TWO-DIMENSIONAL EULER COMPUTATIONS ON A TRIANGULAR MESH USING AN UPWIND, FINITE-VOLUME SCHEME

by

David Lee Whitaker

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Aerospace Engineering

APPROVED:

Bernard Grossman, Chairman

Joseph A. Schetz

Wayne L. Neu

Robert W. Walters

Roger L. Simpson

December 1988

Blacksburg, Virginia

TWO-DIMENSIONAL EULER COMPUTATIONS ON A TRIANGULAR MESH USING AN UPWIND, FINITE-VOLUME SCHEME

by

David Lee Whitaker

Bernard Grossman, Chairman

Aerospace Engineering

(ABSTRACT)

A new numerical procedure has been developed for the finite-volume solution of the Euler equations on unstructured, triangular meshes using a flux-difference split, upwind method. The procedure uses a dual mesh system to implement the finite-volume scheme using conserved variables stored at the vertices of the triangles. The vertices of the triangles are located approximately in the center of the dual mesh cells and conservation is enforced about each dual mesh finite-volume. Techniques are developed for implementing Roe's approximate Riemann solver on unstructured grids. Higher order accuracy is achieved by using MUSCL-differencing. MUSCL-differencing is implemented on an unstructured grid by interpolating the values stored at the vertices of triangular elements to find the value at the outermost point of the three-point MUSCL-differencing formula. Flow solutions are computed using a four-stage Runge-Kutta time integration. Convergence is accelerated using non-standard weighting of the Runge-Kutta stages, variable time-steps, residual smoothing and residual min-

imization. Applications and comparisons with structured grid solutions are made for a supersonic shock reflection problem, the supersonic flow over a blunt body, flow through a simple wedge inlet, and several AGARD 07 working group transonic airfoils. In general, the solutions computed by the upwind solver on the unstructured grids were as accurate as upwind solutions on a structured mesh. The blunt body solution, and some of the transonic airfoil solutions on the unstructured meshes, appeared to be less accurate than the structured mesh solutions. Fortuitously, the structured meshes used in these solutions tended to line up with the shock waves present in the flow-field. The upwind flux-differencing scheme captured the shock wave with greatest accuracy when it was applied normal to the discontinuity, as was done in these test cases.

Acknowledgements

The author wishes to express his sincere and deep appreciation to his advisor, Dr. Bernard Grossman who gave extensive guidance, support and encouragement to this work. Without Dr. Grossman's support and encouragement, this study would not have been possible.

The author would also like to thank Dr. Joseph Schetz, Dr. Wayne Neu, Dr. Roger Simpson and Dr. Robert Walters for serving on his Advisory Committee and for their helpful and useful comments, discussions, and suggestions for this study.

Very special thanks are offered to the author's parents for their continuous moral support through out this research effort.

Thanks are also offered to the author's friends for their support, suggestions, and help during this study.

Table of Contents

I. INTRODUCTION **1**

II. MATHEMATICAL FORMULATION **14**

2.1 Euler Equations 14

2.2 Integral form of the Euler equations 18

2.3 Physical Boundary Conditions 19

2.4 Far-Field Airfoil Boundary Conditions 21

III. NUMERICAL FORMULATION **29**

3.1 Discretization of Governing Equations 29

3.2 Roe's Approximate Riemann Solver 32

3.3 Entropy fix for Roe's Approximate Riemann Solver 43

3.4 Extension of MUSCL-Differencing to Unstructured Meshes 45

3.5 Algorithm and Data Structure 50

3.6 Numerical Boundary Conditions 52

3.6.1 Solid Wall Boundary 52

3.6.2 Far-field Boundary 58

3.6.3 Kutta Condition 60

IV. TIME INTEGRATION 61

4.1 Runge-Kutta Time Stepping 63

4.2 Convergence Acceleration 66

4.2.1 Non-Standard Weighting of Runge-Kutta Stages 66

4.2.2 Local Time Stepping 66

4.2.3 Residual Smoothing 68

4.2.4 Residual Minimization 71

V. RESULTS AND DISCUSSION 75

5.1 Shock Reflection Problem 75

5.2 Blunt Body Problem 77

5.3 10^0 Wedge Inlet. 79

5.4 Subcritical NACA 0012 Airfoil 80

5.5 Supercritical NACA 0012 Airfoil (Case 1) 81

5.6 Supercritical NACA 0012 Airfoil (Case 2) 83

5.7 RAE 2822 Airfoil 85

5.8 NLR 7301 Airfoil 87

5.9 Karman-Trefftz Airfoil and Flap 88

VI. CONCLUDING REMARKS 91

REFERENCES 95

Figures 103

List of Illustrations

Figure 1. Representative cell-centered structured grid. 104

Figure 2. Representative unstructured grid and dual mesh cell. 105

Figure 3. Representative edge of an unstructured mesh. 106

Figure 4. Representative dual mesh cell at grid boundary of an unstructured mesh. 107

Figure 5. Normal extrapolation of boundary conditions on an unstructured mesh. 108

Figure 6. Unstructured mesh for the shock reflection problem. 109

Figure 7. Mach Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$ ...110

Figure 8. Pressure Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$ 111

Figure 9. Density Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$ 112

Figure 10. Wall Pressure Comparison - Shock reflecting from a flat plate, $M_\infty = 2.9$ 113

Figure 11. Unstructured Mesh for the Blunt Body Problem. 114

Figure 12. Mach Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$ 115

Figure 13. Pressure Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$ 116

Figure 14. Density Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$ 117

Figure 15. Symmetry Line Pressure Comparison - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$ 118

Figure 16. Unstructured Mesh for the 10° Wedge Inlet. 119

Figure 17. Mach Contours - 10° Wedge Inlet, $M_\infty = 5$ 120

Figure 18. Pressure Contours - 10° Wedge Inlet, $M_\infty = 5$ 121

Figure 19. Density Contours - 10° Wedge Inlet, $M_\infty = 5$ 122

Figure 20. Lower Wall Pressure Comparison - 10° Wedge Inlet, $M_\infty = 5$ 123

Figure 21. Lower Wall Pressure Comparison - 10° Wedge Inlet, $M_\infty = 5$ 124

Figure 22. Unstructured Grid - General Far-Field View of Airfoil Grids. 125

Figure 23. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$ 126

Figure 24. Mach Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$ 127

Figure 25. Pressure Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$ 128

Figure 26. Density Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$ 129

Figure 27. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$ 130

Figure 28. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 131

Figure 29. Mach Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 132

Figure 30. Pressure Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 133

Figure 31. Density Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 134

Figure 32. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 135

Figure 33. Surface Entropy Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 136

Figure 34. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 137

Figure 35. Surface Entropy Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$ 138

Figure 36. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 139

Figure 37. Mach Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 140

Figure 38. Pressure Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 141

Figure 39. Density Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 142

Figure 40. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 143

Figure 41. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 144

Figure 42. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$ 145

Figure 43. Close-up of Unstructured Mesh - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$ 146

Figure 44. Mach Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$ 147

Figure 45. Pressure Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$ 148

Figure 46. Density Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$ 149

Figure 47. Surface Pressure Comparison - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$ 150

Figure 48. Close-up of Unstructured Mesh - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$ 151

Figure 49. Mach Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$ 152

Figure 50. Pressure Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$ 153

Figure 51. Density Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$ 154

Figure 52. Surface Pressure Comparison - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$ 155

Figure 53. Close-up of Unstructured Mesh - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$ 156

Figure 54. Mach Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$ 157

Figure 55. Pressure Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$ 158

Figure 56. Density Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$ 159

Figure 57. Close-up of Unstructured Mesh - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$ 160

Figure 58. Mach Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$ 161

Figure 59. Pressure Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$ ... 162

Figure 60. Density Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$ 163

Figure 61. Surface Pressure Comparison - Karman-Trefftz Airfoil, $M_\infty = 0.125$, $\alpha = 0°$ 164

Figure 62. Surface Pressure Comparison - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$ 165

Table 1. Comparison of Lift and Drag Coefficients for Karman-Trefftz Airfoil-Flap 90

List of Symbols

A .....one-dimensional Jacobian matrix

a .....speed of sound or scalar wave speed in x-coordinate

B.....two-dimensional Jacobian matrix

b .....scalar wave speed in y-coordinate

C .....dual mesh cell

∂C.....boundary of dual mesh cell C

c.....chord length of airfoil

c_l.....lift coefficient of airfoil

c_d.....drag coefficient of airfoil

c_a.....axial force coefficient of airfoil

c_n.....normal force coefficient of airfoil

D .....total velocity at far-field due to compressible vortex

E.....total energy of flow

F.....flux normal to dual cell side

$\vec{F}$.....conservative flux vector

f, g.....Cartesian components of flux vector

$\hat{f}, \hat{g}$.....components of flux vector in curvilinear coordinates

G.....centroid of mesh triangle

H.....total enthalpy

$\hat{i}, \hat{j}$.....Cartesian unit vectors

J.....Jacobian of a transformation matrix

K.....small parameter for Roe entropy fix

$\hat{k}$.....unit vector normal to cell side

Δl.....length of area-averaged dual mesh cell side edge

M.....Mach number or midpoint of triangular edge

$\hat{n}$.....outward facing unit normal vector

p.....pressure at a point

Q.....vector of state variables

$\hat{Q}$.....vector of conserved variables divided by the Jacobian

$<Q>$.....vector of area-averaged state variables

q.....scalar state variable

R.....vector of Residual terms, radius, or region of integration

$\overline{R}$.....vector of Laplacian averaged Residual terms

S.....eigenvector matrix for one-dimensional Euler equations

s.....parameter produced by Van Albeda's limiter

T.....temperature

t.....time variable

Δt .....time step

$\vec{U}$ .....velocity vector in Cartesian coordiantes

$\overline{U}$, $\overline{V}$ .....contravariant velocity components

u, v.....velocity component in Cartesian coordinates

V .....vertex of triangular mesh

x, y .....Cartesian coodinate system

$x_\xi, x_\eta, y_\xi, y_\eta$....mapping metrics

$\Delta x, \Delta y$.....spatial step in Cartesian coodinate system

$\vec{\nabla}$.....gradient operator

∇^2.....Laplacian operator

Greek symbols

α .....angle of attack

β .....$\sqrt{1 - M_\infty^2}$, supersonic similarity parameter

Γ.....circulation about an airfoil

γ .....ratio of specific heats

Δ_+ .....forward difference

Δ_- .....backward difference

$\Delta(\bullet)$ .....$(\bullet)_R - (\bullet)_L$, right and left difference

δ .....small parameter

ε.....small parameter or internal energy

$\vec{\eta}$ .....vector normal to area-averaged dual mesh cell side

$\hat{\eta}$ .....unit vector normal to area-averaged dual mesh cell side

ξ, η .....curvilinear coordinates

$\xi_x, \xi_y, \eta_x, \eta_y$ ...mapping metrics

θ .....angle in cylindrical coordinates

κ_i .....list of vertices surrounding vertex i

Λ .....diagonal matrix of wave speeds

λ .....wave speed or eigenvalue

ν .....Courant number

$\hat{\nu}$ .....unit vector normal to a median of triangular mesh

ρ .....mass density at a point

σ .....parameter controlling accuracy of MUSCL-differencing

$d\tau$.....integration variable for volume

ϕ.....perturbation velocity potential

Ω .....domain discretized by triangles

ω .....weighting factors for residual minimization

Subscripts

B.....associated with a boundary point

B_x, B_y .....x and y component of vector at boundary point

B.....associated with a boundary point

ff.....far-field condition

i, j, l, k .....indices of triangular vertices

i', j' .....imaginary nodes

ij .....edge defined by nodes i and j

ijk .....triangular centroid defined by nodes i, j, and k

ijl .....triangular centroid defined by nodes i, j, and l

L.....left state

m .....Runge-Kutta stage

R.....right state

∞ .....conditions at infinity

$'$ .....variables after affine transformation

Superscripts

k .....associated with vertex k

l.....associated with vertex l

$(+)$.....non-negative value

$(-)$.....non-positive value

n .....time step level

$\sim$ .....Variable evaluated at a Roe-averaged state

I. INTRODUCTION

The field of computational fluid dynamics (CFD), evolving over the past 25 years, has reached a stage where it is now playing a significant role in the aerodynamic design of many of the more recent civil and military aircraft[1-5]. Rapid advances in development of computational methods for fluid flow have been driven by the need for faster, more accurate design tools. Computational aerodynamics offers many advantages to the aircraft designer that were not previously available. Computational fluid dynamics gives the aircraft designer the numerical equivalent of a wind tunnel. Results from numerical experiments are available everywhere in the flow field, including regions of the flow where fluid parameters can't easily be measured. The results obtained are not contaminated by the presence of measuring instruments. Computational design tools can quickly and cheaply generate numerical results for a particular design test case. The speed at which computations can be made permits a wide variety of configurations to be evaluated. Solutions generated by computational fluid dynamics

may not always be as accurate as experimental results, but numerical algorithms and computers are continually improving both in accuracy and speed. In the author's opinion, the ultimate goal of computational fluid dynamics is the prediction of flow field parameters about realistic three-dimensional bodies to at least within the errors of measured data. This goal includes the geometric resolution of fuselages, wings, tails, wing-body junctures, nacelles and any external stores. All relevant physics of the fluid flow, including shocks, expansion fans, viscous boundary layers, and wakes should be fully resolved and accurately modeled.

Analytical or theoretical methods, c.f. [6], provide a way of obtaining a closed-form or near to closed-form solution of a fluid mechanics problem. In general, simplifying assumptions have to be made to reduce the governing equations to a linear form or perturbation methods must be used to obtain a linear solution to a non-linear problem. With these methods only the simplest possible geometries can be analyzed. However, it must be noted that results from analytical solutions are some of the most useful to the aircraft designer. Since results from analytical techniques are often in the form of equations, good general design information or guidelines are obtained.

Experimental methods, c.f. [6], for analyzing the fluid flow about a model have the capability of generating the most realistic results about any geometric configuration. Unfortunately, a model of the proposed design must be built and wind-tunnel time obtained. Certain flow regimes, such as hypersonic reacting

flows and very high Reynolds number transonic flows, cannot be adequately simulated by available wind tunnels. Wind-tunnel tests are generally the most expensive way of generating flow data due to the large electric power requirements of wind-tunnels and the cost of model construction. An aerodynamic experiment is also one of the most time consuming ways of obtaining data.

The numerical solution of the full Navier-Stokes equations will give a solution which, in principle, could be accurate to within the experimental error of empirical data. The full Navier-Stokes equations cannot be currently solved except for very low Reynolds number flows or steady, laminar flows over simple bodies. The computational grids that are necessary to resolve all length scales that would be present in a high Reynolds number flow would probably contain billions (at least) of grid points in three-dimensional space. Some work has been done on three-dimensional solutions of turbulent flow using the direct simulation approach, but the maximum Reynold's number that can be achieved in these simulations is quite limited[7]. Numerical solutions on grids resolving all turbulence scales appear to be impossible in the foreseeable future. Laminar solutions to the Navier-Stokes equations have been achieved for both two and three-dimensional flows. Laminar solutions to the Navier-Stokes equations are useful in some flow regimes but the utility of a laminar Navier-Stokes solution in the transonic and supersonic regimes is not obvious. In general, laminar Navier-Stokes flow solvers are expensive for even simple two-dimensional flows and are not suitable as design tools at the present time.

Because of the limitations of the Navier-Stokes equations, simplified versions of the Navier-Stokes equations have been sought. The model equations that are obtained from simplifying the Navier-Stokes equations ignore some aspects of the physics of the flow. The smallest step down from the Navier-Stokes equations are the Reynolds-averaged Navier-Stokes equations. The Reynolds-averaged Navier-Stokes equations are generally the most complex set of equations that are presently solved. These equations approximate the viscous shearing stresses produced by turbulence by averaging the stresses over a finite time period. The turbulent shear stresses resulting from this averaging process are modeled as a function of the steady state flow conditions at a point. This modeling of turbulence as viscous stresses permits the solution of high Reynolds number flows. Unfortunately, the Reynolds-averaged equations are more time consuming to solve than the laminar equations and there is a lack of a universally reliable turbulence model for the Reynolds-averaged equations. Prandtl's pioneering work in boundary layer theory showed that the viscous shearing terms of the Navier-Stokes equations are large relative to the convective terms only in a thin layer near the boundary. Flow outside the boundary layer is essentially inviscid in most cases. With this result, the Navier-Stokes equations are simplified to the Euler equations by dropping the viscous shear terms and retaining only the convective flow terms. Each time the governing equations are simplified, information about the flow is lost but the equations become substantially easier to solve. If rotationality of the flow is not important, the Euler equations can be reduced to the full potential equation. The full potential equation is irrotational

so there can be no entropy increase through shocks. Full potential flows are limited to weak shocks which can be approximated by isentropic shocks. Transonic flows about slender bodies can be studied using the transonic small disturbance or transonic small perturbation equation. The transonic small disturbance equation uses linearized boundary conditions so the boundaries can be described as planes in the grid generation process. This property permits the use of rectangular grids in solving the transonic small disturbance equation.

As can be easily seen, the level of sophistication of the flow solvers is greatest for the simplest model equations. Typically, the development of flow solvers for model equations proceeds from the simplest test cases, such as two-dimensional cylinder or airfoil flows, to complete three-dimensional aircraft configurations. As such, the flow solvers for the small disturbance equation are most advanced for solving compressible flows about three-dimensional bodies. The first compressible flow solutions of the transonic small disturbance equation were obtained by Murman and Cole [8]. Their work centered on the solution of two-dimensional airfoil flows using a relaxation scheme to solve the governing equation. Murman and Cole's work was the basis of many axisymmetric and three-dimensional transonic small disturbance solvers. South and Brandt [9] applied multi-grid techniques to the transonic small disturbance equation to increase the computational efficiency. Boppe [10-11] extended the algorithm to fully three-dimensional flows about bodies that included wings, fins, and nacelles. Boppe's work was some of the first to solve compressible, transonic flows about very complicated geometries.

The work of Murman and Cole on the transonic small disturbance equation stimulated work on the full potential equation. Jameson [12] developed an efficient and powerful scheme for solving the full potential equation for a wide range of two-dimensional and three-dimensional problems. Jameson and Caughy [13] extended Jameson's approach to a finite-volume scheme. Grossman and Siclari [14] and Shankar et al [15-17] have developed very powerful three-dimensional marching schemes for solving the full potential equation. Both schemes are limited to supersonic flow in the marching direction but are quite fast and efficient computationally since a solution can be generated with one pass through the computational grid in the marching direction. Siclari's method[18-19] solves the non-conservative form of the full potential equation so all shocks are fitted. Shankar's method solves the conservative form of the full potential equation so isentropic shocks can be accurately captured in the solution process. Both Siclari and Shankar's schemes have been used to solve very complex supersonic flows about missiles and fighter aircraft.

While work was progressing on the transonic small disturbance equation and the full potential equation, work was being done on the Euler equations. The Euler equations were much more difficult and expensive to solve than either the transonic small disturbance equation or the full potential equation. The Euler equations are a coupled system of equations and not a single scalar equation. The Euler equations are generally solved by time marching the solution. Relaxation techniques applied to the steady state form of the Euler equations are not used because of odd-even grid point decoupling. Some early Euler solutions were

obtained by Magnus and Yoshihara [20] and Grossman and Moretti [21]. In 1970, MacCormack [22] introduced his explicit predictor-corrector solver. In an effort to improve the efficiency of Euler and Navier-Stokes solvers, Beam and Warming [23] introduced an implicit solver. In 1981, Jameson, Schmidt and Turkel [24] introduced a Runge-Kutta explicit solver for the Euler equations in finite-volume form. Multi-grid techniques[25-26] have been extended to work with the Euler and Navier-Stokes equations and have improved the efficiency of the solutions.

Up until the early 1980's, most work in computational fluid dynamics on the Euler equations had involved the solution of the governing equations by centrally differencing the equations and adding a dissipation term to decouple the instability associated with odd-even grid points. In the early 1980's, characteristic-based schemes for the Euler equations became prevalent. Steger and Warming [27] developed a characteristic based scheme for the Euler equations, known as flux vector splitting. Flux vector splitting split the flux at a point in the flow domain (not at a grid point) into two components. Each component had only non-negative or non-positive eigenvalues. The eigenvalues of the Euler equations are the wave speeds and direction of information propagation in the flow. Each component of the flux can then be upwinded on the basis of the eigenvalue sign. Van Leer [28] improved flux vector splitting so that discontinuities in the mass and energy flux were removed at sonic points in the flow. Roe [29-30] developed his approximate Riemann solver which linearized the Riemann problem, and splits the flux difference between two grid points. Roe's scheme is a so-called flux

differencing scheme and is essentially a linearized Godunov's solver[31]. Roe determined a thermodynamic state between the two grid points which permits the linearized Riemann solver to satisfy the Rankine-Hugoniot shock jump condition exactly. Unfortunately, Roe's scheme admits an expansion shock unless some form of 'fix-up' is applied to the scheme. Enquist and Osher [32] developed a flux difference splitting which satisfied the entropy condition (i.e. excludes non-physical expansion shocks). The Enquist-Osher scheme is still more expensive to use than Roe's scheme, even when some form of entropy fix is applied to Roe's scheme. All of the upwind schemes are first order schemes. Second order accuracy is necessary in regions of the flow outside of shock waves in order to properly resolve expansion fans and contact surfaces. Van Leer [33] introduced MUSCL style differencing to obtain higher order accuracy. In regions of the flow near shockwaves, the accuracy of the upwind schemes is typically limited to first order to avoid spurious oscillations. Many researchers[34-36] developed and applied limiters, which automatically limit the accuracy of MUSCL differencing to first order in regions of the flow near shocks. Several researchers [37-38] have applied conservative upwind schemes to both two and three-dimensional computations, solving both the Euler and Navier-Stokes equations. Moretti developed a non-conservative, characteristic based method known as the Lambda scheme[39-40]. Since the method solves the non-conservative form of the Euler equations, the Rankine-Hugoniot shock jump conditions are not satisfied. To obtain high accuracy in shocked flows, Moretti developed shock fitting algorithms for the Euler equations which he applied in one and two dimensional flows[41].

In the present research the Euler equations were chosen as the model equation to be solved. Most research in the field is currently directed toward the Euler equations and Euler solvers are now considered state of the art. Euler solvers can easily be extended to laminar Navier-Stokes solvers by simply including the viscous stress terms in the governing equations. Upwind schemes were chosen to solve the Euler equations as these schemes have been found to be more accurate than central difference schemes with a scalar dissipation term[42]. In addition, upwind schemes are regarded as being more robust than central difference schemes. In order to obtain an accurate solution with an upwind scheme less tuning of constants is required than with a central difference scheme where the scalar dissipation term is multiplied by a user-supplied constant.

Solutions to fluid dynamic model problems suffer from the fact that the governing equations are solved on a structured grid[43]. This grid or mesh is generated so that the boundary of the mesh is located at the body surface with the mesh covering the complete flow domain. The two-dimensional mesh is then mapped into computational space as a rectangular, square or other regular geometric body. Grid generation works well when the domain is two-dimensional and simply connected. Multiply-connected domains in two-dimensions or complex simply-connected bodies in three-dimensions present much more of a grid generation problem. Control of grid density becomes a significant problem in two dimensions with multiply-connected domains. Multiply-connected domains in two-dimensions may have points concentrated where there are no significant flow gradients while regions with large flow gradients may suffer from a lack of grid

points. Grids about complicated geometric domains in three dimensions typically have poor control of grid point distribution and may have singularities in the mapping process.

Several approaches have been tried in an attempt to solve the grid generation problem. These approaches include the use of adaptive gridding, zonal or multi-block schemes, Cartesian meshes, and finite-element style schemes. Adaptive gridding[44-46] alters the computational grid during the solution process. Some way of estimating solution error is necessary during the solution process. The grid points are moved to flow regions of high error which are typically regions of high flow gradients. This process can lead to highly stretched grids. Adaptive grid schemes are fairly successful in problems that are simple geometrically where the regions of high flow gradients are small or easily predicted. Very complex flows in three-dimensions, generally do not benefit from adaptive gridding because too much grid stretching occurs. Multi-block or zonal schemes operate by breaking the flow domain into sub-domains of structured meshes which are patched together at the grid interfaces. A variety of researchers [47-52] have been quite successful in patching structured grids together in order to cover the entire computational domain. Benek, Buning, and Steger [53] have overlapped structured grids in three-dimensions. Multi-block schemes essentially use a group of structured grids coupled together into a globally unstructured mesh. The use of a globally unstructured mesh requires that the flow solver store the grid connectivity of the sub-domains. Cartesian meshes[54-57] have been used to solve both the full-potential and Euler equations in both two and three dimen-

sions. Cartesian meshes are simply rectangular meshes which are overlaid over the flow domain. Grid points in Cartesian meshes do not align with the body surface so some special treatment of the boundary conditions is necessary at the body surface. Poor control of grid spacing is also a problem with Cartesian meshes. Only one-dimensional grid stretching can be used in each coordinate direction which means that the mesh tends to be very highly stretched in the outer regions of the flow domain.

Two groups in Europe, one at the University of Wales [58-60] and one at INRIA [61-62] in France, have been doing research on finite element schemes for high speed flows. These groups routinely solve flows about two and three-dimensional bodies on completely unstructured meshes of triangles and tetrahedrons, respectively. A completely unstructured mesh of triangles or tetrahedrons means that all grid connectivity information has to be stored by the flow solver. The lack of any inherent grid structure means that the researcher has complete control of grid point distribution in the flow domain. Increased computer storage and indirect addressing will be necessary in computer codes that use unstructured grids. Increased data storage and indirect addressing implies that a unstructured grid flow solver will run slower than a structured grid code. Many very fast and efficient solution schemes developed for structured grid codes, such as Approximate Factorization[63] and Vertical Line Gauss-Seidel [64], can not even be implemented on unstructured meshes. Generally finite-element solution techniques are limited to explicit time-integration schemes or point implicit schemes (such as point Jacobi or point Gauss-Seidel).

Recently, A. Jameson [65-67] and co-workers have applied centrally differenced, finite-volume schemes to meshes composed of triangles and tetrahedrons. These schemes store flow variables at the vertices of the triangles or tetrahedrons. Fluid fluxes are integrated about the edges of the triangles/tetrahedrons surrounding a vertex using trapezoidal integration. Jameson's scheme uses a scalar dissipation coefficient to ensure convergence and prevent spurious oscillations. Little research has been done on using upwind schemes on unstructured meshes of triangles/tetrahedrons.

The present research is an attempt to extend upwind schemes to unstructured meshes composed of triangles or tetrahedrons. This research includes the extension of upwind schemes to second and third order accuracy using MUSCL differencing on unstructured grids. The present work could be considered the logical extension of Jameson's scheme to upwind differencing. The present scheme utilizes a dual mesh which is the tesselate of the triangular grid. The fluid fluxes are integrated about the sides of the dual mesh. The edges or sides of the triangles lie approximately normal to the edges of the dual mesh and hence an upwind scheme can be applied along the edge of the triangular mesh.

While the present work generalizes to three-dimensions only two-dimensional problems have been solved. Two-dimensional problems were considered adequate for the initial stages of the present work. Several solutions are made on very simple geometric configurations such as a reflecting shock, a simple wedge inlet, and a hypersonic blunt body. These problems are quite simple and struc-

tured grids can easily be generated for these problems. Nevertheless, it was considered necessary to run these simple test cases using an unstructured grid to judge the accuracy obtainable by the upwind solver on an unstructured mesh. Also, the shock reflection and the wedge inlet have analytical solutions. Further test cases involve transonic single element airfoils and a multi-element low-speed airfoil. Airfoils were considered to be very good test cases because there are many Euler solutions available for these particular airfoils and airfoils are very sensitive to solution accuracy.

While the computational fluid dynamics researcher has complete control over the mesh and the mesh spacing when using a grid composed of triangles, automatic generation of the grid is not an inconsequential task. The grids used in this research were generated using the automatic grid generation program developed by R. Löhner [68-69]. R. Löhner's grid generation program uses the advancing front concept and has been extended to three dimensions. The code gives excellent control of grid spacing and mesh point placement.

II. MATHEMATICAL FORMULATION

2.1 Euler Equations

The Euler equations are a coupled set of equations which express the conservation of mass, momentum, and energy in a fluid flow. The equations are valid for an inviscid, adiabatic, compressible, unsteady fluid. The equations are derived for any fluid which can be regarded as a continuous medium, so they are valid for reacting and non-reacting flows. The Euler equations in a conservative, Cartesian, differential form are:

$$\frac{\partial Q}{\partial t} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \tag{2.1}$$

with

(2.2)
$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}$$

and

(2.3)
$$f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix} \qquad g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}$$

The density is ρ, the velocity vector is $\vec{U} = (u\hat{i}, v\hat{j})$, the total energy per unit volume is E, the pressure is p, the orthogonal unit vectors are $(\hat{i}, \hat{j})$. An equation of state is necessary to close the system of equations. Using the perfect gas law, the pressure can be related to the variables in the vector Q as:

(2.4)
$$p = (\gamma - 1)[E - \rho \frac{(u^2 + v^2)}{2}]$$

where gamma is the ratio of specific heats and is taken to be 1.4 in air. The governing equations have been presented in non-dimensional form. The density is non-dimensionalized by ρ_∞, velocity by $|\vec{U}_\infty| = \sqrt{u_\infty^2 + v_\infty^2}$, and both the pressure and energy per unit volume by $\rho_\infty(\vec{U}_\infty \cdot \vec{U}_\infty)$. It is interesting to note that with this non-dimensionalization, the form of the Euler equations remains un-

changed. Numerically, it is important to deal with the equations in a non-dimensional form, such that the non-dimensionalized variables are all of the same order of magnitude. During numerical computations, use of variables which are all of approximately the same order will prevent loss of accuracy due to numerical round-off error. Use of non-dimensionalized variables will ensure that problems relating to a consistent use of units will be eliminated. Non-dimensional variables will reduce programming and user errors during development and program input.

The equations are presented in strong conservation law form (SCLF) in conservative variables Q, as opposed to a non-conservation form using primitive variables: $[\rho, u, v, p]^T$. In the present research, all shock waves that occur in the flow are captured numerically as discontinuities in the flow field during the solution process. It has been shown in reference [70] that only the SCLF of the governing equations can generate a weak solution of the differential equations and satisfy the Rankine-Hugoniot shock jump relations. So in order to capture shocks in the flow domain which are physically correct, it is necessary to use the SCLF of the governing equations. Following the work of Viviand [71], the Euler equations in conservative, Cartesian form can be transformed into a non-orthogonal coordinate system while still maintaining SCLF. If a transformation from a Cartesian coordinate system (x,y), to a general, curvilinear coordinate system (ξ, η), as in:

$$\xi = \xi(x,y) \qquad \eta = \eta(x,y) \tag{2.5}$$

is used, then the Euler equations in SCLF can be written as:

$$(2.6) \qquad \left(\frac{Q}{J}\right)_t + \left[\frac{\xi_x f + \xi_y g}{J}\right]_\xi + \left[\frac{\eta_x f + \eta_y g}{J}\right]_\eta = 0$$

where J is the Jacobian of the transformation and is equal to:

$$(2.7) \qquad J = \xi_x \eta_y - \xi_y \eta_x = (x_\xi y_\eta - x_\eta y_\xi)^{-1}$$

The flux vector in a general, curvilinear coordinate system can be rewritten using contravariant velocities:

$$(2.8) \qquad \frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{f}}{\partial \xi} + \frac{\partial \hat{g}}{\partial \eta} = 0$$

where

$$(2.8) \qquad \hat{Q} = \frac{Q}{J} \qquad \hat{f} = J^{-1}\begin{bmatrix} \rho\overline{U} \\ \rho u\overline{U} + \xi_x p \\ \rho v\overline{U} + \xi_y p \\ \overline{U}(E+p) \end{bmatrix} \qquad \hat{g} = J^{-1}\begin{bmatrix} \rho\overline{V} \\ \rho u\overline{V} + \eta_x p \\ \rho v\overline{V} + \eta_y p \\ \overline{V}(E+p) \end{bmatrix}$$

The non-normalized, contravariant velocities are:

$$(2.9) \qquad \overline{U} = \xi_x u + \xi_y v \qquad\qquad \overline{V} = \eta_x u + \eta_y v$$

The governing equations in a general, curvilinear coordinate system are still non-dimensionalized as in the Cartesian coordinate system. In order for a finite-difference implementation of the differential equations to remain conserva-

tive, the mapping metrics $\xi_x, \xi_y, \eta_x, \eta_y$ must be carefully evaluated in the computational domain. Even then, there is no guarantee that the scheme will be globally conservative using a finite-difference formulation.

2.2 Integral form of the Euler equations

A method which will guarantee global conservation of fluid fluxes is the finite-volume method. The finite-volume method is simply another way of discretizing the governing equations. When using a finite-volume approach, it is necessary to use the integral form of the governing equations. The integral form can be directly derived from first principles as a conservation law. The derivation of the integral form makes no assumption about the mathematical continuity of the fluid dynamic state variables. The differential form of the governing equations can be derived from the integral form using Gauss's theorem. Gauss's theorem makes the assumption that the fluid dynamic variables are mathematically continuous. If the governing differential equation is in pure divergence form (SCLF), the discretized version of the differential equation will be consistent with the conservation law form. Then the discretized differential form will yield the correct shock jump as a mathematical discontinuity. The integral form is:

$$\frac{d}{dt}\iiint_R Q\, d\tau + \iint_S \vec{F} \cdot \hat{n}\, dS = 0 \tag{2.10}$$

where R is the area in two dimensions and the volume in three dimensions of the computational region. The variable S is a closed line in two dimensions and a closed surface in three dimensions and is the boundary of domain R. The flux vector, $\vec{F}$, is $(f\hat{i} + g\hat{j})$ in two dimensional space. The unit normal vector $\hat{n}$ is an outward facing vector to the boundary S. The definition and non-dimensionalization of the other variables remains the same as in section (2.1).

2.3 Physical Boundary Conditions

When solving a differential or integral equation, the solution is completely driven by the boundary conditions. The physical boundary conditions are the boundary conditions that are necessary to close the mathematical governing equations. The physical boundary conditions are not all that is required in a numerical model of the governing equations. Both numerical and physical boundary conditions are needed to close the algebraic equations which represent the governing equations. The numerics involved in closing the algebraic equations will be discussed in chapter three.

An initial condition is necessary to begin the time integration of the unsteady form of the Euler equations. The initial condition must be specified throughout the flow domain. In the present research, free-stream conditions at infinity were

specified everywhere in the flow domain. Using the non-dimensionalization presented in section (2.1), the free-stream conditions are:

$$\rho_\infty = 1 \qquad \vec{U}_\infty = (\cos\alpha\hat{i}, \sin\alpha\hat{j}) \qquad p_\infty = \frac{\rho_\infty}{\gamma M_\infty^2} \tag{2.11}$$

The angle α is the angle that the free-stream flow makes with the local (x, y) body axes, and is generally referred to as the angle of attack.

Along with the initial condition, boundary conditions must be specified along the edge of the flow domain. The boundary conditions along the edge of flow domain can be divided into two groups. One is a wall boundary condition (solid surface boundary condition) and the other is a far-field boundary condition.

For an Euler solver, the only boundary condition at a wall is the specification of of flow tangency, as in:

$$\vec{U} \cdot \hat{n} = 0 \tag{2.12}$$

where $\hat{n}$ is the outward facing normal to the wall. This boundary condition states that there can be no flow through the wall.

For external flows, the far-field boundary conditions are the free-stream conditions given in equation (2.11). A one-dimensional characteristic-based analysis normal to the edge of the computational mesh will show that prescribing all flow variables is incorrect [72]. In a two-dimensional flow, a subsonic inflow requires

that only three of the four flow variables be specified. The remaining flow variable is extrapolated out of the computational domain. For subsonic outflow, three variables are extrapolated and one is specified. For supersonic inflow, all variables are specified, while for supersonic outflow, all variables are extrapolated.

For upwind schemes, it is generally sufficient to specify the far-field boundary conditions. Upwind schemes are characteristic based and will take what information is necessary from the boundaries and ignore the rest. Central difference schemes are not characteristic based and specification of all flow variables at the far-field boundary will lead to a poorly posed problem. Prescribing all far-field boundary conditions in an upwind-based flow solver will work best for external flows, where the far-field boundary is located a large distance from the wall. Internal flows generally require that characteristic based boundary conditions be used for the inflow and outflow so that the solution at the boundary agrees with the rest of the flow field.

2.4 Far-Field Airfoil Boundary Conditions

Solutions about lifting and non-lifting airfoils generate disturbances in the flowfield which extend very long distances from the airfoil. Lifting solutions of the Euler equations generate a vortex about the airfoil which extends to infinity.

Computational meshes used to generate solutions to the governing equations do not extend to infinity. An Euler solver has no source of dissipation other than that of numerical viscosity to remove the vortex from the flowfield. Many chords from the airfoil, a vortex created during the solution procedure will still be existent. Free stream boundary conditions applied at the boundaries of computational meshes cannot account for the vortex. As a consequence, many solutions of the Euler and Navier-Stokes equations about lifting airfoils use grids which extend 50-100 chords from the airfoil [73].

The need to have a mesh that extends 50-100 chord lengths away from the airfoil is at odds with the requirement that a very small mesh spacing be used near the leading and trailing edges to resolve the flow gradients. Most researchers either use an excessive number of grid points in the flowfield which leads to long execution times or use large amounts of grid stretching, leading to grid cells with a high aspect ratio which lowers accuracy and can decrease the convergence rate.

An alternative is to use a simple model of the airfoil vortex as part of the far-field boundary condition. If the far-field boundary conditions include the effect of the vortex, the extent to which the grid extends away from the body can be greatly reduced. This higher order far-field boundary condition will have the effect of increasing solution accuracy while decreasing computational cost. Reference [72] claims that accurate computations can be made on grids that extend less than five chords from the airfoil. This statement is valid for completely subsonic airfoils or transonic airfoils with only small regions of imbedded supersonic flow. Test

cases which involve transonic airfoils near Mach one may have regions of supersonic flow that extend quite far from the airfoil. Large regions of supersonic flow dictate that the computational mesh be extended so that the edge of the grid is well away from such regions.

Correcting the far-field boundaries so they include the higher-order effect of a compressible vortex and doublet is not unknown. Virtually all solutions of the transonic potential equation about airfoils use some form of asymptotic boundary condition for the far-field. Murman and Cole [8] used an expansion of the solution of the nonlinear small disturbance equation for a flat plate airfoil as a far-field boundary condition.

Much of this analysis follows the approach taken by Thomas and Salas [72]. We assume that the flow in the far-field can be represented by the linear small disturbance potential equation:

$$(1 - M_\infty^2)\phi_{xx} + \phi_{yy} = 0 \tag{2.12}$$

where ϕ is the perturbation velocity potential and x and y are distances tangential and normal to the free-stream velocity. M_∞ is the free-stream Mach number. Equation (2.12) can be reduced to an incompressible form:

$$\phi'_{x'x'} + \phi'_{y'y'} = 0 \tag{2.13}$$

using a affine transformation where

(2.14) $$x' = x \qquad y' = \beta y$$

and $\beta^2 = 1 - M_\infty^2$. The transformed variables are related to the non-transformed variables by:

(2.15) $$u = \frac{u'}{\beta^2} \qquad v = \frac{v'}{\beta}$$

(2.16) $$\phi = \frac{\phi'}{\beta^2} \qquad \Gamma = \frac{\Gamma'}{\beta^2}$$

where u and v are the perturbation velocities in wind axes and Γ is the circulation.

The perturbation velocity components due to an incompressible, irrotational vortex [74] in transformed variables is:

(2.17) $$u' = \frac{\Gamma'}{2\pi} \frac{y'}{x'^2 + y'^2} \qquad v' = -\frac{\Gamma'}{2\pi} \frac{x'}{x'^2 + y'^2}$$

Transforming this equation back into normal variables gives a perturbation velocity for a compressible vortex which can be used in a far-field boundary condition, such that:

(2.18) $$u = \frac{\Gamma}{2\pi} \frac{\beta y}{x^2 + \beta^2 y^2} \qquad v = \frac{-\Gamma}{2\pi} \frac{\beta x}{x^2 + \beta^2 y^2}$$

Thomas and Salas [72] used a similiar approach except they solved the transformed small disturbance equation for an airfoil with linearized boundary condi-

tions. Using a series solution approach, they obtained a Laurent series. Analyzing the Laurent series, they found that if the coordinate origin was located at the airfoil's quarter chord point, the leading term of the series solution was a compressible vortex.

The circulation Γ is determined from the airfoil lift coefficient, as:

$$\Gamma = \frac{c_l \, c \, |\vec{U}_\infty|}{2} \tag{2.19}$$

where c_l, c, and $|\vec{U}_\infty|$ are the lift per unit span, chord length and total speed at infinity, respectively. Clearly, integrating momentum and pressure about any closed contour that surrounds the airfoil is going to give an invariant lift and drag. Equations (2.18) and (2.19) are only truly valid for irrotational flows and cannot be applied directly to Euler flows which may have shockwaves present and thus are not irrotational. The circulation computed on closed intervals about the airfoil will vary if a shock wave is present. Nevertheless, as closed contours about the airfoil are chosen at greater and greater distances from the airfoil, the circulation computed about the contour will asymptote to a constant value. The circulation computed about a closed interval far from the airfoil will be equivalent to the circulation found from equation (2.19).

Typically, the axial and normal forces acting on the airfoil are found by integrating the pressure on the airfoil surface, using:

(2.20) $$c_a = \frac{-2}{\rho_\infty c \vec{U}_\infty \cdot \vec{U}_\infty} \oint_S (\rho u(\vec{U} \cdot \hat{n}) + p\hat{n}_x)\, dS$$

(2.21) $$c_n = \frac{-2}{\rho_\infty c \vec{U}_\infty \cdot \vec{U}_\infty} \oint_S (\rho v(\vec{U} \cdot \hat{n}) + p\hat{n}_y)\, dS$$

where c_a and c_n are the normalized forces per unit span and $(\hat{n}_x, \hat{n}_y)$ are the components of the outward facing unit vector $\hat{n}$. The unit vector $\hat{n}$ is normal to the closed interval S. Once c_a and c_n are known c_l and c_d can be easily computed by rotating into a coordinate system normal and tangential to the free stream velocity vector.

(2.22) $$c_l = c_n \cos\alpha - c_a \sin\alpha \qquad c_d = c_n \sin\alpha + c_a \cos\alpha$$

Once c_l is known, Γ is known and can be used to compute the perturbation velocities in equation (2.18).

Substituting equation (2.19) into equation (2.18), rotating the coordinate system and perturbation velocities of equation (2.18) into a body fixed axis system, and summing the perturbation velocities with the free-stream velocity at angle of attack, α, results in:

(2.23) $$u_{ff} = \left|\vec{U}_\infty\right|(\cos\alpha + D\sin\theta) \qquad v_{ff} = \left|\vec{U}_\infty\right|(\sin\alpha - D\cos\theta)$$

where

$$D \equiv \frac{c_l\, c\, \beta}{4\pi R}\left[1 - M_\infty^2 \sin^2(\theta - \alpha)\right]^{-1} \tag{2.24}$$

and R and θ are the radius and polar angle, respectively, in a body oriented axis system. The angle θ is measured counterclockwise from the chord line facing aft of the airfoil quarter-chord point. The new far-field velocity components are u_{ff} and v_{ff}. Clearly, the far-field boundary condition is simply a higher order approximation of the free-stream boundary condition.

The velocity components in the far-field have been modified so the free-stream density and pressure are no longer the proper far-field boundary conditions for the edge of the computational mesh. Assuming constant total enthalpy and constant entropy, the proper far-field density and pressure may be found for the edge of the grid.

$$p_{ff} = \left[p_\infty^{\frac{\gamma-1}{\gamma}} + \rho_\infty \left(\frac{\gamma - 1}{\gamma} \right) \left[\frac{u_\infty^2 + v_\infty^2 - (u_{ff}^2 + v_{ff}^2)}{2 p_\infty^{1/\gamma}} \right] \right]^{\frac{\gamma}{\gamma-1}} \tag{2.25}$$

$$\rho_{ff} = \rho_\infty \left(\frac{p_{ff}}{p_\infty} \right)^{1/\gamma} \tag{2.26}$$

The combination of $\rho_{ff}, u_{ff}, v_{ff}$ and p_{ff} are a complete set of far-field boundary conditions, with a compressible vortex correction, for an airfoil grid.

In an inviscid flow, the circulation about an airfoil is not uniquely determined by the far-field or wall boundary conditions. The rear stagnation point may exist anywhere on the airfoil. A separate boundary condition is necessary to uniquely determine the circulation in the flow field. The Kutta condition is the boundary condition that uniquely determines the circulation in the flow field. The Kutta condition states that the strength of the circulation in the flow field must be just sufficient to force the flow to leave the airfoil smoothly at the trailing edge. Any other value of circulation will move the rear stagnation point away from trailing edge and will cause the velocity at the trailing edge to be infinite and thus singular.

III. NUMERICAL FORMULATION

3.1 Discretization of Governing Equations

We assume the two-dimensional flow domain Ω can be discretized into a group of triangular polygons. The vertices of the triangles in domain Ω are denoted by V_i, where subcript i denotes the i-th vertex in the set of vertices. A dual mesh is defined by breaking each triangle into three quadrilaterals. In the present research, the quadrilaterals were formed by connecting the centroid of each triangle to the midpoints of the three sides of the triangle. Thus the dual mesh cells are formed from the medians of the triangles. A dual mesh cell C_i is defined to be the collection of quadrilaterals sharing the same vertex V_i (see Figure 2). Each dual mesh cell is a finite volume. There is a one to one correspondence between V_i and C_i, with V_i being located approximately in the center of dual mesh cell C_i. The boundary of a dual mesh cell C_i is denoted by ∂C_i. The neighboring vertices of

vertex V_i are contained in a list denoted by κ_i. The number of vertices surrounding each vertex of the triangulation is variable, so the number of neighboring vertices contained in κ_i is also variable.

Flow variables are stored at the vertices V_i and conservation is enforced about the boundary ∂C_i of every dual mesh cell C_i. If we assume that the triangular mesh is geometrically time invariant, the flow variables stored at the vertices V_i can be area averaged according to the formula:

$$(3.1) \qquad <Q_i> = \frac{1}{S(C_i)} \int_{\Omega(C_i)} Q \, dS$$

where $<Q_i>$ represents the area average of flow variables in dual mesh cell C_i and $S(C_i)$ represents the area of C_i, Using an area averaged $<Q_i>$, equation (2.10) can be written for every dual mesh cell C_i.

$$(3.2) \qquad S(C_i) \frac{\partial <Q_i>}{\partial t} = - \oint_{\partial C_i} (f \hat{i} + g \hat{j}) \cdot \hat{n} \, dl$$

where $S(C_i)$ is the area of C_i.

To evaluate the right hand side of (3.2), we sum the flux through the dual cell boundary ∂C_i for each dual mesh cell C_i. Each individual cell side flux that contributes to the flux integral in (3.2) shares a one to one relationship with the edges of the triangular mesh that meet at V_i. Each edge of the triangular mesh

that shares V_i as a vertex is connected to one of the vertices in list κ_i. The resulting flux integral evaluates to:

$$\oint_{\partial C_i} (f\hat{i} + g\hat{j}) \cdot \hat{n}\, dl = \sum_{j=\kappa_i} F_{i,j} \Delta l_{i,j} \tag{3.3}$$

where $F_{i,j}$ is the numerical approximation for the flux associated with the triangular edge connecting V_i and V_j (see Figure 2), $\Delta l_{i,j}$ is the length of the cell side of C_i , associated with same triangular edge, and κ_i is a list of vertices surrounding V_i, with the individual vertex in the list denoted by j.

In order to evaluate $F_{i,j}$ using an upwind scheme it is necessary to have two fluid dynamic states, and a splitting direction if the upwind scheme is being used in multiple dimensions. Observing Figure 3, points V_i and V_j are the two vertices that form an edge of the triangular mesh while G_{ijk} and G_{ijl} are the centroids of two triangles that share edge $\overline{V_iV_j}$. A segment of the boundary ∂C_i of the dual mesh cell which connects G_{ijk} and G_{ijl} (the medians) is broken into two line segments. Denoting M_{ij} as the midpoint of $\overline{V_iV_j}$, $\hat{v}_{ij}^k$ is the unit normal to $\overline{M_{ij}G_{ijk}}$ and $\hat{v}_{ij}^l$ is the unit normal to $\overline{G_{ijl}M_{ij}}$. Both unit vectors are directed toward V_j.

An area averaged unit vector can easily be found for the two median lines as can a boundary length $\Delta l_{i,j}$ (a segment of ∂C_i).

$$\vec{\eta}_{ij} \cdot \hat{i} = (\hat{v}_{ij}^k \cdot \hat{i}) |\overline{M_{ij}G_{ijk}}| + (\hat{v}_{ij}^l \cdot \hat{i}) |\overline{G_{ijl}M_{ij}}|$$

$$\vec{\eta}_{ij} \bullet \hat{j} = (\hat{v}^k_{ij} \bullet \hat{j}) \left| \overline{M_{ij} G_{ijk}} \right| + (\hat{v}^l_{ij} \bullet \hat{j}) \left| \overline{G_{ijl} M_{ij}} \right|$$

$$\Delta l_{i,j} = |\vec{\eta}_{ij}| = \sqrt{(\vec{\eta}_{ij} \bullet \hat{i})^2 + (\vec{\eta}_{ij} \bullet \hat{j})^2} \tag{3.4}$$

$$\hat{\eta}_{ij} = \left[\frac{(\vec{\eta}_{ij} \bullet \hat{i})}{|\vec{\eta}_{ij}|} \hat{i}, \frac{(\vec{\eta}_{ij} \bullet \hat{j})}{|\vec{\eta}_{ij}|} \hat{j} \right]$$

The x,y subscripts indicate vector components in the $(\hat{i}, \hat{j})$ Cartesian system with $\hat{\eta}_{ij}$ being the area averaged unit vector normal to the two median lines connecting G_{ijl} and G_{ijk}. The splitting direction used in Roe's approximate Riemann solver is $\hat{\eta}_{ij}$. An area averaged vector and length $\Delta l_{i,j}$ are used to avoid the computational work of computing two fluxes (one for each median).

3.2 Roe's Approximate Riemann Solver

Upwind schemes are theoretical methods which attempt to exploit the underlying mathematical structure of the governing differential or integral equations to produce more accurate and robust numerical methods for fluid flow. The underlying structure of the Euler equations can be modeled as as series of interacting waves all moving at a characteristic speed and carrying characteristic information. As such, upwind schemes can be said to be characteristic based.

Central difference schemes, which are based on numerics rather than physics, require that a dissipation or viscosity term be computed and explicitly added to the central difference terms to prevent odd-even decoupling of the central derivatives on the computational grid. The amount of artificial viscosity that needs to be added to a central difference scheme to obtain convergence and a smooth, non-oscillatory solution, cannot be determined apriori. Numerical viscosity is a natural part of upwind schemes and does not need to be explicitly added. This feature of upwind schemes makes them robust and tends to make them more naturally monotonic about discontinuities(shocks) in the flow domain. A direct comparison of upwind schemes with central difference schemes, found the upwind schemes to be more accurate than central difference schemes with Roe's approximate Riemann solver to be the most accurate [42].

Implicit schemes for upwind methods are more diagonally dominant than central difference methods, so that powerful relaxation techniques can be used to solve the algebraic equations that result from an upwind method.

In fluid mechanics, the Riemann problem is the wave interaction between two uniform fluid dynamic states. S.K. Godonuv [31] was the first to apply a Riemann solver to one-dimensional compressible Euler flow. Godonuv's solver was an exact solution of the Riemann problem between every adjacent pair of grid points in a one-dimensional computational domain. An exact solution of the Riemann problem requires an iterative solution of the wave interaction for each pair of fluid states. An iterative solution of the Riemann problem is a computa-

tional extravagance when the initial data for the solver is only known to first-order accuracy. P.L. Roe proposed a linearized solution of the Riemann problem [29] which could be solved directly.

Roe's method was derived for only a one-dimensional interaction of characteristic waves. The one-dimensional, unsteady Euler equations can be written as:

$$\frac{\partial Q}{\partial t} + \frac{\partial f}{\partial x} = 0 \tag{3.5}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix} \qquad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix} \tag{3.6}$$

and $E = \rho(\varepsilon + u^2/2)$ and ε is the internal energy of the gas. After closing the system of equations with the perfect gas law, equation (3.5) can be linearized as:

$$\frac{\partial Q}{\partial t} + \frac{\partial f}{\partial Q}\frac{\partial Q}{\partial x} = \frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} = 0 \tag{3.7}$$

where A is the Jacobian matrix of the flux vector with respect to Q. We wish to find exact solutions to this approximation of the governing equation. Roe replaced matrix A with a locally constant matrix $\tilde{A}$ in equation (3.7) and imposed four conditions on $\tilde{A}$ that were:

(i) Vector space Q linearly maps to vector space f.

(ii) As $Q_L \to Q_R \to Q$, $\tilde{A}(Q_L, Q_R) \to A(Q)$, where $A = \partial f / \partial Q$.

(iii) For any Q_L, Q_R, $\tilde{A}(Q_L, Q_R) \times (Q_R - Q_L) = f_R - f_L$.

(iv) The eigenvectors of $\tilde{A}$ are linearly independent.

The subscripts R and L indicate a right and left fluid state, respectively. Condition (ii) maintains consistency with the governing differential equation. Condition (iii) ensures that $\tilde{A}$ satisfies the Rankine-Hugoniot shock jump condition. The Roe-averaged matrix $\tilde{A}$ is matrix A evaluated at the Roe-averaged state. Matrix $\tilde{A}$ is:

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 \\ -(3-\gamma)\tilde{u}^2/2 & (3-\gamma)\tilde{u} & \gamma - 1 \\ -\dfrac{\tilde{u}\tilde{a}^2}{\gamma - 1} + (\gamma - 2)\dfrac{\tilde{u}^3}{2} & \dfrac{\tilde{a}^2}{\gamma - 1} + (3 - 2\gamma)\dfrac{\tilde{u}^2}{2} & \gamma\tilde{u} \end{bmatrix} \tag{3.8}$$

All variables with tildas are evaluated at a Roe-averaged state. Conditions (i) and (iv) ensure that the matrix $\tilde{A}$ has three independent eigenvalues. As such, matrix $\tilde{A}$ can be factored as:

$$\tilde{A} = \tilde{S}\ \tilde{\Lambda}\ \tilde{S}^{-1} \tag{3.9}$$

where $\tilde{\Lambda}$ is the diagonal matrix of wave speeds:

$$\tilde{\Lambda} = \begin{bmatrix} \tilde{u} - \tilde{a} & 0 & 0 \\ 0 & \tilde{u} & 0 \\ 0 & 0 & \tilde{u} + \tilde{a} \end{bmatrix} \tag{3.10}$$

and $\tilde{S}$ and $\tilde{S}^{-1}$ are the right and left eigenvector matrices, respectively:

$$(3.11)\qquad \tilde{S} = \begin{bmatrix} 1 & 1 & 1 \\ \tilde{u} - \tilde{a} & \tilde{u} & \tilde{u} + \tilde{a} \\ \frac{\tilde{a}^2}{\gamma - 1} + \tilde{u}\left(\frac{\tilde{u}}{2} - \tilde{a}\right) & \frac{\tilde{u}^2}{2} & \frac{\tilde{a}^2}{\gamma - 1} + \tilde{u}\left(\frac{\tilde{u}}{2} + a\right) \end{bmatrix}$$

$$\tilde{S}^{-1} = \begin{bmatrix} \frac{\tilde{u}}{2\tilde{a}}\left(1 + \frac{\gamma - 1}{2}\frac{\tilde{u}}{\tilde{a}}\right) & -\frac{1}{2\tilde{a}}\left(1 + (\gamma - 1)\frac{\tilde{u}}{\tilde{a}}\right) & \frac{\gamma - 1}{2\tilde{a}^2} \\ 1 - \frac{\gamma - 1}{2}\frac{\tilde{u}^2}{\tilde{a}^2} & (\gamma - 1)\frac{\tilde{u}}{\tilde{a}^2} & -\frac{(\gamma - 1)}{\tilde{a}^2} \\ -\frac{\tilde{u}}{2\tilde{a}}\left(1 - \frac{\gamma - 1}{2}\frac{\tilde{u}}{\tilde{a}}\right) & \frac{1}{2\tilde{a}}\left(1 - (\gamma - 1)\frac{\tilde{u}}{\tilde{a}}\right) & \frac{\gamma - 1}{2\tilde{a}^2} \end{bmatrix}$$

Roe determined the appropriate Roe-averaged variables, which are an average of the left and right fluid states that insure that the linearized solver satisfies conditions (i)-(iv), in reference [29] using parameter vectors. Virtually any average of the left and right fluid states will satisfy conditions (i), (ii), and (iv). Condition (iii), the Rankine-Hugoniot shock jump condition, is the only condition not satisfied by a simple algebraic averaging of the left and right fluid states. The approach taken by Roe and Pike [75] was to expand the matrix equations of condition (iii) and solve the resulting set of algebraic equations directly. If we multiply $\tilde{A}$ by $\Delta(Q)$, where $\Delta(\bullet) = (\bullet)_R - (\bullet)_L$, the first two equations can be written as:

$$(3.12)\qquad \Delta(\rho u) = \tilde{\rho}\Delta u + \tilde{u}\Delta\rho$$

$$(3.13)\qquad \Delta(\rho u^2) = 2\tilde{\rho}\tilde{u}\Delta u + \tilde{u}^2\Delta\rho$$

Substituting $\tilde{\rho}$ from equation (3.12) into equation (3.13) results in a quadratic equation for $\tilde{u}$:

$$\tilde{u}^2\Delta\rho - 2\tilde{u}\Delta(\rho u) + \Delta(\rho u^2) = 0 \tag{3.14}$$

If we substitute the definition for $\Delta(\bullet)$ and solve for the negative root of $\tilde{u}$ we obtain:

$$\tilde{u} = \frac{\rho_L^{1/2}u_L + \rho_R^{1/2}u_R}{\rho_L^{1/2} + \rho_R^{1/2}} \tag{3.15}$$

Substituting equation (3.15) back into equation (3.12) we obtain for $\tilde{\rho}$:

$$\tilde{\rho} = (\rho_L\rho_R)^{1/2} \tag{3.16}$$

These results are sometimes referred to as 'square root averaging' or 'Roe' averaging. The third equation that results from multiplying $\tilde{A}\Delta Q$ is essentially an expression for the Roe-averaged speed of sound. Substituting equations (3.12) and (3.13) into the third equation gives:

$$\tilde{\rho}\tilde{a}^2\Delta u = \gamma[\Delta(up) - \tilde{u}\Delta p] + \frac{(\gamma-1)}{2}[\Delta(\rho u^3) - \tilde{u}^3\Delta\rho - 3\tilde{\rho}\tilde{u}^2\Delta u] \tag{3.17}$$

Expanding the right hand side of equation (3.17) using the ideal gas law and equations (3.15) and (3.16), we have:

$$\rho_L\rho_R\tilde{a}^2 = \gamma\frac{\rho_L p_R + \rho_R p_L}{\rho_L + \rho_R} + \frac{\gamma-1}{2}\frac{\rho_L^2\rho_R^2(u_R - u_L)^2}{(\rho_L + \rho_R)^2} \tag{3.18}$$

The above expression can be simplified considerably by introducing a Roe-averaged total enthalpy. Defining the Roe-averaged total enthalpy as:

$$\tilde{H} = \frac{\tilde{a}^2}{\gamma - 1} + \frac{\tilde{u}^2}{2} \tag{3.19}$$

and substituting $\tilde{a}$ from equation (3.18), we arrive at:

$$\tilde{H} = \frac{\rho_L^{1/2} H_L + \rho_R^{1/2} H_R}{\rho_L^{1/2} + \rho_R^{1/2}} \tag{3.20}$$

where

$$H_L = \frac{a_L^2}{\gamma - 1} + \frac{u_L^2}{2} \qquad H_R = \frac{a_R^2}{\gamma - 1} + \frac{u_R^2}{2}$$

The Roe-averaged speed of sound can be computed from $\tilde{H}$ and $\tilde{u}$ much more quickly than using equation (3.18).

Once the Roe-averaged conditions are known, it is possible to evaluate $\tilde{A}$ and find the flux present at the interface between the left and right fluid states. The flux at the interface can be upwinded using either the left or right fluid states, as:

$$f(Q_L, Q_R) = f_L + \tilde{S}\,\tilde{\Lambda}^{(-)}\,\tilde{S}^{-1}\,\Delta Q \tag{3.21}$$

$$f(Q_L, Q_R) = f_R - \tilde{S}\,\tilde{\Lambda}^{(+)}\,\tilde{S}^{-1}\,\Delta Q \tag{3.22}$$

where $\Lambda^{(-)}$ and $\Lambda^{(+)}$ are:

(3.23) $$\tilde{\Lambda}^{(+)} = \frac{\tilde{\Lambda} + |\tilde{\Lambda}|}{2} \qquad \tilde{\Lambda}^{(-)} = \frac{\tilde{\Lambda} - |\tilde{\Lambda}|}{2}$$

where $|\tilde{\Lambda}|$ is the matrix of absolute values of the wave speeds. Equations (3.21) and (3.22) can be averaged to give:

(3.24) $$f(Q_L, Q_R) = \frac{1}{2}\left(f_L + f_R - \tilde{S}\,|\tilde{\Lambda}|\,\tilde{S}^{-1}\,\Delta Q\right)$$

The flux $f(Q_L,Q_R)$ is the flux present at the interface between the two Riemann states. In a numerical scheme, $f(Q_L,Q_R)$ is used to represent the flux found at the interface between grid cells. The volume-averaged flow variables stored at the cell centers represent the Riemann states.

The exact solution and Roe's approximate solution of the Riemann problem are truly limited to the one-dimensional form of the governing equations. Nevertheless, one-dimensional Riemann solvers have been applied to multi-dimensional flow. If the crucial assumption is made that conserved quantities in grid cells are exchanged by waves traveling normally to the cell interfaces, Riemann solvers can easily be extended to multiple dimensions. Velocities parallel to the cell interface are ignored and any difference in the parallel component is assumed to take place across the contact surface.

The flux normal to the cell wall can be written as:

(3.25) $$F = (f\hat{i} + g\hat{j}) \bullet (k_x\,\hat{i} + k_y\,\hat{j})$$

where f and g are previously defined flux components and k_x and k_y are the x and y components of the unit vector $\hat{k}$. Unit vector $\hat{k}$ is normal to the cell interface and outwardly directed. Flux F can be linearized with respect to Q and a Jacobian matrix found.

$$B = \frac{\partial F}{\partial Q} \tag{3.26}$$

Following the same arguments used to derive the one-dimensional approximate Riemann solver and satisfying the same four conditions, leads to:

$$\frac{\Delta F}{\Delta Q} = \tilde{B}(Q_L, Q_R) = \tilde{T}\,\tilde{\Lambda}\,\tilde{T}^{-1} \tag{3.27}$$

where

$$\tilde{\Lambda} = \begin{bmatrix} \tilde{U} - \tilde{a} & 0 & 0 & 0 \\ 0 & \tilde{U} & 0 & 0 \\ 0 & 0 & \tilde{U} & 0 \\ 0 & 0 & 0 & \tilde{U} + \tilde{a} \end{bmatrix} \tag{3.28}$$

and $\tilde{U} = \tilde{u}k_x + \tilde{v}k_y$ is the contravariant velocity normal to the cell wall and $\tilde{T}$ and $\tilde{T}^{-1}$ are the left and right eigenvectors at Roe averaged conditions. The 'square root averaging' of the Roe-averaged state is identical to the one-dimensional case, with the addition of $\tilde{v}$.

$$\tilde{\rho} = (\rho_L \rho_R)^{1/2} \qquad \tilde{u} = \frac{\rho_L^{1/2} u_L + \rho_R^{1/2} u_R}{\rho_L^{1/2} + \rho_R^{1/2}} \qquad \tilde{v} = \frac{\rho_L^{1/2} v_L + \rho_R^{1/2} v_R}{\rho_L^{1/2} + \rho_R^{1/2}} \tag{3.29}$$

The expression for the total enthalpy, and the Roe-averaged speed of sound are nearly identical to the one-dimensional case.

$$\tilde{H} = \frac{\tilde{a}^2}{\gamma - 1} + \frac{\tilde{u}^2 + \tilde{v}^2}{2} \tag{3.30}$$

with

$$\tilde{H} = \frac{\rho_L^{1/2} H_L + \rho_R^{1/2} H_R}{\rho_L^{1/2} + \rho_R^{1/2}} \tag{3.31}$$

where

$$H_L = \frac{a_L^2}{\gamma - 1} + \frac{u_L^2 + v_L^2}{2} \qquad H_R = \frac{a_R^2}{\gamma - 1} + \frac{u_R^2 + v_R^2}{2} \tag{3.32}$$

As before $\tilde{a}$ is found from the Roe-averaged total enthalpy. The flux at the cell interface can be found using a formula nearly identical to equation (3.24):

$$F(Q_L, Q_R) = \frac{1}{2}\left(F_L + F_R - \tilde{T}\,|\tilde{\Lambda}|\,\tilde{T}^{-1}\,\Delta Q\right) \tag{3.33}$$

The term $\tilde{T}\,|\tilde{\Lambda}|\,\tilde{T}^{-1}\,\Delta Q$ in equation (3.33) can be reduced to three ΔF flux components [63], each of which is associated with a distinct eigenvalue.

$$|\Delta\tilde{F}| = |\Delta\tilde{F}_1| + |\Delta\tilde{F}_2| + |\Delta\tilde{F}_3| = \tilde{T}\,|\tilde{\Lambda}|\,\tilde{T}^{-1}\,\Delta Q \tag{3.34}$$

with

$$|\Delta\tilde{F}_1| = |\tilde{U}| \begin{bmatrix} \Delta\rho - \dfrac{\Delta p}{\tilde{a}^2} \\ \tilde{u}\left(\Delta\rho - \dfrac{\Delta p}{\tilde{a}^2}\right) + \tilde{\rho}[\Delta u - k_x \Delta\overline{U}] \\ \tilde{v}\left(\Delta\rho - \dfrac{\Delta p}{\tilde{a}^2}\right) + \tilde{\rho}[\Delta v - k_y \Delta\overline{U}] \\ \dfrac{\tilde{u}^2 + \tilde{v}^2}{2}\left(\Delta\rho - \dfrac{\Delta p}{\tilde{a}^2}\right) + \tilde{\rho}\left[\tilde{u}\Delta u + \tilde{v}\Delta v - \tilde{U}\Delta\overline{U}\right] \end{bmatrix}$$

(3.35)
$$|\Delta F_2| = |\tilde{U} + \tilde{a}| \left\{ \frac{\Delta p}{2\tilde{a}^2} + \frac{\tilde{\rho}\Delta\overline{U}}{2\tilde{a}} \right\} \begin{bmatrix} 1 \\ \tilde{u} + k_x\tilde{a} \\ \tilde{v} + k_y\tilde{a} \\ \tilde{H} + \tilde{U}\tilde{a} \end{bmatrix}$$

$$|\Delta F_3| = |\tilde{U} - \tilde{a}| \left\{ \frac{\Delta p}{2\tilde{a}^2} - \frac{\tilde{\rho}\Delta\overline{U}}{2\tilde{a}} \right\} \begin{bmatrix} 1 \\ \tilde{u} - k_x\tilde{a} \\ \tilde{v} - k_y\tilde{a} \\ \tilde{H} - \tilde{U}\tilde{a} \end{bmatrix}$$

where $\Delta\overline{U} = k_x\Delta u + k_y\Delta v$.

In the present scheme, as derived in section one, $F_{i,j}$ in equation (3.3) is simply $F(Q_L,Q_R)$ with the splitting direction $\hat{k} = \hat{\eta}_{i,j}$ in equation (3.4). The accuracy of the numerical scheme is determined by the accuracy to which the left (Q_L) and right (Q_R) fluid states are known.

3.3 Entropy fix for Roe's Approximate Riemann Solver

Roe's approximate Riemann solver may violate the entropy condition. This may happen when an expansion shock passes through a sonic point. At a sonic point, one of the eigenvalues of the linearized matrix, $\tilde{B}$, is zero, as in:

$$(3.36)\quad \tilde{U} = -\tilde{a} \;\Rightarrow\; \tilde{\lambda}_2 = \tilde{U} + \tilde{a} = 0 \quad \text{or} \quad \tilde{U} = \tilde{a} \;\Rightarrow\; \tilde{\lambda}_3 = \tilde{U} - \tilde{a} = 0$$

where $\tilde{\lambda}_1$, $\tilde{\lambda}_2$, and $\tilde{\lambda}_3$ are

$$(3.37)\qquad \tilde{\lambda}_1 = \tilde{U} \qquad \tilde{\lambda}_2 = \tilde{U} + \tilde{a} \qquad \tilde{\lambda}_3 = \tilde{U} - \tilde{a}$$

and are the same as used in equation (3.35). If an eigenvalue of Λ in equation (3.33) is zero, that implies that the eigenvector associated with $\tilde{\lambda}$ are multiplied by zero. Equation (3.33) then reduces to central differencing for the zero wave speed. Consequently, there is no dissipation for an expansion shock at a sonic point, and Roe's scheme will support and propagate the expansion shock. Several researchers have sought solutions or fixes to Roe's approximate Riemann solver to ensure that non-physical expansion shocks are not admitted.

All fixes to Roe's scheme involve adding dissipation to the method when the eigenvalues are small. This ensures that there is always some residual dissipation in Roe's scheme under any conditions.

Harten [76] redefines the eigenvalues when they are small, as in:

$$(3.38) \qquad |\tilde{\lambda}_{new}| = \begin{cases} \left(\dfrac{\tilde{\lambda}_{old}^2 + \varepsilon^2}{2\varepsilon} \right), & |\tilde{\lambda}_{old}| < \varepsilon \\ |\tilde{\lambda}_{old}|, & |\tilde{\lambda}_{old}| \geq \varepsilon \end{cases}$$

where $|\tilde{\lambda}_{new}| \geq \varepsilon/2$ at all times. The parameter ε is a small number, usually between 0.1 and 0.3, which is sufficient to exclude non-physical expansion shocks. This fix provides additional dissipation at all points in the flow where an eigenvalue is small.

Liou and Van Leer [77] define ε so that ε is non-zero only at expansion points.

$$(3.39) \qquad \varepsilon = K \max(\lambda_R - \lambda_L, 0)$$

The parameter K is usually taken to be one. The difference between the right and left eigenvalues is only positive when an expansion is traversing the interface between the two Riemann states. While Liou and Van Leer's fix to Roe's scheme may be more accurate, it does require an additional square root evaluation at each grid point to find the speed of sound. The extra square root evaluation at every grid point can be computationally expensive.

The fix to Roe's scheme is only truly necessary for nonlinear characteristic families. The fix is not necessary for $\tilde{\lambda}_1 = \tilde{U}$.

3.4 Extension of MUSCL-Differencing to Unstructured Meshes

The upwind scheme presented in section 3.2 is only first-order accurate in space. Using a preprocessing approach, usually referred to as MUSCL differencing, the data (Q) stored at the center of the dual mesh cells (V_i) or cell centers is interpolated to the cell interface. In this way, higher-order, left and right Riemann states are found for the upwind solver. On a structured mesh, as in Figure 1, first-order differencing corresponds to:

$$(Q_L)_{i+1/2,j} = Q_{i,j} \qquad (Q_R)_{i+1/2,j} = Q_{i+1,j} \tag{3.40}$$

On an unstructured mesh, as in Figure 3, first-order differencing corresponds to:

$$Q_L = Q(V_i) \qquad Q_R = Q(V_j) \tag{3.41}$$

Higher order differencing requires that state variables (Q) be used in the interpolation which are non-adjacent to the cell interface. A general upwind-biased MUSCL-type scheme for the structured mesh in Figure 1, was obtained in reference [78] by defining:

$$(Q_L)_{i+1/2,j} = Q_{i,j} + \frac{s}{4}[(1-\sigma s)\Delta_- + (1+\sigma s)\Delta_+]Q_{i,j} \tag{3.42}$$

(3.43) $$(Q_R)_{i+1/2,j} = Q_{i+1,j} - \frac{s}{4}[(1-\sigma s)\Delta_+ + (1+\sigma s)\Delta_-]Q_{i+1,j}$$

where

(3.44) $$\Delta_+(\bullet)_{i,j} \equiv (\bullet)_{i+1,j} - (\bullet)_{i,j}$$

(3.45) $$\Delta_-(\bullet)_{i,j} \equiv (\bullet)_{i,j} - (\bullet)_{i-1,j}$$

(3.46) $$s = \frac{2\Delta_+ \Delta_- + \delta}{(\Delta_+)^2 + (\Delta_-)^2 + \delta}$$

and δ is a small number ($\delta = 10^{-6}$) to prevent division by zero in regions of null gradient. The subscripts define the location in the structured mesh of Figure 1, where the Q's are stored. These interpolation formulas were derived on the basis of constant grid spacing. Constant grid spacing is generally not found in a typical computational mesh. Nevertheless, accurate results can be obtained using these formulas when grid spacing between adjacent sets of points does not vary much.

The parameter σ is used to control the order of the accuracy of the MUSCL-differencing. With σ defined to be -1, the scheme will be second-order accurate and fully upwinded. With $\sigma = 1/3$, MUSCL-differencing will be third-order accurate and upwind biased.

The s parameter in the formulas varies between zero and one and is called a "limiter". The limiter serves to limit the accuracy of the MUSCL-differencing

formulas in regions of large gradient. This limiting causes a loss of accuracy in the solution but does insure that oscillations in the numerical solution are prevented. Van Albada's [35,78] limiter (equation (3.46)) was used because it acted in a continuously differentiable manner which tended to prevent limit cycles in the convergence history.

In the interpolation formulas, any form of the state variables can be interpolated or extrapolated. The state variables can be in primitive variable format, $[\rho, u, v, p]^T$, or conserved variable format, $[\rho, \rho u, \rho v, E]$ or even in characteristic variable format. The notation Q has been defined to represent the conserved form of the state variables but in this section Q can refer to primitive, conserved, or characteristic variables. In any case, after the state variables are interpolated or extrapolated in MUSCL-differencing, the conserved form of the flux vector must be found to ensure conservation.

The structured mesh interpolation formulas can be extended to an unstructured mesh composed of triangles. This extension will allow MUSCL-differencing to be used on unstructured meshes.

MUSCL-differencing could be used on unstructured meshes, if for each edge of the mesh, state variables (Q) are located equidistantly along a line from the grid points defining the edge (see Figure 3). This condition exists naturally in a structured mesh but the values of the state variables are not stored at vertices V_i and V_j in an unstructured mesh. The vectors $\vec{V_iV_i}$ and $\vec{V_jV_j}$ are contained in a

triangular element surrounding V_i and V_j, respectively. Once the element containing the vector is found, simple interpolation of the state variables stored at the vertices of the element can be used to find Q at $V_{i'}$ or $V_{j'}$. The simplest way this can be implemented is to use Gauss's theorem in the plane to find the gradient of Q in the triangular element. The state variables (Q) are defined by the three vertices of the element so Q in the element can be represented by a linear, two-dimensional interpolating polynomial. Consequently, the gradient of Q in the element is a constant.

The MUSCL-differencing formulas can be written for the edge in Figure 3, as:

(3.47) $$(Q_L)_{i,j} = Q_i + \frac{s}{4}\{(1-\sigma s)\Delta_- + (1+\sigma s)\Delta_+\}Q_i$$

(3.48) $$(Q_R)_{i,j} = Q_j - \frac{s}{4}\{(1-\sigma s)\Delta_+ + (1+\sigma s)\Delta_-\}Q_j$$

where

(3.49) $$\Delta_-(Q_i) = Q_i - Q_{i'} \qquad \Delta_+(Q_i) = Q_j - Q_i$$

$$\Delta_-(Q_j) = Q_j - Q_i \qquad \Delta_+(Q_j) = Q_{j'} - Q_j$$

and

(3.50) $$\Delta_-(Q_i) = \vec{\nabla}Q_i \bullet \vec{V_{i'}V_i} = \vec{\nabla}Q_i \bullet \vec{V_iV_j}$$

(3.51) $$\Delta_+(Q_j) = \vec{\nabla}Q_j \bullet V_j\vec{V}_{j'} = \vec{\nabla}Q_j \bullet V_i\vec{V}_j$$

The notation $\vec{\nabla}Q_i$ and $\vec{\nabla}Q_j$ represent the gradient in the triangular element surrounding V_i and containing vector $V_i\vec{V}_i$ and surrounding V_j and containing vector $V_j\vec{V}_{j'}$, respectively.

Very general formulas [79] can be derived using Gauss's Theorem in the plane to find the gradients of functions. Values of the function are assumed to be known at the vertices of the polygon.

(3.52) $$\frac{\partial Q}{\partial x} = \frac{\sum_{n=1}^{m} Q_n(y_{n+1} - y_{n-1})}{\sum_{n=1}^{m} x_n(y_{n+1} - y_{n-1})}$$

(3.53) $$\frac{\partial Q}{\partial y} = -\frac{\sum_{n=1}^{m} Q_n(x_{n+1} - x_{n-1})}{\sum_{n=1}^{m} x_n(y_{n+1} - y_{n-1})}$$

where m is the number of vertices comprising the polygon. The formulas for the gradient assume the vertices of the polygon are numbered cyclically and in an anticlockwise manner. The formulas are exact if Q is a linear function, which is the case for a triangular polygon.

3.5 Algorithm and Data Structure

In a computer code to solve two-dimensional fluid flow on an unstructured grid composed of triangular elements, the data storage and algorithm to compute the flux around each dual mesh cell are not insignificant.

A residual, stored at the center of the dual mesh cell with the state variables (see Figure 2), can be defined and computed.

(3.54) $$R_i = \sum_{j=\kappa_i} F_{i,j} \Delta l_{i,j}$$

The residual represents the net amount of flux entering of leaving a dual mesh cell. If the cell residual is zero, the fluid flow at that point is in a steady state condition.

The geometrical data is stored in several matrices:

XY(1:NP,1:2) - the x and y locations of grid points/triangle vertices.

INTMAT(1:NE,1:3) - the three nodes associated with each triangular element (1 to NE).

INDX(1:MEDG,1:6) - the edges of the triangles (1 to NEDG for edges internal to mesh, NEDG + 1 to MEDG for boundary edges).
INDX(1:MEDG,1) is node V_i in Figure 3.
INDX(1:MEDG,2) is node V_j in Figure 3.
INDX(1:MEDG,3) is node V_k in Figure 3.
INDX(1:NEDG,4) is node V_l in Figure 3.

INDX(1:NEDG,5) is the element surrounding V_i and containing vector $\vec{V_iV_i}$.
INDX(1:NEDG,6) is the element surrounding V_j and containing vector $\vec{V_jV_j}$. Both elements in INDX are essentially pointer variables into array INTMAT to find the three nodes used for MUSCL-differencing.

BNODE(1:NB,1:3) - the boundary node and two boundary conditions for NB boundary points.

With this data structure, a complete flux balance over all the dual mesh cell sides can be accomplished with a simple loop over the edges. For example, an upwind scheme using MUSCL-differencing could use the simplified loop algorithm shown here to find the residuals for each dual mesh cell.

```
      DO 10    I = 1, NEDG
        N1  =  INDX(I,1)
        N2  =  INDX(I,2)
        N3  =  INDX(I,3)
        N4  =  INDX(I,4)
        N5  =  INDX(I,5)
        N6  =  INDX(I,6)
        N11 = INTMAT(N5,1)
        N12 = INTMAT(N5,2)
        N13 = INTMAT(N5,3)
        N21 = INTMAT(N6,1)
        N22 = INTMAT(N6,2)
        N23 = INTMAT(N6,3)
        QLEFT  = FUNCTION(VARIABLES AT N1,N2,N11,N12, AND N13)
        QRIGHT = FUNCTION(VARIABLES AT N1,N2,N21,N22, AND N23)
        FLUX = FUNCTION(QLEFT,QRIGHT)
        R(N1) = R(N1) + FLUX
        R(N2) = R(N2) - FLUX
10    CONTINUE
C
      DO 20    I = NEDG+1,MEDG
        N1  =  INDX(I,1)
        N2  =  INDX(I,2)
        N3  =  INDX(I,3)
        QLEFT  = Q(N1)
        QRIGHT = Q(N2)
```

```
      FLUX = CENTRAL_DIFFERENCE FUNCTION(QLEFT,QRIGHT)
      R(N1) = R(N1) + FLUX
      R(N2) = R(N2) - FLUX
20  CONTINUE
```

This algorithm only applies for edges of the mesh that are internal to the grid (edges 1 to NEDG) and edges at the boundary of the grid (edges NEDG+1 to MEDG). Dual cell edges that are on the boundary need special consideration to compute their flux contribution. The boundary edges are a very small part of the total edges present in the mesh. The power of this algorithm is that it is not necessary to recompute any cell side flux more than once in the entire grid. The only extra computational work needed is the zeroing of the residual storage vector before each evaluation of the fluxes.

3.6 Numerical Boundary Conditions

3.6.1 Solid Wall Boundary

Several approaches may be taken to applying the boundary conditions to the solid wall. The only true boundary condition that exists at a solid wall for an Euler flow, is that of flow tangency.

Using a discretized two-dimensional flow domain composed of triangles, with state variables stored at the vertices, the flow variables (Q) are going to be stored

at the wall. The dual mesh system (see Figure 4) will have half cells located on the solid wall boundary with the cell centers, V_i, on the boundary. The most obvious approach to enforce flow tangency is to solve the Euler equations at the boundary. The tangent line to the boundary at V_i is a poorly defined concept in a finite discretization of the boundary. A line connecting the midpoints of the two boundary edges that meet at V_i was chosen as the boundary tangent. The unit vector, $\hat{n}_B$, is taken as the outward facing normal to the boundary tangent at V_i. Connecting the midpoints of the two adjacent boundary edges and taking $\hat{n}_B$ as the unit normal, agrees with the analysis of section 3.1 where a length-averaged unit normal and edge length were found for each face of the dual mesh cell. Writing a two-dimensional flux vector normal to the wall, we obtain:

$$\vec{F} \cdot \hat{n}_B = (f\hat{i} + g\hat{j}) \cdot (n_{Bx}\hat{i} + n_{By}\hat{j}) = \begin{bmatrix} \rho\overline{U} \\ \rho u\overline{U} + pn_{Bx} \\ \rho v\overline{U} + pn_{By} \\ \overline{U}(E+p) \end{bmatrix} \tag{3.55}$$

where $\overline{U} = un_{Bx} + vn_{By}$ is the normalized contravariant velocity normal to the boundary tangent at V_i and n_{Bx} and n_{By} are the x and y components of $\hat{n}_B$. The velocity $\overline{U}$ is zero since $\overline{U}$ is normal to the boundary tangent at V_i. Consequently, the flux normal to the boundary face of the dual mesh cell C_i reduces to:

$$\vec{U} = 0 \Rightarrow \vec{F} \bullet \hat{n}_B = \begin{bmatrix} 0 \\ pn_{Bx} \\ p\hat{n}_{By} \\ 0 \end{bmatrix} \tag{3.56}$$

The pressure at the wall can simply be taken as the pressure stored at vertex V_i. The pressure at the wall is multiplied by the components of the unit vector $\hat{n}_B$ to find the momentum flux through the wall. The flux is multiplied by the length of the boundary tangent line and added to the residual at vertex V_i. A simple central difference can be used to find the flux through the cell sides of C_i which terminate at the start and end points of the boundary tangent. A centrally differenced flux will be second-order accurate while using two points but will be non-dissipative. No dissipation at the wall may lead to numerical instability.

Numerical experiments were performed while enforcing no flux through the wall. No problems were found with the numerical stability at the wall even while computing the boundary edge fluxes using a central difference. Unfortunately, two other problems were found. Large amounts of spurious entropy were generated at the wall by forcing the flux through the wall to be zero. The velocity vector at cell center V_i was not parallel to the boundary tangent line. The velocity vector would be close to tangent but never exactly tangent. Following the work of Mavriplis [80], after the flow was solved at the wall, the velocity was forced to be parallel to the boundary vector. To correct the entropy problem, a normal

extrapolation of the entropy was done. Referring to Figure 5, a search was made of the elements surrounding point V_i which contained a vector parallel to $\hat{n}_B$ and passed through point V_i. Once the element was found, linear interpolation was used between the two vertices of the element (which do not lie on the boundary and are nodes V_{j_2} and V_{j_3} in Figure 5) to find the entropy on a line passing through V_i and normal to the boundary tangent.

(3.57) $$s(V_i) = s(V_{j_3}, V_{j_4}) \qquad \text{in Figure 5}$$

The condition (V_{j_3}, V_{j_4}) indicates interpolation of variables at V_{j_3} and V_{j_4} to a point on a line passing through V_i and normal to the boundary tangent. Once the entropy directly above was found, it was normally extrapolated to surface grid point V_i. The surface pressure was held constant and the surface density was modified to match the normally extrapolated entropy. With these modifications, the boundary condition worked fairly well for most test cases. All spurious entropy was removed from the solution and the velocity vector was now tangent at all points. The residual at the wall grid point (V_i) can not longer be driven to zero, since zero flux through the wall is no longer being completely enforced. The non-zero residual is not a problem. The combination of the residual, the tangency of the velocity vector, and the entropy gradient normal to the wall, is being enforced.

Another boundary condition that was tried was the normal extrapolation of all boundary conditions. The equations applied were (see Figure 5 for geometric details):

$$H(V_i) = H(V_{j_3}, V_{j_4})$$

$$\rho(V_i) = \rho(V_{j_3}, V_{j_4}) \tag{3.58}$$

$$(u(V_i)\hat{i} + v(V_i)\hat{j}) \cdot \hat{n}_B = 0$$

$$T(V_i) = T(V_{j_3}, V_{j_4})$$

where the condition (V_{j_3}, V_{j_4}) indicates interpolation of variables at V_{j_3} and V_{j_4} to a point on a line passing through V_i and normal to the boundary tangent. The variable T is the fluid temperature and H is the total enthalpy. This application of the boundary conditions is identical to the application of boundary conditions in a structured grid code using normal extrapolation. No equations need be solved at the wall. The wall conditions are found entirely from the fluid state interior to the computational domain. This boundary condition is only formally first-order accurate. Solving the equations at the wall is also only formally first-order accurate, but numerical tests indicated that the method was more accurate than normal extrapolation.

Normal extrapolation can be extended to second-order accuracy by using gradient information along the line normal to the boundary tangent. The pressure,

total enthalpy, and entropy can be extrapolated from the interior of the flow domain as in equation (3.58). If the gradients of the total enthalpy and entropy at the wall are assumed to be zero, normal extrapolation of entropy and total enthaply will be second order accurate. The pressure gradient at the wall cannot be assumed to be zero. Using the normal momentum equation, a wall pressure gradient can be found. Once the wall pressure gradient is known, a second order accurate expression can be written for the wall pressure. Using the nomencalture of equation (3.58), the wall boundary conditions can be written as:

$$p(V_i) = p(V_{j_3}, V_{j_4}) - \left(\frac{\partial p}{\partial n}\right)_B \Delta n$$

(3.59)
$$H(V_i) = H(V_{j_3}, V_{j_4})$$

$$s(V_i) = s(V_{j_3}, V_{j_4})$$

$$(u(V_i)\hat{i} + v(V_i)\hat{j}) \cdot \hat{n}_B = 0$$

with the pressure gradient at the wall directed normal to the boundary tangent. The variable Δn is the distance between vertex V_i and the interpolation point. The pressure gradient is found from the normal momentum equation:

(3.60)
$$\left(\frac{\partial p}{\partial n}\right)_B = \frac{\rho(V_i)[u^2(V_i) + v^2(V_i)]}{R(V_i)}$$

where $R(V_i)$ is the local radius of curvature of the boundary at vertex V_i. A Lagrange curve fit was made of the boundary vertex, V_i, and the two adjacent

boundary vertices so that the center of the radius of curvature could be found. Using the radius of curvature and the known conditions at vertex V_i, the wall pressure gradient was found. Once the wall pressure, wall enthalpy, and wall entropy were known, the state variables at the wall could be found to second-order accuracy.

Even higher-order boundary schemes could be found by using flow gradient information from the interior of the domain with the extrapolated data. Third and higher-order boundary conditions, derivable from this type of analysis, were not used in the present study.

Characteristic-based schemes[25,81], such as used in structured, cell vertex schemes, were not used or experimented with due to a lack of time. The proper wall boundary condition for a triangle-based scheme that works under all conditions and yields second-order accuracy is not known. Further work on the wall boundary condition needs to be performed.

3.6.2 Far-field Boundary

Far-field boundary conditions are normally determined by performing a one-dimensional characteristic analysis normal to the boundary. The signs of the eigenvalues determine what conditions may be specified. Subsonic inflow typically has three eigenvalues (wave speeds) directed into the flow domain. Conse-

quently, three state variables may be specified and one state variable must be extrapolated. The situation is completely reversed for subsonic outflow. For a subsonic outflow, three wave speeds are directed out of the flow domain. Three state variables must be extrapolated and one must be specified. For supersonic outflow, all wave speeds are directed out of the domain and all state variables must be extrapolated. For supersonic inflow, all waves are entering the domain so everything must be specified.

Using a characteristic-based solver, as was done in this research, the solver should be able to extract what information is necessary from the boundaries. As a consequence, no special treatment of the boundaries should be necessary. For subsonic inflow and outflow, the boundary conditions were simply specified as free-stream conditions. If the test problem was an airfoil, the far-field boundary conditions for an airfoil were imposed at the far-field. The airfoil far-field vortex correction worked well in practice. To obtain very accurate results, it was necessary to place the far-field boundary at least 10 to 20 chords from the airfoil. The airfoil vortex correction alone could not deal with the doublet effect of the airfoil. If a compressible doublet had been modeled and added to the vortex correction, it is quite possible that the airfoil could be brought very close to the far-field boundaries with no loss of accuracy.

For simple supersonic inflow and outflow boundaries (where the component of the flow normal to the boundary was still supersonic), the flow variables were imposed or extrapolated, respectively. The free-stream conditions are imposed for

supersonic inflow. The same procedure that was used for normal extrapolation on the wall, was used to extrapolate the variables stored interior to the grid to grid points on the boundary. Applications of the supersonic inflow and outflow boundary conditions to numerical problems appeared to work well in practice.

3.6.3 Kutta Condition

The Kutta condition ensures that the flow around an airfoil has a continuous pressure at the trailing edge and meets smoothly at the trailing edge point. The trailing edge fixes the stagnation point on the body. In analytical solutions of inviscid, irrotational, incompressible flow, the Kutta condition must be specifically imposed. If the Kutta condition is not specifically imposed, the stagnation point will not be present at the trailing edge and the velocity and pressure at the trailing edge will be singular. In analytical solutions of compressible, rotational, inviscid flow, it is not clear if there is some naturally occuring mechanism in the flow to fix the stagnation point at the trailing edge of the airfoil.

In our numerical solution, nothing was done to force the Kutta condition to be imposed. In all airfoil results, the Kutta condition was naturally imposed by some mechanism in the flow or numerical viscosity present in the flow. This result agrees with previous Euler flow computations performed about airfoils (see Reference [24]).

IV. TIME INTEGRATION

The spatially differenced terms determine the steady state behavior of the governing equations. At steady state, the residual defined in section 3.5, is zero at all grid points.

$$R_i = \sum_{j=\kappa_i} F_{i,j} \Delta l_{i,j} = 0 \tag{4.1}$$

The governing equation for Euler flow can be written in terms of the residual and the volume-averaged Q, as:

$$S(C_i) \frac{\partial < Q_i >}{\partial t} = -R_i \tag{4.2}$$

where $S(C_i)$ and $< Q_i >$ are the area and area-averaged Q of dual mesh cell C_i.

The residual is completely determined by the spatially differenced terms. The time-integration process is completely independent of the spatial differencing and the steady state result should be independent of the time integration process.

Two basic types of time integration methods are available. One is explicit time integration and the other is implicit time integration. Each differs in the discrete time level at which the residual vector is evaluated. Implicit time integration requires that the residual be evaluated at the new discrete time level. The residual and the Q vector are solved for simultaneously at the new time level. In the case of non-linear equations, such as the Euler equations, the residual at the new time level can only be approximated using a linearization of the residual function in terms of the old (known) and new state variables (Q). The scheme is then essentially a Newton method for a system of non-linear equations.

The linearization of the flux vector in terms of Q is very difficult for an upwind scheme, and particuliarly difficult for Roe's approximate Riemann solver. In general, the left hand side of the matrix will be sparse and very large. The sparseness and size of the matrix makes it very difficult and computationally inefficient to try and recover a Newton method by using direct inversion. An unstructured mesh has no underlying structure which could be exploited by a numerical algorithm. Inversion by lines, a very popular technique in structured grid codes, is not possible on unstructured meshes. It is possible to use domain splitting and solve the equations for a Newtonian solver over groups of dual mesh

cells. A domain splitting technique is not simple to program and debug for an unstructured mesh.

The primary thrust of the present research was the application of upwind methods to unstructured meshes so the geometric flexibility of finite element codes could be obtained while achieving the high accuracy of characteristic methods.

4.1 Runge-Kutta Time Stepping

Explicit time integration schemes are simple to apply and easy to program. Explicit schemes involve only the data stored at a single point when time integration is performed. This property is useful for unstructured grids since there is not underlying structure of the grid to be exploited by the time integration scheme. Explicit schemes can be adapted efficiently to a wide variety of computer architectures. Virtually all explicit schemes will vectorize on vector processors. Since the time update of individual grid points does not depend on neighboring points, explicit schemes can be easily adapted to parallel processing computers. Explicit methods require no linearization of the spatial terms or boundary conditions. The cell centered residual can be evaluated based on known conditions at the grid points. Both time accurate and non-time accurate solutions can be obtained using explicit time integration schemes in the same computer program. Second-order time accuracy can be easily obtained using explicit methods. The primary

disadvantage of explicit schemes is their slow convergence rate compared to implicit schemes. This slow convergence rate may be more pronounced for upwind difference schemes as compared to central difference schemes, when using explicit time integrators.

A very popular scheme for finding solutions to systems of differential equations is Runge-Kutta time integration. Runge-Kutta methods are one-step methods and require no starting solution. The classical Runge-Kutta methods are memory inefficient since a four-stage Runge-Kutta scheme needs all four time levels stored to perform a final update of the state variables. A. Jameson [24] developed a variation of Runge-Kutta schemes referred to as Runge-Kutta time stepping. Runge-Kutta time stepping requires the storage of two levels of the state variables and one level of the residuals for any number of stages. An m-stage Runge-Kutta time stepping scheme is:

$$
\begin{aligned}
Q_i^{(0)} &= Q_i^n \\
Q_i^{(1)} &= Q_i^{(0)} - \alpha_1 \, \Delta t \, R_i^{(0)} \\
&\cdots \\
Q_i^{(m-1)} &= Q_i^{(0)} - \alpha_{m-1} \, \Delta t \, R_i^{(m-2)} \\
Q_i^{(m)} &= Q_i^{(0)} - \alpha_m \, \Delta t \, R_i^{(m-1)} \\
Q_i^{n+1} &= Q_i^{(m)}
\end{aligned}
\tag{4.1}
$$

where Q^n is Q after n time steps, and Q^{n+1} is Q after $n + 1$ time steps. The (0) to (m) superscripts represent stages or time steps in the Runge-Kutta time stepping. Residuals are evaluated as a function of Q at the same stage, so that for stage

m, $R_i^{(m)} = R(Q_i^{(m)})$. The α_1 to α_m are weighting factors for the time stepping and can be defined as:

$$\alpha_k = \frac{1}{m-k+1} \qquad \text{for } k = 1, \ldots, m \tag{4.2}$$

These values of α_k will give m-order accuracy in time for a linear equation. Typically, we are interested in a steady state solution and not time accuracy, so the values of α_k are modified to obtain a fast convergence rate. A four-stage Runge-Kutta time stepping scheme was used, as:

$$\begin{aligned}
Q_i^{(0)} &= Q_i^n \\
Q_i^{(1)} &= Q_i^{(0)} - \alpha_1 \Delta t \, R_i^{(0)} \\
Q_i^{(2)} &= Q_i^{(0)} - \alpha_2 \Delta t \, R_i^{(1)} \\
Q_i^{(3)} &= Q_i^{(0)} - \alpha_3 \Delta t \, R_i^{(2)} \\
Q_i^{(4)} &= Q_i^{(0)} - \alpha_4 \Delta t \, R_i^{(3)} \\
Q_i^{n+1} &= Q_i^{(4)}
\end{aligned} \tag{4.3}$$

Four-stage Runge-Kutta time stepping was chosen since experimentally developed weighting factors, α_k, were available. The experimentally determined weighting coefficients had been found for a flux-vector split computer code and were optimized for fast convergence.

4.2 Convergence Acceleration

Due to the slow convergence rate of explicit schemes, several techniques were used in an attempt to increase the convergence rate. Some techniques were successful under almost any condition, while others were only useful for certain grid configurations or flow conditions.

4.2.1 Non-Standard Weighting of Runge-Kutta Stages

In reference [82], Turkel and Van Leer apply Runge-Kutta time stepping to a flux-vector split code. Turkel and Van Leer experimentally determined weighting factors for the Runge-Kutta scheme. They used $\alpha_1 = 0.170$, $\alpha_2 = 0.273$, $\alpha_3 = 0.500$ and $\alpha_4 = 1.000$. These weighting factors give a faster convergence rate than the standard weighting coefficients. The convergence rate for an upwind code is still poor when compared to a central difference code using Runge-Kutta time stepping.

4.2.2 Local Time Stepping

Local time stepping was performed to accelerate convergence by advancing the solution in each dual mesh cell to the limit of cell stability. Local time stepping implies that all dual mesh cells are advanced in time at a constant Courant

number. As the spatial size of the dual mesh cells vary so will the local time step. Since each mesh cell is being integrated in time at a different time step, time accuracy of the solution is lost. This is unimportant since the steady state solution is the only solution of interest.

A two-dimensional scalar advection equation was analyzed to find the proper expression for the Courant number.

$$\frac{\partial q}{\partial t} + a\frac{\partial q}{\partial x} + b\frac{\partial q}{\partial y} = 0 \tag{4.4}$$

Simple Euler explicit time integration was used in the model equation since it is difficult to analyze Runge-Kutta time stepping using a Von Neumann stablility analysis. Use of Euler explicit time integration will not give a correct form for the stability limit but it was considered superior to using a simple one-dimensional "Courant number" for each spatial dimension. The spatial terms were approximated using a Courant-Issacson-Rees (CIR) scheme which is the scalar equivalent of Roe's approximate Riemann solver. No MUSCL-differencing was performed to obtain higher order accuracy. The solution scheme for the two-dimensional advection equation was first-order in space and time. The wave speeds, a and b, were taken as positive to simplify the analysis. The final form of the scalar equation that was analyzed using the Von Neumann method (see Figure 1), was:

$$\frac{q_{i,j}^{n+1} - q_{i,j}^{n}}{t^{n+1} - t^{n}} + a\frac{q_{i,j}^{n} - q_{i-1,j}^{n}}{\Delta x} + b\frac{q_{i,j}^{n} - q_{i,j-1}^{n}}{\Delta y} = 0. \tag{4.5}$$

where Δx and Δy are the mesh spacing and subscripts i and j are the x and y locations, respectively, of scalar q in the mesh.

The Von Neumann stability analysis yielded a stability limit for the scalar advection equation, as:

$$\Delta t \leq \frac{\sqrt{5}}{5} \frac{\Delta x \Delta y}{a\Delta y + b\Delta x} \tag{4.6}$$

where $\Delta t = t^{n+1} - t^n$. This form of the expression can be rewritten for the Euler equations as:

$$\Delta t_i \leq v \frac{\Delta x_i \Delta y_i}{(|u_i| + a_i)\Delta y_i + (|v_i| + a_i)\Delta x_i} \tag{4.7}$$

where v is taken to be the Courant number. Grid spacings, Δx_i and Δy_i, at vertex V_i were found by averaging the x and y components of the edges that meet at vertex V_i. The time step, Δt_i, was re-evaluated after 10 or 20 time steps using equation (4.7) to maintain stability as u and v change during the time integration.

4.2.3 Residual Smoothing

Explicit time stepping of hyperbolic conservation laws requires that the domain of dependence of the numerical scheme contain at least that of the original partial differential equation. This requirement limits the maximum time step that can be used at each grid point. Runge-Kutta time stepping can be used to increase

the maximum time step by increasing the number of stages used. Van der Houwen, in reference [83], showed that only marginal improvement in the maximum time step can be found by increasing the number of Runge-Kutta stages. In the case of a upwind scheme, the residual is very expensive to evaluate due to the numerical complexity of the spatial differencing method.

Another approach to increasing the maximum time step [66] is to average the residual at a point with the surrounding residuals. This averaging increases the support of the method and permits significantly larger time steps. Residual smoothing is essentially a filtering technique, with the residuals being filtered through a Laplacian filter. First-order, explicit averaging can be written as:

$$\overline{R}_i = R_i + \varepsilon \nabla^2 R_i \tag{4.8}$$

where $\overline{R}_i$ is the smoothed residual and $\nabla^2 R_i$ is an undivided Laplacian which can be defined on an unstructured mesh as:

$$\nabla^2 R_i = \sum_{j=\kappa(i)} (R_j - R_i) \tag{4.9}$$

The variable $\kappa(i)$ is a list of the vertices surrounding vertex V_i. The undivided Laplacian is essentially approximated by taking the sum of the differences between the dual mesh cell centered residual and the surrounding residuals.

Explicit smoothing of the residuals gives little support for the time integration scheme, hence the maximum time step is only marginally improved. The maximum time step can be improved greatly by performing the residual smoothing implicitly. An implicit form of residual smoothing can be written as:

$$\overline{R}_i = R_i + \varepsilon \nabla^2 \overline{R}_i \tag{4.10}$$

The resulting set of equations can easily be solved by using Jacobi iteration. Jacobi iteration is used so that the resulting scheme will vectorize on a vector processor.

If a large enough ε is used the support becomes infinite and the maximum time step will approach infinity. In order for the support to be infinite the equations must be solved exactly which cannot be efficiently done using Jacobi iteration for a large value of ε. Using a value of $\varepsilon = 0.5$, suggested in reference [66], the resulting system of equations is highly diagonally dominant. Two passes of Jacobi iteration on the diagonally dominant system of equations were found to be sufficient to find a good approximation of $\overline{R}_i$ at all vertices. Reference [82] suggested that the residual smoothing be performed after every stage of the Runge-Kutta time stepping. Smoothing the residuals is computationally inexpensive compared to evaluating the residuals using an upwind scheme. Consequently, residual smoothing is a very effective method for improving the convergence rate of an explicitly time integrated upwind scheme. The residual smoothing scheme worked well under all tested conditions and was little affected by the grid configuration or flow conditions.

4.2.4 Residual Minimization

M. Hafez *et al* [83] developed a minimal residual technique for the Euler equations. The method attempts to minimize the L_2 norm of the residual. Let:

$$Q^* = Q^{n-1} + \omega(Q^n - Q^{n-1}) \tag{4.11}$$

where Q^* is a vector of the state variables that minimize $R(Q^*) \bullet R(Q^*)$. The free parameter, ω, is found to minimize $R(Q^*) \bullet R(Q^*)$. If the residual is assumed to be a linear function of Q, then the following equation can be written:

$$R(Q^*) = R^{n-1} + \omega(R^n - R^{n-1}) \tag{4.12}$$

The inner product (that which is to be minimized) can be written as:

$$|R(Q^*)|^2 = |R^{n-1} + \omega(R^n - R^{n-1})|^2 \tag{4.13}$$

where the vertical bars raised to the power of two represent the inner product and not the square of the inner product. Applying the condition:

$$\frac{\partial\left[|R(Q^*)|^2\right]}{\partial\omega} = 0 \tag{4.14}$$

and solving for ω results in:

$$\omega = -\frac{R^{n-1} \bullet (R^n - R^{n-1})}{(R^n - R^{n-1}) \bullet (R^n - R^{n-1})} \tag{4.15}$$

Residual minimization requires that an extra time level of the residual (R) and state variable (Q) be stored. Once ω is found, the current time level and the previous time level of Q are extrapolated using equation (4.11) to find Q^*.

Two relaxation parameters can be used to obtain a better approximation to Q^*. We can rewrite the equation for Q^* as:

$$Q^* = Q^{n-1} + \omega_1(Q^n - Q^{n-1}) + \omega_2(Q^{n-1} + Q^{n-2}) \tag{4.16}$$

Assuming a linear relationship between Q and R again gives:

$$R^* = R^{n-1} + \omega_1(R^n - R^{n-1}) + \omega_2(R^{n-1} + R^{n-2}) \tag{4.17}$$

with

$$|R^*|^2 = |R^{n-1} + \omega_1(R^n - R^{n-1}) + \omega_2(R^{n-1} + R^{n-2})|^2 \tag{4.18}$$

Minimizing the above inner product with respect to ω_1 and ω_2 as in:

$$\frac{\partial[|R^*|^2]}{\partial\omega_1} = 0 \qquad \frac{\partial[|R^*|^2]}{\partial\omega_2} = 0 \tag{4.19}$$

results in a system of equations for ω_1 and ω_2 :

$$\begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = - \begin{bmatrix} d \\ e \end{bmatrix} \tag{4.20}$$

where

$$
\begin{aligned}
\beta_{11} &= (R^n - R^{n-1}) \bullet (R^n - R^{n-1}) \\
\beta_{12} &= (R^{n-1} - R^{n-2}) \bullet (R^n - R^{n-1}) \\
\beta_{21} &= (R^{n-1} - R^{n-2}) \bullet (R^n - R^{n-1}) \\
\beta_{22} &= (R^{n-1} - R^{n-2}) \bullet (R^{n-1} - R^{n-2}) \\
d &= R^{n-1} \bullet (R^n - R^{n-1}) \\
e &= R^{n-1} \bullet (R^{n-1} - R^{n-2})
\end{aligned}
\tag{4.21}
$$

Two parameter residual minimization requires that two time levels of the residual and state variable be stored. The increased storage requirement is generally not a problem on virtual-memory minicomputers or supercomputers. Two parameter residual minimization leads to greatly improved approximations of Q^* and a subsequent improvement in the convergence rate. The residual minimization technique can easily be derived for three, four or more parameters using the present derivation. Using three or more parameters requires an extra level of storage for each parameter. Since the residual (R) is assumed to a be a linear function of Q , a point of diminishing returns is soon reached by adding successively more parameters to the acceleration scheme. Two parameter residual minimization appears to be a good compromise between complexity, storage requirements, and convergence acceleration.

The two parameter method was applied every 10 to 20 time steps and required very little computational time. The number of time steps between application of the minimization did not appear to be critical. The method complemented Runge-Kutta time stepping. Runge-Kutta time stepping tends to eliminate short

wave length errors in the solution. Short wave length errors could be considered to be the local solution error between two adjacent grid points. The residual minimization technique minimized long wave length errors. The long wave length errors can be considered to be the solution error between grid points that are non-adjacent and separated by large distances. The residual minimization technique is equally applicable for both structured and unstructured meshes since it assumes no connectivity between any of the grid points.

The technique worked well sometimes and was ineffective at other times. The acceleration scheme worked best for fully supersonic flows on grids that had little variation in dual mesh cell size. Grids that had a large variation in cell size tended to have little convergence acceleration. Transonic airfoils typically require very fine meshes at the leading and trailing edges of the airfoil. The farfield might be 10-100 chords away from the airfoil with very large mesh spacing on the order of 100 to a 1000. Transonic airfoils on highly stretched meshes showed little improvement in convergence rate. Supersonic internal flows or external flows on a restricted domain showed large increases in convergence rate.

V. RESULTS AND DISCUSSION

In order to verify the computational method and theory, several test cases were examined over a wide variety of flows and geometries.

5.1 Shock Reflection Problem

The shock reflection is a simple test problem that involves the reflection of an oblique shock wave off the surface of a flat plate. The problem has been studied extensively by many researchers. The free-stream Mach number is 2.9 and the incident shock angle is 29°. The grid is rectangular shaped (see Figure 6) and extended $0 \leq X \leq 4$ in the X coordinate and $0 \leq Y \leq 1.5$ in the Y coordinate. The mesh was composed of 3291 triangular elements, and 1697 vertices. Of the 1697 vertices, 101 lay on the boundary of the mesh. The mesh was generated with a

finer distribution of grid points in the region of the shock in order to improve the results and decrease the computational work.

In this test case, a value of $\sigma = 1/3$ was used in the MUSCL-differencing formula. Use of $\sigma = 1/3$ should give third order accuracy for the cell side fluxes in regions of the flow outside of shocks. Van Albada's limiter was used in the MUSCL-differencing formula in order to reduce spurious oscillations in the shock region.

For $1 \leq Y \leq 1.5$ on the inflow plane and the upper boundary, post shock conditions are prescribed at the boundary points. For $0 \leq Y < 1$, free-stream conditions are prescribed for the inflow boundary. At the outflow plane, all conditions are extrapolated from the interior of the flow domain.

Contour plots of constant Mach number, pressure and density lines appear in Figures 7, 8 and 9, respectively. The contour lines appear to be smooth and straight with little noise. The reflected shock appears to be as sharp and well defined as the incident shock. This fact is not surprising in an unstructured mesh. An unstructured grid lines up equally well with any one-dimensional flow feature regardless of feature orientation. At the outflow plane, the contour lines are kinked due to the method used to extrapolate the flow variables.

In Figure 10, two methods enforcing boundary tangency at the wall are compared with the exact solution. One method solves the equations at the wall, extrapolates

entropy, and forces the velocity at the wall to be tangent. The other method uses normal extrapolation of all boundary conditions. Both solutions compare well with the exact solution and the shock is accurately located in both cases. The shock appears less sharp with the extrapolated boundary conditions but is possibly more accurately located. A small oscillation appears in the shock when the equations are solved at the wall.

5.2 Blunt Body Problem

The blunt body that was studied in the present research is the leading edge of a panel holder in the Nasa 8-foot high temperature tunnel. This problem has be solved by many researchers and functions as a generic blunt body problem.

A plot of the mesh generated about the blunt body appears in Figure 11. The mesh has 10605 triangular elements, 5470 vertices. The boundary of the mesh was composed of 333 vertices. All coordinates were non-dimensionalized by the radius of the cylinder. The origin of the coordinate system was placed at the center of the cylinder (circular arc). The computational mesh extended from $-2 \leq X \leq 1$ in the X coordinate and $0 \leq Y \leq 4$ in the Y coordinate. The free-stream Mach number was 6.57 and $\gamma = 1.38$ to partially account for the real gasdynamic effects of the fluid used in 8-foot high temperature tunnel.

Free-stream conditions were prescribed for the circular arc portion of the mesh in Figure 11. The symmetry line and the surface of the cylinder-wedge were treated as solid surface boundary conditions. A first-order extrapolation was used to determine the boundary conditions at the outflow plane.

All solutions were generated using $\sigma = -1$, which should give second-order accuracy. A plot of Mach contours, pressure contours, and density contours can be seen in Figures 12, 13, and 14, respectively. The shock appears quite sharp and the expansion lines quite smooth in the contour plots.

The solid surface boundary condition was imposed by normal extrapolation and normal extrapolation with a pressure correction at the wall surface. Solving the equations at the wall was attempted but was unsuccessful. Flow-field pressures along the line of symmetry are compared with R. Walters's [64] upwind, structured grid solution in Figure 15 . The shock, using both boundary conditions, is sharp and accurately located. Neither shock is as accurate as Walters's solution. The purely extrapolated boundary condition appears to give a solution that agrees more closely with Walters's solution. The mesh in the region of the shock was as fine as the mesh used by R. Walters. R. Walter's mesh was composed of grid lines that ran approximately parallel to the bow shock. The unstructured mesh, that was used, has no preferred geometric orientation. One-dimensional Riemann solvers used in multiple dimensions are known to capture shocks more accurately when they are applied normal to the shock.

5.3 10^0 Wedge Inlet.

The 10° wedge inlet is a fairly common test problem that possesses an exact solution for the lower surface. The mesh was generated to determine what was possible using adaptive gridding. Mesh points were placed in the domain where shocks and expansion fans were known to exist.

A plot of the mesh can be seen in Figure 16. The lower left point of the mesh was selected as the origin. The mesh extends from $0m \leq X \leq 0.1m$ in The X coordinate and $0m \leq Y \leq 0.02m$ in the Y coordinate. The 10° wedge is located at $X = 0.02m$ and terminates at $X = 0.04m$. The mesh was composed of 12441 triangles and 6344 vertices. The mesh had 245 vertices on the boundary. The free-stream Mach number was 5.0.

At the inflow, the boundary conditions were prescribed. The lower and upper boundaries were treated as flow tangency conditions and the equations were solved at the wall (with extrapolation of entropy and forcing the velocity to be tangent) and normal extrapolation of all conditions. At the outflow, first-order extrapolation was used. A value of $\sigma = -1$ was used to give second-order accuracy in the MUSCL-differencing formula.

Contour plots of Mach number, pressure and density appear in Figures 17, 18, and 19, respectively. The shock off the wedge, the reflected shock, and the expansion fan seem to be captured almost perfectly. Lower wall surface pressures

are compared in Figures 20 and 21. Figure 20 compares the exact solution with the computed solution that solved the equations at the wall. Figure 21 compares the exact solution with the computed solution that used normal extrapolation of all conditions. The extrapolated solution is less oscillatory and appears to agree with the exact solution better. Neither boundary condition captures the pressure rise at the base of the wedge. Inspection of the grid leads the author to believe that the mesh was not fine enough at the wedge base to capture the solution accurately.

5.4 Subcritical NACA 0012 Airfoil

The NACA 0012 was tested at a free-stream Mach of 0.63 and an angle of attack of 2°. The flow around the airfoil is subcritical at all points. This test case was the subject of a GAMM workshop for Euler solvers in the past and has been extensively tested.

The outer boundary of the mesh was a circle centered at the half-chord point of the airfoil. The radius of the outer boundary was 20 chords. A plot of the outer boundary can be seen in Figure 22. For this case, a value of $\sigma = 1/3$ was used for third-order accuracy in the MUSCL-differencing formula. No limiting of the solution was done since no shocks were present in the flow-field. The far-field vortex boundary condition was used. At the surface of the airfoil, the equations

were solved, entropy was extrapolated normally, and the wall velocity was forced to be tangent. At the trailing edge point of the airfoil, the equations were solved but no attempt was made to force velocity tangency or extrapolate entropy. A closeup of the mesh appears in Figure 23. The mesh was composed of 8381 triangles and 4291 vertices. The mesh had 201 vertices on the boundaries.

Mach contours, pressure contours, and density contours appear in Figures 24, 25, and 26, respectively. The contour plots appear to be smooth and noise free. The coefficient of lift and drag were: $C_l = 0.33209$ and $C_d = -0.00039$. The lift agreed well with other results [85]. The drag should have been zero. Surface pressure comparisons are made with the reference solution of Lock [86] in Figure 27. The solution appears to agree everywhere except at the leading edge. There seem to be sufficient grid points at the leading edge. Possibly the spacing of the grid points normal to the leading edge was too great.

5.5 Supercritical NACA 0012 Airfoil (Case 1)

The NACA 0012 airfoil was tested at a Mach number of 0.8 and an angle of attack of 1.25°. This test case is currently a test problem for AGARD Fluid Dynamics Panel Working Group 07 and has been extensively tested.

The outer boundary of the mesh was located 20 chords from the half-chord point of the airfoil. The vortex far-field boundary condition was applied. For this case,

$\sigma = 1/3$ was used for third-order accuracy in the MUSCL-differencing formula. Two methods were used to enforce flow tangency at the wall. The first method solved the equations at the wall, extrapolated entropy, and forced the wall velocity to be tangent to the boundary tangent. The second method extrapolated all conditions and corrected the wall pressure using the normal momentum equation at the wall. The Euler equations were solved at the trailing edge point with no correction. Van Albada's limiter was used with the extrapolated boundary conditions but not when the equations were solved at the wall. This result is quite surprising. Apparently the shock that is generated on the airfoil is weak enough to permit the solution be obtained without a limiter. A closeup of the grid can be seen in Figure 28. The grid was composed of 10467 triangles and 5374 vertices. The grid had 280 vertices laying on the boundary of the mesh.

Mach contours, pressure contours, and density contours are present in Figures 29, 30, and 31, respectively. The shocks appear to be well resolved, including the weak shock present on the lower surface. Surface pressures are compared with the solution of Mavriplis [80] in Figures 32 and 34. Mavriplis solves the Euler equations on a mesh of unstructured triangles using a central difference scheme. In both cases, the surface pressure agreement was good except in regions near the trailing edge and in the shocks. Both boundary conditions yielded slightly higher pressures (both lower and upper surface) near the trailing edge as compared to Mavriplis's solution. Both conditions yielded sharper, more well defined shocks than Mavriplis's method, though the present work had more grid points on the

airfoil. In Figures 33 and 35, surface entropy is compared with Mavriplis. The surface entropy was defined as:

$$\frac{\left[\frac{p}{p_\infty}\right]}{\left[\frac{\rho}{\rho_\infty}\right]^\gamma} - 1 \tag{5.1}$$

Both boundary conditions arrive at the surface entropy through first-order extrapolation. The unlimited solution agrees with Mavriplis and appears to have lower entropy error overall near the leading and trailing edges. This result suggests that Van Albada's limiter is generating small errors in the entropy in regions of rapid compression (airfoil leading edge) which are convected along the airfoil surface.

The airfoil lift and drag coefficient were: $C_l = 0.37243$ and $C_d = 0.02358$ (extrapolation with wall pressure correction) and $C_l = 0.36299$ and $C_d = 0.02276$ (equations solved at wall). Both agree well with published results.

5.6 Supercritical NACA 0012 Airfoil (Case 2)

The NACA 0012 airfoil was tested at a Mach number of 0.85 and an angle of attack of 1°. This test case is currently a test problem for AGARD Fluid Dynamics Panel Working Group 07 and has been extensively tested. Large shocks

form on the lower and upper surface of the airfoil. The lift and drag are highly dependent on the capture and proper location of the shocks.

The outer boundary of the mesh was located 20 chords from the half-chord point of the airfoil. The vortex far-field boundary condition was applied. For this case, $\sigma = 1/3$ was used for third-order accuracy in the MUSCL-differencing formula. Three methods were used to enforce flow tangency at the wall. The first method solved the equations at the wall, extrapolated entropy, and forced the wall velocity to be tangent to the boundary tangent. The second method extrapolated all conditions to the airfoil surface. The third method extrapolated all conditions and corrected the wall pressure using the normal momentum equation at the wall. The Euler equations were solved at the trailing edge point with no correction. Van Albada's limiter was used with all boundary conditions. A closeup of the grid can be seen in Figure 36. The grid was composed of 10122 triangles and 5201 vertices. The grid had 280 vertices laying on the boundary of the mesh.

Mach contours, pressure contours, and density contours are present in Figures 37, 38, and 39, respectively. The shocks appear to be well resolved as well as the slip line off the trailing edge. Surface pressures are compared with the solution of Thomas *et al* [87] in Figures 40, 41, and 42. The solution of Thomas *et al* is an upwind scheme solved on structured mesh. None of the solutions matched Thomas *et al* solution on the upper surface of the airfoil before the shock location. Both upper and lower surface pressures were high relative to the comparison solution. Both the first and third tangency boundary condition located the shocks

accurately. The second boundary condition caused the shock to be mislocated slightly relative to the first and third boundary condition. All three boundary conditions captured the shocks less sharply than Thomas *et al* solution although the first boundary condition did the best. Thomas *et al* used a structured grid, which for an airfoil, tends to line up with the shocks and permit one-dimensional upwind solvers to operate normal to the shock.

The airfoil lift and drag coefficient were: $C_l = 0.3534$ and $C_d = 0.05727$ (first boundary condition) and $C_l = 0.3611$ and $C_d = 0.06556$ (second boundary condition) and $C_l = 0.38848$ and $C_d = 0.05614$ (third boundary condition). The lift and drag coefficients generated by the first and third boundary condition agree well with other published work. The drag coefficient of the second boundary condition is too high.

5.7 RAE 2822 Airfoil

The RAE 2822 airfoil was tested at a Mach number of 0.75 and an angle of attack of 3°. This test case is currently a test problem for AGARD Fluid Dynamics Panel Working Group 07 and has been extensively tested. The airfoil was designed in the United Kingdom by the Royal Aircraft Establishment to be a supercritical airfoil. The airfoil was tested in a wind tunnel by the RAE at the

stated conditions which are far away from the supercritical design conditions. A very powerful shock forms on the upper surface which can be difficult to resolve.

The outer boundary of the mesh was located 20 chords from the half-chord point of the airfoil. The vortex far-field boundary condition was applied. For this case, $\sigma = 1/3$ was used for third-order accuracy in the MUSCL-differencing formula. Only one method was used to enforce flow tangency at the wall. Flow tangency was enforced by solving the equations at the wall, extrapolating entropy, and forcing the wall velocity to be tangent to the boundary. The Euler equations were solved at the trailing edge point with no correction. Van Albada's limiter was used with all boundary conditions. A closeup of the grid can be seen in Figure 43. The grid was composed of 8898 triangles and 4546 vertices. The grid had 230 vertices laying on the boundary of the mesh.

Mach contours, pressure contours, and density contours are present in Figures 44, 45, and 46, respectively. The shock is well resolved as is the slip line coming off the rear of the airfoil. Surface pressures are compared with the solution of Lerat [88] in Figure 47. The shock is much sharper than Lerat's solution but properly located. The leading edge suction peak is not achieved, probably due to a lack of mesh resolution normal to the leading edge. The upper surface pressure does not achieve the C_P of Lerat's solution.

The airfoil lift and drag coefficient were: $C_l = 1.09104$ and $C_d = 0.04491$. These lift and drag coefficients agree well with published results.

5.8 NLR 7301 Airfoil

The NLR 7301 airfoil was tested at a Mach number of 0.720957 and an angle of attack of $-0.194°$. This test case is currently a test problem for AGARD Fluid Dynamics Panel Working Group 07 and has been extensively tested. The airfoil was designed in the Netherlands using a hodograph technique to be a supercritical airfoil. The stated conditions are the design conditions for shock free operation.

The outer boundary of the mesh was located 20 chords from the half-chord point of the airfoil. The vortex far-field boundary condition was applied. For this case, $\sigma = 1/3$ was used for third-order accuracy in the MUSCL-differencing formula. Only one method was used to enforce flow tangency at the wall. Flow tangency was enforced by solving the equations at the wall, extrapolating entropy, and forcing the wall velocity to be tangent to the boundary. The Euler equations were solved at the trailing edge point with no correction. The solution was run unlimited so no limiter was used. A closeup of the grid can be seen in Figure 48. The grid was composed of 9906 triangles and 5135 vertices. The grid had 364 vertices laying on the boundary of the mesh.

Mach contours, pressure contours, and density contours are present in Figures 49, 50, and 51, respectively. The contours appear to be very smooth and noise free. A small shock appears to be forming on the upper surface. Surface pres-

sures are compared with the solution of Walters [89] in Figure 52. The small incipient shock agrees well with the small shock that Walters found. The solutions agree very well at all other locations on the airfoil.

The airfoil lift and drag coefficient were: $C_l = 0.60494$ and $C_d = 0.00007$. These lift and drag coefficients are in line with published results. The very low drag coefficient is a measure of the accuracy of this solution.

5.9 Karman-Trefftz Airfoil and Flap

The airfoil and flap were created using the Karman-Trefftz transformation. An analytical incompressible solution was found for this airfoil combination using the Karman-Trefftz transformation. Since this was a dual element airfoil and the triangulation scheme was capable of discretizing any domain, it was decided to run this test case. The airfoil and flap were tested at a Mach number of 0.125 and an angle of attack of zero. The Mach number was chosen to be low to reduce the compressibility effects yet avoid the problem of numerical noise at ultra-low Mach numbers.

The outer boundary of the mesh was located 40 chords from the half-chord point of the airfoil. The vortex far-field boundary condition was applied. For this case, $\sigma = -1$ was used for second-order accuracy in the MUSCL-differencing formula. Only one method was used to enforce flow tangency at the wall. Flow

tangency was enforced by normal extrapolation of all flow conditions. The equations were not solved at the walls due to time limitations. First-order extrapolation leads to the fastest convergence of the method. The solution was run unlimited so no limiter was used. A closeup of the grid can be seen in Figure 53 for the airfoil and flap. A closeup of the grid about the flap can be seen in Figure 57. The grid was composed of 14587 triangles and 7566 vertices.

Mach contours, pressure contours, and density contours are present in Figures 54, 55, and 56, respectively of the airfoil and flap. The pressure and density contours appear to have some noise present on the lower surface of the airfoil but are smooth overall. See Figures 58, 59, and 60, to see a blowup of the Mach contours, pressure contours, and density contours about the airfoil-flap junction and the flap. Surface pressure comparisons are made on both the airfoil and flap in Figures 61 and 62. The solutions are compared to the Karman-Trefftz [90] incompressible solution. Predicted pressures on the main airfoil agree very well with the Karman-Trefftz solution. Predicted pressures on the flap agree fairly well except at locations near the leading edge. Apparently the mesh was not fine enough at the airfoil-flap junction. The leading edge of the flap does not appear to be geometrically resolved very well in Figure 57.

Using the Prandtl-Galuert Rule, the differences between the analytical incompressible solution and the predicted solution should be less than one percent. The predicted lift and drag coefficients are in fair agreement with the analytical incompressible solution.

Table 1. Comparison of Lift and Drag Coefficients for Karman-Treffetz Airfoil-Flap.

	Main Airfoil		Flap		Total	
	C_l	C_d	C_l	C_d	C_l	C_d
Present Work	1.7328	-0.0931	0.3379	0.0884	2.0707	-0.0046
Analytical Solution	1.6915	-0.0898	0.3366	0.0897	2.0281	-0.0001

VI. CONCLUDING REMARKS

The objective of this research was to apply upwind schemes to a mesh of unstructured triangles in two-dimensions and obtain the accuracy that was possible on a structured mesh using upwind methods. Achieving this accuracy required that MUSCL-differencing also be extended to a mesh of unstructured triangles.

Discretizing two-dimensional domains with triangles, gives the CFD researcher the ability to solve any two-dimensional problem. Triangles will discretize multiply-connected domains of any geometric complexity. The distribution of points or nodes can be controlled by the CFD researcher. Structured mesh solvers tend to have very poor point distribution about complicated multiply-connected domains. Currently, mesh generators exist which can discretize any two-dimensional domain with excellent control of point distribution and grid stretching. The complexity of the grid generation process increases relative to structured mesh generators, but this is a small problem when one considers the additional flow solving capabilities of unstructured meshes.

Considering the available results, the author believes that the application and use of upwind schemes on unstructured meshes is as accurate as the use of upwind schemes on structured meshes. Cases where the unstructured mesh was as fine as the structured mesh, yet the shock was more smeared, were caused by the structured mesh aligning with the flow features. Structured grid solutions about the blunt body and airfoil test cases, possessed grids that had grid lines that tended to line up with the shock wave. Structured grids for airfoils, such as O and C meshes, have grid lines that lay almost perpendicular to the airfoil surface. Transonic airfoils tend to have shock waves on the aft part of the airfoil that are perpendicular to the airfoil surface. The blunt body grid used by Walters, had grid lines that lay approximately parallel to the body surface. The bow shock was also parallel to the body surface. One-dimensional upwind schemes which have been applied in multiple-dimensions will capture shock waves much more crisply if they are oriented normal to the shock. The unstructured grid solver has grid cells at all orientations to the flow. No direction is preferred by the unstructured solver. For these geometrically simple test cases, the structured grids are almost a perfect match. The match between the structured grid and the flow phenomena is fortuitous. So, it is not surprising that the upwind structured solvers yield a slightly better result than the unstructured solvers.

Unstructured solvers have the ability to employ mesh adaptation much more effectively than structured meshes. Since the unstructured mesh has no underlying connectivity, grid points and elements can easily and naturally be added to the original grid. Structured meshes can use grid adaptation by moving

(stretching) the mesh or by adding grid points. Mesh stretching has severe limitations. Adding grid points to a structured grid requires that an unstructured way of storing the connectivity of the mesh be used. A sophisticated grid adaptation scheme can adapt to one-dimensional flow features such as shocks and optimize the use of the upwind flow solvers. Mesh adaptation could also be used to remove grid points in regions of small or null gradients. Overall use of grid points would be optimized by a mesh adaptation algorithm.

For multiply-connected two-dimensional domains, the structured grid solvers would lose the advantage of lining up with the shock wave, and the unstructured solvers would be superior. Current research interests in CFD are directed toward the solution of the Euler equations about realistic vehicles in three-dimensions. In three-dimensions, a structured grid may not even be easily created and it certainly would not have a distribution of grid points which would match the flowfield. Mesh adaptation would only improve the abilities of the three-dimensional solver. Three-dimensional solvers tend to need an enormous number of grid points. Using mesh adaptation in three-dimensions would optimize the number of grid points for a specified accuracy.

Unstructured grid solvers will always be slower than a comparable structured grid solver. Many of the fast algorithms for solving CFD problems exploit the structure of the grid and hence data. Even if fast algorithms are developed for an unstructured mesh solver, it will still be necessary to store the connectivity of the grid and compute the storage locations of flow variables. The primary reason

for the slow convergence of the present research was the use of explicit time integration schemes with upwind solvers. Upwind solvers tend to work poorly with explicit schemes. Use of implicit time integration with domain splitting will make the unstructured solver more competitive with the structured solver. Mesh adaptation will virtually guarantee that an unstructured mesh solver will use fewer grid points than a structured mesh solver for comparable accuracy.

The surface boundary conditions still remain a problem. Solving the equations at the wall, with entropy correction, tends to give the most accurate results but there were oscillations. Extrapolation of all conditions with a pressure correction at the wall was best for a few problems and gave results comparable to solving the equations at the wall. Pure extrapolation was relatively inaccurate.

Unstructured solvers, using mesh adaptation, should be capable of sufficient grid resolution in boundary layers to solve the Navier-Stokes equations. The viscous flux vectors can easily be computed on a dual mesh and preliminary work has been done.

In the author's opinion, unstructured grids are the natural approach to take to solve flows about realistic bodies in three-dimensions that are of current and future research interest.

REFERENCES

1. Chapman, D.R., "Computational Aerodynamics Development and Outlook", Dryden Lecture, *AIAA Journal*, Vol. 17, 1979, pp. 1293-1313.

2. Lomax, H., "Some Prospects for the Future of Computational Fluid Dynamics", *AIAA Computational Fluid Dynamics Conference*, June 1981, pp. 3-15.

3. Rubbert, P.E. and Tinoco, E.N., "Impact of Computational Methods on Aircraft Design", AIAA Paper 83-2060, August 1983.

4. Tunlinius, J.R., Bonner, E., Shankar, V., "Impact of Computational Aerodynamics on Aircraft Design", AIAA Paper 83-2062, August 1983.

5. Jameson, A., "Successes and Challenges in Computational Aerodynamics", AIAA Paper 87-1184, June 1987.

6. Anderson, D.A., Tannehill, J.C., Pletcher, R.H., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, 1984.

7. Rogallo, R.S., "Numerical Experiments in Homogeneous Turbulence", NASA TM-81315, 1981.

8. Murman, E.M. and Cole, J.D., "Calculation of Plane Steady Transonic Flows", *AIAA Journal*, Vol. 9, 1971, pp. 114-121.

9. South Jr., J.C. and Brandt, A., "Application of a Multi-Level Grid Method to Transonic Flow Calculations", *Transonic Flow Problems In Turbomachinery*, Hemisphere Publishing Co., 1977, pp. 180-207.

10. Boppe, C.W., "Computational Transonic Flow about Realistic Aircraft Configurations", AIAA Paper 78-104, January 1978.

11. Boppe, C.W., "Computational Transonic Flow about Realistic Aircraft Configurations", NASA CR 3243, May 1980.

12. Jameson, A., "Iterative Solution of Transonic Flows Over Airfoils and Wings, Including Flows at Mach 1", *Communications in Pure Applied Mathematics*, Vol. 27, 1974, pp. 283-309.

13. Jameson, A. and Caughy, D.A., "A Finite Volume Method for Transonic Potential Flow Calculations", AIAA Paper 77-635, June 1977.

14. Grossman, B. and Siclari, M.J., "Nonlinear Supersonic Potential Flow over Delta Wings", *AIAA Journal*, Vol. 19, no. 5, May 1981, pp. 573-581.

15. Shankar, V. and Osher, S., "An Efficient Full Potential Implicit Method based on Characteristics for Analysis of Supersonic Flows", AIAA Paper 82-0974, June 1982.

16. Shankar, V., "Conservative Full Potential, Implicit Marching Scheme for Supersonic Flows", AIAA Paper 81-1004, June 1981.

17. Clever, W.C. and Shankar, V., "Non-linear Potential Analysis Techniques for Supersonic/Hypersonic Configuration Design", NASA CR 166078, March 1983.

18. Siclari, M.J., "Computation of Nonlinear Supersonic Potential Flow Over Three-Dimensional Surfaces", *Journal of Aircraft*, Vol. 20, no. 5, May 1983, p. 462.

19. Siclari, M.J., "The NCOREL Computer Program for 3-D Nonlinear Supersonic Conical Flows", NASA CR 3694, August 1983.

20. Magnus, R. and Yoshihara, H., "Inviscid Transonic Flow Over Airfoils", *AIAA Journal*, Vol. 8, 1970, pp. 2157-2162.

21. Grossman, B. and Moretti, G., "Time-Dependent Computation Of Transonic Flows", AIAA Paper 70-1322, October 1970.

22. MacCormack, R.W., "The Effect of Viscosity in Hyper-Velocity Impact Cratering", AIAA Paper 69-354, 1969.

23. Beam, R.W. and Warming, R.F., "An Implicit Finite Difference Algorithm for Hyperbolic System in Conservation Form", *Journal of Computational Physics*, Vol. 23, 1976, pp. 87-110.

24. Jameson, A., Schmidt, W. and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes", AIAA Paper 81-1259, 1981.

25. Ni, Ron Ho., "A Multiple Grid Scheme for Solving the Euler Equations", *AIAA Journal* , Vol. 20, 1982, pp. 1565-1571.

26. Jameson, A., "Solution of the Euler Equations by a Multigrid Method", *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327-356.

27. Steger, J.L. and Warming, R.F., "Flux Vector Splitting of the Inviscid Gas Dynamics Equations with Applications to Finite Difference Methods", *Journal of Computational Physics*, Vol. 40, 1981, pp. 263-293.

28. Van Leer, B., "Flux-Vector Splitting for the Euler Equations", ICASE Report 82-30, September 1982, also in *Lecture Notes in Physics*, Vol. 170, 1982, pp. 507-512.

29. Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes", *Journal of Computational Physics*, Vol. 43, 1981, pp. 357-372.

30. Roe, P.L., "Characteristic Based Schemes for the Euler Equations", *Annual Review of Fluid Mechanics*, Vol. 18, 1986, pp. 337-365.

31. Godunov, S.K., "Finite-Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics", *Mat. Sbornik*, Vol. 47, 1959, pp. 271-306. (In Russian).

32. Enquist, B. and Osher, S., "Stable and Entropy Satisfying Approximations for Transonic Flow Calculations", *Math. Comp.*, Vol. 34, 1980, pp. 45-75.

33. Van Leer, B., "Computational Methods for Ideal Compressible Flow", *Von Karman Institute for Fluid Dynamics, Lecture Series 1983-04 Computational Fluid Dynamics*, March 7-11 1983.

34. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godonuv's Method", *Journal of Computational Physics*, Vol. 32, 1979, pp. 101-136.

35. Van Albada, G.D., Van Leer, B. and Roberts Jr., W.W., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics", *Astron. Astrophys.*, Vol. 108, 1982, pp. 76-84.

36. Sweby, P.K., "High Resolution Schemes Using Flux Limiters For Hyperbolic Conservation Laws", *Siam J. Numer. Anal.*, Vol. 21, No. 5, 1984.

37. Thomas, J.L., Van Leer, B. and Walters, R.W., "Implicit Flux-Split Schemes for the Euler Equations", AIAA Paper 85-1680, July 1985.

38. Thomas, J.L. and Walters, R.W., "Upwind Relaxation Algorithms for the Navier-Stokes Equations", Proc. AIAA 7th CFD Conf., AIAA Paper 85-1501-CP, July 1985.

39. Moretti, G., "The Lambda Scheme", *Computers and Fluids*, Vol. 7, 1979, pp. 191-205.

40. Moretti, G., "An Efficient Euler Solver, With Many Applications", AIAA Paper 87-0352, January 1987.

41. Moretti, G. and Lippolis, A., "Shock-Fitting Calculations of Airfoils and Intakes", GAMM Workshop, Rocquencourt (France), June 1986, to appear in *Notes on Num. Fl. Dyn.*, Vieweg Publ., 1987.

42. Van Leer, B., Thomas, J.L., Roe, P.L., and Newsome, R.W., "A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations", AIAA Paper 87-1104, June 1987.

43. Thompson, Joe F., Warsi, Z.U.A. and Mastin, C. Wayne, *Numerical Grid Generation, Foundations and Applications*, Elsevier Publishing Co., Inc., New York, 1985.

44. Brackbill, J.U., "Coordinate System Control: Adaptive Meshes", *Numerical Grid Generation, Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and their Use in the Numerical Solution Of Partial Differential Equations*, J.F. Thompson, editor, Elsevier Publishing Co., Inc., New York, pp. 277-294.

45. Dannenhoffer III, J.F. and Baron, J.R., "Robust Grid Adaptation for Complex Transonic Flows", AIAA Paper 86-0495, January 1986.

46. Nakahashi, K. and Deiwert, G.S., "A Self-Adaptive-Grid Method with Application to Airfoil Flow", AIAA Paper 85-1525, July 1985.

47. Rai, M.M., "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations", AIAA Paper 84-0164, January 1984.

48. Rai, M.M., "An Implicit, Conservative, Zonal-Boundary Scheme for Euler Equation Calculations", AIAA Paper 85-0488, January 1985.

49. Hessenius, K.A. and Rai, M.M., "Three Dimensional, Conservative, Euler Computations using Patched Grid Systems and Explicit Methods", AIAA Paper 86-1081, May 1986.

50. Belk, D.M. and Whitfield, D.L., "Time Accurate Euler Equations Solutions on Dynamic Blocked Grids", AIAA Paper 87-1127, June 1987.

51. Walters, R.W., Reu, T., Thomas, J.L. and McGrory, W.D., "Zonal Techniques for Flowfield Simulation about Aircraft", Presented at *Symposium on Advances and Trends in Computational Structural Mechanics and Fluid Dynamics*, October 17-19 1988.

52. Shaw, J., Forsey, C.R., Weatherill, N.P. and Rose, K.E., "A Block Structured Mesh Generation Technique for Aerodynamic Geometries", "Numerical Grid generation in Computational Fluid Dynamics", Edit. J. Hauser and C. Taylor, Proc. Int. Conf., Landshut, July 1986, Pineridge Press, Swansea, pp. 329-340.

53. Benek, J.A., Buning, P.G. and Steger, J.L., "A 3-D Chimera Grid Embedding Technique", AIAA Paper 85-1523, July 1985.

54. Wedan, B. and South, J.C., "A Method for Solving the Transonic Full-Potential Equation for general Configurations", AIAA Paper 83-1889, July 1983.

55. Grossman, B. and Whitaker, D.L., "Supersonic Flow Computations Using a Rectangular Coordinates Finite-Volume Method", AIAA Paper 86-0442, January 1986.

56. Choi, S.K. and Grossman, B., "A Flux-Vector Split, Finite-Volume Method for Euler's Equations on Non-Mapped Grids", AIAA Paper 88-0227, January 1988.

57. Gaffney, R.L., Hassan, H.A. and Salas, M.D., "Euler Calculations for Wings Using Cartesian Grids", AIAA Paper 87-0356, January 1987.

58. Löhner, R., Morgan, K., Peraire, J. and Zienkiewicz, O.C., "Finite Element Methods for High Speed Flows", AIAA Paper 85-1531, July 1985.

59. Morgan, K., Peraire, J., Thareja, R.R. and Stewart, J.R., "An Adaptive Finite Element Scheme for the Euler and Navier-Stokes Equations", AIAA Paper 87-1172, June 1987.

60. Peraire, J., Peiro, J., Formaggia, L., Morgan, K. and Zienkiewicz, O.C., "Finite-Element Euler Computations in Three-Dimensions", AIAA Paper 88-0032, January 1988.

61. Angrand, F., Dervieux, A., Boulard, V., Periaux, J. and Vijayasundaram, G., "Transonic Euler Simulations by Means of Finite-Element Explicit Schemes", AIAA Paper 83-1924, July 1983.

62. Stoufflet, B., Periaux, J., Fezoui, F. and Dervieux, A., "Numerical Simulation of 3-D Hypersonic Euler Flows Around Space Vehicles Using Adapted Finite Elements", AIAA Paper 87-0560, 1987.

63. Walters, R.W. and Thomas, J.L., "Advances in Upwind Relaxation Methods", *State of the Art Surveys in Computational Mechanics*, Chapter 4, ASME special publication, A.K. Noor, Ed., 1987.

64. Walters, R.W. and Dwoyer, D.L., "An Efficient Iteration Strategy for the Solution of the Euler Equations", AIAA Paper 85-1529, July 1985.

65. Jameson, A. and Mavriplis, D., "Finite-Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh", AIAA Paper 85-0435, January 1985.

66. Mavriplis, D. and Jameson, A., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes", AIAA Paper 87-0353, January 1987.

67. Mavriplis, D. and Jameson, A., "Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes", ICASE Report 87-53, July 1987.

68. Löhner, R. and Parikh, P., "Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method", AIAA Paper 88-0515, January 1988.

69. Löhner, R. and Morgan, K., "An Unstructured Multigrid Method for Elliptic Problems", *Int. J. Num. Meth. Eng. 24*, 1987, pp. 101-115.

70. Hindman, R.G., "Generalized Coordinate Forms of Governing Fluid Equations and Associated geometrically Induced Errors", *AIAA Journal*, Vol. 20, no. 10, October 1982, pp. 1359-1367.

71. Viviand, H., "Conservation Forms of Gas Dynamic Equations", *La Recherche Aerospatiale*, no. 1, January-February 1974, pp. 65-68.

72. Thomas, J.L. and Salas, M.D., "Far-Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations", AIAA Paper 85-0020, January 1985.

73. Whitfield, D.L., Thomas, J.L., Jameson, A. and Schmidt, W., "Computations of Transonic Viscous-Inviscid Interacting Flow", *Proceedings of Second Symposium of Numerical and Physical Aspects of Aerodynamic Flows*, California State University, January 1983.

74. Karamcheti, K., *Priciples of Ideal-Fluid Aerodynamics*, Robert E. Krieger Publishing Company, Inc., Malabar, Florida, 1980.

75. Roe, P.L. and Pike, J., "Efficient Construction and Utilization of Approximate Riemann Solutions", *Computing Methods in Applied Sciences and Engineering, VI*, Glowinski, R. and Lions, J.L. (ed.), Elsevier Science Publishers B.V. (North-Holland), INRIA, 1984.

76. Harten, A., "On a Class of High Resolution Total-Variation-Stable Finite-Difference Schemes", *Siam J. Numer. Anal.*, Vol. 21, No. 1, February 1984.

77. Liou, M-S. and Van Leer, B., "Choice of Implicit and Explicit Operators for the Upwind Differencing Scheme", AIAA Paper 88-0624, January 1988.

78. Anderson, W.K., Thomas, J.L. and Van Leer, B., "A Comparison of Finite Volume Flux Vector Splittings for the Euler Equations", AIAA Paper 85-0122, January 1985.

79. Roe, P.L., "Discrete Models for the Numerical Analysis of Time-Dependent Multidimensional Gas Dynamics", ICASE Report No. 85-18, March 1985, also NASA CR-172574, 1985.

80. Mavriplis, D.J., "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes", *AIAA/ASME/SIAM/APS 1st National Fluid Dynamics Congress*, AIAA Paper 88-3706-CP, pp. 419-426.

81. Usab Jr., W.J., "Embedded Mesh Solutions of the Euler Equation Using a Multiple-Grid Method", Doctoral Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, December 1983.

82. Turkel, E. and Van Leer, B., "Flux-Vector Splitting and Runge-Kutta Methods for the Euler Equations", ICASE Report No. 84-27, June 1984, also in NASA CR-172415, 1984.

83. Van der Houwen, P.J., *Construction of Integration Formulas for Initial Value Problems*, North-Holland Publishing Co., 1977.

84. Hafez, M., Parlette, E. and Salas, M., "Convergence Acceleration of Iterative Solutions of Euler Equations for Transonic Flow Computations", AIAA Paper 85-1641, July 1985.

85. Pulliam, T.H. and Barton, J.T., "Euler Computations of AGARD Working Group 07 Airfoil Test Cases", AIAA Paper 85-0018, AIAA 23rd Aerospace Sciences Meeting, January 1985.

86. Lock, R.C., "Test Cases for Numerical Methods in Two-Dimensional Transonic Flows", AGARD Report No. 575, November 1970.

87. Thomas, J.L., Walters, R.W., Van Leer, B., and Anderson, W.K., "Implicit Finite-Volume Algorithms for the Flux-Split Euler equations", 1986 GAMM Workshop on Euler Solvers.

88. Lerat, A. and Sides, J., "A New Finite Volume Method for the Euler Equations with Applications to Transonic Flows", *Numerical Methods in Fluid Dynamics*, edited by P.L. Roe, Academic Press, New York, NY, 1982, pp. 245-288.

89. Walters, R.W., Private Communication.

90. Williams, B.R., "An Exact Test Case for the Plane Potential Flow About Two Adjacent Lifting Aerofoils", Aeronautical Research Council, Reports and Memoranda No. 3717, London, 1973.

Figures

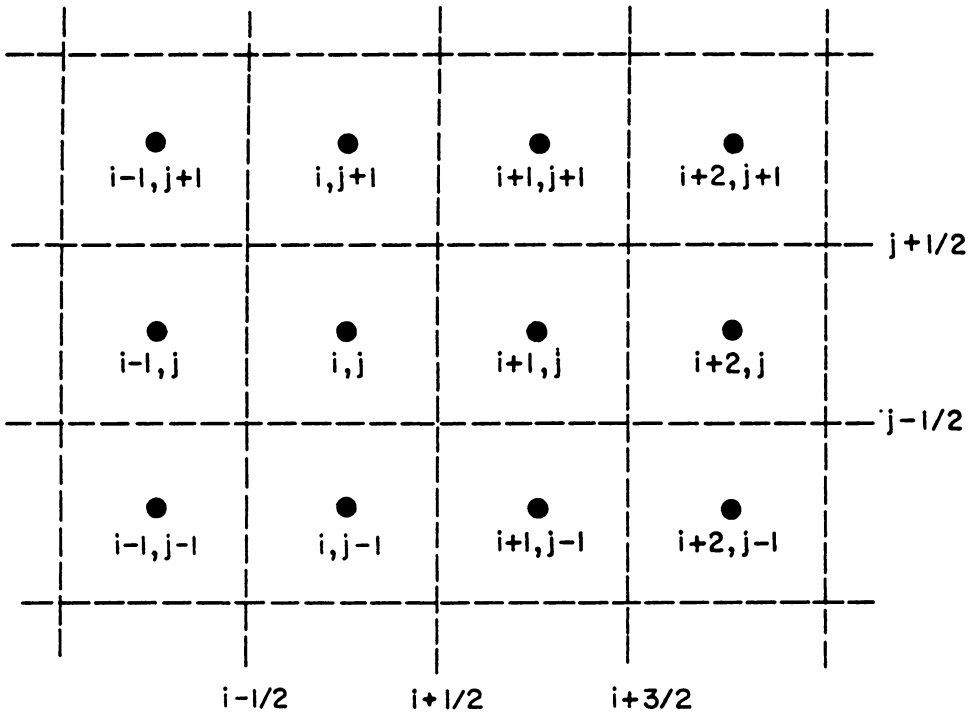

Figure 1. Representative cell-centered structured grid.

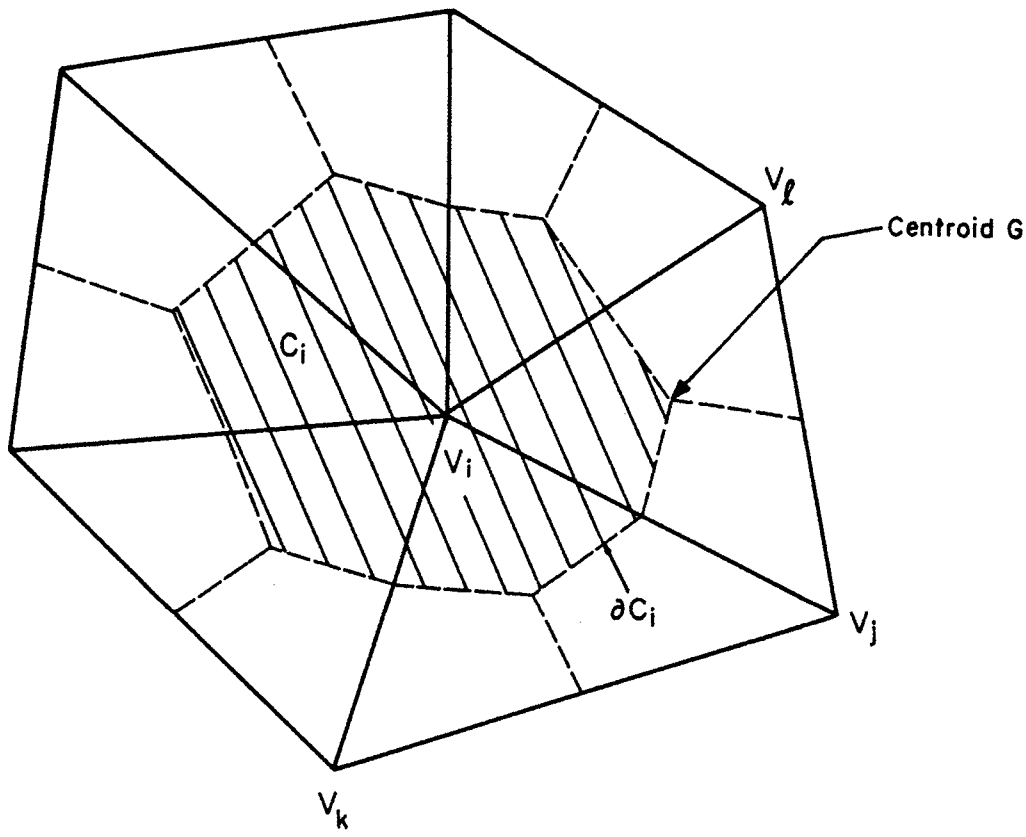

Figure 2. Representative unstructured grid and dual mesh cell.

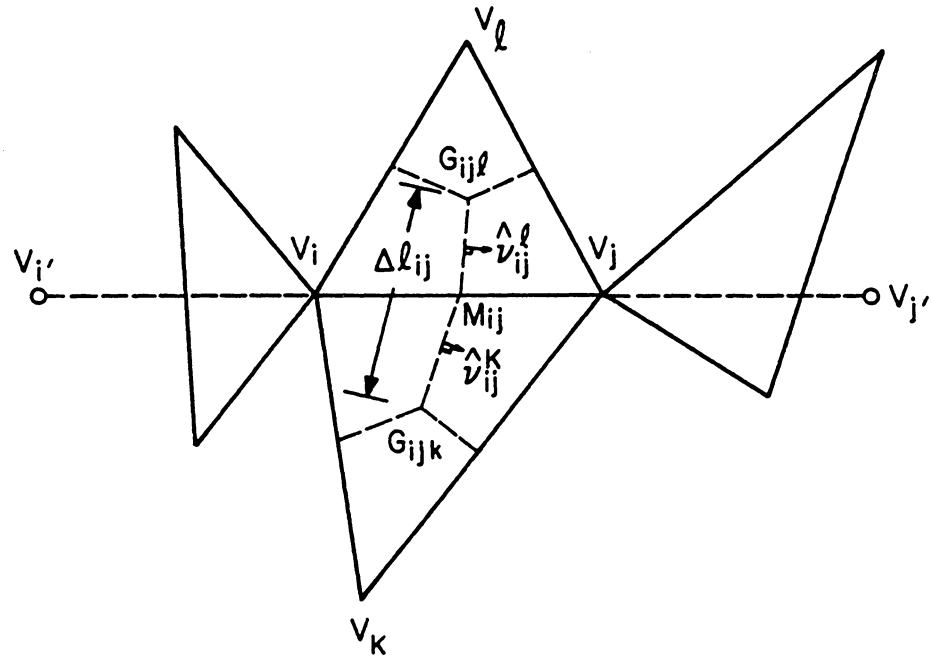

Figure 3. Representative edge of an unstructured mesh.

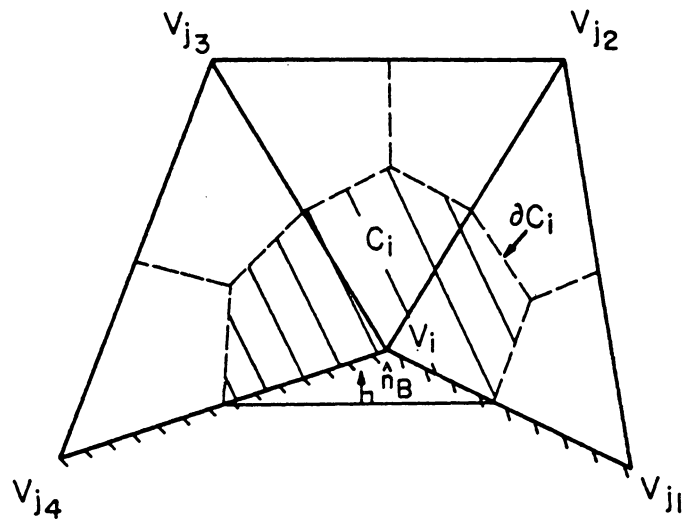

Figure 4. Representative dual mesh cell at grid boundary of an unstructured mesh.

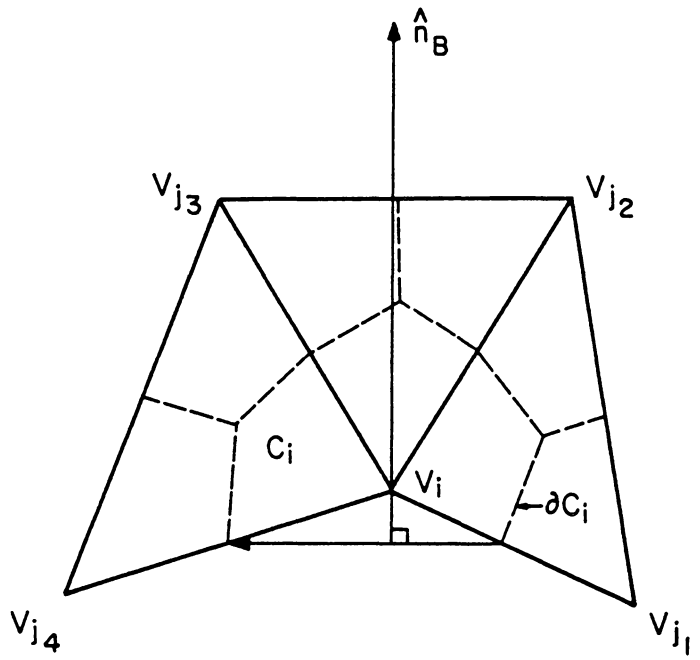

Figure 5. Normal extrapolation of boundary conditions on an unstructured mesh.

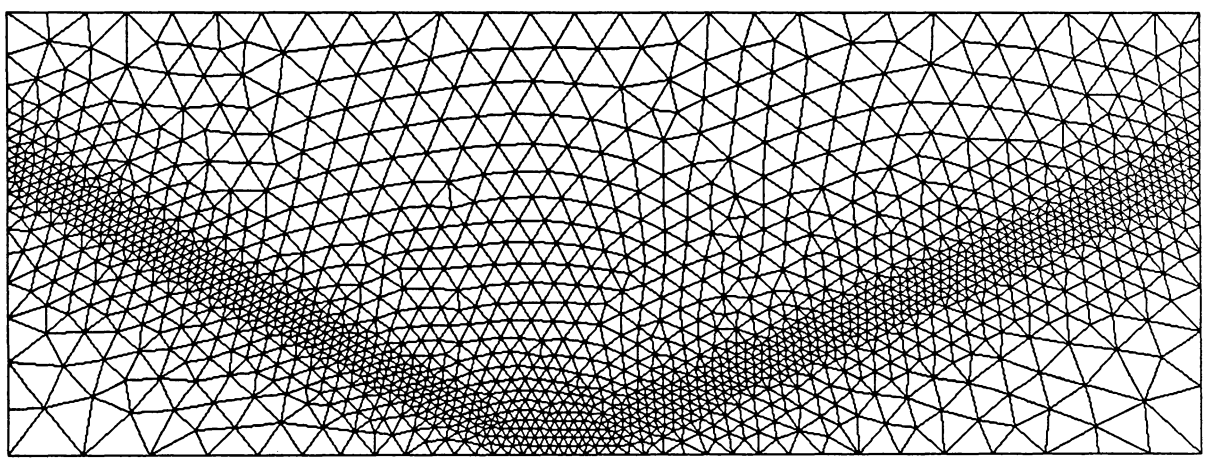

Figure 6. Unstructured mesh for the shock reflection problem: 3291 elements, 1697 nodes, 101 bnodes.

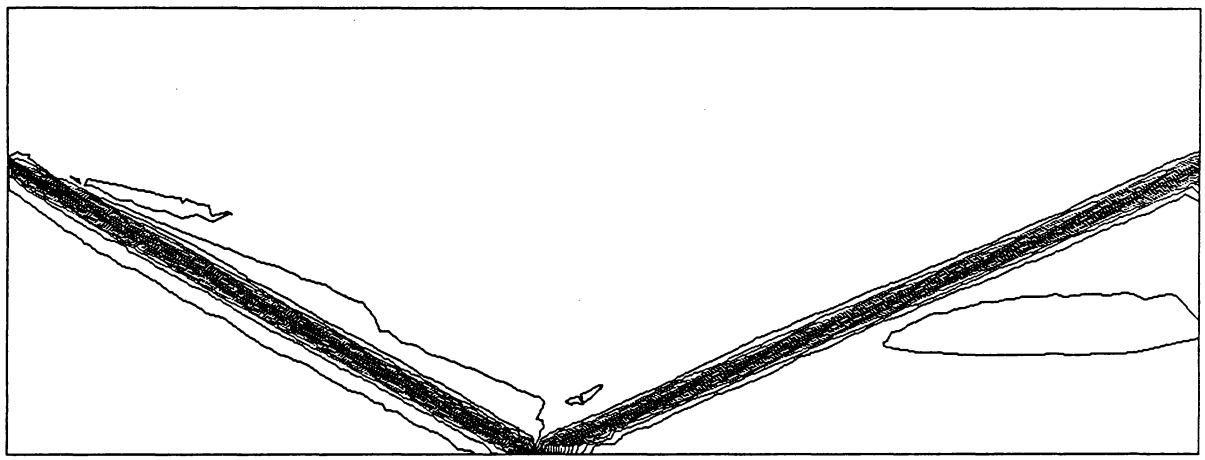

Figure 7. Mach Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$: $\sigma = 1/3$, 3291 elements, 1697 nodes, 101 bnodes, Equations solved at the wall.

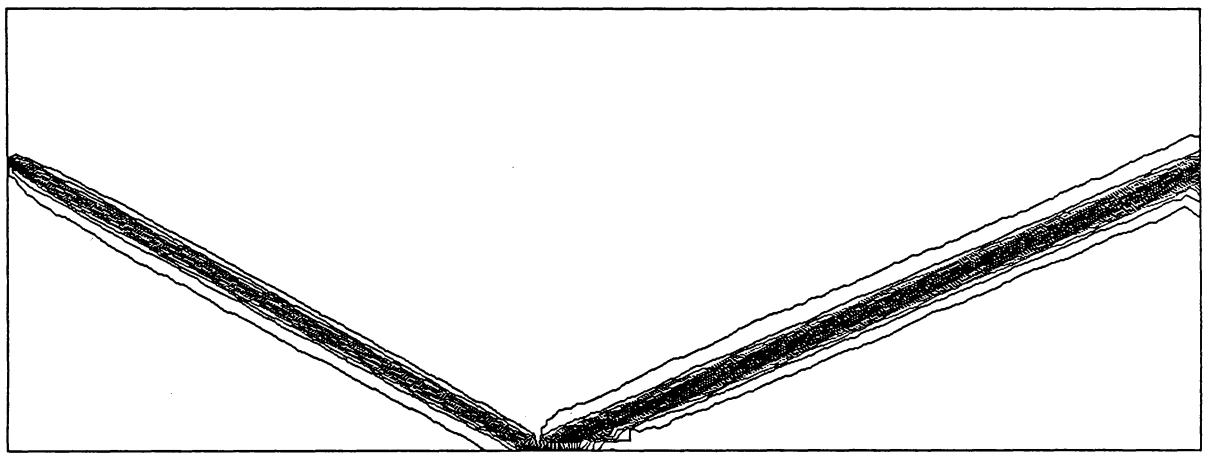

Figure 8. Pressure Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$: $\sigma = 1/3$, 3291 elements, 1697 nodes, 101 bnodes, Equations solved at the wall.

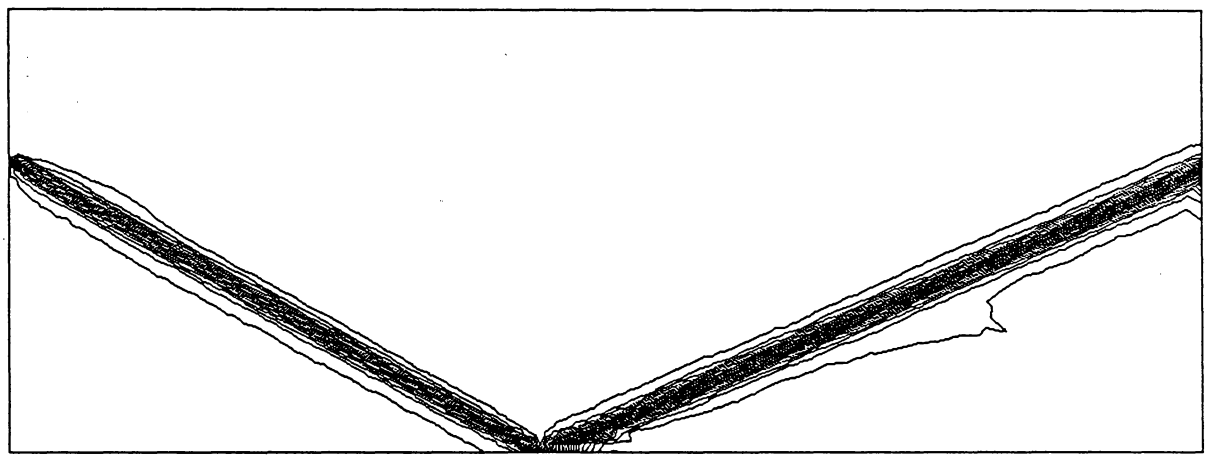

Figure 9. Density Contours - Shock reflecting from a flat plate, $M_\infty = 2.9$: $\sigma = 1/3$, 3291 elements, 1697 nodes, 101 bnodes, Equations solved at the wall.

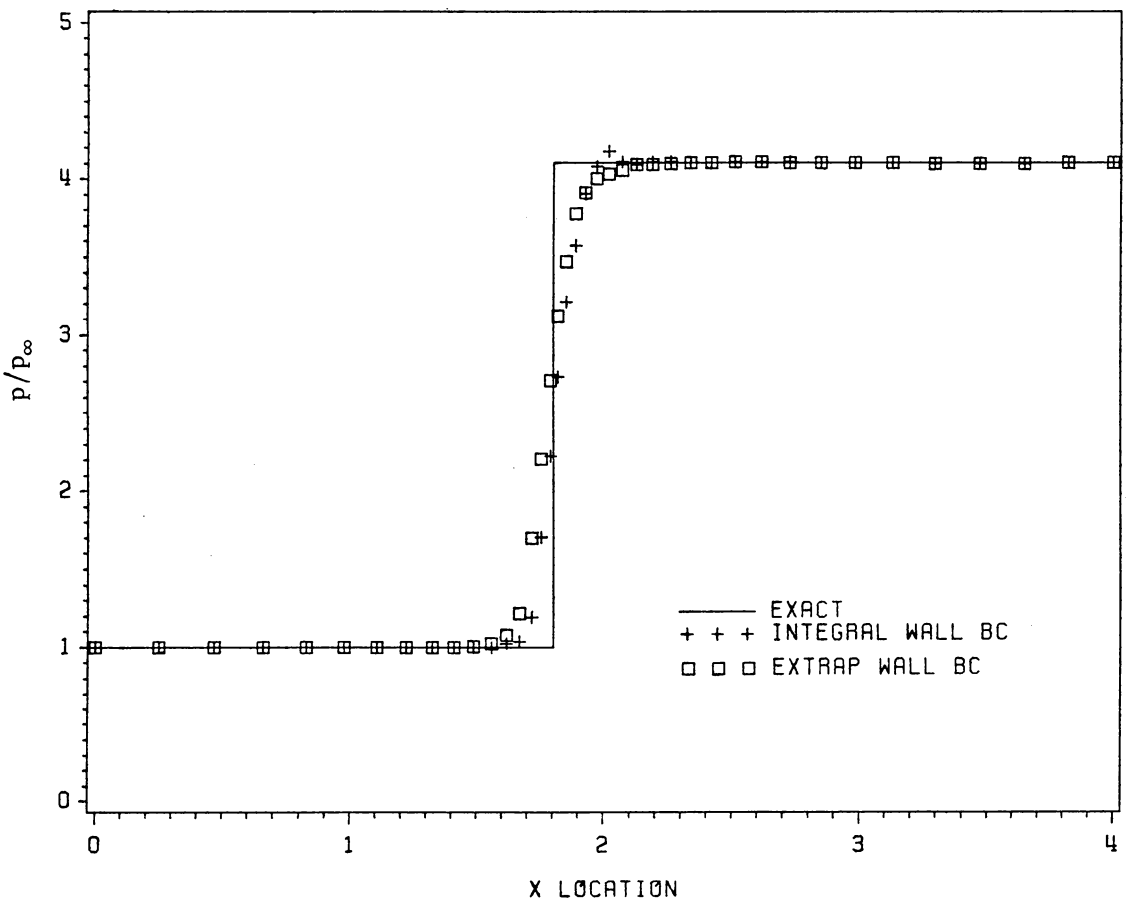

Figure 10. Wall Pressure Comparison - Shock reflecting from a flat plate, $M_\infty = 2.9$: $\sigma = 1/3$, 3291 elements, 1697 nodes, 101 bnodes, Solving the equations at the wall, normal extrapolation and the exact solution.

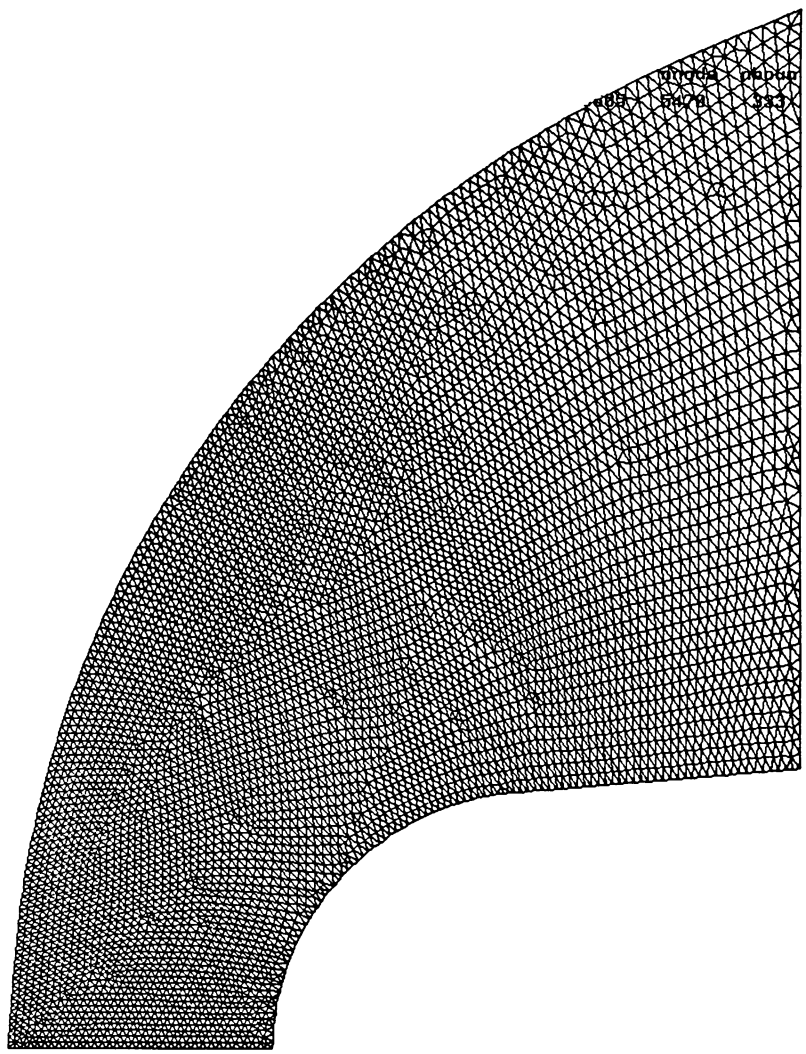

Figure 11. Unstructured Mesh for the Blunt Body Problem: 10605 elements, 5470 nodes, 333 bnodes.

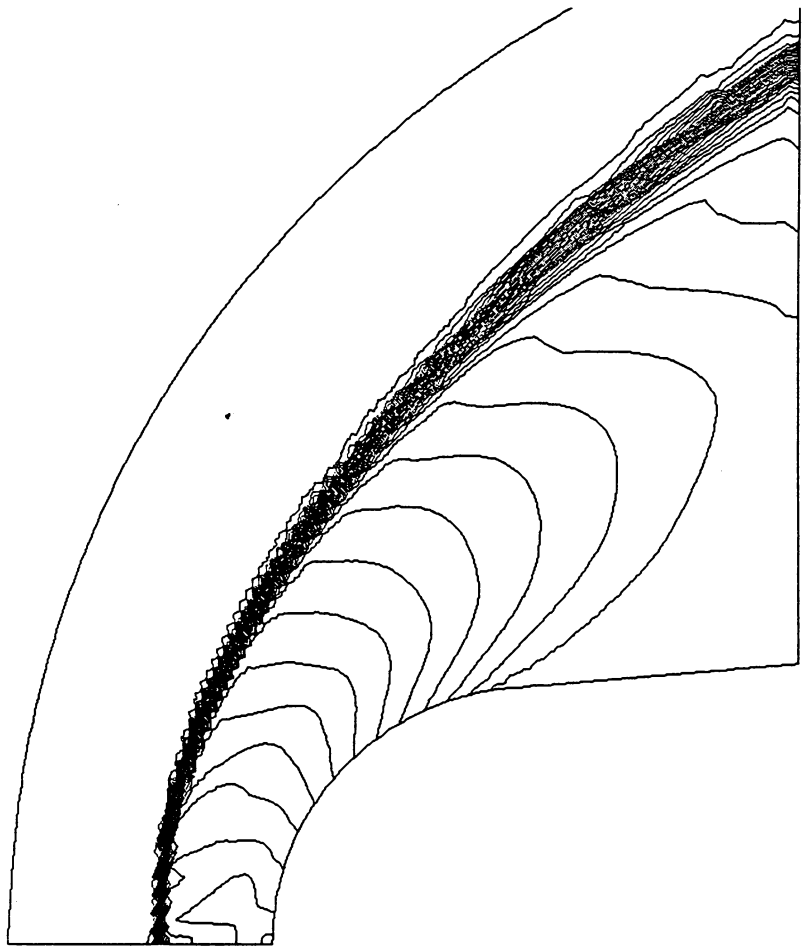

Figure 12. Mach Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$: $\sigma = -1$, 10605 elements, 5470 nodes, 333 bnodes, Normal extrapolation plus wall pressure correction by normal momentum equation.

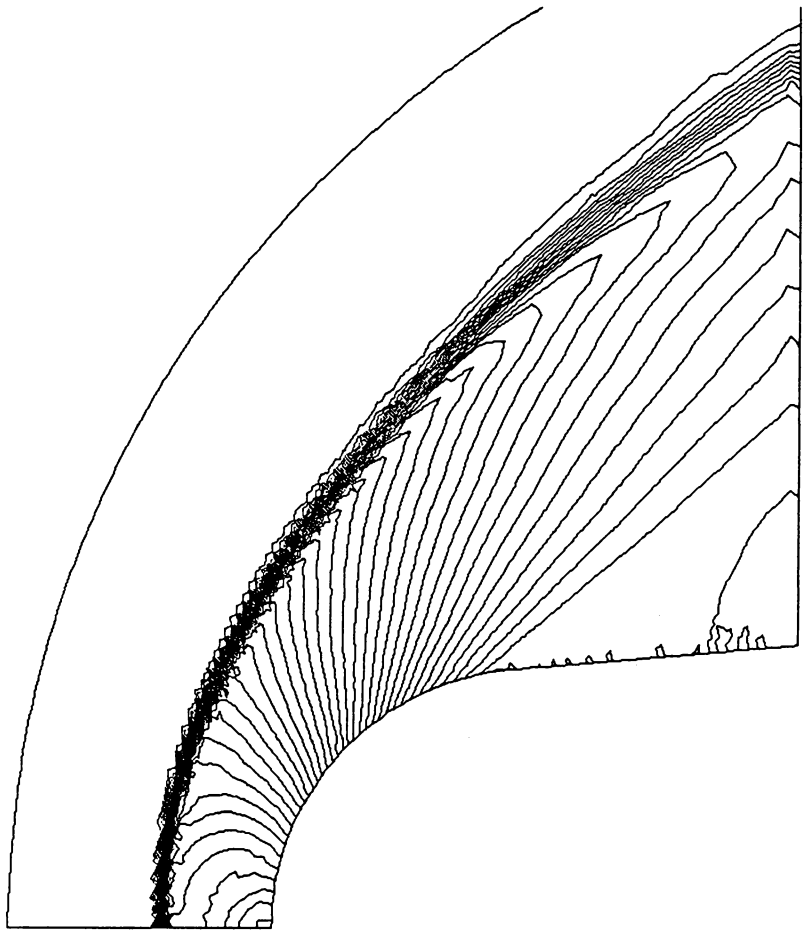

Figure 13. Pressure Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$: $\sigma = -1$, 10605 elements, 5470 nodes, 333 bnodes, Normal extrapolation plus wall pressure correction by normal momentum equation.

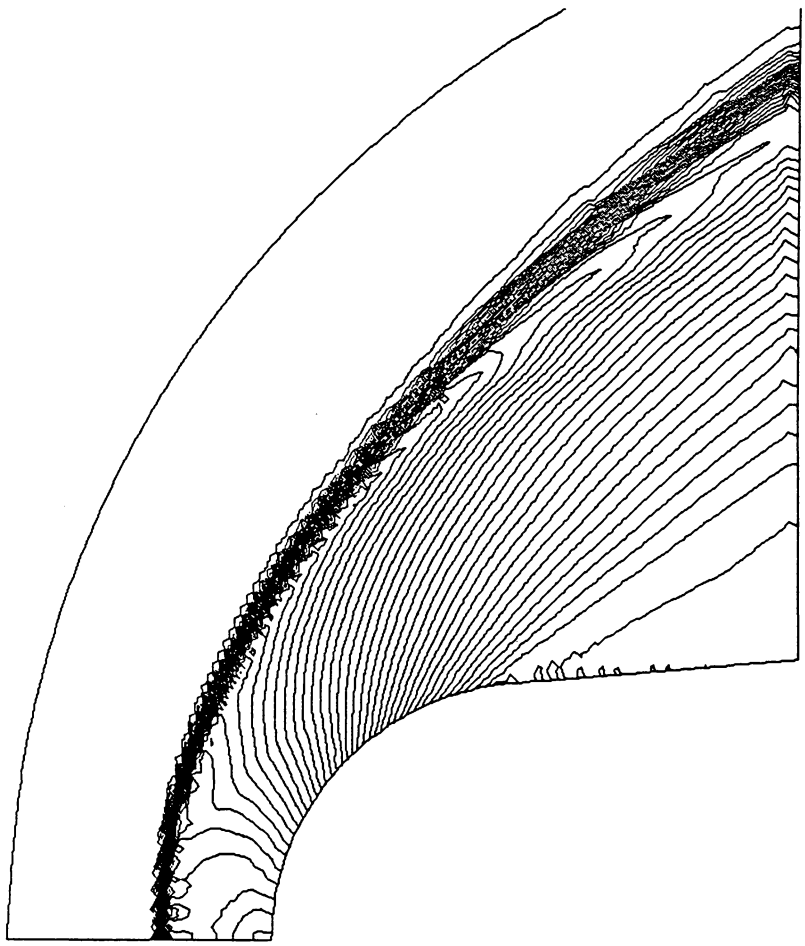

Figure 14. Density Contours - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$: $\sigma = -1$, 10605 elements, 5470 nodes, 333 bnodes, Normal extrapolation plus wall pressure correction by normal momentum equation.

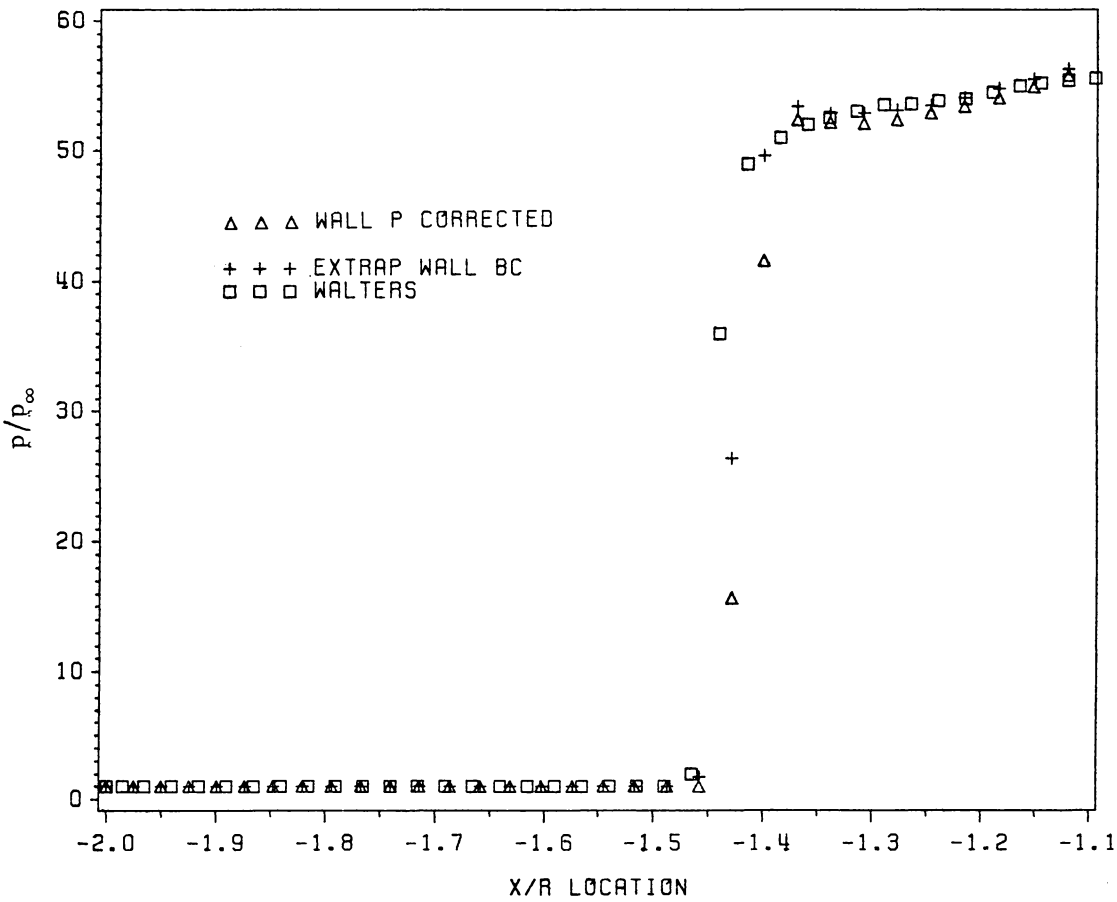

Figure 15. Symmetry Line Pressure Comparison - Blunt Body, $M_\infty = 6.57$, $\gamma = 1.38$: $\sigma = -1$, 10605 elements, 5470 nodes, 333 bnodes, Comparison of normal extrapolation, normal extrapolation plus pressure correction and Walters solution.

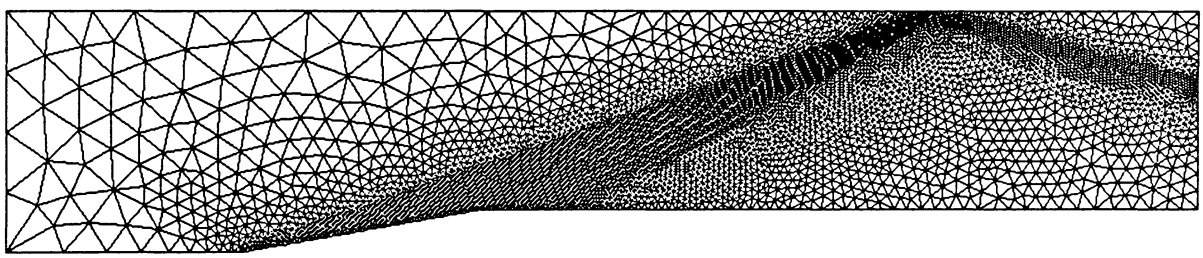

Figure 16. Unstructured Mesh for the 10° Wedge Inlet: 12441 elements, 6344 nodes, 245 bnodes.

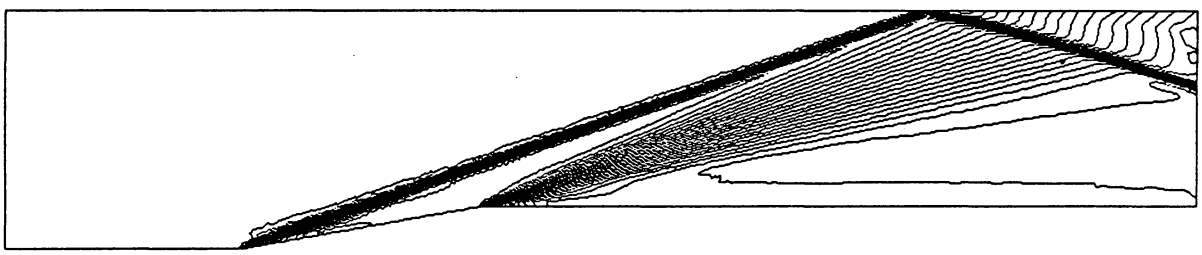

Figure 17. Mach Contours - 10° Wedge Inlet, $M_\infty = 5$: $\sigma = 1/3$, 12441 elements, 6344 nodes, 245 bnodes, Equations solved at the wall.

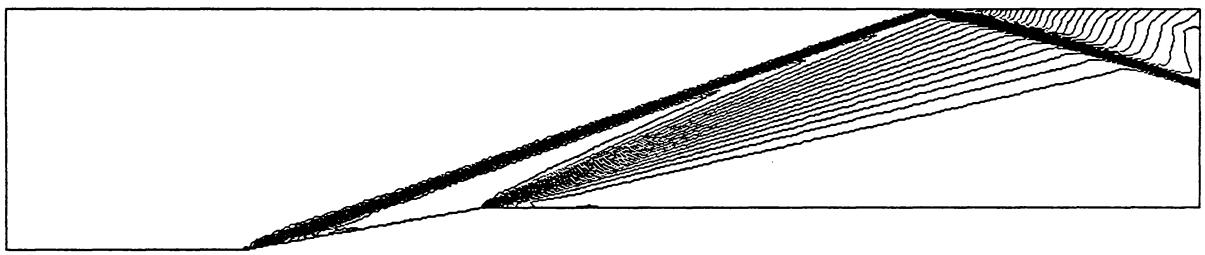

Figure 18. Pressure Contours - 10° Wedge Inlet, $M_\infty = 5$: $\sigma = 1/3$, 12441 elements, 6344 nodes, 245 bnodes, Equations solved at the wall.

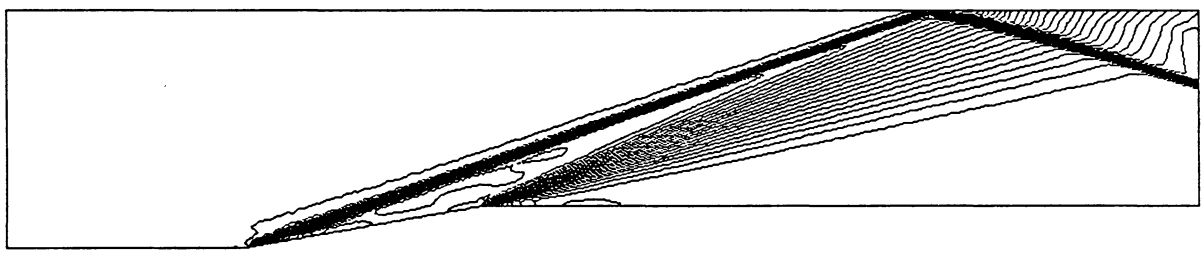

Figure 19. Density Contours - 10° Wedge Inlet, $M_\infty = 5$: $\sigma = 1/3$, 12441 elements, 6344 nodes, 245 bnodes, Equations solved at the wall.

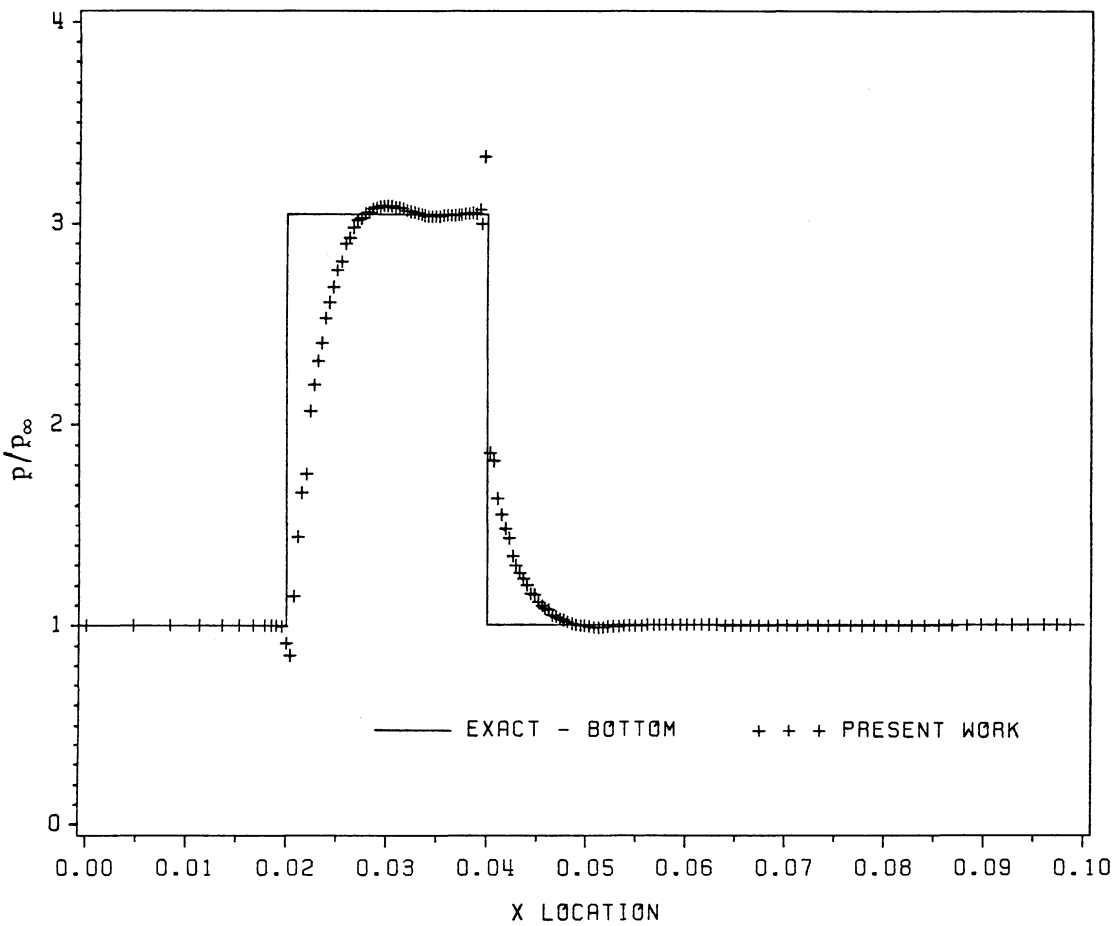

Figure 20. Lower Wall Pressure Comparison - 10° Wedge Inlet, $M_\infty = 5$: $\sigma = 1/3$, 12441 elements, 6344 nodes, 245 bnodes, Equations solved at the wall versus the exact solution.

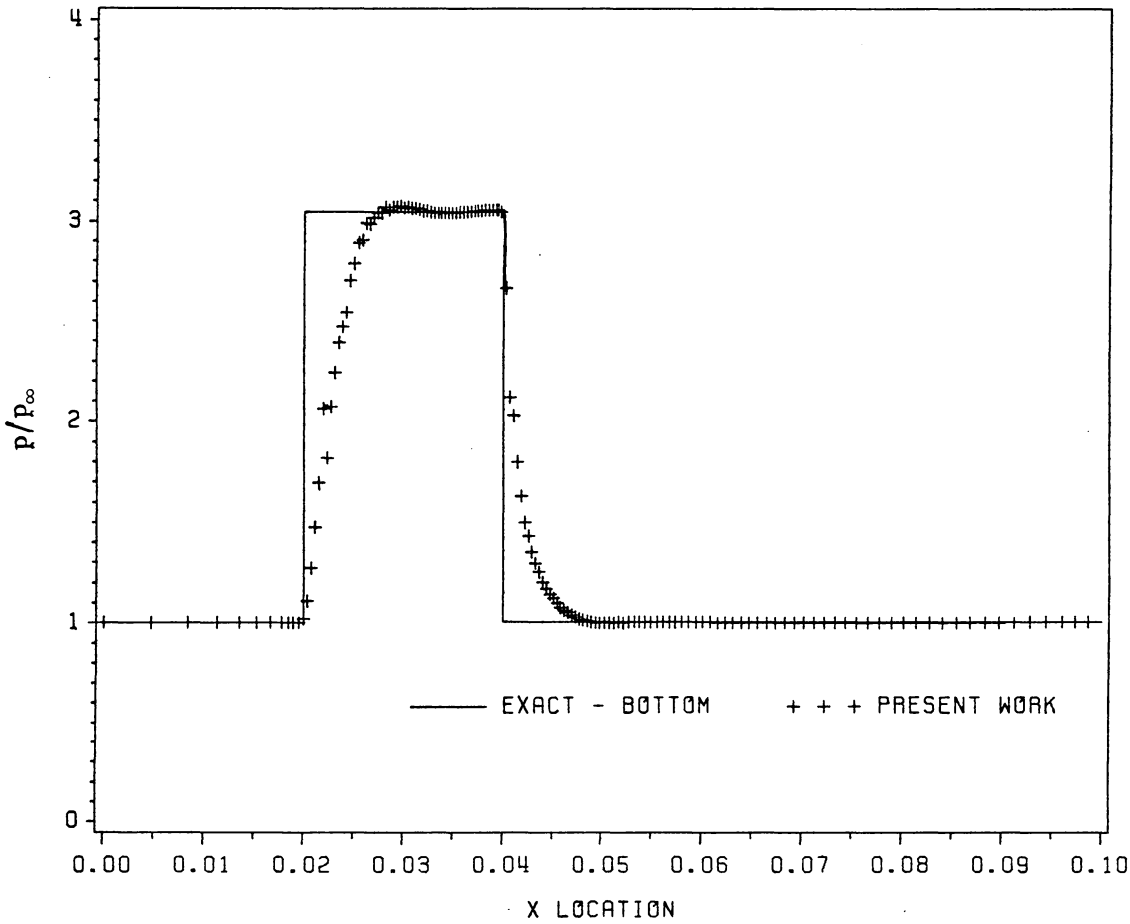

Figure 21. Lower Wall Pressure Comparison - 10° Wedge Inlet, $M_\infty = 5$: $\sigma = 1/3$, 12441 elements, 6344 nodes, 245 bnodes, Normal extrapolation versus the exact solution.

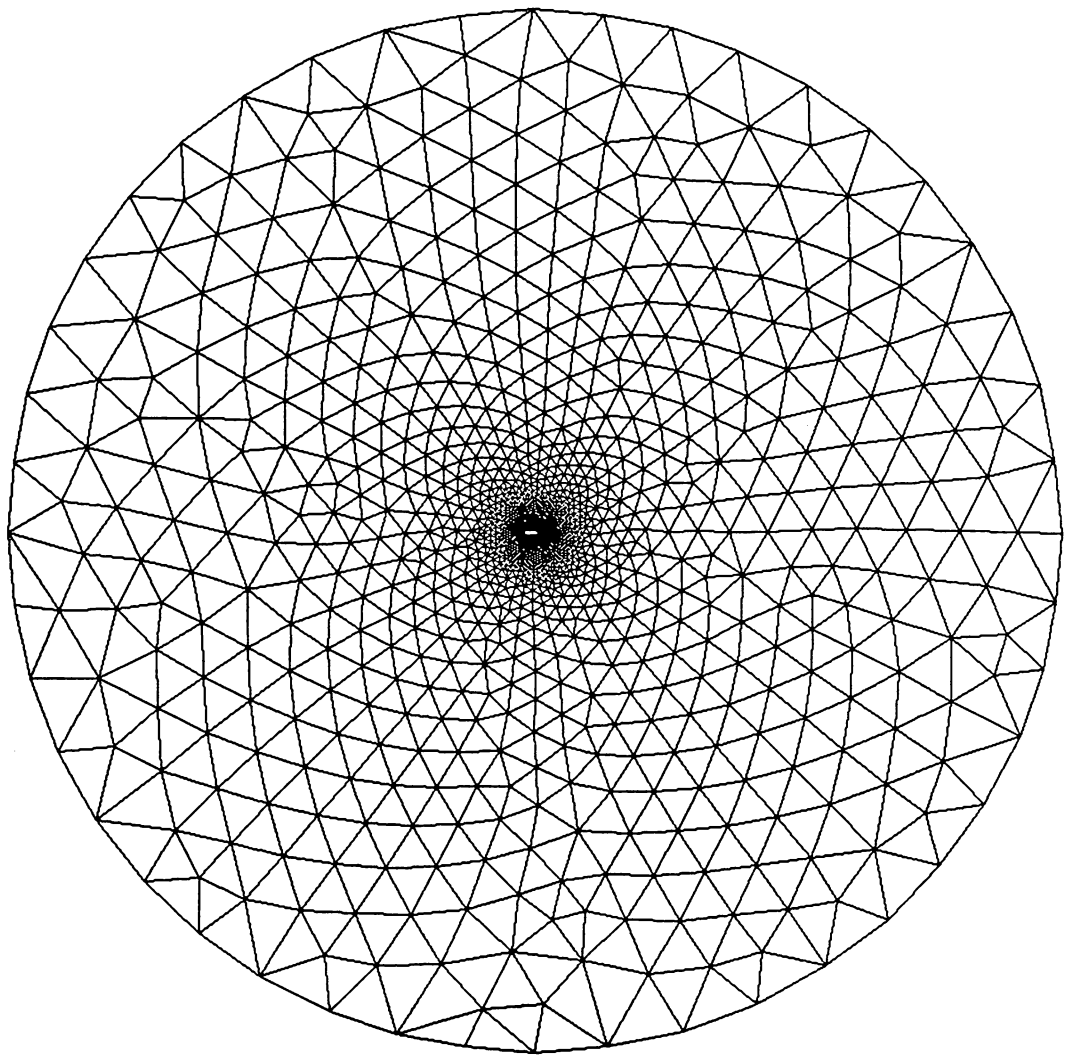

Figure 22. Unstructured Grid - General Far-Field View of Airfoil Grids: Grid radius is 20 chords.

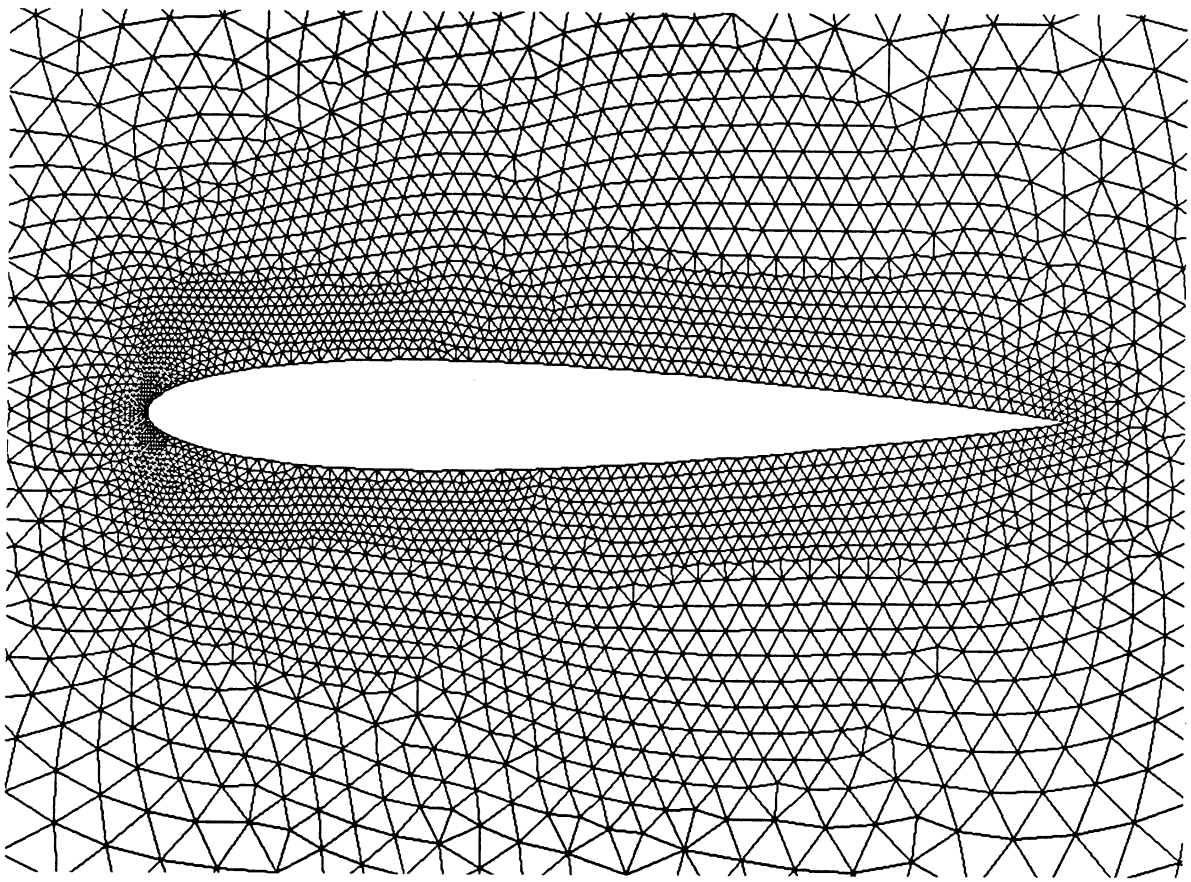

Figure 23. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$: 8381 elements, 4291 nodes, 201 bnodes

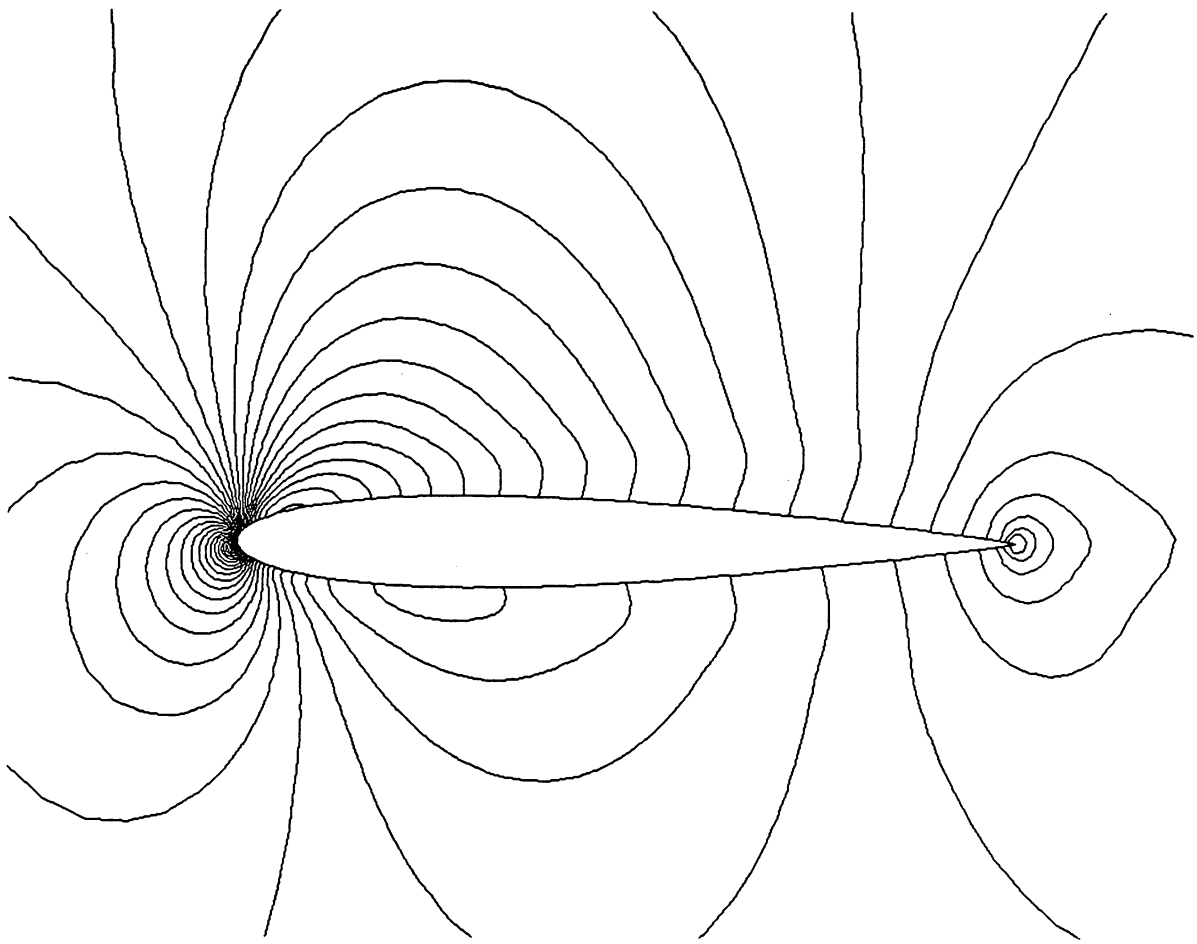

Figure 24. Mach Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$: $\sigma = -1$, $C_l = 0.33209$, $C_d = -0.00039$, 8381 elements, 4291 nodes, 201 bnodes, Equations solved at the wall.

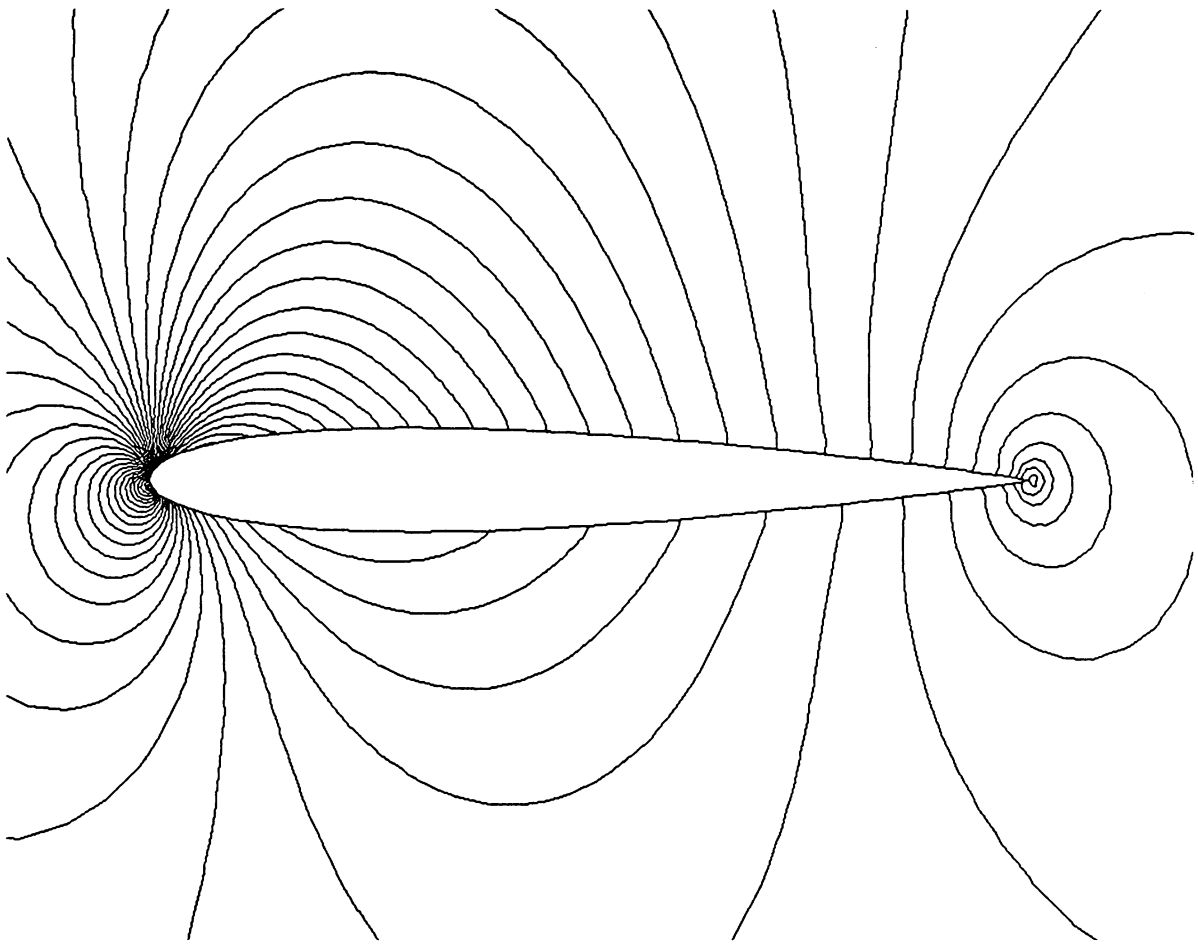

Figure 25. Pressure Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$: $\sigma = -1$, $C_l = 0.33209$, $C_d = -0.00039$, 8381 elements, 4291 nodes, 201 bnodes, Equations solved at the wall.

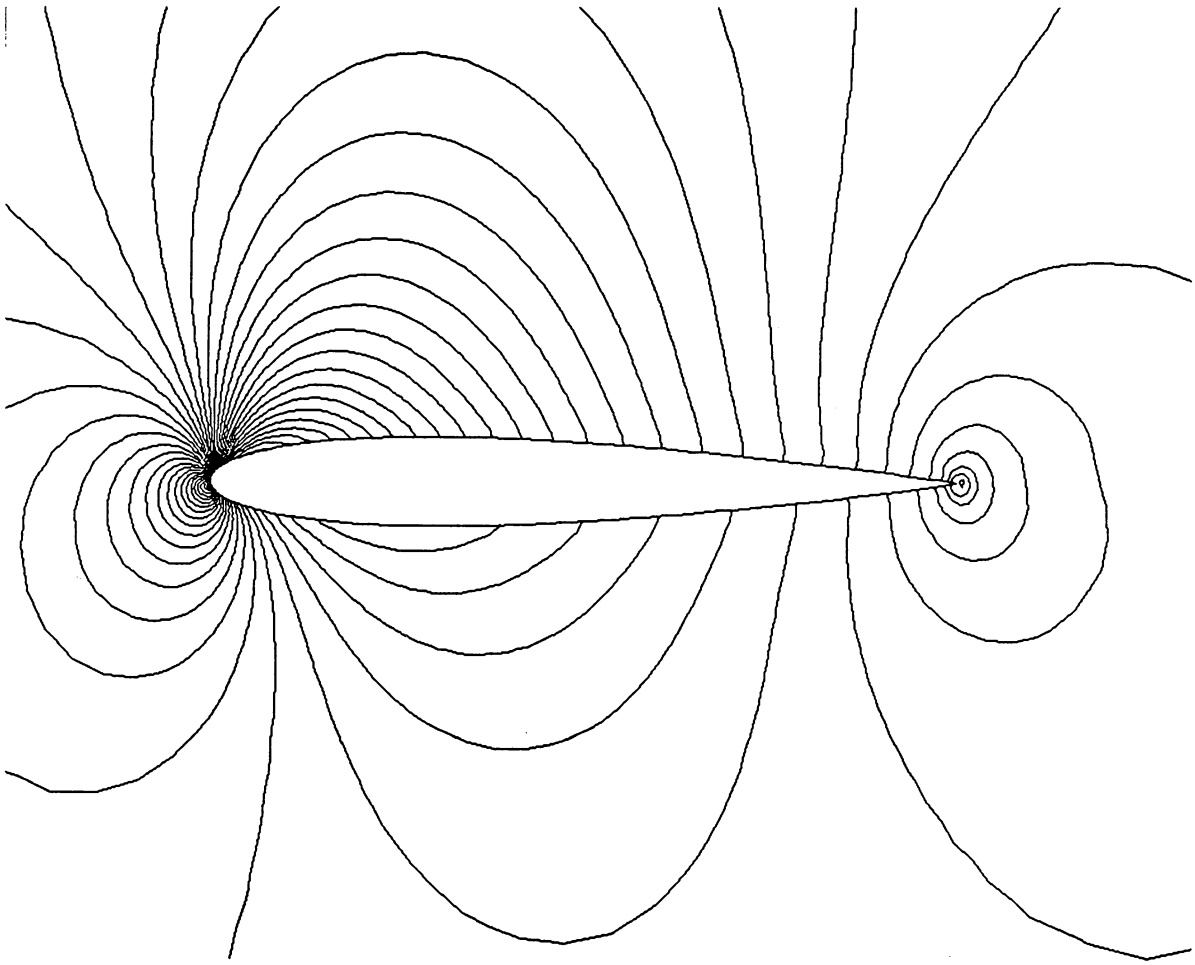

Figure 26. Density Contours - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$: $\sigma = -1$, $C_l = 0.33209$, $C_d = -0.00039$, 8381 elements, 4291 nodes, 201 bnodes, Equations solved at the wall.

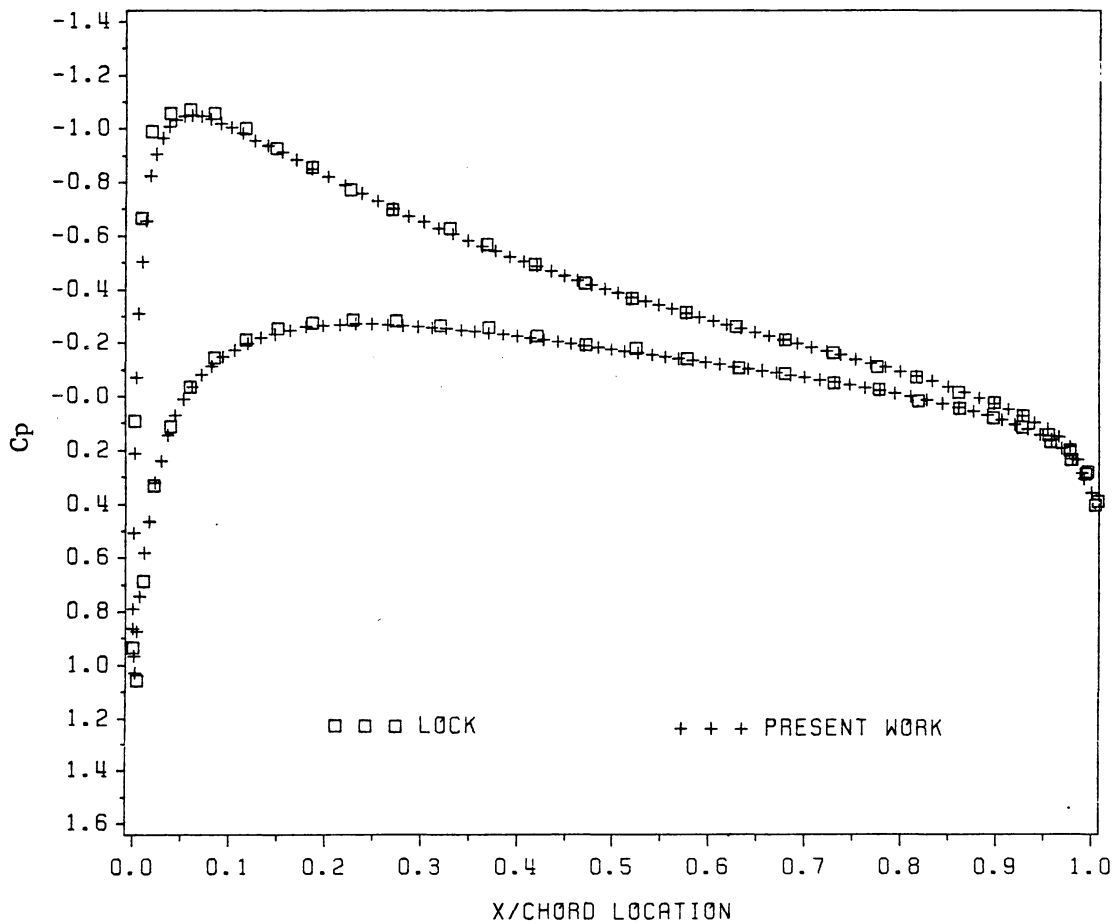

Figure 27. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.63$, $\alpha = 2°$: $\sigma = -1$, $C_l = 0.33209$, $C_d = -0.00039$, 8381 elements, 429 nodes, 201 bnodes, Equations solved at the wall versus Lock (Ref. 86).

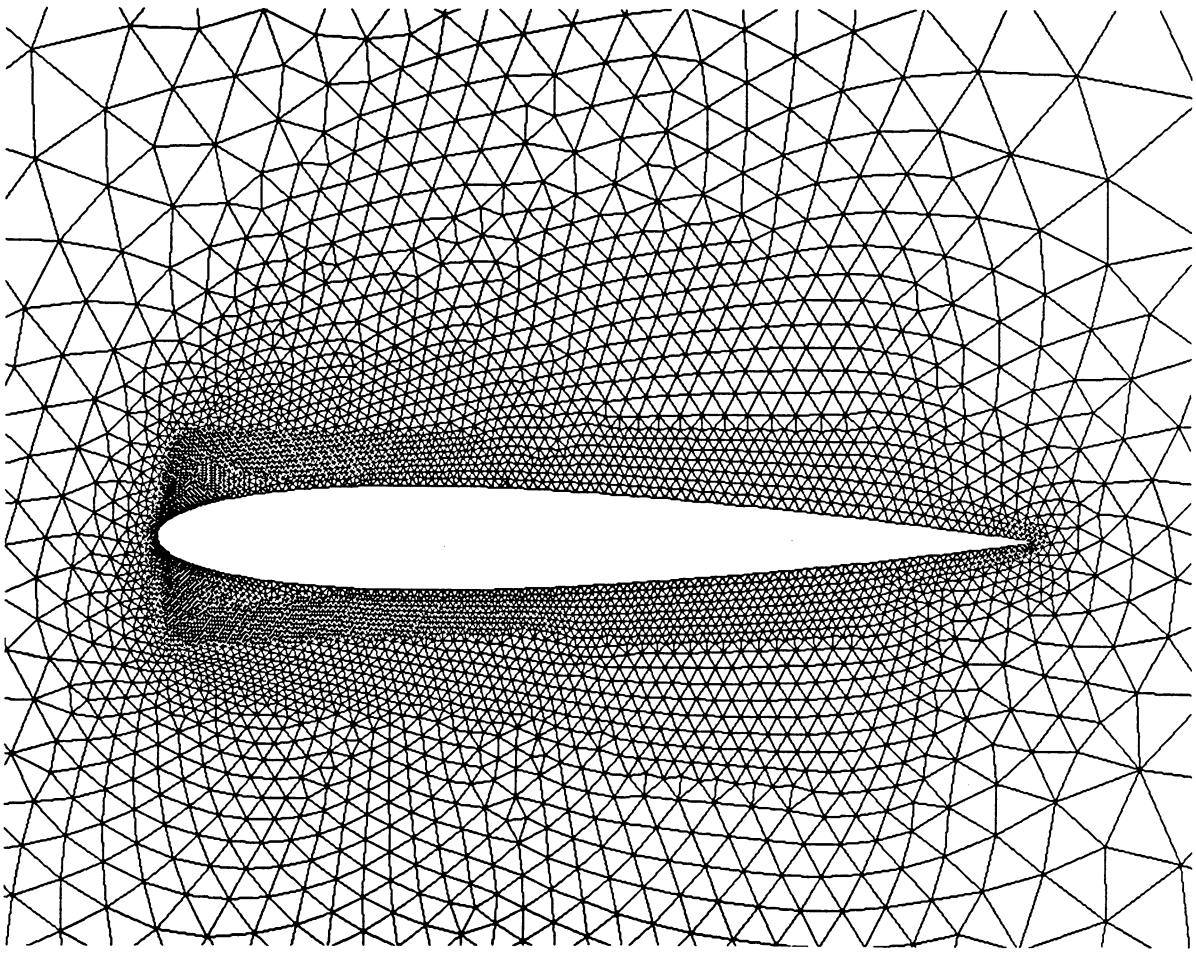

Figure 28. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: 10467 elements, 5374 nodes, 280 bnodes.

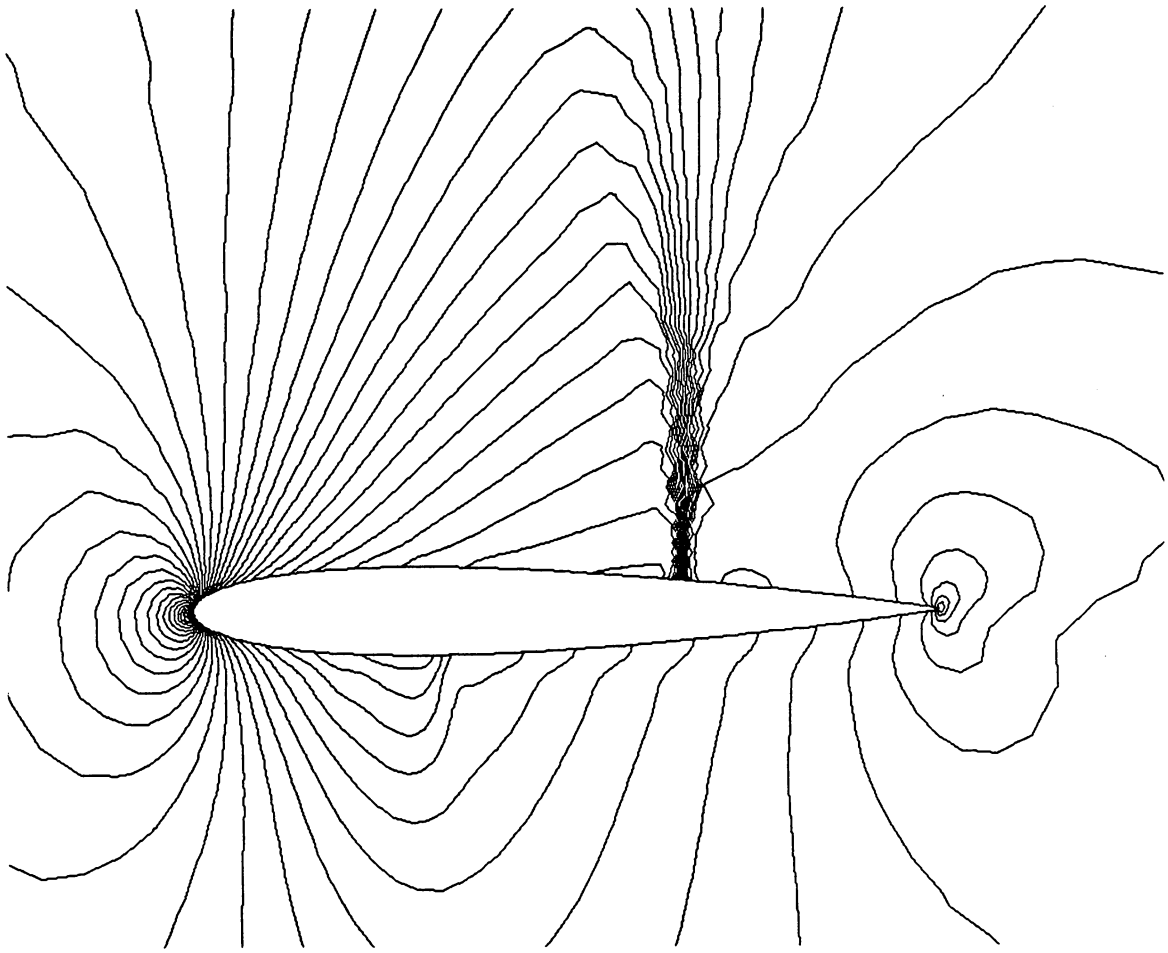

Figure 29. Mach Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = 1/3$, $C_l = 0.37243$, $C_d = 0.02358$, 10467 elements, 5374 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

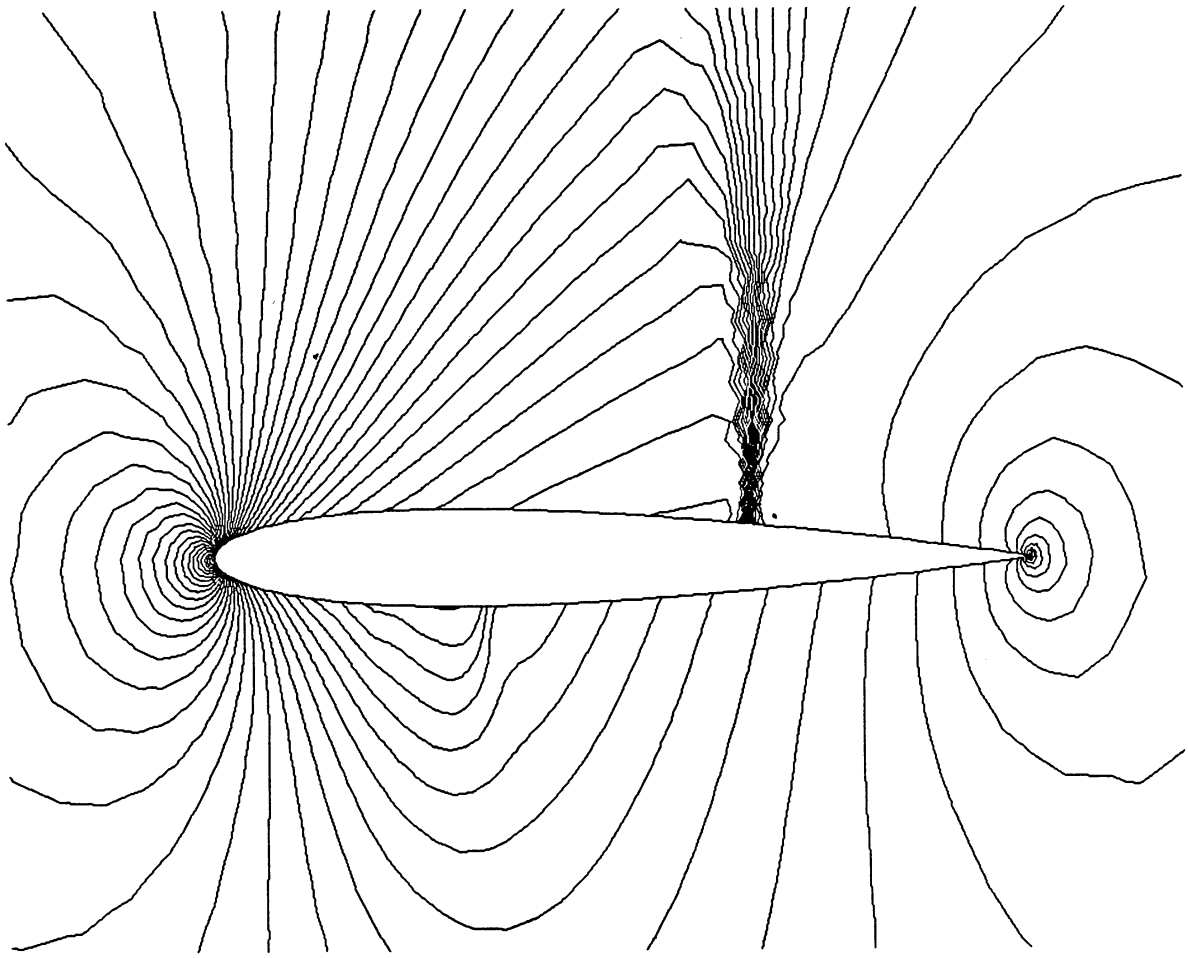

Figure 30. Pressure Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = 1/3$, $C_l = 0.37243$, $C_d = 0.02358$, 10467 elements, 5374 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

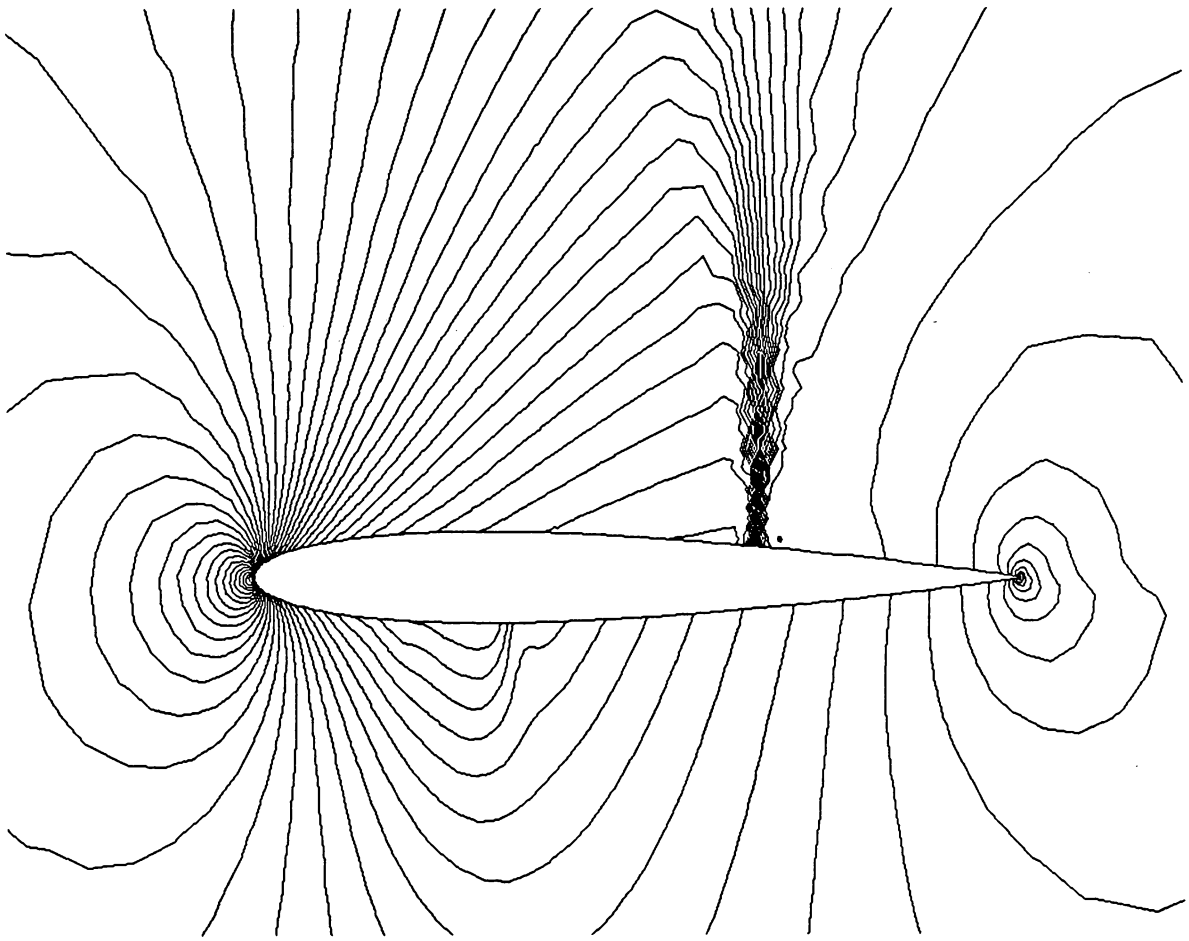

Figure 31. Density Contours - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = 1/3$, $C_l = 0.37243$, $C_d = 0.02358$, 10467 elements, 5374 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

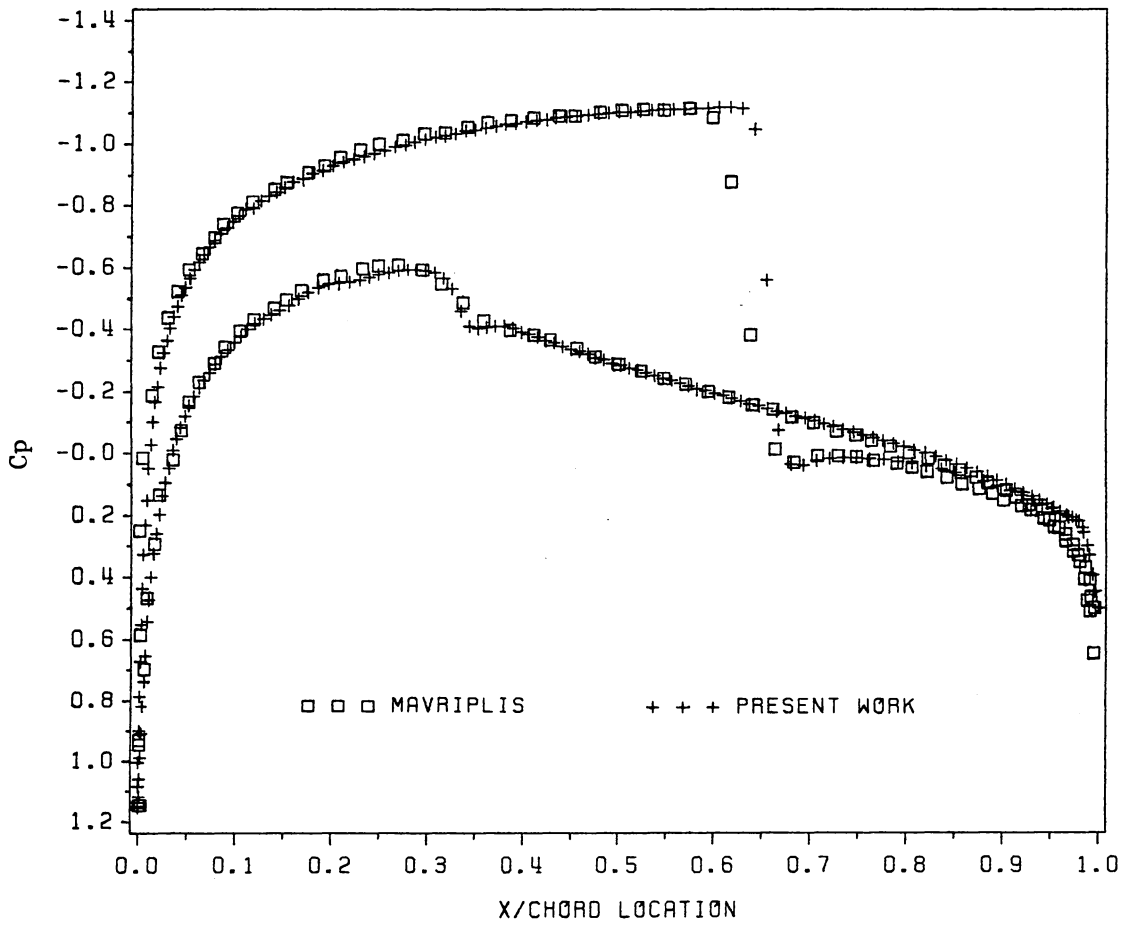

Figure 32. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = 1/3$, $C_l = 0.37243$, $C_d = 0.02358$, 10467 elements, 5374 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation versus Mavriplis (Ref. 80).

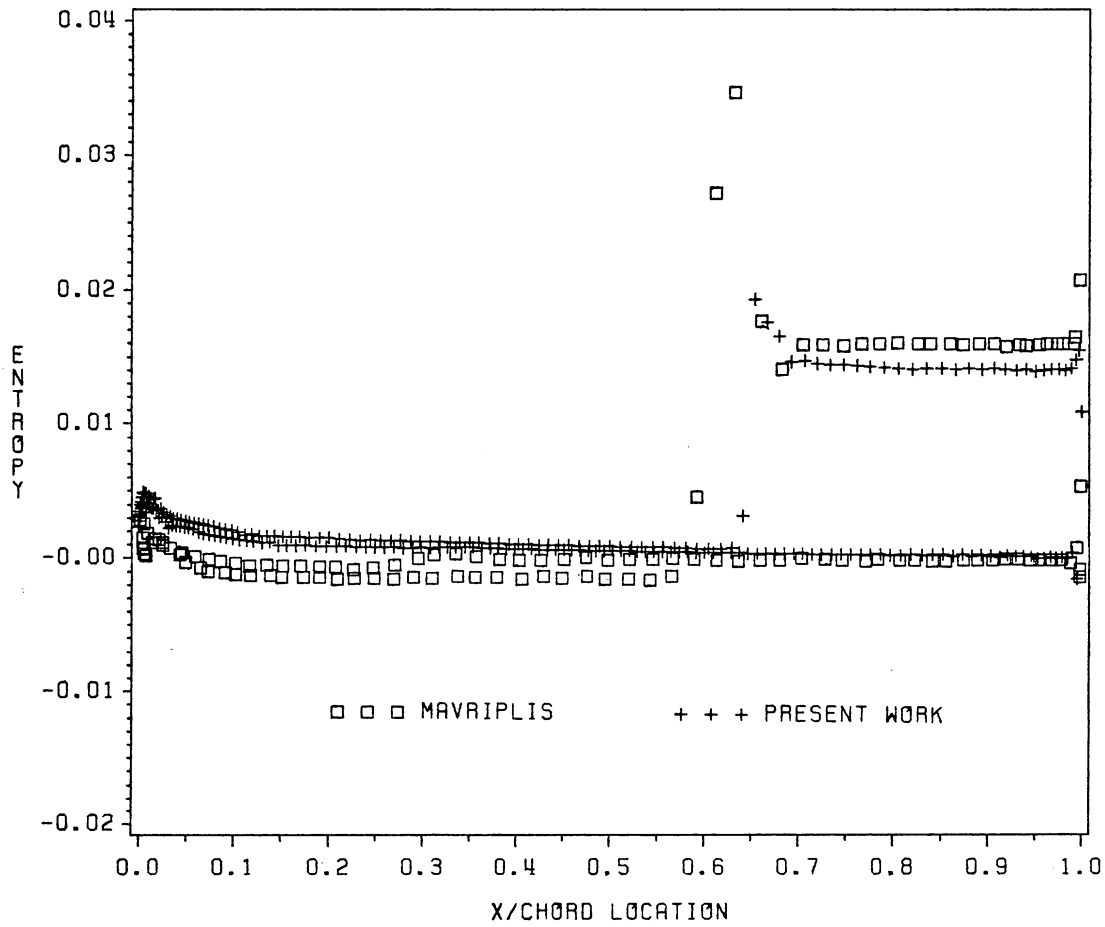

Figure 33. Surface Entropy Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = 1/3$, $C_l = 0.37243$, $C_d = 0.02358$, 10467 elements, 5374 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation versus Mavriplis (Ref. 80).

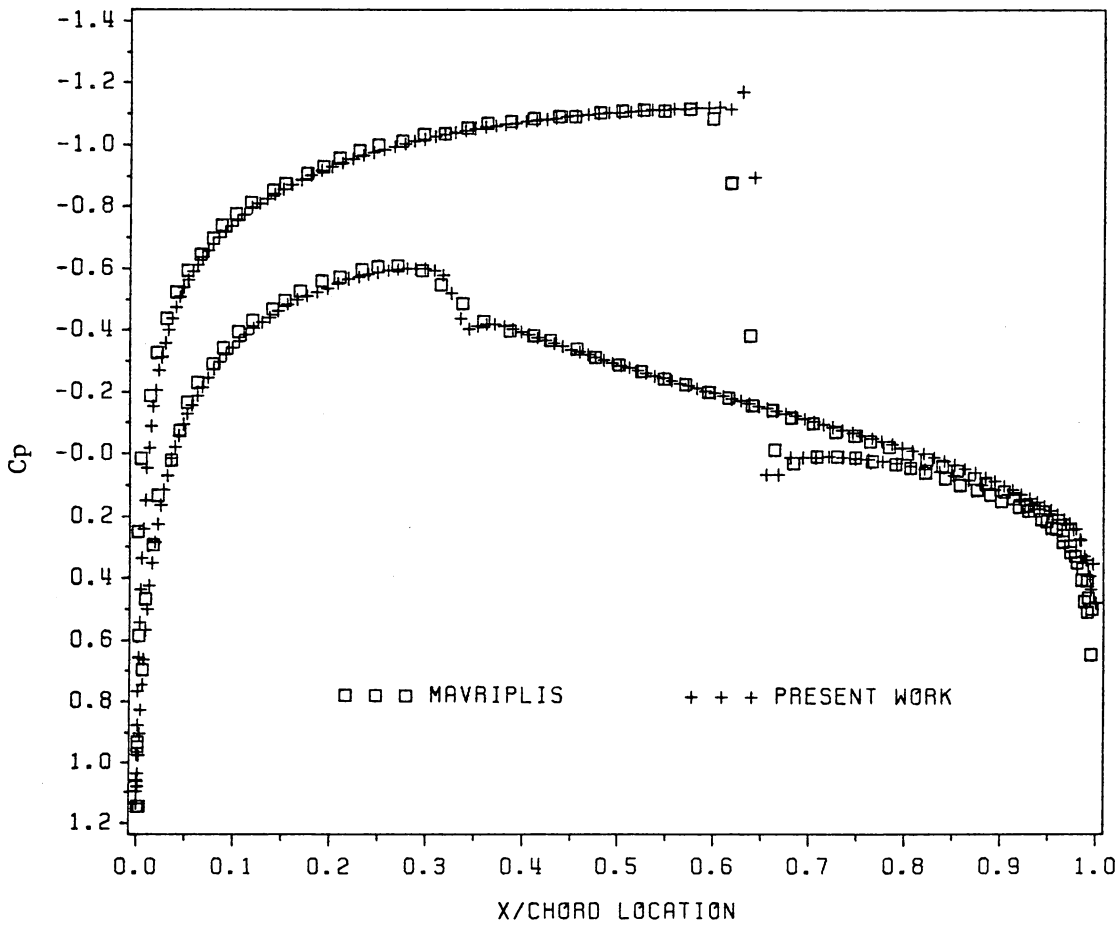

Figure 34. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = -1$, $C_l = 0.36299$, $C_d = 0.02276$, no limiting, 10467 elements, 5374 nodes, 280 bnodes, Equations solved at the wall versus Mavriplis (Ref. 80).

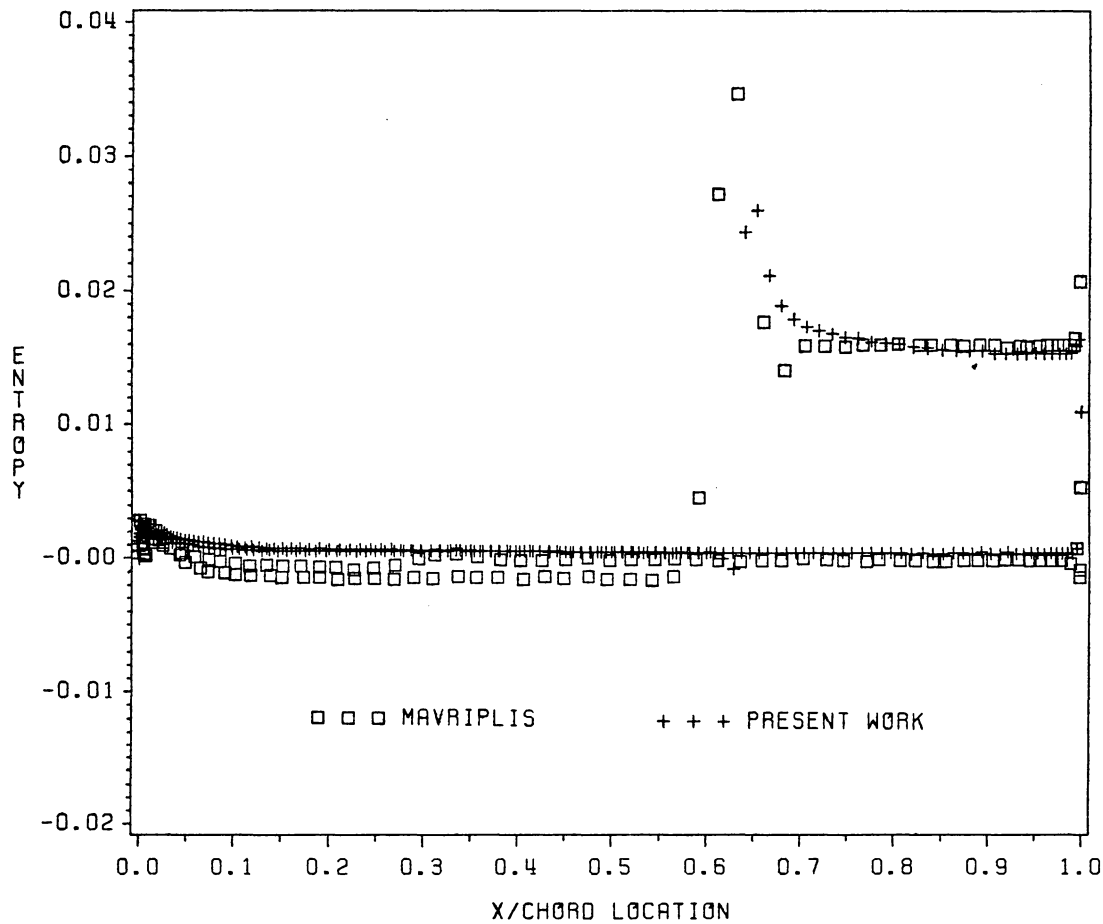

Figure 35. Surface Entropy Comparison - NACA 0012, $M_\infty = 0.80$, $\alpha = 1.25°$: $\sigma = -1$, $C_l = 0.36299$, $C_d = 0.02276$, no limiting, 10467 elements, 5374 nodes, 280 bnodes, Equations solved at the wall versus Mavriplis (Ref. 80).

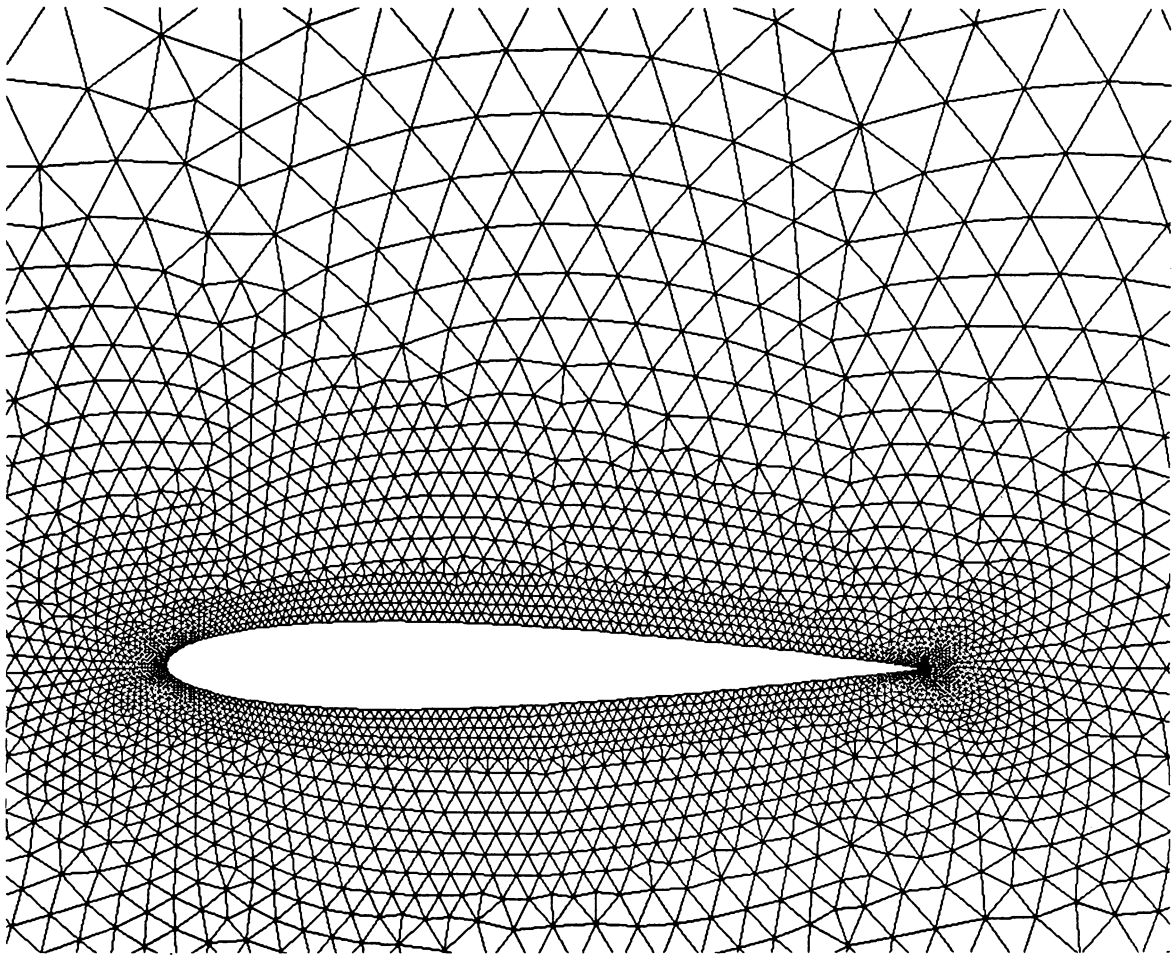

Figure 36. Close-up of Unstructured Mesh - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: 10122 elements, 5201 nodes, 280 bnodes.

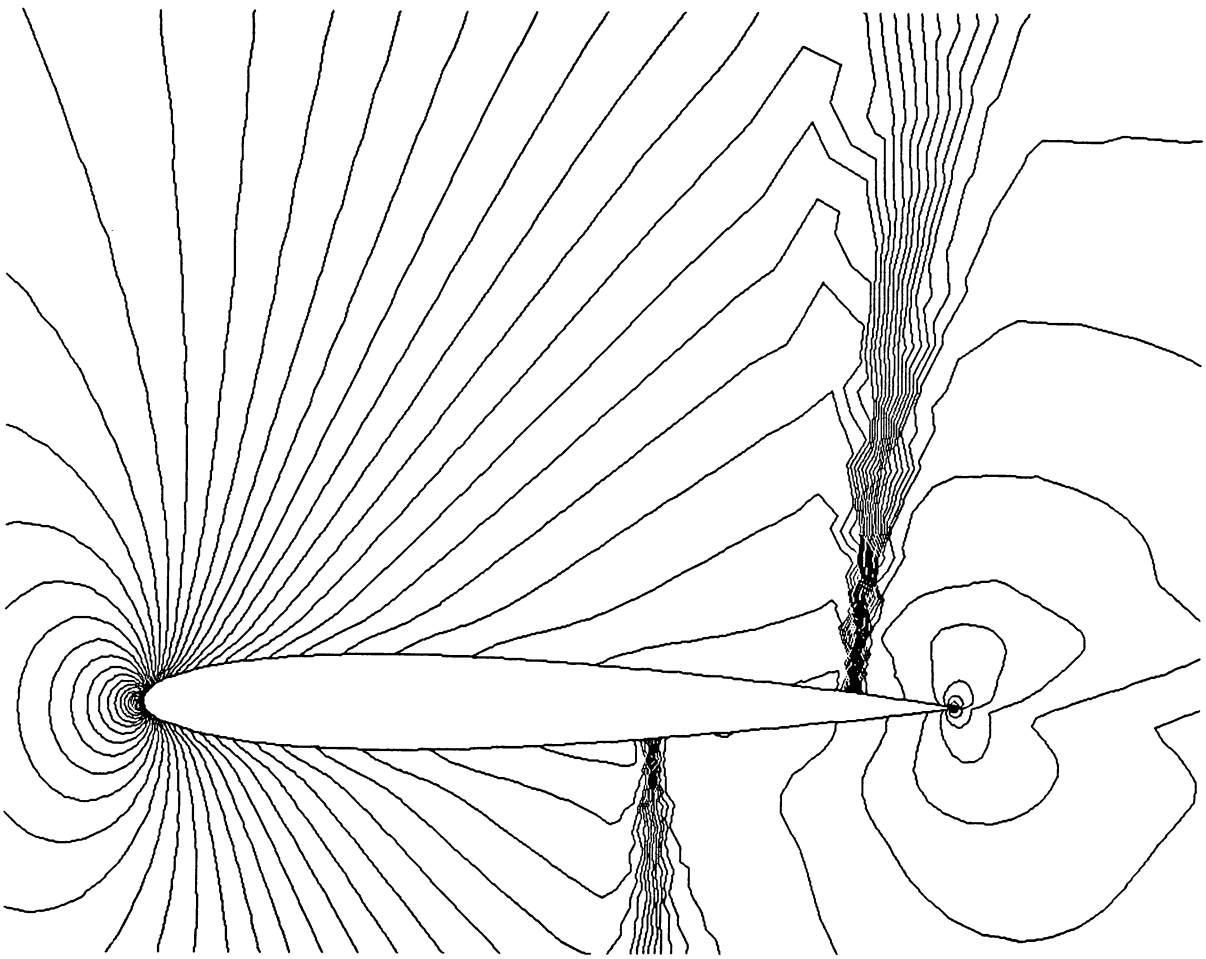

Figure 37. Mach Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.38848$, $C_d = 0.05614$, 10122 elements, 5201 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

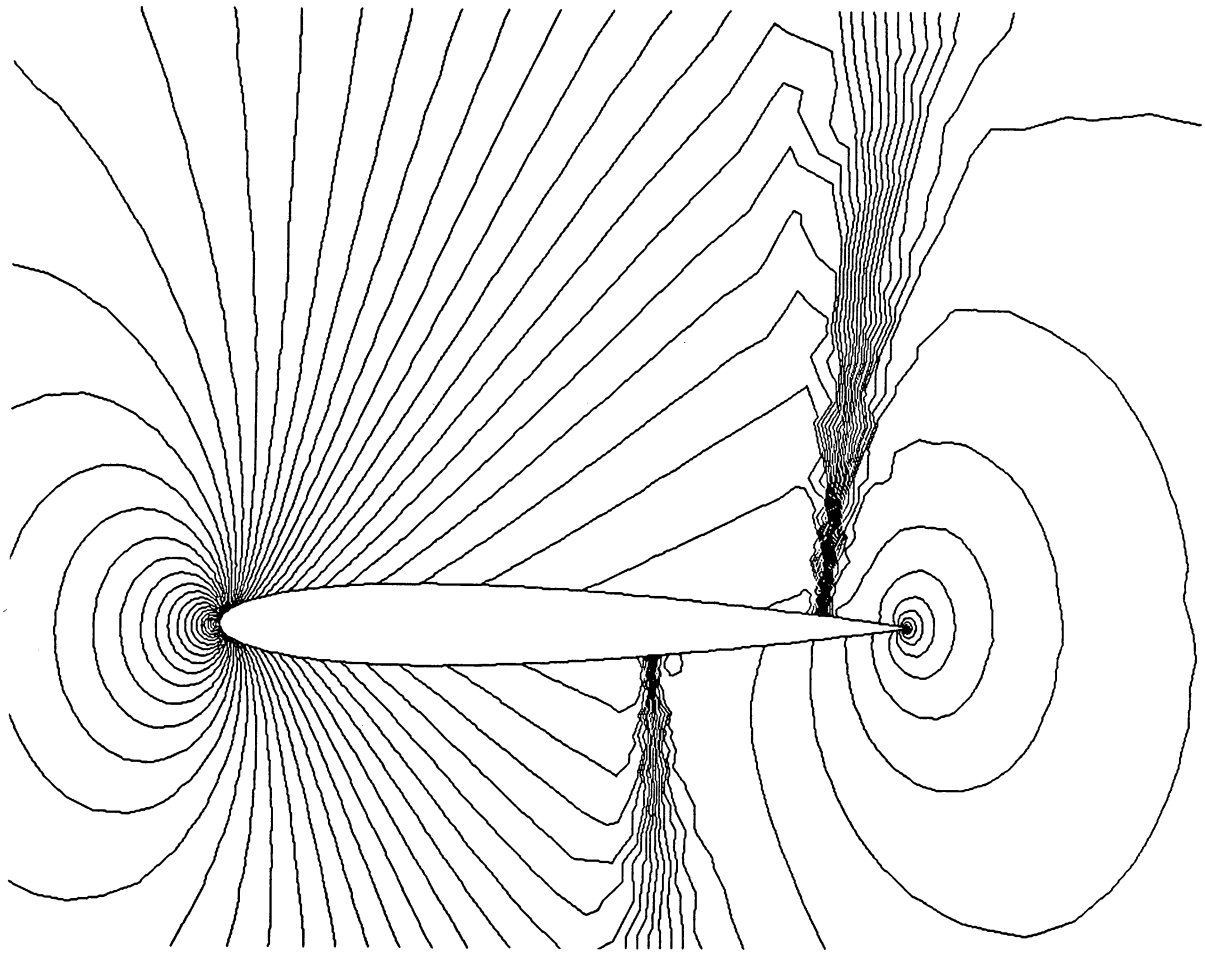

Figure 38. Pressure Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.38848$, $C_d = 0.05614$, 10122 elements, 5201 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

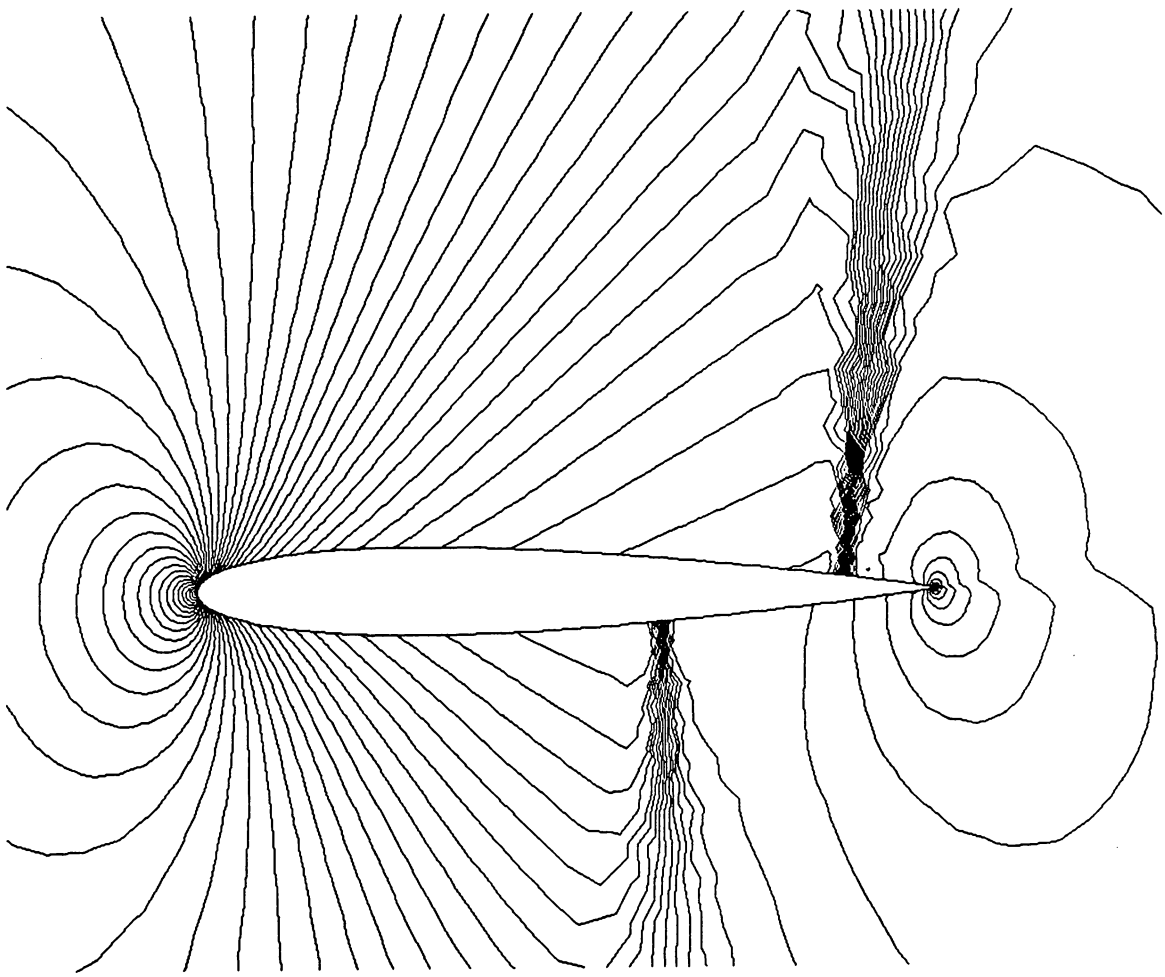

Figure 39. Density Contours - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.38848$, $C_d = 0.05614$, 10122 elements, 5201 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation.

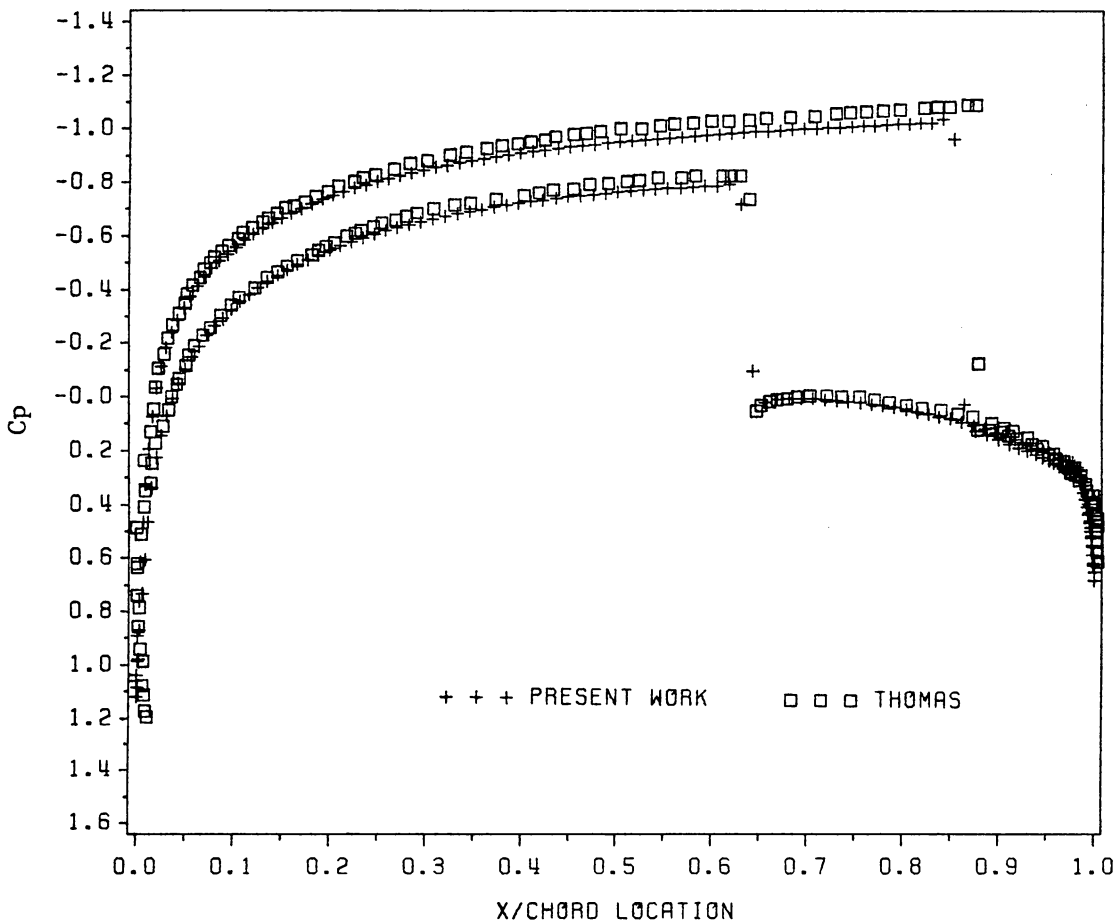

Figure 40. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.3534$, $C_d = 0.05727$, 10122 elements, 5201 nodes, 280 bnodes, Equations solved at the wall versus Thomas (Ref. 87).

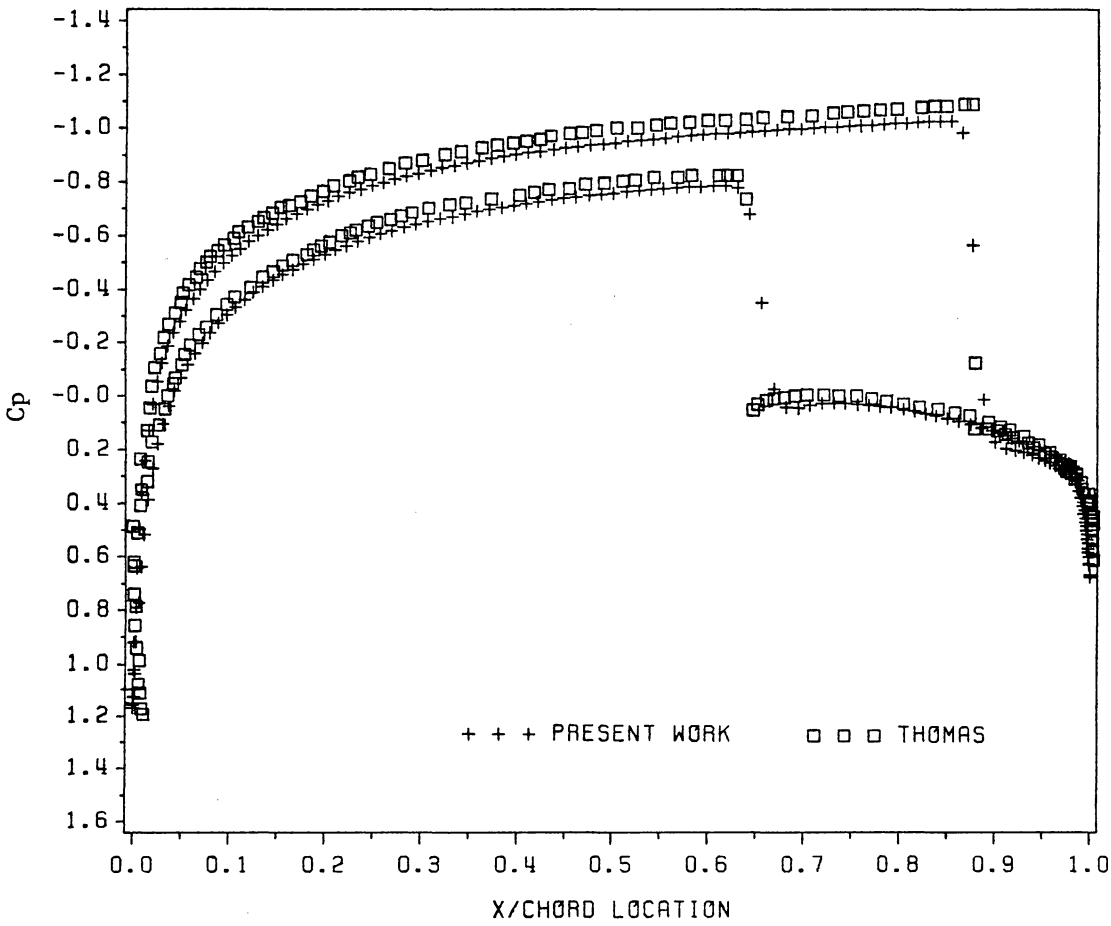

Figure 41. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.3611$, $C_d = 0.06556$, 10122 elements, 5201 nodes, 280 bnodes, Normal extrapolation versus Thomas (Ref. 87).

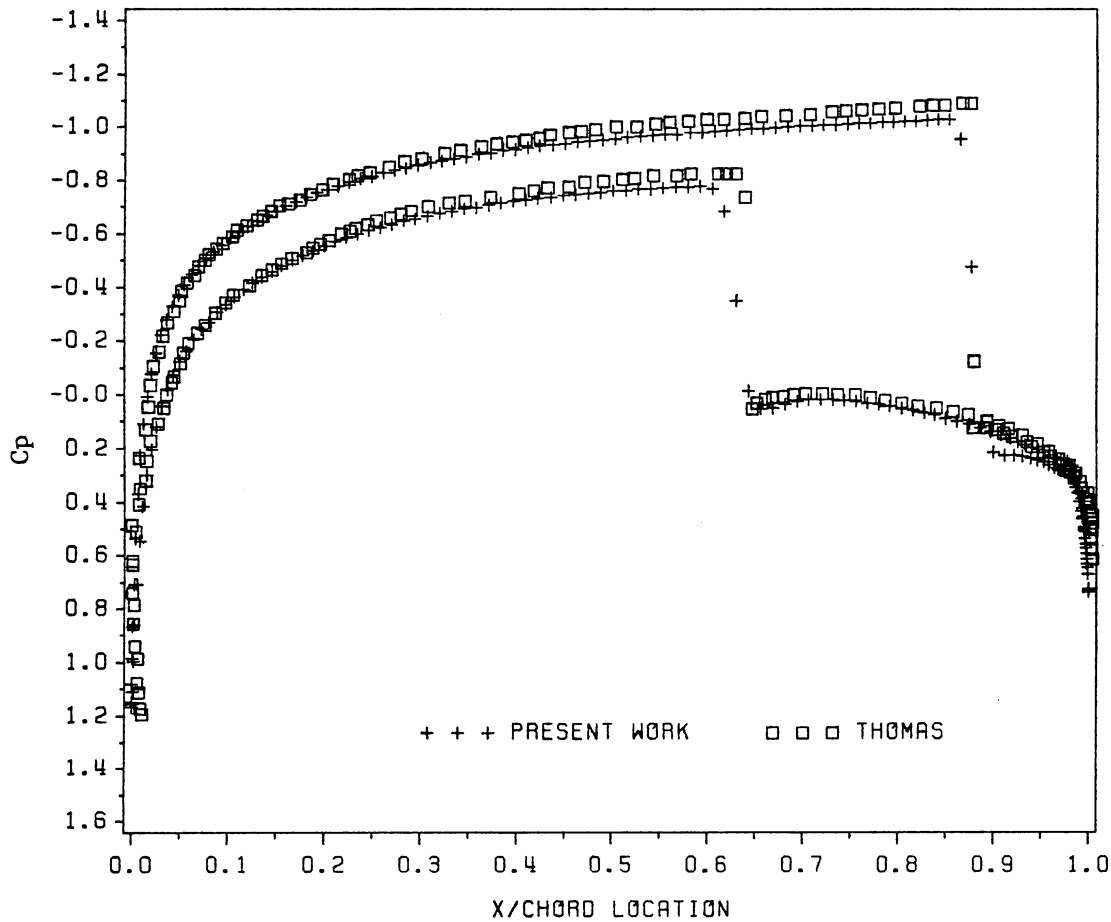

Figure 42. Surface Pressure Comparison - NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$: $\sigma = 1/3$, $C_l = 0.38848$, $C_d = 0.05614$, 10122 elements, 5201 nodes, 280 bnodes, Normal extrapolation with wall pressure correction using the normal momentum equation versus Thomas (Ref. 87).

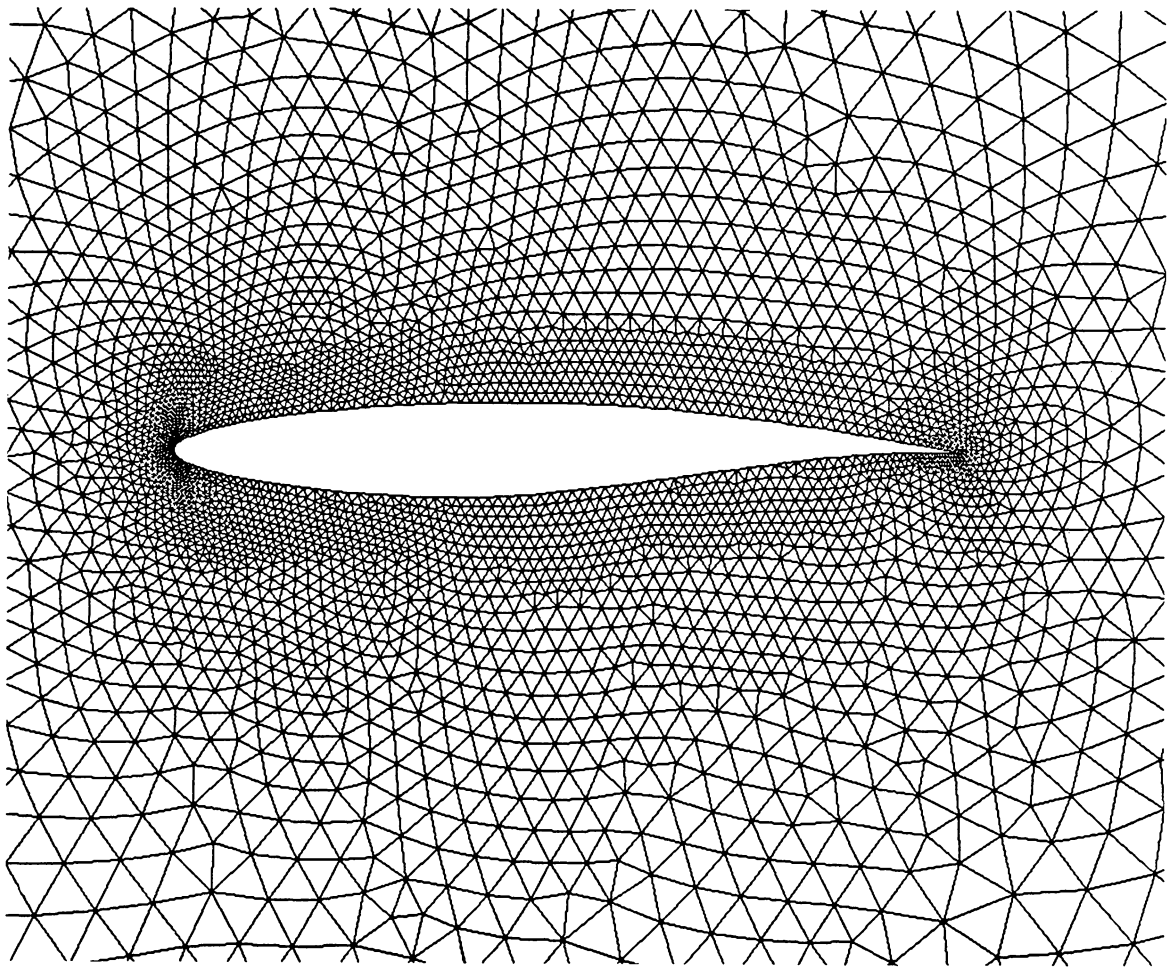

Figure 43. Close-up of Unstructured Mesh - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$: 8898 elements, 4564 nodes, 230 bnodes.

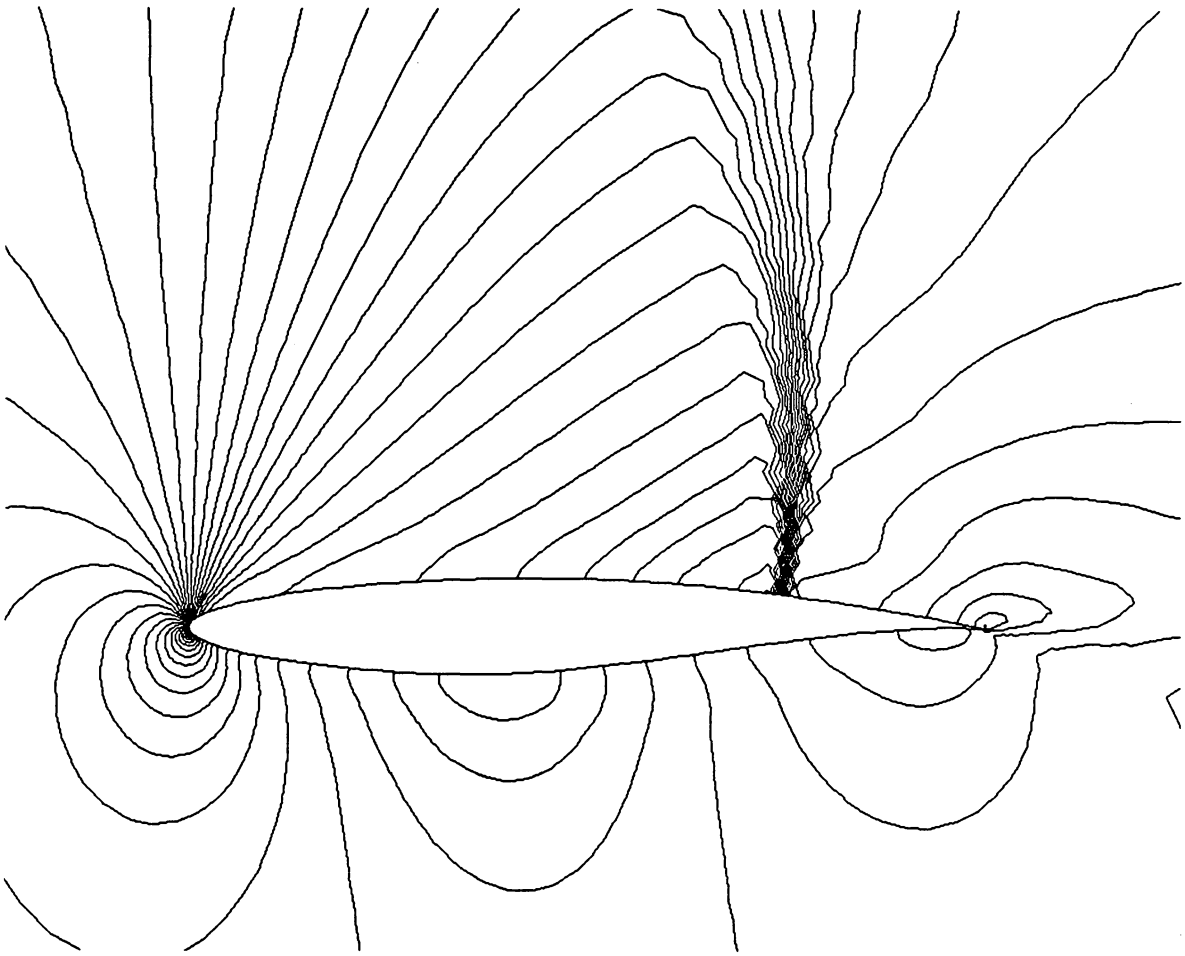

Figure 44. Mach Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$: $\sigma = 1/3$, $C_l = 1.09104$, $C_d = 0.04491$, 8898 elements, 4564 nodes, 230 bnodes , Equations solved at the wall.

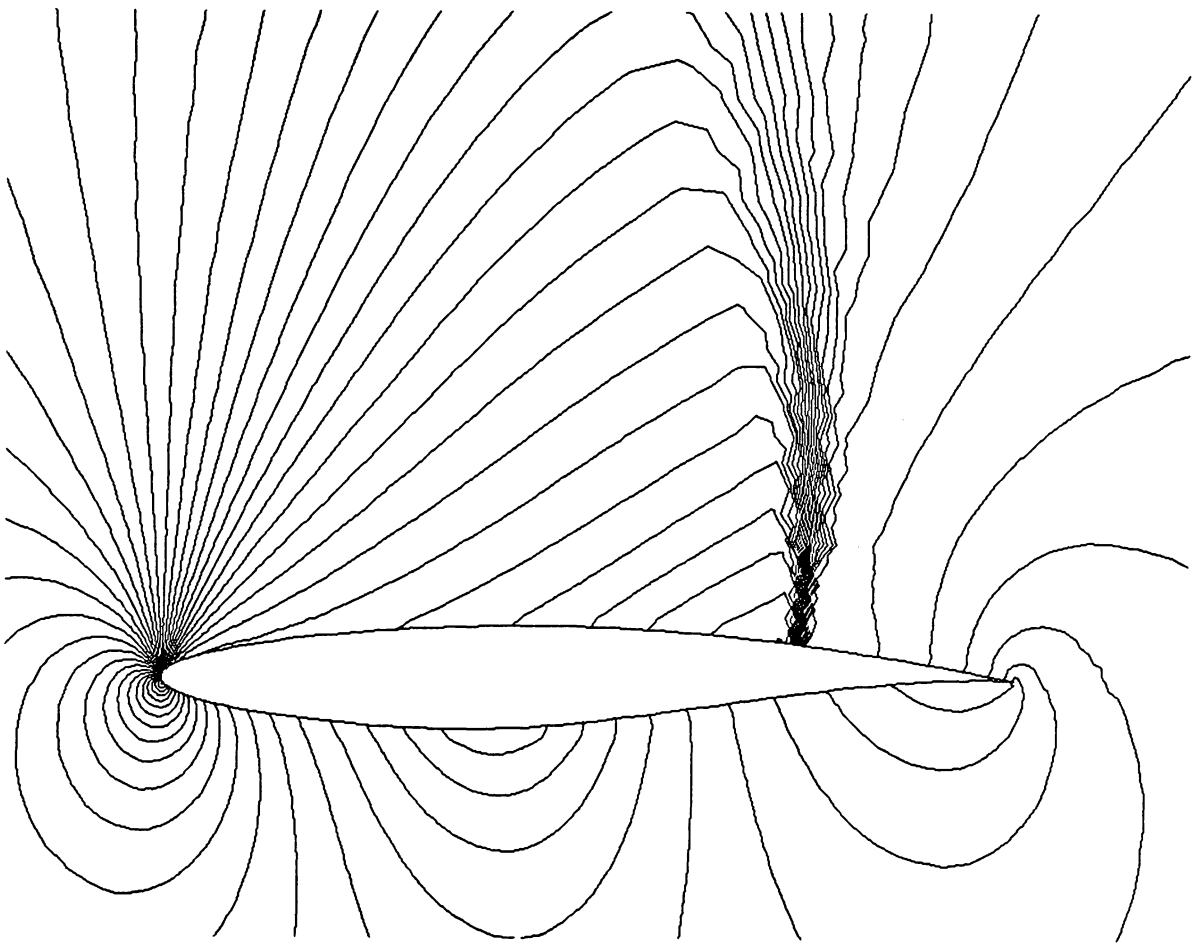

Figure 45. Pressure Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$: $\sigma = 1/3$, $C_l = 1.09104$, $C_d = 0.04491$, 8898 elements, 4564 nodes, 230 bnodes, Equations solved at the wall.

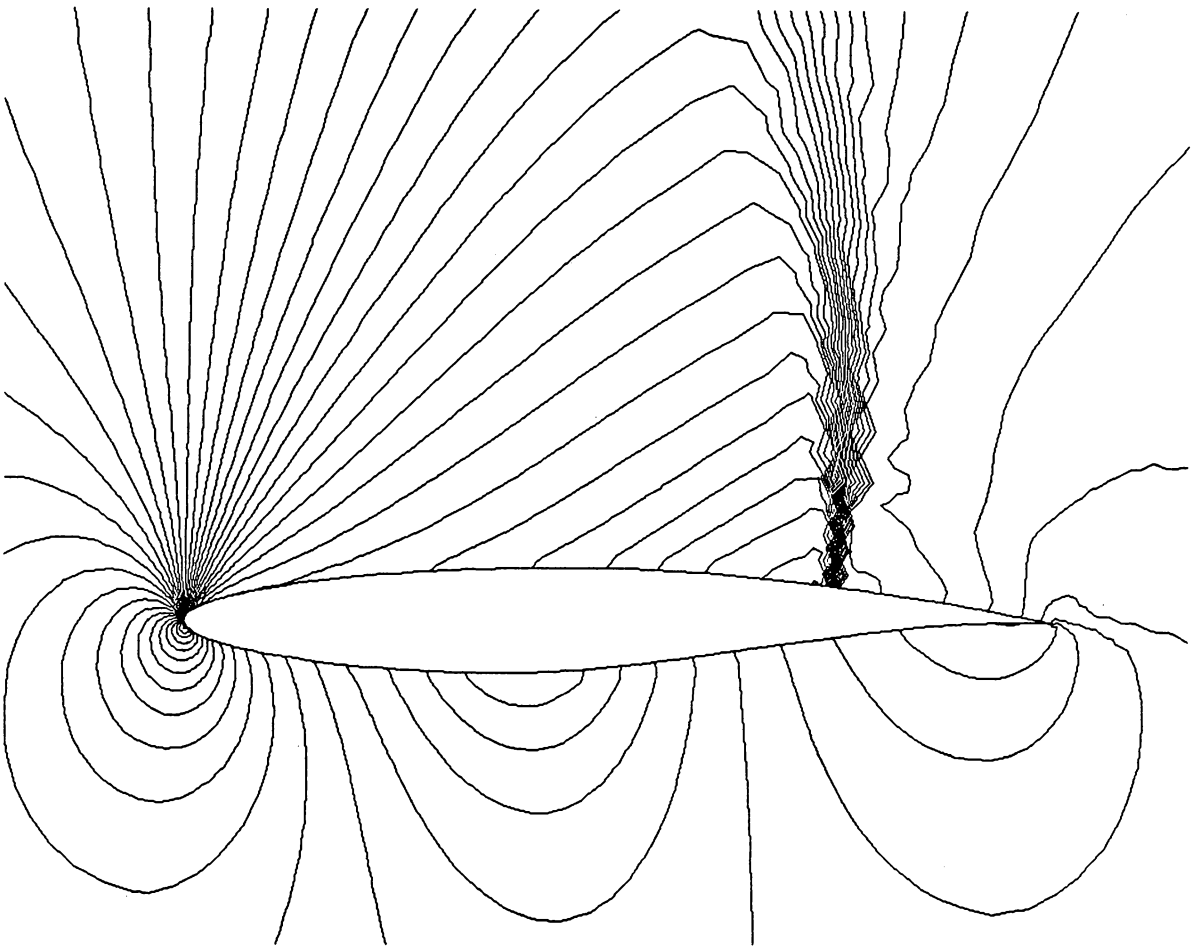

Figure 46. Density Contours - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$: $\sigma = 1/3$, $C_l = 1.09104$, $C_d = 0.04491$, 8898 elements, 4564 nodes, 230 bnodes, Equations solved at the wall.

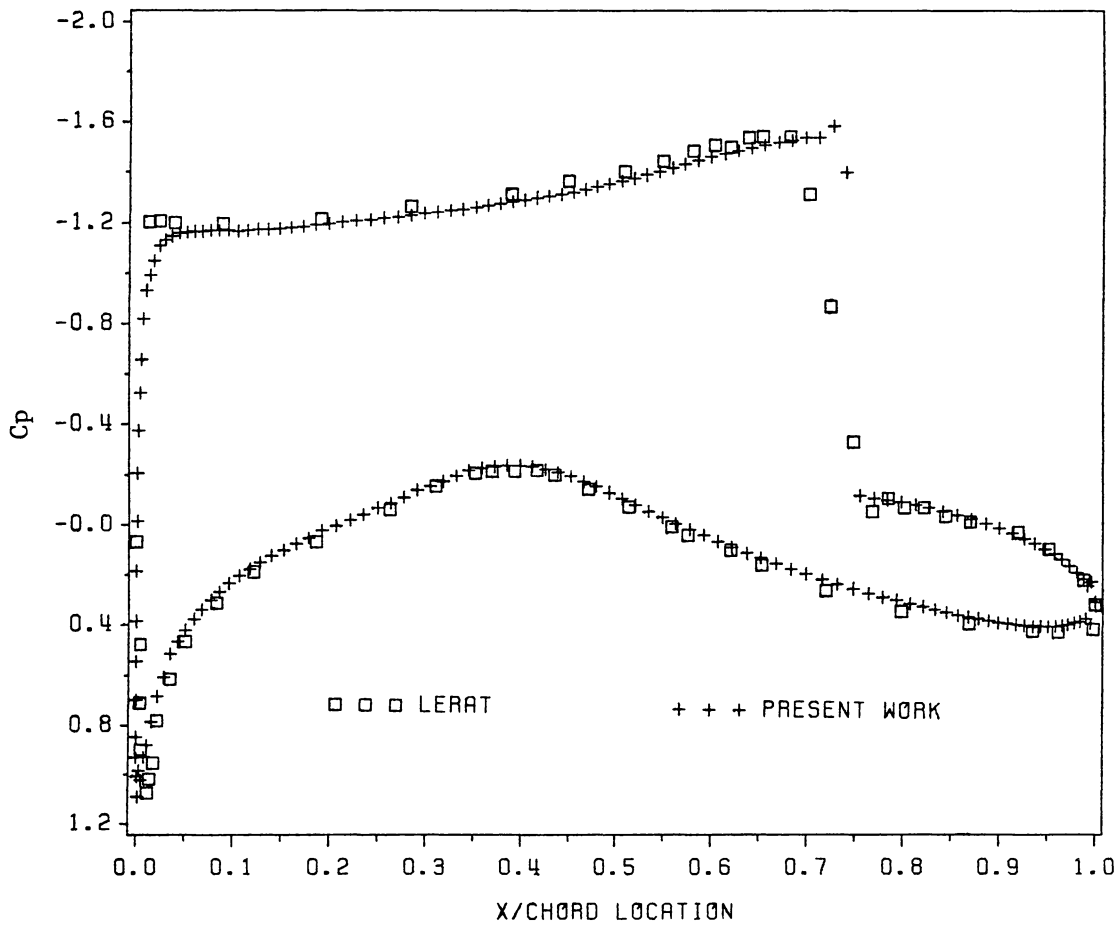

Figure 47. Surface Pressure Comparison - RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$: $\sigma = 1/3$, $C_l = 1.09104$, $C_d = 0.04491$, 8898 elements, 4564 nodes, 230 bnodes, Equations solved at the wall versus Lerat (Ref. 88).

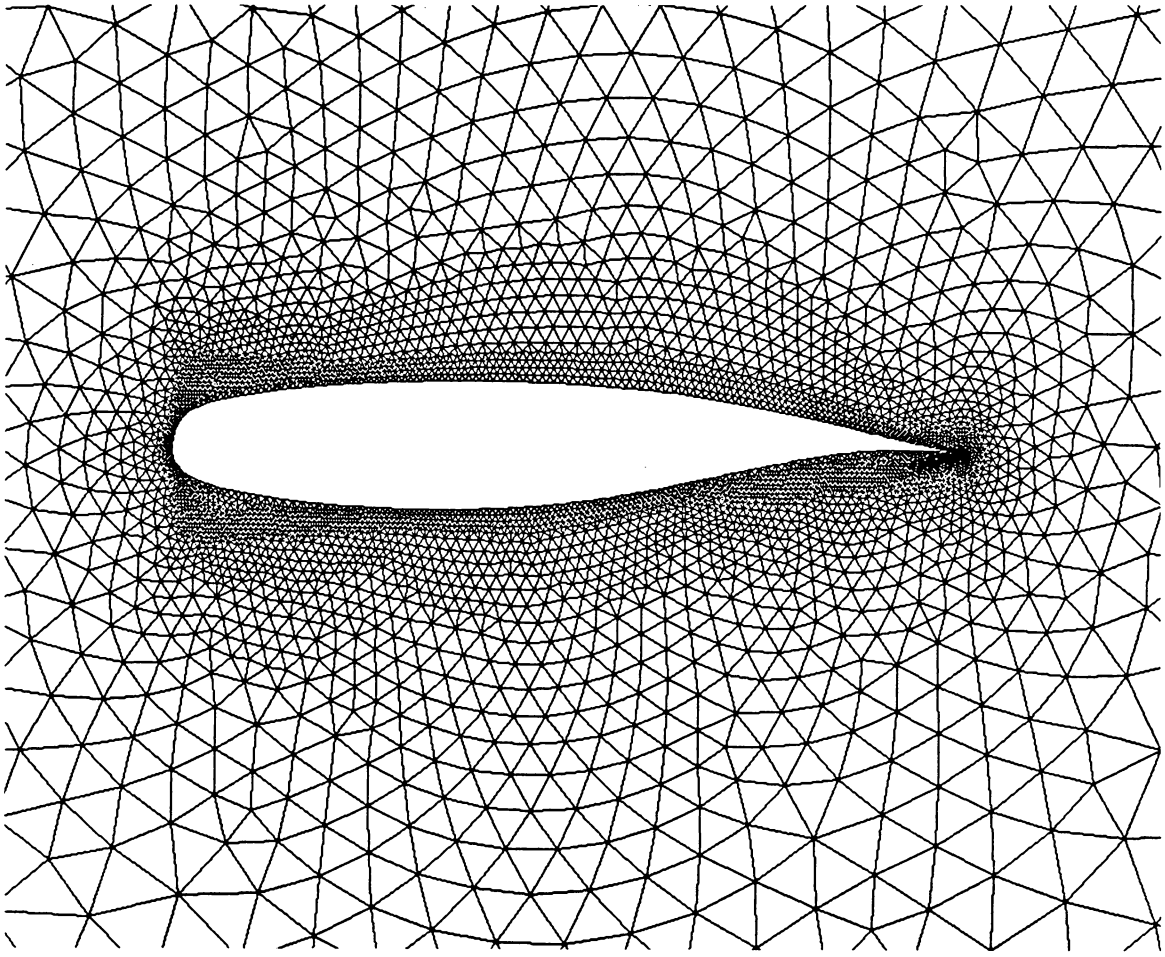

Figure 48. Close-up of Unstructured Mesh - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$: 9906 elements, 5135 nodes, 364 bnodes.

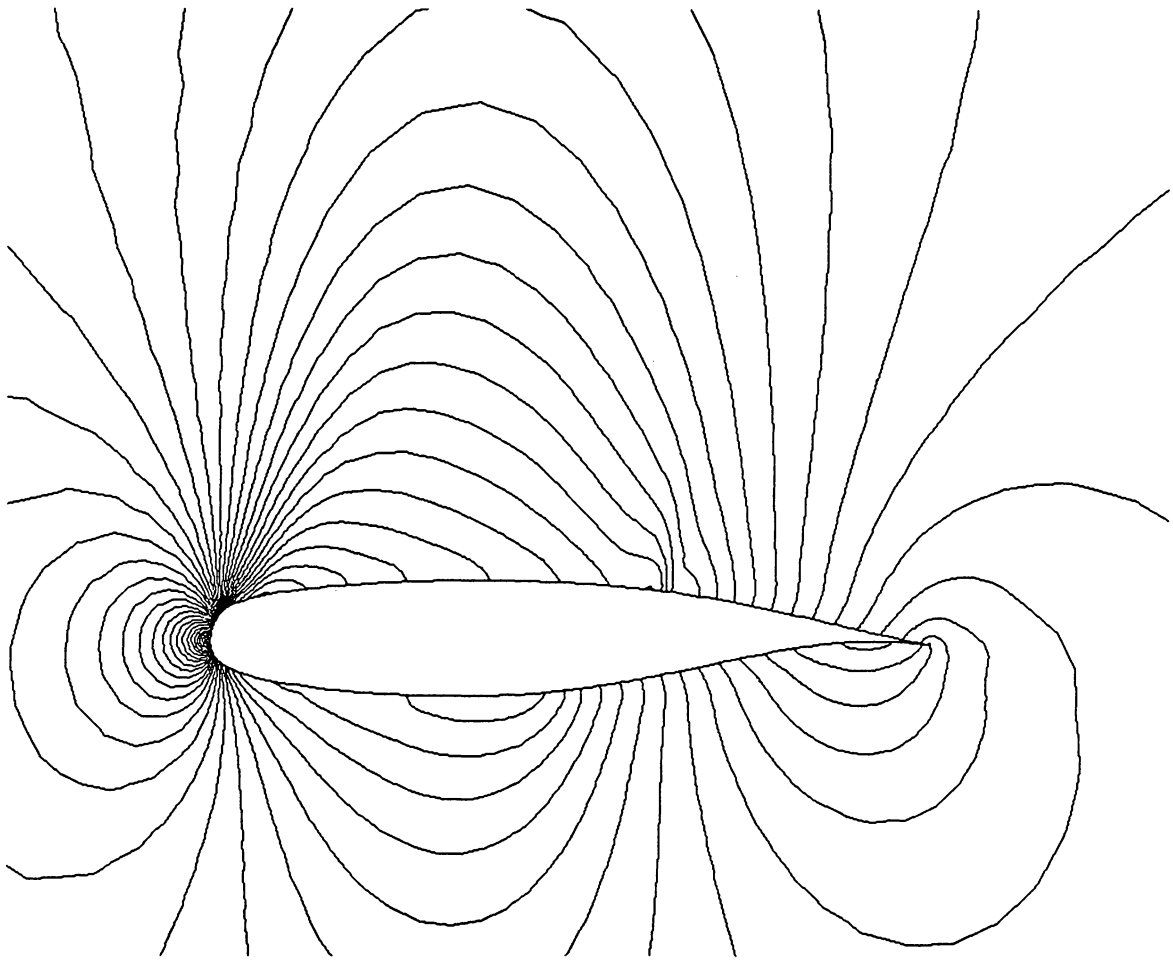

Figure 49. Mach Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$: $\sigma = 1/3$, $C_l = 0.60494$, $C_d = 0.00007$, 9906 elements, 5135 nodes, 364 bnodes, Equations solved at the wall.

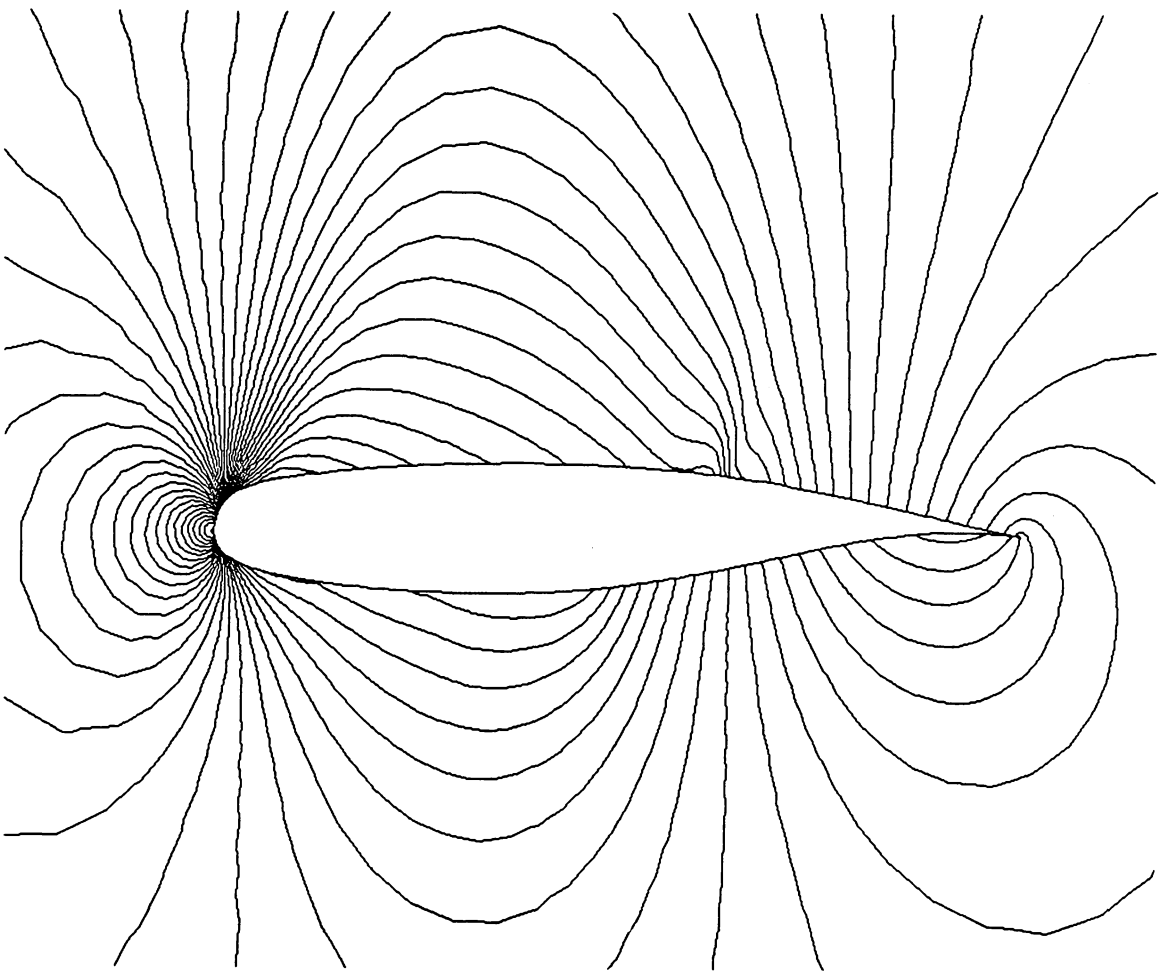

Figure 50. Pressure Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$: $\sigma = 1/3$, $C_l = 0.60494$, $C_d = 0.00007$, 9906 elements, 5135 nodes, 364 bnodes, Equations solved at the wall.

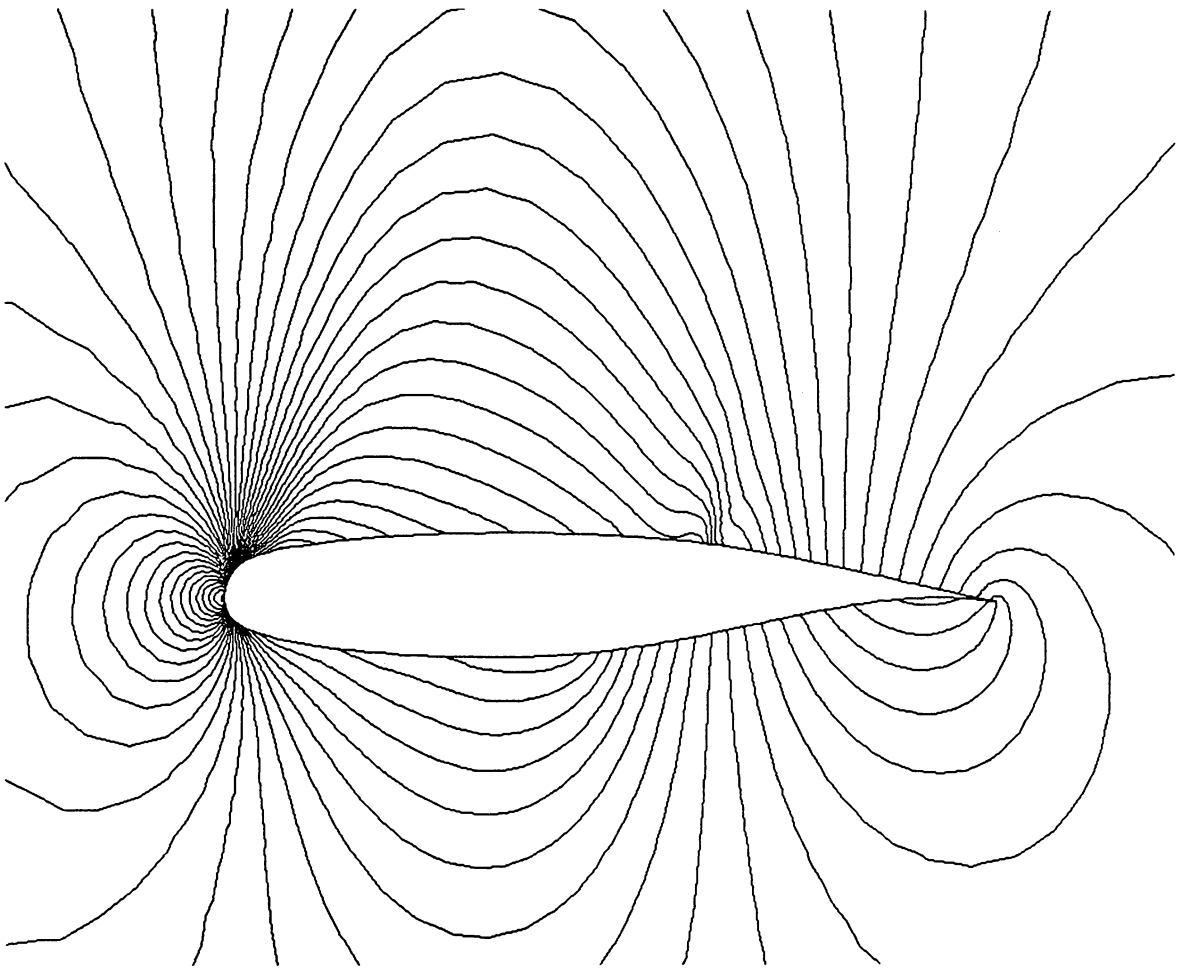

Figure 51. Density Contours - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$: $\sigma = 1/3$, $C_l = 0.60494$, $C_d = 0.00007$, 9906 elements, 5135 nodes, 364 bnodes, Equations solved at the wall.

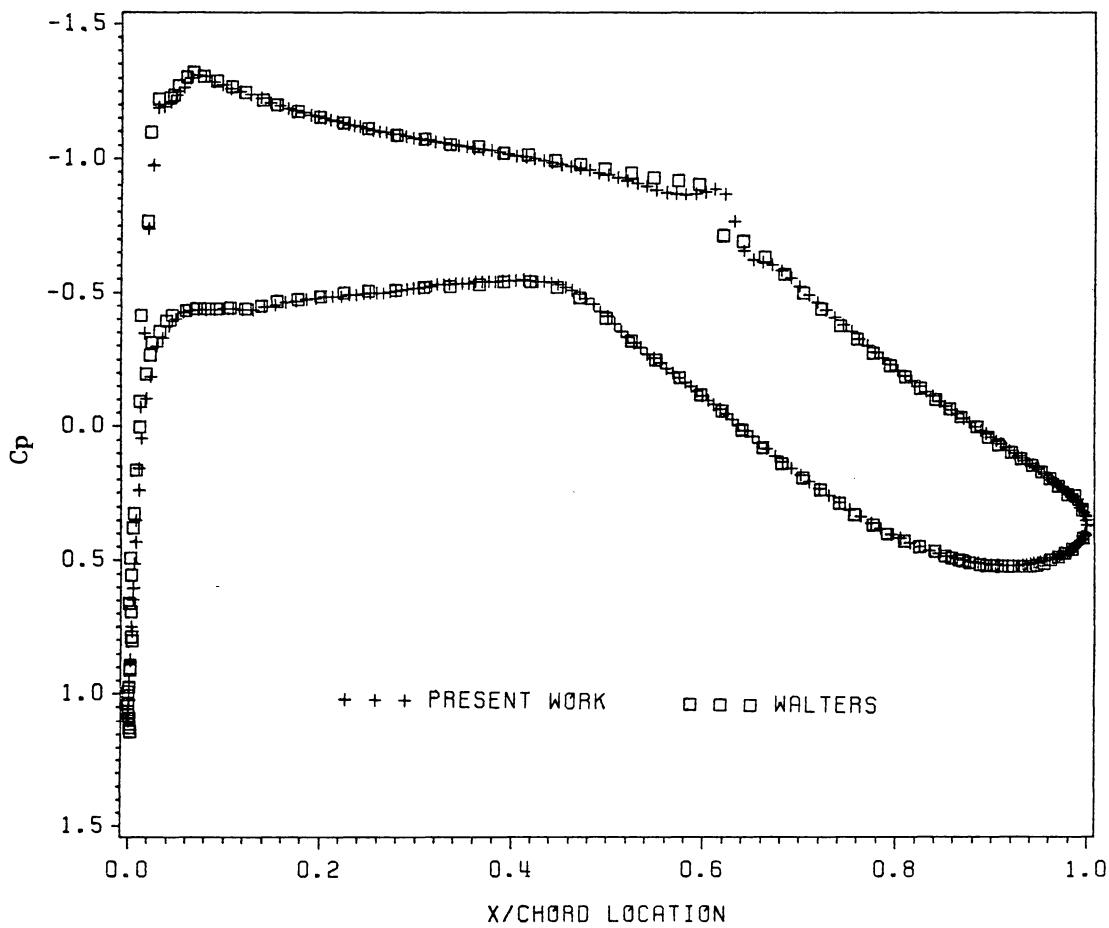

Figure 52. Surface Pressure Comparison - NLR 7301, $M_\infty = 0.720957$, $\alpha = -0.194°$: $\sigma = 1/3$, $C_l = 0.60494$, $C_d = 0.00007$, 9906 elements, 5135 nodes, 364 bnodes, Equations solved at the wall versus Walters (Ref. 89).

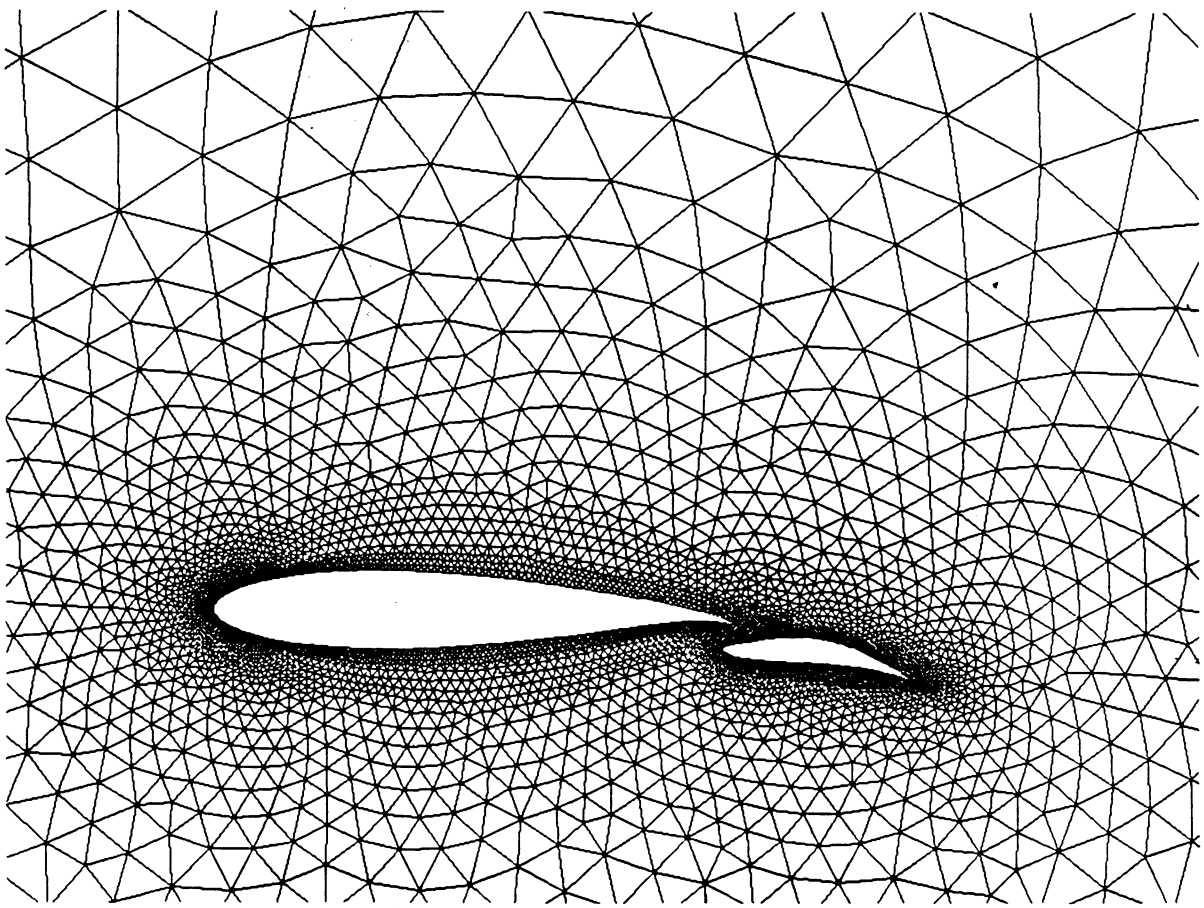

Figure 53. Close-up of Unstructured Mesh - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$: 14587 elements, 7566 nodes.

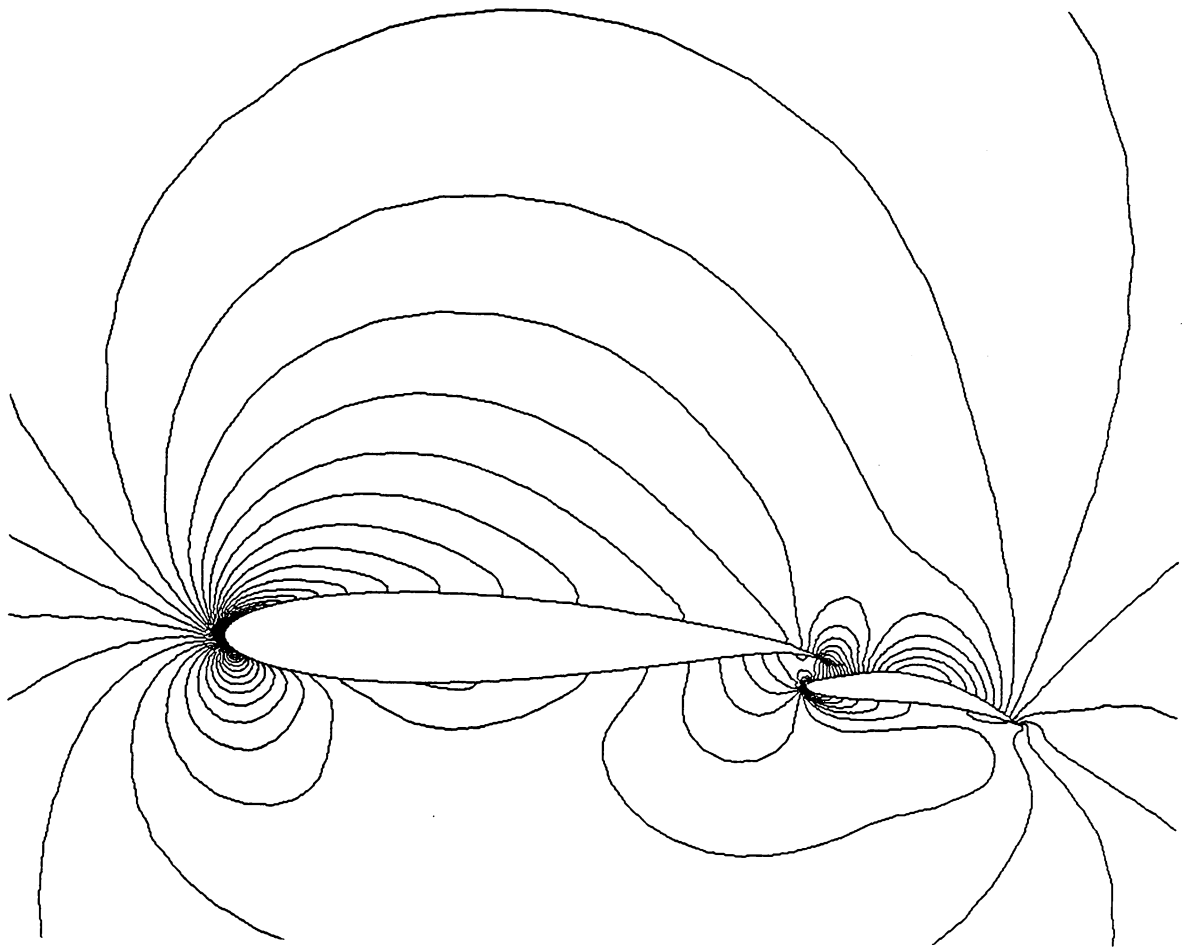

Figure 54. Mach Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

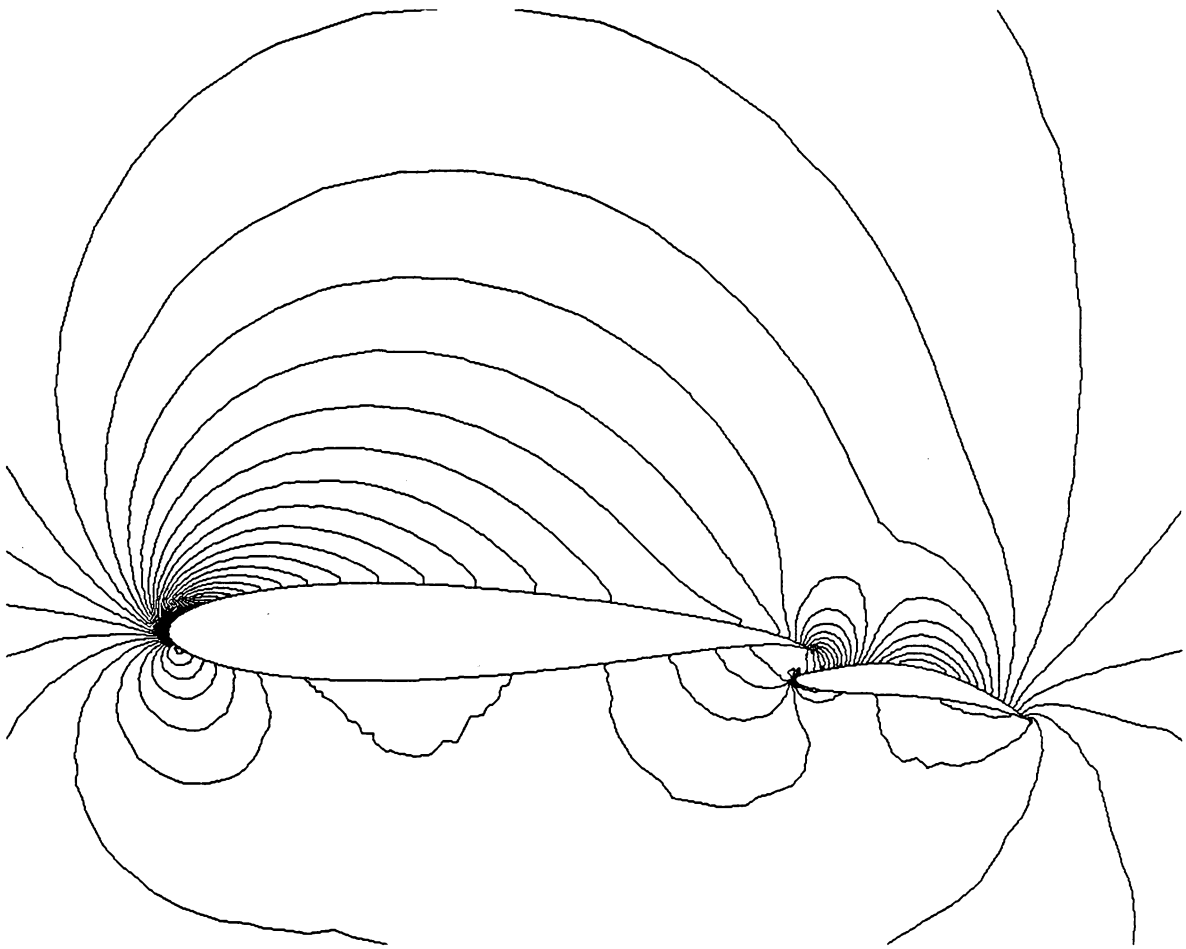

Figure 55. Pressure Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

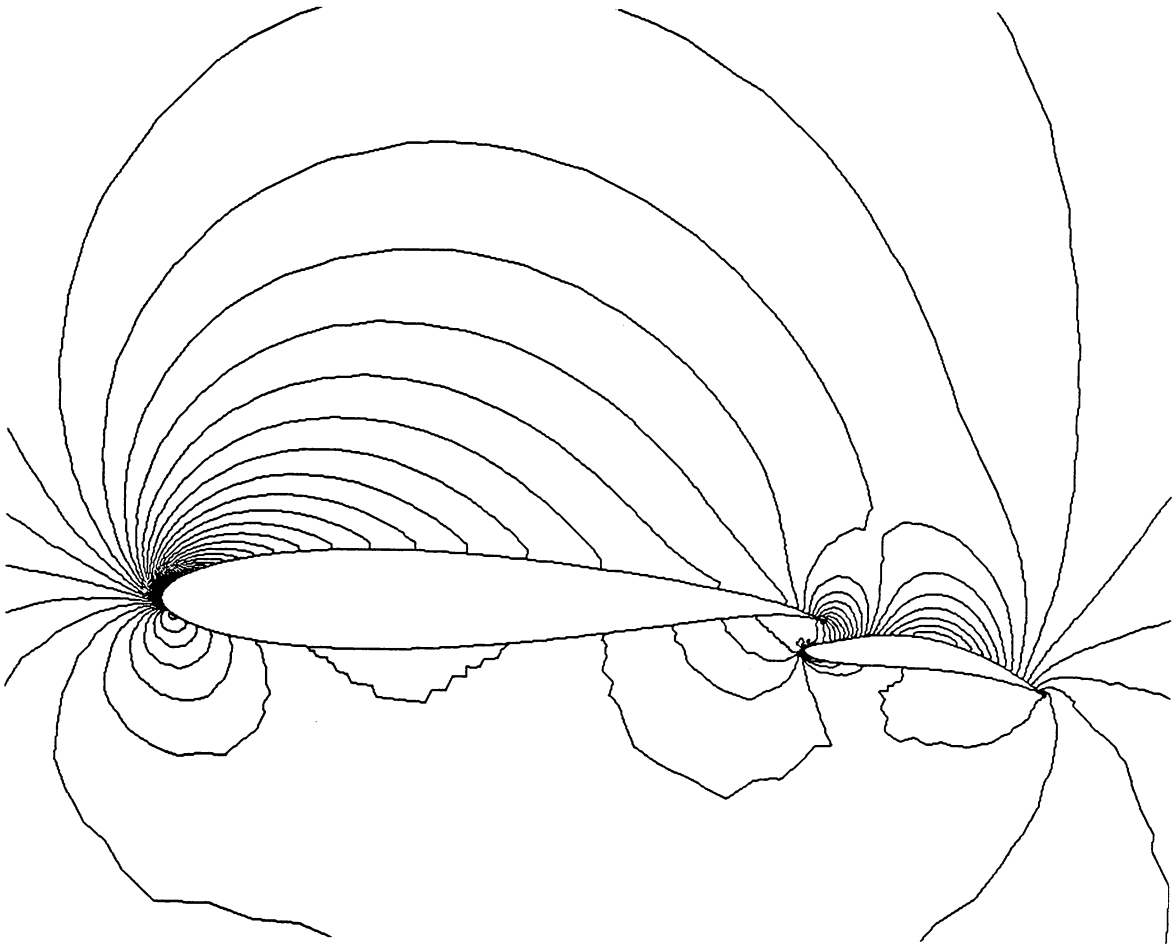

Figure 56. Density Contours - Karman-Trefftz Airfoil and Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

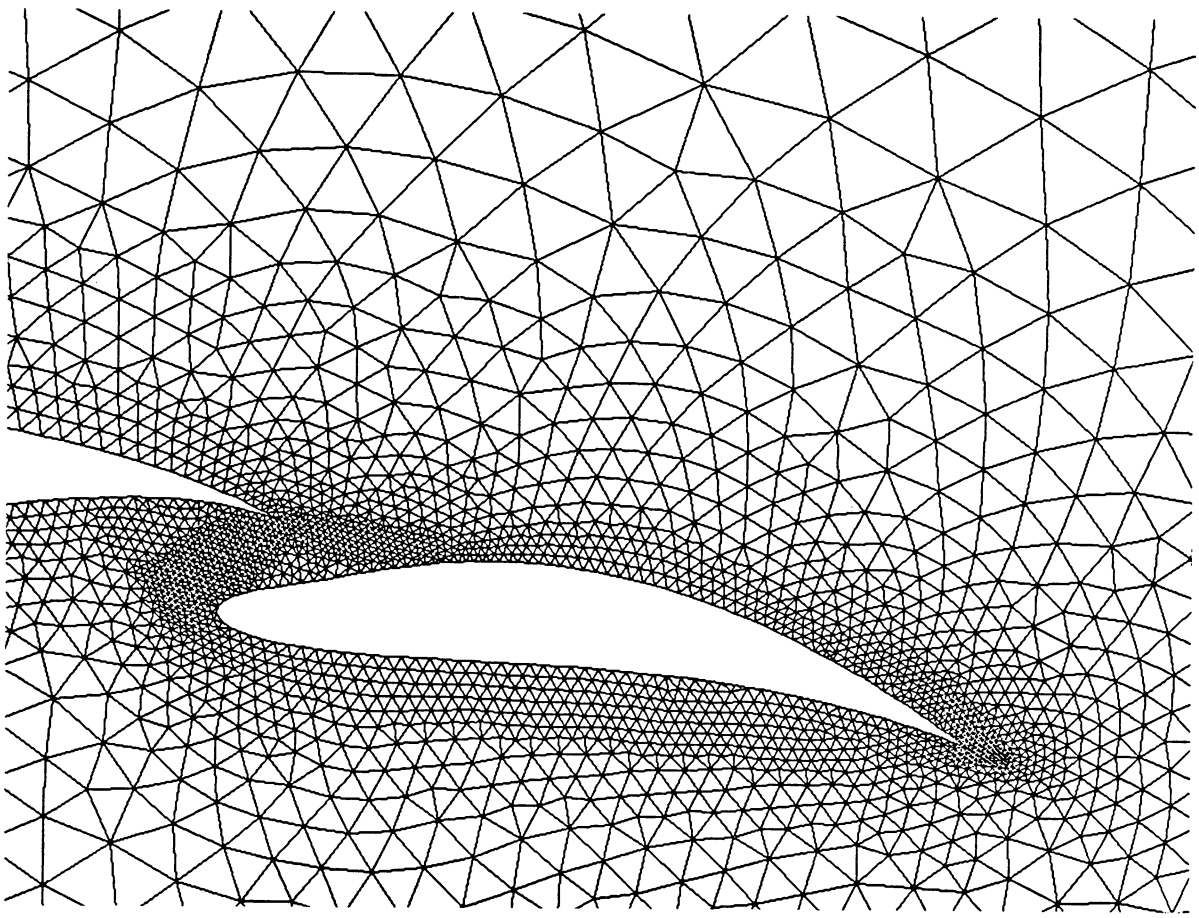

Figure 57. Close-up of Unstructured Mesh - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$: 14587 elements, 7566 nodes.

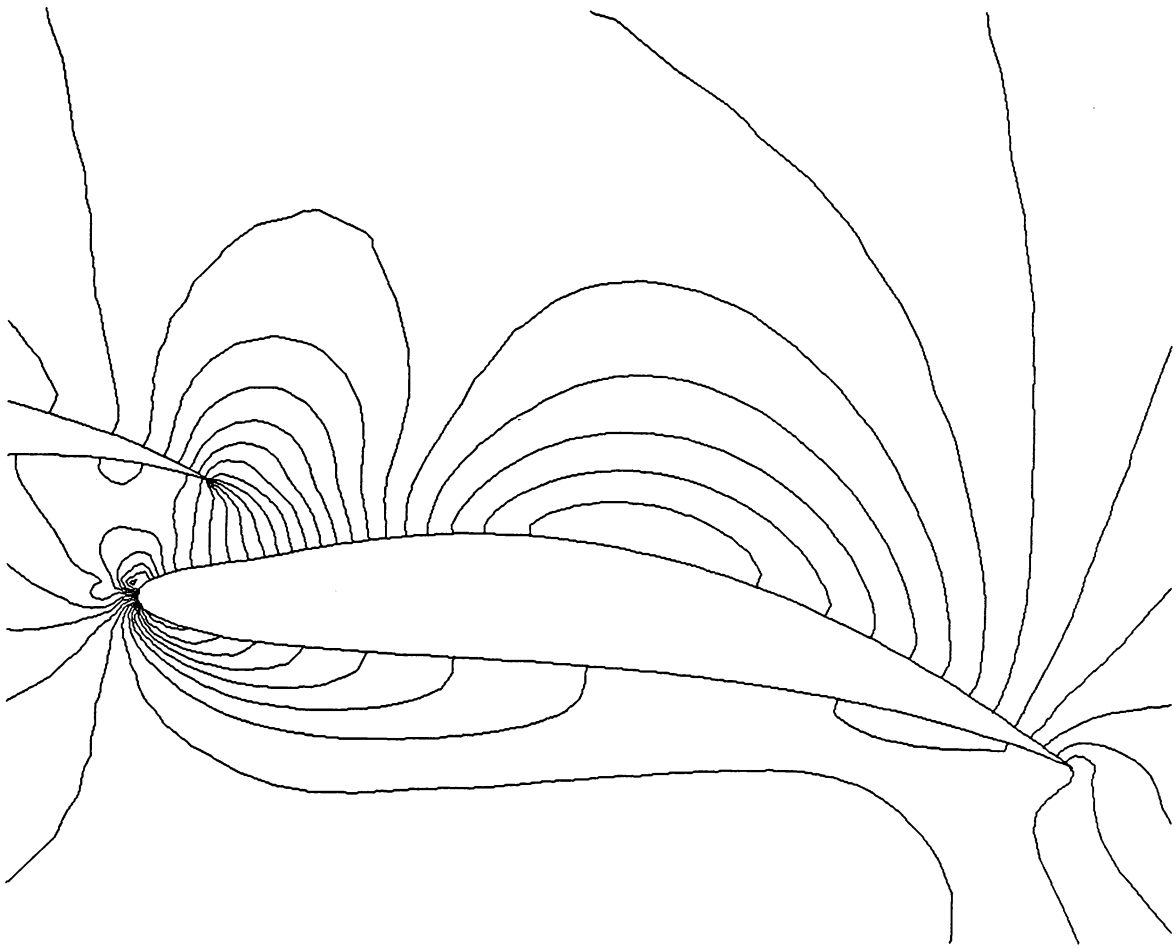

Figure 58. Mach Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP: $C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

Figure 59. Pressure Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

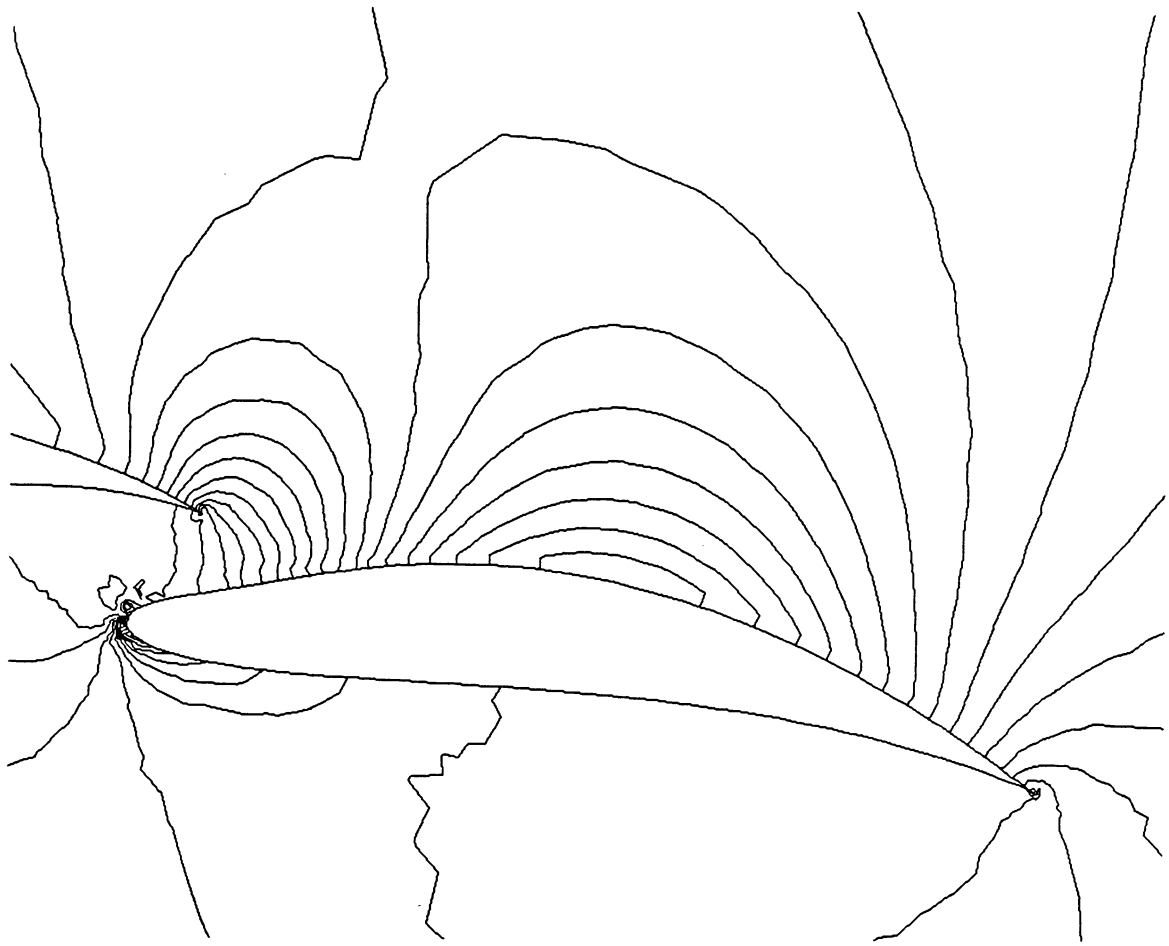

Figure 60. Density Contours - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation.

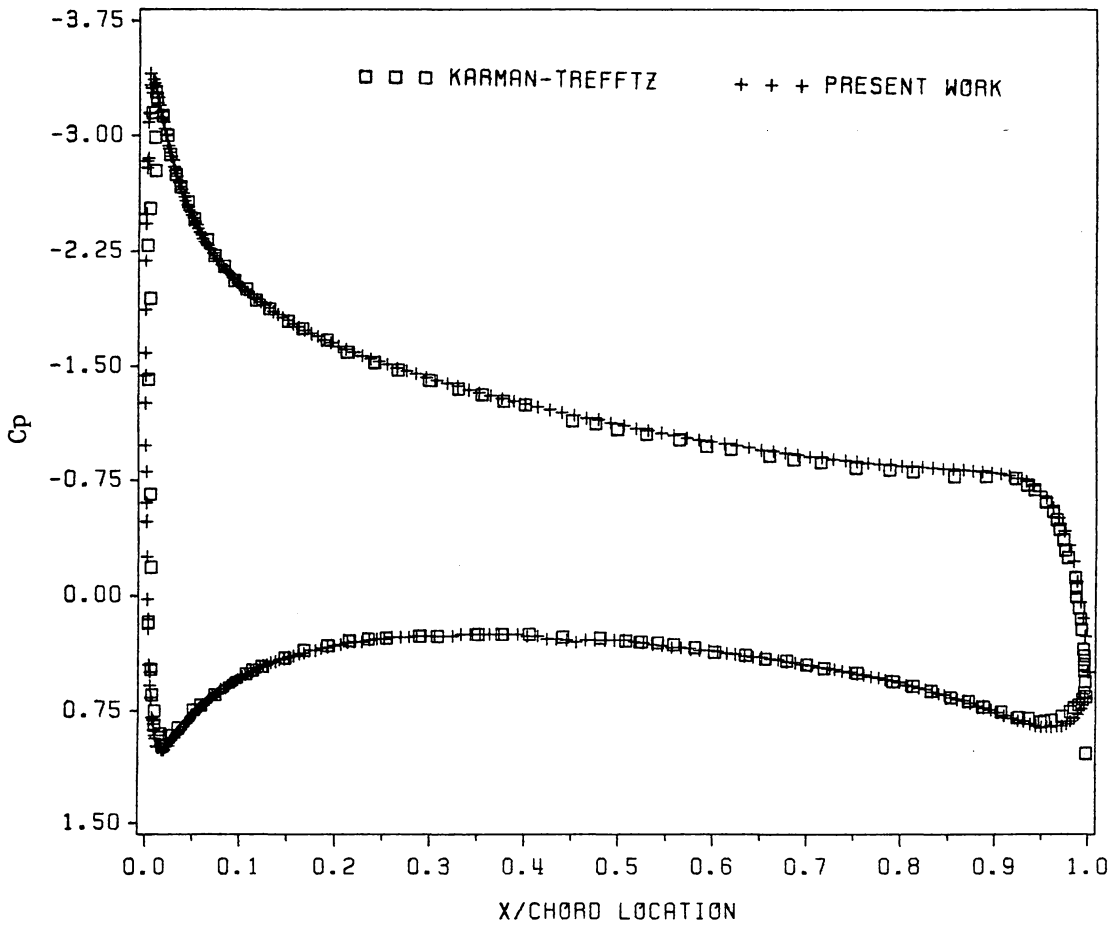

Figure 61. Surface Pressure Comparison - Karman-Trefftz Airfoil, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation versus the incompressible exact solution (Ref. 90).

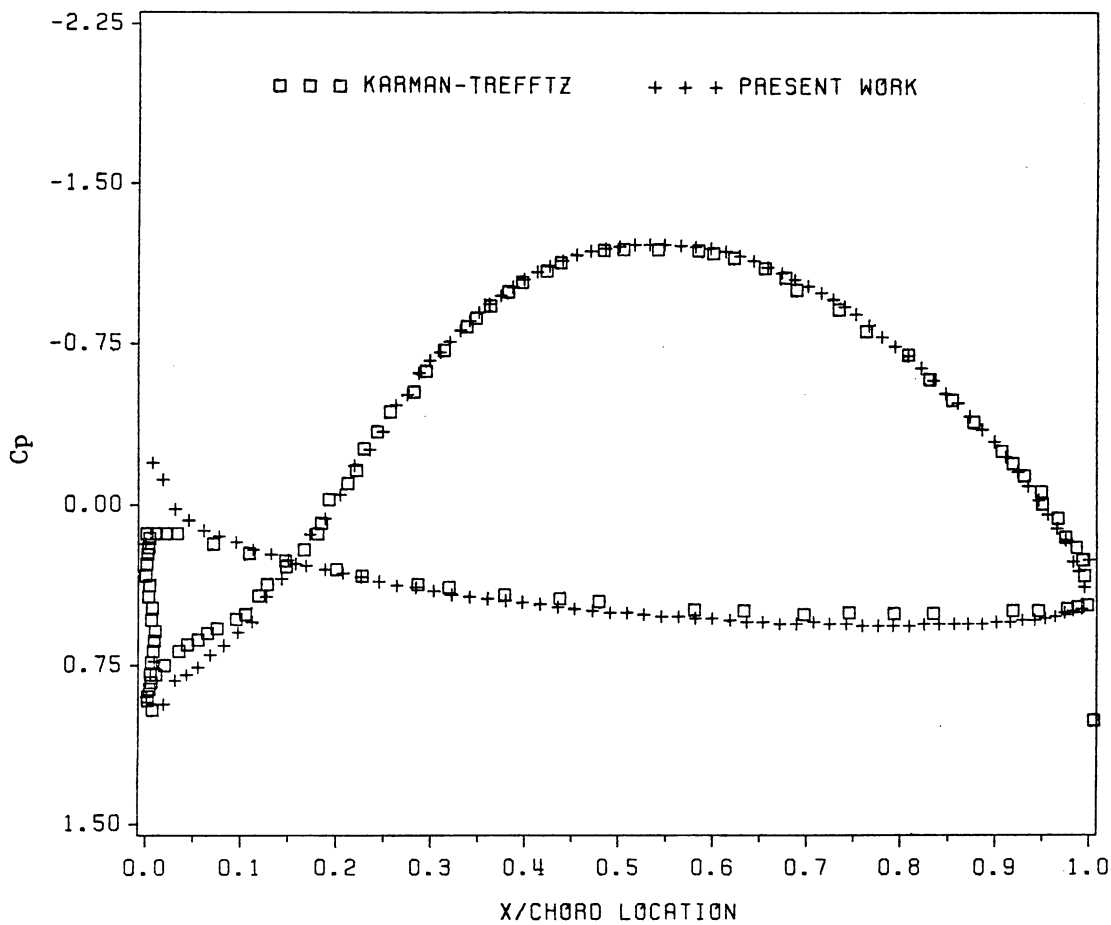

Figure 62. Surface Pressure Comparison - Karman-Trefftz Flap, $M_\infty = 0.125$, $\alpha = 0°$: $\sigma = -1$, AIRFOIL:$C_l = 1.73280$, $C_d = -0.09309$, FLAP:$C_l = 0.33791$, $C_d = 0.08844$, 14587 elements, 7566 nodes, Normal extrapolation versus the incompressible exact solution (Ref. 90).

The vita has been removed from the scanned document