EFFECT OF THE JACOBIAN EVALUATION ON DIRECT SOLUTIONS OF THE EULER EQUATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖMER ONUR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF AEROSPACE ENGINEERING

DECEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Nafiz Alemdaroğlu
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Sinan Eyi
Supervisor

Examining Committee Members

Prof. Dr. Haluk Aksel _____

Prof. Dr. Nafiz Alemdaroğlu _____

Assoc. Prof. Dr. Sinan Eyi _____

Assoc. Prof. Dr. Yusuf Özyörük _____

Assoc. Prof. Dr. İsmail H. Tuncer _____

ABSTRACT

EFFECT OF JACOBIAN EVALUATION ON DIRECT SOLUTIONS OF THE EULER EQUATIONS

Onur, Ömer

M. S., Department of Aerospace Engineering

Supervisor: Assoc. Prof. Sinan Eyi

December 2003, 109 Pages

A direct method is developed for solving the 2-D planar/axisymmetric Euler equations. The Euler equations are discretized using a finite-volume method with upwind flux splitting schemes, and the resulting nonlinear system of equations are solved using Newton's Method. Both analytical and numerical methods are used for Jacobian calculations. Numerical method has the advantage of keeping the Jacobian consistent with the numerical flux vector without extremely complex or impractical analytical differentiations. However, numerical method may have accuracy problem and may need longer execution time.

In order to improve the accuracy of numerical method detailed error analyses were performed. It was demonstrated that the finite-difference perturbation magnitude and computer precision are the most important parameters that affect the accuracy of numerical Jacobians. A relation was developed for optimum perturbation magnitude that can minimize the error in numerical Jacobians. Results show that

very accurate numerical Jacobians can be calculated with optimum perturbation magnitude.

The effects of the accuracy of numerical Jacobians on the convergence of flow solver are also investigated. In order to reduce the execution time for numerical Jacobian evaluation, flux vectors with perturbed flow variables are calculated for only related cells. A sparse matrix solver based on LU factorization is used for the solution, and to improve the Jacobian matrix solution some strategies are considered. Effects of different flux splitting methods, higher-order discretizations and several parameters on the performance of the solver are analyzed.

Keywords: Direct Flow Solution, 2-D Planar/Axisymmetric Euler Equations, Newton's Method, Numerical Jacobians, Analytical Jacobians, Sparse Matrix Solvers, Upwind Flux Splitting Methods

ÖZ

JACOBIANLARIN DEĞERLENDİRİLMESİNİN EULER DENKLEMLERİNİN DİREKT ÇÖZÜMLERİNE ETKİSİ

Onur, Ömer
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Sinan Eyi

Aralık 2003, 109 sayfa

2-boyutlu düzlemsel/eksensimetrik Euler denklemleri için bir direkt çözüm metodu geliştirilmiştir. Euler denklemleri akış yönlü akı bölme yöntemlerinin kullanıldığı bir sonlu-hacim metodu ile ayrıştırılmış, ve ortaya çıkan doğrusal olmayan denklemler sistemi Newton Metodu ile çözülmüştür. Jacobian hesaplamalarında analitik ve sayısal metodların her ikisi de kullanılmıştır. Sayısal metod çok karışık yada uygulanamayan analitik türevler içermeden Jacobianı sayısal akı yöneyiyle tutarlı olarak saklama yararına sahiptir. Buna rağmen, sayısal metodun doğruluk problemi olabilir ve daha uzun uygulama zamanı gerektirebilir.

Sayısal metodun doğruluğunu ilerletmek için detaylı hata analizleri yapılmıştır. Gösterilmiştir ki sonlu-farklar değiştirme büyüklüğü ve bilgisayar kesinliği sayısal Jacobianların doğruluklarını etkileyen en önemli parametrelerdir. Sayısal Jacobianlardaki hatayı en aza indirgeyen en uygun değiştirme büyüklüğü için bir

bağlantı geliştirilmiştir. Sonuçlar en uygun değiştirme büyüklüğü kullanılarak çok doğru sayısal Jacobianların hesaplanmasının mümkün olduğunu göstermiştir.

Sayısal Jacobianların doğrulularının akış çözücünün yakınsaması üserine etkileri de incelenmiştir. Sayısal Jacobianları değerlendirilmesindeki uygulama zamanını düşürmek için, değiştirilen akış değişkenlerini içeren akı yöneyleri sadece ilgili hücrelerde hesaplanmıştır. Akış çözümü için LU çarpanlarına ayırma yöntemini temel alan bir seyrek matris çözücü kullanılmıştır ve Jacobian matris çözümünü geliştirmek için bazı stratejiler uygulanmıştır. Farklı akı bölme yöntemlerinin, yüksek-dereceli ayrıştırmaların, ve birçok parametrenin çözücünün performansı üzerindeki etkileri analiz edilmiştir.

Anahtar Kelimeler: Direkt Akış Çözümü, 2-Boyutlu Düzlemsel/Eksensimetrik Euler Denklemleri, Newton Metodu, Sayısal Jacobianlar, Analitik Jacobianlar, Seyrek Matris Çözücüleri, Akış Yönlü Akı Bölme Yöntemleri

To everybody who always have a place in my heart,

ACKNOWLEDGMENTS

My first appreciation is to my supervisor, Assoc. Prof. Dr. Sinan Eyi, who shares all his experience and knowledge with me during this study. Without his advises, guidance and easy disposition this study could not be realized.

I would like to thank to the entire staff of the Department of Aerospace Engineering; all the instructors for giving their best for my academic education, all the research assistants for realizing a peaceful environment, and technical and administrative staff for their great assistance to all of us.

My special thanks go to Mustafa Kaya who has been my roommate for the last three years with his great patience and indulgence. I also wish to thank all my friends especially Aycan Okan, Cengizhan Bahar, Emre Yavuzoğlu, Özgür Demir, Demet Ülker, Gizem Karslı and Ebru Sarıgöl for their supports, understandings and motivations throughout this study.

I also want to thank my sweetheart Songül Erol for bringing happiness and joy to my life for the last four months.

And finally, I am very grateful to my parents Hatice & İsmail Onur for their endless love, great encouragement and patient supports during my entire life.

I want to finish with the following epigram about life:

This is your life, and it's ending one minute at a time.

Until that minute, try to enjoy life and live best in all time.

TABLE OF CONTENTS

ABSTRACT iii
ÖZ v
ACKNOWLEGDEMENTS viii
TABLE OF CONTENTS ix
LIST OF TABLES xii
LIST OF FIGURES xiii
LIST OF SYMBOLS xvi

CHAPTER

1. INTRODUCTION 1
   1.1 Motivation 1
   1.2 Objective 3
   1.3 Literature Survey 3
   1.4 Outline 6
2. FLOW MODEL 8
   2.1 Introduction 8
   2.2 Governing Equations 9
   2.3 Spatial Discretization 12
   2.4 Flux Splitting 14
      2.4.1 Steger-Warming Flux Vector Splitting 17
      2.4.2 Van Leer Flux Vector Splitting 18
      2.4.3 Roe Flux Difference Splitting 19
   2.5 Higher-Order Schemes with Limiters 21

2.5 Boundary Conditions 23
2.5.1 Inflow BC 24
2.5.2 Outflow BC 24
2.5.3 Symmetry BC 25
2.5.4 Wall BC 25
3. SOLUTION METHOD 26
3.1 Introduction 26
3.2 Newton's Method 27
3.3 Evaluation of the Jacobians 28
3.3.1 Analytical Jacobian Derivation 28
3.3.2 Numerical Jacobian Calculation 30
3.4 Matrix Structure 32
3.5 Matrix Solution Strategies 35
3.5.1 Frozen Jacobian 35
3.5.2 Initial Guess 36
4. ACCURACY OF NUMERICAL JACOBIANS 39
4.1 Introduction 39
4.2 Error Analysis 40
4.2.1 Truncation Error 40
4.2.2 Condition Error 41
4.2.3 Total Error 42
4.3 Effect of Computer Precision 43
4.4 Optimum Perturbation Magnitude Analysis 44
4.5 Test Case Results 46
5. SOLVER PERFORMANCE 64
5.1 Introduction 64
5.2 Test Case Results 65
5.3 Results for Different Flux Splitting Schemes 78
5.4 Results for Higher-Order Discretizations 86
5.5 Results for Different Geometry and Flow Conditions 92
6. CONCLUSION AND RECOMMENDATIONS 99

REFERENCES 103

APPENDIX A: ANALYTICAL FLUX JACOBIANS FOR STEGER-WARMING FLUX VECTOR SPLITTING SCHEME 106

LIST OF TABLES

TABLE

4.1 ε_{opt} analysis using average values for single precision.....49
4.2 ε_{opt} analysis using machine epsilon for single and double precision.....49
4.2 Optimum perturbation magnitude ε_{opt} results for single precision.....50
4.2 Optimum perturbation magnitude ε_{opt} results for double precision 50

LIST OF FIGURES

FIGURES

2.1 Generalized transformation from the physical to the computational domain 10
2.2 A typical control volume 14
3.1 5-point stencil 33
3.2 Matrix structure from 5-point stencil 33
3.3 9-point stencil 34
3.4 Matrix structure from 9-point stencil 34
4.1 15^{o} ramp geometry 51
4.2 Effect of ε on $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian error contour for forward differencing in single precision 52
4.3 Effect of ε on $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian error contour for forward differencing in double precision 53
4.4 $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian contours for forward differencing in single precision 54
4.5 $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian contours for forward differencing in double precision 55
4.6 Effect of control on total max. and avg.errors for $\partial \hat{R}_{pl} / \partial \hat{W}$ in single precision 56
4.7 Effect of control on total max. and avg.errors for $\partial \hat{R}_{pl} / \partial \hat{W}$ in double precision 57
4.8 Effect of ε on total max. error for all Jacobians in single precision 58
4.9 Effect of ε on total max. error for all Jacobians in double precision 59

4.10 Effect of ε on total avg. error for all Jacobians in single precision 60
4.11 Effect of ε on total avg. error for all Jacobians in double precision 61
4.12 Effect of axisymmetry on total max. and avg.errors for $\partial\hat{R}/\partial\hat{W}$ in single precision 62
4.13 Effect of axisymmetry on total max. and avg.errors for $\partial\hat{R}/\partial\hat{W}$ in double precision 63
5.1 Mach contours for 1st order Steger-Warming scheme in double precision 69
5.2 Effect of Δt on the convergence history for planar density residual 70
5.3 Effect of Δt on the convergence history for axisymmetric density residual 71
5.4 Effect of freezing on the convergence history for planar density residual 72
5.5 Effect of freezing on the convergence history for axisymmetric density residual 73
5.6 Effect of ε on the convergence history for planar density residual in single precision 74
5.7 Effect of ε on the convergence history for planar density residual in double precision 75
5.8 Effect of ε on the convergence history for axisymmetric density residual in single precision 76
5.9 Effect of ε on the convergence history for axisymmetric density residual in double precision 77
5.10 Mach contours for 1st order Van Leer scheme in double precision 80
5.11 Mach contours for 1st order Roe scheme in double precision 81
5.12 Effect of different flux splitting schemes on the convergence history for planar density residual in single precision 82
5.13 Effect of different flux splitting schemes on the convergence history for planar density residual in double precision 83
5.14 Effect of different flux splitting schemes on the convergence history for axisymmetric density residual in single precision 84
5.15 Effect of different flux splitting schemes on the convergence history for axisymmetric density residual in double precision 85

5.16 Mach contours for 2nd order Steger-Warming scheme in double precision 88
5.17 Mach contours for 2nd order Van Leer scheme in double precision 89
5.18 Mach contours for 2nd order Roe scheme in double precision 90
5.19 Effect of different 2nd order flux splitting schemes on the convergence history for density residual in double precision 91
5.20 Bump geometry 94
5.21 Mach contours for 2nd order Roe scheme in double precision for supersonic flow 95
5.22 Convergence history for 2nd order Roe scheme in double precision for supersonic flow 96
5.23 Mach contours for 2nd order Roe scheme in double precision for subsonic flow 97
5.24 Convergence history for 2nd order Roe scheme in double precision for subsonic flow 98

LIST OF SYMBOLS

LATIN SYMBOLS

A	Jacobian matrix of F flux vector
B	Jacobian matrix of G flux vector
c	Speed of sound
e	canonical vector
e_t	Total energy per unit volume
E	Round-off error
E_C	Condition error
E_R	Bound of E_C or precision error
E_T	Truncation error
E_{TOTAL}	Total error
$f(x)$	A function of variable x
$\tilde{f}(x)$	Computed value of f(x)
F	Inviscid flux vector in x-direction
G	Inviscid flux vector in y-direction
h	Enthalpy
H	Axisymmetric source vector
J	Jacobian of transformation from Cartesian to generalised coordinates
k	Metrics
$\tilde{k}$	Directional cosines
M	Mach number
p	Pressure
Q_Λ	Right eigenvector matrix of A, B
r	Ratio of differences

R	Residual vector of the system
R_{ax}	Axisymmetric case residual value
R_{frz}	Freezing residual value
R_{pl}	Planar case residual value
Δt	Time-like term added to Jacobian matrix diagonal
Δt^0	initial value for Δt
Δt_{rm}	Removal value for Δt
u,v	Velocity components
U,V	Contravariant velocity components
$\tilde{U}$	Contravariant velocity components in terms of $\tilde{k}$
x,y	Components of Cartesian coordinates
W	Vector of conserved flow variables in Cartesian coordinates

GREEK SYMBOLS

α	Factor to reduce W
Δ	Forward difference operator
ε	Finite-difference perturbation magnitude
ε_M	Computer precision or machine epsilon
ε_{OPT}	Optimum value of ε
ϕ	Interpolation limiter function
γ	Ratio of specific heats
$\xi,,\eta$	Components of curvilinear coordinates
κ	Interpolation order parameter
λ	Eigenvalue
Λ	Diagonal matrices including eigenvalues of A, B
ρ	Density
σ	Axisymmetry parameter
ζ	A value between the original and perturbed variable

∂	Partial differentiation operator
$\in$	A small number to prevent division by zero
$\mathbb{F}$	Generalized flux vector
∇	Backward difference operator

OVERLINES

^	Property in generalized coordinates
~	Roe averaged value

SUBSCRIPTS

i,j	Cell centred grid indices
x,y	Differentiation with respect to x, y
ξ,η	Differentiation with respect to ξ, η
∞	Free-stream value

SUPERSCRIPTS

m	Number of possible highest bits in the binary representation of mantissa
n	Newton's method iteration number
-	Negative (left) value
+	Positive (right) value

CHAPTER 1

INTRODUCTION

1.1 Motivation

Computational Fluid Dynamics (CFD) has become a valuable tool and being widely used in all areas of science and engineering with the rapid progress in the computer technology. Many methods have been developed to understand the physics of the flow. These analyses provide a level of detail that is difficult to match with alternative analytical and experimental methods. Considering the high time and work cost of other methods, the advantage and importance of predicting the flow physics with CFD models is increasing day by day.

Many problems of interest in science and engineering only involve steady fluid flow; even the design of aerospace vehicles is predominated by steady flows. The mission performance of the vehicle is usually determined by its capability in steady flight regimes like cruise, and transient maneuvers have usually secondary influence. However, most of the CFD methods in common use solve an unsteady system of equations although the problem of interest may be steady. The use of the unsteady equation system is obviously appropriate for unsteady flows. For steady flows, the equilibrium solution is found by advancing the unsteady equation in time until a steady state is achieved.

The use of unsteady equations for finding steady solutions is very important in the development of CFD solution algorithms, and the resulting codes are efficient

and robust. However, such computation of steady flows may not be the best way. The steady flow can be obtained using “direct” solution methods. Direct solution methods consider the domain as a whole and compute the steady flow without an advance in time of an unsteady analysis. Indeed, for the solution of steady flows with direct methods, neither time nor any time-like variable appears in the set of governing equations or the solution algorithm.

Advantages of the direct solution approach may include stability, efficiency, increased area of applicability, and the availability of additional information about the flow that can not be reached using unsteady equations. Solving the whole domain at once without considering any time or any time-like variable makes this approach much more stable. An increase in efficiency is possible, because generally very small number of iterations are required by a direct method for the flow to converge. The increased area of applicability comes out from the ease of obtaining sensitivity derivatives for design optimization, or including structural or thermal parameters in the system for such analyses.

The direct solution technique is Newton’s method. This method is widely used for finding the solution of a non-linear system of algebraic equations, and providing quadratic convergence. Although Newton’s method has been available for long years, the current development of very powerful computers has now made the procedure more applicable for extremely large systems of equations of CFD.

Solving the whole domain at once requires the calculation of Jacobian matrix, which may be very large according to the CFD model used to predict the flow physics. Also, derivation of the Jacobian matrix entries by analytical means become more difficult, as the discretization of the fluid flow equations becomes more complex. Thus, accurate computation of the Jacobians numerically, and fast solution of the matrix are very important for the performance of direct solution technique.

1.2 Objective

The first objective of this study is to analyze the accuracy of numerical Jacobians used in the solver considering the effects of finite-difference perturbation magnitude and computer precision. The second objective is to investigate the effects of the accuracy of Jacobians on the performance of the direct flow solver in terms of convergence and CPU time. The third objective is to improve the efficiency of the Jacobian matrix solution using some strategies like diagonal-term addition, Jacobian freezing. Also, the developed solver is tested for different flux splitting methods, and higher-order discretization schemes. A fourth objective can be to investigate the benefits of using the same flux calculation scheme for both Jacobian and residual calculation in terms of the convergence of the solver.

1.3 Literature Survey

Newton's method has been used by several researchers to address a variety of fluid dynamics problems. In the cited literature, there are a number of common motivations for using Newton's method to solve the fluid flow. Many researchers were drawn to Newton's method because of its quadratic convergence. A flow solution, which may take hundreds or thousands of iterations with iterative methods, can be obtained in ten or twenty iterations with Newton's method. Some researchers have used Newton's method for strongly coupled multidisciplinary analyses, which is another important feature of the method.

Wigton [1] calculated the flow about multi-element airfoils based a streamline formulation of the Euler equations. A Newton's method was used for the solution of the multi-element airfoil problem because of the poor convergence rate observed for transonic calculations using the conventional approach of solving unsteady equations. Wigton used the symbolic manipulation expert system MACSYMA to compute the Jacobian matrix derivatives and to output the Fortran code. He described a

technique called nested dissection node reordering which significantly reduces the storage requirements and factorization time relative to the usual banded matrix approach. Although it is complicated, the efficiency gains make three-dimensional calculations using Newton's method practical. This feature is very important and similar operations are included in today's advanced sparse matrix solvers.

Bender and Khosla [2] applied Newton's method to potential and simplified Navier-Stokes equations to the cavity problem and to transonic airfoils. One interesting feature was the use of a residual reduction correction, which can be very useful in case of poor initial conditions or convergence problems.

One of the best implementations of Newton's method was transonic airfoil computation of Venktakrishnan [3]. Van Leer flux-vector splitting and Roe's flux difference splitting were applied to full Euler and Navier-Stokes equations. Venktakrishnan used nested dissection like Wigton with advanced sparse matrix inversion routines, and a diagonal term modification that improves the Newton's method greatly for convergence even from poor initial guess instead of Bender and Khosla's residual reduction. This diagonal term modification is also applied in this study.

Orkwis [4] developed a new Newton's solver for calculating 2-D planar or axisymmetric, laminar or turbulent Navier-Stokes equations. He first analyzed high-speed planar flows [5] and extended his solver to axisymmetric flows [6]. In all cases, a second order Roe's flux difference splitting scheme is used for inviscid flux calculations. Turbulent flows are calculated using Baldwin–Lomax turbulent model. Like Wigton, he used the symbolic manipulation system MACSYMA to calculate exact Jacobian matrix entries. Orkwis mentioned that these Jacobians took approximately 40000 lines of Fortran code to implement, some of which may not be vectorizable. The diagonal term modification of Venktakrishnan is employed in order to overcome non-optimal initial conditions.

In the later studies, Orkwis compared the performance of several Newton's and quasi-Newton's method solvers [7]. He showed that despite of not having quadratic convergence, quasi-Newton's methods could be more efficient than the exact Newton's method. After his studies with Kim [8] on Jacobian matrix simplification ideas like partial and global freezing for Newton and Newton-like methods, they showed that approximate methods can also give quadratic or better convergence rates with proper implementations.

Felker [9] developed a method for directly solving steady, 2-D, compressible Navier-Stokes equations with fluid/structure coupling. The inviscid terms are discretized using Roe's flux difference splitting method. An algebraic eddy viscosity model represents the effect of turbulence. This fluid dynamics model has been coupled with a finite-element structure model. By this way, both the fluid flow and the structural deformations in static aeroelasticity are analyzed. The efficiency of the full Newton method is compared with that of a modified one and the effect of the data storage is analyzed.

Whitfield and Taylor [10] presented a Newton-relaxation solver for solving both 3-D compressible and incompressible flows. High order Roe flux difference splitting scheme is used. Since obtaining the Jacobian matrix analytically is impractical for such a discretization, it is approximated using numerical Roe flux vectors. This is one of the first implementations of numerical Jacobian calculation.

As a continuation of the studies of Whitfield, Vanden [11, 12] applied direct and iterative methods to solve 3-D Euler equations. Euler equations are discretized using numerical derivatives of the numerical flux vector. Direct methods include solution of block-tridiagonal systems with a block LU factorization followed by forward or backward substitution. Instead of conventional matrix structure, a diagonal plane structure was presented to decrease the memory requirement. The robustness of several iterative methods is verified with comparison with direct methods.

Orkwis and Vanden [13] combined their studies in order to compare numerical and analytical approaches for forming the Jacobian matrix that they used. The derivation of the analytical Jacobians using the MACSYMA manipulation system and the numerical differentiation procedure to find the numerical Jacobians is explained in detail. They used 2-D laminar compressible Navier-Stokes equations discretized with Roe flux splitting scheme. Studies over supersonic flat plate and compression corner geometries showed that numerical method has the practicality and simplicity advantages over the exact Jacobian approach.

One of the recent studies on flux Jacobians is of Aberle and Shumlak [14]. They used Roe's approximate Riemann solver over ideal 1-D magneto-hydrodynamic equations. The accuracy, convergence and performance of the analytical and numerical methods to determine the flux Jacobians are compared. It is found that the accuracy and the convergence are identical, while analytical formulation requires less execution time.

1.4 Outline

Chapter 2 introduces the basic theory of 2-D planar/axisymmetric Euler equations in generalized coordinates. Finite-volume spatial discretization of the governing equations including Steger-Warming, Van Leer, and Roe's upwind flux splitting methods are explained in detail. Higher order discretization schemes are discussed with the introduction of limiters. In addition, possible boundary conditions for the flow model are presented.

The direct solution of the Euler equations is studied in Chapter 3. Newton's method is introduced, and the numerical and analytical calculations of the flux and residual Jacobians are explained in detail. The structure of the Jacobian matrix required in the Newton's method is discussed. The solution strategies of Jacobian ma-

trix to improve the performance like freezing and good initial guess are also presented.

Chapter 4 is related to the accuracy of numerical Jacobians. The effects of finite-difference perturbation magnitude and computer precision on the accuracy of numerical Jacobians are investigated. After possible types of errors are explained in detail, errors between numerical and analytical flux and residual Jacobians are analyzed for the test case being supersonic ramp. An optimum finite-difference perturbation magnitude is obtained in two ways; using a trial-error procedure and a derived relation from the error analysis. The precision of the computer is discussed. Several graphical and tabulated results are presented.

The effect of the Jacobians on the performance of the developed direct flow solver is studied in Chapter 5. The convergence history and CPU time results are presented for the test case. Comparisons for different flux splitting schemes, higher order discretizations, and another geometry with different flow conditions are done. Again, several tabulated and graphical results are presented in this chapter.

Finally, Chapter 6 makes some conclusions about the study and recommendations for future research.

CHAPTER 2

FLOW MODEL

2.1 Introduction

The flow model as a whole should have the capability to retain the flow physics for the given flow conditions. The governing equations of the fluid flow, the employed discretization scheme, the choice of appropriate boundary conditions, and the grid density are very important factors for a better flow simulation. With simple flow models, the high computational cost due to large grid sizes and high-level flow physics may be reduced, but accuracy of the flow simulation may not be reliable.

In this study, 2-D planar/axisymmetric Euler equations are solved in generalized coordinate system. Euler equations have capability of solving inviscid rotational flow. Although the Navier-Stokes equations with turbulence modeling provide better solution including viscous flow, because of several reasons they are not used in this study. The main objective of this thesis is to compare the performance of numerical and analytical Jacobians in terms of accuracy and convergence. However, it is not easy to obtain analytical Jacobians for Navier-Stokes equations and turbulence modeling. In addition, the grid size will be larger for Navier-Stokes equations. This will make the size of Jacobian matrix even larger, and the solution of this system may not be possible with present computer resources. Thus, starting with a simpler flow model that retains the requirements is reasonable. After sufficient effort is done, with the use of new and modified solvers models that are more complex can be considered.

2.2 Governing Equations

The universal laws of the conservation of mass, momentum, and energy are the basis of the fundamental equations of fluid dynamics. The steady, 2-D planar/axisymmetric Euler equations in Cartesian coordinates [15], written in non-dimensional form without body forces are;

$$\frac{\partial F(W)}{\partial x} + \frac{\partial G(W)}{\partial y} + \sigma H(W) = 0 \tag{2.1}$$

For a 2-D planar flow $\sigma = 0$, and for a 2-D axisymmetric flow $\sigma = 1$, with x and y the axial and radial directions respectively. Here, the conserved flow variable vector W, the flux vectors F and G, and the axisymmetric source vector H are:

$$W = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_t \end{bmatrix} \qquad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e_t + p)v \end{bmatrix}$$
$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e_t + p)u \end{bmatrix} \qquad H = \frac{1}{y}\begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ (\rho e_t + p)v \end{bmatrix} \tag{2.2}$$

where ρ is the density, u and v are the x and y components of the velocity vector, respectively, p is the pressure, e_t is the total energy per unit volume. Pressure is obtained from the ideal gas relation as:

$$p = (\gamma - 1)\rho\left[e_t - \frac{1}{2}(u^2 + v^2)\right] \tag{2.3}$$

In order to apply the numerical algorithm and boundary conditions easily to an arbitrary geometry, the governing equations in the physical domain or Cartesian coordinates must be transformed to the computational domain or generalized coordinates [15, 16]. Figure 2.1 shows the 2-D coordinate transformation between physical and computational domains, where x and y are the physical, ξ, *and* η are the curvilinear coordinates, respectively.

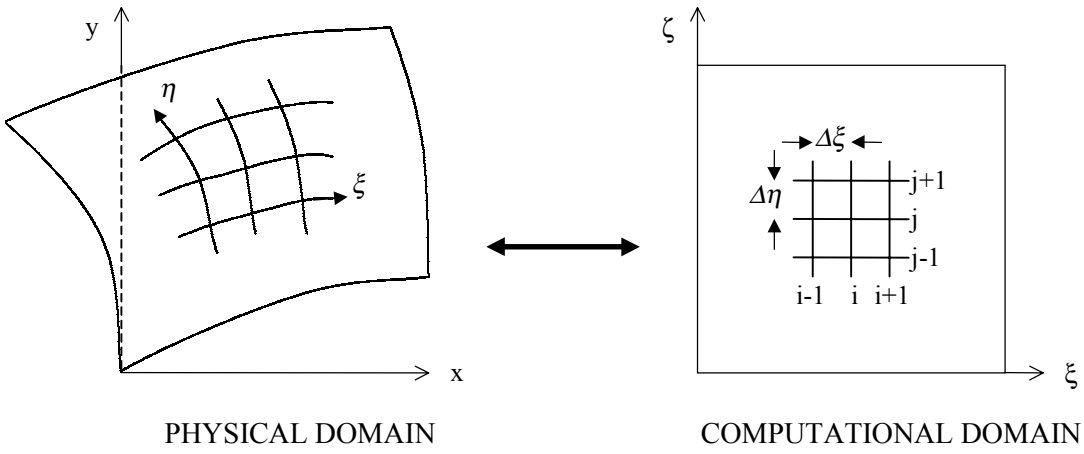


Figure 2.1 Generalized transformation from the physical to the computational domain

The general transformation and its inverse are of the form:

$$\xi = \xi(x, y) \qquad x = x(\xi, \eta)$$
$$\eta = \eta(x, y) \qquad y = y(\xi, \eta) \tag{2.4}$$

By using the chain rule of partial differentiation, the partial derivatives in the physical domain become

$$\begin{aligned}\frac{\partial}{\partial x} &= \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial y} &= \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta}\end{aligned} \tag{2.5}$$

where the metrics ξ_x, η_x, ξ_y, η_y are obtained in the following manner. The differential expressions in their matrix form are:

$$\begin{aligned} d\xi &= \xi_x dx + \xi_y dy & \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} &= \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \\ d\eta &= \eta_x dx + \eta_y dy & & \\ dx &= x_\xi d\xi + x_\eta d\eta & \begin{bmatrix} dx \\ dy \end{bmatrix} &= \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \\ dy &= y_\xi d\xi + y_\eta d\eta & & \end{aligned} \tag{2.6}$$

Therefore,

$$\begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}^{-1} \tag{2.7}$$

Thus, the transformation metrics are:

$$\xi_x = J y_\eta \qquad \xi_y = -J x_\eta \qquad \eta_x = -J y_\xi \qquad \eta_y = J x_\xi \tag{2.8}$$

where J is the coordinate transformation Jacobian, defined as:

$$J = \frac{\partial(\xi,\eta)}{\partial(x,y)} = \begin{vmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{vmatrix} = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix}^{-1} = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \tag{2.9}$$

The metrics can be determined by using a finite difference scheme in the computational domain. Applying this generalized transformation to Equation (2.1), the following transformed equations are obtained for Euler equations:

$$\frac{\partial \hat{F}(\hat{W})}{\partial \xi} + \frac{\partial \hat{G}(\hat{W})}{\partial \eta} + \sigma \hat{H}(\hat{W}) = 0 \tag{2.10}$$

where the flux terms are:

$$\hat{W} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_t \end{bmatrix} \qquad \hat{G} = J^{-1}\begin{bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ (\rho e_t + p)V \end{bmatrix}$$
$$\hat{F} = J^{-1}\begin{bmatrix} \rho U \\ \rho uU + \xi_x p \\ \rho vU + \xi_y p \\ (\rho e_t + p)U \end{bmatrix} \qquad \hat{H} = J^{-1}\frac{1}{y}\begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ (\rho e_t + p)v \end{bmatrix} \tag{2.11}$$

where U and V are contravariant velocity components defined as,

$$\begin{aligned} U &= \xi_x u + \xi_y v \\ V &= \eta_x u + \eta_y v \end{aligned} \tag{2.12}$$

2.3 Spatial Discretization

One of the most common approaches for spatial discretization is the finite volume method. In this method, the computational domain is divided into quadrilateral cells and the flow variables are defined at the cell centers. The grid points define cell corners and the fluxes are defined at the cell faces.

The differential form of the steady, 2-D planar/axisymmetric Euler equations given in Equation (2.10) can be discretized for an arbitrary quadrilateral control volume as given in Figure 2.1.

$$\frac{\delta_\xi \hat{F}}{\Delta\xi} + \frac{\delta_\eta \hat{G}}{\Delta\eta} + \sigma \hat{H} = 0 \tag{2.13}$$

where the spatial derivatives of the flux vectors are written conservatively as flux balances across the cell.

$$\begin{aligned} \delta_\xi \hat{F} &= (\hat{F}_{i+1/2,j} - \hat{F}_{i-1/2,j}) \\ \delta_\eta \hat{G} &= (\hat{G}_{i,j+1/2} - \hat{G}_{i,j-1/2}) \end{aligned} \tag{2.14}$$

The i±1/2 and j±1/2 denotes a cell interface and the flow variables are assumed constant over each cell. The fluxes are calculated at the cell faces by using the flow variables interpolated from the cell center values according the order of spatial discretization.

The computational domain is chosen to have equal spacing ($\Delta\xi = \Delta\eta = 1$) to simplify the differencing as shown in Figure 2.2. Equation (2.13) can then be written as:

$$(\hat{F}_{i+1/2,j} - \hat{F}_{i-1/2,j}) + (\hat{G}_{i,j+1/2} - \hat{G}_{i,j-1/2}) + \sigma \hat{H}_{i,j} = 0 \tag{2.15}$$

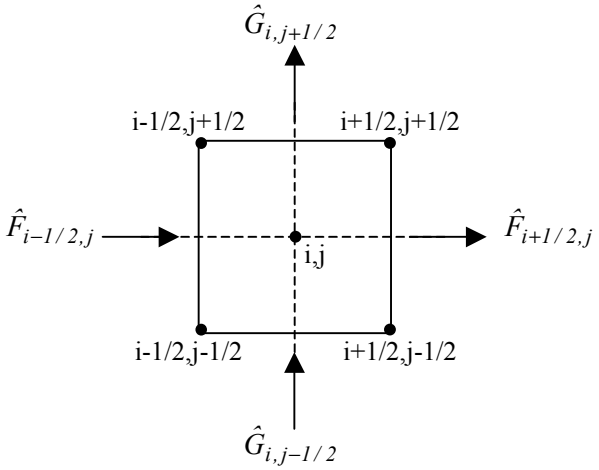


Figure 2.2 A typical control volume

The Euler equations have convective flux that represent the inviscid phenomena and are hyperbolic in nature. There are two different approaches to calculate the flux. The first approach is central differencing. In this method, the fluxes are calculated based on the averaged flow variables at the cell interface. Central schemes are easy to implement, but they require artificial dissipation. Another approach for the inviscid flux calculations is the upwinding schemes, which requires no explicit artificial dissipation. In this study, upwinding flux splitting schemes are used for the spatial discretization of the flux vector.

2.4 Flux Splitting

In general, the flux vectors of Euler equations given in Equation (2.1) have a special property. Steger and Warming [17] showed that since F and G are homogeneous functions of degree one in W, flux vectors are exactly equal to their Jacobian times the flow variable vector:

$$F=AW \qquad G=BW \tag{2.16}$$

where

$$A = \frac{\partial F}{\partial W} \qquad B = \frac{\partial G}{\partial W}$$

Considering wave splitting procedure, A and B Jacobian matrices can be diagonalized as;

$$A = Q_\Lambda \Lambda_A Q_\Lambda^{-1} \qquad B = Q_\Lambda \Lambda_B Q_\Lambda^{-1} \tag{2.17}$$

where Λ_A, Λ_B are the diagonal matrices including eigenvalues of A, B and Q_Λ is the matrix of corresponding right eigenvectors.

Since any eigenvalue can be splitted into positive and negative parts, the positive and negative Jacobian matrices become:

$$\begin{aligned} A &= Q_\Lambda (\Lambda_A^+ + \Lambda_A^-) Q_\Lambda^{-1} = Q_\Lambda \Lambda_A^+ Q_\Lambda^{-1} + Q_\Lambda \Lambda_A^- Q_\Lambda^{-1} = A^+ + A^- \\ B &= Q_\Lambda (\Lambda_B^+ + \Lambda_B^-) Q_\Lambda^{-1} = Q_\Lambda \Lambda_B^+ Q_\Lambda^{-1} + Q_\Lambda \Lambda_B^- Q_\Lambda^{-1} = B^+ + B^- \end{aligned} \tag{2.18}$$

Then, the splitted flux vectors are:

$$\begin{aligned} F &= (A^+ + A^-) W = F^+ + F^- \\ G &= (B^+ + B^-) W = G^+ + G^- \end{aligned} \tag{2.19}$$

F^+ is a subvector associated with the positive eigenvalues of A, meaning that it is a flux in the positive x-direction carrying information from left to right by positive wave speeds. Since F^+ is associated only with information coming from upstream, leftside flow variables W^- are used in its computation. Considering F^- is a subvector associated with the negative eigenvalues of A, it is computed by rightside flow variables W^+. Similar conditions apply for G^+ and G^-.

$$F^+ = A^+W^- \qquad G^+ = B^+W^-$$
$$F^- = A^-W^+ \qquad G^- = B^-W^+ \tag{2.20}$$

Implementing this splitting procedure to our case, the interface fluxes given in Equation (2.15) can be constructed as:

$$\hat{F}_{i\pm1/2,j} = \hat{F}^+(\hat{W}^-_{i\pm1/2,j}) + \hat{F}^-(\hat{W}^+_{i\pm1/2,j})$$
$$\hat{G}_{i,j\pm1/2} = \hat{G}^+(\hat{W}^-_{i,j\pm1/2}) + \hat{G}^-(\hat{W}^+_{i,j\pm1/2}) \tag{2.21}$$

Consequently, the upwind discretized form of the steady, 2-D planar/axisymmetric Euler equations can be written as:

$$\left[\hat{F}^+(\hat{W}^-_{i+1/2,j}) + \hat{F}^-(\hat{W}^+_{i+1/2,j})\right] - \left[\hat{F}^+(\hat{W}^-_{i-1/2,j}) + \hat{F}^-(\hat{W}^+_{i-1/2,j})\right]$$
$$+\left[\hat{G}^+(\hat{W}^-_{i,j+1/2}) + \hat{G}^-(\hat{W}^+_{i,j+1/2})\right] - \left[\hat{G}^+(\hat{W}^-_{i,j-1/2}) + \hat{G}^-(\hat{W}^+_{i,j-1/2})\right] \tag{2.22}$$
$$+\sigma\hat{H}(\hat{W}_{i,j}) = 0$$

The usage of upwind schemes mainly depends on the flux splitting and differs by modifications made on the splitting procedure. The main upwind flux vector splitting schemes are Steger-Warming, and Van Leer are described below. The flux difference splitting scheme by Roe is also considered. Since direct flow solution is concerned, any of the schemes could be employed in the discretized residual and Jacobian calculation. Because of their simplicity, Steger-Warming flux Jacobians are often used with discretized residuals of any schemes. These are analyzed detail in the following chapters. However, it can be thought that for a better convergence, the residual and Jacobian calculations have to be consistent.

2.4.1 Steger-Warming Flux Vector Splitting

Since eigenvalue splitting is not unique and several splittings are possible, it's convenient to define a generalized flux vector in terms of eigenvalues. For steady, 2-D planar/axisymmetric Euler equations given in Equation (2.10) it can be written as:

$$\mathbb{F} = \frac{\rho}{2\gamma}\begin{bmatrix} 2(\gamma-1)\lambda_1 + \lambda_3 + \lambda_4 \\ 2(\gamma-1)\lambda_1 u + \lambda_3(u + c\tilde{k}_1) + \lambda_4(u - c\tilde{k}_1) \\ 2(\gamma-1)\lambda_1 v + \lambda_3(v + c\tilde{k}_2) + \lambda_4(v - c\tilde{k}_2) \\ (\gamma-1)\lambda_1(u^2+v^2) + \frac{\lambda_3}{2}((u + c\tilde{k}_1)^2 + (v + c\tilde{k}_2)^2) \\ + \frac{\lambda_4}{2}((u - c\tilde{k}_1)^2 + (v - c\tilde{k}_2)^2) + \frac{(3-\gamma)(\lambda_3+\lambda_4)c^2}{2(\gamma-1)} \end{bmatrix} \tag{2.23}$$

where the eigenvalues λ_i, the speed of sound c and the directional cosines $\tilde{k}_i$ are defined as:

$$\begin{aligned} \lambda_1 &= uk_1 + vk_2 \\ \lambda_3 &= \lambda_1 + c\sqrt{k_1^2 + k_2^2} \\ \lambda_4 &= \lambda_1 - c\sqrt{k_1^2 + k_2^2} \end{aligned} \tag{2.24}$$

$$c = \sqrt{\gamma(\gamma-1)\left(e_t - \frac{1}{2}(u^2+v^2)\right)} \tag{2.25}$$

$$\begin{aligned} \tilde{k}_1 &= \frac{k_1}{\sqrt{k_1^2 + k_2^2}} \\ \tilde{k}_2 &= \frac{k_2}{\sqrt{k_1^2 + k_2^2}} \end{aligned} \tag{2.26}$$

The ξ-directional flux vector $\hat{F}$ can be obtained from Equation (2.23) by replacing k_1 and k_2 with ξ_x and ξ_y, and the η-directional flux vector $\hat{G}$ can be obtained by replacing k_1 and k_2 with η_x and η_y, respectively.

Steger-Warming [17] proposed the following definitions for the splitting of the eigenvalues. The positive and negative flux vectors are obtained from Equation (2.23) by substituting all λ's by $\lambda^{+'}$s and $\lambda^{-'}$s respectively.

$$\lambda_i^{\pm} = \frac{\lambda_i \pm |\lambda_i|}{2} \tag{2.27}$$

However, this scheme has problems when eigenvalues are zero being at sonic points (*M=1*) and stagnation points (*M=0*), meaning that the positive and negative flux vectors are not continuously differentiable. Beside of taking extra cautions in the differentiation process, the oscillations around discontinuities may be reduced defining a small number $\in$.

$$\lambda_i^{\pm} = \frac{\lambda_i \pm \sqrt{\lambda_i^2 + \in^2}}{2} \tag{2.28}$$

2.4.2 Van-Leer Flux Vector Splitting

Van-Leer [18] introduced a different method for splitting fluxes, which is not based on wave speed splitting. The generalized flux vector is expressed as a function of contravariant Mach numbers.

- For supersonic flow ($|M| > 1$);

$$\begin{aligned} \mathbb{F}^+ &= \mathbb{F} \qquad \mathbb{F}^- = 0 \qquad \text{for } M \geq 1 \\ \mathbb{F}^+ &= 0 \qquad \mathbb{F}^- = \mathbb{F} \qquad \text{for } M < -1 \end{aligned} \tag{2.29}$$

- For subsonic flow ($|M| < 1$);

$$\mathbb{F}^{\pm} = \pm\rho c \frac{1}{4}(M \pm 1)^2(\tilde{k}_1 + \tilde{k}_2)\begin{bmatrix} 1 \\ \frac{1}{\gamma}(-\tilde{U} \pm 2c)\tilde{k}_1 + u \\ \frac{1}{\gamma}(-\tilde{U} \pm 2c)\tilde{k}_2 + v \\ \frac{\tilde{U}}{\gamma+1}\left(-\tilde{U} \pm 2c\right) + \frac{2a^2}{\gamma^2-1} + \frac{u^2+v^2}{2} \end{bmatrix} \tag{2.30}$$

where M is the contravariant Mach number in ξ or η direction, and the speed of sound c and the directional cosines $\tilde{k}_i$ are defined in Equations (2.25) and (2.26).

$$M = \frac{\tilde{U}}{c} \qquad \tilde{U} = \frac{uk_1 + vk_2}{\sqrt{k_1^2 + k_2^2}} = u\tilde{k}_1 + v\tilde{k}_2 \tag{2.31}$$

Again, the ξ-directional flux vector $\hat{F}$ can be obtained from Equation (2.30) by replacing k_1 and k_2 with ξ_x and ξ_y, and the η-directional flux vector $\hat{G}$ can be obtained by replacing k_1 and k_2 with η_x and η_y, respectively.

2.4.3 Roe Flux Difference Splitting

Roe [19] extended flux-splitting idea by considering the Riemann problem of discontinuous flow variables. He showed that with a new definition for the flux vectors Rankine-Hugonoit relationship holds exactly across a shock. Instead of splitting the flux vectors as in Equations (2.19) and (2.20), constant Jacobian matrices $\tilde{A}$ and $\tilde{B}$ are defined as a function of Roe averaged flow variables $\tilde{W}$

$$\begin{aligned} F &= \tilde{A}\tilde{W} & F(W^-) - F(W^+) &= \left|\tilde{A}\right|(W^- - W^+) \\ G &= \tilde{B}\tilde{W} & G(W^-) - G(W^+) &= \left|\tilde{B}\right|(W^- - W^+) \end{aligned} \tag{2.32}$$

$\tilde{A}$ and $\tilde{B}$ Jacobian matrices can be diagonalized as:

$$\begin{aligned} \tilde{A}&=\tilde{Q}_\Lambda \tilde{\Lambda}_A \tilde{Q}_\Lambda^{-1} & \tilde{B}&=\tilde{Q}_\Lambda \tilde{\Lambda}_B \tilde{Q}_\Lambda^{-1} \\ \left|\tilde{A}\right|&=\tilde{Q}_\Lambda \left|\tilde{\Lambda}_A\right| \tilde{Q}_\Lambda^{-1} & \left|\tilde{B}\right|&=\tilde{Q}_\Lambda \left|\tilde{\Lambda}_B\right| \tilde{Q}_\Lambda^{-1} \end{aligned} \tag{2.33}$$

where $\tilde{\Lambda}_A$, $\tilde{\Lambda}_B$ are the diagonal matrices including eigenvalues of $\tilde{A}$, $\tilde{B}$, and $\tilde{Q}_\Lambda$ is the matrix of corresponding right eigenvectors.

$$\begin{aligned} F(W^-)\text{-}F(W^+)&=\sum \tilde{Q}_\Lambda \left|\tilde{\lambda}_A\right| \tilde{Q}_\Lambda^{-1}(W^- - W^+) \\ G(W^-)\text{-}G(W^+)&=\sum \tilde{Q}_\Lambda \left|\tilde{\lambda}_B\right| \tilde{Q}_\Lambda^{-1}(W^- - W^+) \end{aligned} \tag{2.34}$$

Considering the positive and negative values of the eigenvalues, the interface flux vectors can be written in two ways:

$$\begin{aligned} \hat{F}_{i\pm 1/2,j} &= \hat{F}(\hat{W}^-_{i\pm 1/2,j}) + \sum_{\tilde{\lambda}_A<0} \tilde{Q}_\Lambda \left|\tilde{\lambda}_A\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i\pm 1/2,j} - \hat{W}^-_{i\pm 1/2,j}) \\ \hat{F}_{i\pm 1/2,j} &= \hat{F}(\hat{W}^+_{i\pm 1/2,j}) - \sum_{\tilde{\lambda}_A>0} \tilde{Q}_\Lambda \left|\tilde{\lambda}_A\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i\pm 1/2,j} - \hat{W}^-_{i\pm 1/2,j}) \\ \hat{G}_{i,j\pm 1/2} &= \hat{G}(\hat{W}^-_{i,j\pm 1/2}) + \sum_{\tilde{\lambda}_B<0} \tilde{Q}_\Lambda \left|\tilde{\lambda}_B\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i,j\pm 1/2} - \hat{W}^-_{i,j\pm 1/2}) \\ \hat{G}_{i,j\pm 1/2} &= \hat{G}(\hat{W}^+_{i,j\pm 1/2}) - \sum_{\tilde{\lambda}_B>0} \tilde{Q}_\Lambda \left|\tilde{\lambda}_B\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i,j\pm 1/2} - \hat{W}^-_{i,j\pm 1/2}) \end{aligned} \tag{2.35}$$

Finally, the interface flux vectors that are splitted in Equation (2.21) can be written as an average of both definitions above.

$$\begin{aligned} \hat{F}_{i\pm 1/2,j} &= \frac{1}{2}\left[\hat{F}(\hat{W}^+_{i\pm 1/2,j}) + \hat{F}(\hat{W}^-_{i\pm 1/2,j}) - \sum \tilde{Q}_\Lambda \left|\tilde{\lambda}_A\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i\pm 1/2,j} - \hat{W}^-_{i\pm 1/2,j})\right] \\ \hat{G}_{i,j\pm 1/2} &= \frac{1}{2}\left[\hat{G}(\hat{W}^+_{i,j\pm 1/2}) + \hat{G}(\hat{W}^-_{i,j\pm 1/2}) - \sum \tilde{Q}_\Lambda \left|\tilde{\lambda}_B\right| \tilde{Q}_\Lambda^{-1}(\hat{W}^+_{i,j\pm 1/2} - \hat{W}^-_{i,j\pm 1/2})\right] \end{aligned} \tag{2.36}$$

Although Roe's original notation includes R and L subscripts as the right and left states respectively, in this study, + and – subscripts are used as positive (right) and negative (left) states in order to be consistent with the flux vector splitting notation. Roe averaged density $\tilde{\rho}$ and any other Roe averaged flow variable $\tilde{W}$, which are required in the calculation of the constant Jacobian matrices, are defined as:

$$\tilde{\rho} = \sqrt{\rho^- \rho^+} \qquad \tilde{W} = \frac{\sqrt{\rho^-}W^- + \sqrt{\rho^+}W^+}{\sqrt{\rho^-} + \sqrt{\rho^+}} \tag{2.37}$$

2.5 Higher-Order Schemes with Limiters

As explained before, the flow variables are assumed constant for a computational cell given in Figure 2.1. However, the interface fluxes given in Equation (2.21) or (2.36) require the flow variables on cell faces, $\hat{W}^{\pm}_{i\pm 1/2,j}$ or $\hat{W}^{\pm}_{i,j\pm 1/2}$. A first order interpolation can be realized easily as follows:

$$\hat{W}^-_{i+1/2} = \hat{W}_i \qquad \hat{W}^+_{i+1/2} = \hat{W}_{i+1} \tag{2.38}$$

For higher order spatial discretizations, these conserved variables are determined from an upwind-biased interpolation of the primitive variables at cell centers. This is called MUSCL (Monotonic upstream-centered Scheme for Conservation Laws) [20] and general form can be written as:

$$\begin{aligned} \hat{W}^-_{i+1/2} &= \hat{W}_i + \left\{ \frac{\phi}{4}\left[(1-\kappa)\nabla + (1+\kappa)\Delta \right] \right\}_i \\ \hat{W}^+_{i+1/2} &= \hat{W}_{i+1} - \left\{ \frac{\phi}{4}\left[(1+\kappa)\nabla + (1-\kappa)\Delta \right] \right\}_{i+1} \end{aligned} \tag{2.39}$$

where the difference operators are:

$$\Delta_i = \hat{W}_{i+1} - \hat{W}_i \qquad \nabla_i = \hat{W}_i - \hat{W}_{i-1} \tag{2.40}$$

Order of the discretization and type of the differencing are determined by assigning different values to ϕ and κ. Considering $\phi = 0$ and $\kappa = 0$, a first order interpolation can be reached. With $\phi = 1$ and $\kappa = -1$ a straight second-order interpolation can be obtained. By changing the values of these parameters, the order of accuracy can be increased up to third order.

In higher order spatial discretizations, numerical oscillations are expected where large flow gradients occur. In order to control and reduce the order in these regions, flux limiters can be used. Actually, instead of a number ϕ can be used as a limiter, which is a function of differences through their ratio *r*. The ratio of differences can be defined as:

$$r_i = \frac{\Delta_i + \in}{\nabla_i + \in} \tag{2.41}$$

where $\in$ is a small number to prevent the division by zero in the zero gradient flow regions.

Then, Equation (2.39) can be rewritten as:

$$\begin{aligned} \hat{W}^{-}_{i+1/2} &= \hat{W}_i + \left\{ \frac{\phi(r)}{4} \left[(1-\kappa)\nabla + (1+\kappa)\Delta \right] \right\}_i \\ \hat{W}^{+}_{i+1/2} &= \hat{W}_{i+1} - \left\{ \frac{\phi(1/r)}{4} \left[(1+\kappa)\nabla + (1-\kappa)\Delta \right] \right\}_{i+1} \end{aligned} \tag{2.42}$$

There are several types of limiter functions, which may change the flux calculations. Some common limiters are:

- Min-Mod limiter:

$$\phi(r) = max(0, min(r, 1)) \tag{2.43}$$

- Superbee limiter:

$$\phi(r) = max(0, min(2r, 1), min(r, 2)) \tag{2.44}$$

- Van Leer limiter:

$$\phi(r) = \frac{r + |r|}{1 + |r|} \tag{2.45}$$

- Van Albada limiter:

$$\phi(r) = \frac{r + r^2}{1 + r^2} \tag{2.46}$$

In the regions of small gradient flows, the value of ϕ reaches one and actually uses no limiter. On the contrary, in the regions of very large gradient flows, its value goes to zero reducing the interpolation to first order. Van Albada limiter will be used in this study.

2.6 Boundary Conditions

According to the grid geometry and nature of the problem, several boundary conditions can be defined for the flow model. Both the selections of the appropriate boundary conditions and their successful implementation to the problem are very important. Four types of boundary conditions; inflow-outflow, symmetry and wall are considered. They are implemented to the flow by ghost cells

2.6.1 Inflow BC

Supersonic flow is the simplest case. Since all the information travels from outside into the computational domain, no information travels from inside across the boundary. In other words, the inflow boundary condition does not require to specify any variable from interior cells, all of the flow variables are set to the free-stream values in the ghost cells.

However for subsonic case, information can propagate from inside across the inflow boundary. Thus, one of the four flow variables has to be specified from inside. If Riemann invariants are used here, it is assumed that the flow is isentropic at the inlet boundary. In order to eliminate entropy generation, the inlet region should be far away from the geometry. At the far field boundary, the density and pressure are calculated from the speed of sound and entropy.

2.6.2 Outflow BC

The outflow boundary conditions are similar to those in inflow boundary. Supersonic flow is again the simplest case. In this case no information can travel upstream of the boundary, so all of the flow variables are extrapolated from the interior cells. Flow leaving the computational domain is unaffected by the outlaw boundary condition.

For subsonic flow, information can propagate again across the outflow boundary. Thus, three of the four flow variables are determined from inside flow, and one is specified by outflow boundary condition. A typical choice can be to specify the exit static pressure. If Riemann invariants are used here, it is assumed that the flow is isentropic at the outlet.

2.6.3 Symmetry BC

In the symmetry boundary condition, all the flow variables are extrapolated from the interior cells. If the symmetry line is parallel with the x-axis, flow variables are equal on both sides of the symmetry boundary, only the negative value of normal velocity component of the interior cell is assigned to the corresponding ghost cell.

2.6.4 Wall BC

Since inviscid flow is concerned, solid boundary can be treated very similar to the symmetry boundary. The density, tangential component of the velocity and total energy are equally extrapolated from the interior cells. According to the meaning of the wall, the normal component of the velocity is zero preserving no mass flux into or out of the wall. Only the tangential velocity component is maintained over the solid boundary. This is realized just as the same in the symmetry boundary, the normal velocity is defined to have equal magnitude but opposite sign across the boundary. Pressure can also be extrapolated from normal momentum equation.

CHAPTER 3

SOLUTION METHOD

3.1 Introduction

The solution method is closely related to the governing equations of the fluid flow. In iterative methods, whether the problem of interest is steady or unsteady, an unsteady set of equations is solved. For steady flows, the equilibrium solution is found by advancing the unsteady equation in time until a steady state is achieved. Although this iterative procedure can be efficient and robust, it is not clear that it is the best method for computing steady flows. With direct solution schemes, it is possible to compute the steady flow without an advance in time of an unsteady analysis. With direct methods, stability is increased since no time or any time-like variable appears in the set of governing equations or in the solution algorithm, and faster convergence is possible.

The direct solution technique employed in this study is the Newton's method. Newton's method is a widely used procedure for finding the solution of a system of non-linear algebraic equations for hundreds of years. However, the relatively recent development of powerful computers has now made the procedure practical for extremely large systems of equations.

In this study, the main objective is to develop a direct flow solver and analyze the effects of Jacobian calculation on the direct flow solutions. Several computational experiments are done related to accuracy and convergence of the

direct flow solution. Since the full matrix direct solvers require the solution of a huge Jacobian matrix even for a small computational domain, a simple flow model that retains the requirements is constructed. This simple model includes 2-D planar/axisymmetric Euler equations as governing equations. Since the discrete residual Jacobian matrix is a function of flux Jacobians, the calculation of these Jacobians is the main concern.

The first way to obtain the flux and residual Jacobians is analytically. As the discretization of the governing equations become more complex, derivation of analytical flux Jacobians becomes more difficult; even there may be some terms that cannot be differentiated analytically. Then, the best alternative is to compute the flux and residual Jacobians numerically as accurate as possible. Since numerical flux Jacobians are calculated by finite differencing the flux vectors, the perturbation magnitude used comes out as the most important parameter to obtain better accuracy. After the accurate calculation of flux Jacobians, the correct formation of the Jacobian matrix is also very important. In order to solve the Jacobian matrix efficiently several modifications and strategies can be employed to the solution method. Freezing the matrix after appropriate number of iterations has significant effect on the rapid convergence of the flow. Adding a time-like diagonal term or limiting the change in the solution for a number of iterations can be very important for stable convergence from poor initial conditions.

3.2 Newton's Method

The system of non-linear equations of the discretized governing equations can be written in the form:

$$\hat{R}(\hat{W}) = 0 \tag{3.1}$$

where $\hat{R}$ is the residual vector of the system, $\hat{W}$ is the flow variable vector.

Then, the general Newton's method is:

$$\left(\frac{\partial \hat{R}}{\partial \hat{W}}\right)^n \Delta\hat{W}^n = -R(\hat{W}^n) \tag{3.2}$$

The increment $\Delta\hat{W}$ at the nth iteration is found by solving the above system. The new values of flow variable vector $\hat{W}$ at the (n+1)th iteration is given by:

$$\hat{W}^{n+1} = \hat{W}^n + \Delta\hat{W}^n \tag{3.3}$$

3.3 Evaluation of the Jacobians

In direct flow solver, the Jacobian matrix has to be evaluated. The Jacobian matrix elements are the residual Jacobians that are functions of flux Jacobians. To investigate the accuracy of the numerical Jacobians both the numerical and analytical flux and residual Jacobians are calculated. In the study, analytical Jacobians are derived for first-order Steger Warming discretization. However, flux vectors are discretized by first/second order Steger-Warming/Van Leer/Roe schemes and the numerical Jacobians are calculated numerically for all these cases.

3.3.1 Analytical Jacobian Derivation

It is obvious that the calculation of flux Jacobians by manual differentiation is time-consuming and likely to be erroneous. However, for Euler fluxes with Steger-Warming flux-splitting discretization, this could be a bit feasible and easy. Thus, the analytical flux Jacobians are calculated by hand. The derivation procedure is given in Appendix A.

The residual Jacobians are functions of these flux Jacobians and directly found by summation of them. Considering Equation (2.22) and (2.38), first-order Steger-Warming discretized residual form of the steady, 2-D planar/axisymmetric Euler equations is:

$$\begin{aligned}&\left[\hat{F}^{+}(\hat{W}_{i,j})+\hat{F}^{-}(\hat{W}_{i+1,j})\right]-\left[\hat{F}^{+}(\hat{W}_{i-1,j})+\hat{F}^{-}(\hat{W}_{i,j})\right]\\&+\left[\hat{G}^{+}(\hat{W}_{i,j})+\hat{G}^{-}(\hat{W}_{i,j+1})\right]-\left[\hat{G}^{+}(\hat{W}_{i,j-1})+\hat{G}^{-}(\hat{W}_{i,j})\right]\\&+\sigma\hat{H}(\hat{W}_{i,j})=\hat{R}(\hat{W}_{i,j})\end{aligned} \tag{3.4}$$

Since the residual is first-order discretized, it is only a function of 5-point stencil, the discretized residual Jacobians are as follows:

$$\begin{aligned}&\frac{\partial\hat{R}_{i,j}}{\partial\hat{W}_{i,j}}=\hat{A}^{+}_{i,j}-\hat{A}^{-}_{i,j}+\hat{B}^{+}_{i,j}-\hat{B}^{-}_{i,j}\\&\frac{\partial\hat{R}_{i,j}}{\partial\hat{W}_{i+1,j}}=\hat{A}^{-}_{i+1,j}\qquad\frac{\partial\hat{R}_{i,j}}{\partial\hat{W}_{i,j+1}}=\hat{B}^{-}_{i,j+1}\\&\frac{\partial\hat{R}_{i,j}}{\partial\hat{W}_{i-1,j}}=-\hat{A}^{+}_{i-1,j}\qquad\frac{\partial\hat{R}_{i,j}}{\partial\hat{W}_{i,j-1}}=-\hat{B}^{+}_{i,j-1}\end{aligned} \tag{3.5}$$

The most critical part in the flux Jacobian calculation was taking the derivative of positive or negative eigenvalues with respect to flow variables, since they include absolute values of actual eigenvalues and this directly changes the flux vector equation. As given in the Appendix A, the derivative of $\lambda_i^{\pm}$ with respect to $\hat{W}_j$ can be found as:

$$\begin{aligned}&\lambda_i^{\pm}=\frac{\lambda_i\pm|\lambda_i|}{2}\\&\frac{\partial\lambda_i^{\pm}}{\partial\hat{W}_j}=\frac{\partial\lambda_i}{\partial\hat{W}_j}\cdot\left(\frac{1\pm sign(\lambda_i)}{2}\right)\end{aligned} \tag{3.6}$$

Note that the eigenvalues on the right are the actual eigenvalues without splitting. Although it seems very easy, during the coding this part has to be made very carefully.

Instead of taking the derivatives by hand, the flux Jacobians can be obtained from a symbolic manipulation system like MACSYMA. However, Orkwis [4] stated that, since MACSYMA cannot recognize repeatable patterns and is incapable of common simplifications, several simplifications and manipulations have to be considered for a desired result. It can be said that although exact Jacobians can be obtained for this case, it will be very hard to obtain this result for more complex discretized flow equations. Thus, investigation of numerical differentiation can be a good choice.

3.3.2 Numerical Jacobian Calculation

The flux Jacobians can be calculated numerically by one-sided derivative of numerical flux vector $\mathbb{F}_i(\hat{W})$ with respect to flow variable vector $\hat{W}_j$, using a small number being the finite-difference perturbation magnitude ε:

$$\frac{\partial \mathbb{F}_i}{\partial \hat{W}_j} = \frac{\mathbb{F}_i(\hat{W} + \varepsilon \cdot e_j) - \mathbb{F}_i(\hat{W})}{\varepsilon} \tag{3.7}$$

Here e_j is the j^{th} canonical vector. This value implies that the perturbation magnitude is closely related to the flow variable vector. The numerical residual Jacobians can also be calculated in the same manner. Instead of summing up the calculated numerical flux Jacobians, perturbing the residual $R(\hat{W})$ and taking the difference with the original one seems more reasonable.

$$\frac{\partial \hat{R}_i}{\partial \hat{W}_j} = \frac{R_i(\hat{W} + \varepsilon \cdot e_j) - R_i(\hat{W})}{\varepsilon} \tag{3.8}$$

The value of ε does not have to be positive necessarily. It is obvious that with employing a positive ε the derivative will be forward differenced, while with a negative ε backward differenced derivative could be obtained. The choice of the sign of the finite-difference perturbation magnitude can be very important when the value of the perturbed flow variable is very close to even smaller than ε:

In numerical Jacobian calculation, the same flux vector is used for both the original and perturbed flow variables. In Steger-Warming scheme, one of the most critical parts was being the flux vectors non-differentiable where the eigenvalues may change sign. With the perturbed flow variables, eigenvalues may change sign and cause to use a different flux vector. In order to prevent this, the effect of the perturbation on the sign of the eigenvalues must be checked, and the choice of backward or forward differencing has to be made carefully.

For better accuracy of the numerical Jacobians, the errors should be minimized with a good choice of ε. A detailed error analysis for the flux Jacobians to obtain an optimum finite-difference perturbation magnitude is presented in the next chapter.

As mentioned before, the numerical Jacobians require only the coding of flux discretization that will output the flux vector for both the original and perturbed flow variable vectors. This reuse of the same code for the calculation of flux vector is one of the big advantages of numerical approach. Moreover, for cases in which analytical Jacobians are impossible or too difficult to obtain, numerical approximation still brings a solution. With the computation of only the flux vector, higher-order discretizations can be considered easily.

Using the original flow variables, the flux vector and the residual is calculated for the whole domain. Since the same flux vector is used for the perturbed flow variables, the flux and residual values that are not affected from the perturbation are also calculated. This is an unnecessary time consuming process. In order to reduce time, flux vectors with perturbed flow variables are computed for only neighbouring cells. By this way, numerical Jacobian evaluation method becomes nearly as fast as the analytical method

3.4 Matrix Structure

The Jacobian matrix requires partial derivatives of every residual equation with respect to every flow variable. Fortunately, most of these derivatives are zero because of the fact that the discretized residual equations only depend on local variables. For better efficiency, only the non-zero elements of the Jacobian matrix have to be computed and stored. However, direct full matrix solvers require the whole matrix to be constructed, which limits the Newton's method. The storage and factorization costs of this type of matrix structure as if it were full would are prohibitively expensive for large problems. Thus, only storing the non-zero elements of the Jacobian matrix and employing a sparse matrix solver is the most required strategy for best efficiency.

The Jacobian matrix of the complete system is square, with dimensions equal to the total number of flow variables in the system. Considering the test case, 15° ramp geometry, the computations are realized with a 33x25 grid, meaning 34x26 cells. Since there are 4 flow variables at each cell, the total number of variables is 3536 and the Jacobian matrix has $3536^2 \cong 12.5$ million elements. For first-order upwind discretizations, a 5-point stencil is required, which produces a block tridiagonal matrix made up of 5 4x4-block bands as given in Figures 3.1 and 3.2. And in second-order discretizations 9-point stencil is employed, which produces a block tridiagonal matrix made up of 9 4x4-block bands as given in Figures 3.3 and

3.4. Thus, the elements of the Jacobian matrix except this block bands and the boundary conditions, are zero.

Although a full matrix solver using LU factorization [21] is tried to employ in the first stages of the study, due to storage and factorization costs sparse matrix solution is necessary. Thus, the sparse matrix solver package UMFPACK [22] is used in this study.

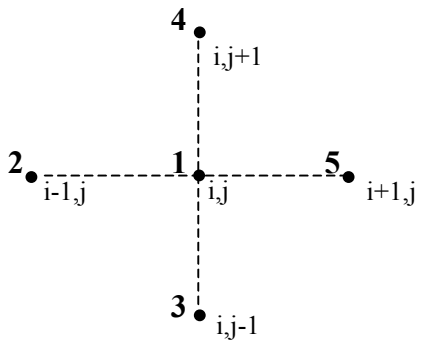


Figure 3.1 5-point stencil

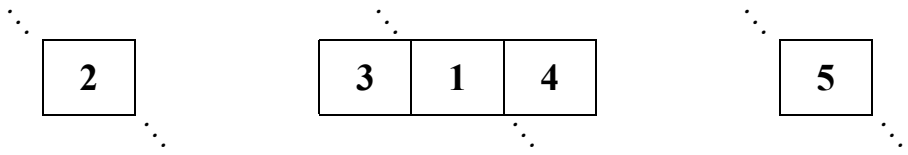


Figure 3.2 Matrix structure from 5-point stencil

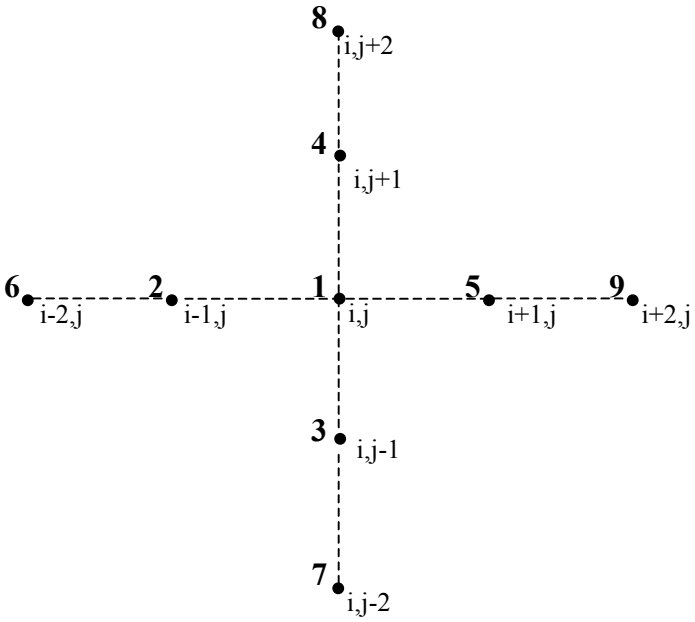


Figure 3.3 9-point stencil

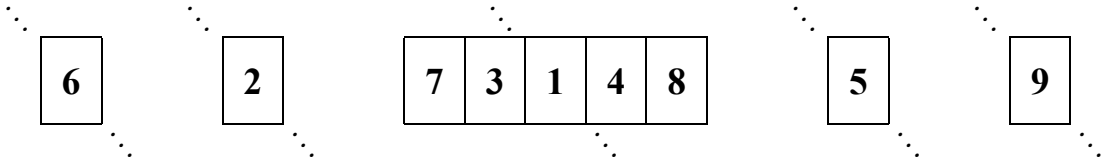


Figure 3.4 Matrix structure from 9-point stencil

3.5 Matrix Solution Strategies

Although, most of the elements of the Jacobian matrix are zero, conventional direct solvers require the whole matrix to be constructed to factorize or invert it. To handle this expensive operation, several strategies can be employed separately. Actually, inverting process is very hard in case of a very large Jacobian matrix. Therefore, most of the methods solve the system without a direct inversion of the matrix. One of the most common one is factorizing by LU decomposition. Also a combination of direct/iterative inversion routines can be employed to invert large sparse matrices like Orkwis [4]. Freezing the Jacobian matrix and starting with a good initial guess are also very important for an efficient convergence.

In this study, UMFPACK sparse matrix solver package [22] is used in order to solve Jacobian matrix. UMFPACK "Unsymmetric-pattern Multifrontal Package" is mainly based on converting the full matrix into sparse storage mode and factorizing it using a sequence of small dense frontal matrices by LU factorization. Usage of a sparse matrix solver increased the efficiency of the flow solver greatly. Moreover, strategies as frozen Jacobian, and better initial guess are considered in this study.

3.5.1 Frozen Jacobian

In the general Newton's method, the Jacobian matrix has to be recomputed and refactored with each iteration, since the matrix is a non-linear function of the changing flow variables. However, great gains in efficiency can be obtained by only computing and factoring the Jacobian matrix once, and using this frozen Jacobian matrix for all subsequent iterations. Since this is an approximation to the true system Jacobian, at each iteration the same convergence could not be reached. Actually, this is not very important since several iterations can be realized using the frozen Jacobian in the time required computing and factorizing a new one.

Nevertheless, freezing the Jacobian after some considered convergence is achieved makes the strategy more effective and reliable. Because Newton's Method has a tendency to diverge in the early stages of iterations. After this period is overcomed, rapid convergence is observed. As the divergence risk disappears in those stages, freezing becomes safe. As an example, in our test case, flow solutions are assumed to be converged when L_∞-norm of the density residual becomes 10^{-14} in double computer precision, and 10^{-5} in single computer precision. And when the residual drops to values around 10^{-2}-10^{-4} freezing the Jacobian matrix becomes feasible and improves the convergence time. A freezing before that value may cause the Newton's Method diverge. Thus, the time when to apply freezing is very important.

3.5.2 Initial Guess

The general Newton's method requires a good initial guess for convergence. This is one of the drawbacks of the method. The method fails in case of a poor initial guess. In this study, flow variables are initialized with their free-stream values. Although it may a poor initial guess, the results given in this study are obtained using this initialization.

Several ideas are available to modify the Newton's method to handle with poor initial conditions. One of them is a time-like term addition to the Jacobian matrix diagonal [3]. The modified Newton's method becomes:

$$\left(\frac{1}{\Delta t}[I] + \frac{\partial \hat{R}}{\partial \hat{W}} \right)^n \Delta \hat{W}^n = -R(\hat{W}^n) \tag{3.9}$$

As $\Delta t \rightarrow \infty$, the original Newton's method is achieved. Choosing a small initial value Δt^0, and gradually increasing its value according the L_2-norm of the residuals

will increase the stability of the Newton's method in poor initial conditions. A new value of Δt can be obtained from:

$$\Delta t^n = \Delta t^0 \frac{\left\| R(\hat{W}^0) \right\|_2}{\left\| R(\hat{W}^n) \right\|_2} \tag{3.10}$$

where L_2-norm is defined as:

$$\left\| X \right\|_2 = \sqrt{\sum_{i=1}^{n} \left| X_i \right|^2} \qquad X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \tag{3.11}$$

Considering this modification whether as a time derivative of $\hat{W}$ making the method integrating in a time like fashion, or just an addition to increase the strength of the matrix diagonal, iteration from poor initial conditions are improved. But, since this method converges linearly until Δt gets very large, the computation time is increased.

The time-like diagonal term addition is employed in this study considering that initialization from free-stream can be a poor guess, and the solver may not work without such an improvement. L_2-norm is computed including all the residuals in the domain. One important point was that this modification is required only in the first stages of the iterations. After certain number of iterations, the conditions are not poor anymore. So, instead of waiting Δt to get larger and larger, the diagonal term in terms of Δt may not be added anymore to the Jacobian matrix. This sudden withdrawal may cause oscillations but if the right time was chosen very faster convergence can be achieved. For the test case, $\Delta t^0 = 1$ is used in all calculations for 2-D planar calculations. And when Δt becomes around 1.5, this diagonal term is taken out from the system. This improves the number of iterations to convergence and CPU time greatly. For 2-D axisymmetric calculations, there are already axisymmetric terms

added to the diagonal of the Jacobian matrix. Thus, there is no need for such a modification to improve the initial guess. In axisymmetric calculation, no diagonal term is added to the Jacobian matrix

Another modification that may make the use of poor initial guess possible is limiting the change in the solution to reduce the residual [2]. In case of the residual not decreasing, the entire flow variable vector is reduced by some factor α.

$$\hat{W}^{n+1} = \alpha \Delta \hat{W}^{n} + \hat{W}^{n} \tag{3.12}$$

However, the correct value for α for a residual reduction can only be determined by trial. According to Orkwis [4], this idea may not allow the formation of shock waves in case, thus may not be practical as the diagonal term modification. In this study, only time-like diagonal term addition is employed.

CHAPTER 4

ACCURACY OF NUMERICAL JACOBIANS

4.1 Introduction

Despite of increasing stability, direct solution of the discretized governing equations requires the calculation of Jacobian matrix that is a function of flux Jacobians. The preferable way to obtain these Jacobians is analytical methods. As the discretization of the governing equations become more complex, derivation of analytical Jacobians becomes more difficult. Then, the best alternative is to compute the Jacobians numerically as accurate as possible. Since numerical flux Jacobians are calculated by finite differencing the flux vectors, the perturbation magnitude used comes out as the most important parameter to obtain better accuracy.

Several flux calculation schemes like central differencing, Steger-Warming, Van Leer or Roe upwind differencing schemes could be employed in the governing equation discretization and Jacobian calculation. Because of their simplicity, Steger-Warming flux Jacobians are often used even with discretized residuals of other schemes. However, as also shown in this study, to get best convergence the flux calculation in residual and Jacobian calculation must be consistent.

Although the calculation of flux Jacobians by manual differentiation seems impractical, the simplicity of Steger-Warming flux discretization makes it possible. Thus, the analytical flux Jacobians are calculated by hand. Several symbolic mathematics software packages can be used for analytical differentiation. As in some

of the studies in the literature, analytical Jacobians can be obtained from the symbolic manipulation expert system MACSYMA.

In this study, the accuracy of the numerical Jacobians is investigated by comparing the values of numerical and analytical Jacobians. Effect of perturbation magnitude ε, on the accuracy of the numerical Jacobians is studied. For different ε values, maximum and average errors between the analytical and numerical flux and residual Jacobian values are analyzed. The effect of backward and forward differencing is investigated. To obtain an optimum perturbation magnitude, a detailed error analysis is performed for the numerical Jacobian calculation. Computations are performed with both single and double precision in order to see the effect of computer precision on the accuracy. Also, a method is investigated to find the optimum perturbation magnitude in single precision. The results for supersonic ramp geometry are presented in Section 4.5.

4.2 Error Analysis

In numerical Jacobian calculation, mainly two types of errors occur. These are truncation and condition errors. Truncation error is due to neglected terms in the Taylor's series expansion. Truncation error increases with the perturbation magnitude. Condition error is associated with numerical noise and caused by loss of numerical precision. This error may result from computer round-off error. Condition error in finite difference derivatives generally increases with decreasing the perturbation magnitudes.

4.2.1 Truncation Error

As in the case of numerical flux Jacobian calculation, the first derivative of a function $f(x)$ can be approximated as a forward difference for some perturbation magnitude $\varepsilon<1$:

$$\frac{\Delta f(x)}{\Delta x} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon} \tag{4.1}$$

However, since the Taylor series expansion includes higher order terms, the actual derivative is:

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon} - \frac{\partial^2 f(x)}{\partial x^2} \cdot \frac{\varepsilon}{2} - \frac{\partial^3 f(x)}{\partial x^3} \cdot \frac{\varepsilon^2}{6} - \ldots \tag{4.2}$$

Therefore, the truncation error consists of the neglected terms in the Taylor series:

$$E_T(\varepsilon) = \frac{\Delta f(x)}{\Delta x} - \frac{\partial f(x)}{\partial x} = \frac{\partial^2 f(x)}{\partial x^2} \cdot \frac{\varepsilon}{2} + \frac{\partial^3 f(x)}{\partial x^3} \cdot \frac{\varepsilon^2}{6} + \ldots = \frac{\partial^2 f(\zeta)}{\partial x^2} \cdot \frac{\varepsilon}{2} \tag{4.3}$$

where ζ=[x, $x+\varepsilon$].

The above equation shows that truncation error increases with the perturbation magnitude.

4.2.2 Condition Error

Due to computer precision, the exact value of a function $f(x)$ and its computed value $\tilde{f}(x)$ can be different due to round-off error $E(x)$:

$$\begin{aligned} \tilde{f}(x) &= f(x) + E(x) \\ \tilde{f}(x+\varepsilon) &= f(x+\varepsilon) + E(x+\varepsilon) \end{aligned} \tag{4.4}$$

The first derivative of the function $\tilde{f}(x)$ can be computed as:

$$\frac{\Delta\tilde{f}(x)}{\Delta x} = \frac{\tilde{f}(x+\varepsilon) - \tilde{f}(x)}{\varepsilon} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon} + \frac{E(x+\varepsilon) - E(x)}{\varepsilon} \tag{4.5}$$

So, the condition error comes out to be:

$$E_C(\Delta x) = \frac{\Delta\tilde{f}(x)}{\Delta x} - \frac{\Delta f(x)}{\Delta x} = \frac{E(x+\varepsilon) - E(x)}{\varepsilon} \tag{4.6}$$

Considering an error bound $E_R = \max\{\ |E(x)|\ ,\ |E(x+\Delta x)|\ \}$, the maximum of condition error can be written as:

$$E_C(\varepsilon) = \frac{2 \cdot E_R}{\varepsilon} \tag{4.7}$$

4.2.3 Total Error

The total error is simply the sum of the truncation and condition errors. Actually, rounding error also exists, but is not considered since it's negligible when compared to these errors.

$$E_{TOTAL}(\varepsilon) = E_C(\varepsilon) + E_T(\varepsilon) = \frac{2 \cdot E_R}{\varepsilon} + \frac{\partial^2 f(\zeta)}{\partial x^2} \cdot \frac{\varepsilon}{2} \tag{4.8}$$

It can be seen that the total error is highly dependent on perturbation magnitude. If the perturbation magnitude is too small, condition error dominates and if the perturbation magnitude is too large, truncation error becomes more important. So, there must an optimum value for the perturbation magnitude that gives a minimum error.

4.3 Effect Of Computer Precision

In this study, the accuracy between the analytical and numerical flux and residual Jacobians are analyzed for different ε values as explained before. All the analyses are made for both single and double computer precision in order to see the effect of computer precision on the accuracy. This is realized without changing the solver code, but using the optimization property of the Fortran compiler that makes the computations in double precision.

The precision is the property of the computer processor, and defined according to the machine epsilon, ε_M . This is the smallest number that the computer recognizes as being bigger than zero. Machine precision is dependent upon number of bytes used in the representation of a real number as well as the distribution of the bits associated with representing the mantissa verses the exponent. This distribution is usually a property of the hardware; however, software like optimizing compilers can extend the precision but at the price of speed. Generally speaking, a reasonable estimate of ε_M can be given as follows:

$$\varepsilon_M = \frac{1}{2^m} \qquad \text{such that} \qquad 1 + \varepsilon_M > 1 \tag{4.9}$$

where m is the number of possible highest bits in the binary representation of the mantissa.

All the calculations of this study are realized on 1.5GHz Pentium IV dual processor with openMosix cluster running on Linux 2.4. Although openMosix provides dynamic load balancing between cluster nodes and continuously attempts to optimize the resource allocation, this property may be more important in the use of parallel processing and in this study, the main concerns are processor power and the memory usage. The compiler used in the study is Lahey/Fujitsu Optimizing Fortran

95 Compiler “lf95” which has options that the code can be compiled in single or double precision.

Knowing the precision of the computer before realizing the computations is very important since the accuracy in terms of very small numbers are studied. The machine epsilon ε_M values of this compiler-computer configuration can be found according to Equation (4.9). Then, for single precision $\varepsilon_M \cong 3.0 \times 10^{-8}$, and for double precision $\varepsilon_M \cong 5.6 \times 10^{-17}$ values are reached. In order to realize feasible and reliable analyses, all the numerical values used in the computations have to be greater than these limits.

4.4 Optimum Perturbation Magnitude Analysis

The optimum perturbation magnitude value can be found nearly some special methods developed. One known method is looking at the ε value where the derivative of Equation (4.8) is zero:

$$\frac{\partial E_{TOTAL}(\varepsilon)}{\partial \varepsilon} = -\frac{2 \cdot E_R}{\varepsilon^2} + \frac{1}{2} \cdot \frac{\partial^2 f(\xi)}{\partial x^2} = 0 \tag{4.10}$$

$$\varepsilon_{OPT} = 2 \cdot \sqrt{\frac{E_R}{\frac{\partial^2 f(\xi)}{\partial x^2}}} \tag{4.11}$$

The above equation requires the calculation of second derivative and the bound of condition error. In many cases for the second derivative a finite difference relation has to be employed which also brings errors, and it’s not applicable at points where the second derivative is zero. Considering the second derivative in the order of one can also be a good approximation. The bound of condition error is actually the precision error and for the single precision case, it can be found by subtraction of

double and single precision calculations of the function. Another consideration can be taking the precision error equal to the machine epsilon for both single and double precision cases. Taking the second derivative as one Equation (4.11) becomes:

$$\varepsilon_{OPT} = 2 \cdot \sqrt{\varepsilon_M} \tag{4.12}$$

In this study, the optimization method explained above is employed. Optimum value for finite-difference perturbation magnitude is calculated using Equation (4.12) for both single and double precision. In addition, for single precision case, ε_{OPT} is found from averaged second derivative and precision error values for each flux vector. Here, the precision error is simply found by subtraction of double and single precision calculations of the flux values. The second derivative is calculated using a forward/backward finite-difference method using double precision. Although, this also brings errors, the difference in the optimization for each flux vector can be seen by this way.

In order to control the reliability of the optimization method, as a trial-error like procedure, flux Jacobians are computed and their effect on the accuracy of the numerical flux Jacobians are analyzed for several finite-difference perturbation magnitude values. Both the maximum error and the average error in the domain are calculated. As explained in the following section in detail, nearly optimum perturbation magnitude values are obtained minimizing each maximum and average error between analytical and numerical flux and residual Jacobians. Another consideration was the choice of backward or forward differencing in the calculation of the numerical Jacobians. This analysis using both positive and negative perturbation magnitudes would give information about this choice.

4.5 Test Case Results

The results are obtained for the test case being 15° supersonic ramp geometry as shown in Figure 4.1. A 33x25 grid is used. The inlet Mach number, M_∞ is 2.0. At the inlet and outlet flow is supersonic. Thus, at the inlet flow variables are initialized with their free stream values, while at the outlet they are equated to the inner values. In addition, wall and symmetry boundary conditions are employed. These geometry and flow conditions are suitable to include the effect of shocks on the errors in numerical Jacobians. In this part of the study, only the accuracy of numerical Jacobians is studied. Numerical and analytical Jacobians are calculated for an already converged solution and they are compared with each other. Analytical Jacobians are obtained for first-order Steger-Warming scheme. A detailed procedure of obtaining Jacobians numerically and analytically is given in Chapter 3.3.

For different finite-difference perturbation magnitude, ε values, changes of maximum and average errors in numerical Jacobians are analyzed. The numerical Jacobians are calculated using both forward and backward differencing to investigate their effects. All the study is realized for both single and double precision in order to see the effect of computer precision on the accuracy. For observing the change in all possible Jacobian terms, all four flux vectors $\hat{F}^+, \hat{F}^-, \hat{G}^+, \hat{G}^-$ and residuals for both planar and axisymmetric cases $\hat{R}_{pl}, \hat{R}_{ax}$ are considered.

Figure 4.2 and Figure 4.3 show the error contours in numerical flux Jacobian $\partial \hat{F}_2^+ / \partial \hat{W}_1$, similarly Figure 4.4 and Figure 4.5 show the flux Jacobian $\partial \hat{F}_2^+ / \partial \hat{W}_1$ contour itself calculated with single and double precision. This flux Jacobian is arbitrarily chosen in order to show the behaviour of the contours. It's seen that for higher perturbation magnitudes the error contours look like the Jacobian contours. This is the case where truncation error is dominant, and this similarity of the contours makes sense since this error is higher at high gradient regions. As the perturbation magnitude is decreased, the contour becomes more scrambled decreasing error.

However, with further decrease in perturbation magnitude, although scrambled contour shape is conserved, the errors start to increase. If the perturbation magnitude is too small, condition error dominates and the loss of precision increases the error.

As explained in Chapter 3.3.2, the value of ε does not have to be positive necessarily, both forward and backward differencing can be employed in the calculation of the numerical flux and residual Jacobians. When the value of the perturbed flow variable is very close to or even smaller than ε, this choice may be very important. In Steger-Warming scheme, when the eigenvalues are close to zero, with the perturbed flow variable eigenvalues may change sign and cause to use a different flux vector. As a control mechanism, the effect of the perturbation on the sign of the eigenvalues must be checked, and the appropriate differencing has to be chosen.

Figure 4.6 and Figure 4.7 show the effect of this control mechanism on total errors for the planar residual Jacobian, $\partial \hat{R}_{pl} / \partial \hat{W}$ in single and double precision. From Figure 4.6 showing the single precision case, it is observed that without this control, forward difference gives more accurate results compared to backward formulation in terms of average error. However, for the average and maximum errors, a decrease is observed with the application of control mechanism. Considering Figure 4.7 showing the double precision case, it is seen that the control mechanism has a great importance. Again, without control, forward difference seems to give better results compared to backward formulation, but all the errors are very large and look like constant without affecting from the perturbation magnitude. With the use of control mechanism, a large decrease in all errors is observed, even the maximum error decreases from the order of 10^{-1} to 10^{-7}. This shows that in all calculations, the usage of this control mechanism is necessary to realize an accurate study.

Looking at the general behaviour of all figures, it is obvious that with double precision the errors become very small; the maximum error is on the order of 10^{-8}, for the average error the order drops to 10^{-9}. With single precision, only an order of 10^{-3} can be achieved for maximum and average errors. Thus, the usage of double precision improves the accuracy significantly. Again, from the general view of the figures, it is observed that there is no significant difference between the forward and backward differencing in the calculation of the numerical Jacobians. Because the control mechanism employed changes the differencing procedure in case of a problem.

Figure 4.8 and Figure 4.9 show the effect of perturbation magnitude on maximum error in all numerical Jacobians. Since maximum errors are considered, the planar residual Jacobian $\partial \hat{R}_{pl} / \partial \hat{W}$ error follows up the path of maximum values of all flux Jacobian errors. The optimum value of ε can be seen that nearly 4-5x10^{-8} giving an error around 10^{-7} for double precision, and 6-7x10^{-4} giving an error in the order of 10^{-1} for single precision.

Figure 4.10 and Figure 4.11 show the effect of perturbation magnitude on average error in all numerical Jacobians. In this case average error for $\partial \hat{R}_{pl} / \partial \hat{W}$ looks like as the average of flux Jacobian errors as expected. The optimum value of ε can be observed again nearly 4-5x10^{-8} giving an error in the order of 10^{-9} for double precision, and 6.4-9.8x10^{-4} giving an error around 10^{-4} for single precision.

Figure 4.12 and Figure 4.13 show the effect of axisymmetry on average and maximum errors for the residual Jacobian in single and double precision. The axisymmetric residual Jacobians $\partial \hat{R}_{ax} / \partial \hat{W}$ only include some extra terms due axisymmetric source. Thus, it is not expected to have results much different from the planar case. As a matter of fact, all maximum and average errors in both double and single precision came out as pretty much the same for the axisymmetric case.

In Figure 4.8 through Figure 4.13, results show that the error is highly dependent on perturbation magnitude, and there's an optimum value that gives the minimum error. As explained before, if the perturbation magnitude is too small, condition error increases and if the perturbation magnitude is too large, truncation error grows up. In figures, there is nothing seen related to the errors for the flux Jacobian $\partial \hat{F}^- / \partial \hat{W}$. This was obvious since $\hat{F}^-$ is completely zero in our test case due to supersonic flow at all points.

In order to find the optimum value for the finite-difference perturbation magnitude, the optimization method given by Equation (4.11) is realized for single precision as explained in Section 4.3. In the method, to eliminate the errors caused by zero second derivatives, ε_{opt} value is calculated from the averaged second derivative and precision error values, which can be seen from Table 4.1. Considering the second derivative values in the order of 1 and the precision error equal to machine epsilon, the optimization method given by Equation (4.12) is also used for both single and double precision, and the results are given in Table 4.2. Optimum values come out in the order of 10^{-4} for single precision and in the order of 10^{-8} for double precision.

Table 4.1 ε_{opt} analysis using average values for single precision.

	Optimization Method (avg.)		
Flux	Precision Error (avg.)	Second Derivative (avg.)	ε_{opt} (avg.)
F^+	$13.5 \cdot 10^{-7}$	9.092	$7.7 \cdot 10^{-4}$
F^-	-	-	-
G^+	$5.0 \cdot 10^{-7}$	3.215	$7.9 \cdot 10^{-4}$
G^-	$4.6 \cdot 10^{-7}$	3.049	$7.7 \cdot 10^{-4}$

Table 4.2 ε_{opt} analysis using machine epsilon for single and double precision.

	Optimization Method (ε_M)		
Precision	Precision Error ($=\varepsilon_M$)	Second Derivative (=1)	ε_{opt} (ε_M)
Single	$3.0 \cdot 10^{-8}$	1.	$3.5 \cdot 10^{-4}$
Double	$5.6 \cdot 10^{-17}$	1.	$1.5 \cdot 10^{-8}$

Table 4.3 gives the optimum perturbation magnitude values for different flux Jacobians found by trial-error as shown in Figure 4.8 through Figure 4.11, and using the optimization method. For single precision, especially ε_{opt} values minimizing total average errors between analytical and forward-differenced numerical flux Jacobians in the trial-error procedure and the corresponding values for the optimization method are very close to each other. For double precision, the same optimum value is found for all flux Jacobians. Since $\hat{F}^-$ is completely zero for supersonic flow over our test case, no related errors can be seen in the table for its flux Jacobian.

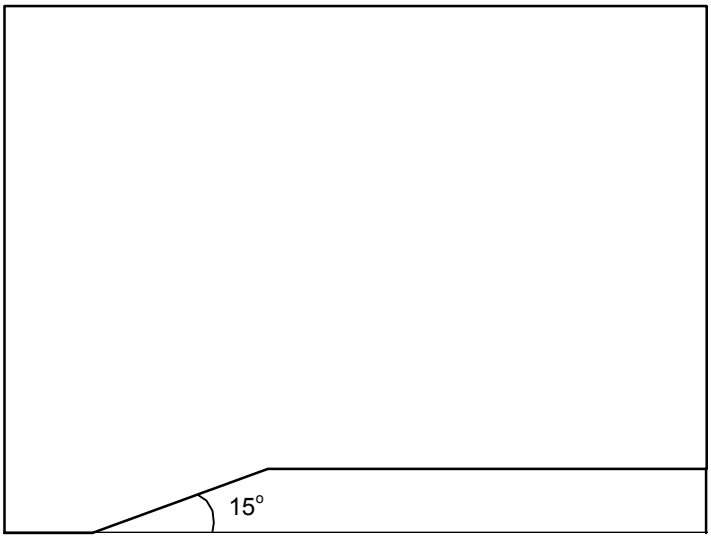
Table 4.3 Optimum perturbation magnitude ε_{opt} results for single precision.

	Trial-Error Procedure		Optimization Method	
Jacobian	ε_{opt} (max. error)	ε_{opt} (avg. error)	ε_{opt} (avg.)	ε_{opt} (ε_M)
F^+	$9.8 \cdot 10^{-4}$	$6.4 \cdot 10^{-4}$	$7.7 \cdot 10^{-4}$	$3.5 \cdot 10^{-4}$
F^-	-	-	-	
G^+	$1.5 \cdot 10^{-5}$	$6.4 \cdot 10^{-4}$	$7.9 \cdot 10^{-4}$	
G^-	$1.5 \cdot 10^{-5}$	$6.4 \cdot 10^{-4}$	$7.8 \cdot 10^{-4}$	

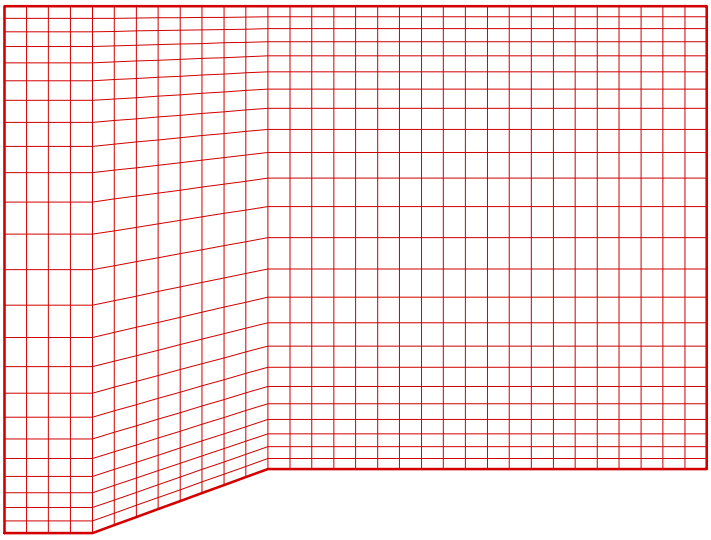
Table 4.4 Optimum perturbation magnitude ε_{opt} results for double precision.

	Trial-Error Procedure		Optimization Method
Jacobian	ε_{opt} (max. error)	ε_{opt} (avg. error)	ε_{opt} (ε_M)
F^+	$4.0 \cdot 10^{-8}$	$4.0 \cdot 10^{-8}$	$1.5 \cdot 10^{-8}$
F^-	-	-	
G^+	$4.0 \cdot 10^{-8}$	$4.0 \cdot 10^{-8}$	
G^-	$4.0 \cdot 10^{-8}$	$4.0 \cdot 10^{-8}$	

Although, it's an approximation, the optimization method is a good approach to find the optimum perturbation magnitude. Instead of spending time on trial-error procedure, employing this simple optimization method give very good results. As a result, for single precision $7 \cdot 10^{-4}$, and for double precision $4 \cdot 10^{-8}$ can be used as the optimum perturbation magnitude values for all flux Jacobians.

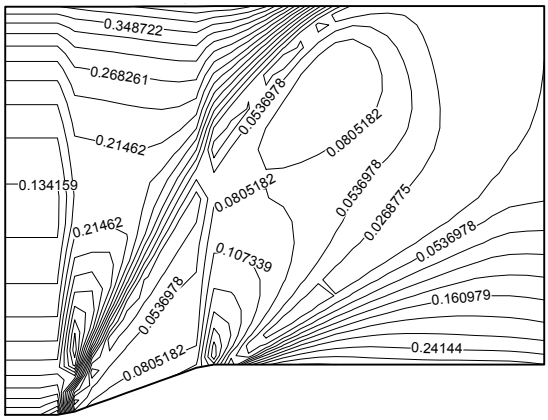


(a) General view



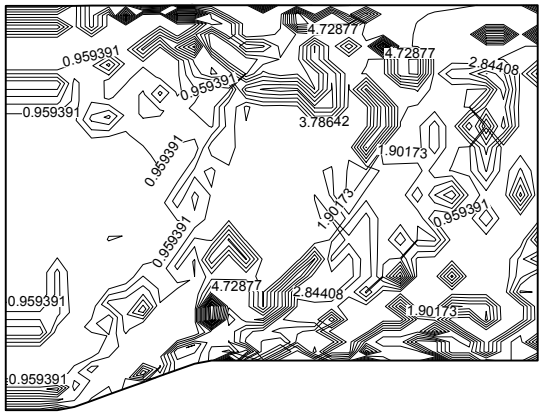
(b) 33x25 grid

Figure 4.1 15° ramp geometry



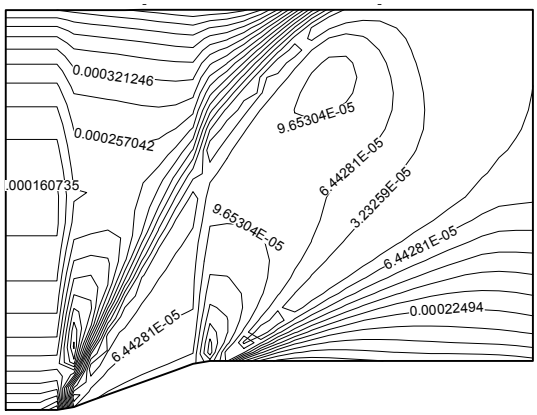
(a) $\varepsilon = 1 \times 10^{-1}$

(b) $\varepsilon = 7 \times 10^{-4}$

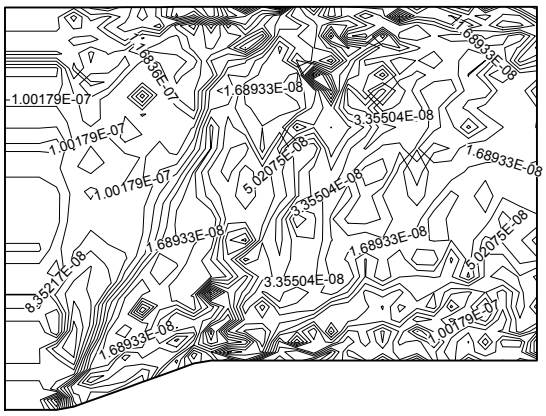


(c) $\varepsilon = 1 \times 10^{-7}$

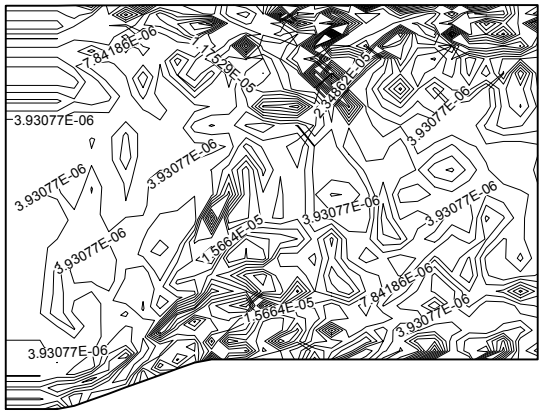
Figure 4.2 Effect of ε on $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian error contours for forward differencing in single precision.



(a) $\varepsilon = 1 \times 10^{-4}$

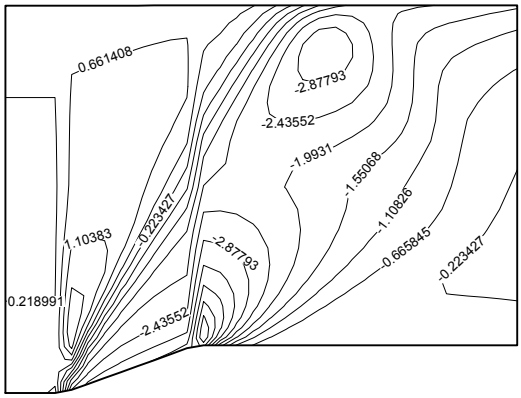


(b) $\varepsilon = 4 \times 10^{-8}$

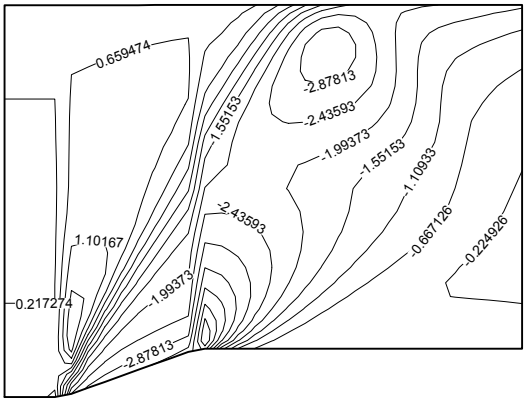


(c) $\varepsilon = 1 \times 10^{-10}$

Figure 4.3 Effect of ε on $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian error contours for forward differencing in double precision.

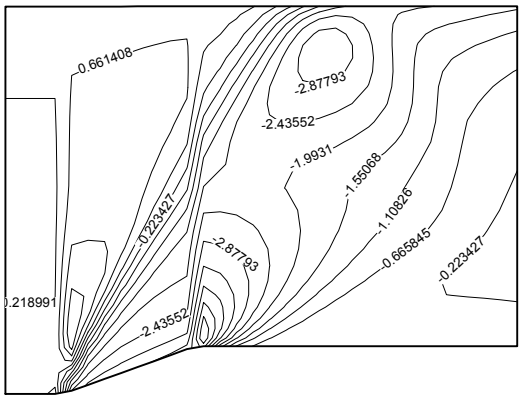


(a) Analytical Jacobian

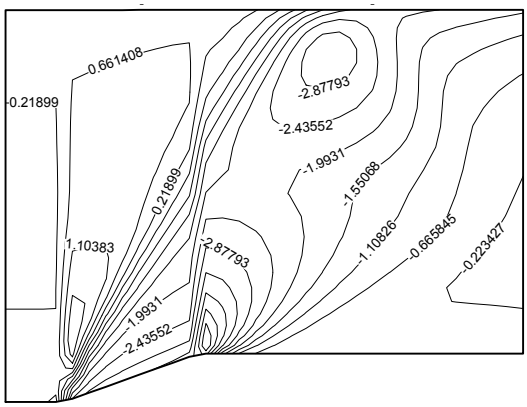


(b) Numerical Jacobian with $\varepsilon = 7 \times 10^{-4}$

Figure 4.4 $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian contours for forward differencing in single precision.

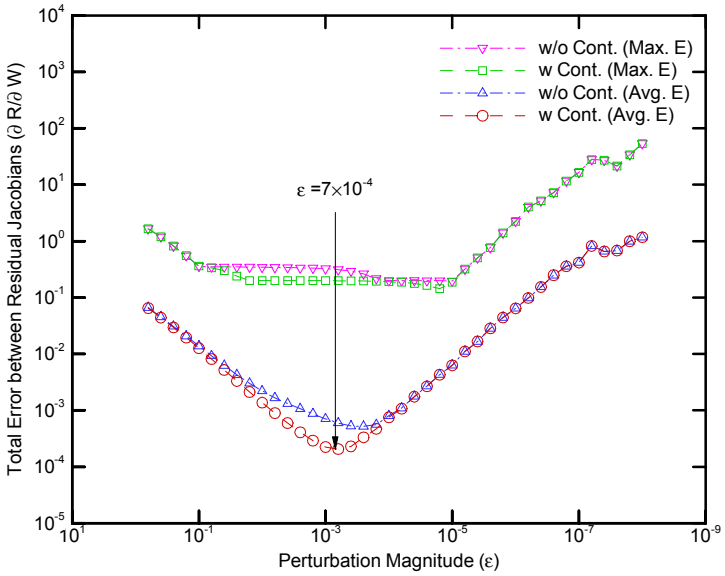


(a) Analytical Jacobian

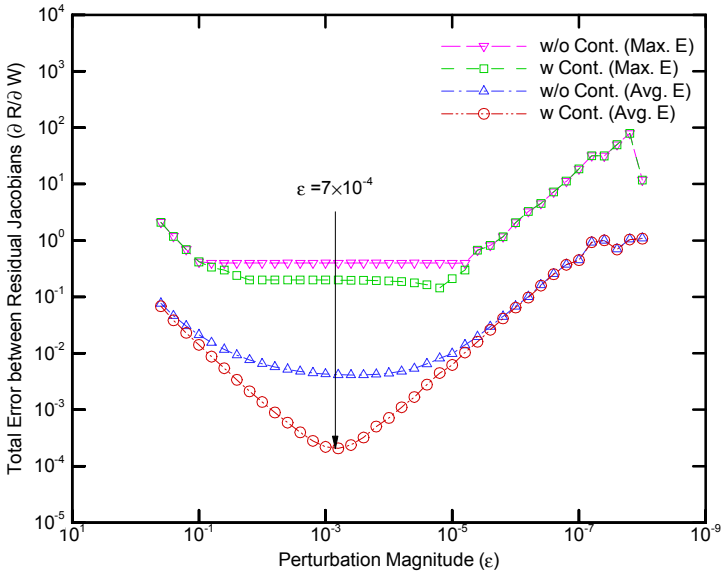


(b) Numerical Jacobian with $\varepsilon = 4 \times 10^{-8}$

Figure 4.5 $\partial \hat{F}_2^+ / \partial \hat{W}_1$ flux Jacobian contours for forward differencing in double precision.

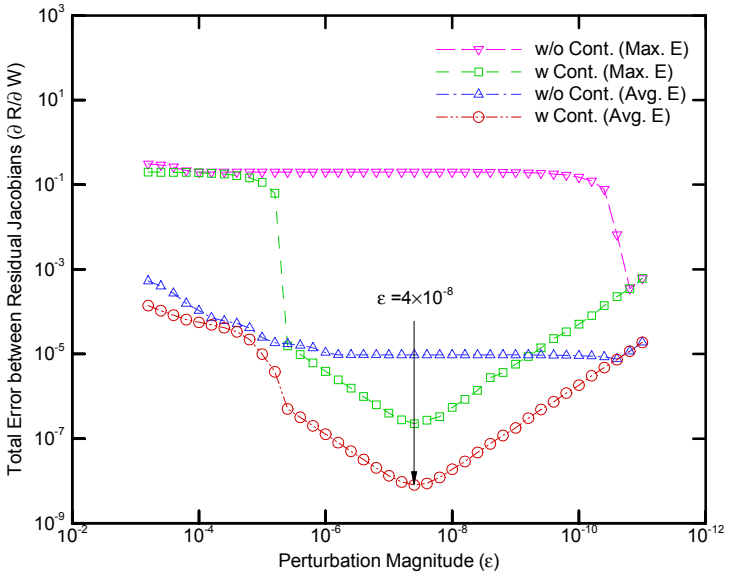


(a) Forward differencing

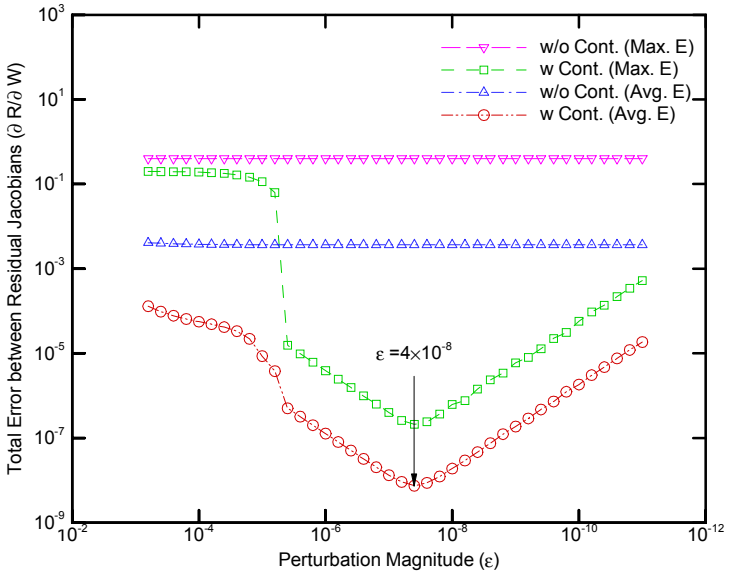


(b) Backward differencing

Figure 4.6 Effect of control on total max. and avg. errors for $\partial \hat{R}_{pl} / \partial \hat{W}$ in single precision.

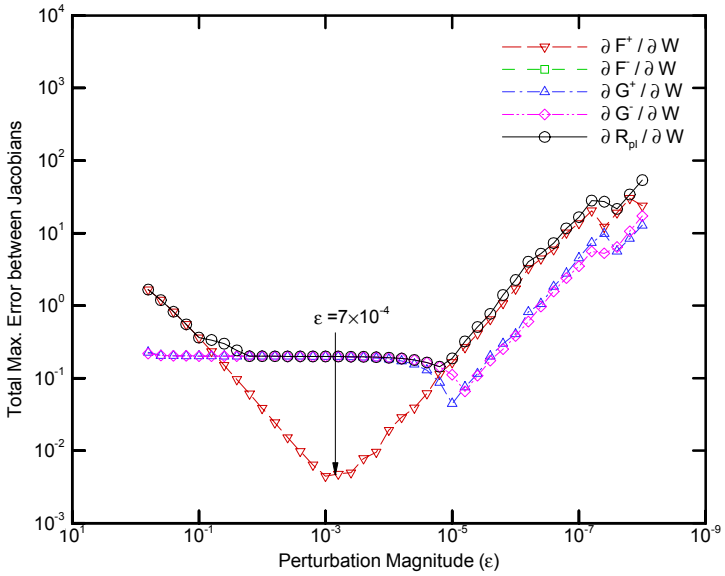


(a) Forward differencing

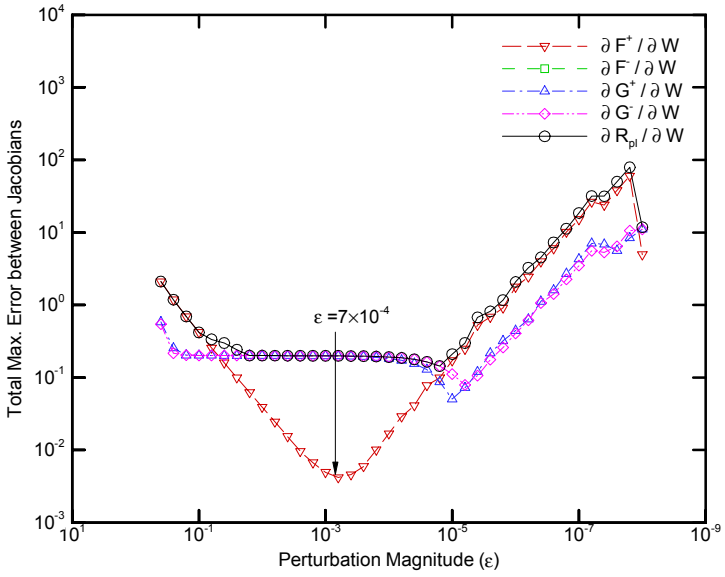


(b) Backward differencing

Figure 4.7 Effect of control on total max. and avg. errors for $\partial \hat{R}_{pl} / \partial \hat{W}$ in double precision.

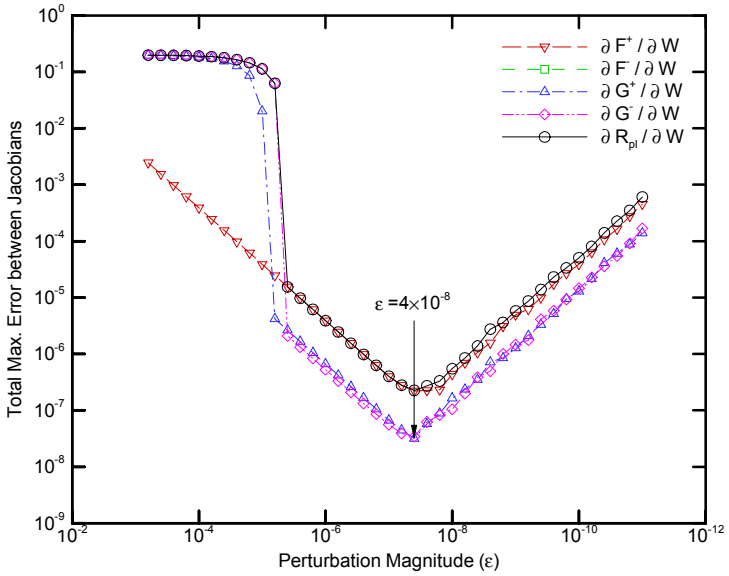


(a) Forward differencing

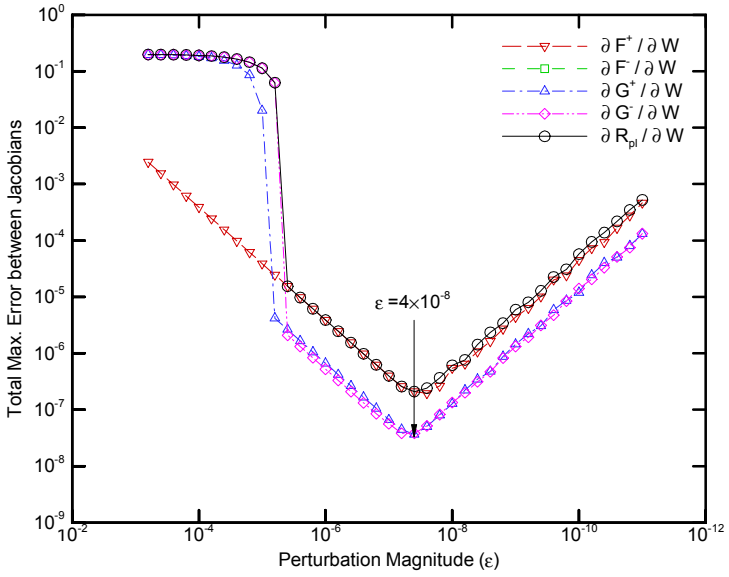


(b) Backward differencing

Figure 4.8 Effect of ε on total max. error for all Jacobians in single precision.

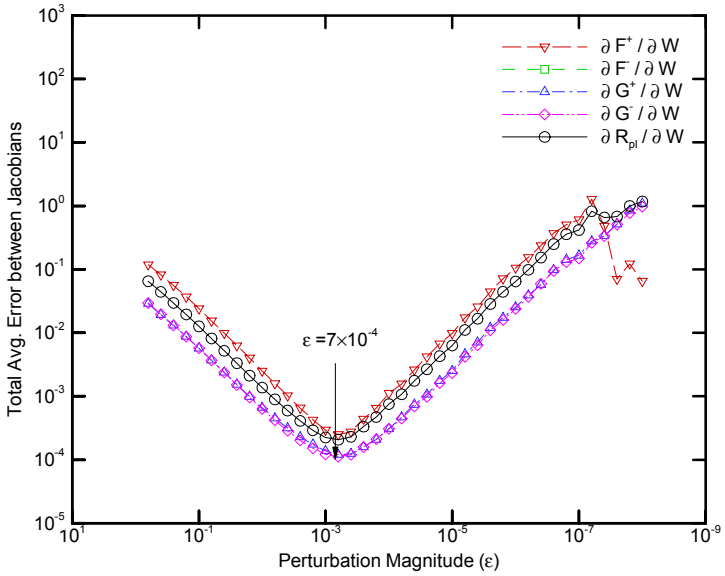


(a) Forward differencing

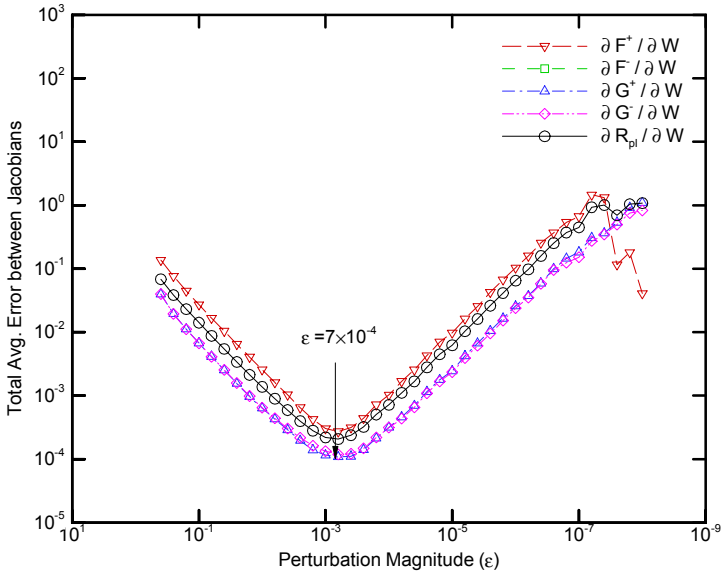


(b) Backward differencing

Figure 4.9 Effect of ε on total max. error for all Jacobians in double precision.

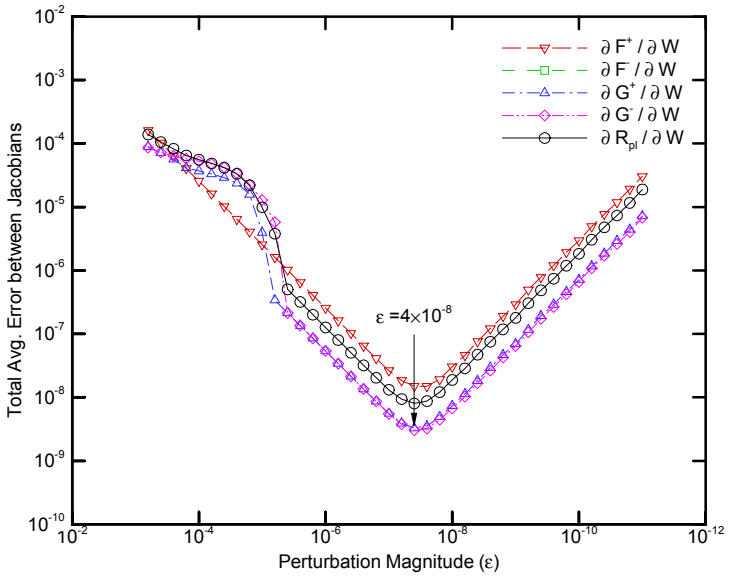


(a) Forward differencing

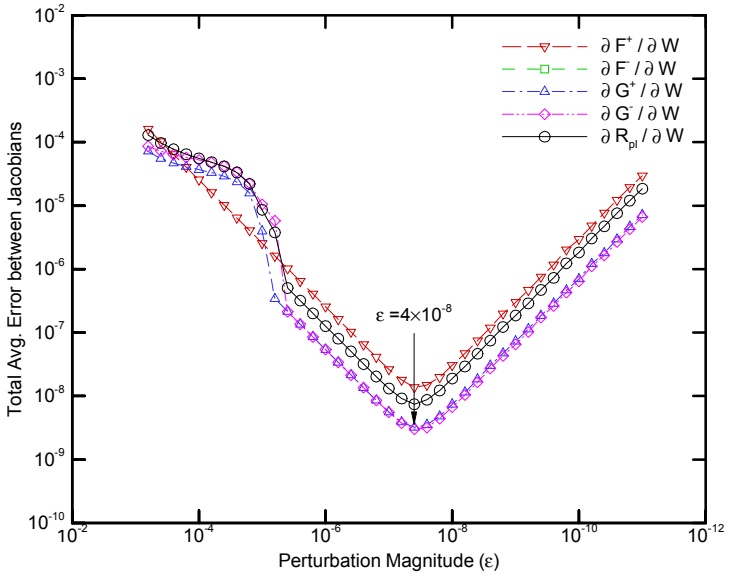


(b) Backward differencing

Figure 4.10 Effect of ε on total avg. error for all Jacobians in single precision.

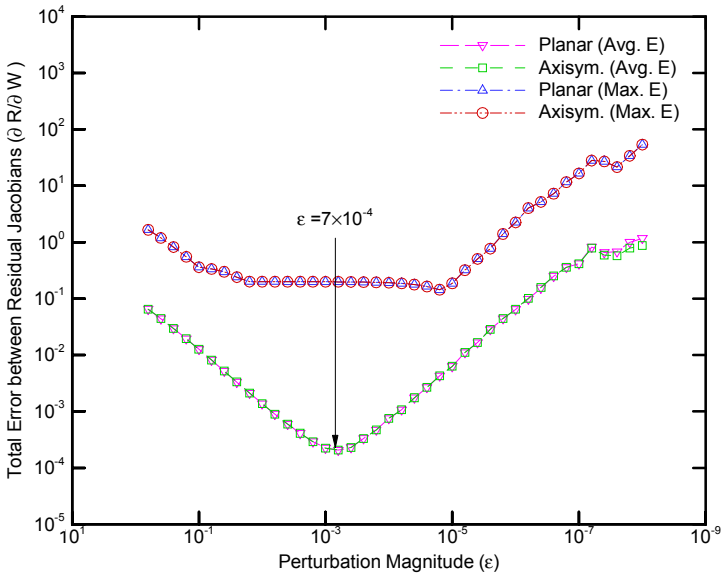


(a) Forward differencing

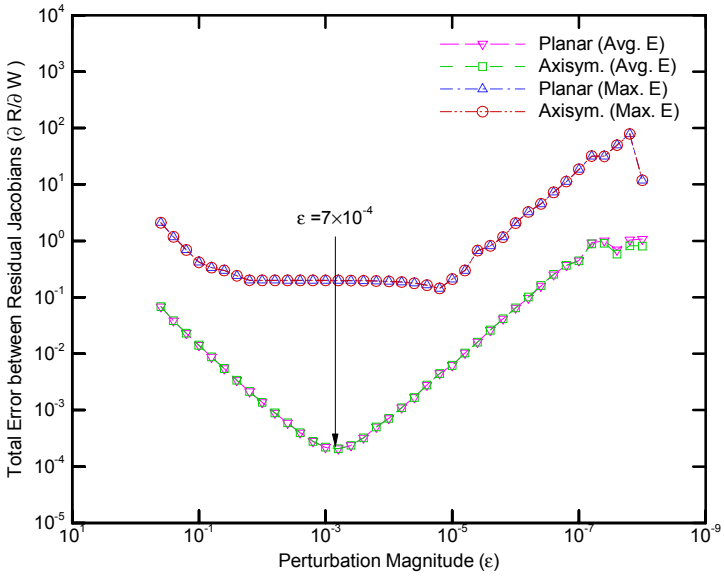


(b) Backward differencing

Figure 4.11 Effect of ε on total avg. error for all Jacobians in double precision.

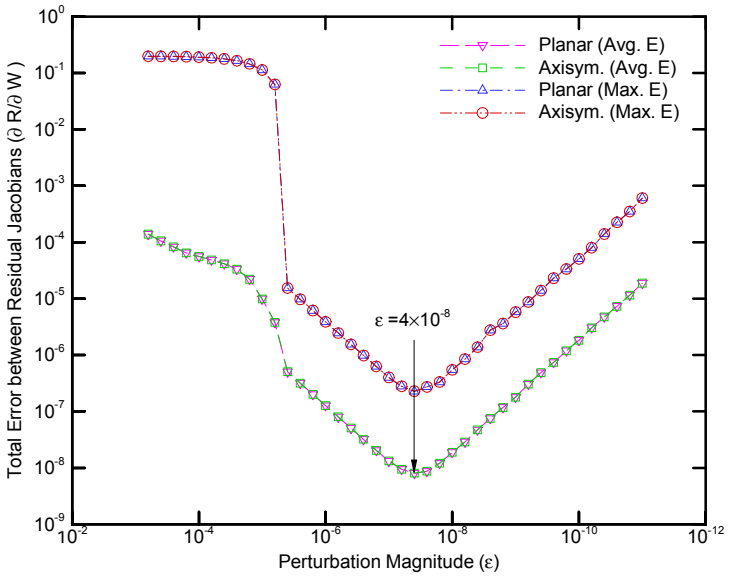


(a) Forward differencing

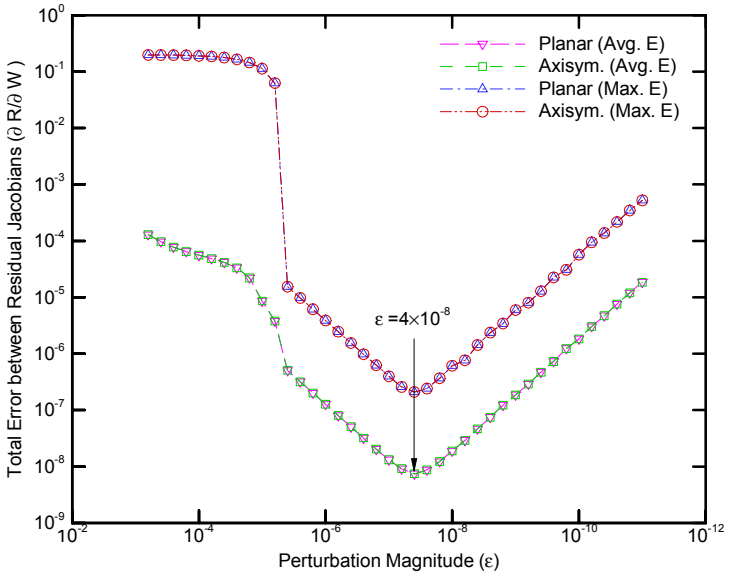


(b) Backward differencing

Figure 4.12 Effect of axisymmetry on total max. and avg. errors for $\partial \hat{R} / \partial \hat{W}$ in single precision.



(a) Forward differencing



(b) Backward differencing

Figure 4.13 Effect of axisymmetry on total max. and avg. errors for $\partial \hat{R} / \partial \hat{W}$ in double precision.

CHAPTER 5

SOLVER PERFORMANCE

5.1 Introduction

In case of developing a direct solver, the solution technique employed is the Newton's method. Although it is a well-known procedure for long time, due to the requirement of solving a huge matrix at once, it could not gain popularity until big improvements in the computer technology were realized.

In this study, a direct 2-D planar/axisymmetric Euler flow solver is developed. Governing equations are discretized by finite-volume method with upwind flux splitting schemes, and the resulting nonlinear system of equations are solved using Newton's Method with numerical/analytical Jacobian matrices. The performance of the solver is analyzed by several experiments related to convergence. First, the effects of Jacobian calculation on the direct flow solver are investigated. Correct formation and efficient solution of the Jacobian matrix is the main concern of the solver. UMFPACK sparse matrix solver package is used to minimize the storage and factorization costs. Also, several modifications and strategies are employed to the solution method. For better convergence, freezing the matrix after appropriate number of iterations is discussed. A time-like diagonal term addition in the first few steps of the iteration is applied to maintain the convergence of the solver even from poor initial conditions. The results for supersonic ramp geometry are presented in Section 5.2.

Several flux-splitting procedures like Van Leer and Roe schemes are analyzed and compared in Section 5.3. Also, higher order discretizations are compared in terms of flow convergence and CPU time. The results for second order Steger-Warming, Van Leer and Roe schemes with Van Albada limiter are given in Section 5.4. Finally, the solver performance on another geometry with different flow conditions is investigated. Section 5.5 presents the results for bump geometry for supersonic and subsonic flow conditions.

5.2 Test Case Results

The results are obtained for the test case being 15° supersonic ramp geometry with 33x25 grid as shown in Figure 4.1. As mentioned before M_∞=2.0 flow and inlet-outlet, wall and symmetry boundary conditions are employed. These geometry and flow conditions are suitable to visualize the effect of shocks. In this part of the study, the solver developed is run from a free-stream initialization of the flow. The analytical Jacobians are found from first-order Steger-Warming upwinding discretization of the governing equations. A detailed procedure of obtaining Jacobians numerically and analytically is given in Section 3.3.

All the calculations of this study are realized using Lahey/Fujitsu Optimizing Fortran 95 Compiler "lf95" on 1.5GHz Pentium IV dual processor openMosix cluster running on Linux 2.4. As discussed in Section 4.4, this compiler has an optimization property that may greatly improve solution procedure. Without changing the code, the code can be made to run in double precision mode using the optimization parameters. The machine epsilon ε_M of this compiler-computer configuration is $\varepsilon_M \cong 3.0 \times 10^{-8}$ for single precision, and $\varepsilon_M \cong 5.6 \times 10^{-17}$ for double precision. The CPU time required for the solution to converge, is the property of the specified computer processor again.

The formation and calculation of Jacobian matrix is one of the most critical parts of direct solvers. In the first stages of this study, a full matrix solver using LU factorization is used in the code. However, due to its high storage and factorization costs a sparse matrix solver UMFPACK is adapted to the code. Considering the test case, 15° ramp geometry with a 33x25 grid meaning 34x26 cells, and 4 flow variables at each cell, the total number of variables is 3536 and the Jacobian matrix has $3536^2 \cong 12.5$ million elements. In terms of file size this huge matrix is around 240 MB. In first-order Steger-Warming upwinding discretizations, 5-point stencil produces a block tridiagonal matrix including 5 4x4-block bands with the rest having the value zero. With the initial full matrix LU solver, the whole matrix including zeros has to be solved. Due to high storage, memory costs the solution procedure took nearly 26 hours of CPU time. However with the adaptation of UMFPACK, the performance of the solver improved greatly. After getting rid of the unnecessary zero values, the size of the matrix drops significantly to 1-2 MB. And the solution converges rapidly in 10-20 seconds of CPU time.

Considering both the analytical and numerical Jacobian evaluation methods using first-order Steger-Warming scheme, the solver performance is analyzed in terms of convergence history and the required CPU time. Actually the results for optimum value of ε obtained in the previous chapter is compared with the analytical results. The use of diagonal term addition and Jacobian matrix freezing are investigated to improve the efficiency of the solver. Both planar and axisymmetric cases are considered in the analyses. The numerical Jacobians are calculated using both forward and backward differencing to investigate their importance. All the study is realized for both single and double precision in order to see the effect of computer precision on the convergence.

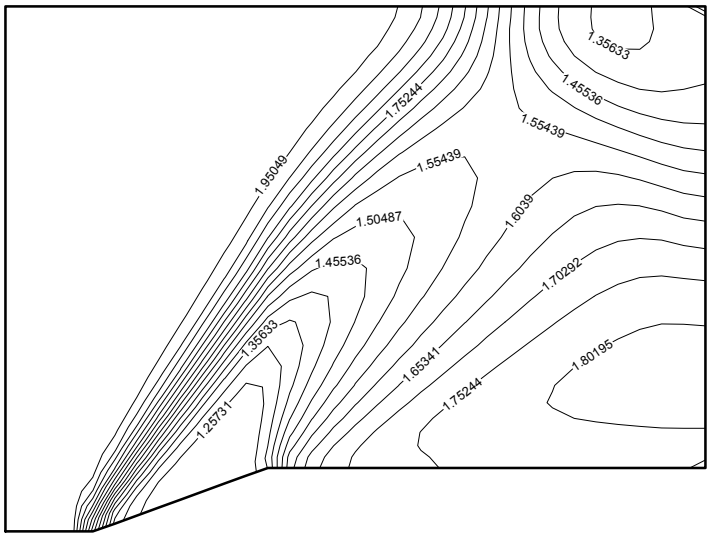
Figure 5.1 shows the flow solution in terms of Mach numbers obtained for first-order Steger-Warming scheme. Mach contours for both planar and axisymmetric cases show the general behaviour of a first-order method. The shock can be seen but it's not very sharp.

Looking at the general behaviour of all convergence history figures, it is obviously seen that with double precision the density residual converges to the order of 10^{-15}, while with single precision, only an order of 10^{-6} can be achieved. Thus, the usage of double precision improves the order of convergence significantly without too much change in the number of iterations and CPU time.

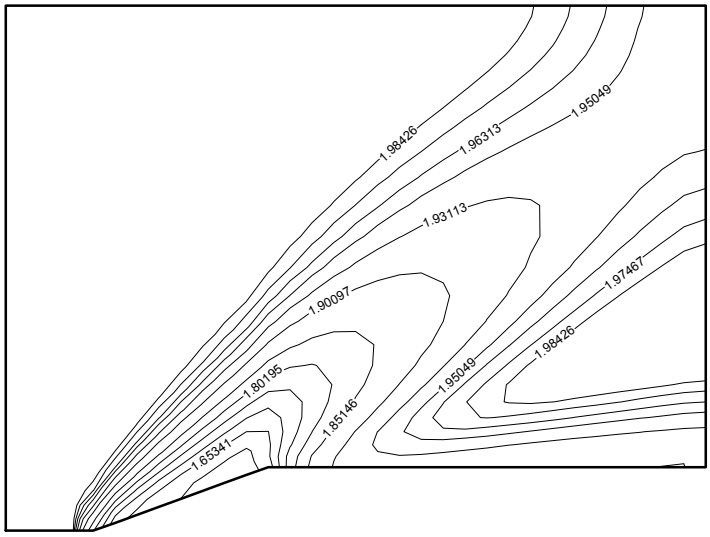
The best combination of when to add and remove the diagonal term, and when to freeze the Jacobian matrix is very important to obtain the best performance of the solver. After a trial-error study using the analytical Jacobians, it is observed that diagonal term addition is necessary in case of planar flow. Since a free-stream initialization is used, this modification is required to overcome the poor initial guess. Figure 5.2 shows the effect of diagonal term addition on the convergence history for planar density residual with CPU time values. As mentioned before, removing the diagonal term reduces the number of iterations and CPU time for convergence. Δt_{rm}=1.5 gives the best performance with around 10 iterations and 10-13 seconds of CPU time. However, since the axisymmetric source terms behave like an addition to the diagonal of the Jacobian matrix, another modification is unnecessary for axisymmetric case. Thus, as shown from Figure 5.3, best results are obtained for no Δt. The solution converges in 6-8 iterations. This corresponds 6-9 seconds of CPU time.

Another trial-error study is realized, to find the order of residual at which the freezing is applied, using the analytical Jacobians and the Δt values desired above. With the use of freezing, although the number of iterations increases, the CPU time spent for convergence decreases since the Jacobian calculation procedure is skipped. Figure 5.4 shows the effect of freezing on the convergence history for planar density residual and the corresponding CPU time values. For single precision, $R_{frz}= 1x10^{-2}$ gives the same convergence as no freezing case with a less CPU time. For double precision, $R_{frz}= 1x10^{-4}$ gives the same convergence as no freezing case with a less CPU time. Considering Figure 5.5 that give the effect of freezing for the axisymmetric case, $R_{frz}= 1x10^{-2}$ comes out as the best value with less number of iterations and CPU time.

After these trial-error procedures, using the best combinations obtained for diagonal term modification and Jacobian freezing, the effect of finite-difference perturbation magnitude on the convergence of the flow is analyzed through Figures 5.6 and 5.9. From these figures, it is observed that there is no significant difference between the forward and backward differencing in the calculation of the numerical Jacobians, because the control mechanism employed changes the differencing procedure in case of a problem. It can be seen that the best convergence with least CPU time is always obtained with the use of optimum perturbation magnitude, being $\varepsilon = 7\text{x}10^{-4}$ for single precision and $\varepsilon = 4\text{x}10^{-8}$ for double precision. Although for some other ε values the flow may not converge, with the optimum perturbation magnitude nearly the same convergence history is obtained with the analytical method. For the planar case, convergence is obtained in 10-11 iterations taking nearly 15 seconds of CPU time. For the axisymmetric case, convergence is obtained around 6-8 iterations taking nearly 9 seconds of CPU time.

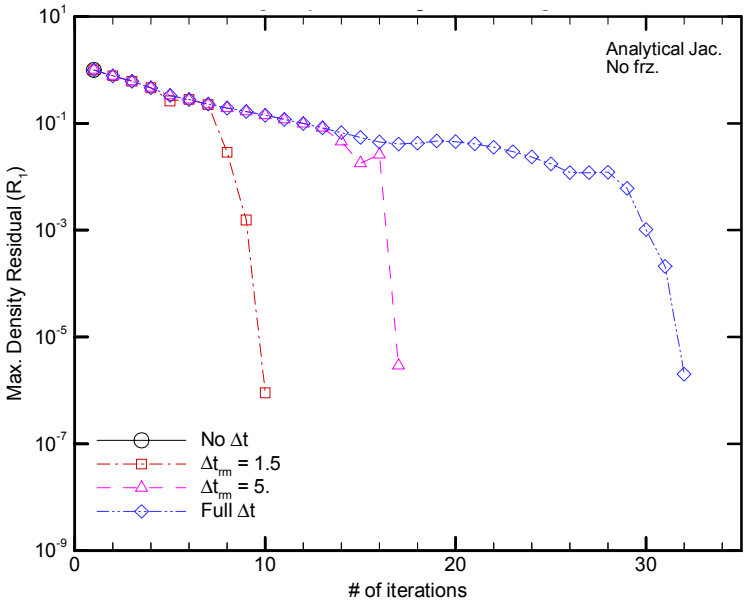


(a) 2-D planar case



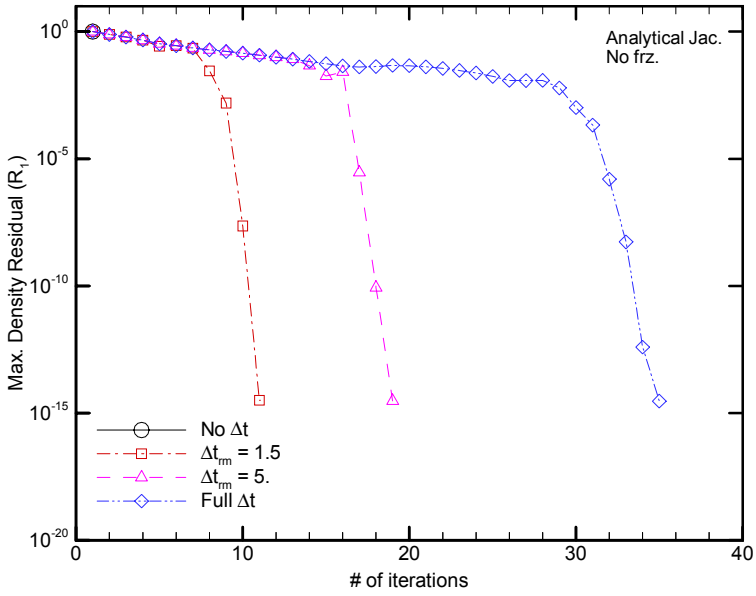
(b) 2-D axisymmetric case

Figure 5.1 Mach contours for 1st order Steger-Warming scheme in double precision.



Δt	CPU Time (s)
No Δt	NaN
Δt_{rm}= 1.5	10.04
Δt_{rm}= 5.	16.55
Full Δt	30.84

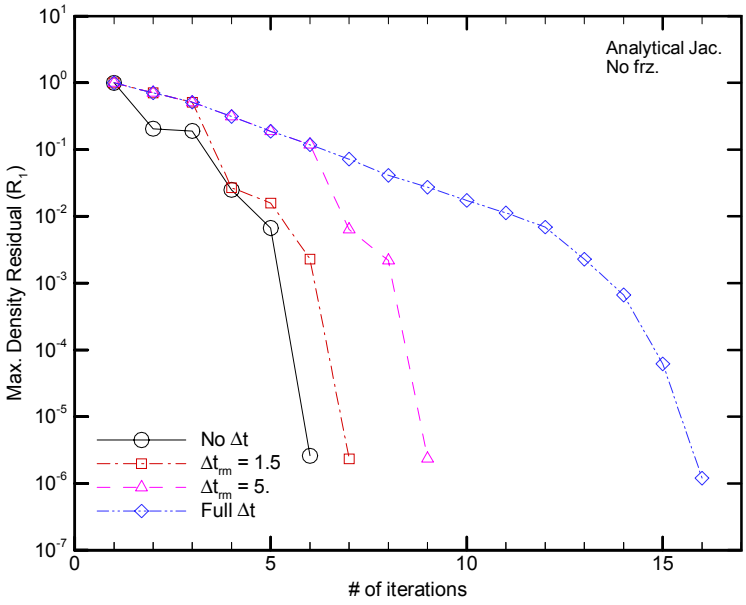
(a) Single precision



Δt	CPU Time (s)
No Δt	NaN
Δt_{rm}= 1.5	13.32
Δt_{rm}= 5.	22.65
Full Δt	257.27

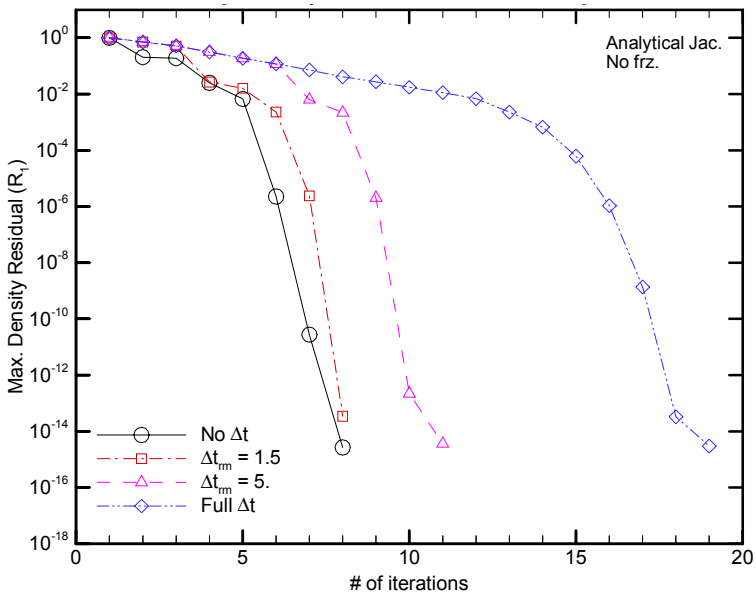
(b) Double precision

Figure 5.2 Effect of Δt on the convergence history for planar density residual.



Δt	CPU Time (s)
No Δt	6.05
Δt_{rm}= 1.5	6.99
Δt_{rm}= 5.	8.95
Full Δt	15.62

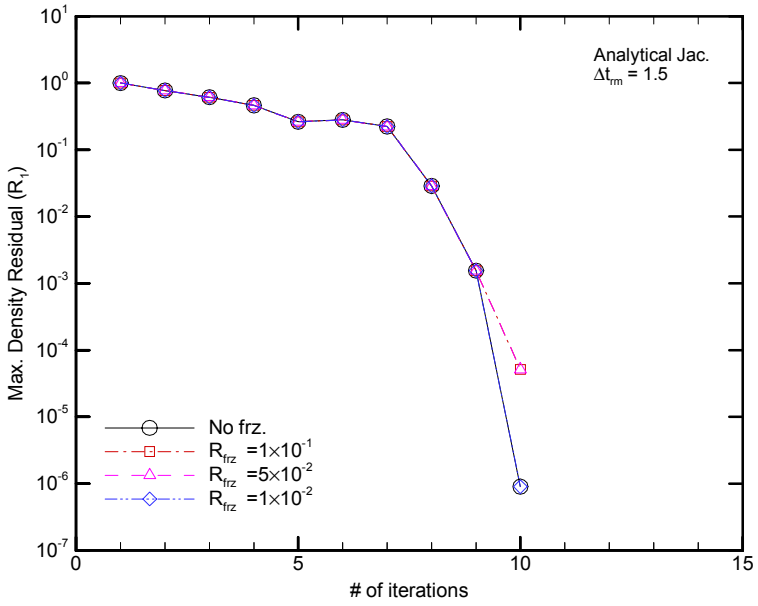
(a) Single precision



Δt	CPU Time (s)
No Δt	9.79
Δt_{rm}= 1.5	9.84
Δt_{rm}= 5.	13.28
Full Δt	22.59

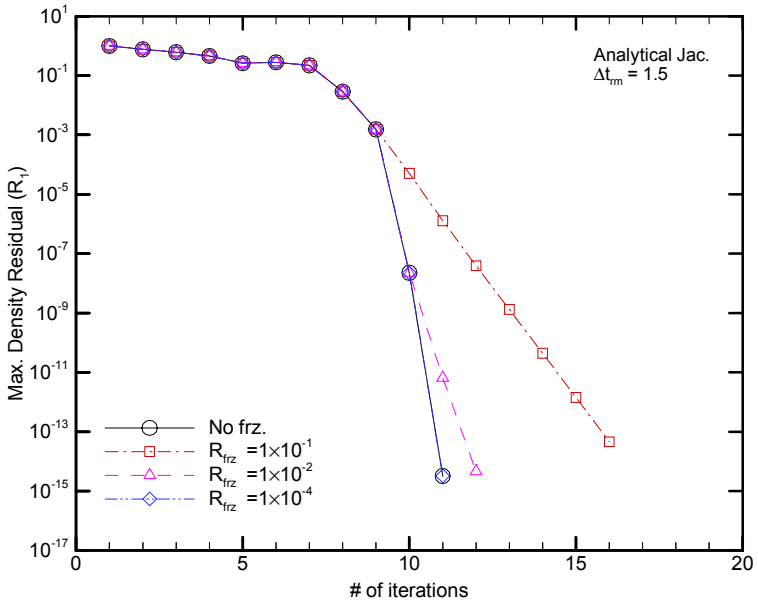
(b) Double precision

Figure 5.3 Effect of Δt on the convergence history for axisymmetric density residual.



Tolerance	CPU Time (s)
No freeze	10.04
R_{frz}= 1x10^{-1}	8.75
R_{frz}= 5x10^{-1}	8.64
R_{frz}= 1x10^{-2}	9.28

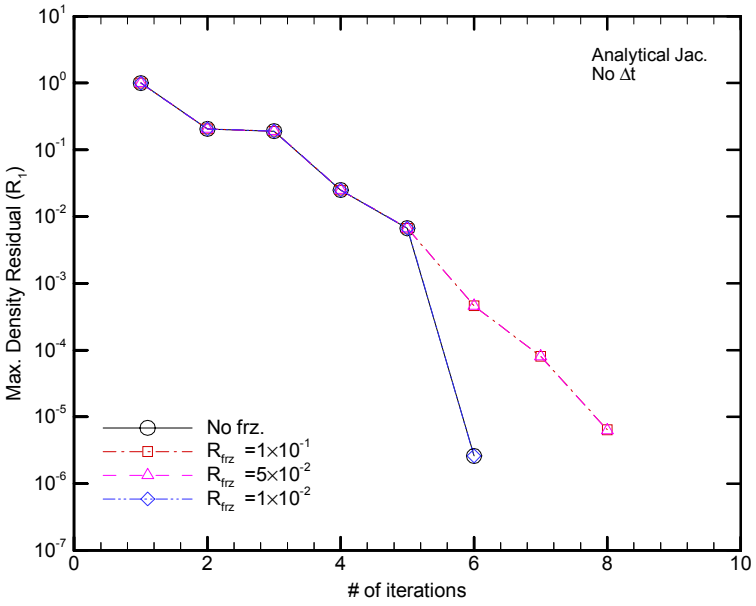
(a) Single precision



Tolerance	CPU Time (s)
No freeze	13.32
R_{frz}= 1x10^{-1}	12.85
R_{frz}= 1x10^{-2}	12.20
R_{frz}= 1x10^{-4}	12.51

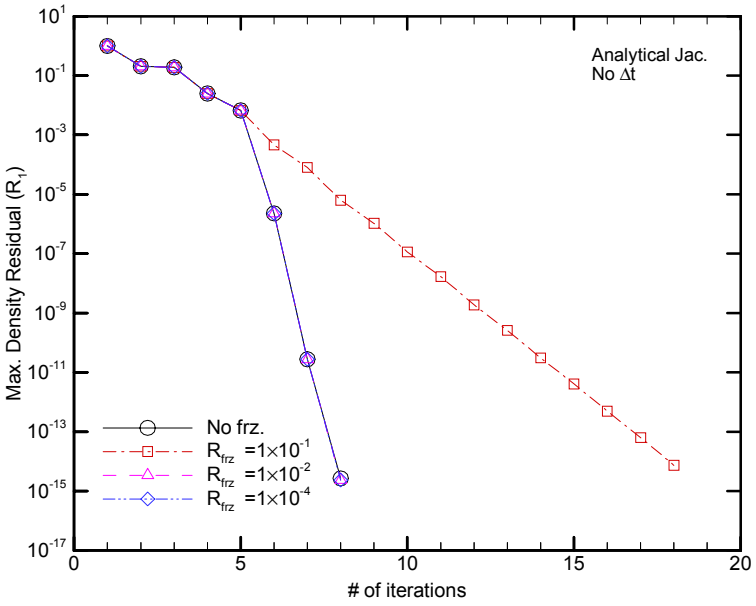
(b) Double precision

Figure 5.4 Effect of freezing on the convergence history for planar density residual.



Tolerance	CPU Time (s)
No freeze	6.05
R_{frz}= 1x10^{-1}	5.52
R_{frz}= 5x10^{-1}	8.20
R_{frz}= 1x10^{-2}	6.04

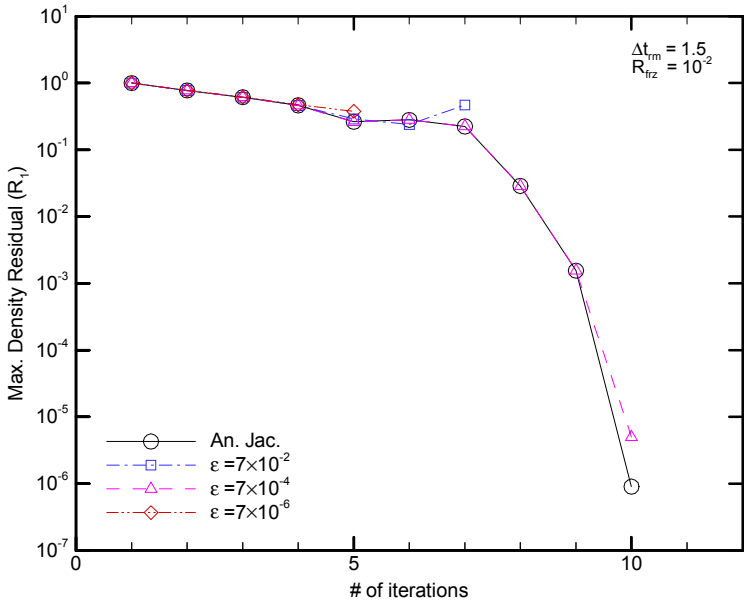
(a) Single precision



Tolerance	CPU Time (s)
No freeze	9.79
R_{frz}= 1x10^{-1}	10.32
R_{frz}= 1x10^{-2}	8.15
R_{frz}= 1x10^{-4}	8.23

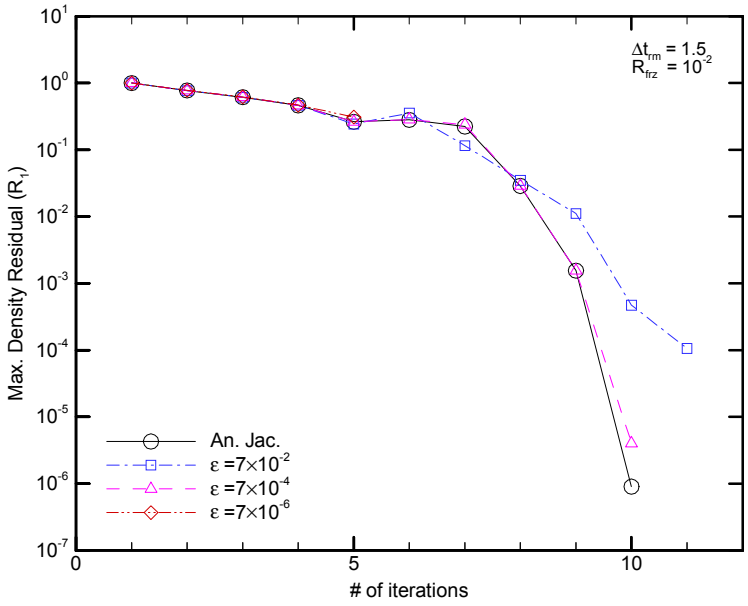
(b) Double precision

Figure 5.5 Effect of freezing on the convergence history for axisymmetric density residual.



Jacobian	CPU Time (s)
Analytic	9.37
$\varepsilon = 7\text{x}10^{-2}$	NaN
$\varepsilon = 7\text{x}10^{-4}$	14.53
$\varepsilon = 7\text{x}10^{-6}$	NaN

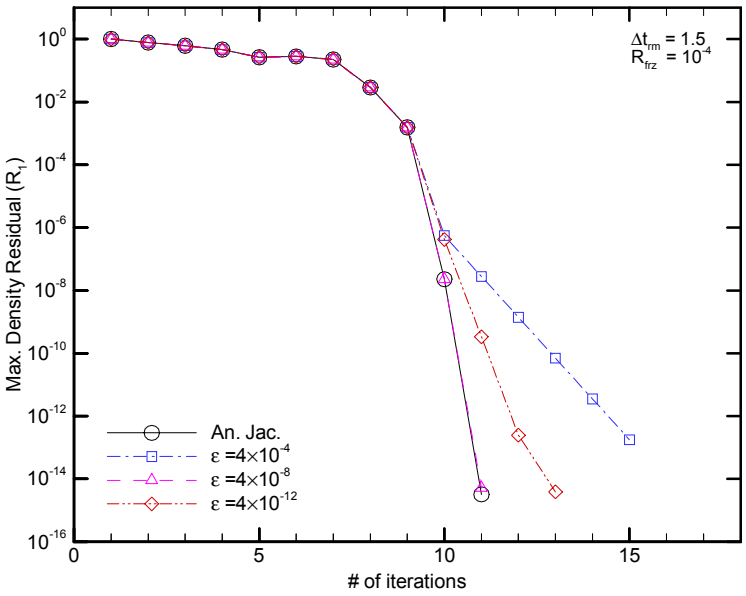
(a) Forward differencing



Jacobian	CPU Time (s)
Analytic	9.37
$\varepsilon = 7\text{x}10^{-2}$	15.97
$\varepsilon = 7\text{x}10^{-4}$	14.47
$\varepsilon = 7\text{x}10^{-6}$	NaN

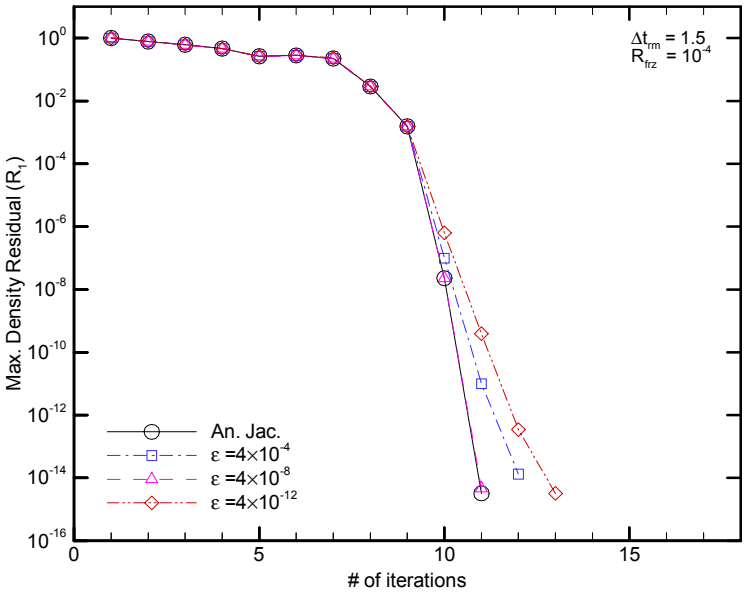
(b) Backward differencing

Figure 5.6 Effect of ε on the convergence history for planar density residual in single precision.



Jacobian	CPU Time (s)
Analytic	12.69
$\varepsilon = 4x10^{-4}$	22.05
$\varepsilon = 4x10^{-8}$	16.27
$\varepsilon = 4x10^{-12}$	19.11

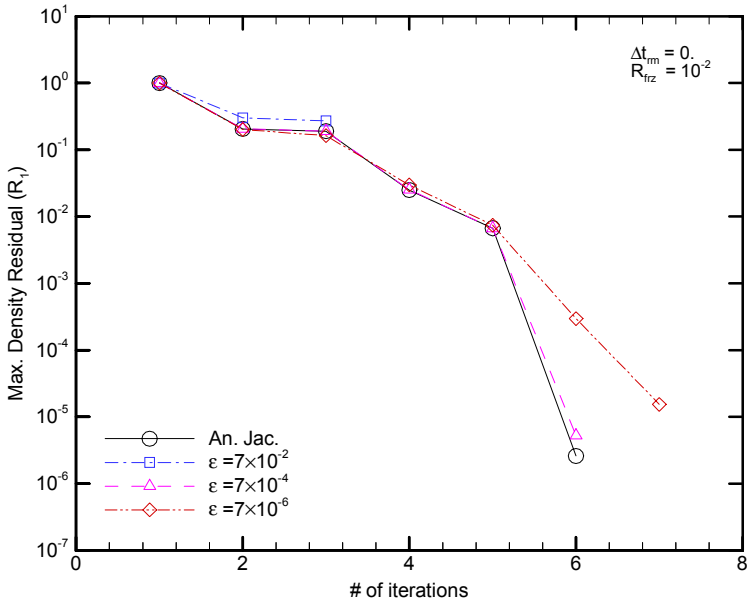
(a) Forward differencing



Jacobian	CPU Time (s)
Analytic	12.69
$\varepsilon = 4x10^{-4}$	17.71
$\varepsilon = 4x10^{-8}$	16.26
$\varepsilon = 4x10^{-12}$	19.01

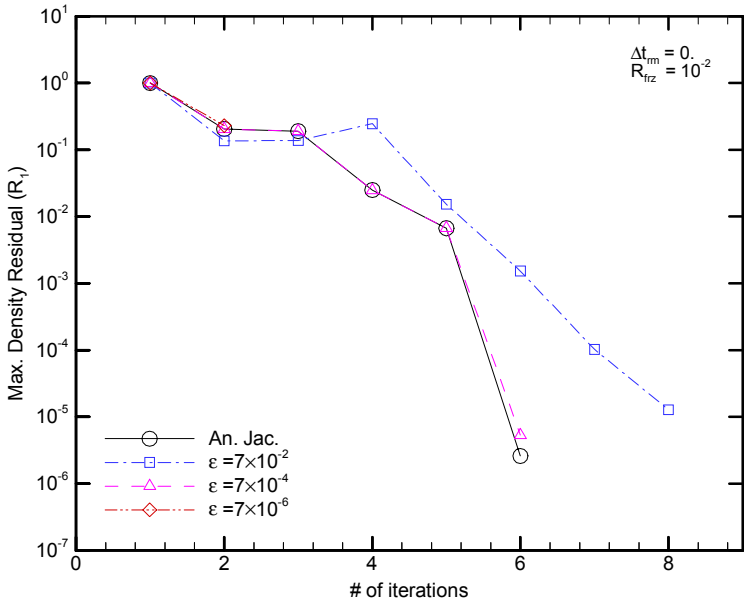
(b) Backward differencing

Figure 5.7 Effect of ε on the convergence history for planar density residual in double precision.



Jacobian	CPU Time (s)
Analytic	5.99
$\varepsilon = 7\text{x}10^{-2}$	NaN
$\varepsilon = 7\text{x}10^{-4}$	8.29
$\varepsilon = 7\text{x}10^{-6}$	8.64

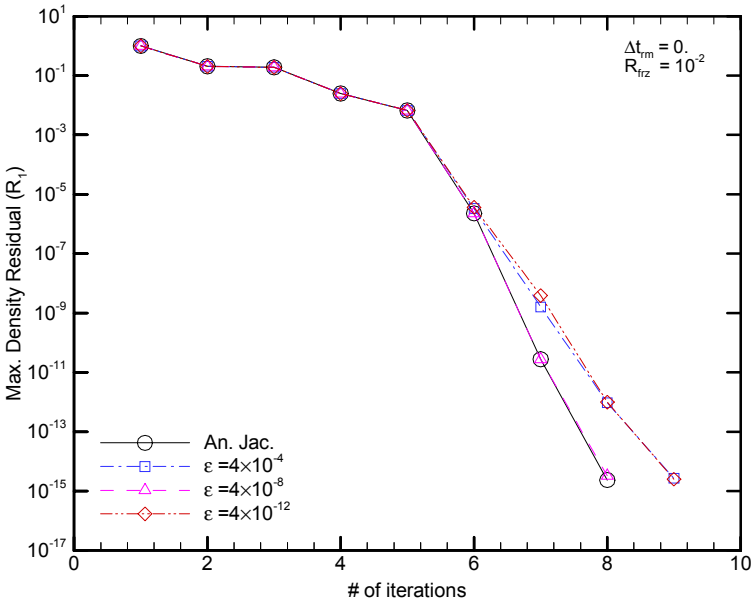
(a) Forward Differencing



Jacobian	CPU Time (s)
Analytic	5.99
$\varepsilon = 7\text{x}10^{-2}$	9.54
$\varepsilon = 7\text{x}10^{-4}$	8.8
$\varepsilon = 7\text{x}10^{-6}$	NaN

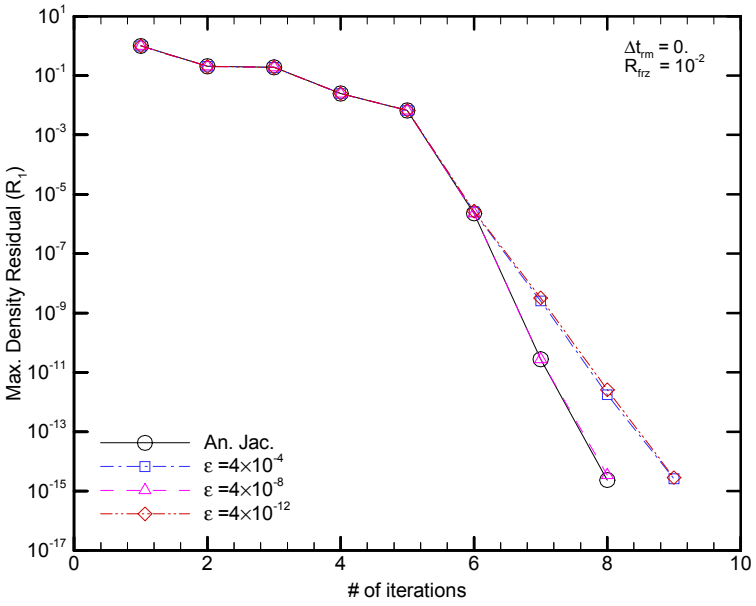
(b) Backward differencing

Figure 5.8 Effect of ε on the convergence history for axisymmetric density residual in single precision.



Jacobian	CPU Time (s)
Analytic	8.25
$\varepsilon = 4x10^{-4}$	10.06
$\varepsilon = 4x10^{-8}$	9.72
$\varepsilon = 4x10^{-12}$	10.08

(a) Forward differencing



Jacobian	CPU Time (s)
Analytic	8.25
$\varepsilon = 4x10^{-4}$	10.19
$\varepsilon = 4x10^{-8}$	9.66
$\varepsilon = 4x10^{-12}$	10.08

(b) Backward differencing

Figure 5.9 Effect of ε on the convergence history for axisymmetric density residual in double precision.

5.3 Results for Different Flux Splitting Schemes

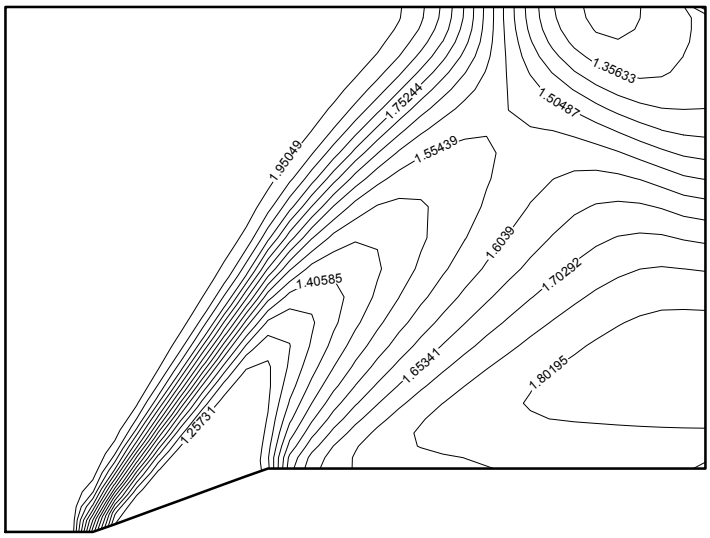
The results are again obtained for the test case being 15° supersonic ramp geometry having 33x25 grid with M_∞=2.0 flow and inlet-outlet, wall and symmetry boundary conditions. After the Steger-Warming scheme is studied in detail, in this section the numerical Jacobians are found from first-order Van Leer and Roe upwinding discretization of the governing equations. The benefits of using the same flux calculation scheme for both Jacobian and residual calculation are analyzed in terms of the convergence of the solver.

Considering numerical Jacobian evaluation methods using first-order Van Leer and Roe's schemes, the solver performance is analyzed in terms of convergence history and the required CPU time. Actually the results for optimum value of ε obtained in the previous chapter is used in the calculations. Also, the results for analytical Steger-Warming Jacobians with Van Leer and Roe discretized residuals are compared. The appropriate values of diagonal term addition and Jacobian matrix freezing are used to reflect the best performance of the solver. Both planar and axisymmetric cases are considered in the analyses. The numerical Jacobians are calculated using forward differencing. All the study is realized for both single and double precision in order to see the effect of computer precision on the convergence.

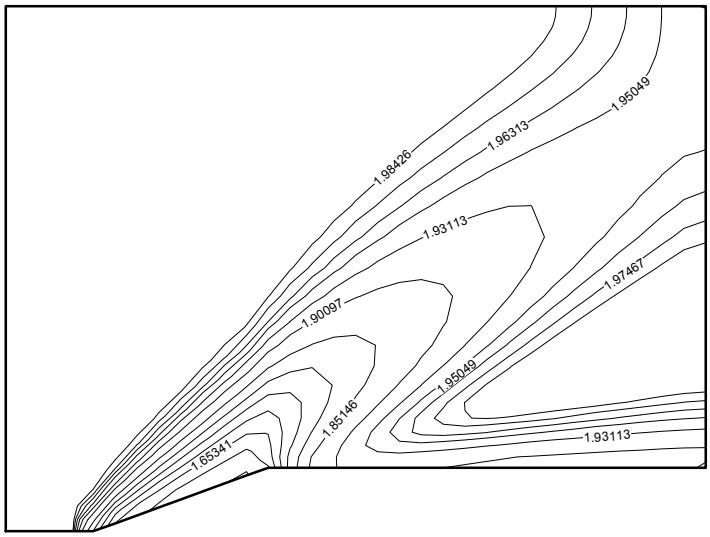
Mach contours are given for first-order Van Leer scheme in Figure 5.10, and for first-order Roe scheme in Figure 5.11. Comparing the contours with Figure 5.1 of first-order Steger-Warming, although all the figures are very similar, Roe seems to give better results.

Looking at the general behaviour of all convergence history figures, it is again obvious that the usage of double precision improves the order of convergence significantly without too much change in the number of iterations and CPU time.

The effect of different flux splitting schemes on the convergence of the flow is analyzed through Figures 5.12 and 5.15. First of all, a general trend observed is that Steger-Warming has the fastest convergence in terms of both iterations and CPU times, and Roe is the slowest one compared to the others. Considering Figure 5.12(a) through Figure 5.15(a), it is observed that using different schemes in the calculation of the Jacobian and the residual makes the performance of the solver worse. Van Leer or Roe residuals with Steger-Warming analytical Jacobians converge much slower than Van Leer or Roe residuals used with its corresponding numerical Jacobian. Especially, Roe reflects this well. In case of double precision, this regression in convergence can be seen well. Considering Figure 5.13, Roe with its numerical Jacobians converged in 17 iterations taking 29.4 seconds of CPU time, while with the use of Steger-Warming analytical Jacobians convergence is possible after 150 iterations taking 99.62 seconds of CPU time. Thus, using the same flux calculation scheme for both Jacobian and residual calculation are very important for a faster convergence of the solver. When numerical Jacobians of the same flux calculation scheme is used, very good convergence results are obtained for first-order Van Leer and Roe schemes.

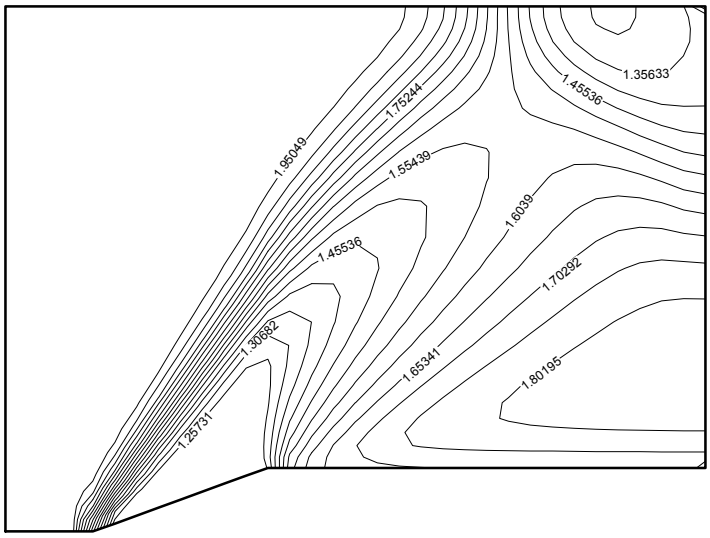


(a) 2-D planar case

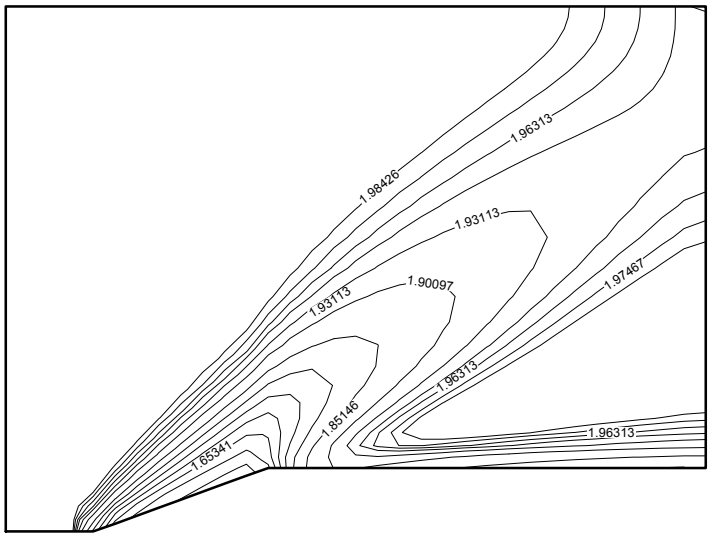


(b) 2-D axisymmetric case

Figure 5.10 Mach contours for 1st order Van Leer scheme in double precision.

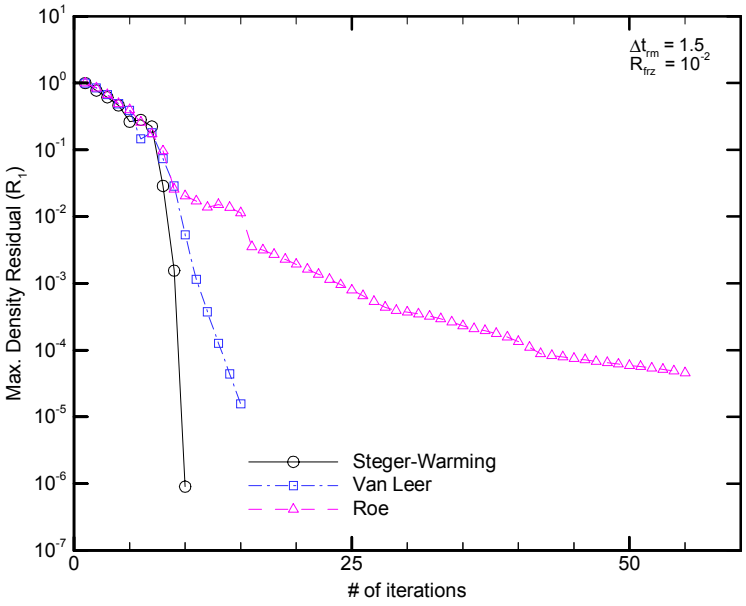


(a) 2-D planar case



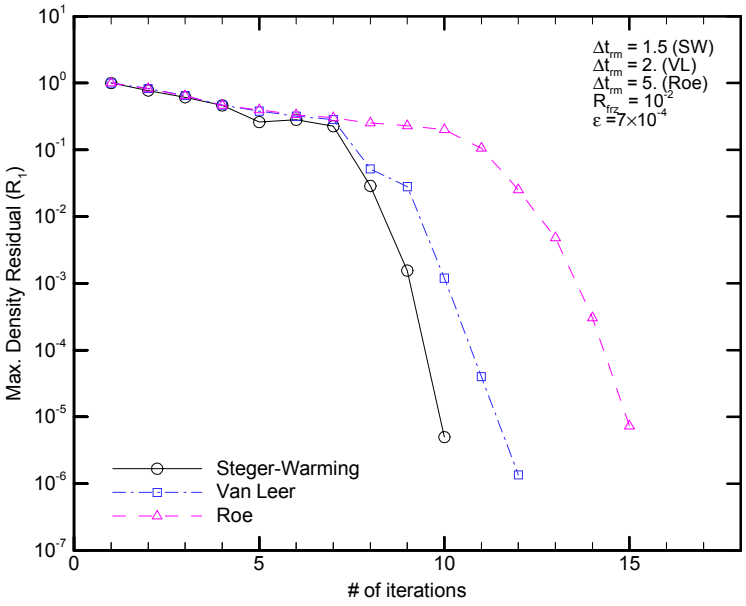
(b) 2-D axisymmetric case

Figure 5.11 Mach contours for 1st order Roe scheme in double precision.



FS	CPU Time (s)
SW	9.37
VL	12.55
Roe	30.75

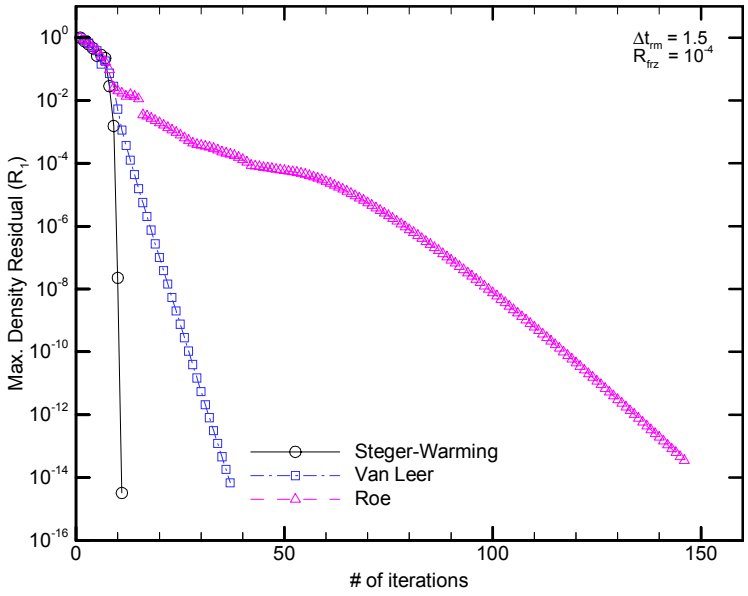
(a) S-W Analytical Jacobians



FS	CPU Time (s)
SW	14.53
VL	15.72
Roe	29.24

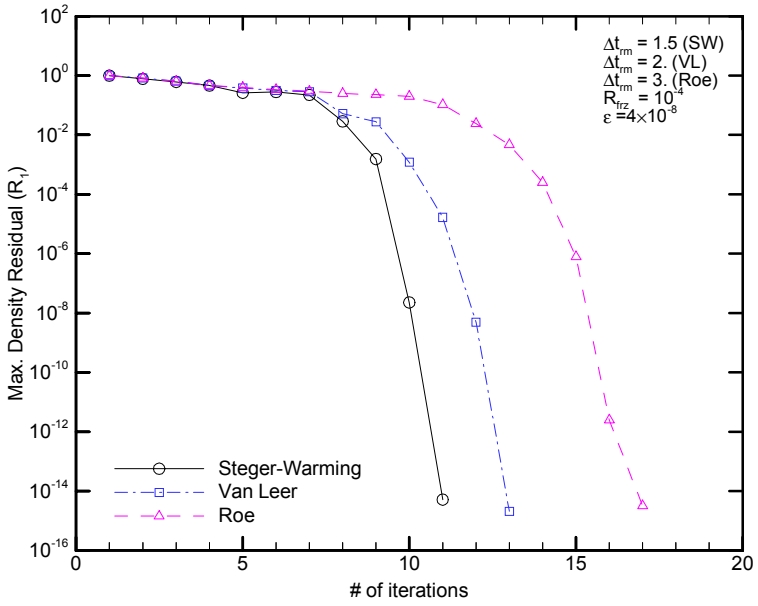
(b) Numerical Jacobians

Figure 5.12 Effect of different flux splitting schemes on the convergence history for planar density residual in single precision.



FS	CPU Time (s)
SW	12.69
VL	26.51
Roe	99.62

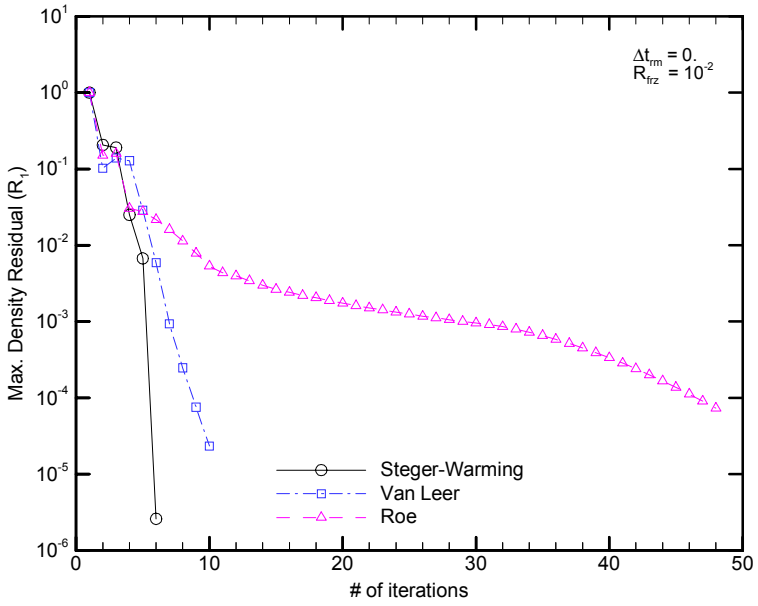
(a) S-W Analytical Jacobians



FS	CPU Time (s)
SW	16.27
VL	28.44
Roe	29.40

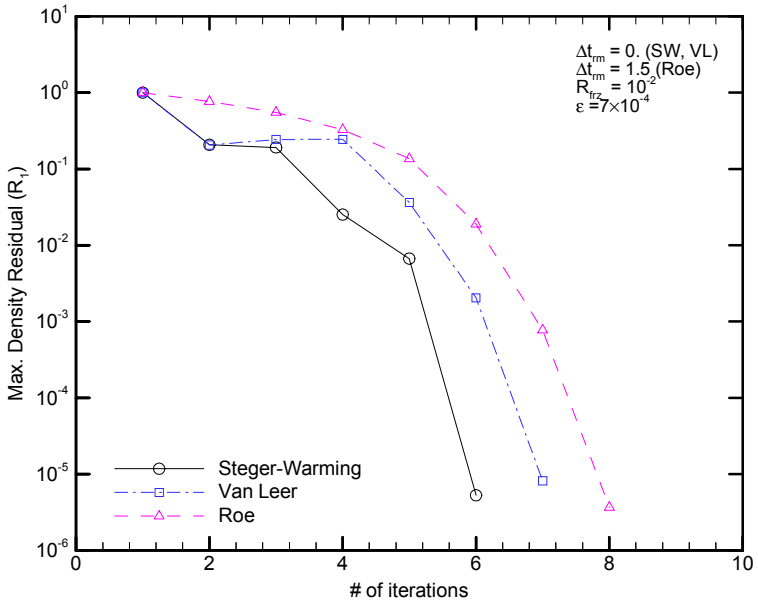
(b) Numerical Jacobians

Figure 5.13 Effect of different flux splitting schemes on the convergence history for planar density residual in double precision.



FS	CPU Time (s)
SW	5.99
VL	8.13
Roe	27.69

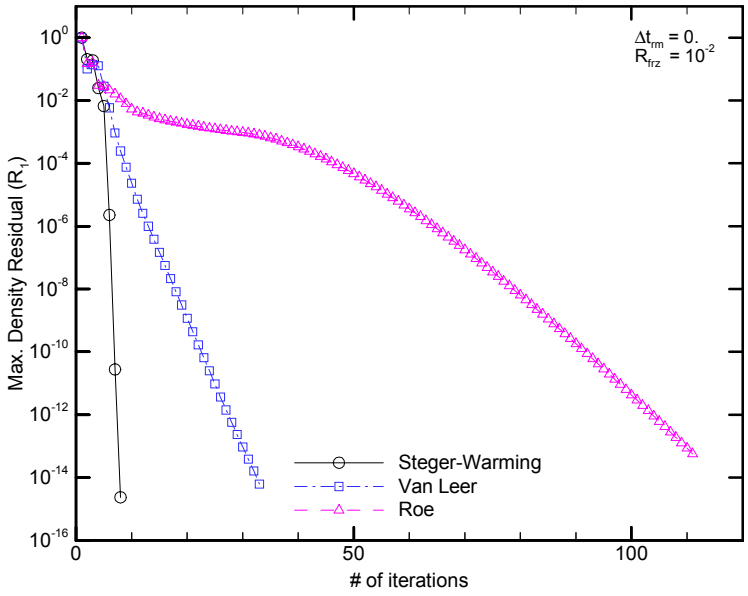
(a) S-W Analytical Jacobians



FS	CPU Time (s)
SW	8.29
VL	9.04
Roe	15.59

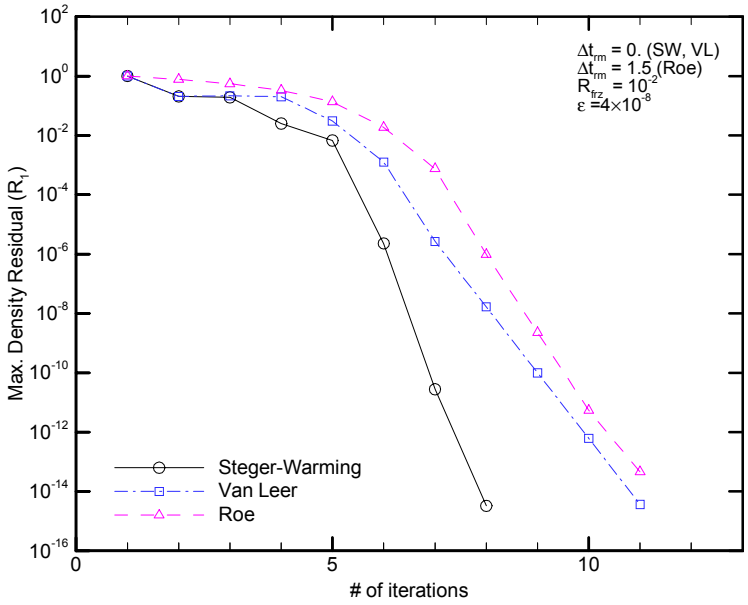
(b) Numerical Jacobians

Figure 5.14 Effect of different flux splitting schemes on the convergence history for axisymmetric density residual in single precision.



FS	CPU Time (s)
SW	8.25
VL	18.33
Roe	128.81

(a) S-W Analytical Jacobians



FS	CPU Time (s)
SW	9.72
VL	11.53
Roe	14.99

(b) Numerical Jacobians

Figure 5.15 Effect of different flux splitting schemes on the convergence history for axisymmetric density residual in double precision.

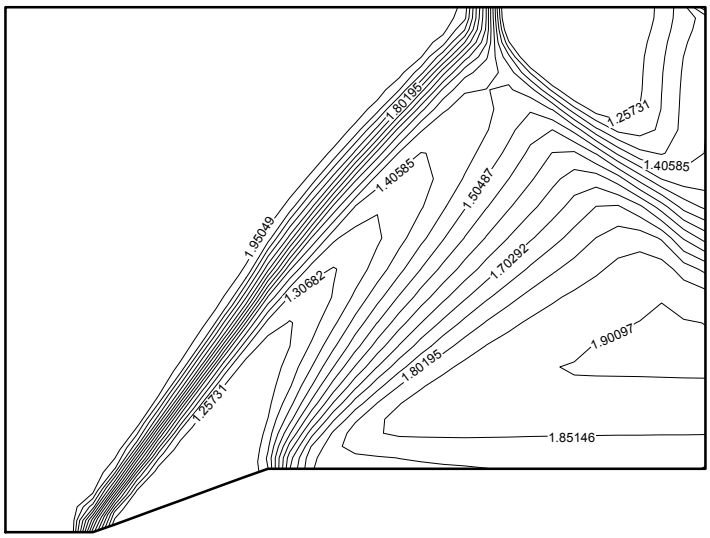
5.4 Results for Higher-Order Discretizations

The results are obtained for the test case being 15° supersonic ramp geometry having 33x25 grid with M_∞=2.0 flow and inlet-outlet, wall and symmetry boundary conditions. After the first-order schemes are considered, in this section the numerical Jacobians are found from second-order Steger-Warming, Van Leer and Roe upwinding discretization of the governing equations. As explained in Section 2.5, Van Albada limiter is used in the extrapolation of the flow variables. The behaviours of different flux splitting schemes with the limiter are analyzed in terms of the convergence of the solver.

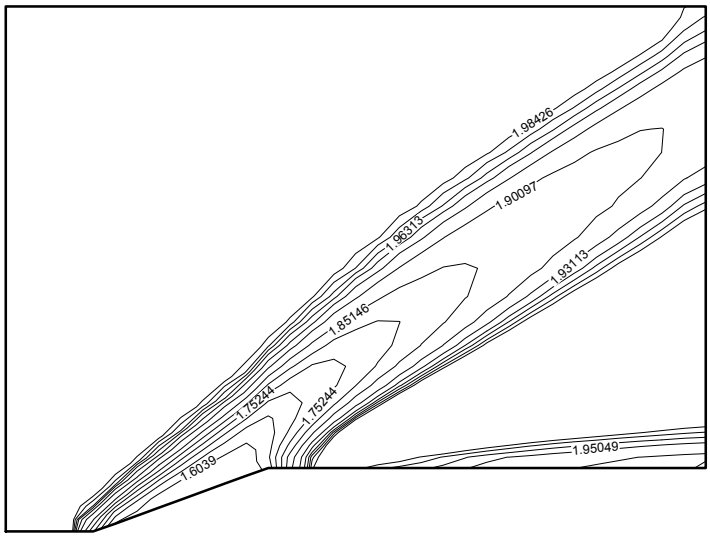
Considering numerical Jacobian evaluation methods using second-order Steger-Warming, Van Leer and Roe's schemes, the solver performance is analyzed in terms of convergence history and the required CPU time. The optimum values of ε obtained in the previous chapters are used in the calculations. Both planar and axisymmetric cases are considered in the analyses. The numerical Jacobians are calculated using forward differencing, double precision is used to get a better convergence performance. The appropriate values of diagonal term addition and Jacobian matrix freezing are used. After some trial-error study it is observed that matrix freezing does not improve the convergence, on the contrary causes divergence. This may be due to the nature of the second-order discretization. Time-like diagonal term addition to the matrix diagonal seems to be very important in higher-order schemes. Even removing them after some number of iterations may cause the code to diverge. Besides all of these parameters, the convergence problem is obviously seen in second-order schemes. Although it does not diverge, the solution goes into a limit-cycle and residual does not decrease anymore. This is due to the nature of the flux limiter employed in the solver. Actually, the choice and application of limiters is another challenging task that must be analyzed in detail.

Mach contour results for second-order flux splitting schemes are given through Figures 5.16 and 5.18. Comparing the contours it can be observed that Roe gives a better solution than the other two schemes. Looking at the shock and expansion waves, the contours of Roe are much sharper than that of Van Leer or Steger-Warming schemes. When these second-order contours are compared with the first-order results given in Figure 5.1, Figure 5.10 and Figure 5.11, it is obviously seen that with the second-order discretization the shock and expansion parts of the contour are captured much better.

The effect of different second-order flux splitting schemes on the convergence of the flow is analyzed in Figure 5.19 and Table 5.12. First of all, a general trend observed is that Steger-Warming has the fastest convergence in terms of both iterations and CPU times, and Roe is the slowest one compared to the others. Considering Figure 5.19, the number of iterations for the planar flow to converge is much higher than that is required for the first-order schemes. Nevertheless, in case of planar flow, full convergence in the order of 10^{-14} is obtained with the use of second-order flux splitting methods and Van Albada's limiter. However, for the axisymmetric case as given in Figure 5.20, the solution goes into a limit-cycle with the residual value around 10^{-8}, and does not decrease anymore. Implementation of another limiter or considering a totally different procedure for higher-order schemes may improve the solution to a fully convergence.

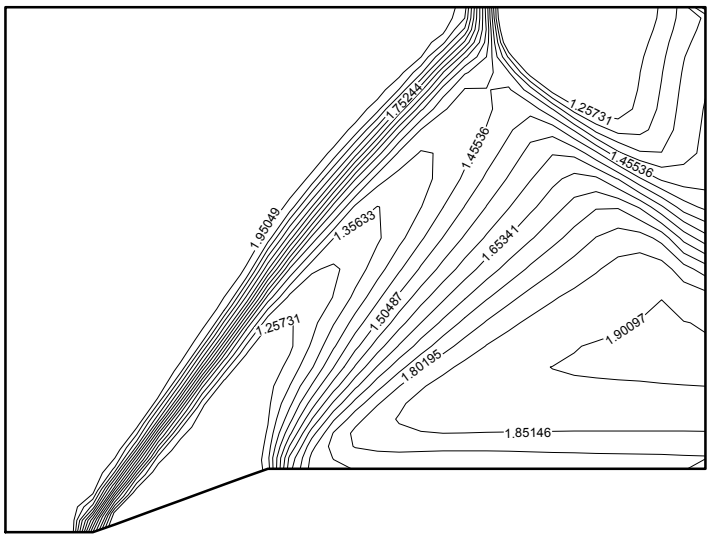


(a) 2-D planar case

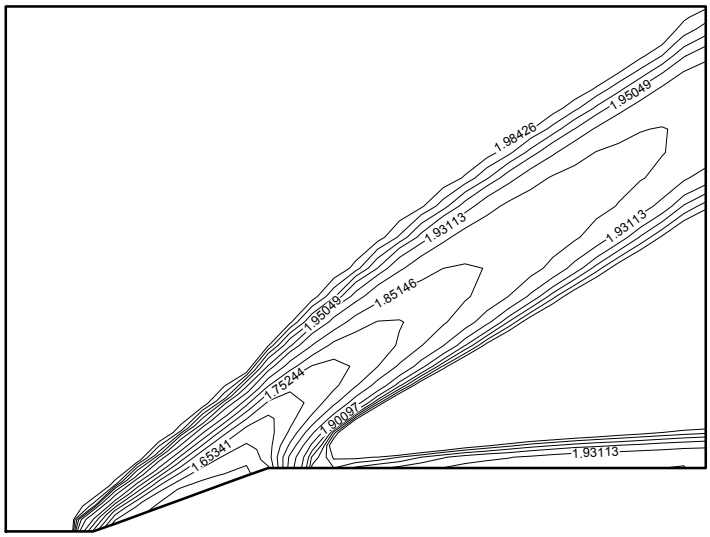


(b) 2-D axisymmetric case

Figure 5.16 Mach contours for 2nd order Steger-Warming scheme in double precision.

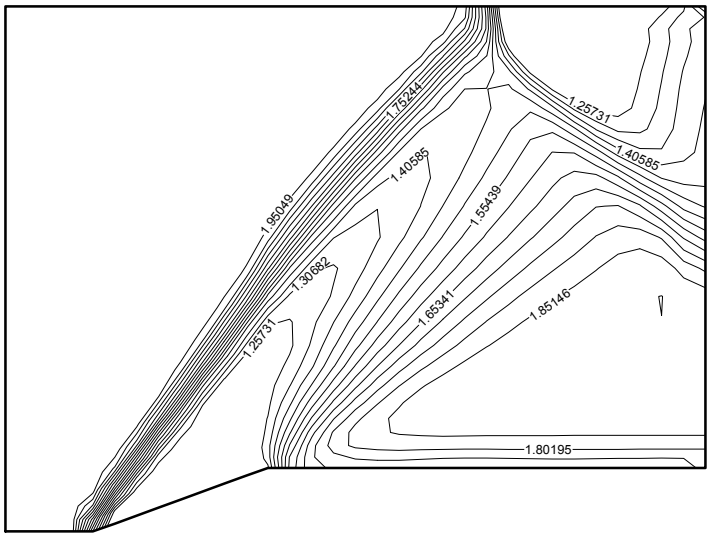


(a) 2-D planar case

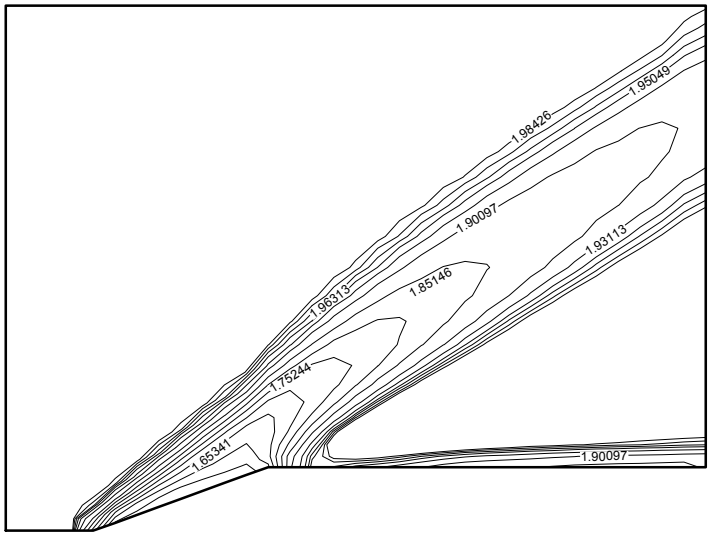


(b) 2-D axisymmetric case

Figure 5.17 Mach contours for 2nd order Van Leer scheme in double precision.

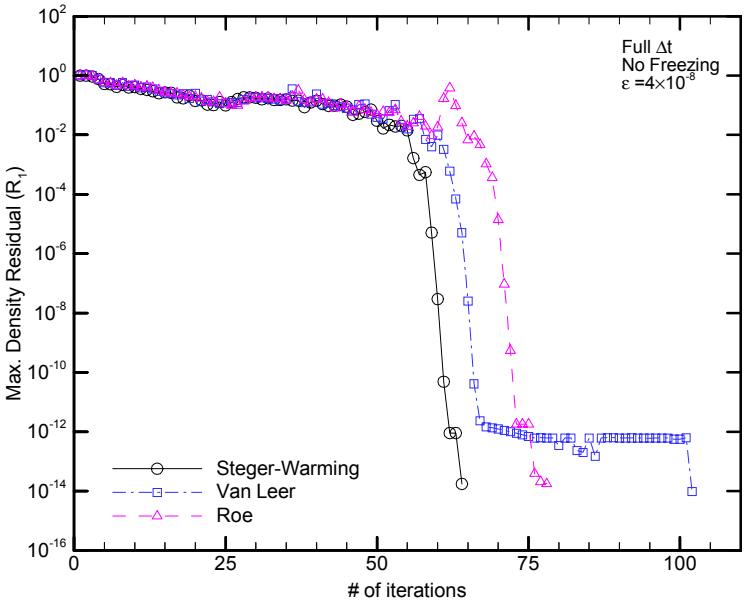


(a) 2-D planar case



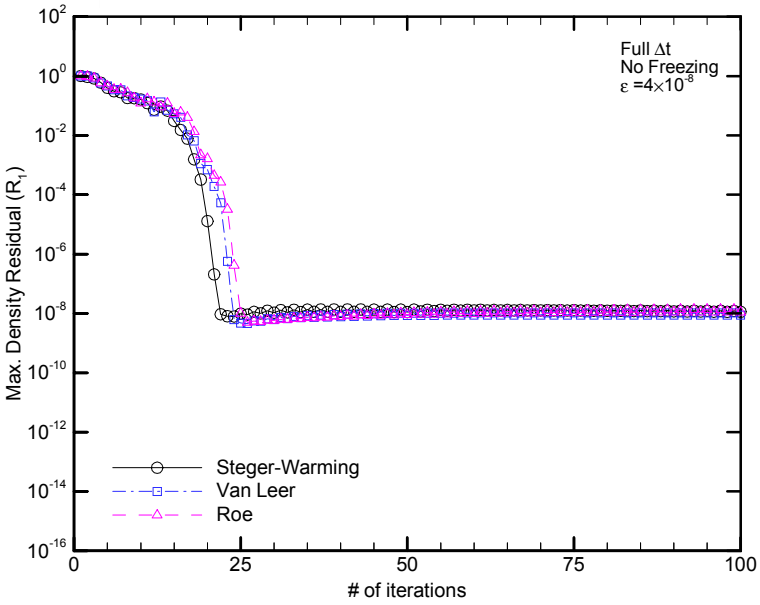
(b) 2-D axisymmetric case

Figure 5.18 Mach contours for 2nd order Roe scheme in double precision.



FS	CPU Time (s)
SW	194.9
VL	334.67
Roe	335.28

(a) 2-D planar case



FS	CPU Time (s)
SW	Limit cycle
VL	Limit cycle
Roe	Limit cycle

(b) 2-D axisymmetric case

Figure 5.19 Effect of different 2nd order flux splitting schemes on the convergence history for density residual in double precision.

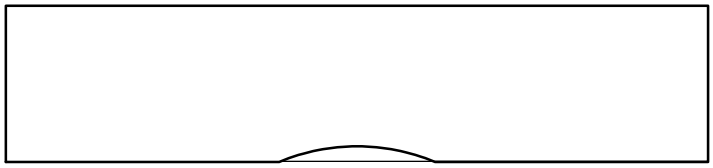
5.5 Results for Different Geometry and Flow Conditions

Until this section, all the results are obtained for the test case, 15° supersonic ramp geometry. In this part the solver is tested for another geometry and a different flow condition. A bump geometry with a finer grid can be useful to analyze since it also shows the effect of the grid size. Considering the flow conditions, since supersonic flow is used in all previous calculations, a different case can be considered as subsonic flow.

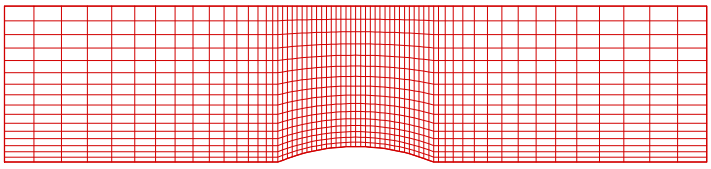
The bump geometry used has 65x17 grid as shown in Figure 5.20. First, M_∞=2.0 supersonic flow with inlet-outlet, wall and symmetry boundary conditions is analyzed. Then the same grid with M_∞=0.5 subsonic flow is considered. The second-order Roe upwinding discretization with Van Albada's limiter is employed in the calculations. The solver performance is analyzed in terms of convergence history and the required CPU time. The numerical Jacobians are calculated using forward differencing with the optimum value of ε in double precision. Both planar and axisymmetric cases are considered in the analyses. Again, no matrix freezing and full diagonal term addition is employed in the calculations as in the previous section. Since a finer grid is used with second-order discretization, it is expected to have slower convergence. Again limit-cycle convergence at higher residuals can come into picture.

Mach contours for second-order Roe flux splitting scheme are given in Figure 5.21 and Figure 5.23. Considering the supersonic case, similar behaviour of the flow with the ramp geometry is observed. Shock formation and expansion are captured with the solver. And for subsonic case, no shock forms as expected. The contours may not be so sharp since the results are not for the best converged solution.

The convergence history for second-order Roe flux is given in Figure 5.19 for supersonic flow, and in Figure 5.20 for subsonic flow. For all cases of bump geometry, the solution goes into a limit-cycle with the use of second-order discretization with Van Albada's limiter. Considering the supersonic flow, the results are much better since residual value decreases around 10^{-12} for planar flow, and 10^{-8} for axisymmetric flow. However, in subsonic flow, the residual decreases until 10^{-6} and oscillates around this value as limit-cycle. Although these results are not very bad, limit-cycle makes the solution unsteady and in order to obtain fully converged solution detailed analyses of higher-order schemes is needed.

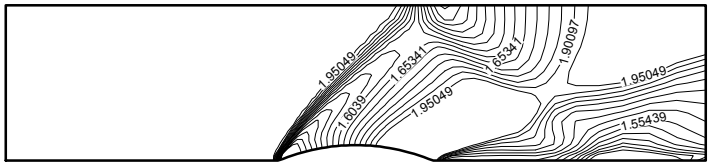


(a) General view

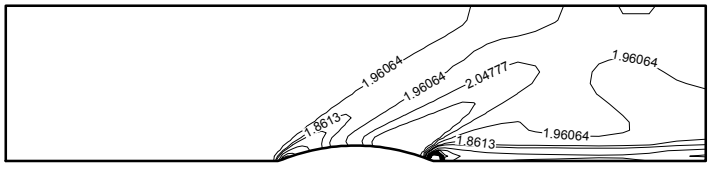


(b) 65x17 grid

Figure 5.20 Bump Geometry

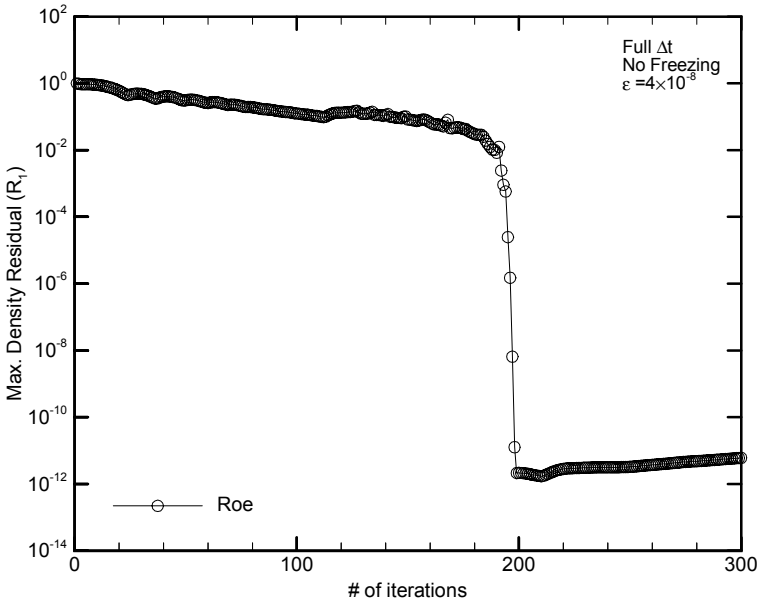


(a) 2-D planar case

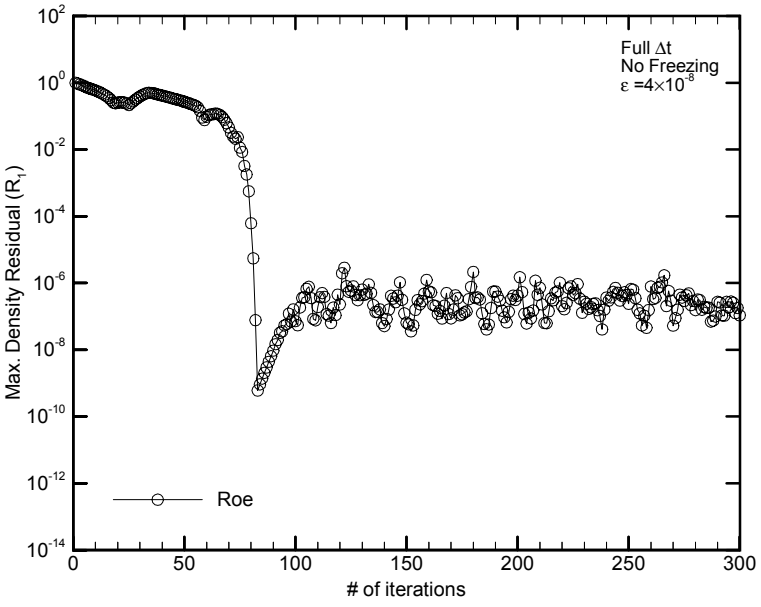


(b) 2-D axisymmetric case

Figure 5.21 Mach contours for 2nd order Roe scheme in double precision for supersonic flow.

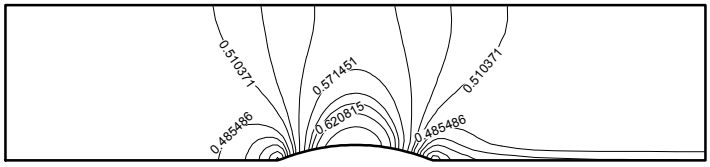


(a) 2-D planar case

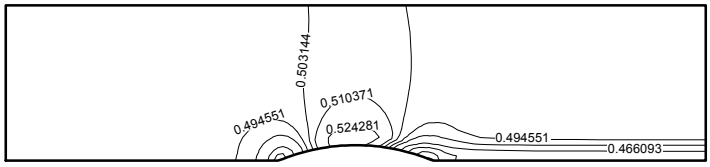


(b) 2-D axisymmetric case

Figure 5.22 Convergence history for 2nd order Roe scheme in double precision for supersonic flow

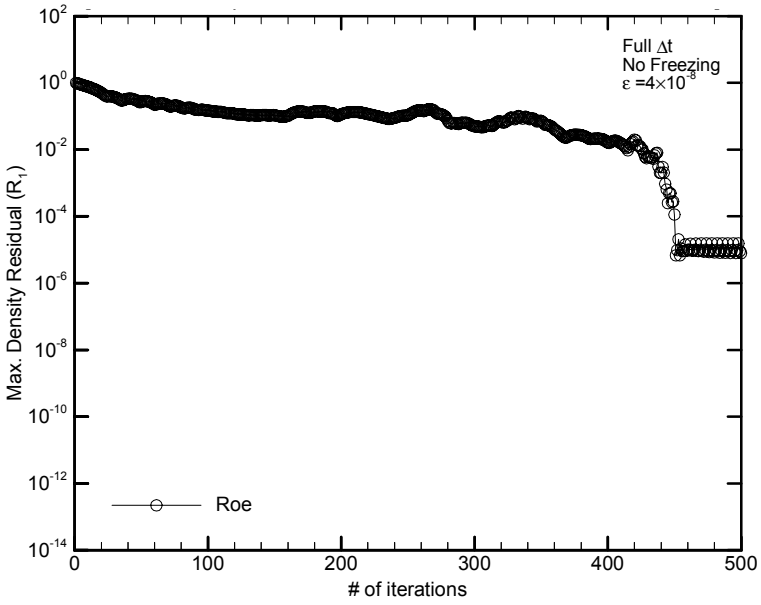


(a) 2-D planar case

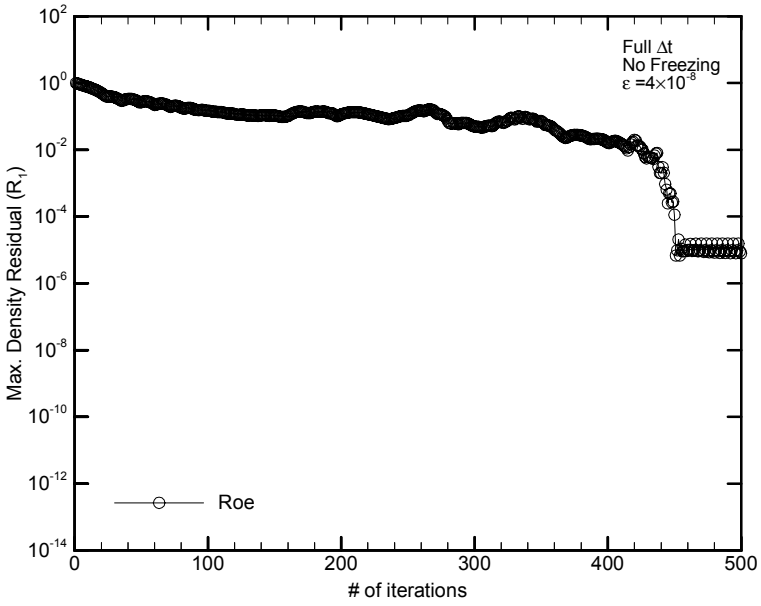


(b) 2-D axisymmetric case

Figure 5.23 Mach contours for 2nd order Roe scheme in double precision for subsonic flow.



(a) 2-D planar case



(b) 2-D axisymmetric case

Figure 5.24 Convergence history for 2nd order Roe scheme in double precision for subsonic flow

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

In this study, a direct method is developed for 2-D planar/axisymmetric Euler equations. Upwind flux splitting schemes are used in the finite-volume discretization of governing equations. The discretized nonlinear system of equations are solved using Newton's Method. The required Jacobian matrix is calculated using both analytical and numerical methods. UMFPACK sparse matrix solver is adapted to the code and used to solve the Jacobian matrix. Test calculations for accuracy and convergence are realized using first-order Steger Warming flux splitting scheme for flow over a 15^{o} supersonic ramp geometry. Effects of different flux splitting methods, higher-order discretizations and several parameters on the performance of the solver are analyzed

The first objective of this study was to analyze the accuracy of numerical Jacobians considering the effects of finite-difference perturbation magnitude and computer precision. The choice of forward or backward differencing only becomes important when the value of the perturbed flow variable is in the order of finite-difference perturbation magnitude. Thus, a control mechanism related to the choice of differencing is employed to overcome the different flux calculation problem in Steger-Warming scheme. After a detailed error analyses, it was demonstrated that the finite-difference perturbation magnitude together with computer precision is the most important parameter that affect the accuracy of numerical Jacobians. Test results showed that double precision improves the accuracy significantly decreasing the

order from 10^{-3} to 10^{-8} and with an optimum perturbation magnitude around 10^{-8} very accurate numerical flux Jacobians with an error minimized to 10^{-7} can be calculated. Also, using an optimization method it is found that, for single precision a value around $7 \cdot 10^{-4}$ and for double precision a value around $4 \cdot 10^{-8}$ can be used as optimum perturbation magnitude for all flux Jacobians. Nevertheless, the control mechanism used in first order Steger-Warming Jacobian calculation can be improved to give the best accuracy for all perturbation magnitude values.

The second objective was to investigate the effects of the accuracy of Jacobians on the performance of the direct flow solver in terms of convergence and CPU time. Calculation of the Jacobian numerically keeps the Jacobian consistent with the numerical flux vector without extremely complex or impractical analytical differentiations. Even higher-order discretizations, for which analytical derivation may be difficult to obtain, can be easily handled with the for numerical Jacobian evaluation. However, numerical method may have accuracy problem and may need longer execution time. In order to reduce this execution time, flux vectors with perturbed flow variables are calculated for only related cells. UMFPACK sparse matrix solver is used to get rid of the high storage and memory requirements of full matrix solvers. For the test case, with the optimum perturbation magnitude nearly the same convergence history with the analytical method is obtained. For 2-D planar flow, convergence is obtained in 10-11 iterations taking nearly 15 seconds of CPU time. For 2-D axisymmetric flow, convergence is obtained around 6-8 iterations taking nearly 9 seconds of CPU time. These results are very good and obtained by implementing some strategies to the Jacobian matrix solution.

Actually, the third objective was to improve the Jacobian matrix solution with some strategies. A time-like term addition to the matrix diagonal, is a very important strategy in case of poor initial guess. One important point was that this addition is required only in the first stages of the iterations, until the conditions become not poor anymore. Although a sudden withdrawal may cause oscillations, faster convergence can be achieved if the right time is chosen. Jacobian freezing is also important to

decrease the execution time. The time when to apply freezing is very important since early freezing may cause to divergence. For the test case, a detailed analysis is realized to find the best combination for diagonal-term addition and matrix freezing strategies. For the test case, quick withdrawal of the diagonal term improved the number of iterations to convergence and CPU time greatly for 2-D planar flow. For 2-D axisymmetric calculations, there are already axisymmetric terms added to the diagonal of the Jacobian matrix, providing no need for diagonal term addition. More advanced strategies can be considered to improve the Jacobian matrix solution.

The developed solver is also tested for different flux splitting methods. By this way a fourth objective, to investigate the benefits of using the same flux calculation scheme for both Jacobian and residual calculation in terms of the convergence of the solver is realized. For test case, after the Steger-Warming scheme is studied in detail, the numerical Jacobians are found from Van Leer and Roe upwinding schemes. It is observed that Van Leer or Roe residuals with Steger-Warming analytical Jacobians converge much slower than Van Leer or Roe residuals used with its corresponding numerical Jacobian. Especially, Roe reflects this well. In one case, Roe with its numerical Jacobians converged in 17 iterations taking 29.4 seconds of CPU time, while with the use of Steger-Warming analytical Jacobians convergence can be possible up to 150 iterations taking 99.62 seconds of CPU time. Thus, using the same flux calculation scheme for both Jacobian and residual calculation are very important for a faster convergence of the solver. Very good convergence results are obtained for first-order Van Leer and Roe schemes in case of numerical Jacobians are used.

Higher-order discretization schemes are also included in the study. Second-order Steger-Warming, Van Leer and Roe upwinding are used with Van Albada limiter. Different from the previous work done in the study, the convergence problem is clearly seen in higher-order schemes. For 2-D axisymmetric flow, the solution goes into a limit-cycle with the residual value around 10^{-8}, and does not decrease anymore. This may be due to the nature of the flux limiter employed in the solver. Choice and application of limiters is another area that must be analyzed in detail.

Finally, different from the test case, bump geometry with a finer grid is analyzed considering a subsonic flow as a different case than supersonic flow. The second-order Roe upwinding scheme with Van Albada's limiter is used in the calculations. For all cases of bump geometry, the solution goes into a limit-cycle. Considering the supersonic flow, the results are much better since residual value decreases around 10^{-12} for planar flow, and 10^{-8} for axisymmetric flow. In subsonic flow, the residual oscillates around 10^{-6}. Limit-cycle makes the solution unsteady and in order to obtain fully converged solution implementation of another limiter or considering a totally different procedure for higher-order schemes have to be considered.

REFERENCES

[1] Wigton, L. B., ”Application of MACSYMA and Sparse Matrix Technology to Multi-element Airfoil Calculations”, AIAA Paper 87-1142, 1987.

[2] Bender, E.E., and Kosla, P.K., ”Application of Sparse Matrix Solvers and Newton’s Method to Fluid Flow Problems”, AIAA Paper 88-3700, 1988.

[3] Venkatakrishnan, V., “Newton Solution of Inviscid and Viscous Problems”, *AIAA Journal*, Vol. 27, July 1989, pp. 885-891.

[4] Orkwis, P. D., *A Newton’s Method Solver for the Two-Dimensional and Axisymmetric Navier-Stokes Equations*, Ph.D. Dissertation, North Carolina State University, Raleigh, NC, 1990.

[5] Orkwis, P. D., and McRae, D. S., ”Newton’s Method Solver for High-Speed Viscous Separated Flowfields”, *AIAA Journal*, Vol. 30, January 1992, pp. 78-85.

[6] Orkwis, P. D., and McRae, D. S., ”Newton’s Method Solver the Axisymmetric Navier-Stokes Equations”, *AIAA Journal*, Vol. 30, January 1992, pp. 1507-1514.

[7] Orkwis, P. D., ”Comparison of Newton’s and Quasi-Newton’s Method Solvers for the Navier-Stokes Equations”, *AIAA Journal*, Vol. 31, May 1993, pp. 832-836.

[8] Kim, D. B., and Orkwis, P. D., ”Jacobians Update Strategies for Quadratic and Near-Quadratic Convergence of Newton and Newton-Like Implicit Schemes”, AIAA Paper 93-0878, Proceedings of the AIAA 31st Aerospace Sciences Meeting & Exhibit, Reno, Nevada, January 1993.

[9] Felker, F. F., *Direct Solutions of the Navier-Stokes Equations With Application to Static Aeroelasticity*, Ph.D. Dissertation, Stanford University, 1992.

[10] Whitfield, D. L., and Taylor, L. K., ”Discretized Newton-Relaxation Solution of High Resolution Flux-Difference Split Schemes”, AIAA Paper 91-1539, 1991.

[11] Vanden, K. J.*, Direct and Iterative Algorithms for the Three-Dimensional Euler Equations*, Ph.D. Dissertation, Mississippi State University, Mississippi, 1992.

[12] Vanden, K. J., and Whitfield, D. L., ”Direct and Iterative Algorithms for the Three-Dimensional Euler Equations”, AIAA Paper 93-3378, 1993.

[13] Orkwis, P. D., and Vanden, K. J., ”On the Accuracy of Numerical versus Analytical Jacobians”, AIAA Paper 94-0176, Proceedings of the AIAA 32nd Aerospace Sciences Meeting, Reno, Nevada, January 1994.

[14] Aberle, C., and Schumlak, U., ”Application of Analytical Methods to Computing Numerical Flux Jacobians”, AIAA Paper 01-31093, Proceedings of the AIAA 15th Computational Fluid Dynamics Conference, Anahelm, California, June 2001.

[15] Hoffmann, A. K. and Chiang, S. T., *Computational Fluid Dynamics for Engineers,* Vol. I-II, Publication of Engineering System, Austin, Texas 78713.

[16] Fletcher, C. A. J., *Computational Techniques for Fluid Dynamics*, Vol. I-II, Springer-Verlag, 1987.

[17] Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods", *Journal of Computational Physics*, Vol. 40, 1981, pp. 263-293.

[18] Van Leer, B., "Flux Vector Splitting for the Euler Equations", ICASE Report 82-30, September 1982.

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

[20] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. I-II, John Wiley & Sons, 1990.

[21] *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing*, Cambridge University Press, 1992.

[22] Davis, T. A., *UMFPACK Version 4.1 User Manual*, University of Florida, Florida, 2003.

[23] Dennis, J. E., and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Non-linear Equations*, Prentice-Hall, Eaglewood Cliffs, New Jersey, 1983.

[24] Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis*, Third Edition, Addison-Wesley, 1984.

APPENDIX A

ANALYTICAL FLUX JACOBIANS FOR STEGER-WARMING FLUX VECTOR SPLITTING SCHEME

The generalized flux vector of 2-D Euler equations in generalized coordinates can be written as:

$$\mathbb{F} = \frac{\rho}{2\gamma}\begin{bmatrix} 2(\gamma-1)\lambda_1 + \lambda_3 + \lambda_4 \\ 2(\gamma-1)\lambda_1 u + \lambda_3(u + c\tilde{k}_1) + \lambda_4(u - c\tilde{k}_1) \\ 2(\gamma-1)\lambda_1 v + \lambda_3(v + c\tilde{k}_2) + \lambda_4(v - c\tilde{k}_2) \\ (\gamma-1)\lambda_1(u^2+v^2) + \frac{\lambda_3}{2}((u + c\tilde{k}_1)^2 + (v + c\tilde{k}_2)^2) \\ + \frac{\lambda_4}{2}((u - c\tilde{k}_1)^2 + (v - c\tilde{k}_2)^2) + \frac{(3-\gamma)(\lambda_3+\lambda_4)c^2}{2(\gamma-1)} \end{bmatrix} \tag{A.1}$$

where the speed of sound c, the eigenvalues λ_i and the directional cosines $\tilde{k}_i$ are defined as:

$$c = \sqrt{\gamma(\gamma-1)\left(e_t - \frac{1}{2}(u^2+v^2)\right)}$$

$$\lambda_1 = uk_1 + vk_2 \qquad \tilde{k}_1 = \frac{k_1}{\sqrt{k_1^2 + k_2^2}}$$

$$\lambda_3 = \lambda_1 + c\sqrt{k_1^2 + k_2^2} \qquad \tilde{k}_2 = \frac{k_2}{\sqrt{k_1^2 + k_2^2}}$$

$$\lambda_4 = \lambda_1 - c\sqrt{k_1^2 + k_2^2}$$

Then taking derivative of each flux $\mathbb{F}_i$ with respect to each flow variable W_j, flux Jacobians $\mathbb{F}_{ij}$ are obtained as follows:

$$\begin{aligned}
\mathbb{F}_{11} &= \frac{\rho}{2\gamma}\left(2(\gamma-1)\lambda_{11}+\lambda_{31}+\lambda_{41}\right)+\frac{\mathbb{F}_1}{\rho} \\
\mathbb{F}_{12} &= \frac{\rho}{2\gamma}\left(2(\gamma-1)\lambda_{12}+\lambda_{32}+\lambda_{42}\right) \\
\mathbb{F}_{13} &= \frac{\rho}{2\gamma}\left(2(\gamma-1)\lambda_{13}+\lambda_{33}+\lambda_{43}\right) \\
\mathbb{F}_{14} &= \frac{\rho}{2\gamma}\left(2(\gamma-1)\lambda_{14}+\lambda_{34}+\lambda_{44}\right)
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
\mathbb{F}_{21} &= u\left(\mathbb{F}_{11}-\frac{\mathbb{F}_1}{\rho}\right)+\tilde{k}_1\frac{\rho}{2\gamma}\left(c_1(\lambda_3-\lambda_4)+c(\lambda_{31}-\lambda_{41})+\frac{c}{\rho}(\lambda_3-\lambda_4)\right) \\
\mathbb{F}_{22} &= u\mathbb{F}_{12}+\frac{\mathbb{F}_1}{\rho}+\tilde{k}_1\frac{\rho}{2\gamma}\left(c_2(\lambda_3-\lambda_4)+c(\lambda_{32}-\lambda_{42})\right) \\
\mathbb{F}_{23} &= u\mathbb{F}_{13}+\tilde{k}_1\frac{\rho}{2\gamma}\left(c_3(\lambda_3-\lambda_4)+c(\lambda_{33}-\lambda_{43})\right) \\
\mathbb{F}_{24} &= u\mathbb{F}_{14}+\tilde{k}_1\frac{\rho}{2\gamma}\left(c_4(\lambda_3-\lambda_4)+c(\lambda_{34}-\lambda_{44})\right)
\end{aligned} \tag{A.3}$$

$$\begin{aligned}
\mathbb{F}_{31} &= v\left(\mathbb{F}_{11}-\frac{\mathbb{F}_1}{\rho}\right)+\tilde{k}_2\frac{\rho}{2\gamma}\left(c_1(\lambda_3-\lambda_4)+c(\lambda_{31}-\lambda_{41})+\frac{c}{\rho}(\lambda_3-\lambda_4)\right) \\
\mathbb{F}_{32} &= v\mathbb{F}_{12}+\tilde{k}_2\frac{\rho}{2\gamma}\left(c_2(\lambda_3-\lambda_4)+c(\lambda_{32}-\lambda_{42})\right) \\
\mathbb{F}_{33} &= v\mathbb{F}_{13}+\frac{\mathbb{F}_1}{\rho}+\tilde{k}_2\frac{\rho}{2\gamma}\left(c_3(\lambda_3-\lambda_4)+c(\lambda_{33}-\lambda_{43})\right) \\
\mathbb{F}_{34} &= v\mathbb{F}_{14}+\tilde{k}_2\frac{\rho}{2\gamma}\left(c_4(\lambda_3-\lambda_4)+c(\lambda_{34}-\lambda_{44})\right)
\end{aligned} \tag{A.4}$$

$$\mathbb{F}_{41} = \frac{1}{2}(u^2+v^2)\left(\mathbb{F}_{11} - \frac{2\mathbb{F}_1}{\rho}\right) + \frac{\rho}{2\gamma}(u\tilde{k}_1 + v\tilde{k}_2)\left(c_1(\lambda_3 - \lambda_4) + c(\lambda_{31} - \lambda_{41})\right)$$
$$+ \frac{\rho c}{\gamma(\gamma-1)}\left(c_1(\lambda_3+\lambda_4) + \frac{c}{2}(\lambda_{31}+\lambda_{41}) + \frac{c}{2\rho}(\lambda_3+\lambda_4)\right)$$
$$\mathbb{F}_{42} = \frac{1}{2}(u^2+v^2)\mathbb{F}_{12} + u\frac{\mathbb{F}_1}{\rho} + \frac{\rho}{2\gamma}(u\tilde{k}_1 + v\tilde{k}_2)\left(c_2(\lambda_3 - \lambda_4) + c(\lambda_{32} - \lambda_{42})\right)$$
$$+ \frac{\rho c}{\gamma(\gamma-1)}\left(c_2(\lambda_3+\lambda_4) + \frac{c}{2}(\lambda_{32}+\lambda_{42})\right) + \tilde{k}_1\frac{c}{2\gamma}(\lambda_3 - \lambda_4)$$
$$\mathbb{F}_{43} = \frac{1}{2}(u^2+v^2)\mathbb{F}_{13} + v\frac{\mathbb{F}_1}{\rho} + \frac{\rho}{2\gamma}(u\tilde{k}_1 + v\tilde{k}_2)\left(c_3(\lambda_3 - \lambda_4) + c(\lambda_{33} - \lambda_{43})\right)$$
$$+ \frac{\rho c}{\gamma(\gamma-1)}\left(c_3(\lambda_3+\lambda_4) + \frac{c}{2}(\lambda_{33}+\lambda_{43})\right) + \tilde{k}_2\frac{c}{2\gamma}(\lambda_3 - \lambda_4)$$
$$\mathbb{F}_{44} = \frac{1}{2}(u^2+v^2)\mathbb{F}_{14} + \frac{\rho}{2\gamma}(u\tilde{k}_1 + v\tilde{k}_2)\left(c_4(\lambda_3 - \lambda_4) + c(\lambda_{34} - \lambda_{44})\right)$$
$$+ \frac{\rho c}{\gamma(\gamma-1)}\left(c_4(\lambda_3+\lambda_4) + \frac{c}{2}(\lambda_{34}+\lambda_{44})\right) \qquad \text{(A.5)}$$

where the derivatives of speed of sound c and eigenvalues λ_i with respect to flow variables W_j, being c_j and λ_{ij} respectively are defined as:

$$c_1 = \frac{\gamma(\gamma-1)}{2\rho c}(u^2+v^2+e_t) \qquad \lambda_{11} = -\frac{\lambda_1}{\rho}$$
$$c_2 = -\frac{\gamma(\gamma-1)u}{2\rho c} \qquad \lambda_{12} = \frac{k_1}{\rho}$$
$$c_3 = -\frac{\gamma(\gamma-1)v}{2\rho c} \qquad \lambda_{13} = \frac{k_2}{\rho}$$
$$c_4 = -\frac{\gamma(\gamma-1)}{2\rho c} \qquad \lambda_{14} = 0$$

$$\lambda_{31} = -\frac{\lambda_1}{\rho} + c_1\sqrt{k_1^2+k_2^2} \qquad \lambda_{41} = -\frac{\lambda_1}{\rho} - c_1\sqrt{k_1^2+k_2^2}$$
$$\lambda_{32} = \frac{k_1}{\rho} + c_2\sqrt{k_1^2+k_2^2} \qquad \lambda_{42} = \frac{k_1}{\rho} - c_2\sqrt{k_1^2+k_2^2}$$
$$\lambda_{33} = \frac{k_2}{\rho} + c_3\sqrt{k_1^2+k_2^2} \qquad \lambda_{43} = \frac{k_2}{\rho} - c_3\sqrt{k_1^2+k_2^2}$$
$$\lambda_{34} = c_4\sqrt{k_1^2+k_2^2} \qquad \lambda_{44} = -c_4\sqrt{k_1^2+k_2^2}$$

The ξ-directional flux vector $\hat{F}$ and its Jacobian matrices $\hat{A}$ can be obtained from Equation (A.1) to Equation (A.5) by replacing k_1 and k_2 with ξ_x and ξ_y, and the η-directional flux vector $\hat{G}$ and its Jacobian matrices $\hat{B}$ can be obtained by replacing k_1 and k_2 with η_x and η_y, respectively.

The positive and negative flux vectors and their Jacobian matrices are obtained from Equation (A.1) to Equation (A.5) by substituting all λ's by λ^{+}'s and λ^{-}'s respectively. Here, $\lambda_i^{\pm}$ and its derivative with respect to flow variables W_j, $\lambda_{ij}^{\pm}$ are defined as:

$$\lambda_i^{\pm} = \frac{\lambda_i \pm |\lambda_i|}{2} \qquad \lambda_{ij}^{\pm} = \lambda_{ij}\left(\frac{1 \pm sign(\lambda_i)}{2}\right)$$