PERFORMANCE ANALYSES OF NEWTON METHOD FOR MULTI-BLOCK STRUCTURED GRIDS

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

BY

ERDEM AYAN

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

SEPTEMBER 2011

Approval of the thesis:

PERFORMANCE ANALYSES OF NEWTON METHOD FOR MULTI-BLOCK STRUCTURED GRIDS

submitted by **ERDEM AYAN** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _______________
Dean, Graduate School of **Natural and Applied Science**

Prof. Dr. Ozan Tekinalp _______________
Head of Department**, Aerospace Engineering**

Asoc. Prof. Dr. Sinan Eyi _______________
Supervisor**, Aerospace Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Cevdet Çelenligil _______________
Aerospace Engineering Dept., METU

Asoc. Prof. Dr. Sinan Eyi _______________
Aerospace Engineering Dept., METU

Asoc. Prof. Dr. Dilek Funda Kurtuluş _______________
Aerospace Engineering Dept., METU

Asst. Prof. Dr. Oğuz Uzol _______________
Aerospace Engineering Dept., METU

Göktan Güzel, MSc. _______________

Date:

ABSTRACT

PERFORMANCE ANALYSES OF NEWTON METHOD FOR MULTI-BLOCK STRUCTURED GRIDS

Ayan, Erdem
M.S., Department of Aerospace Engineering
Supervisor: Assoc. Prof. Dr. Sinan Eyi
September 2011, 76 pages

In order to make use of Newton's method for complex flow domains, an Euler multi-block Newton solver is developed. The generated Newton solver uses Analytical Jacobian derivation technique to construct the Jacobian matrices with different flux discretization schemes up to the second order face interpolations. Constructed sparse matrices are solved by parallel and series matrix solvers. In order to use structured grids for complex domains, multi-block grid construction is needed. Each block has its own Jacobian matrices and during the iterations the communication between the blocks should be performed. Required communication is performed with "halo" nodes. Increase in the number of grids requires parallelization to minimize the solution time. Parallelization of the analyses is performed by using matrix solvers having parallelization capability. In this thesis, some applications of the multi-block Newton method to different problems are given. Results are compared by using different flux discretization schemes. Convergence, analysis time and matrix solver performances are examined for different number of blocks.

Keywords: Multi-Block Newton Method, Flux Jacobian, CFD

ÖZ

ÇOK BLOKLU YAPISAL AĞ SİSTEMİ İÇİN NEWTON YÖNTEMİNİN PERFORMANS ANALİZİ

Ayan, Erdem
Yüksek Lisans, Havacilik ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Sinan Eyi
Eylül 2011, 76 sayfa

Bu tezde, kompleks geometrilerin akış çözümünde kullanılmak üzere geliştirilmiş olan çok bloklu Newton metod çözücüsünün performans analizi anlatılmaktadır. Euler denklemlerinin çözümü için gerekli olan Jacobian matrislerinin oluşturulmasında farklı akış ayrıklaştırma teknikleri ile birlikte analitik türetme yöntemi kullanılmıştır. Oluşturulan seyrek matrisler, matris çözücülerinin yardımıyla paralel ve seri olarak çözülmüştür. Çözücünün çok bloklu hale getirilmesiyle birlikte her bir blok için ayrı bir Jacobian matrisi analitik türetme yöntemi ile oluşturulmuş ve her bir iterasyonda bloklar arasındaki gerekli olan iletişim "sanal" noktalar kullanımı ile gerçekleştirilmiştir. Geliştirilen çözücü farklı tipteki problemlere uyarlanmıştır. Farklı sayıda blok kullanımıyla birlikte yakınsama durumu ve analiz süresi performansı çeşitli büyüklüklerdeki akış problemleriyle değerlendirilmiştir. Ayrıca farklı matris çözücülerinin de kendi aralarında çözüm süreleri ve doğrulukları açısından değerlendirmeler yapılmıştır.

Anahtar Kelimeler: Çok Bloklu Newton Metod, Akı Jacobianı, Hesaplamalı Akışkanlar Dinamiği

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asoc. Prof. Dr. Sinan Eyi for his guidance support, encouragement and patience throughout the study.

I also would like to thank Mr. Yüksel Ortakaya, Chief of the Aeromechanics Department in TAI for his technical support, understanding and tolerance.

I would like to thank to all members of işçet group and to my TAI colleagues Ahmet Alper Ezertaş, Aykut Tamer, Ceren Öztürk, Özge Polat, Eda Toprakkale and Halil Kaya for their moral support and encouragements.

Special thanks to my friend Eray Göksu for supporting me all the time and all the things we have shared.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

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
et	Total energy per unit volume
f(x)	A function of variable 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
M	Mach number
p	Pressure
QΛ	Right eigenvector matrix of A, B
r	Ratio of differences
R	Residual vector of the system
Rax	Axisymmetric case residual value
Rfrz	Freezing residual value
Rpl	Planar case residual value
Δt	Time-like term added to Jacobian matrix diagonal
Δt0	Initial value for Δt
Δtf	Removal value for Δt
u,v	Velocity components

U,V	Contravariant velocity components
U	Contravariant velocity components in terms of k
x,y	Components of Cartesian coordinates

GREEK SYMBOLS

α	Factor to reduce W
Δ	Forward difference operator
ε	Finite-difference perturbation magnitude
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

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 became one of the branches of aerodynamics which complements the experimental and theoretical approaches since late 1970s. It immediately became one of the tools to model the fluid flow in industry. Its usage is increasing everyday and it became an important part of the design and analyses processes for more and more companies. It is possible to model and examine velocity, temperature, pressure and species included in fluid flow within a solution domain with CFD.

Since the analysis requires numerical methods and algorithms, defining and modeling the fluid flow and solving the modeled flow is a difficult task generally and for some cases it is impossible without using computers. Before introducing with CFD; scientists and engineers use some empirical methods or experiments. These experiments generally refer to the wind tunnel tests. However, usage of empirical methods is possible only for similar cases which are investigated before. They can provide only a very limited accuracy. On the other hand, using wind tunnel tests is not possible for most of the situations due to the high cost and huge amount of cases. Most of the aerospace companies use wind tunnel tests when CFD is inadequate to solve the problem or when in their detailed design processes for very limited number of cases. Increasing capacity of aerospace industry requires detailed analyzes of aircrafts with high accuracy and CFD tries to fulfill this requirement.

Fast, robust and accurate methods & algorithms are required to solve aerodynamic flows. The flow solver must have capability to handle a variety of the flow conditions with different configurations. Several methods are presented to balance the speed and robustness of the solver. The main problem is to find the suitable method and an adequate model to solve the fluid flow to satisfy the requirements.

1.2 Background

Computational fluid dynamics is used to model the problems and to find solutions to the partial differential equations on a computational domain. Required calculations are performed in the nodes within the domain. These nodes can be arranged by structured or unstructured orders. Both of them have advantages and disadvantages. Structured grid includes distributed nodes in an organized pattern around the geometry. They require lower storage requirement and they work well for simple configurations. However, it is not always possible to construct structured grid on the flow domain for complex geometries. For these cases, unstructured grids can fit the geometry. Moreover, they require less complicated grid generation techniques and higher adaptability to the flow solution. When the usage of single block structured grid is impossible, in order to make use of less overhead and lower storage feature of structured grid, multi-block approach is developed.

CFD solvers can be divided into two parts according to the time-marching methods namely, explicit and implicit methods. Explicit methods are simple and computationally easy. However, they are not stable when higher time steps are used. Hence, they require more iteration for convergence. Using multi-grid techniques decreases the required iteration for convergence. Implicit methods, on the other hand, are not easy to implement. Moreover, they require higher computational time per each iteration. However, the total required number of iteration decreases for convergence. In CFD one of the fastest implicit methods is the Newton method.

Creating fast and efficient flow solver is limited to the time required to solve large linear system of equations. Direct inversion of these matrices is not possible for some cases or very expensive up to the end of 90's. For this reason, one of the used methods in this field is approximate-Newton method and the other one is Jacobian free inexact Newton method. They all include iterative methods to inexactly solve the systems of linear equations. They reduce required time at each iteration.

With increasing capability of computers, due to quadratic convergence rate Newton's method becomes again a powerful method. However, it requires exact linearization of the residual equations to satisfy the quadratic convergence property. Solving the flow domain requires the calculation of Jacobian matrix which includes derivatives of the residual function with respect to the flow variables. Depending on the constructed grid; the Jacobian matrix may be very large. Reducing size of the Jacobian matrix can be done by using multi-block grids. However, in this case the number of the Jacobian matrices increases and another discussion appears about the fastest and robustness of the increasing number of blocks with reducing sizes. There are several methods for performing multi-block analyses. The main problem is the data transfers between the blocks. The most common ways are using "halo" nodes or evaluating simultaneously approaching terms (SAT).

The other problem is the derivation of the entries in the Jacobian matrix. They can be evaluated by analytical or numerical means. Analytical derivation of the entries becomes more difficult when the discretization of the equations become more complex. Numerical derivation, on the other hand, is simpler but it cannot guarantee higher accuracy without some error analyses as preliminary studies. The methods used to deal with these problems directly influence the efficiency and usage of the Newton's method.

1.3 Objectives

The main objective of this thesis is to use multi-block approach with Newton method for 2-D Euler equations. Performance analyses are aimed to be done to evaluate the abilities of the generated code. Corresponding performance parameters are accuracy, robustness and required computational time for convergence. Analysis time comparison is performed for varying numbers of blocks in the domain. One block and multi-block grids are constructed and analyzed with same solver parameters to compare accuracy. Moreover, flow variables are examined around the block interfaces to evaluate the implemented block interface boundary conditions. Since modeling the whole domain is impossible with one block for complex geometries, another objective appears. With multi-block approach, Newton method can be used also for complex geometries. In addition, reduction of the computational time is aimed by making parallelization of the analyses. The other objective is to examine the usage of the Newton's method with the direct sparse matrix solvers. PARDISO and UMFPACK sparse matrix solvers are used in the study.

1.4 Literature Survey

Several researchers used Newton method in their studies for modeling fluid flow. Generally they chose Newton Method due to the high convergence rate and high accuracy.

Wington [1] used Newton's method in his study. The article is about modeling fluid flow around multi-element airfoil. He analyzed multi-element airfoil in transonic flows. He used Newton's method to make use of the quadratic convergence property of the method. In order to solve the Jacobian matrix, Symbolic manipulation expert system MACSYMA, symbolic derivation tool, was used in his study. While solving the large linear sparse systems, to reduce the huge storage requirements and great factorization time, he generated the nested dissection node reordering technique, which is being used for some of the sparse matrix solvers today.

Bender and Khosla [2] investigated the Newton's method for the solution of the viscid-inviscid compressible flows. They worked on initial conditions of Newton's method to increase convergence rate and prevent early divergence. Two modifications are presented to reduce the sensitivity of the initial guess at transonic Mach numbers. For such problems, they found that applied exact Newton method exhibits high sensitivity to initial conditions.

Venkatakrishnan [3] used Newton's method to compute viscous flows in a robust manner. In his study, quadratic convergence is realized by using exact linearization with Roe scheme. He showed that within 3-4 iterations the steady solution results are obtained and the convergence rates are independent to the Reynolds number.

Van Dam et al.[4] used fully-implicit technique to combine direct solution technique based on banded Gauss elimination with Newton's method for laminar incompressible flows. Orkwis [5] developed a solver using Newton's method to solve 2-D & axisymmetric [6] and laminar & turbulent flows [7]. In his studies; Navier-Stokes equations are approximated by flux difference splitting methods with Glaister's approach. With the addition of the geometric conservation law, the freestream reproduction is satisfied for the axisymmetric solver. Van Albada limiter is used to reduce spurious oscillations. As in the Wington's study MACSYMA is used to determine Jacobian Matrices entries.

Orkwis [8] made performance comparisons of the exact and quasi Newton methods. He found that in spite of not having quadratic convergence rate, quasi Newton methods are more efficient than the exact method which has quadratic convergence in terms of CPU usage. In his another work which is with Kim [9], Orkwis showed that with matrix simplifications like partial and global freezing methods, approximate methods can also have quadratic convergence rate.

Whitfield and Taylor [10] presented numerical methods to evaluate jacobian matrices for the cases at which obtaining the Jacobian matrices entries are impractical with analytical method. They applied this approach both for compressible and incompressible flows with high order ROE discretizations.

Vanden [11], [12] developed direct and iterative algorithms to solve a finite volume discretization of the 3-D Euler equations in curvilinear coordinates. He showed in his study that the iterative algorithms superior in terms of time required for completion of the analyses.

Vanden and Orkwis [13] made performance comparison of analytical and numerical Jacobian matrices in exact Newton method. MACSYMA is used for analytical Jacobians. On the other hand finite differencing is used for numerical Jacobians. They showed in their study that the convergence performance of both matrices is same. They stated that for simpler cases when the linear systems of equations are simple; analytical evaluation can be chosen but for the complex schemes while performing the linearization of the equations numerical evaluation will be the better choice.

Saad and Schultz [14] presented an iterative method for solving linear systems in their studies. Approximate Newton's method by applying GMRES with first order Jacobian approximations are investigated by Venkatakrishnan [15], Mavriplis [16] and Rogers [17].

Forstyh and Jiang [18] made comparisons for the quasi Newton methods and made simplifications on the Jacobian matrices. They found that, despite of the expensive pre-conditioner, an inexact Newton method is more effective than approximate methods.

Brown and Saad [19] analyzed inexact and approximate methods when they are combined with linesearch techniques and model trust region algorithms. Their method does not require Jacobian matrix storage and it accurate linearization of the residual function is performed with no storage limit.

Researchers in this field are focused on quasi Newton methods in last decade due to the difficulties in exact Newton's method. After nearly ten silence years; researchers again started to use exact Newton's method in their studies due to the improvements in sparse matrix solvers with advanced algorithms. One of the most common sparse matrix solver is introduced by Davis [20] namely UMFPACK. MUMPS, another one, is introduced by Amestoy and Duff [21] for distributed memory parallel usage. Later, WSMP and PARDISO developed by

Gupta [22] and Schenk [23]. Usage of these matrix solvers is common among the researchers.

Eyi and Onur [24] used UMFPACK as sparse matrix solver with exact Newton method. They analyzed inviscid supersonic flow problem on ramp geometry. They made analytical and numerical Jacobian comparison. Gelfgat [25] used MUMPS to solve sparse matrices and T'ien and Raju [26] evaluated the capabilities of multifrontal solvers. In their combustion problem, they used UMFPACK and demonstrated that usage of direct matrix solvers decreases the required computational time. Eyi and Ezertas [27] examined the usage of the exact Newton's method with sparse matrix solvers. They used UMFPACK multi-frontal solver while making comparison of the numerical and analytical Jacobian calculations.

In order to use structured grids around the complex geometries, researchers made multi-block analyses in their studies. Nichols and Zingg [28] developed a three-dimensional muli-block Newton-Krylov Euler solver with GMRES subspace algorithm to solve linear system of equation. The multi-block strategy they followed facilitates the treatmeant of arbitrarily complex geometries with point to point matching at their interfaces. Kam [29] developed similar one with Spalart-Allmaras turbulence model, Rumpfkeil [30] made airfoil optimization for unsteady flows and Nemec [31] made shape design of aerodynamic configuration with multi-block Newton-Krylov tool. All of these researchers have used halo nodes at block interfaces in their studies.

Hicken [32] uses summation-by-parts (SBP) technique with simultaneous-approximation-terms in his thesis and showed that by developing a scalable preconditioner one can make use of advantage of the SBP-SAT discretization over using halo nodes in the block interfaces. Leung [33] also used simultaneously approaching terms in his thesis and performed parallel aerodynamic shape optimization in three dimensions and he discussed the performance of the data transfer between the different blocks via simultaneously approaching terms. Finally, Huan, Hicken and Zingg [34] used two different schemes for the interfaces and boundary schemes. In the first scheme they use standard difference operators up to third-order global accuracy and special near-boundary operators to preserve stability for fifth-order global accuracy. In the second scheme they used summation-by-parts with simultaneous-approximation-terms and they showed that the interface schemes that do not involve halo nodes offer several advantages when the error introduced at mesh interfaces are compared.

1.5 Outline

In Chapter 2; governing flow equations are introduced with discussions of its basic theories. Different discretization techniques which are used in this study are described with required formulations. Implemented boundary conditions with block interface applications for multi-block grids are explained for different flow regimes.

In Chapter 3; exact Newton method is presented with advantageous and disadvantageous. Analytical and numerical derivation methods of the Jacobian matrices are given with discussions on their usage comparisons. The way of the implementation of the initial conditions and boundary conditions into the Jacobian matrices are explained. Verification of the generated program is made by performing airfoil and channel flow analyses. Since the generated program is inviscid, verifications are made by comparison with other authors' studies in literature.

In Chapter 4; performance analyses results of the multi-block Newton method are given. A nozzle problem and a channel flow problem are used for performance analyses. First, successful implementations of the block interface boundary conditions are shown. The flow domain is divided into different number of blocks for both of the problem and their performance analyses results are presented in terms of accuracy, iterations required for convergence and required CPU time. Then, UMFPACK and PARDISO sparse matrix solvers are compared. The effect of the diagonal addition term (Δt) to the convergence of the flow solution is analyzed and the variation of the CPU time spent is tabulated.

CHAPTER 2

GOVERNING FLOW EQUATIONS

2.1 Introduction

Prepared flow model requires retaining the fluid characteristics of the flow conditions. This flow model should have advanced spatial and time discretization techniques to satisfy the needs. It is vital to decide the flow model properties before solving the problem. Two main problems arise at this point. Required accuracy and cost determination studies identifies the flow model properties. Required accuracy depends on the aim of the modeled problem and the analysis which is performed. Actually, accuracy and the cost terms depend on each other. The accurate solution for complex flow problems needs advanced strategies on computational meshes and high resolution on grids and this can cause increase in the cost. Cost determination study generally referred as the required time to solve the problem. The cost of the analyzed problem increases with the increase in the level of scheme complexity. With simpler methods the cost of the problem can be decreased however the required accurate results may not be reached.

In this study; implicit direct solution is used for time discretization which is accepted as computationally expensive. Flux jacobian matrix is used in this direct solution technique. Number of flow equations with the grid size determines the size of the matrix and the computational cost of the problem. In order to decrease the computational cost, flow is modeled with two dimensional Euler

equations. Structured grids are used for the modelling of the flow field. Using structured grids makes easier to construct Jacobian matrices and makes easier to store the required data of the problem.
Problems which have complex geometries cannot be modeled with one grid block and required multi-block grids to solve the flow. In this study; analyses of the complex geometries are performed by using multi-block grids. Data transfers between the blocks are achieved by using halo nodes on the block boundaries.

For discretization of the equations the Steger-Warming [35], Van Leer [36], AUSM [37] and Roe [38] upwind schemes are used. Moreover, for the second order accurate analysis, MUSCL [39] interpolation is used with the required of Van Albada's [40] and Venkatakrishnan's [41] continuous limiters.

2.2 Governing Equations

Mass, momentum and energy conservations construct the basis of the flow analysis. These conservations can be presented by Navier-Stokes equations. For a two-dimensional inviscid flow with density ρ, velocities (u, v) in Cartesian coordinates (x, y), pressure p and total energy e_t, the equations are given by;

$$\frac{\partial \mathrm{Q}}{\partial \mathrm{t}}+\frac{\partial \mathrm{F}}{\partial \mathrm{x}}+\frac{\partial \mathrm{G}}{\partial \mathrm{y}}=0 \tag{2.1}$$

where Q is the vector of conservative variables; F and G are the inviscid convective fluxes and the first term drops when the problem is steady.

When they are written in detailed form;

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_t \end{bmatrix} \quad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(\rho e_t + p) \end{bmatrix} \quad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(\rho e_t + p) \end{bmatrix} \tag{2.2}$$

Pressure can be calculated from the ideal gas relation as;

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

2.3 Coordinate Transformation

For easily implementation of the numerical algorithms on an arbitrary geometry, the governing equation in the physical domain must be transformed to the computational domain. A generalized coordinate transformation is used to map the curvilinear structured grids into square grids. Figure 2-1 shows the 2-D transformation of the physical domain to computational domain. Coordinates of the physical domain are x, y and the coordinates of the computational domain are $\boldsymbol{\xi}$, $\boldsymbol{\eta}$.

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

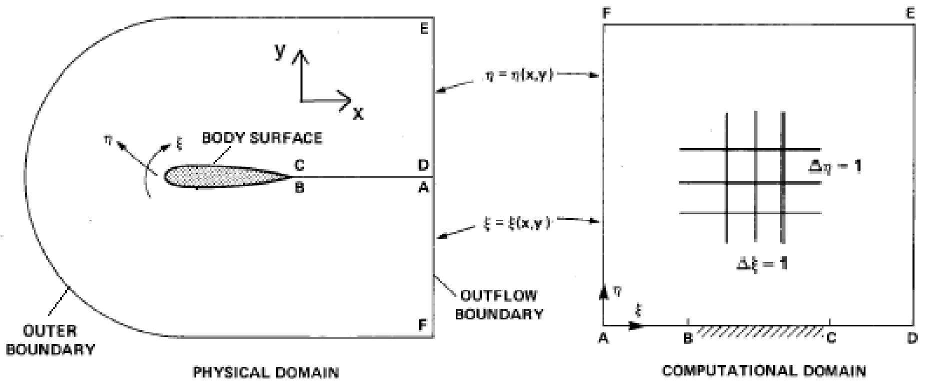


Figure 2-1 Generalized transformation from Physical domain to computational domain

In computational domain; the grid spacing is equal to one between the grid nodes. The Euler equations for steady problem in 2-D can be re-written in the form:

$$\frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} = 0 \tag{2.5}$$

Corresponding flux terms are specified as following;

$$\hat{F} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (e_t + p)U \end{bmatrix} \quad \hat{G} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (e_t + p)V \end{bmatrix} \tag{2.6}$$

U and V are contravariant velocities and given by:

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} \tag{2.7}$$

Directly computing the metric terms is not easy. Instead, the metric terms are computed based on the reverse transformation:

$$\begin{bmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}^{-1} \tag{2.8}$$

The transformation metrics appears as;

$$\begin{aligned} \xi_x = J y_\eta \quad \xi_y = -J x_\eta \\ \eta_x = -J y_\xi \quad \eta_y = J x_\xi \end{aligned} \tag{2.9}$$

2.4 Spatial Discretization

Euler equations can be represented in the form:

$$R(Q) = 0 \tag{2.10}$$

where the flow residual is;

$$R(Q) = \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} \tag{2.11}$$

The flow variables are stored at cell centers. However, for flux calculations flow variables which are stored at cell centers are interpolated to the cell faces. Required grids are generated by the commercial software program GRIDGEN.

A typical control volume is shown in Figure 2-2.

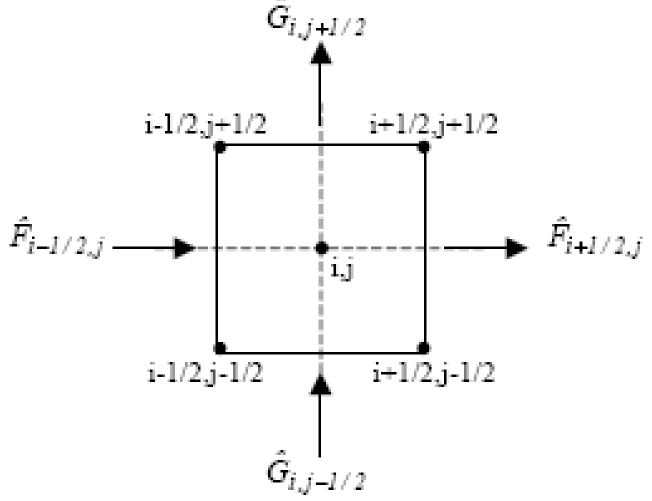


Figure 2-2 A typical control volume

By considering flux balances across the cell, spatial derivatives of the flux vectors can be written as:

$$\frac{\partial \hat{F}}{\partial \xi} = (\hat{F}_{i+\frac{1}{2},j} - \hat{F}_{i-\frac{1}{2},j})$$

$$\frac{\partial \hat{G}}{\partial \xi} = (\hat{G}_{i,j+\frac{1}{2}} - \hat{G}_{i,j-\frac{1}{2}}) \tag{2.12}$$

Flow variables are considered as constant within the each cell and the fluxes are denoted at the interfaces between the cells.

Then the Equation (2.11) can be rewritten as:

$$(\hat{F}_{i+\frac{1}{2},j} - \hat{F}_{i-\frac{1}{2},j}) + \left(\hat{G}_{i,j+\frac{1}{2}} - \hat{G}_{i,j-\frac{1}{2}}\right) = 0 \tag{2.13}$$

Central and upwind schemes are the two alternatives of performing spatial discretization of the fluxes.

While using the central schemes, calculation of the fluxes are performed based on the averaged flow variables at cell interfaces. The advantage of using central scheme is the implementation. It is easy to implement. However, artificial dissipation is required for central schemes. Upwind schemes, on the other hand, do not necessitate artificial dissipation but the implementation is not so easy.

Using upwind schemes can be better way to capture possible discontinuities in the flow. However, for some cases, usage of the limiter functions should be required to correctly model of the flow.
In this thesis upwind schemes are used with flux vector splitting and flux difference splitting methods.

2.4.1 Flux Vector Splitting

Flux-vector splitting can be accepted as the first level of upwind schemes. the convective fluxes can be constructed with two different ways in the flux vector splitting schemes. In the first way; the sign of the characteristic variables identifies the convective fluxes. In the second way; the flux vectors can be decomposed directly into convective and pressure parts.

2.4.1.1 Van Leer Scheme

Van Leer scheme is based on the characteristic decomposition of the convective fluxes. It can be defined here as Mach number splitting.

$$M_n = M_L^+ + M_R^- \tag{2.14}$$

Where the split Mach numbers are defined as

$$M_L^+ = \begin{cases} M_L & if\ M_L \geq 1 \\ \frac{1}{4}\left(M_L + 1\right)^2 & if\ \left|M_L\right| < 1 \\ 0 & if\ M_L \leq -1 \end{cases} \qquad M_R^- = \begin{cases} 0 & if\ M_R \geq 1 \\ \frac{1}{4}\left(M_R - 1\right)^2 & if\ \left|M_L\right| < 1 \\ M_R & if\ M_R \leq -1 \end{cases} \tag{2.15}$$

The Mach numbers, M_L and M_R, are calculated using the left and right states;

$$M_L = \frac{U_L}{c_L} \quad , \quad M_R = \frac{U_R}{c_R} \tag{2.16}$$

In the case of subsonic flows; where, $\left|M_n\right| < 1$ the positive and negative flux parts are given by:

$$F_c^{\pm} = \begin{bmatrix} f_{mass}^{\pm} \\ f_{mass}^{\pm}\left(u + \eta_x \frac{-V \pm 2c}{\gamma}\right) \\ f_{mass}^{\pm}\left(v + \eta_y \frac{-V \pm 2c}{\gamma}\right) \\ f_{energy}^{\pm} \end{bmatrix} \tag{2.17}$$

Corresponding mass and energy flux components are defined as:

$$\begin{aligned} f_{mass}^{+} &= \rho_L c_L \frac{(M_L + 1)^2}{4} \\ f_{mass}^{-} &= \rho_R c_R \frac{(M_R - 1)^2}{4} \\ f_{energy}^{\pm} &= f_{mass}^{\pm} \left\{ \frac{\left((\gamma - 1)U \pm 2c\right)^2}{2(\gamma^2 - 1)} + \frac{u^2 + v^2 - U^2}{2} \right\}_{L/R} \end{aligned} \tag{2.18}$$

In the case of supersonic flow, where $|M_n| > 1$, the fluxes are given by:

$$\begin{array}{lll} F_c^+ = F_c \;, & F_c^- = 0 & if\ M_n \geq 1 \\ F_c^+ = 0 \;, & F_c^- = F & if\ M_n \leq -1 \end{array} \tag{2.19}$$

2.4.1.2 AUSM scheme

The Advection Upstream Splitting Method, AUSM, was introduced by Liou and Steffen. The AUSM scheme includes convected and pressure parts.
Splitting according to the Mach number is performed as in the Van Leer scheme.

Corresponding flux splitting formulation is given below:

$$F_c^+ = M_L^+ \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho ch_T \end{bmatrix}_L + \begin{bmatrix} 0 \\ \eta_x p_L^+ \\ \eta_y p_L^+ \\ 0 \end{bmatrix} \qquad F_c^- = M_R^- \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho ch_T \end{bmatrix}_R + \begin{bmatrix} 0 \\ \eta_x p_R^- \\ \eta_y p_R^- \\ 0 \end{bmatrix} \tag{2.20}$$

and pressure is splitted as follows:

$$p_L^+ = \begin{cases} p_L & if\ M_L \geq 1 \\ \frac{p_L}{4}(M_L+1)^2(2-M_L) & if\ |M_L| < 1 \\ 0 & if\ M_L \leq -1 \end{cases}$$

$$p_R^- = \begin{cases} 0 & if\ M_R \geq 1 \\ \frac{p_R}{4}(M_R-1)^2(2-M_R) & if\ |M_R| < 1 \\ p_R & if\ M_R \leq -1 \end{cases} \tag{2.21}$$

2.4.1.3 Steger-Warming Scheme

Eigenvalue splitting is not unique. There are lots of ways of splitting methods. In Steger-Warming Scheme; the convective fluxes vectors can be calculated as follows:

$$F_c^{\pm} = \frac{\rho}{2\gamma}\begin{bmatrix} 2(\gamma-1)\lambda_1^{\pm} + \lambda_2^{\pm} + \lambda_3^{\pm} \\ \left(2(\gamma-1)\lambda_1^{\pm} + \lambda_2^{\pm} + \lambda_3^{\pm}\right)u + c\left(\lambda_2^{\pm} - \lambda_3^{\pm}\right)\eta_x \\ \left(2(\gamma-1)\lambda_1^{\pm} + \lambda_2^{\pm} + \lambda_3^{\pm}\right)v + c\left(\lambda_2^{\pm} - \lambda_3^{\pm}\right)\eta_y \\ \left(2(\gamma-1)\lambda_1^{\pm} + \lambda_2^{\pm} + \lambda_3^{\pm}\right)\frac{u^2+v^2}{2} + cU\left(\lambda_2^{\pm} - \lambda_3^{\pm}\right) + c^2\frac{\lambda_2^{\pm} + \lambda_3^{\pm}}{\gamma-1} \end{bmatrix} \tag{2.22}$$

In Equation 2.22 η_x and η_y represents the components of face normal vector. The speed of sound and eigenvalues are defined as:

$$c = \sqrt{\gamma(\gamma-1)\left(e_t - \frac{1}{2}\left(u^2+v^2\right)\right)}$$
$$\lambda_1 = U\,,\quad \lambda_2 = U + c\,,\quad \lambda_3 = U - c \tag{2.23}$$

For positive sign fluxes, F^+; corresponding velocities, speed of sound and energy terms are calculated from the left state flow variables. Similarly, for negative signed fluxes, F^-; they are calculated from right state variables.

By splitting the eigenvalues in terms of their signs:

$$\lambda_i^{\pm} = \frac{\lambda_i \pm |\lambda_i|}{2} \tag{2.24}$$

Using this scheme with previously defined eigenvalues can cause problems when the eigenvalues are equal to zero at sonic points and stagnation points. This behavior makes the function discontinuous at these points. In order to reduce the discontinuity a small number **ε** is used.

$$\lambda_i^{\pm} = \frac{\lambda_i \pm \sqrt{\lambda^2+\varepsilon^2}}{2} \tag{2.25}$$

2.4.2 Flux Difference Splitting

In contrast to the flux-vector splitting schemes, flux difference splitting methods considers not only the direction of wave propagation but also the waves themselves.

2.4.2.1 Roe Scheme

Roe's approximation is based on the decomposition of the flux difference over a face of the control volume. Roe's averaged Jacobian matrix satisfies the homogeneity property. Total flux can be defined with left and right state flow variables as below:

$$F_c = J_{RL} Q \;= |J_{RL}|(q^R - q^L)\; F_c = F^L - F^R \tag{2.26}$$

The diagonalized Roe's Jacobian matrix is:

$$\begin{aligned} \boldsymbol{J}_{RL} &= \tilde{\boldsymbol{Q}}_A\, \tilde{\boldsymbol{\Lambda}}_J\, \tilde{\boldsymbol{Q}}_A^{-1} \\ |\boldsymbol{J}_{RL}| &= \tilde{\boldsymbol{Q}}_A \left|\tilde{\boldsymbol{\Lambda}}_J\right| \tilde{\boldsymbol{Q}}_A^{-1} \end{aligned} \tag{2.27}$$

Diagonal matrix, which is composed of eigenvalues of the Jacobian, is denoted with $\tilde{\Lambda}_J$, and the right eigenvectors matrix is $\tilde{Q}_\Lambda$. When the equations are compiled into each other:

$$F_c = F(q^R) - F(q^L) = \sum \tilde{Q}_\Lambda |\tilde{\lambda}_f| Q_\Lambda^{-1}(q^R - q^L) \tag{2.28}$$

By considering the eigenvalues signs, the flux vector can take the below form:

$$\begin{aligned} F_c &= F(q^L) + \sum_{\lambda_f<0} \tilde{Q}_\Lambda |\tilde{\lambda}_f| \, Q_\Lambda^{-1}(q^R - q^L) \\ F_c &= F(q^R) - \textstyle\sum_{\lambda_f>0} \tilde{Q}_\Lambda |\tilde{\lambda}_f| \, Q_\Lambda^{-1}(q^R - q^L) \end{aligned} \tag{2.29}$$

Then averaging the two flux vectors, the equation (2.28) can be rewritten as:

$$F_c = 0.5\left[F(q^R) - F(q^L) - \sum \tilde{Q}_\Lambda |\tilde{\lambda}_f| Q_\Lambda^{-1}(q^R - q^L)\right] \tag{2.30}$$

Corresponding Roe's Jacobian matrix and the Roe's averaged variables are given below:

$$J_{RL} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\gamma-3}{2}u_{RL}^2 & (3-\gamma)u_{RL} & \gamma-1 \\ -h_{T_{RL}}u_{RL} + \frac{(\gamma-1)u_{RL}^3}{2} & h_{T_{RL}} - (\gamma-1)u_{RL}^2 & \gamma u_{RL} \end{bmatrix} \tag{2.31}$$

$$u_{RL} = \frac{\sqrt{\rho_R}u_R + \sqrt{\rho_L}u_L}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

$$h_{T_{RL}} = \frac{\sqrt{\rho_R}h_R + \sqrt{\rho_L}h_L}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

$$\rho_{RL} = \sqrt{\rho_R \rho_L} \tag{2.32}$$

2.4.3 High Order Schemes with Limiters

In the calculations; flow variables are assumed as constant within the cell. However, when the flux calculations at the interfaces of the cells are needed to be evaluated, the required computation of the flow variables are performed at the cell faces.

With first order interpolation; the flow variables are as follows:

$$q^L_{i+\frac{1}{2}} = q_i \quad , q^R_{i+\frac{1}{2}} = q_{i+1}$$
$$q^L_{i-\frac{1}{2}} = q_{i-1} \ , \ q^R_{i-\frac{1}{2}} = q_i \tag{2.33}$$

High order accuracy can be achieved by varying the flow variables within the cell. The Monotonic Upstream Centered Scheme, MUSCL, is used for high order reconstructions.

The interpolation formula is as below:

$$q^L_{i+\frac{1}{2}} = q_i + \frac{1}{4}\{\phi(r)[(1-\kappa)\nabla + (1+\kappa)\Delta]\}_i$$
$$q^R_{i+\frac{1}{2}} = q_{i+1} - \frac{1}{4}\{\phi(r)[(1+\kappa)\nabla + (1-\kappa)\Delta]\}_i \tag{2.34}$$

Δ_i *and* ∇_i are the difference operators and they are shown below:

$$\Delta_i = q_{i+1} - q \quad \nabla_i = q_i - q_{i-1}$$
$$r_i = \frac{\Delta_i}{\nabla_i} \tag{2.35}$$

Type of differencing method and the order of the corresponding discretization can be determined by defining different values to φ and κ.

First order accuracy can be obtained when Φ=0 and κ=0, and when Φ=1 and κ=1 the accuracy is increased to the second order. By varying values of these parameters more, the order of accuracy can be increased further.

Second and higher-order upwind spatial discretizations require usage of limiters in order to prevent oscillations at some situations. For example shock waves can cause oscillations in the solutions. Reduction of the slopes can prevent those oscillations which are used in the interpolation. Slopes of the corresponding functions are made zero at strong discontinuities to reduce the order of discretization to first order where the gradients are large.

In Equation 2.36 limiters are denoted by $\phi(r)$. They are functions of the forward and backward difference operators. Continuous limiter functions are used while performing analytical differentiation of the fluxes for the Jacobian derivation through this study. Where of the two continuous limiter functions are written below:

$$\phi(r)=\frac{2r}{r^2+1} \qquad for \quad \kappa=0$$
$$\phi(r)=\frac{3r}{2r^2-r+2} \qquad for \quad \kappa=\frac{1}{3} \qquad (2.36)$$

In this study limiters are activated only at high gradient regions in order to reduce the discretization accuracy to first order.

In equations 2.36; when κ = 0 limiters leads to Van Albada limiter, and when κ =1/3 limiters leads to Hemker- Koren limiter. For some cases; Van Albada limiter is defined with κ = 0 case to prevent the activation of the limiter in smooth regions by introducing an additional parameter,∈. Similar modification is performed for κ =1/3 scheme by Venkatakrishnan. Modified interpolations formulas are given below:

For κ = 0
$$\delta=\frac{(a^2+\in)b_i+(b^2+\in)a}{a^2+b^2+2\in} \qquad (2.37)$$

For κ = 1/3
$$\delta=\frac{(2a^2+\in)b+(b^2+2\in)a}{a^2+b^2-ab+3\in} \qquad (2.38)$$

where
$$a_L=\Delta_i \quad , \quad b_L=\nabla_i$$
$$a_R=\nabla_{i+1} \quad , \quad b_R=\Delta_{i+1} \qquad (2.39)$$

In equations from 2.36 to 2.40, ∈ is used as a small number to make it works only at high gradient regions.

2.5 Boundary Conditions and Block Interfaces

In order to impose the flow properties to the whole domain; determination of the boundary conditions are required. Physical domain boundaries include wall, far-field, inflow and outflow boundaries. Special consideration is required for the block interfaces for multi-block analysis. Varying according to the boundary condition type, some specific treatments are needed to correctly evaluate the flow variables and fluxes on the domains. Ghost cells are used for the implementation of the physical domains. One layer ghost cell is used at the physical boundaries in this study.

At the interior block interfaces, necessary data transfers between corresponding blocks are performed with halo nodes and special consideration is required at the interior boundaries such as wake-cut.

2.5.1 Far Field Boundary Conditions

External flow numerical simulations have to be conducted within a bounded domain. For this reason, artificial far-field boundary conditions are required. While performing the numerical implementation of the far-field boundary, two basic requirements should be taken into consideration. The far-field boundaries should simulate the flow as if the boundaries are at infinity and the disturbances which occur in the inner domain should not be reflected back into the flow field.
Characteristic based boundary conditions approach is used to define required flow conditions at the far-field. Eigenvalue signs of the convective flux Jacobians determine the direction of the information along the characteristic lines. It can be towards to the computational domain or to the out-of the boundaries. Therefore, depending on the local Mach number, four different types of far-field boundary conditions should be investigated. [42]

Supersonic Inflow: All eigenvalues have the same sign and the conservative variables on the boundary are determined by free stream values only.

Supersonic Outflow: All eigenvalues have the same sign and the conservative variables on the boundary are determined by the solution inside the domain.

Subsonic Inflow: Three characteristics (velocity components and density) enter to the domain and one characteristic (pressure) leaves the domain. Therefore, one characteristic variables are extrapolated from inside and the others are calculated from free stream values.

Subsonic Inflow: Three characteristics leave domain and should be extrapolated from inside but the other variable must be determined externally.

2.5.2 Wall Boundary Conditions

The aim of the wall boundary condition is to model no flow through the boundaries. Since the flow is inviscid, required simulation can be performed by using symmetry condition. At wall boundaries; the magnitudes of the velocity components in the normal direction are equal but have opposite signs on the both sides of the boundaries. The density, tangential component of the velocity and energy is extrapolated from the interior cells to the ghost cells of the wall-boundary.

2.5.3 Computational Boundary Conditions

Symmetry and wake-cut boundary conditions are two of the most widely used computational boundary conditions. Symmetry boundary conditions are used when the domain and the flow characteristics are symmetric about an axis. In symmetry boundary conditions, while constructing the ghost cells, all of the flow variables are extrapolated from the interior cells and only the sign of the normal velocity component is in the opposite direction. In the wake-cut boundary conditions; the solution is computed to fourth-order using the data in front and behind of the wake cut as follows:

$$q_{iwc} = \frac{1}{6}(-q_{iwc-2} + 4q_{iwc-1} + 4q_{iwc+1} - q_{iwc+2}) \tag{2.41}$$

2.5.4 Block Interfaces

For multi-block grids; special considerations is required for the data transfer between the block interfaces. The block boundaries of the neighbors are overlapped in the streamwise direction when halo nodes are used.

The blocks are solved independently during start-up. As it is shown in Figure 2-3 the last interior column of Block 1 is specified as halo column of Block 2 and the first interior column of Block 2 is specified as halo column of Block 1. At each iteration; flow variables in halo columns are updated from the corresponding interior cells of the neighboring block. Two different results come out at the interface of the two blocks and then the solutions are subsequently averaged. At steady state the block interfaces becomes completely transparent as it is a single block.

Block interfaces; which are in the cross-stream directions are treated as wake-cuts and such boundaries do not require halo nodes. As it is in the wake-cut boundaries; flow variables in the last two columns of Block 1 and in first two columns of Block 2 are used to calculate the fluxes on the blocks interfaces.

(Figure 2-4). However, it should be noted that the halo nodes work for both of the conditions

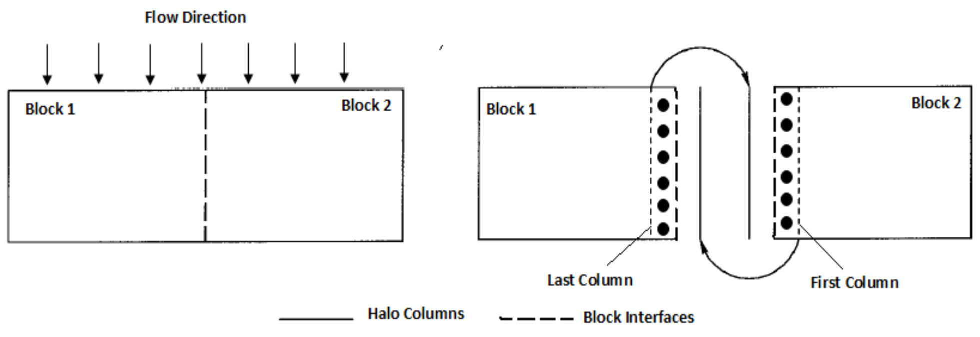


Figure 2-3 Blocks Interface in the Streamwise Direction

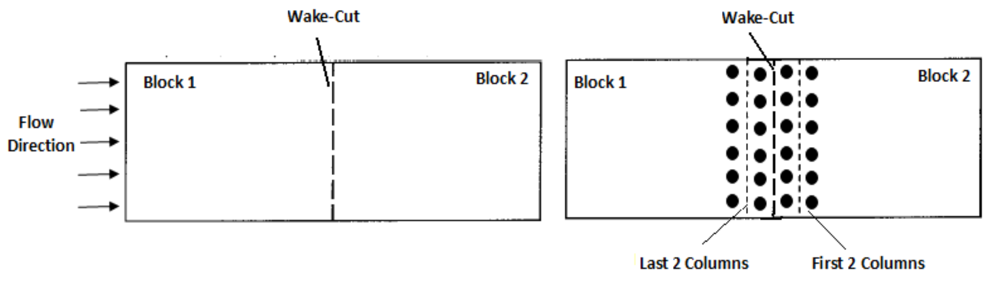


Figure 2-4 Blocks Interface is Perpendicular to the Flow

CHAPTER 3

SOLUTION METHOD

3.1 Introduction

Iterative methods are being used for many years to solve the governing equations of inviscid and viscous flows. During these studies explicit and implicit schemes are used. In these schemes, iterative approach is used to solve the large linear systems of equations which come from linearization in time. At this point; direct solution to this linear system of equation introduces Newton's method. However, this has not been used so much due to the large memory requirements. However, it is now easy to use Newton method with powerful computers.

In this chapter, Newton method is presented with the details of the methodology. Moreover, the flux Jacobian matrix structure, implementation of the initial and boundary conditions, solving methods of the large sparse matrices are explained. At the end of the chapter, the verification of the generated code is done by comparing the code results with other authors' results in literature. Airfoil and channel flow problems are chosen for verification.

3.2 Newton's Method

The discrete Euler equations are shown with a set of nonlinear algebraic equations, represented here by the vector equation;

$$R(q) = 0 \tag{3.1}$$

residual vector of the spatial discretization is shown as **R**(q). Residual is non-linear function of conservative flow variables, q.

By applying Newton's method to the residual equation; following linear system can be obtained;

$$A^n \Delta q^n = -R^n \tag{3.2}$$

$$R^n = R(q^n) \text{ and } \Delta q^n = q^{n+1} - q^n \tag{3.3}$$

Where the matrix is;

$$A_{ij}{}^n = \frac{\partial R_i}{\partial q_j}(q^n) \tag{3.4}$$

and $A_{ij}{}^n$ is specified as flow Jacobian.

Only the first-order Taylor series expansion is used and assuming that at the end of the iterations desired convergence is obtained, i.e. $R^{n+1}(q)$ =0. The Newton's Method can be defined as following:

$$(\frac{\partial \mathrm{R}}{\partial \mathrm{q}})^n \Delta q^n = -R(q^n) \tag{3.5}$$

Calculation for each iteration is given by;

$$Q^{(n+1)} = Q^{(n)} + \Delta Q^{(n)} \tag{3.6}$$

Then the analysis continues until the residual decreases below the desired value.

3.2.1 Flux Jacobian Evaluation

Flux Jacobian matrix calculation is needed for the solution of Euler equations. Elements of the matrix are the residual derivatives with respect to the flow variables vector. In this study; derivative calculations are performed with analytical and numerical methods.

3.2.1.1 Analytical Jacobian Derivation

Discretized flux residual can be evaluated as:

$$\begin{aligned} R_{i,j}(q) = {} & \left[F^+\left(q_{i+\frac{1}{2},j}{}^L\right) + F^-\left(q_{i+\frac{1}{2},j}{}^R\right)\right] \text{-} \left[F^+\left(q_{i-\frac{1}{2},j}{}^L\right) + F^-\left(q_{i-\frac{1}{2},j}{}^R\right)\right] + \\ & \left[G^+\left(q_{i,j+\frac{1}{2}}{}^L\right) + G^-\left(q_{i,j+\frac{1}{2}}{}^R\right)\right] \text{-} \left[G^+\left(q_{i,j-\frac{1}{2}}{}^L\right) + G^-\left(q_{i,j-\frac{1}{2}}{}^R\right)\right] = 0 \end{aligned} \tag{3.7}$$

In equation 3.7 $\mathrm{R}_{\mathrm{i,j}}$ refers to the cell residual. Derivatives of each cell residuals with respect to the flow variables construct the Jacobain matrix. Corresponding discretized residual Jacobians can be written as:

$$\begin{aligned} \frac{\partial R_{i,j}}{\partial Q_{k,l}} = {} & A^+_{i+1/2,J} \frac{\partial q_{i+1/2,j}{}^L}{\partial q_{k,l}} - A^+_{i+1/2,J} \frac{\partial q_{i-1/2,j}{}^L}{\partial q_{k,l}} + A^-_{i+1/2,J} \frac{\partial q_{i+1/2,j}{}^R}{\partial q_{k,l}} - A^-_{i-1/2,J} \frac{\partial q_{i-1/2,j}{}^R}{\partial q_{k,l}} + \\ & B^+_{i,J+1/2} \frac{\partial q_{i,j+1/2}{}^L}{\partial q_{k,l}} - B^+_{i,J+1/2} \frac{\partial q_{i,j-1/2}{}^L}{\partial q_{k,l}} + B^-_{i,J+1/2} \frac{\partial q_{i,j+1/2}{}^R}{\partial q_{k,l}} - B^-_{i,J-1/2} \frac{\partial q_{i,j-1/2}{}^R}{\partial q_{k,l}} \end{aligned} \tag{3.8}$$

For first order discretizations; k and l values in Equation 3.8 changes from i-1 to i+1 and j-1 to j+1, respectively.

$$q^L_{i+\frac{1}{2},j} = q_{i,j}\,,\quad q^L_{i-\frac{1}{2},j} = q_{i-1,j}\,,\quad q^L_{i,j+\frac{1}{2}} = q_{i,j}\,,\quad q^L_{i,j-\frac{1}{2}} = q_{i,j-1}$$
$$q^R_{i+\frac{1}{2},j} = q_{i+1,j}\,,\quad q^R_{i-\frac{1}{2},j} = q_{i,j}\,,\quad q^R_{i,j+\frac{1}{2}} = q_{i,j+1}\,,\quad q^R_{i,j-\frac{1}{2}} = q_{i,j}$$
(3.9)

Corresponding Jacobian matrices in first order discretization for a 5-point stencil can be calculated as:

$$\frac{\partial R_{i,j}}{\partial q_{i,j}} = A^+_{i+1/2,j} - A^-_{i-1/2,j} + B^-_{i,j+1/2} - B^-_{i,j-1/2}$$
$$\frac{\partial R_{i,j}}{\partial q_{i+1,j}} = A^-_{i+1/2,j} \quad \frac{\partial R_{i,j}}{\partial q_{i-1,j}} = -A^+_{i-1/2,j}$$
$$\frac{\partial R_{i,j}}{\partial q_{i,j+1}} = B^-_{i,j+1/2} \quad \frac{\partial R_{i,j}}{\partial q_{i,j-1}} = -B^+_{i,j-1/2}$$
(3.10)

For the second order spatial discretization; MUSCL scheme is used to calculate the flow variables at the cell faces. Interpolation of flow variables are performed at the center of the neighboring cells. With the help of limiter functions MUSCL scheme is made differentiable with respect to flow variables. Therefore, analytical Jacobian calculations are performed easily for high order schemes.

For analytical flux calculations; it can be said that the residuals can be calculated accurately and the order of error is equal to the round-off error. Although it requires detailed hand calculations, analysis time of the prepared program is short. On the other hand, when the complexity of the discretized residual equations increases, the analytical Jacobian derivation turns into a crucial problem. For these situations, numerical Jacobian studies will be a better choice.

3.2.1.2 Numerical Jacobian Derivation

Numerical Jacobian derivation method is another way to evaluate Jacobian fluxes. In this method; the numerical Jacobian can be evaluated with a small finite-difference perturbation magnitude, ε. With the i^{th} component of the residual vector j^{th} component of the flow variable vector and the j^{th} component of the unit vector the usage of ε is as follows:

$$\frac{\partial R_i}{\partial Q_j} = \frac{\partial R_i(q+e_j\varepsilon)-R_i(q)}{\varepsilon} \tag{3.11}$$

Perturbation magnitude can be positive or negative. However, the sign of it is so important when it is too close to the flow variable. First of all; the sign of ε determines the type of differencing method. Using positive value makes the differentiation forward and using negative value makes it backward. Then; while using numerical Jacobians; eigenvalue signs should be checked during the calculations to be sure that the perturbation magnitude does not cause to vary the flux vector. Finally it determines the accuracy of the analyses; error can be minimized with a good choice of it.

The numerical Jacobian method is used especially when the discretized residual equations are complex. Since for such cases; analytically derivation of the Jacobians are not easy. However, the computation time of the numerical Jacobian method is greater when compared with the analytical method and it requires some preliminary work for higher accuracy.

3.3 Structure of Jacobian Matrix

The Jacobian matrix is constructed with the partial derivatives of the cell residuals with respect to the flow variables. Generated matrix is a large sparse matrix but since the residual equations are depends on the local flow variables most of the elements are zero. While solving the large sparse matrix, only the nonzero elements are stored since evaluating all off the elements will be too expensive especially for the large problems.

Residual equation with a first-order discretization, requires five-point stencil. This produces block diagonal matrix with five 4x4-blocks. When second-order discretization is used; the required grid number increases to nine and the constructed block diagonal matrix includes nine 4x4-blocks. Hence, only these block bands and the elements corresponding to the boundary conditions are non-zero. Other elements in the matrix are zero.

For solving sparse matrixes UMFPACK and PARDISO packages are used. Details of the packages are discussed in the following chapters.

4 points are required for first order analyses and 9 points are required for second order analyses in 2-D Euler equations. Constructed stencils are shown in Figure 3-1 both for first order and second order 2-D Euler equations. For first order analysis 4 points are required and for second order analysis required points increase to 9 in 2-D Euler equations.

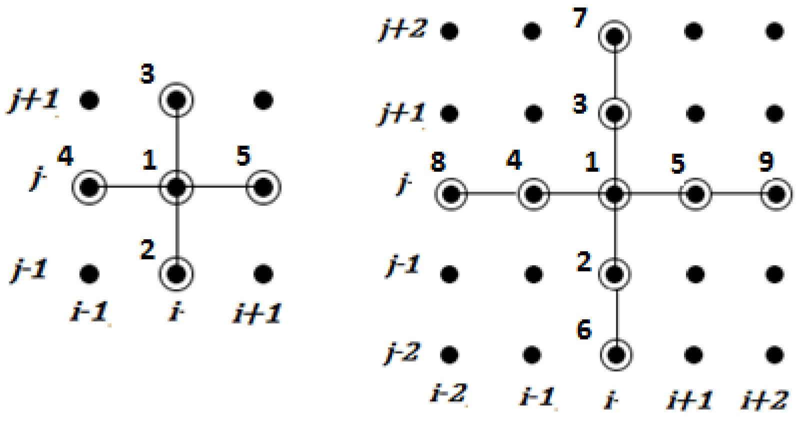


Figure 3-1 5 Points and 9 points stencils

Residual derivatives with respect to the flow variables are written for first order and second order Euler equations;

$$3 = \frac{\partial R}{\partial q_{i,j+1}}$$

$$4 = \frac{\partial R}{\partial q_{i-1,j}} \qquad 1 = \frac{\partial R}{\partial q_{i,j}} \qquad 5 = \frac{\partial R}{\partial q_{i+1,j}}$$

$$2 = \frac{\partial R}{\partial q_{i,j-1}} \tag{3.12}$$

$$7 = \frac{\partial R}{\partial q_{i,j+2}}$$

$$3 = \frac{\partial R}{\partial q_{i,j+1}}$$

$$8 = \frac{\partial R}{\partial q_{i-2,j}} \quad 4 = \frac{\partial R}{\partial q_{i-1,j}} \quad 1 = \frac{\partial R}{\partial q_{i,j}} \quad 5 = \frac{\partial R}{\partial q_{i+1,j}} \quad 9 = \frac{\partial R}{\partial q_{i+2,j}}$$

$$2 = \frac{\partial R}{\partial q_{i,j-1}}$$

$$6 = \frac{\partial R}{\partial q_{i,j-2}} \qquad (3.13)$$

Matrix structures for first and second order Euler equations are shown in Figure 3-2 and in Figure 3-3.

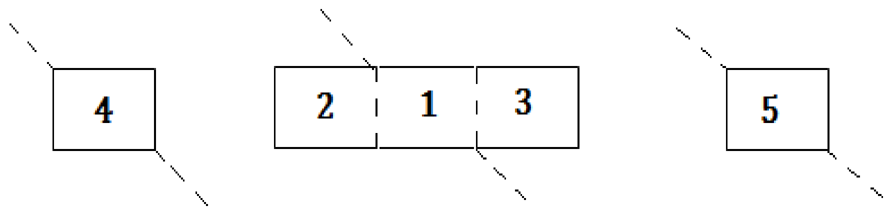


Figure 3-2 Matrix structure for first order analysis

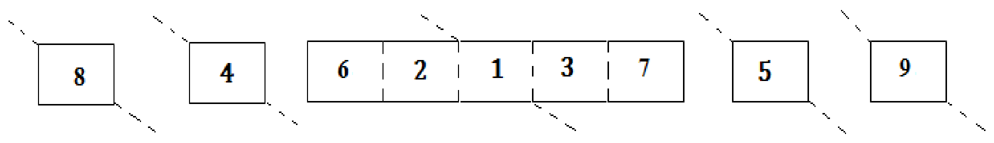


Figure 3-3 Matrix structure for second order analysis

3.4 Implementation of Initial Conditions

Good initial condition is necessary for Newton's method to obtain better convergence and to prevent early divergence. In general; free-stream conditions are used to implement initial conditions. However, by following such a way, one may not obtain desired results with Newton method for most of the problem cases. In order to use Newton method for these cases, if possible, a better guess for initial condition is required. If the problem is complex, making initial guess is difficult and one has to follow better procedure to get desired results. There are some studies for this situation in literature and the widely used one is adding a term to the diagonal of the Jacobian matrix. The aim to use this approach is to make the matrices more stable by having dominant diagonal.

Newton method with the additional term can be presented as;

$$\left(\frac{1}{\Delta t}[I] + \frac{\partial \hat{R}}{\partial \hat{Q}}\right) \Delta \hat{Q}^n = -R(\hat{Q}^n) \tag{3.14}$$

When $\Delta t \rightarrow \infty$, the modified Newton method becomes equal to the original one. The magnitude of Δt depends on the L_2-norm of the residuals. At the first iteration; the magnitude of the Δt is small and it becomes greater at the further iterations.

L_2-norm of the residuals can be shown as;

$$\Delta t^n = \Delta t^0 \frac{\|R(q^0)\|_2}{\|R(q^n)\|_2} \tag{3.15}$$

Since quadratic convergence cannot be obtained at initial iterations, this method increases the computational time. At initial iterations, convergence stays linear and quadratic convergence can be obtained as the additional term goes to infinity.

So, initial iterations require additional terms to make the Newton method works. However, at further iterations flow conditions become good enough to be used as initial condition without using additional term anymore. This may cause oscillation in the results. However, if the correct time is selected for removing additional term, the desired convergence can be reached.

3.5 Implementation of Boundary Conditions

In order to implement the boundary conditions, flow variables relations should be defined between the ghost cells and the interior cells. Relations can be defined explicitly by using the previous iterations. After each iteration both of the interior cells and ghost cells are updated by solving the Jacobian matrices. In this study, implicit boundary conditions are used by defining the required Jacobian matrix and the right hand side matrix entries and by solving them simultaneously. The linearization of the equations that define the relations between the ghost cells and interior cells are needed to be used in the implicit analysis. The corresponding linearized equation is given below:

$$[A]\Delta q_{interior} = [B]\Delta q_{GHOST} \tag{3.16}$$

Thes linearized equations varies according to the flow conditions. For supersonic inlet and supersonic outlet boundary conditions A and B matrices entries are equal to 1 but for subsonic boundary conditions a bit complicated calculations should be performed to evaluate the corresponding matrices.

3.6 Solution Method

The Jacobian matrix should be constructed and factorized in order to get the flow variables at each iteration. Derivatives of the residual function with respect to the flow variables form the Jacobain matrix. There are several methods to solve such matrices in the literature. Size of the matrix depends on the generated grid. Actually, most of the entries in the Jacobian matrix are zero. Hence this simplifies storing and solving the matrices.

In this study, matrix solver packages PARDISO (Parallel Direct Solver) and UMFPACK (Unsymmetric-pattern MultiFrontal PACKage) are used to solve the Jacobian matrices. Both of the matrices use LU decomposition to solve the sparse matrices. Both of them are able to solve large sparse symmetric and unsymmetric linear matrices. PARDISO can use shared-memory and distributed-memory multiprocessors. Solver time comparison for both of the packages is performed throughout this study.

3.7 Flow Solver Verification

Since the generated tool is inviscid, it is difficult the compare the results with experimental data. Instead of this, the generated tool results are compared with other authors' results in literature. NACA0012 airfoil is chosen for external flow test case and a channel flow with bump geometry is chosen for internal flow test case.

Grid generated for the channel flow having 10 % thickness to chord ratio is shown in Figure 3-4. Generated grid totally increases 129x33 nodes. Boundary conditions that are used for this problem will be discussed in the performance analyses part. For verification of the channel flow bump geometry, second order Van Leer scheme is used. 0.5 Mach and 0.675 Mach are chosen for test case conditions and corresponding Mach contours are compared with [43], which are shown in Figure 3-5 and in Figure 3-6.

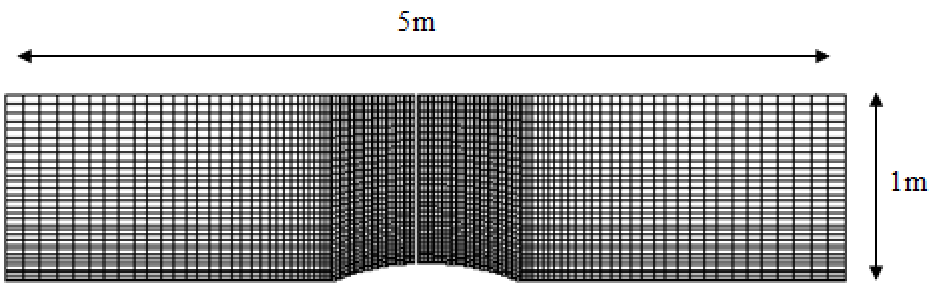


Figure 3-4 Channel Flow Grid (129x33)

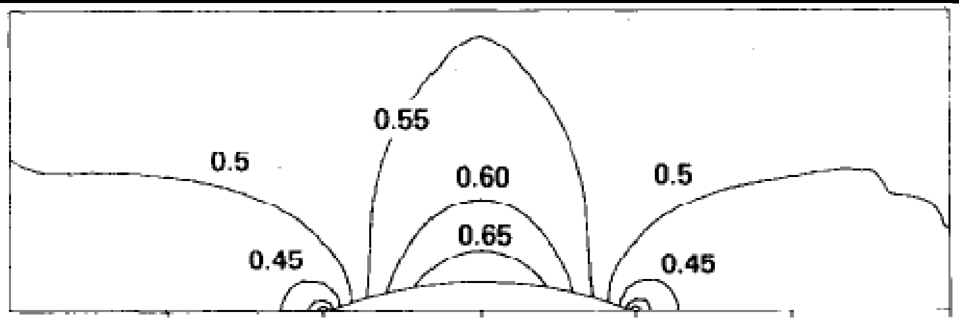
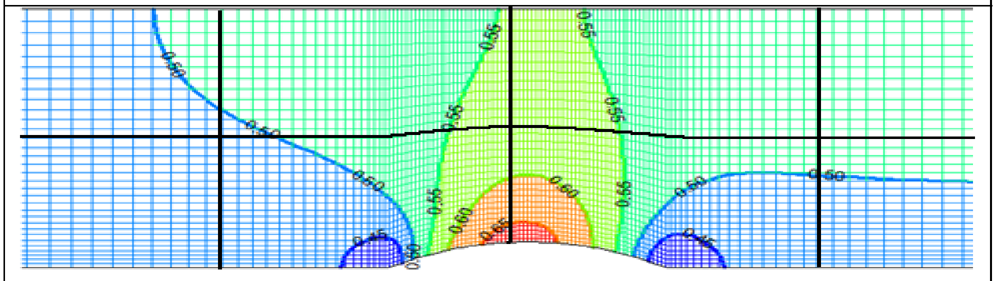
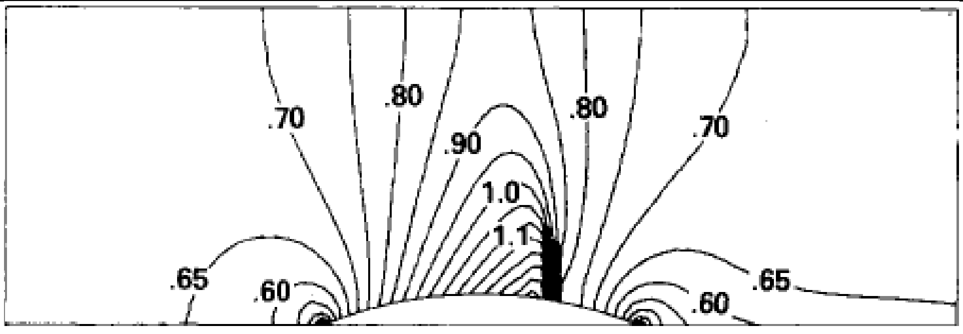
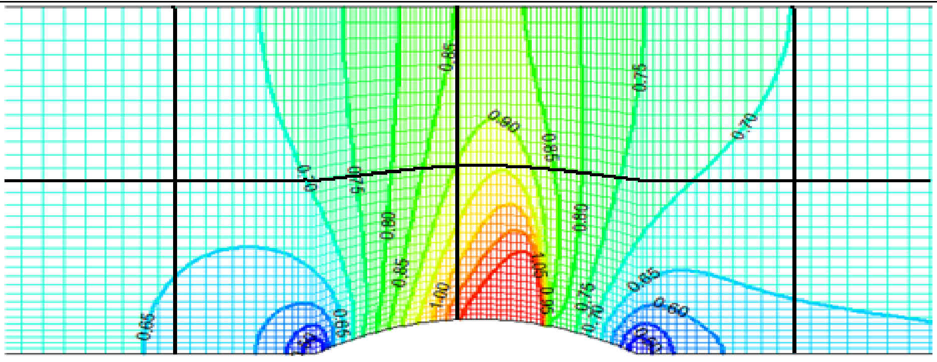

Mach Contours which are given in [43] for 0.5 Mach inlet velocity

Mach contours which are calculated from generated tool for 0.5 Mach inlet velocity

Figure 3-5 Channel Flow Mach Contours Comparison (0.5 Mach)



Mach Contours which are given in [43] for 0.675 Mach inlet velocity



Mach contours that are calculated from generated tool for 0.675 Mach inlet velocity

Figure 3-6 Channel Flow Mach Contours Comparison (0.675 Mach)

As it is demonstrated in Figure 3-5 and in Figure 3-6; flow solver is good at predicting the channel flow. Results are compatible with the ones given in [43] both for 0.5 Mach and 0.675 Mach. The black lines in the Mach contours figures represent the block interfaces which will be discussed in the following chapter.

NACA 0012 airfoil is chosen as a test case for external flow. C type grid is used for the flow domain and shown in Figure 3-7 with boundary conditions. Grid size is equal to 275x65.

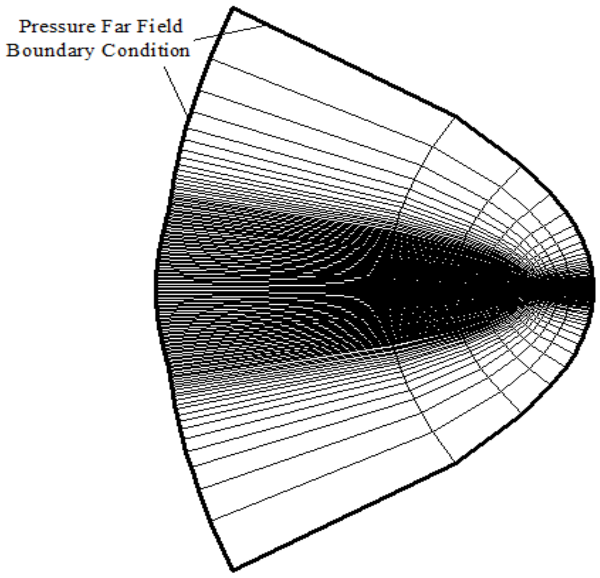


Figure 3-7 C Type Grid for NACA 0012 (Outer View)

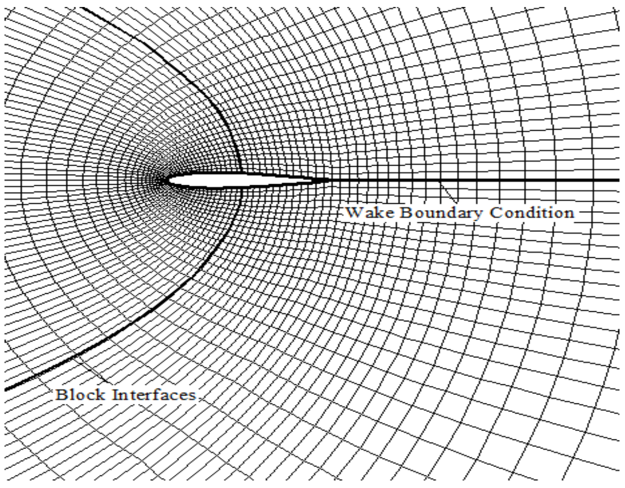
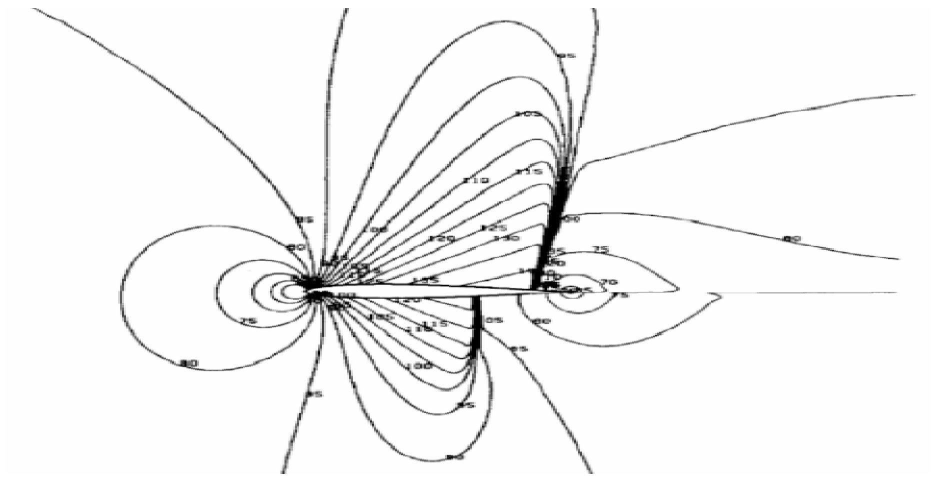
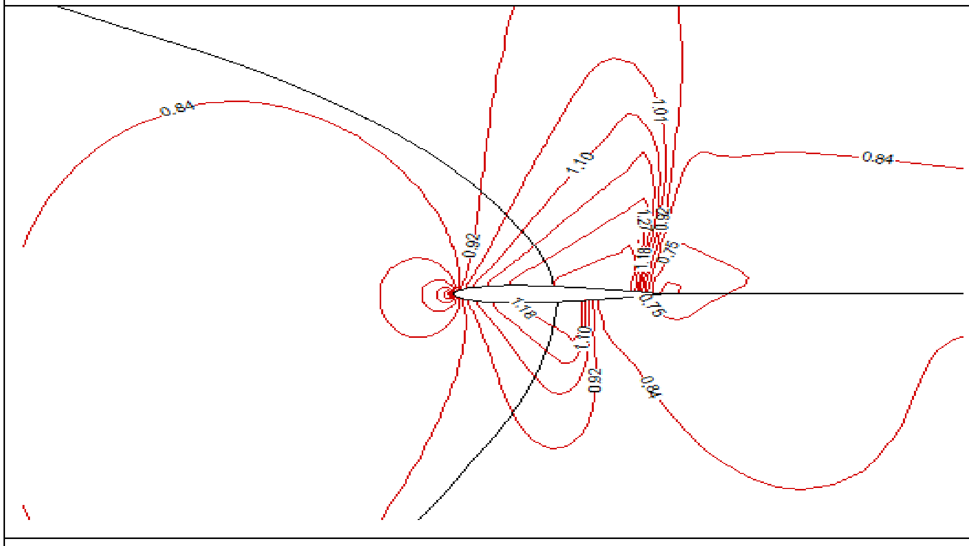


Figure 3-8 C Type Grid for NACA0012 (Inner View)

The analyses are performed for 0.85 Mach with 1^{o} angle of attack. Second order with Van-Leer scheme is used and corresponding Mach contours are compared with the ones given in [44] and demonstrated in Figure 3-9.



Mach Contours which are given in [44] for 0.85 Mach and 1^{o} angle of attack



Mach contours that are calculated from generated tool for 0.85 Mach and 1^{o} angle of attack

Figure 3-9 NACA0012 Mach Contours Comparison (0.85 Mach and 1^{o} angle of attack)

As it is shown in Figure 3-9 and mach contours are similar with the ones shown in [44]. Corresponding Cp comparison is given in Figure 3-10.

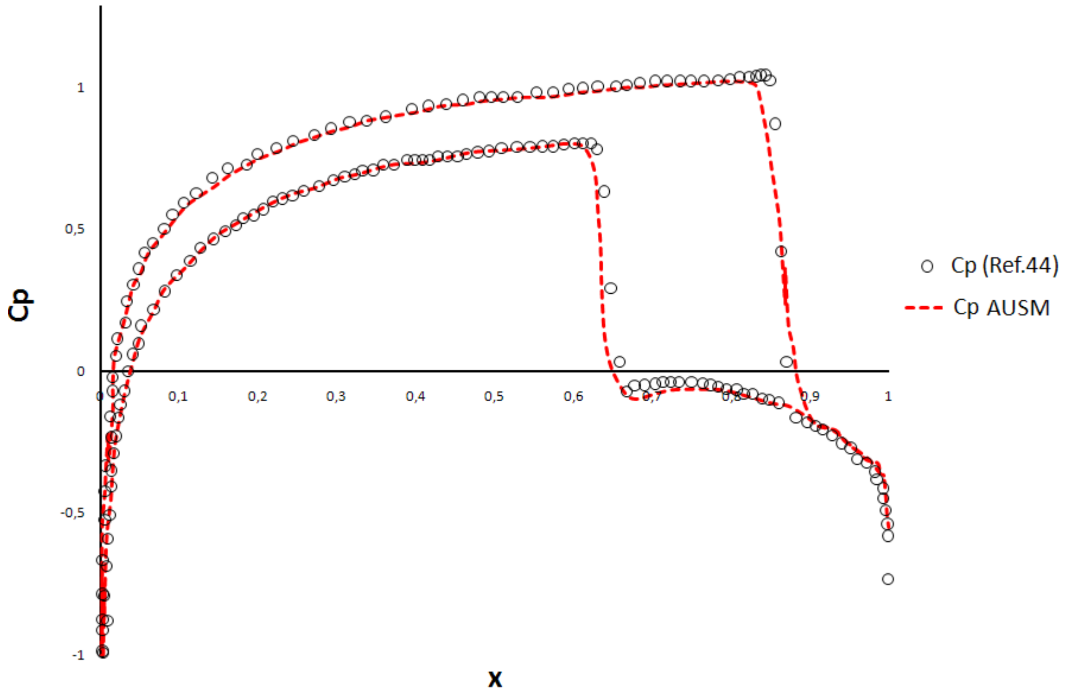


Figure 3-10 NACA0012 CP Comparison

CHAPTER 4

FLOW SOLVER PERFORMANCE

4.1 Introduction

In this chapter, performance analyses of the multi-block Newton method are discussed. An axisymmetric nozzle and a channel flow problems are chosen to be used for the performance analyses. The flow domain is divided into different number of blocks to examine the effect of number of blocks to accuracy, the convergence rate and CPU time spent. Block interface is moved to the different locations in the flow domain to check the behavior of the interface boundary condition at different flow conditions. Same problems are solved with different sparse matrix solvers to compare their speed and accuracy.

4.2 Problems Definition

An axisymmetric nozzle and a channel flow problems are chosen for performance analyses of the multi-block Newton method. The axisymmetric nozzle geometry is shown in Figure 4-1.

Corresponding analyses parameters for the axisymmetric nozzle problem are;

Inlet Area: 0.138474 m^2

Throat Area: 0.0314 m^2

Total Temperature: 3130 K

Total Pressure: 17425611 Pa

Inlet Velocity: 0.1 Mach

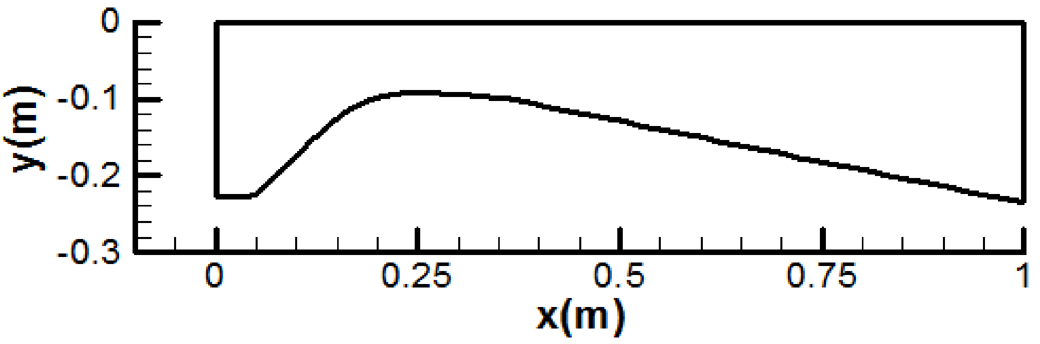


Figure 4-1 Axisymmetric Nozzle Geometry

Moreover, channel flow problem is used to evaluate the performance of the generated program for supersonic inflow conditions and examine block boundary conditions around shock and expansion waves with 1.65 Mach inflow speed at sea level standard atmospheric conditions. The channel flow with the bump geometry is shown in Figure 4-2.

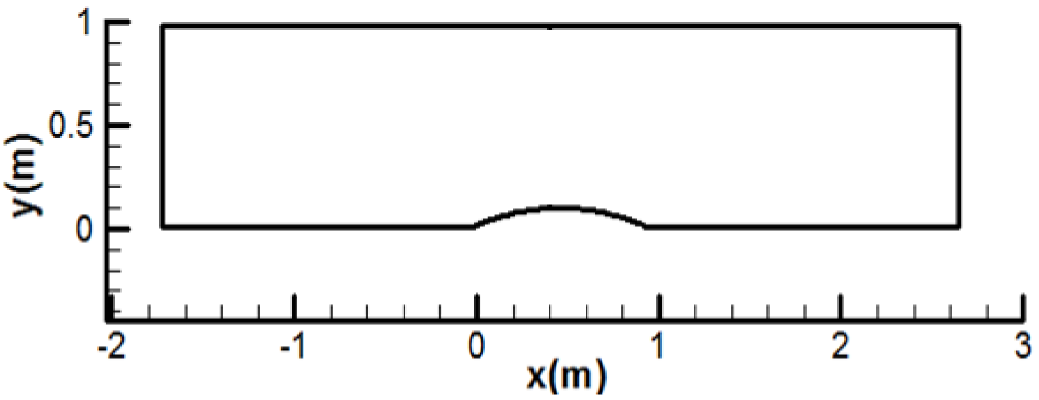


Figure 4-2 Channel Flow with a 10% Thick Circular Arc Bump

Boundary conditions that are used in the analyses are shown in Figure 4-3 and in Figure 4-4. At inlet and outlet boundaries; Riemann invariants are used to apply characteristic type boundary condition. Symmetry boundary condition is used at the upper boundary to make the domain axisymmetric and the wall boundary condition is used at the lower boundary.

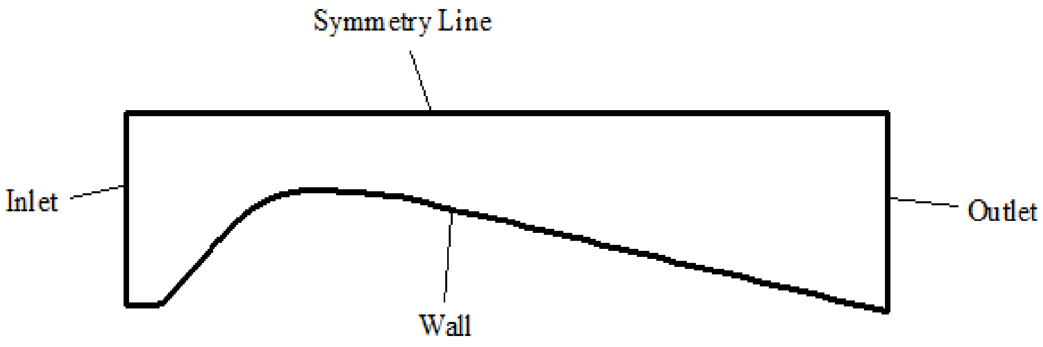


Figure 4-3 Axisymmetric Nozzle Boundary Conditions

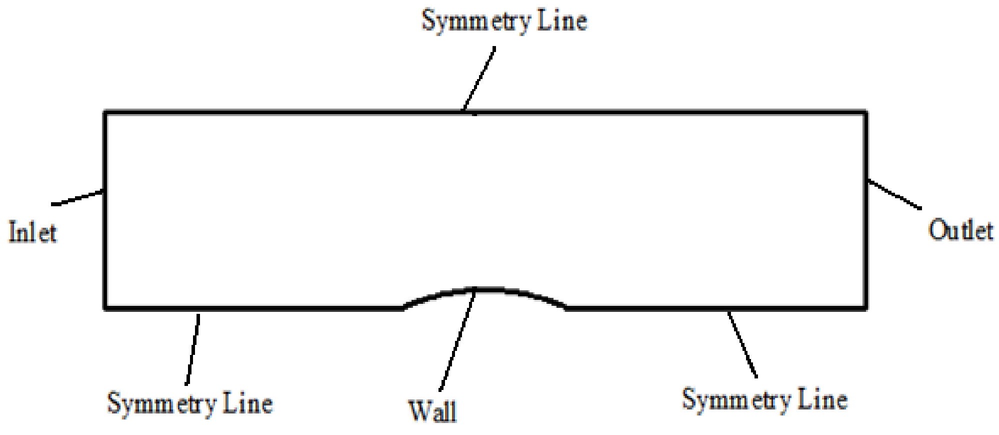
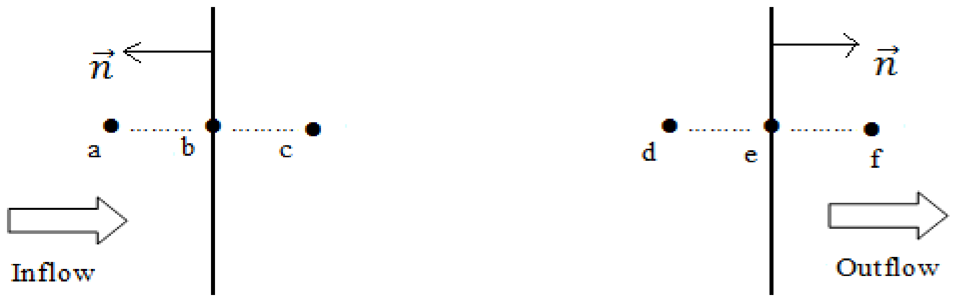


Figure 4-4 Channel Flow Boundary Conditions

Boundary conditions are implemented as explained in Chapter.2. In axisymmetric nozzle problem; subsonic inflow & supersonic outflow and in channel problem supersonic inflow and supersonic outflow conditions are used. Signs of the eigenvalues of the convective fluxes determine the direction of the transformed information.

For subsonic inflow boundary condition, characteristic variables are determined from the freestream values. One of the characteristic variables is extrapolated from the interior of the flow domain. Following boundary condition equations can be derived for inflow and outflow boundaries



$$p_b = \frac{1}{2}\{p_a + p_c - p_a c_0[n_x(u_a - u_c) + n_y(v_a - v_c)]\}$$
$$p_b = p_a + (p_b - p_a)/c_0^2$$
$$\mathrm{u}_b = \mathrm{u}_a - n_x(p_a - p_b)/\rho_0 c_0 \tag{4.1}$$
$$v_b = v_a - n_y(p_a - p_b)/\rho_0 c_0$$

For supersonic inflow, all of the eigenvalues have the same sign. All of the conservative variables on the inlet boundary are determined from the freestream flow variables.

$$\overrightarrow{Q_b} = \overrightarrow{Q_a} \tag{4.2}$$

For supersonic outflow, all of the eigenvalues have the same sign. All of the conservative variables on the outlet boundary are determined from the inside of the domain.

$$\overrightarrow{Q_e} = \overrightarrow{Q_d} \tag{4.3}$$

4.3 Grid Parameters

Axisymmetric nozzle grid, which is used for the analyses, includes 82x21 and the channel problem grid includes 129x33 number of nodes. Since the analyses are inviscid, boundary layer is not constructed on wall boundaries. Single block grids are shown in Figure 4-5 and in Figure 4-6.

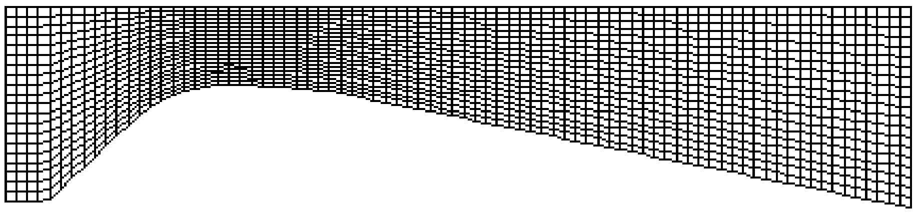


Figure 4-5 Axisymmetric Nozzle Single Block Grid

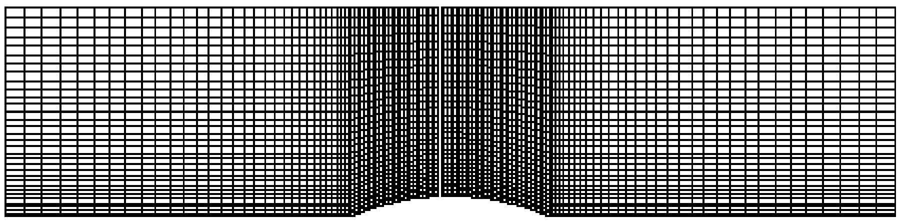


Figure 4-6 Channel Single Block Grid

Size of the one Jacobian matrix can be calculated with the following formula. By considering four flow variables in 2-D Euler equations. (Figure 4-7)

$$Jacobian\ Matrix\ Size = 4 \times number\ of\ i\ grids \times number\ of\ j\ grids \quad (4.4)$$

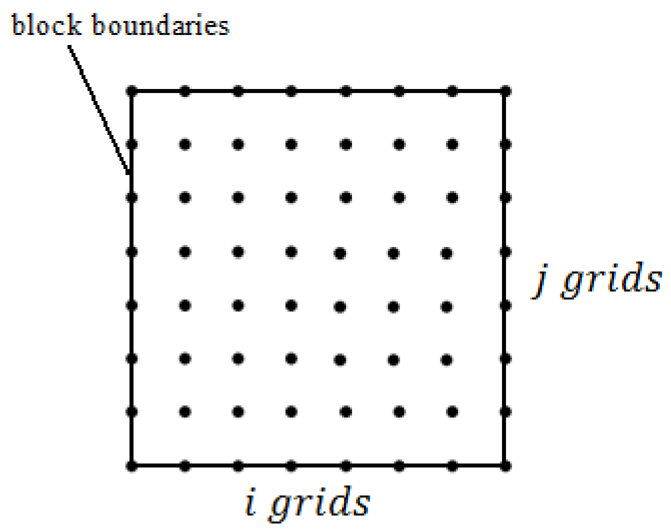
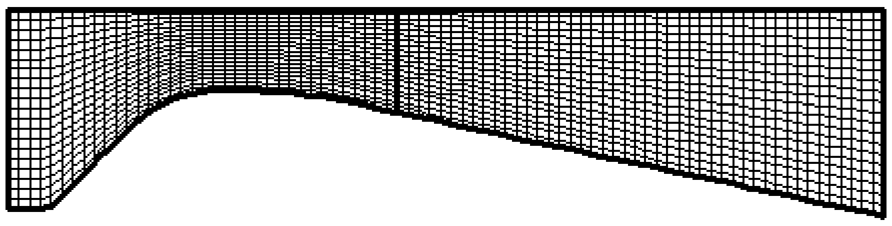


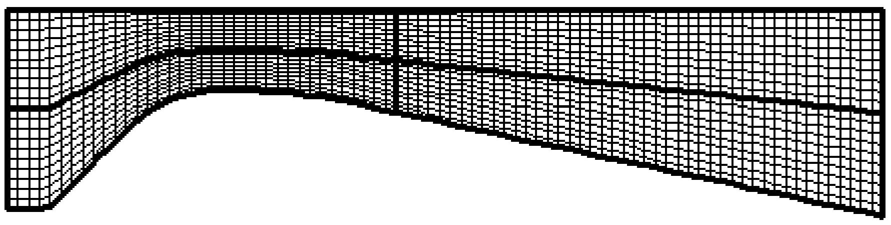
Figure 4-7 2-D Single Block Grids

For multi-block Newton method performance evaluation, same grid is used and divided into different number of blocks as it is shown in Figure 4-8. In the analysis, number of solved Jacobian matrices is equal to the number of blocks.

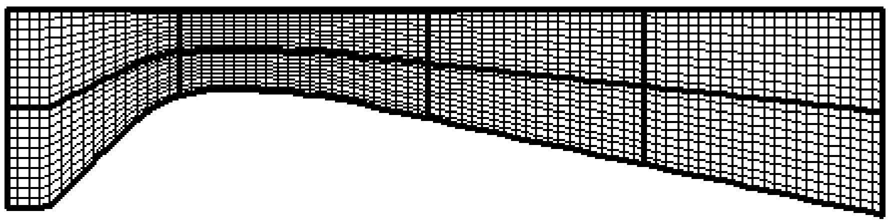
In following axisymmetric nozzle multi-block analyses; constructed blocks have nearly same sizes as shown in Figure 4-8. Totally 1722 nodes are tried to be distributed within the whole blocks equally for axisymmetric nozzle problem.



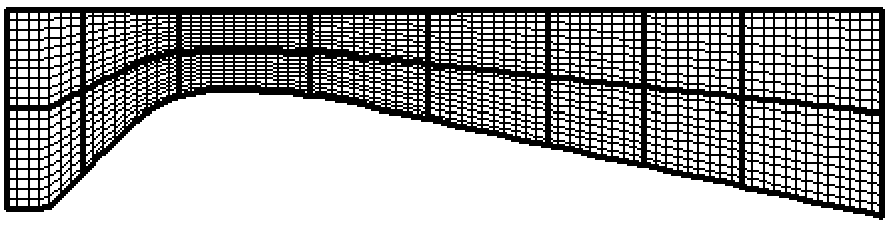
2 Blocks Grid



4 Blocks Grid



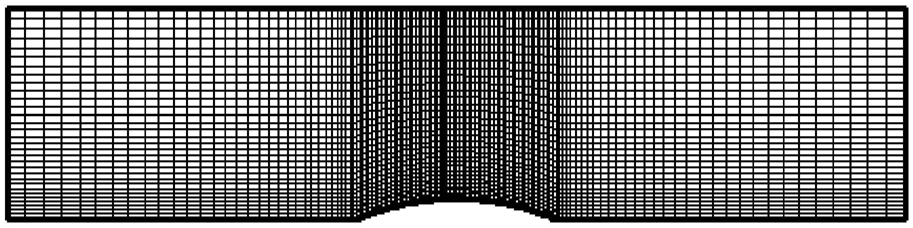
8 Blocks Grid



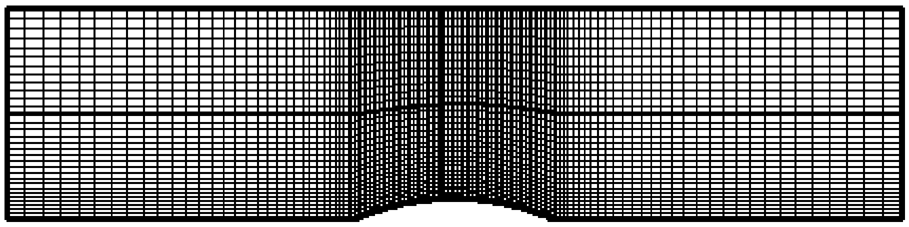
16 Blocks Grid

Figure 4-8 Multi-Block Grids for Axisymmetric Nozzle Problem

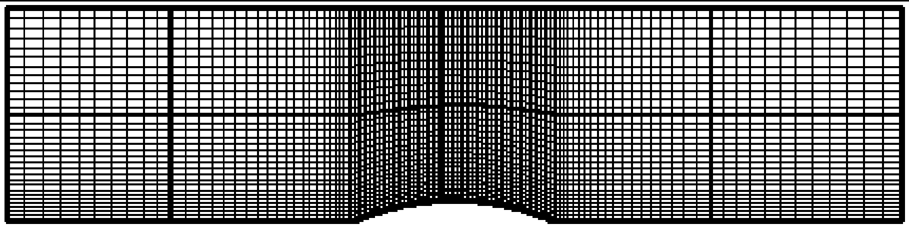
In following channel flow multi-block analyses; constructed blocks have nearly same sizes as shown in Figure 4-9. Totally 4257 nodes are tried to be distributed within the whole blocks equally for channel problem.



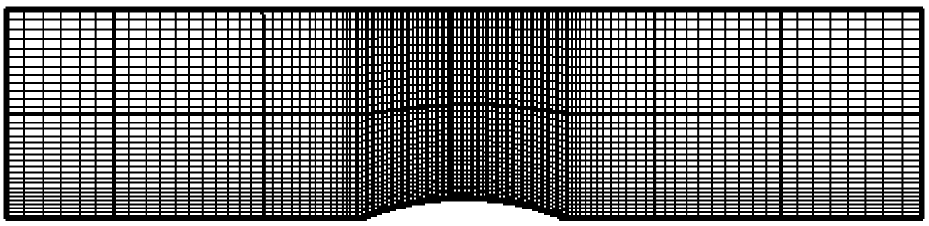
2 Blocks Grid



4 Blocks Grid



8 Blocks Grid



12 Blocks Grid

Figure 4-9 Multi-Block Grids for Channel Problem

4.4 Flow Solution

Axissymetric Nozzle problem is a good example for internal flow. Air enters from the inlet of the nozzle with 0.1 Mach and leaves from the outlet with nearly 3.25 Mach. Mach contours are shown in Figure 4-10 for different number of blocks.

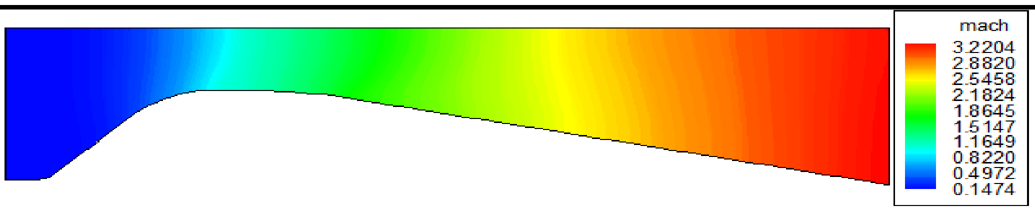
Having a supersonic inflow and outflow boundaries, channel problem is solved to examine the code performance in catching shock and expansion waves. Mach contours of the bump geometry are shown in Figure 4-11.

At all of the analyses; same solution parameters are used except $\Delta t,$ additional term for diagonal, which will be discussed later in this chapter.

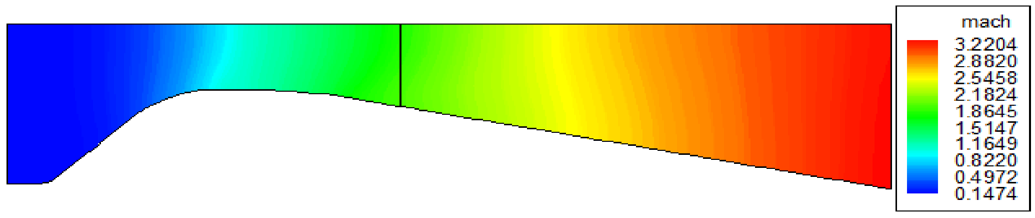
First of all, at all of the analyses; analytically derived flux Jacobian matrix entries are used. Numerical derivation method is not used for this comparison problems and PARDISO is used as a matrix solver. All of the results are converged solution results and the convergence criterion is below 10^{-12}. Residuals are calculated as the differences of the net fluxes that pass through a cell at each iteration. For convergence the normalized difference of the net fluxes are expected to be below 10^{-12}. Since discretization order and splitting scheme may change the results, second order discretization is used with Van Leer flux vector splitting scheme at all of the analyses.

As it is seen from Figure 4-10 and Figure 4-11, the block interface boundary conditions work well both for subsonic and supersonic flow conditions. As it is explained in Chapter 2, communications between the blocks are performed with using halo nodes. In these problems, single column halo nodes are used. Blocks are solved separately and two different results occur at the interface. Then the halo nodes are updated for the next iteration and the fluxes at the interfaces are averaged to make the blocks compatible. At steady state, the interfaces become completely transparent.

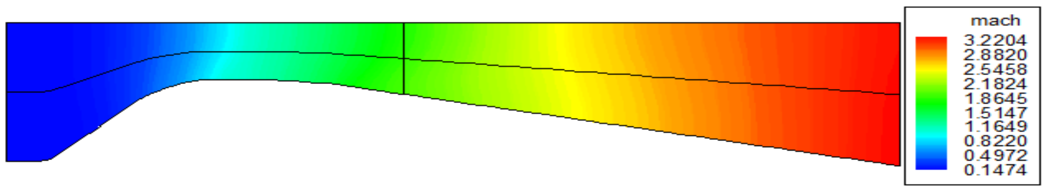
When the calculated flow variables between the two sides of the interface have much difference, the Newton method for the both blocks does not converge to the same result and may have a stability problem. For these cases, Δt the diagonal additional term, should be arranged logically to keep the both sides of the blocks compatible.



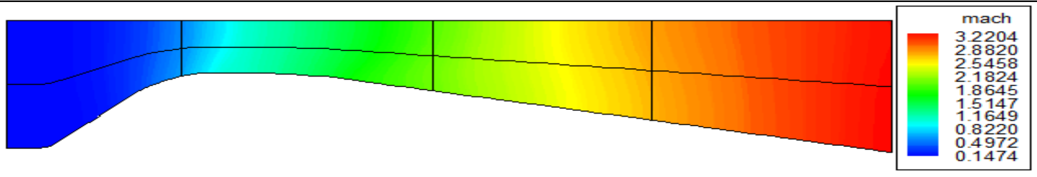
1 Block Contours



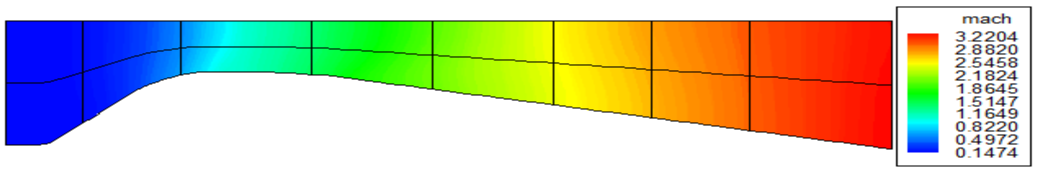
2 Blocks Contours



4 Blocks Contours

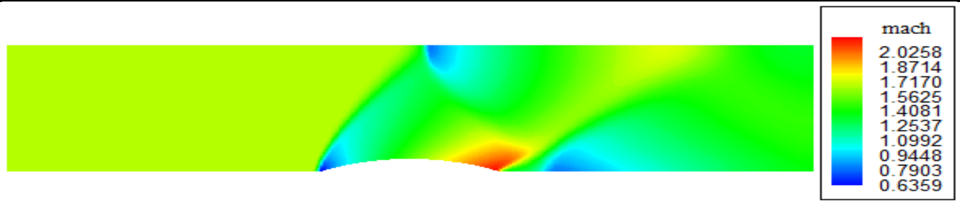


8 Blocks Contours

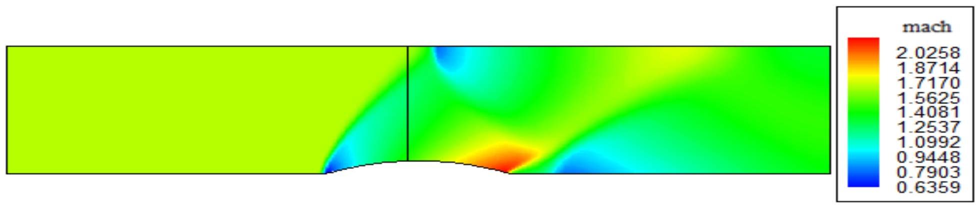


16 Blocks Contours

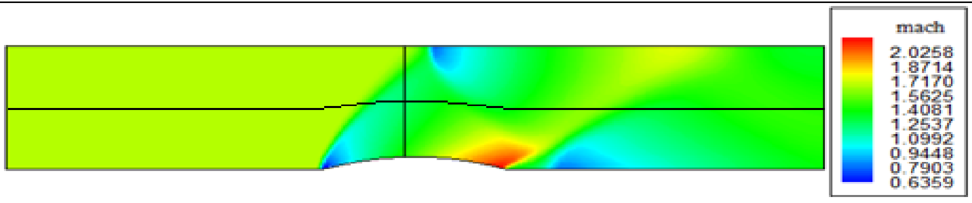
Figure 4-10 Mach Contours Comparison for Different Number of Blocks (Axisymmetric Nozzle Problem)



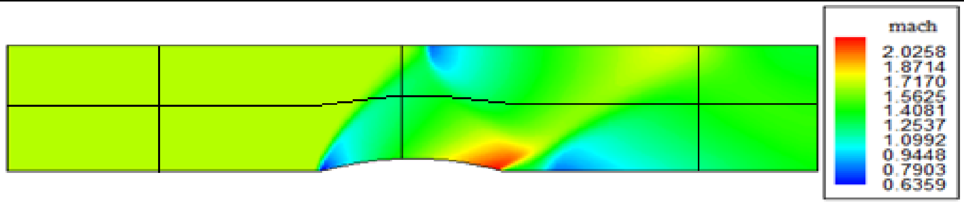
1 Block Contours



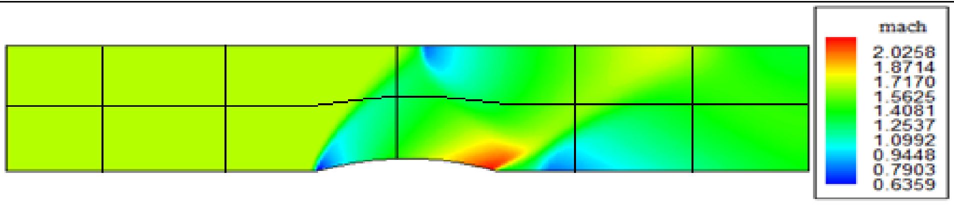
2 Blocks Contours



4 Blocks Contours



8 Blocks Contours



12 Blocks Contours

Figure 4-11 Mach Contours Comparison for Different Number of Blocks (Channel Problem)

4.5 Performance of Convergence

The most critical factor that affects the convergence of the Newton method is the way of term addition (Δt) to the diagonal of the Jacobian matrix. Newton scheme does not have convergence problems and has quadratic convergence rate if there is not an additional term to the diagonal. However, Newton method has problems when the initial conditions are poor. To improve the initial conditions, one has to use diagonal addition terms and when the sufficient propagation is reached the additional terms can be cancelled to make use of the quadratic convergence rate ability of the method. If the additional term is not cancelled after some propagation, the convergence rate will become linear and the required iteration number for convergence will increase. In Table 4.1 effect of $\boldsymbol{\Delta t}$ on convergence is shown for first order discretization.

Table 4.1 Effect of Δt on Convergence, First Order Discretization (Axisymmetric Nozzle)

		Iterations required for convergence				
Δt initial	Δt final	1 Block	2 Blocks	4 Blocks	8 Blocks	16 Blocks
1	5	601	828	1143	1398	1712
1	50	903	1156	1318	1550	1928
1	500	1180	1306	1670	1981	2345
1	5000	1616	1702	2103	2461	2811
10	50	35	89	118	176	250
10	500	67	96	121	176	274
10	5000	77	109	122	191	275
10	50000	93	111	122	193	275
100	500	11	24	60	92	113
100	5000	15	26	61	92	113
100	50000	17	28	63	92	113
100	500000	18	28	63	98	113
500	5000	9	21	50	77	107
500	50000	10	23	52	77	107
500	500000	11	23	52	77	107
500	5000000	11	23	53	77	107
5000	50000	7	19	62	72	113
5000	500000	8	20	63	72	113
5000	5000000	8	20	63	72	113
5000	50000000	8	21	67	72	113

In Table 4.2, effect of Δt on convergence is shown for second order discretization. Actually, it can be said that, there is not an explicit rule for calculation of Δt. Most suitable additional term can be found by trial and error method. Generally the lower Δt increases the number of iterations required for convergence and the higher one increase the risk of divergence. The required additional term depends on the problem parameters and the initial condition that is given to the program.

Table 4.2 of Δt on Convergence, Second Order Discretization (Axisymmetric Nozzle)

		Iterations required for convergence				
Δt initial	Δt final	1 Block	2 Blocks	4 Blocks	8 Blocks	16 Blocks
1	5	2748	2918	*	*	*
1	50	*	*	*	*	*
1	500	*	*	*	*	*
1	5000	*	*	*	*	*
10	50	126	292	*	*	*
10	500	178	311	*	*	*
10	5000	279	317	*	*	*
10	50000	337	317	*	*	*
100	500	12	23	80	*	*
100	5000	15	26	82	*	*
100	50000	17	26	82	*	*
100	500000	20	26	*	*	*
500	5000	10	19	61	714	1128
500	50000	11	19	63	727	1130
500	500000	12	19	63	741	1130
500	5000000	12	*	*	741	1130
5000	50000	7	21	68	*	*
5000	500000	9	19	69	*	*
5000	5000000	9	21	71	*	*
5000	50000000	9	*	*	*	*

(*) Not converged cases

Iterations required for convergence increases with increasing number of blocks. The single block Jacobian matrix is solved implicitly and it quadratically converges within nearly 10 iterations. However, when the whole domain is divided into some blocks, due to the interface boundary condition, the required number of iteration increases. When the interface boundary condition is used, the fully implicit property of the Jacobian matrices disappears with the interventions during each iteration.

In Figure 4-12 and in Figure 4-13 residuals are compared for different numbers of blocks. Same analysis parameters are used for all of the cases. Δt initial is 1000 and Δt final is 50000.

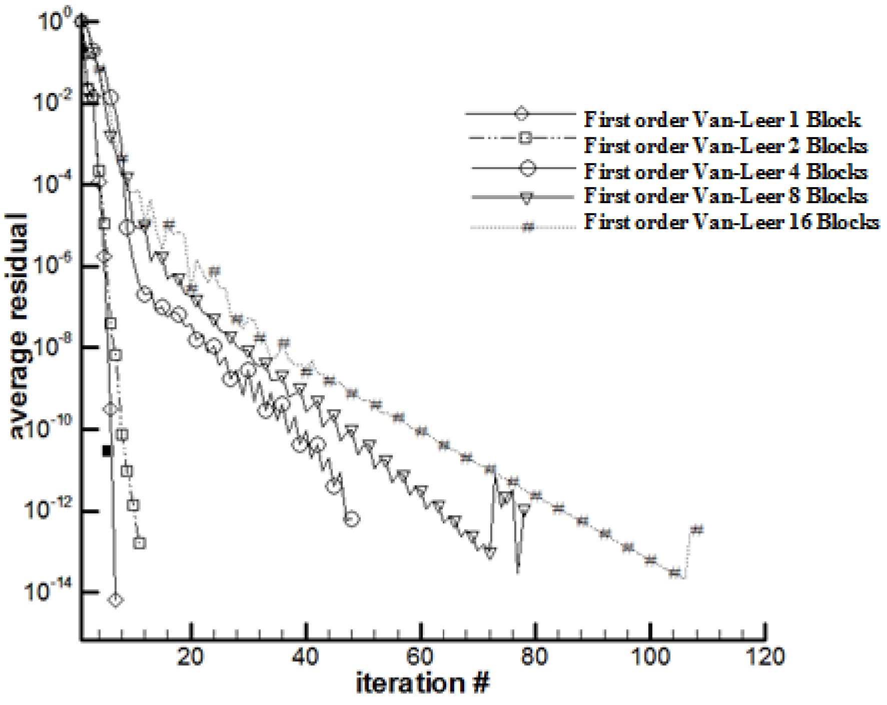


Figure 4-12 Residual Comparison for Δt initial =1000 Δt final=50000 (Axisymmetric Nozzle)

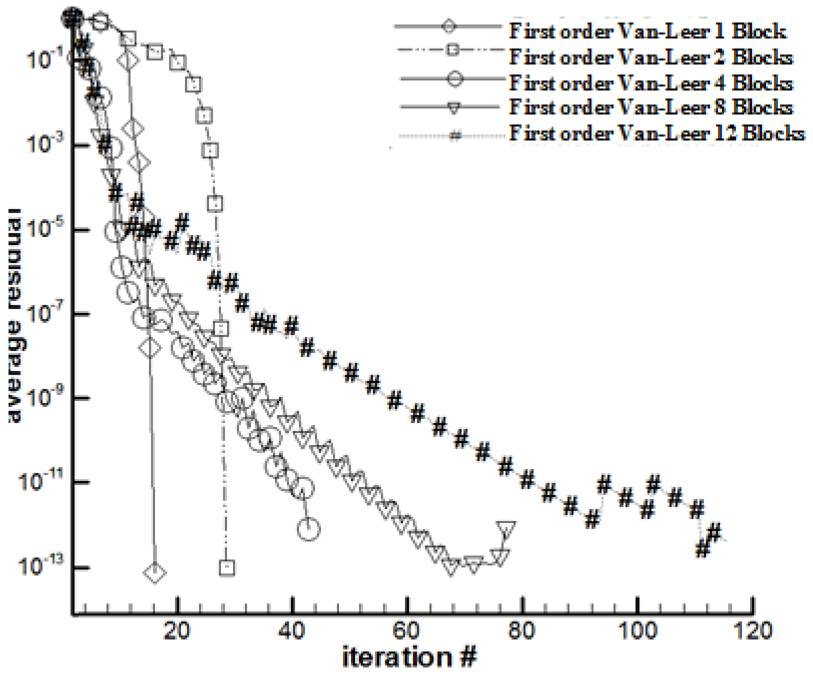


Figure 4-13 Comparison for Δt initial =1000 Δt final=50000 (Channel)

4.6 CPU time Required for Convergence

CPU time required for convergence is shown in the figures from Figure 4-14 to Figure 4-21. In Newton method analyses; construction of the Jacobian matrices and solving the constructed Jacobian matrices are two main parts that require more CPU time when compared with the other parts of the analyses. Jacobian matrices entries are calculated with analytical derivation method and the required time is directly proportional with the matrix size.

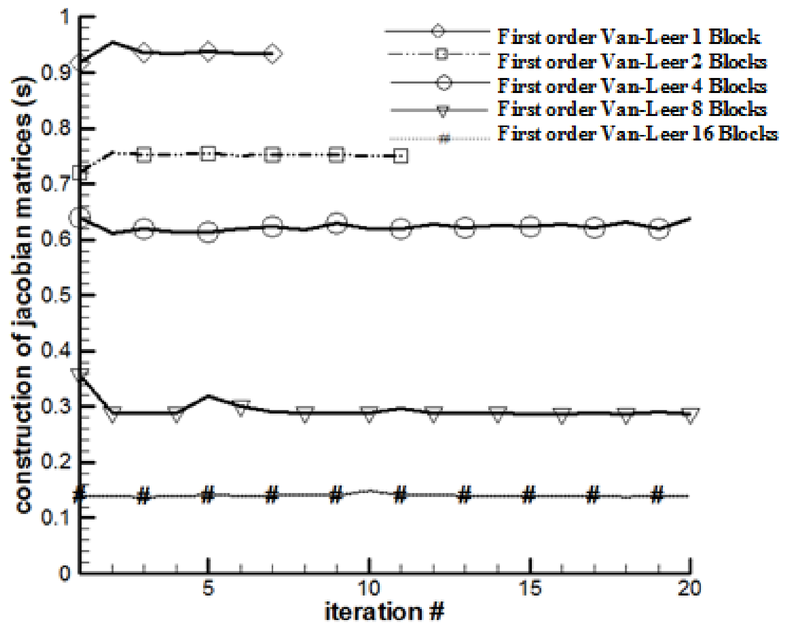


Figure 4-14 Time Required for Construction of Jacobian Matrices per one-block (First 20 iterations - Axisymmetric Nozzle)

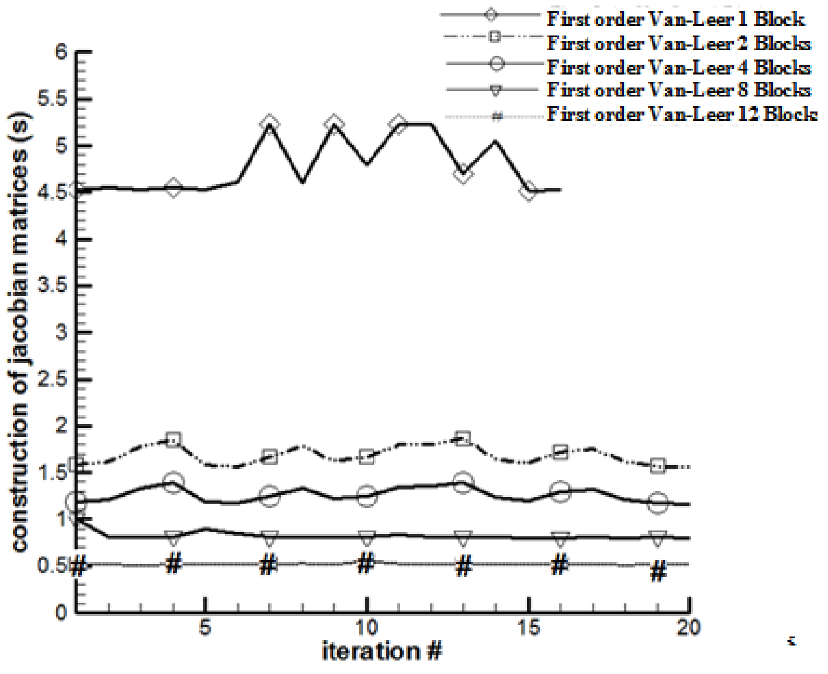


Figure 4-15 Time Required for Construction of Jacobian Matrices per one-block (First 20 iterations - Channel)

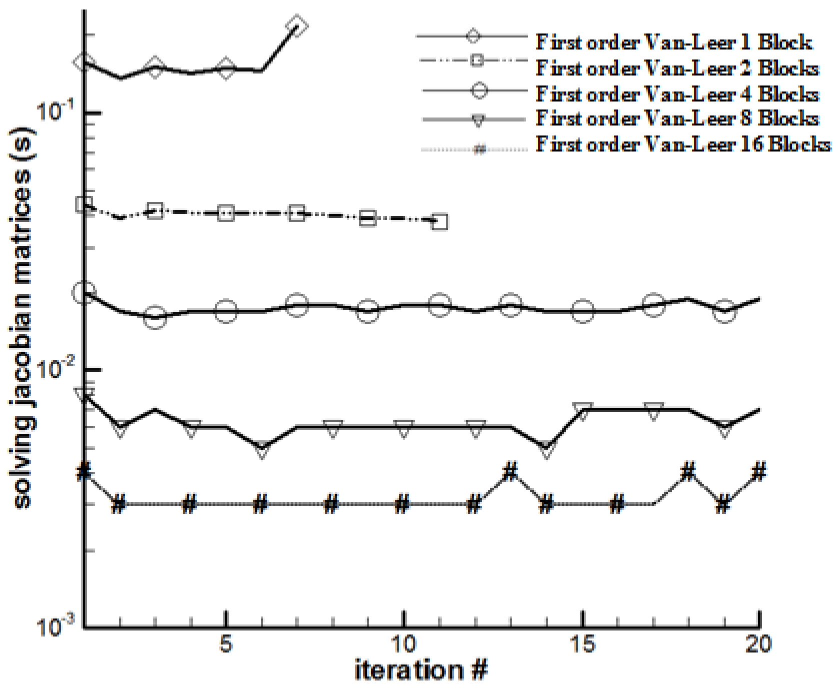


Figure 4-16 Time Required for Solving Jacobian Matrices per one-block (First 20 iterations - Axisymmetric Nozzle)

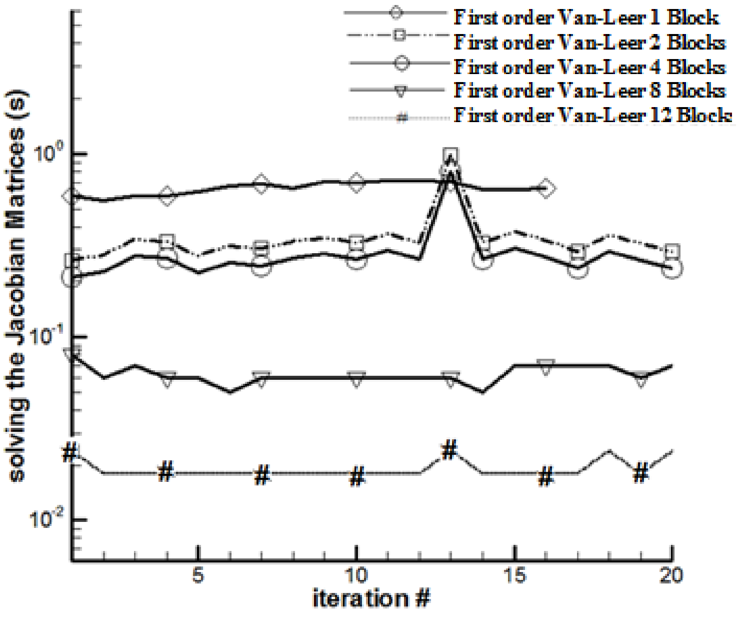


Figure 4-17 Time Required for Solving Jacobian Matrices per one-block (First 20 iterations - Channel)

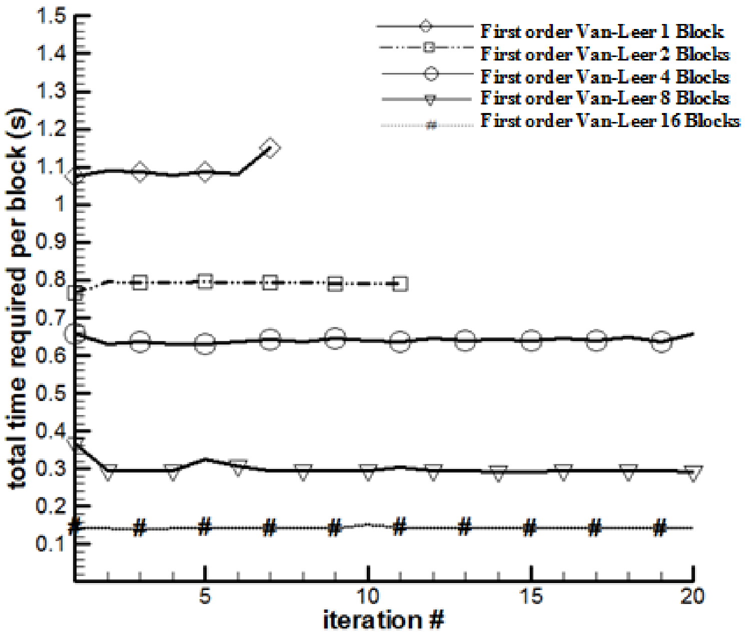


Figure 4-18 Total time required per one-block (First 20 iterations - Axisymmetric Nozzle)

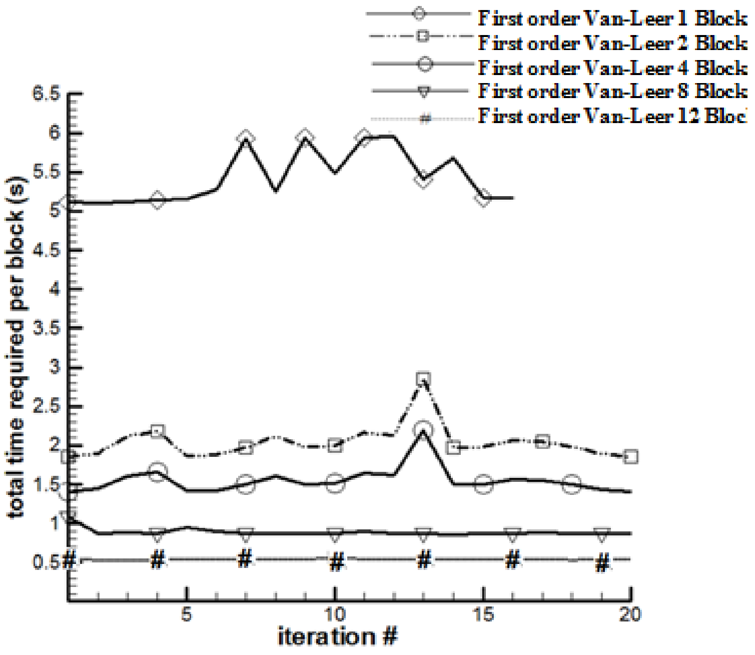


Figure 4-19 Total time required per one-block (First 20 iterations - Channel)

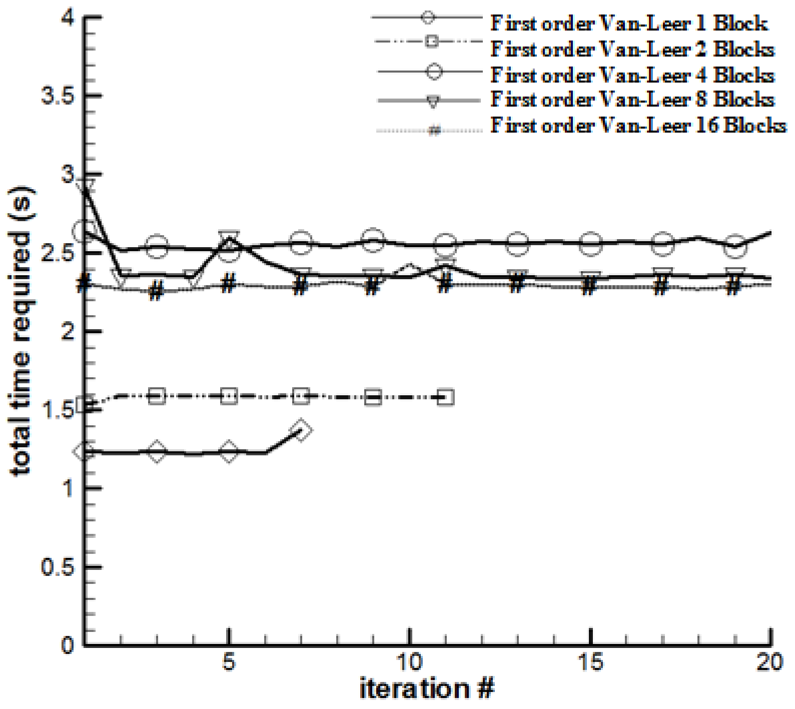


Figure 4-20 Total time required (First 20 iterations - Axisymmetric Nozzle)

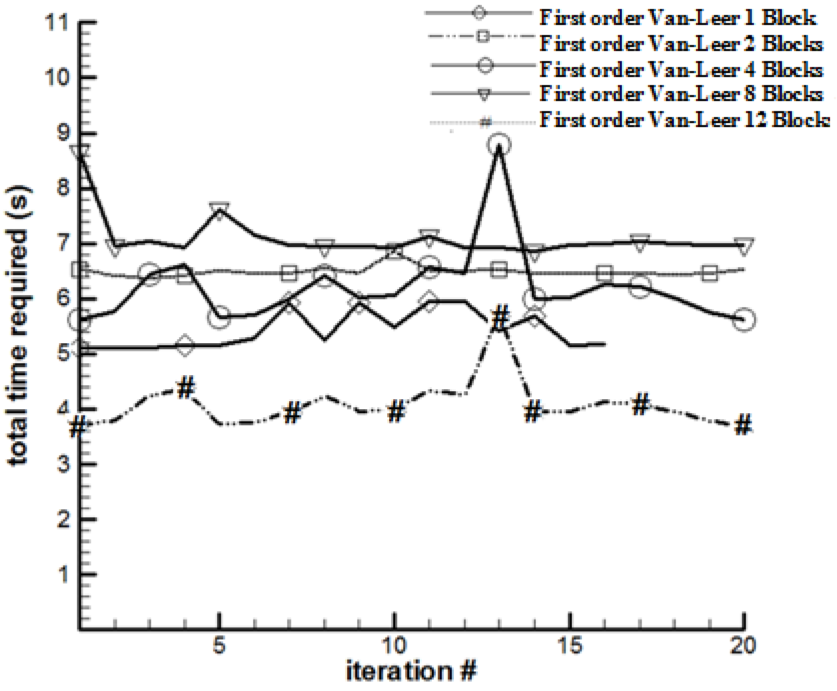


Figure 4-21 Total time required (First 20 iterations - Channel)

Table 4.3 Grid Size Information (Axisymmetric Nozzle)

	Total Grid Size	**Grid Size per One Block (Nearly)**
1 Block	82*21	82*21
2 Blocks	82*21	41*21
4 Blocks	82*21	41*11
8 Blocks	82*21	21*11
16 Blocks	82*21	11*11

With increasing number of blocks, due to the decreasing size of the each block, CPU time required for constructing and solving the Jacobian matrices per one block decreases. However, due to the increase in the required iteration number for convergence, the total time required rises with increasing number of blocks. In Table 4.3, total grid sizes and grid sizes per one block is given for following discussions

In Figure 4-14 and in Figure 4-15, CPU time required for construction of the Jacobian matrices is shown. The main factor that affects the required CPU time is the size of the Jacobian matrix. It generally linearly affects the required CPU time, so there is a linear decrease in the construction of the Jacobian matrices with increasing number of blocks per one iteration. The other factors which affect the CPU time required for construction of the matrix are flux vector splitting method, order of the discretization and the boundary conditions. Depending of the model, performed calculations vary.

In Figure 4-16 and in Figure 4-17 CPU time required for solving the Jacobian matrices per one iteration are demonstrated. Size of the matrix quadratically affects the required CPU time and with increasing number of blocks there is a quadratic decrease in solving the Jacobian matrices per one iteration.

Total time required per one iteration of the single block is given in Figure 4-18 and in Figure 4-19. One block case requires more CPU time as expected when only the results per one block are considered. In Figure 4-20 and in Figure 4-21 total CPU time of the all blocks per one iteration are shown. Increasing number of blocks generally increases the total required time. However, when the size of the each block in problems decreases so much, quadratic decrease in CPU time for solving the matrices affects the total CPU time and may keep it stationary. However, again when the total CPU time required for convergence compared, it can be said that, the increasing number of blocks rises the total CPU time at all conditions.

In the analyses; PARDISO and UMFPACK are used as sparse matrix solvers. Both of them are discussed in Chapter 3. They follow the same methodology to solve large sparse matrices. The only difference is PARDISO divides the Jacobian matrices into different CPU nodes and solves the Jacobian matrices parallel. On the other hand, UMFPACK does not have parallelization capability.

In Table 4.4, CPU time required for different size of grids at each iteration is shown. When the grid size increases, required CPU time for an iteration increases as expected. The big percentage of the increase in CPU time is due to the construction of the jacobian matrices. Increase in the grid size, augments the time required for solving the Jacobian matrices but the increase is not as much as in the construction part. Making multi-block grids decreases the total time required per one iteration especially when the grid size is greater. Performance of PARDISO is better than UMFPACK for all size of grids which are used in this thesis due to the parallelization capability in terms of CPU time required. 8 cores &2 CPU 2.33 GHz with 8 GB RAM are used for analyses. In order to decrease the total required CPU time, one should decrease the time required for construction of the Jacobian matrices. In Table 4.5 total CPU time comparison is given and as mentioned earlier increasing number of blocks rises the total CPU time. Having a greater contribution, matrix construction time can be decreased by parallelization of this part but the grid size for 2-D Euler solver is not as big as the ones those analyzed here. However, parallelization would be necessary when viscous force terms are added or when the 3-D version of the code is generated.

Table 4.4 CPU time required for different size of grids per one iteration (Axisymmetric Nozzle)

	Constructing the Jacobian Matrices (s)		*Solving the Jacobian Matrices (s)*			
			PARDISO		*UMFPACK*	
	One Block	*16 Blocks*	*One Block*	*16 Blocks*	*One Block*	*16 Blocks*
*82*21 Grid*	0.9	0.13*16=2.08	0.15	0.004*16=0.064	0.16	0.006*16=0.096
*405*60 Grid*	51	2.8*16=44.8	5.13	0.15*16=2.4	8.11	0.41*16=6.56
*810*120 Grid*	122	6.85*16=109.6	25.12	0.95*16=15.2	-	1.55*16=24.8

Table 4.5 Total CPU time comparison (Axisymmetric Nozzle)

	Time Required (s)			
	Per One Iteration		*Total*	
	One Block	*16 Blocks*	*One Block*	*16 Blocks*
*82*21 Grid*	1.05	2.14	18.9	516.392
*405*60 Grid*	56.13	47.2	1010.34	5569.6
*810*120 Grid*	147.12	124.2	2648.16	14655.6

CHAPTER 5

CONCLUSION

In this thesis, exact multi-block Newton's method was successfully applied for 2-D Euler equations. In house developed tool was used for the analyses and it was improved throughout the thesis study. Euler equations were discretizied by using finite volume method with different upwind methods and second order accuracy was used for the spatial discretizations. Multi-block analyses were performed by using halo nodes and corresponding modifications were done in analytical Jacobian calculations and matrix solver parts. Block interface boundary conditions were added and made compatible with the different flow regimes. Generated tool was made adaptable to the different type of problems with implementations of the various boundary conditions. Two different matrix solver programs, UMFPACK and PARDISO were used for the Jacobian matrix solutions. Both of them have capability to solve highly sparse large matrices, which are inevitable in this study and they were made compatible with the generated tool. Airfoil and channel flow problems were used for verification of the tool.

Multi-block performance analyses were made for axisymmetric nozzle and channel flow problems. Accuracy, convergence and CPU time spent comparison was made by dividing the whole domain into different number of blocks. Flow contours were investigated around the block interfaces to examine the data transfers between the neighboring blocks. Grids having different sizes were used for these analyses to make reliable comparisons. In these comparisons; it was seen that the increasing number of blocks rises the total CPU time spent due to the increase in the required iterations for convergence. The greater part of the total CPU time spent aroused from Jacobian construction part and it was tried to be reduced by simplifications in the corresponding subroutines. CPU time spent of the UMFPACK and PARDISO matrix solvers were compared and using PARDISO was seen to be more suitable for this analysis because of its parallelization capability.

In the performance analyses, it was seen that with good initial guess, Newton's method provides quadratic convergence rate. Poor initial guesses, on the other hand, caused divergence or tremendous increase in the required iterations. Diagonal addition term was used in the analysis to strength the initial guesses. Required iteration numbers were tabulated for different values of additional term. Initial and the withdrawal values of those added diagonal terms were found to be critical factors on the convergence of the solution.

In order to increase the accuracy, viscous force calculation terms and turbulence models should be added to the generated tool. These additions would require higher grid sizes for the analyses. Those increases in the grid size will cause more CPU time and the parallelization of the Jacobian matrix construction part would be necessary.

Usage of the exact Newton Method is not suitable for very complicated problems and when the flow domains have higher grid sizes. Instead of exact Newton method, matrix free Newton Krylov method or inexact Newton method can be used for these types of problems. Moreover, for multi-block analyses simultaneously approaching terms (SAT) with summation by parts (SBP) approach can be used to replace halo nodes to increase accuracy

REFERENCES

[1] Wington, 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] Van Dam, C. P., M. Hafez and J. Ahmad, "Calculations of viscous flow with separation using Newton's method and direct solver", AIAA Journal, Vol. 28, No. 5, 1990, pp. 937-939.

[5] 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.

[6] 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.

[7] Orkwis, P. D., and Mc Rae, D. S., "Newton's Method Solver for High Speed Viscous Separated Flowfields", AIA Journal, Vol. 30, January 1992, pp. 1507-1514.

[8] 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.

[9] Kim, D. B., and Orkwis, P. D., "Jacobian 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 and Meeting&Exhibit, Reno, Nevada, January 1993.

[10] Whitfield, D. L., and Taylor, L. K., “Discretizerd Newton-Relaxation Solutionof 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 Iteartive Algorithms for the Three-Dimensional Euler Equations”, AIAA Paper 93-3378, 1993.

[13] Orkwis,P.,D., and Venden, K.,J., ”On the Accuracy of Numerical Versus Analytical Jacobians”, AIAA Paper 94-0176, Proceedings of the AIAA 32nd Aerospace Science Meeting, Reno, Neveda1994, January 1994.

[14] Saad, Y., and Schultz, M.H., “GMRES: A Generalized Minimal Residual Algorithm For Solving Non-Symmetric Linear Systems”, SIAM J. Sci. Stat. Comput. Vol. 7, July 1986.

[15] Venkatakrishnan, V., and Mavripilis, D.J., “Implicit Solvers for Unstructured Meshes”, AIAA-91-1537-CP, 1991.

[16] Venkatakrishnan, V., “Implicit Schemes and Parallel Computing in Unstructured Grid CFD”, ICASE report 95-28 CR-195071, NASA, 1995

[17] Rogers, S.E., “A Comparison of Implicit Schemes for the Incompressible Navier Stokes Equations with Artificial Compressibility”, AIAA 95-0567, January 1995

[18] Forsyth, P.A., and Jiang, H., “Iterative Methods for Full Newton Solution of the Euler Equations”, Sixth International Symposium on Computational Fluid Dynamics, pp. 318-323 , Lake Tahoe, Nevada, September 1995

[19] Brown, P., and Saad, Y., “Convergence Theory of Nonlinear Newton Krylov Algorithms”, SIAM J. Optimization, Vol. 4., pp. 297-330

[20] Davis, T. A., UMFPACK Version 4.1 User Manual, University of Florida

[21] Amestoy, P. R., Duff, I. S. and L'Excellent, J. –y., "Multifrontal parallel Distributed Symmetric and Unsymmetric Solvers", Computer Methods in Applied Mechanics and Engineering, Vol. 184, Issues 2-4, 14 April 2000, pp. 501-520

[22] Gupta, A., "Recent Advances in Direct Methods for Solving Unsymmetric Sparse Systems of Linear Equations", ACM Transactions on Mathematical Software, Vol. 28(3), 2002, pp.301-324.

[23] Schenk, O. and Gartner, A., "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO", Journal of Future Generation Computer Systems, Vol.20, 2004, pp.475-487.

[24] Onur, O. and Eyi, S., "Effects of the Jacobian Evaluation on Newton's Solution of the Euler Equations", *International Journal for Numerical Methods in Fluids*, Vol.49, pp 211-231,2005.

[25] Gelfgat, A. Y., "Stability of the Convective Flows in Cavites: Solution of benchmark problems by a low-order finite volume method", *International Journal for Numerical Methods in Fluids*, Vol.53, pp 485-506.

[26] Raju,M.P,Tien,J.S. "Development of Direct Multifrontel Solvers for Combustion Problems", *International Journal of Computation and Methodology,* 1521-0626, Vol. 53, Issue 3,2008,pp.189-205

[27] Ezertaş A. A., Master of Science Thesis,"Sensitivity Analysis Using Finite Difference and Analytical Jacobians" METU 2009

[28] Nichols J. C., Zingg D. W., "A Three-Dimensional Multi-Block Newton-Krylov Flow Solver for the Euler Equations", AIAA paper, 2005

[29] Kam D. C. W.,Master of Science Thesis, "A Three Dimensional Newton-Krylov Navier-Stokes Flow Solver Using a One-Equation Turbulence Model", University of Toronto, 2007

[30] Rumpfkeil M. P., Doctor of Philosophy, "Airfoil Optimization for Unsteady Flows with Application to High-Lift Noise Reduction" University of Toronto, 2008

[31] Nemec M. Doctor of Philosophy, "Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach" University of Toronto, 2003

[32] Hicken J. E.,, Doctor of Philosopy, "Efficient Algoriths for Future Aircraft Design: Contributions to Aerodynamic Shape Optimization" University of Toronto, 2009

[33] Leung T. M-M, "A Newton-Krylov Approach to Aerodynamic Shape Optimization in Three Dimensions" University of Toronto, 2010.

[34] Huan X, Hicken J. E., Zingg D. W., "Interface and Boundary Schemes for High-Order Methods" , AIAA paper, 2009.

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

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

[37] Liou, M.-S. "A sequel to AUSM: AUSM+" Journal of Computational Physics, Vol 129(1996),pp.364-382

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

[39] Van Leer, B., "Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov's Method", Journal of Computational Physics, Vol. 32, "979,pp."0"-"36.

[40] Van Albada, G.D., Van Leer, B., Roberts, W.W., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics", Astronomy and Astrophysics, Vol 108, 1982, pp. 76-84,

[41] Venkatakrihnan, V., "Preconditioned conjugate gradient methods for the compressible Navier- Stokes equations", AIAA Journal, Vol. 29, June 1991, pp. 1092-1100.

[42] Blazek J., "Computational Fluid Dynamics: Principles and Applications", Elsevier, 2005.

[43] Ni, R. H., "Aa Multiple-Grid Scheme for Solving the Euler Equations", AIAA Journal, Vol.20, no.11, 1982, pp.1565-1571.

[44] AGARD Subcomitte C., Test Cases for Inviscid Flow Field Methods, AGARD Advisory Repor 2111, 1986.

.