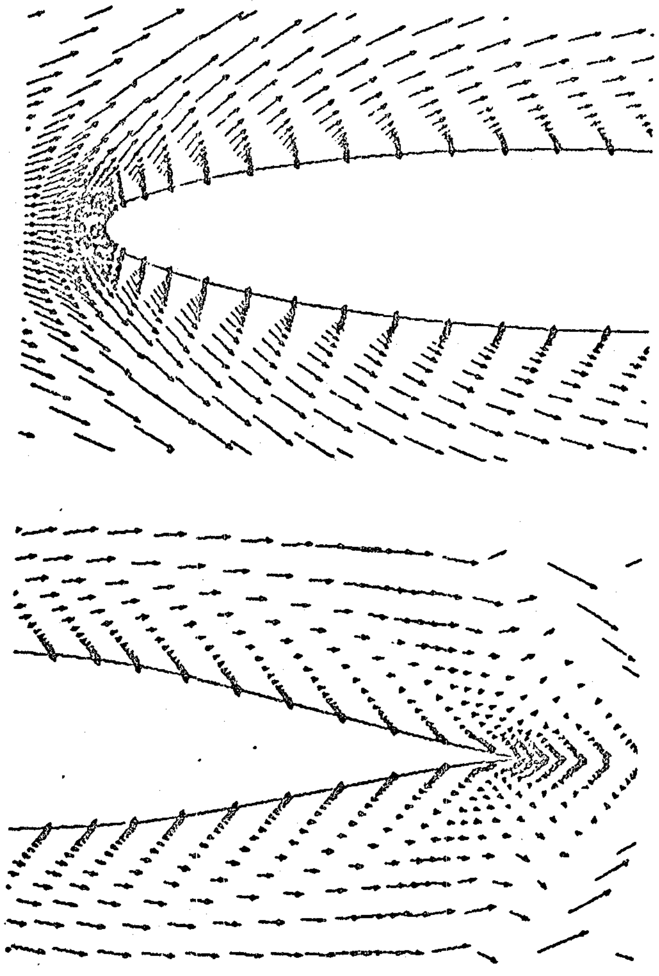
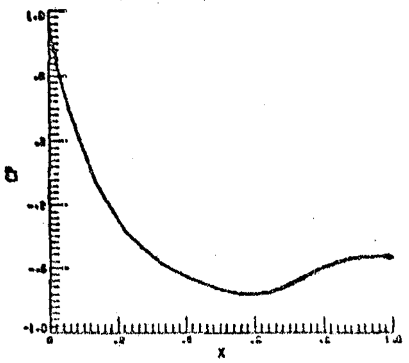


Fig. 59 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, $\hat{t}$ = 5.0, Re = 1000

(a)



(b)

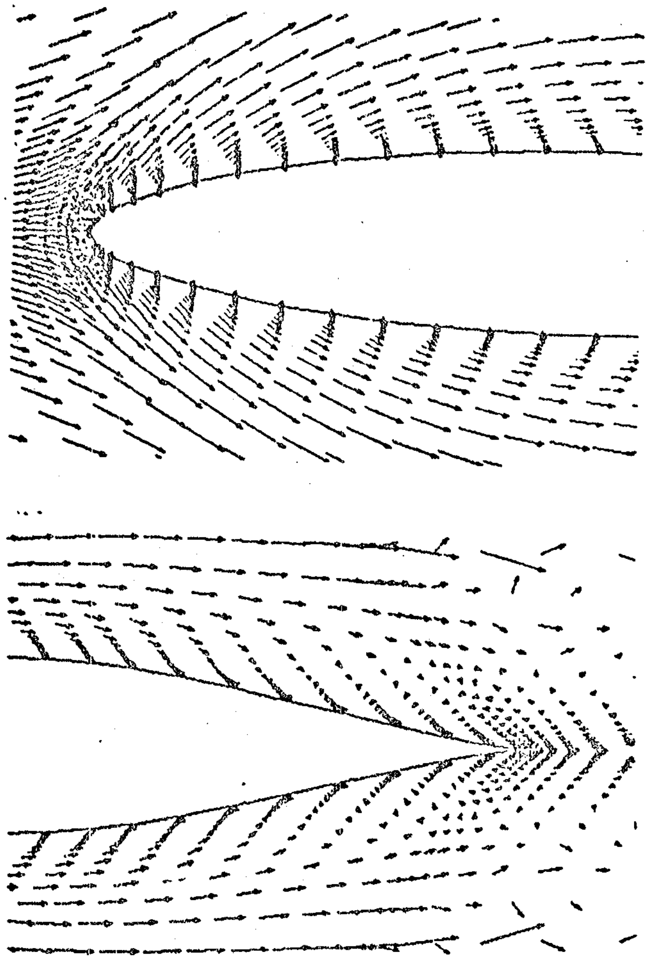
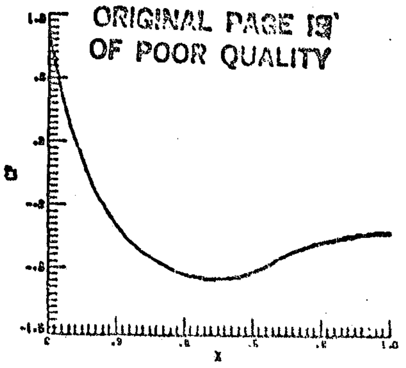


Fig. 60 (a) Surface Pressure Distribtuion
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 6.0, Re = 1000

(a)



(b)

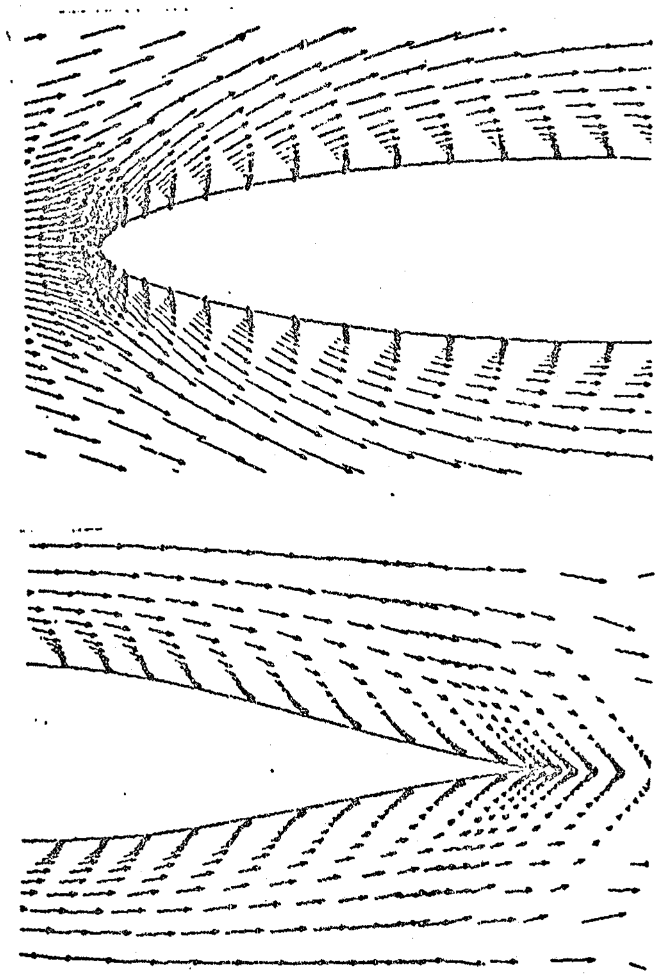
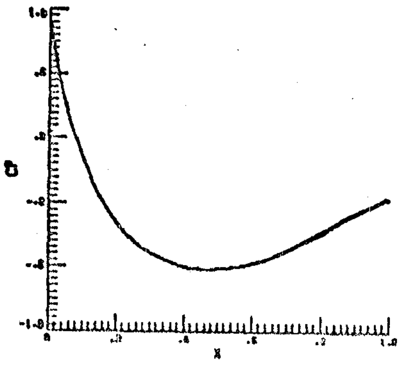


Fig. 61 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 7.0, Re = 1000 .

(a)



(b)

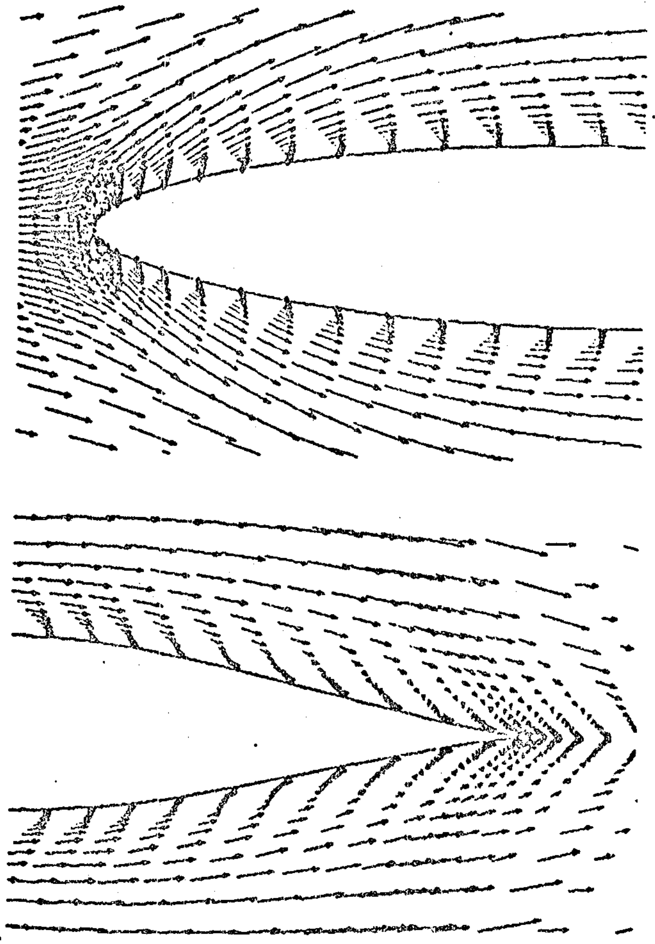
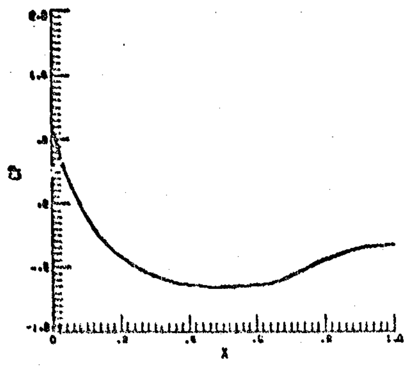


Fig. 62 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 8.0, Re = 1000

ORIGINAL PAGE IS
OF POOR QUALITY

(a)



(b)

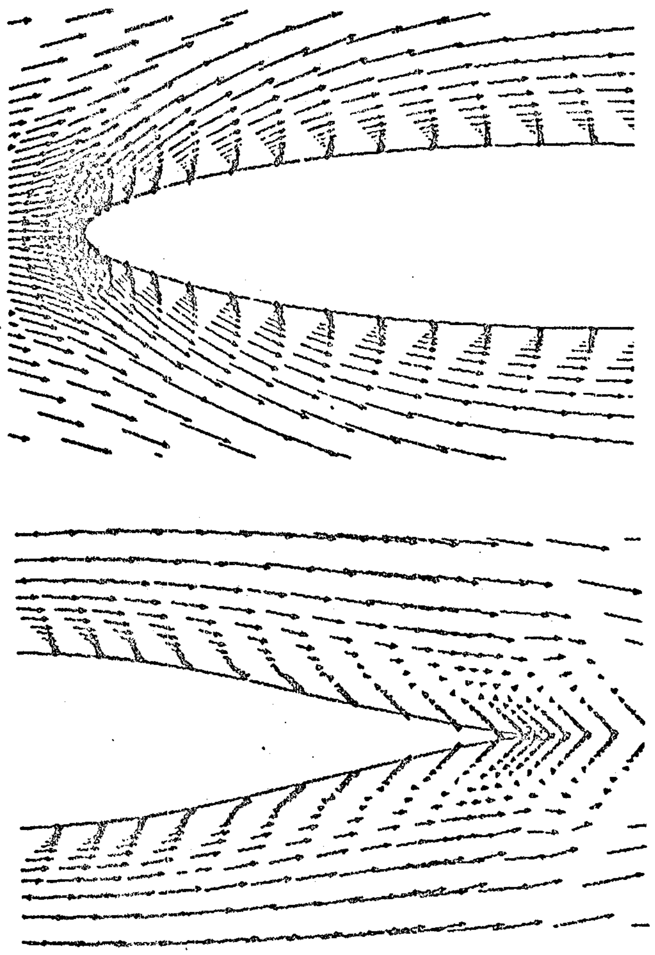
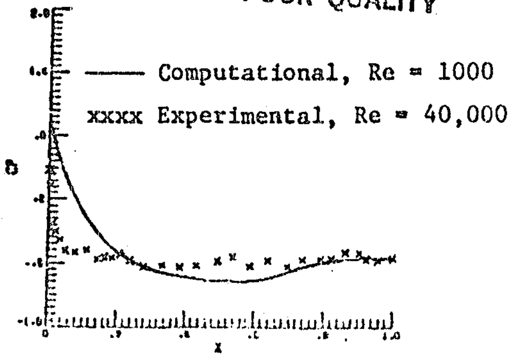


Fig. 63 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 9.0, Re = 1000 .

(a)



(b)

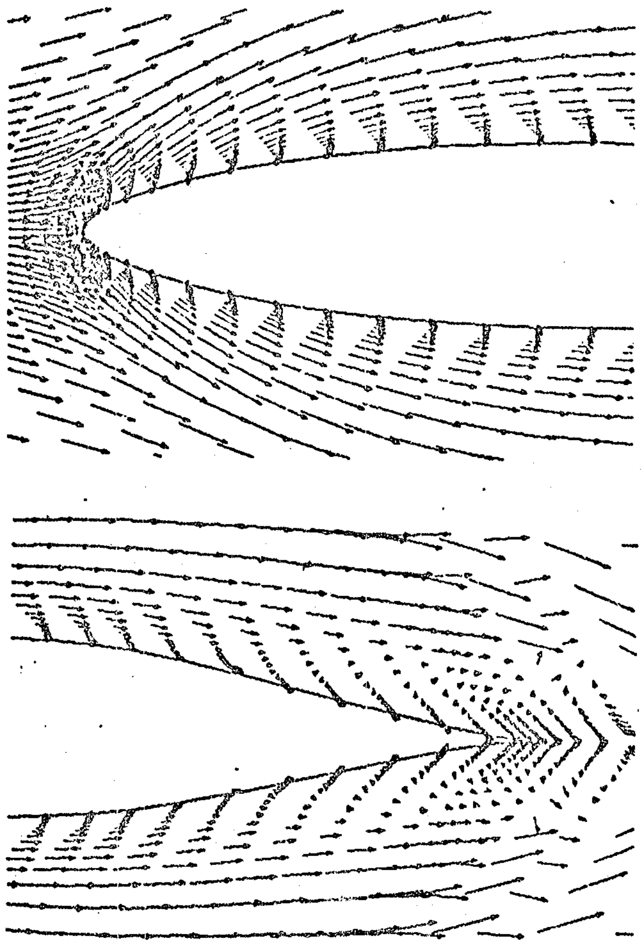
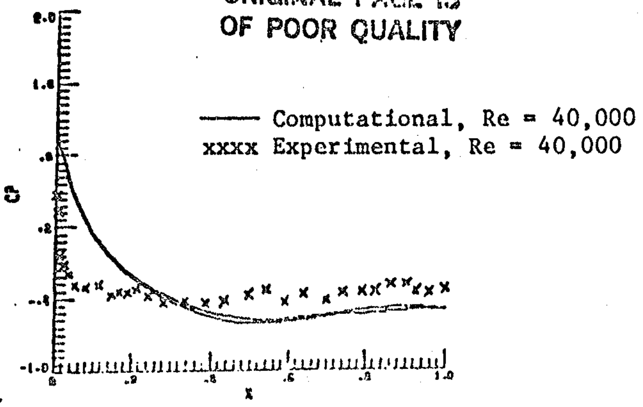


Fig. 64 (a) Surface Pressure Distribution
(b) Leading & Trailing-edge Velocity Vector Fields
Laminar Flow, t = 10.0, Re = 1000 .

(a)



(b)

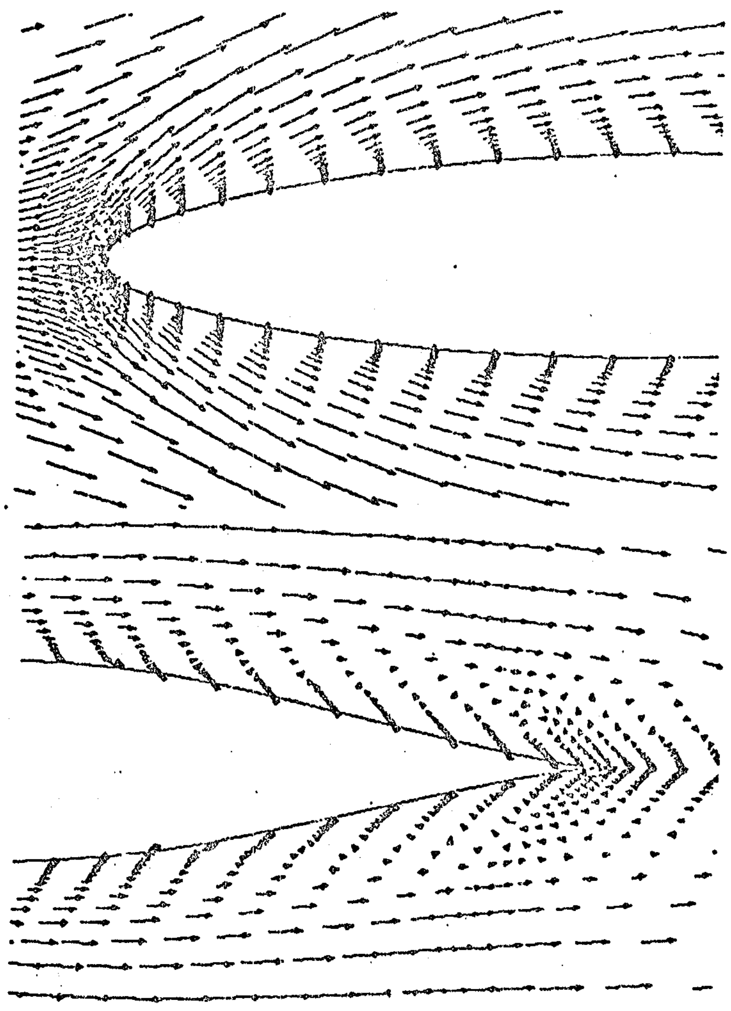


Fig. 65 (a) Surface Pressure Distribution, (b) Leading & Trailing Edge Velocity Vector Fields. Turbulent flow, t = 20.0, Re = 40,000. $\mu_a = \Omega \text{Re} J|\underset{\sim}{\nabla} \cdot \underset{\sim}{V}|$, $\Omega = \phi|\underset{\sim}{\omega}|$

REFERENCES

1. Goldstein, S., "Fluid Mechanics in the First Half of This Century," Ann. Ref. Fluid Mech. I, 1969, p. 1.

2. Courant, R., Friedricks, K. O., and Lewy, H., "Über die partiellen Differenzengleichugen ds Mathematischen Physik," Math. Ann. 100, 1928, p. 32.

3. Richtmeyer R. D., and Morton, K. W., Difference Methods for Initial-Value Problems, 2nd ed., Interscience Publishers, New York, 1967.

4. MacCormack, R. W., "The effect of viscosity in Hypervelocity Impact Cratering," AIAA paper 69-354, 1969.

5. Chapman, D. R., "Computational Aerodynamics Development and Outlook," AIAA J., 17, 1979, p. 1293

6. Bailey, F. R., "A View Toward Future Fluid Dynamics Computing," AIAA paper 78-1112, 1978.

7. Lomax, H., "Some Prospects for the Future of Computation Fluid Dynamics Computing," AIAA J., 20, 1982, p. 1033.

8. Turkel, E., "Progress in Computational Physics," Computer and Fluids, 11, 1983, p. 121.

9. Roache, P. J., Computational Fluid Dynamics, 1st ed., Hermosa Publishers, New Mexico, 1976.

10. Thompson, J. F., et. al., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," J. of Computational Physics, 15, 1974.

11. Thompson, J. F., "A Survey of Grid Generation Techniques in Computa-Fluid Dynamics," AIAA paper 83-0447, 1983.

12. Thompson, J. F., Warsi Z. U. A., and Mastin, C. W., "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," J. of Computational Physics, 47, 1982, p. 1.

13. Thompson, J. F., (Ed.), Numerical Grid Generation, Elsevier, New York, 1982.

14. Briley, W. R., and McDonald, H., "Solutions of the Three-Dimensional Compressible Navier Stokes Equations by an Implicit Technique," Lecture Notes in Physics, Springer Verlag, 35 p. 105

15. Beam, R. M. and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA J., 16, 1978, p. 393.

16. MacCormack, R. W., "A Rapid Solver for Hyperbolic Systems of Equations," Lecture Notes in Physics, Springer-Verlag, 59.

17. MacCormack, R. W., "A Numerical Method for Solving the Equations of Compressible Flow," AIAA paper 81-0110, 1981.

18. Lomax, H., "Recent Progress in Numerical Techniques for Flow Simulation," AIAA J., 14, 1976, p. 512.

19. Marvin, J. G., "Turbulence Modeling for Computational Aerodynamics," AIAA J., 21, 1983, p. 941.

20. Levin, R. D., "Supercomputers," Scientific American, January 1982, p. 118.

21. Mead, C. and Conway, L., Introduction to VLSI Systems, 1st ed., Addison-Wesley Publishing Company, Massachusetts, 1980.

22. Cebeci, T., Hirsh, R., Keller, H., and Williams, P., "Studies of Numerical Methods for the Plane Navier-Stokes Equations," Computer Method in App. Mech. and Eng., 27, 1981, p. 13.

23. Mehta, U. B. and Lavan, Z., "Starting Vortex, Separation Bubbles and Stall: A Numerical Study of Laminar Unsteady Flow Around an Airfoil," J. Fluid Mech., 67, 1975, p. 227.

24. Thompson, J. F., Thames, F. C., Hodge, J. K., Shank, S. P., Reddy, R. N., and Mastin, C. W., "Solutions of the Navier-Stokes Equations in Various Flow Regimes on Fields Containing any Number of Arbitrary Bodies Using Body-Figged Coordinate Systems," 5th International Conference on Numerical Methods in Fluid Dynamics, Euschede, The Netherlands, 1976.

25. Hodge, J. K., Stone, A. L. and Miller, T. E., "Numerical Solution of Airfoils Near Stall in Optimized Boundary-Fitted Curvilinear Coordinates," AIAA J., 17, 1979, p. 458.

26. Sugavanam, A. and Wu, J. C., "Numerical Study of Separated Turbulent Flow Over Airfoils," AIAA J., 20, 1982, p. 464.

27. Hegna, H., "Numerical Solution of Incompressible Turbulent Flow Over Airfoils Near Stall," AIAA J., 20, 1982, p. 29.

28. Bernard, R. S., "Approximate Factorization for Incompressible Flow," Ph.D. Dissertation, Mississippi State University, 1981.

29. Moitra, A., "An Implicit Solution of the Three-Dimensional Navier-Stokes Equatins for an Airfoil Spanning a Wind Tunnel," Ph.D. Dissertation, Mississippi State University, 1982.

30. Freeman, L.M., "The Use of an Adaptive Grid in a Solution of the Navier-Stokes Equatins for Incompressible Flow," Ph.D. Dissertation, Mississippi State University, 1982.

31. Mueller, T. J. and Batill, S. M., "Experimental Studies of Separation on a Two-Dimensional Airfoil at Low Reynolds Numbers," AIAA J., 20, 1982, p. 457.

32. Baldwin, B. S. and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, 1978.

33. Varga, R. S., Matrix Iterative Analysis, Prentice-Hall, Inc., New Jersey.

34. Thompson, J. F. and Mastin, C. W., "Order of Difference Expressions in Curvilinear Coordinate Systems," Advances in Grid Generation, ASME, NY, 1983, p. 17.

35. Sorensen, R. L., "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equation," NASA TM 81198.

36. Hirt, C. W. and Harlow, F. H., "A General Corrective Procedure for the Numerical Solution of Initial Value Problems," J. of Computational Physics, 2, 1967.

37. Cebeci, T., "Calculation of Compressible Turbulent Boundary Layers with Heat and Mass Transfer," AIAA paper 70-741, 1970.

38. Smith, R. E., et. al., "The Solution of the Three-Dimensional Viscous-Compressible Navier-Stokes Equations on a Vector Computer," Advances in Computer Methods for Partial Differential Equations, IMACS, p. 245, 1979.

39. Shang, J. S., et. al., "Performance of a Vectorized Three-Dimensional Navier-Stokes Code on the CRAY-1 Computer," AIAA J., Vol. 18, 9, p. 1073, 1980.

40. Spradley, L. W., et. al., "Solution of the Three-Dimensional Navier-Stokes Equations on a Vector Processor," AIAA J., Vol. 19, 10, 1981, p. 1302.

41. Gordon, P., "Nonsymmetric Difference Equations," J. Soc. Indus. Appl. Math., Vol. 13, 3, 1965, p. 667.

42. Scala, S. M., "Solution of the Time-dependent Navier-Stokes Equations for the Flow Around a Circular Cyliner," AIAA J., Vol. 6, 5, 1968, p. 815.

43. Gourlay, A. R., "Hopscotch: A Fast Second-Order Partial Differential Equation Solver," J. Inst. Maths. Appl., Vol. 7, 1971, p. 217.

44. Gourlay, A. R., et. al., "Generat Hopscotch Algorithm for the Numerical Soltuion of Partial Differential Equations," J. Inst. Maths. Appl., Vol. 7, 1971. p. 217.

45. Gane, C. R., et. al., "Block Hopscotch Procedures for Second Order Parabolic Differential Equations," J. Inst. Maths. Appl., Vol. 19, 1977, p. 205.

46. Evans, D. J., et. al., "The Solution of Elliptic Partial Differential Equations by a New Block Over-Relaxation Technique," Intern. J. Computer Math., Vol. 10, 1982, p. 269.

47. Gourlay, A. R., et. al., "Hopscotch Methods for Elliptic Partial Differential Equations," J. of Computational and Applied Math., Vol. 5, 2, p. 103, 1979.

48. Greenberg, J. B., "Semi-Implicit Hopscotch-Type Methods for the Time Dependent Navier-Stokes Equations," AIAA J., Vol. 20, 8, 1982, p. 1064.

49. South, J. C., "Processor Algorithms for Transonic Flow Calculations," AIAA J., Vol. 18, 7, p. 786, 1980.

50. CDC CYBER 200 FORTRAN LANGUAGE 1.4, Reference Manual, Control Data Corporation, California.

51. Gourlay, A. R., et. al., "The Construction of Hopscotch Methods for Parabolic and Elliptic Equations in Two Dimensions with Mixed Derivatives," J. of Computational and Applied Math, Vol. 3, 3, 1977, p. 20.

52. Thompson, J. F., Personal Communication, Department of Aerospace Engineering, Mississippi State University, 1983.

53. Erlich, L. W., "An Ad Hoc SOR Method," Elliptic Problem Solver, Ed. Schultz, M., Academic Press, NY, 1981.

54. Vinokur, M., "On One-Dimensional Stretching Functions for Finite-Difference Calculations," J. of Compu. Phy., Vol. 50, 1983, p. 215.

End of Document