PERFORMANCE COMPARISON OF NEWTON AND NEWTON-GMRES METHODS IN 3-D FLOW ANALYSIS

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

BY

BUKET YILDIZLAR

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

FEBRUARY 2014

Approval of the thesis:

PERFORMANCE COMPARISON OF NEWTON AND NEWTON-GMRES METHODS IN 3-D FLOW ANALYSIS

submitted by **BUKET YILDIZLAR** 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 Sciences** _____

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

Assoc. Prof. Dr. Sinan Eyi
Supervisor, **Aerospace Engineering Dept., METU** _____

Examining Committee Members

Prof. Dr. Sinan Akmandor
Aerospace Engineering Department, METU _____

Assoc. Prof. Dr. Sinan Eyi
Aerospace Engineering Department, METU _____

Assoc. Prof. Dr. Oğuz Uzol
Aerospace Engineering Department, METU _____

Assoc. Prof. Dr. D. Funda Kurtuluş
Aerospace Engineering Department, METU _____

Dr. Özgür Ekici
Mechanical Engineering Department, Hacettepe University _____

Date: 07.02.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Buket Yıldızlar

Signature :

ABSTRACT

PERFORMANCE COMPARISON OF NEWTON AND NEWTON-GMRES METHODS IN 3-D FLOW ANALYSIS

Yıldızlar, Buket

M.S., Department of Aerospace Engineering

Supervisor: Assoc. Prof. Dr. Sinan Eyi

February 2014, 53 pages

Because of CPU time problems, an alternative to Newton's method is investigated in order to make a flow analysis in a 3-D Supersonic Nozzle. Calculation and forming of the Jacobian matrix get harder as the system gets larger. On the contrary, Newton-GMRES approach does not require direct access to the Jacobian matrix. Due to the fact that it provides a dramatic decrease in CPU time, Newton-GMRES method is examined.

To compare their performance on a supersonic nozzle, 3-D Euler Equations are solved with Newton's and Newton-GMRES methods respectively. A parametric study is conducted for Newton-GMRES method to find the optimal solution with respect to CPU times elapsed. In order to analyse Newton-GMRES method's behavior on larger systems, different test cases are generated. The code is developed for Newton-GMRES method with Fortran77.

Keywords: 3-D Euler Equations, Newton-GMRES Method, Newton's Method Jacobian

ÖZ

3-BOYUTLU AKIŞLARDA NEWTON VE NEWTON-GMRES YÖNTEMLERİNİN PERFORMANSLARININ KARŞILAŞTIRILMASI

YILDIZLAR, Buket
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Sinan EYİ

Şubat 2014, 53 sayfa

Newton Yöntemi 3 Boyutlu Süpersonik Lülelerde yapılan akış analizlerinde çok fazla CPU gereksinimi yaratmaktadır. Bunun doğurduğu maliyeti azaltma isteği, yeni bir yöntem arayışına sebep olmuştur. Jakobi matrisinin hesaplanması ve oluşturulması özellikle sistem büyüdükçe zorlaşmaktadır. Diğer bir yandan Newton-GMRES yöntemi, direkt olarak Jacobi matrisine ihtiyaç duymaz. Özellikle büyük ve seyrek matrislerle sağladığı yakınsaklık değeri ve CPU'daki bariz tasarrufu sebebiyle Newton-GMRES metodu ele alınmıştır.

3 Boyutlu Süpersonik bir lüledeki performanslarını karşılaştırmak amacıyla 3 Boyutlu Euler denklemleri sırayla her iki yöntemle çözülmüştür. Newton-GMRES yönteminde kullanılan parametreler üzerinden bir kıyaslama ve optimum elde etme çalışması gerçekleştirilmiştir. Newton-GMRES yönteminin büyük sistemlerdeki performansını gözlemleyebilmek için farklı büyüklüklerde test durumları yaratılmıştır. Newton-GMRES yöntemi için gerekli kod Fortran 77 yardımıyla geliştirilmiştir

Anahtar kelime; 3 Boyutlu Euler Denklemleri, Newton-GMRES Yöntemi, Newton Yöntemi, Jakobi Matrisi.

To everybody who loves me

ACKNOWLEDGEMENTS

I would like to express my appreciation to my thesis supervisor Assoc. Prof. Dr. Sinan Eyi for his great guidance and invaluable help during my master thesis.

I wish to express my sincere thanks to my thesis committee, Prof. Dr. İ.Sinan Akmandor, Assoc. Prof. Dr. Oğuz Uzol, Assoc. Prof. Dr. D. Funda Kurtuluş and to Dr. Özgür Ekici.

I am grateful to Naz Tuğçe Öveç for helping me on coding. Without her help, I could not even start

Special thanks to Sogand Yousefbeigi and Tuğçe Garip for their great support and motivation

I also thank to all of my friends that tried to help me as much as they can, no matter the problem was.

I am thankful to my parents and sisters for their trust and patience. Also thanks to our new family member Kaan for bringing joy to my last 3 months with his nice photos.

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK-112M129)

.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background.

CFD is an important research area which has been used most of the engineering and science disciplines since the computer technology is progressing day by day. As long as it consumes plenty of time and effort to solve physics of flow either analytically or experimentally, the demand of estimation of the solution of a flow by the help of CFD methods is increasing more than expected

Explicit methods were one of the oldest concerns of CFD society. Due to the memory problems of computers, they were the only alternative to direct solution method. The main disadvantage of these explicit methods is stability. Recent advances in computer technology and solution algorithms allow efficient solution of very large linear systems of equations. These advances have been motivating researchers to develop implicit algorithms to solve the flow equations since usage of implicit methods is more beneficial compared to the explicit ones. Implicit flow solvers are more stable and the residual can be reduced to very low values within a small number of iterations. Today, still there is a tendency to use explicit schemes for some class of unsteady flow problems. However the equations of different disciplines can be strongly coupled with flow equations in an implicit algorithm.

Newton's method can be named as the most common implicit solution technique in CFD. It is a root finding algorithm in the vicinity of a suspected root. It provides quadratic convergence. In spite of this superior convergence, it requires the exact linearization of the residuals. This linearization process can only be done after

calculations of Jacobian matrices. They are matrices which are composed of derivatives of flux vectors with respect to flow variables. In large sparse systems, it requires a sustainably large computer memory to store the Jacobian matrix. Also evaluation and factorization of the Jacobians are other important issues that need memory. Although the size of this matrix can be very large, it is sparse in the most of the flow problems. The selection of good initial solution is important in Newton method. If the initial solution is not chosen properly, Newton method may diverge .

To keep the advantages and to avoid disadvantages of Newton method, Jacobian-free Newton methods are getting more attention and especially the memory problem led researchers to canalize to inexact Newton methods instead of exact Newton method. Inexact Newton methods are iterative techniques to solve the flow in an approximate manner in order to benefit from the amount of work per iteration. Quasi-Newton methods belong to the class of Inexact Newton methods. The idea behind this class of methods is to solve the linear system arises from each Newton step with a linear solver.

One of these Quasi-Newton methods is Newton-GMRES method. GMRES is derived as a linear solver for large, sparse, unsymmetric semi-positive definite systems. It belongs to Krylov subspace methods family. Krylov methods are the most common used linear iterative solvers and GMRES is the most preferred one. It is very attractive while dealing with Jacobian matrices. It does not need exact Jacobian. One can decrease the CPU time needed to solve the flow by using quasi-Newton method. On the other hand, loss of the superior convergence that is supplied by Newton's method is inevitable. The question is the limit of this exchange between work load and accuracy. The aim of study is to make a comparison between Newton's Method and Newton-GMRES method on a supersonic nozzle geometry.

1.2 Scope Of The Thesis

The object of this study is to implement Newton-GMRES method into supersonic flows as an alternative to Newton method which is a reliable and efficient solver. The flow analysis is based on the three dimensional Euler equations. More than one goal

are planned to achieve. A unique version of the Newton-GMRES code is one of the targets of this work. Then implementation of it into 3-D Euler equations is carried out in order to observe its performance on a 3-D supersonic nozzle. To make a reliable comparison, same order of residual is aimed to achieve in both of the methods Newton and Newton-GMRES.

1.3 Literature Survey

As it was mentioned in the background part, with the help of improvement in computer technology, implicit methods gained value in CFD in the early nineties. Implicit methods gave the scientists the easiness of coupling the equations from different disciplines with flow equations in one algorithm. The most favorable of these techniques was Newton's method because of its superior convergence. With respect to other iterative techniques, Newton's method results in earlier iterations, which makes it attractive.

Wigton [1] did one of the first implementations of Newton's method. While solving a transonic flow over a multi-element airfoil with conventional solution techniques, he faced with unsuccessful convergence histories. Then, he implemented Newton's method into flow. To compute the Jacobian matrix, he preferred to use MACSYMA, a symbolic mathematics system. He developed a method named nested dissection node reordering which decreases the memory requirements of large spare systems' storage. Due to the efficient results obtained, Newton's method became a promising method in 3-D applications. Today still similar techniques are used for large sparse systems.

One of the important disadvantages of Newton's method is the significance of initial conditions selection. Bender and Khosia [2] studied on the use of Newton's method for the solution of inviscid compressible and viscous incompressible transonic flows. They concentrated on vanishing the problem of sensitivity of Newton's method on initial conditions. It was concluded that the minimization of Euclidian norm reduces this importance of initial conditions selection.

Venkarakrishan [3] also worked on the same topic with Wigton: solution of transonic flows over an airfoil. It was one of the most remarkable implementation of Newton's method and the modification of adding diagonal term into the advance sparse matrix solution method enhanced the convergence even better than Bender and Khosia in case of using poor initial conditions. They demonstrated that the method displayed a quadratic convergence. One of his deductions was the usage of Newton's method's inconvenience since it requires high CPU as the system gets larger. In other words, he pointed out that application of Newton's method was impractical on 3-D problems. With his conclusions, scientists started to interest in quasi Newton's methods.

Orkwis [4] worked both exact Newton's and quasi-Newton's methods and he showed that quasi-Newton's method solvers did not exhibit quadratic convergence, but could be more efficient than the exact Newton's method in selected cases in the meaning of lower memory requirements.

The main idea of the quasi-Newton's methods is to combine a linear solver with Newton's method and to solve Newton equations arise in each step with this linear solver. The Jacobian matrices are either simplified or approximated. As long as there are many kinds of iterative solvers, there is a wide range of this type of solution techniques. As Krylov subspace methods have a special feature of using matrices just as an operator, there is a demand to this class of linear solvers. The most preferred one in Krylov family is GMRES [5].

Before GMRES, Conjugate gradient like methods were used as solver for symmetric positive definite matrices [6]. With combining other solution procedures it provides rapid and reliable convergence histories. However it did not work with nonsymmetric systems. [6]. Since nonsymmetric matrices are very common types faced in CFD, CG like methods didn't become popular in CFD community. In early eighties some implementations of CG with modifications were tried with nonsymmetric matrices by Elman [7]. GMRES was found by Saad and Schultz in

1983 as a variant of CG methods that can be applied to nonsymmetric systems. It finds out the best possible solution in *k*-dimensional Krylov subspace.

GMRES can also be applied to linear systems without combining with a Newton's method. Wigton [8] derived another version of GMRES that is applicable to nonlinear systems in 1984. The most important drawback of this technique was storage problem. It requires much more memory capacity due to this nonlinearity fact.

To reduce this memory requirement matrix free methods combining with Krylov linear solvers were begun to use. Gear and Saad [9], Brown and Hindmarsh [10] and Chan and Jackson [11] are some of the earlist works done on this subject. Brown and Saad [12] are the first who introduced matrix free Newton-GMRES method. In order to improve the global convergence properties, they also proposed a procedure called linesearch backtracking.

1.4 Outline

In the second Chapter flow model is formed. Governing 3-D Euler Equations in generalized form are introduced. Spatial discretization types used are explained. General information on upwind schemes is given and Van Leer flux splitting method is presented which was selected to be used in Newton Method. Boundary conditions that were applied on geometry are briefly described.

Third Chapter is composed of solution algorithms. Newton Method is defined. Selected version of Jacobian matrix calculation is explained. Then Newton-GMRES method is presented. Information on Givens rotation and minimization is given. Commonly used preconditioner types are introduced.

Convergence histories of Newton and Newton-GMRES methods are given in Chapter 4. Plots of Mach number, pressure and velocity vectors in both Newton and Newton-GMRES methods are shown. Comparisons of CPU-time in each test cases

are done. The effect of parameters used in Newton-GMRES method on convergence acceleration and CPU time are examined in “Parametric Study” part. Preliminary analysis is also added under the section of “Solution of Three Equations”.

Conclusion is performed in last chapter. Future work suggestions are done.

CHAPTER 2

FLOW ANALYSIS

Prediction of behavior of fluids in design phase saves scientists from huge financial loss. For given specific flow conditions, flow model should be formed in such a way that selections must have the capability to protect physics of the flow. The flow model directly affects the accuracy of the solution. Advanced flow models are composed of advanced time discretization, dense grids and good selection of boundary conditions. Using highly resolved grids and/or complex flow properties will be reflected in cost of computation. Simplified versions would reduce that cost while decreasing the accuracy of the flow simulation.

Size of the Jacobian is a function of the number of equations solved and the size of the grid. To solve a flow, either Navier-Stokes or Euler Equations can be selected. It should be noted that, Navier-Stokes equations requires much more memory, which is one of the disadvantages of them. As long as our study is composed of a comparison of Newton's method and Newton-GMRES, size of Jacobian is still important. 3-D Euler Equations are solved in order to keep Jacobian size smaller in Newton's method.

2.1 Governing Equations

Euler Equations are simplified versions of Navier-Stokes Equations. One can obtain them by excluding viscous terms. They can be named as approximation to Navier-Stokes equations. Euler equations derived from conservation of mass (continuity), momentum, and energy in a control volume. 3-D Euler Equations in Cartesian coordinates can be written as follows under assumptions of steady inviscid flow:

$$\frac{\partial F(Q)}{\partial x}+\frac{\partial G(Q)}{\partial y}+\frac{\partial H(Q)}{\partial z}=0 \tag{1}$$

where

$$Q=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho e_t\end{bmatrix} \qquad F=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ (\rho e_t+p)u\end{bmatrix} \tag{2}$$

$$G=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho wv\\ (\rho e_t+p)v\end{bmatrix} \qquad H=\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ (\rho e_t+p)w\end{bmatrix}$$

Here, Q denotes flow variables vector where the others showing the flux vector. *u, v* and *w* are the components of velocity in x y and z coordinates respectively. ρ is the density, *p* pressure and e_t is the total energy per unit volume. γ is defined as specific heat ratio. Pressure can be computed from ideal gas relation as follows:

$$p=(\gamma-1)\rho(e_t-\frac{u^2+v^2+w^2}{2}) \tag{3}$$

As it is much easier to work with generalized coordinates on an arbitrary geometry, the governing equations are transformed into generalized coordinates from Cartesian coordinates. While *x,y* and *z* are the Cartesian coordinates, ξ, η and ζ are curvilinear coordinates.

$$\begin{aligned}\xi&=\xi(x,y,z)\\ \eta&=\eta(x,y,z)\\ \zeta&=\zeta(x,y,z)\end{aligned} \tag{4}$$

Partial derivates in the Cartesian coordinates $\frac{\partial}{\partial x},\frac{\partial}{\partial y},\frac{\partial}{\partial z}$ are transformed into generalized form as:

$$\frac{\partial}{\partial x}=\xi_x\frac{\partial}{\partial \xi}+\eta_x\frac{\partial}{\partial \eta}+\zeta_x\frac{\partial}{\partial \zeta}$$
$$\frac{\partial}{\partial y}=\xi_y\frac{\partial}{\partial \xi}+\eta_y\frac{\partial}{\partial \eta}+\zeta_y\frac{\partial}{\partial \zeta}$$
$$\frac{\partial}{\partial z}=\xi_z\frac{\partial}{\partial \xi}+\eta_z\frac{\partial}{\partial \eta}+\zeta_z\frac{\partial}{\partial \zeta} \qquad (5)$$

ξ_x, ξ_y, ξ_z, η_x, η_y η_z,ζ_x, ζ_y and ζ_z are the transformation metrics and they can be computed by the following procedure:

$$d\xi=\xi_x dx+\xi_y dy+\xi_z dz \qquad dx=x_\xi d\xi+x_\eta d\eta+x_\zeta d\zeta$$
$$d\eta=\eta_x dx+\eta_y dy+\eta_z dz \qquad dy=y_\xi d\xi+y_\eta d\eta+y_\zeta d\zeta$$
$$d\zeta=\zeta_x dx+\zeta_y dy+\zeta_z dz \qquad dz=z_\xi d\xi+z_\eta d\eta+z_\zeta d\zeta \qquad (6)$$

Hence;

$$\begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix}=\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix}\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \qquad \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}=\begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}\begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} \qquad (7)$$

An equality can be written between these two transformation matrices as follows. Also, the determinant of one of these matrices is equal to transformation Jacobian.

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix}=\begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1}$$

$$J=\det\left\|\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix}\right\| \qquad (8)$$

By using metrics, the 3-D Euler equations in Cartesian coordinates may be transformed into generalized coordinates is written in a form of:

$$\frac{\partial \hat{F}(\hat{Q})}{\partial \xi}+\frac{\partial \hat{G}(\hat{W})}{\partial \eta}+\frac{\partial \hat{H}(\hat{W})}{\partial \zeta}=0 \qquad (9)$$

where

$$\hat{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix}, \qquad \hat{F} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (\rho e_t + p) U \end{bmatrix}$$

$$\hat{G} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (\rho e_t + p) V \end{bmatrix}, \qquad \hat{H} = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (\rho e_t + p) W \end{bmatrix} \tag{10}$$

U, *V* and *W* are contravariant velocities. They are calculated as:

$$U = \xi_x u + \xi_y v + \xi_z w$$
$$V = \eta_x u + \eta_y v + \eta_z w$$
$$W = \zeta_x u + \zeta_y v + \zeta_z w \tag{11}$$

2.2 Spatial Disretization

Finite volume method is applied as spatial discretization. In FVM, the flow domain is divided into cells. The flow variables are stored in the center of the cells. Fluxes are computed on the cell faces. The cells' corners coincide with grid points. After computation of spatial derivatives of flux vectors, the differential form of the steady, 3-D Euler equations given in Equation (1) can be discretized for an arbitrary hexahedral control volume.

Spatial derivatives of flux vectors can be computed from the flux balance across a cell

$$\delta_\xi \hat{F} = \hat{F}_{i+1/2,j,k} - \hat{F}_{i-1/2,j,k}$$
$$\delta_\eta \hat{G} = \hat{G}_{i,j+1/2,k} - \hat{G}_{i,j-1/2,k}$$
$$\delta_\zeta \hat{H} = \hat{H}_{i,j,k+1/2} - \hat{H}_{i,j,k-1/2} \tag{12}$$

Then, discretized steady 3-D Euler equation becomes:

$$\frac{\delta \hat{F}}{\Delta \xi}+\frac{\delta \hat{G}}{\Delta \eta}+\frac{\delta \hat{G}}{\Delta \zeta}=0 \tag{13}$$

For a cell centered finite volume method Equation (13) can be arranged as:

$$(\hat{F}_{i+1/2,j,k}-\hat{F}_{i-1/2,j,k})+(\hat{G}_{i,j+1/2,k}-\hat{G}_{i,j-1/2,k})+(\hat{H}_{i,j,k+1/2}-\hat{H}_{i,j,k-1/2})=0 \tag{14}$$

Also it should be noted that, flow domain is divided into equal spaces:

$$\Delta\xi=\Delta\eta=\Delta\zeta=1$$

In above equations, cell interfaces are denoted as *i±1/2, j±1/2* and *k±1/2*. Flow variables at cell interfaces are evaluated by interpolation from the cell center. With using flow variables at cell interfaces, fluxes are calculated. The interaction between the neighbor cells generates the flux. This flux forms the convective portion of Euler Equations and it shows hyperbolic character.

By spatial discretization, $\hat{F}_{i\pm1/2,j,k},\hat{G}_{i,j\pm1/2,k},\hat{H}_{i,j,k\pm1/2}$ can be written as:

$$\begin{aligned}
\hat{F}_{i\pm1/2,j,k}&=\left[\hat{F}^{+}(\hat{Q}^{L}_{i\pm1/2,j,k})+\hat{F}^{-}(\hat{Q}^{R}_{i\pm1/2,j,k})\right]\\
\hat{G}_{i,j\pm1/2,k}&=\left[\hat{G}^{+}(\hat{Q}^{L}_{i,j\pm1/2,k})+\hat{G}^{-}(\hat{Q}^{R}_{i,j\pm1/2,k})\right]\\
\hat{H}_{i,j,k\pm1/2}&=\left[\hat{H}^{+}(\hat{Q}^{L}_{i,j,k\pm1/2})+\hat{H}^{-}(\hat{Q}^{R}_{i,j,k\pm1/2})\right]
\end{aligned} \tag{15}$$

A simplification can be done by approximating the variables at the cell faces to the variables defined at the closest cell centers. *L* and *R* is defined as left and right.

$$\begin{aligned}
\hat{Q}^{L}_{i+1/2}&=\hat{Q}_{i} \quad , \quad \hat{Q}^{R}_{i+1/2}=\hat{Q}_{i+1}\\
\hat{Q}^{L}_{j+1/2}&=\hat{Q}_{j} \quad , \quad \hat{Q}^{R}_{j+1/2}=\hat{Q}_{j+1}\\
\hat{Q}^{L}_{k+1/2}&=\hat{Q}_{k} \quad , \quad \hat{Q}^{R}_{k+1/2}=\hat{Q}_{k+1}
\end{aligned} \tag{16}$$

If higher order of accuracy is desired to be achieved, MUSCL (Monotonic Upstream-Centered Scheme Conservation Law)[13] scheme interpolation can be used. The flow variables at the cell faces are computed from the flow variables at the centers of the four neighboring cells in case of using MUSCLE.

$$\hat{Q}^{L}_{i+1/2} = \hat{Q}_i + \frac{1}{4}\left\{\phi(r)\left[(1-\kappa)\nabla + (1+\kappa)\Delta\right]\right\}_i$$
$$\hat{Q}^{R}_{i+1/2} = \hat{Q}_{i+1} - \frac{1}{4}\left\{\phi(r)\left[(1+\kappa)\nabla + (1-\kappa)\Delta\right]\right\}_{i+1} \quad (17)$$

where *r* equals to:

$$r_i = \frac{\Delta_i}{\nabla_i}$$

Δ and ∇ are forward and backward operators and defined as:

$$\Delta_i = \hat{Q}_{i+1} - \hat{Q}_i \quad , \quad \nabla_i = \hat{Q}_i - \hat{Q}_{i-1}$$

The parameter $\kappa \in [-1,1]$ specifies the order of differencing. If it is equal to 1, the differencing turns into central differencing. -1 gives second-order fully-upwind differencing and 1/3 means second-order fully-upwind differencing

$\phi(r)$ is named as limiter function. It is used to avoid the solution affected by oscillations and also prevent from spurious solutions that take place where high gradients are observed.

Through the study two kinds of limiter functions are used: with $\kappa = 0$ and $\kappa = 1/3$ respectively. For $\kappa = 0$, Van Albada and for $\kappa = 1/3$ Karen limiter functions are chosen. Then the equations given in Eq. (17) transforms into a form below:

$$\hat{Q}^{L}_{i+1/2} = \hat{Q}_i + \delta^{L}_{i+1/2}$$
$$\hat{Q}^{R}_{i+1/2} = \hat{Q}_{i+1} - \delta^{R}_{i+1/2} \quad (18)$$

For $\kappa = 0$:

$$\delta = \frac{(a^2+\varepsilon)b_i + (b^2+\varepsilon)a}{a^2+b^2+2\varepsilon}$$

For $\kappa = 1/3$

$$\delta = \frac{(2a^2+\varepsilon)b + (b^2+2\varepsilon)a}{a^2+b^2-ab+3\varepsilon}$$

where

$$a_L = \Delta_i \quad , \quad b_L = \nabla_i$$
$$a_R = \nabla_{i+1} \quad , \quad b_R = \Delta_{i+1}$$

As it is mentioned above, the reason of using limiter function is to prevent the solution from being affected from oscillations. Hence, in order to deactivate the function in smooth regions, a small number ε is added to the formulation of δ. It is defined as 0,0008 in both Koren's and Van Albada limiters.

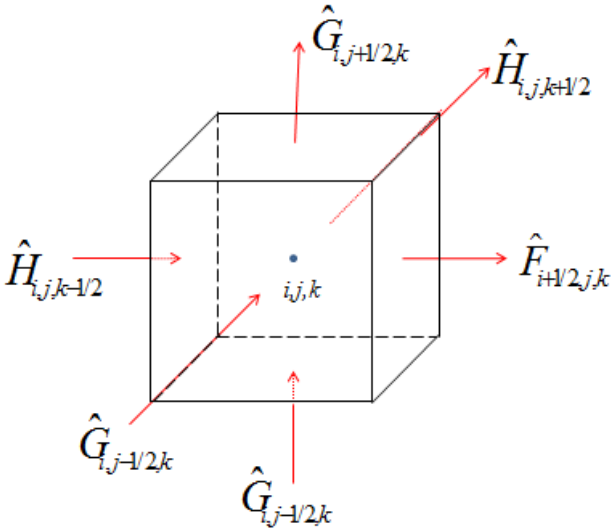

Figure 2.1 Control Volume Representation

2.3 Flux Splitting

It is seen that, after spatial discretization $\hat{F}_{i\pm1/2,j,k}, \hat{G}_{i,j\pm1/2,k}, \hat{H}_{i,j,k\pm1/2}$ are formed in terms of $\hat{F}^+, \hat{F}^-, \hat{G}^+, \hat{G}^-, \hat{H}^+, \hat{H}^-$. These functions may be calculated by using different flux splitting techniques as Steger Warming, Van Leer , Roe or AUSM. Steger Warming and Van Leer [14] are belong to family of flux vector splitting and the others are known as flux difference splitting techniques.

The flux vector can be computed either by central differencing method or upwind schemes. The first way depends on averaging the flow variables at the cell interfaces.

The advantage of using upwind schemes is, it does not need artificial dissipation which is used in central differencing technique.

In this study, Van Leer method is applied. In Van Leer scheme, flux vector is splitted with respect to the contravariant Mach number, *M*. Splitted fluxes are examined with respect to Mach number, *M*, at first as below;

$$Supersonic\,Case: |M| \geq 1$$

$$F^{+} = \begin{cases} F, & M \geq 1 \\ 0, & M \leq 1 \end{cases}$$

$$F^{-} = \begin{cases} 0, & M \geq 1 \\ F, & M \leq 1 \end{cases}$$

$$Subsonic\,Case\ |M| < 1$$

$$F^{\pm} = \rho c \frac{(M \pm 1)^2}{4} (\tilde{k}_1 + \tilde{k}_2 + \tilde{k}_3) \begin{bmatrix} 1 \\ \left(\frac{-\tilde{U} \pm 2c}{\gamma} \right) \tilde{k}_1 + u \\ \left(\frac{-\tilde{U} \pm 2c}{\gamma} \right) \tilde{k}_2 + v \\ \left(\frac{-\tilde{U} \pm 2c}{\gamma} \right) \tilde{k}_3 + w \\ \left(\frac{-\tilde{U} \pm 2c}{\gamma + 1} \right) \tilde{U} + \frac{2a^2}{\gamma^2 - 1} + \frac{u^2 + v^2 + w^2}{2} \end{bmatrix} \qquad (19)$$

where

$$\tilde{U} = \frac{u\xi_x + v\xi_y + w\xi_z}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}, \quad \tilde{k}_1 = \frac{\xi_x}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}, \quad \tilde{k}_2 = \frac{\xi_y}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}, \quad \tilde{k}_3 = \frac{\xi_z}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}$$

c is the speed of sound and γ is the specific heats ratio. The flux vector in η direction, $\hat{G}^{\pm}$, is formed by replacing ξ_x, ξ_y and ξ_z with η_x, η_y and η_z respectively. Also same action is carried on in order to obtain $\hat{H}^{\pm}$.

2.4 Boundary Conditions

Boundary conditions used are inlet boundary conditions, outlet boundary conditions wall boundary conditions and symmetry boundary conditions. Quarter of the nozzle is analysed. The boundary condition that is valid on the selected portion of the geometry are shown on the Figure 2.2

The implementation of boundary conditions to the flow is done by ghost cells. Specified boundary conditions are loaded into these ghost cells. They are added to the exterior of the flow.

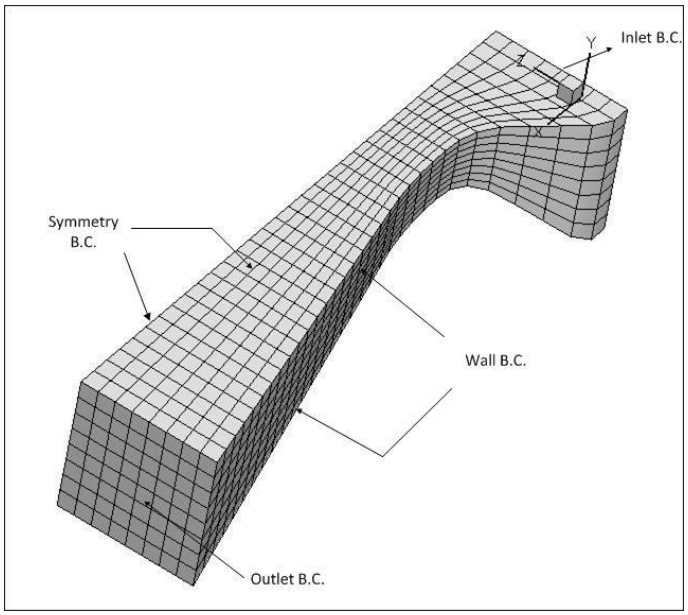

Figure 2.2 Boundary Conditions

2.4.1 Inlet Boundary Conditions

The information that propagates from the inlet, changes direction up to being supersonic or subsonic.

In case of being supersonic, all of the parameters are calculated from the information coming from outside. Hence it is the simplest case. Ghost cells are composed from the variables set from free stream.

In subsonic case, a part of the information proceeds from inside to outside. The airflow at inlet is subsonic in the nozzle analysed. Pressure values are used from inner cells of the nozzle inlet. Because of being 3-D, 5 different parameters are needed. The other parameters which are total pressure, total temperature, Mach number and density are calculated from information propagating from outside. Pressure is calculated by Equation (3) given before. Note that, at inlet, there is a flow in only one direction, $v = 0, w = 0$. ρu is specified from the mass flow rate through the nozzle.

$$p_{boundary} = (\gamma - 1)\rho_{\text{int}erior}(e_t - \frac{u^2 + v^2 + w^2}{2})_{\text{int}erior}$$

$$\rho_{boundary} = \frac{1}{2}(\gamma p_{boundary} + \sqrt{(\gamma p_{boundary})^2 + 2(\gamma - 1)(\rho u)^2}$$

$$(\rho u)_{boundary} = \rho u$$

$$(\rho v)_{boundary} = 0$$

$$(\rho w)_{wall} = 0$$

$$(\rho e_t)_{boundary} = \frac{p_{boundary}}{\gamma - 1} + \frac{1}{2}\frac{(\rho u)^2}{\rho}$$

2.4.2 Outlet Boundary Conditions

If the downstream velocity is supersonic, all of the parameters are taken from interior cells. In subsonic case, one of the parameters is taken from upstream. As the geometry used is a supersonic nozzle, supersonic case is valid for our case.

2.4.3 Wall Boundary Conditions

Density, tangential velocity and total energy are taken from the interior cells. Since there is a wall, mass flux is equal to zero on wall boundary. Hence, normal component of the velocity is zero. In order to preserve it zero, a symmetry plane is formed and an artificial velocity which has the same magnitude but on opposite direction is put on the wall boundary.

$$U_n = U_{interior} \cdot n$$
$$\rho_{boundary} = \rho_{interior}$$
$$(\rho u)_{boundary} = (\rho u)_{interior} - 2\rho_{interior} U_n n_x$$
$$(\rho v)_{boundary} = (\rho v)_{interior} - 2\rho_{interior} U_n n_y$$
$$(\rho w)_{boundary} = (\rho w)_{interior} - 2\rho_{interior} U_n n_z$$
$$(\rho e_t)_{boundary} = (\rho e_t)_{interior}$$

In the equation above, $U_{interior}$ is the velocity defined at the center of the cell at the neighbor of wall. n represents the unit normal at the wall surface. It is composed of n_x, n_y, n_z. u_n, v_n, w_n and $u_{interior}, v_{interior}, w_{interior}$ are the components of normal velocity and interior velocity respectively

2.4.4 Symmetry Boundary Conditions

In nozzle geometry, it is very effective to analyse the flow for the quarter of the nozzle and extrapolate the results to the full geometry by using symmetry planes. Since the magnitudes calculated are the same for the other ¼ parts of the geometry, only manipulating the directions are enough to have all information in each cell of the nozzle.

All of the variables are same on each quarter of the nozzle except normal velocity components to the symmetry planes. They are taken symmetric with respect to the symmetry line.

CHAPTER 3

SOLUTION ALGORITHM

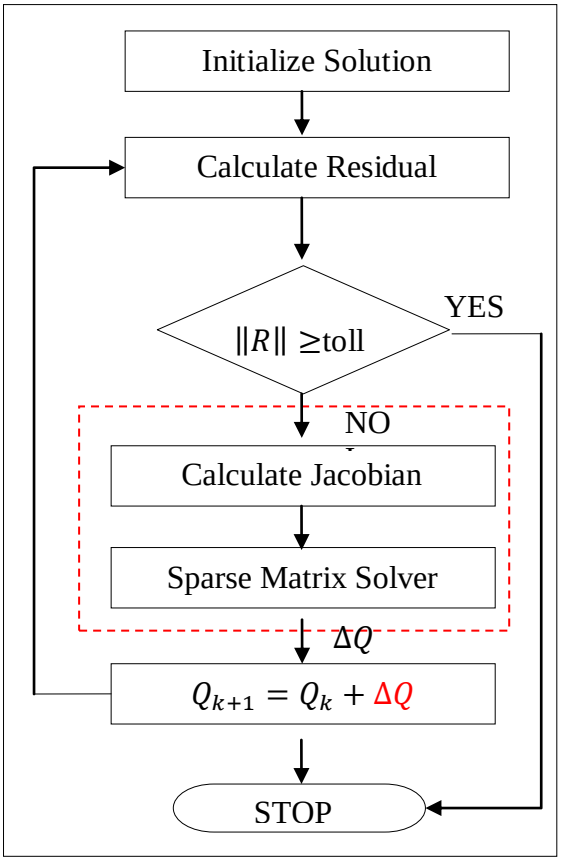

Figure 3.1 Solution Algorithm of Newton's Method

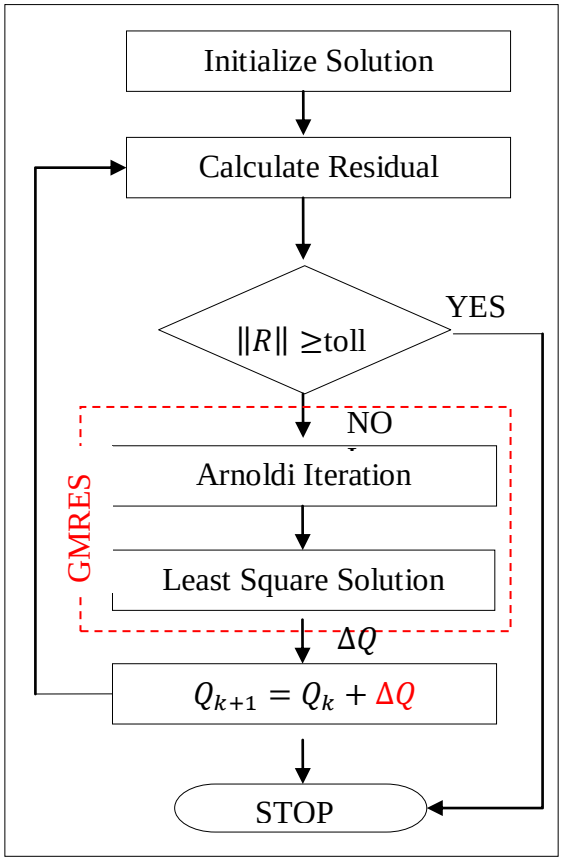

Figure 3.2 Solution Algorithm of Newton-GMRES Method

Two different solution algorithm is used separately. First of the approaches is Newton's Method. 3-D Euler equations are solved with Newton's method. Jacobian calculations are done analytically. Then Newton-GMRES method is applied to flow. Newton's algorithm is modified with adding GMRES linear solver. Details on

GMRES method are also be given in Section 3.2. The diagrams of solution algorithms are given in Figure 3.1 and Figure 3.2 above.

3.1 Newton's Method

As the 3-D Euler equations discretized above, now they can be written in a form of:

$$\hat{R}(\hat{Q}) = 0 \tag{20}$$

$\hat{R}$ denotes the residual vector and it can be written as

$$\hat{R}(\hat{Q}) = \frac{\partial \hat{F}(\hat{Q})}{\partial \xi} + \frac{\partial \hat{G}(\hat{Q})}{\partial \eta} + \frac{\partial \hat{H}(\hat{Q})}{\partial \zeta} \tag{21}$$

$\hat{R}\left(\hat{Q}\right)$ which is a system of nonlinear equations can be linearized by Taylor expansion about $(k)^{th}$ iteration Note that higher order terms are neglected.

$$\hat{R}(\hat{Q}_{k+1}) = \hat{R}(\hat{Q}_k) + \left(\frac{\partial \hat{R}}{\partial \hat{Q}}\right)_k \Delta \hat{Q}_k \tag{22}$$

Here, $\frac{\partial \hat{R}}{\partial \hat{Q}}$ is the Jacobian matrix. Equating the residual to 0 in $(k+1)^{th}$ iteration gives:

$$\left(\frac{\partial \hat{R}}{\partial \hat{Q}}\right)_k \Delta \hat{Q}_k = -R(\hat{Q}_k) \tag{23}$$

The flow variable vector $\hat{Q}$ is updated at the $(k+1)^{th}$ iteration as follows:

$$\hat{Q}_{k+1} = \hat{Q}_k + \Delta \hat{Q}_k \tag{24}$$

As it is shown above, in the solution of Euler equations with Newton's method, Jacobian matrix must be computed. The entries of Jacobian matrix are the derivatives of the residual vector with respect to the flow variables vector. In the calculation of these derivatives a finite difference method or analytical derivation method can be used, and the resulting matrices are called numerical or analytical Jacobians, respectively. In this study analytical derivation method is used. Its advantages and analytical derivation of Jacobian matrix can be found in the next section.

3.1.1 Jacobian Matrix

There are two different techniques in order to compute the Jacobian matrix; analytical or numerical. Numerical method is an easier application which based on finite differencing the flux vector. It is simple and can be performed no matter how complex is the scheme. However, numerical calculation has accuracy problems. Analytical calculation is better on accuracy but it requires differentiation which consumes more time and effort [15]. Also recalculation for each flux discretization scheme, makes the process much more complex. In this study, Jacobian matrix will be calculated analytically because of its accuracy advantage.

Analytical Calculation of Jacobian Matrix

The discretized residual vector can be formed by plugging Equation 15 into Equation (14):

$$\begin{aligned}\hat{R}_{i,j,k} = &\left[\hat{F}^{+}_{i+\frac{1}{2},j,k}(\hat{Q}^{L}) + \hat{F}^{-}_{i+\frac{1}{2},j,k}(\hat{Q}^{R})\right] - \left[\hat{F}^{+}_{i-\frac{1}{2},j,k}(\hat{Q}^{L}) + \hat{F}^{-}_{i-\frac{1}{2},j,k}(\hat{Q}^{R})\right] \\ &+ \left[\hat{G}^{+}_{i,j+\frac{1}{2},k}(\hat{Q}^{L}) + \hat{G}^{-}_{i,j+\frac{1}{2},k}(\hat{Q}^{R})\right] - \left[\hat{G}^{+}_{i,j-\frac{1}{2},k}(\hat{Q}^{L}) + \hat{G}^{-}_{i,j-\frac{1}{2},k}(\hat{Q}^{R})\right] \\ &+ \left[\hat{H}^{+}_{i,j,k+\frac{1}{2}}(\hat{Q}^{L}) + \hat{H}^{-}_{i,j,k+\frac{1}{2}}(\hat{Q}^{R})\right] - \left[\hat{H}^{+}_{i,j,k-\frac{1}{2}}(\hat{Q}^{L}) + \hat{H}^{-}_{i,j,k-\frac{1}{2}}(\hat{Q}^{R})\right]\end{aligned} \tag{25}$$

The Jacobian matrix composes of the derivatives of the residual vector at each cell with respect to the flow variables. Taking the derivatives of residual $\hat{R}_{i,j,k}$ with respect to a flow variable $\hat{Q}_{i,j,k}$, gives the residual Jacobian:

$$
\begin{aligned}
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k}} = {} & \hat{A}^{+}_{i+1/2,j,k}\frac{\partial \hat{Q}^{L}_{i+1/2,j,k}}{\partial \hat{Q}_{i,j,k}} + \hat{A}^{-}_{i+1/2,j,k}\frac{\partial \hat{Q}^{R}_{i+1/2,j,k}}{\partial \hat{Q}_{i,j,k}} \\
& - \hat{A}^{+}_{i-1/2,j,k}\frac{\partial \hat{Q}^{L}_{i-1/2,j,k}}{\partial \hat{Q}_{i,j,k}} - \hat{A}^{-}_{i-1/2,j,k}\frac{\partial \hat{Q}^{R}_{i-1/2,j,k}}{\partial \hat{Q}_{i,j,k}} \\
& + \hat{B}^{+}_{i,j+1/2,k}\frac{\partial \hat{Q}^{L}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j,k}} + \hat{B}^{-}_{i,j+1/2,k}\frac{\partial \hat{Q}^{R}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j,k}} \\
& - \hat{B}^{+}_{i,j-1/2,k}\frac{\partial \hat{Q}^{L}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j,k}} - \hat{B}^{-}_{i,j-1/2,k}\frac{\partial \hat{Q}^{R}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j,k}} \\
& + \hat{C}^{+}_{i,j,k+1/2}\frac{\partial \hat{Q}^{L}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k}} + \hat{C}^{-}_{i,j,k+1/2}\frac{\partial \hat{Q}^{R}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k}} \\
& - \hat{C}^{+}_{i,j,k-1/2}\frac{\partial \hat{Q}^{L}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k}} - \hat{C}^{-}_{i,j,k-1/2}\frac{\partial \hat{Q}^{R}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k}}
\end{aligned}
\tag{26}
$$

where

$$
\hat{A}^{+} = \frac{\partial \hat{F}^{+}}{\partial \hat{Q}^{L}}, \quad \hat{A}^{-} = \frac{\partial \hat{F}^{-}}{\partial \hat{Q}^{R}}
$$

$$
\hat{B}^{+} = \frac{\partial \hat{G}^{+}}{\partial \hat{Q}^{L}}, \quad \hat{B}^{-} = \frac{\partial \hat{G}^{-}}{\partial \hat{Q}^{R}}
$$

$$
\hat{C}^{+} = \frac{\partial \hat{H}^{+}}{\partial \hat{Q}^{L}}, \quad \hat{C}^{-} = \frac{\partial \hat{H}^{-}}{\partial \hat{Q}^{R}}
$$

Analytical derivation of Residual Jacobian needs three sets of derivatives. $\hat{A}^{\pm}_{i+1/2,j,k}$, $\hat{A}^{\pm}_{i-1/2,j,k}$, $\hat{B}^{\pm}_{i,j+1/2,k}$, $\hat{B}^{\pm}_{i,j-1/2,k}$, $\hat{C}^{\pm}_{i,j,k+1/2}$, $\hat{C}^{\pm}_{i,j,k-1/2}$ denote the derivatives of splitted fluxes with respect to flow variables interpolated at the cell faces. To compute these derivatives, Steger-Warming, Van Leer or AUSM flux schemes can be differentiated.

The others are the derivates of flow variables interpolated at cell faces with respect to the flow variables at cell centers. They can be computed with either first or second order discretization.

A first order of accuracy can be achieved by using first order discretization which leads to equate the flow variables at right ($\hat{Q}^R$) to the values at the cell center of the cell that is at the right cell face and ($\hat{Q}^L$) to the variables defined at the face of the cell that is at left. .

$$
\begin{aligned}
&\hat{Q}^L_{i+1/2,\,j,k} = \hat{Q}_{i,j,k} \quad , \quad \hat{Q}^R_{i+1/2,\,j,k} = \hat{Q}_{i+1,j,k} \\
&\hat{Q}^L_{i-1/2,\,j,k} = \hat{Q}_{i-1,j,k} \quad , \quad \hat{Q}^R_{i-1/2,\,j,k} = \hat{Q}_{i,j,k} \\
&\hat{Q}^L_{i,\,j+1/2,k} = \hat{Q}_{i,j,k} \quad , \quad \hat{Q}^R_{i,\,j+1/2,k} = \hat{Q}_{i,j+1,k} \\
&\hat{Q}^L_{i,\,j-1/2,k} = \hat{Q}_{i,j-1,k} \quad , \quad \hat{Q}^R_{i,\,j-1/2,k} = \hat{Q}_{i,j,k} \\
&\hat{Q}^L_{i,\,j,k+1/2} = \hat{Q}_{i,j,k} \quad , \quad \hat{Q}^R_{i,\,j,k+1/2} = \hat{Q}_{i,j,k+1} \\
&\hat{Q}^L_{i,\,j,k-1/2} = \hat{Q}_{i,j,k-1} \quad , \quad \hat{Q}^R_{i,\,j,k-1/2} = \hat{Q}_{i,j,k}
\end{aligned}
\tag{27}
$$

In first order discretization, Jacobian, $\dfrac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k}}$, is computed by:

$$
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k}} = \hat{A}^+_{i+1/2,j,k} - \hat{A}^-_{i-1/2,j,k} + \hat{B}^+_{i,j+1/2,k} - \hat{B}^-_{i,j-1/2,k} + \hat{C}^+_{i,j,k+1/2} - \hat{C}^-_{i,j,k-1/2} \tag{28}
$$

$$
\begin{aligned}
&\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i+1,j,k}} = \hat{A}^-_{i+1/2,j,k} \quad , \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j+1,k}} = \hat{B}^-_{i,j+1/2,k} \quad , \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k+1}} = \hat{C}^-_{i,j+1/2,k} \\
&\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i-1,j,k}} = -\hat{A}^+_{i-1/2,j,k} \quad , \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j-1,k}} = -\hat{B}^+_{i,j-1/2,k}, \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j-1,k}} = -\hat{C}^+_{i,j,k-1/2}
\end{aligned}
\tag{29}
$$

where

$$
\begin{aligned}
&\hat{A}^+_{i\mp1/2,j,k} = \frac{\partial \hat{F}^+_{i\mp1/2,j,k}}{\partial \hat{Q}^L_{i\mp1/2,j,k}}, \quad \hat{A}^-_{i\mp1/2,j,k} = \frac{\partial \hat{F}^-_{i\mp1/2,j,k}}{\partial \hat{Q}^R_{i\mp1/2,j,k}} \\
&\hat{B}^+_{i,j\mp1/2,k} = \frac{\partial \hat{G}^+_{i,j\mp1/2,k}}{\partial \hat{Q}^L_{i,j\mp1/2,k}}, \quad \hat{B}^-_{i,j\mp1/2,k} = \frac{\partial \hat{G}^-_{i,j\mp1/2,k}}{\partial \hat{Q}^R_{i,j\mp1/2,k}} \\
&\hat{C}^+_{i,j,k\mp1/2} = \frac{\partial \hat{H}^+_{i,j,k\mp1/2}}{\partial \hat{Q}^L_{i,j,k\mp1/2}}, \quad \hat{C}^-_{i,j,k\mp1/2} = \frac{\partial \hat{H}^-_{i,j,k\mp1/2}}{\partial \hat{Q}^R_{i,j,k\mp1/2}}
\end{aligned}
$$

In second order discretization, the flow variables at the cell faces are evaluated by using 4 neighboring cell. Flow variables at the centers of neighboring cells are interpolated by MUSCL method to find the variables at the cell face.Then the interpolated flow variables in one direction are :

$$\begin{aligned}
\hat{Q}^{L/R}_{i-\frac{1}{2},\,j,k} &= \hat{Q}^{L/R}_{i-\frac{1}{2},\,j,k}\left(\hat{Q}_{i-1,j,k}, \hat{Q}_{i,j,k}, \delta^{L/R}_{i-\frac{1}{2},\,j,k}\right) \\
\hat{Q}^{L/R}_{i+\frac{1}{2},\,j,k} &= \hat{Q}^{L/R}_{i+\frac{1}{2},\,j,k}\left(\hat{Q}_{i,j,k}, \hat{Q}_{i+1,j,k}, \delta^{L/R}_{i+\frac{1}{2},\,j,k}\right) \\
\delta^{L/R}_{i-\frac{1}{2},\,j,k} &= \delta^{L/R}_{i-\frac{1}{2},\,j,k}\left(\hat{Q}_{i-2,j,k}, \hat{Q}_{i-1,j,k}, \hat{Q}_{i,j,k}, \hat{Q}_{i+1,j,k}\right) \\
\delta^{L/R}_{i+\frac{1}{2},\,j,k} &= \delta^{L/R}_{i+\frac{1}{2},\,j,k}\left(\hat{Q}_{i-1,j,k}, \hat{Q}_{i,j,k}, \hat{Q}_{i+1,j,k}, \hat{Q}_{i+2,j,k}\right)
\end{aligned} \tag{30}$$

The variables in other directions can be derived similarly.

The Jacobian matrices are :

$$\begin{aligned}
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k}} &= \hat{A}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i+1/2,j,k}}{\partial \hat{Q}_{i,j,k}} + \hat{A}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i+1/2,j,k}}{\partial \hat{Q}_{i,j,k}} - \hat{A}^{+}_{i-1,j,k}\frac{\partial \hat{Q}^{L}_{i-1/2,j,k}}{\partial \hat{Q}_{i,j,k}} - \hat{A}^{-}_{i-1,j,k}\frac{\partial \hat{Q}^{R}_{i-1/2,j,k}}{\partial \hat{Q}_{i,j,k}} \\
&+ \hat{B}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j,k}} + \hat{B}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j,k}} - \hat{B}^{+}_{i,j-1,k}\frac{\partial \hat{Q}^{L}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j,k}} - \hat{B}^{-}_{i,j-1,k}\frac{\partial \hat{Q}^{R}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j,k}} \\
&+ \hat{C}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k}} + \hat{C}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k}} - \hat{C}^{+}_{i,j,k-1}\frac{\partial \hat{Q}^{L}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k}} - \hat{C}^{-}_{i,j,k-1}\frac{\partial \hat{Q}^{R}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k}}
\end{aligned} \tag{31}$$

$$\begin{aligned}
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i\mp1,j,k}} &= \hat{A}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i+1/2,j,k}}{\partial \hat{Q}_{i\mp1,j,k}} + \hat{A}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i+1/2,j,k}}{\partial \hat{Q}_{i\mp1,j,k}} - \hat{A}^{+}_{i-1,j,k}\frac{\partial \hat{Q}^{L}_{i-1/2,j,k}}{\partial \hat{Q}_{i\mp1,j,k}} - \hat{A}^{-}_{i-1,j,k}\frac{\partial \hat{Q}^{R}_{i-1/2,j,k}}{\partial \hat{Q}_{i\mp1,j,k}} \\
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j\mp1,k}} &= \hat{B}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j\mp1,k}} + \hat{B}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j\mp1,k}} - \hat{B}^{+}_{i,j-1,k}\frac{\partial \hat{Q}^{L}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j\mp1,k}} - \hat{B}^{-}_{i,j-1,k}\frac{\partial \hat{Q}^{R}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j\mp1,k}} \\
\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k\mp1}} &= \hat{C}^{+}_{i,j,k}\frac{\partial \hat{Q}^{L}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k\mp1}} + \hat{C}^{-}_{i,j,k}\frac{\partial \hat{Q}^{R}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k\mp1}} - \hat{C}^{+}_{i,j,k-1}\frac{\partial \hat{Q}^{L}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k\mp1}} - \hat{C}^{-}_{i,j,k-1}\frac{\partial \hat{Q}^{R}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k\mp1}}
\end{aligned} \tag{32}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i+2,j,k}} = \hat{A}^{+}_{i+1/2,j,k} \frac{\partial \hat{Q}^{L}_{i+1/2,j,k}}{\partial \hat{Q}_{i+2,j,k}} + \hat{A}^{-}_{i+1/2,j,k} \frac{\partial \hat{Q}^{R}_{i+1/2,j,k}}{\partial \hat{Q}_{i+2,j,k}}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i-2,j,k}} = -\hat{A}^{+}_{i-1/2,j,k} \frac{\partial \hat{Q}^{L}_{i-1/2,j,k}}{\partial \hat{Q}_{i-2,j,k}} - \hat{A}^{-}_{i-1/2,j,k} \frac{\partial \hat{Q}^{R}_{i-1/2,j,k}}{\partial \hat{Q}_{i-2,j,k}}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j+2,k}} = \hat{A}^{+}_{i,j+1/2,k} \frac{\partial \hat{Q}^{L}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j+2,k}} + \hat{A}^{-}_{i,j+1/2,k} \frac{\partial \hat{Q}^{R}_{i,j+1/2,k}}{\partial \hat{Q}_{i,j+2,k}} \tag{33}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j-2,k}} = -\hat{A}^{+}_{i,j-1/2,k} \frac{\partial \hat{Q}^{L}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j-2,k}} - \hat{A}^{-}_{i,j-1/2,k} \frac{\partial \hat{Q}^{R}_{i,j-1/2,k}}{\partial \hat{Q}_{i,j-2,k}}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k+2}} = \hat{A}^{+}_{i,j,k+1/2} \frac{\partial \hat{Q}^{L}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k+2}} + \hat{A}^{-}_{i,j,k+1/2} \frac{\partial \hat{Q}^{R}_{i,j,k+1/2}}{\partial \hat{Q}_{i,j,k+2}}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k-2}} = -\hat{A}^{+}_{i,j,k-1/2} \frac{\partial \hat{Q}^{L}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k-2}} - \hat{A}^{-}_{i,j,k-1/2} \frac{\partial \hat{Q}^{R}_{i,j,k-1/2}}{\partial \hat{Q}_{i,j,k-2}}$$

3.2 Newton-GMRES Method

In Equations (31), (32) and (33) Jacobian matrices are shown. Their derivation and calculation are given in previous section. The complexity of their calculation is easy to see. To avoid from this forming and computation work load, Newton-GMRES method is studied.

To solve the system of nonlinear equations given in Equation (3), inexact methods were investigated. They can be generalized as finding $\Delta\hat{Q}_k$ such in each iteration step, k.

$$\left\| \hat{R}(\hat{Q}_k) + \hat{R}'(\hat{Q}_k)\Delta Q_k \right\| \le \eta_k \left\| \hat{R}(\hat{Q}_k) \right\| \tag{34}$$

Note that, when $\eta_k = 0$ equation (33) gets the form of

$$\left(\frac{\partial \hat{R}}{\partial \hat{Q}} \right)_k \Delta\hat{Q}_k = -R(\hat{Q}_k) \tag{35}$$

which corresponds to Newton's method.

The importance of selection of η_k is to prevent the solution from oversolving. This problem was investigated by Einsenstat [16] and some procedures were presented. In this work Equation 36 is used in order to skip first unnecessary iterations.

$$\eta_k = \gamma \left(\frac{\left\| R(\hat{Q}_k) \right\|}{\left\| R(\hat{Q}_{k-1}) \right\|} \right)^{\alpha} \tag{36}$$

where $\gamma \in [0,1]$ and $\alpha \in (1,2]$

Pueyo and Zingg [17] found that taking $\eta_k = 0,5$ in first 10 iterations and then selecting $\eta_k = 0,1$ is giving a better efficiency for the cases they analysed. By inspring from Pueyo and Zingg, it is tried to examine if there are much more efficient solutions by combining different η_k values manually. Each of the approaches are applied and most appropriate one is chosen with respect to residuals calculated.

Newton-GMRES method can be defined as an implementation of Newton's method combined with the iterative linear algebra method GMRES in order to be used as an approximate solver for each Newton step. Newton-GMRES is an iterative method which is one of the Newton-Krylov methods which are a kind of inexact Newton Methods.

While using Newton-GMRES method, there is no need to compute Jacobian matrix which is one of the most important feature of this method. It only requires the action of the Jacobian $\hat{R}'$ on a vector v which can be approximated by finite difference and this process leads us to make computation without evaluating a matrix which means the process is *matrix free* [18]

$$R'(\hat{Q})v \approx \frac{R(\hat{Q} + \varepsilon v) - R(\hat{Q})}{\varepsilon} \tag{37}$$

where ε is a scalar which is used to perturb the flow variables, $\hat{Q}$. While applying this method, the selection of ε directly affects the solution. Nielsen [19] showed that

$$\varepsilon \left\| v \right\|_2 \approx \sqrt{\varepsilon_m} \tag{38}$$

where ε_m is machine zero. In the computer used in thesis work, the machine zero is known as 10^{-16} and v values are lower than 1. By trial, it is seen that, selecting ε as 10^{-7} gives best results. It controls the truncational and round off errors. It should be noted that, this selection is so close to Nielsen's formulization and setting it to a number does not require calculation of ε again and again.

Krylov subspace methods are for solution of linear problems $Ax = b$. It starts with an initial x_0 and at each step it determines an iterate x with a correction in the Krylov subspace. Hence, they don't need direct access to the entries of A

$$K_k \equiv span\left\{r_0, Ar_0, ..., A^{n-1}r_0\right\} \tag{39}$$

where r is the initial residual..

The algorithm of application of GMRES to k^{th} Newton equation is given below as outlined in [20]

GMRES Algorithm

1.Select $\Delta\hat{Q}_k^0$ and set m=0

$r_k^0 = -\hat{R}'(\hat{Q}_k)\Delta\hat{Q}_k^0 - \hat{R}(\hat{Q}_k)$

$\beta_k = \left\|r_k^0\right\|,\ v_1 = r_k^0 / \beta_k$

2.Construct Hessenberg matrix by carrying out Arnoldi process [21]

While $\left\|r_k^m\right\| \geq \eta_k \left\|\hat{R}(\hat{Q}_k)\right\|$ do

Set $m = m+1$

Calculate $\hat{R}'(\hat{Q}_k)v_n$ then

$h_{i,m} = (\hat{R}'(\hat{Q}_k)v_m)^T v_i \quad i = 1, 2, ...m$

$v_{m+1} = \hat{R}'(\hat{Q}_k)v_m - \sum_{i=1}^{m} h_{i,m}v_i$

$h_{m+1,m} = \left\|v_{m+1}\right\|$

$v_{m+1} = v_{m+1} / h_{m+1,m}$

3.Find the vector y_m such that :

$$\min \left\| \beta_k e_1 - H_m y_m \right\|_2 \text{ where } e_1 = (1,0,0,0....)^T$$

(Apply Givens type transformation to H_m to compute the Q-R factorization to make easier to solve the least square system)

4.Set

$$\left\| r_k^m \right\| = \left\| \beta_k e_1 - H_m y_m \right\|$$

5.Define

$$V_m \equiv \left[v_1, v_2, v_3 ... v_m \right]$$

$$\Delta \hat{Q}_k^m = \Delta \hat{Q}_k^0 + V_m y_m$$

6.Iterate until residual vector r_k^m satisfies the stopping iteration which is:

$$\left\| r_k^m \right\| \le \eta_k \left\| \hat{R}(\hat{Q}_k) \right\| \text{ or } \Delta \hat{Q}_k = \Delta \hat{Q}_k^m$$

The step size found by GMRES is used to perform Newton iteration:

$$\hat{Q}_{k+1} = \Delta \hat{Q}_k + \hat{Q}_k \tag{40}$$

$\Delta \hat{Q}_k$ is named as descent direction. It is taken as 0 at first iteration.

3.2.1 Arnoldi Iteration

Step 2 can be defined as the heart of GMRES where the Arnoldi iteration takes place. The part given as calculation of v, h starting from $\hat{R}'(\hat{Q}_k) v_n$ is named as Arnoldi algorithm. The aim of the Arnoldi process is to form an orthonormal basis for a Krylov subspace. For a given matrix A and a non-zero vector x and with a dimension defined by m, using Arnoldi iteration one constructs a matrix V such that:

$$col\ span(V) = span(x, Ax, A^2 x, ..., A^{m-1} x) \tag{41}$$

where $V^T V = I$

By applying Arnoldi iteration, instead of analyzing a given large scale matrix A, only examining a small subset of A that includes the rightmost or largest eigenvalue in magnitude is provided. Projecting A into a low dimensional subspace, one may approximate these large eigenvalues into an easier form. Then problem becomes a

small eigenvalue problem which is easy to solve with well known QR factorization techniques explained in linear least square problem solution section.

Arnoldi iterations stops when v_{m+1} cannot be computed. In other words, while $\hat{R}'(\hat{Q}_k)v_m - \sum_{i=1}^{m} h_{i,m}v_i = 0$ Arnoldi algorithm finishes. The procedure ends with forming an upper Hessenberg matrix from a given dense matrix. An upper Hessenberg matrix, H_{ij}, is defined as a matrix whose entries are zero where $j \le i-2$. On Equation (42), an Hessenberg matrix is formed for $5x4$ as an illustration.

$$H = \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ 0 & . & . & . \\ 0 & 0 & . & . \\ 0 & 0 & 0 & . \end{bmatrix} \tag{42}$$

3.2.2 Linear Least Square Problem Solution

After constructing the Hessenberg matrix in step 2, one solves the linear least square system. It is not a must but in order to lower the CPU time elapsed in solution process of linear least square problem, one may still shift the matrix achieved into a lower or upper matrix for simplicity. The procedures that can be applied are given in next section. In this section, the last version of the matrix is taken into account and solution of minimization problem is investigated. Consider the over determined linear system

$Ax = b$ where $A_{mxn}, m \ge n$

Here, x is the solution of the least square that minimizes the Euclidian norm of $r = b - Ax$ where r is the residual vector.

$$\min \|r\|_2^2 = \min \|b - Ax\|_2^2 \tag{43}$$

There are different algorithms that find the minimum which are normal equations, QR factorization and singular value decomposition, SVD. The first method is the cheapest one. However, because of the lack of accuracy, it is not very commonly

used technique. If the system is full rank; ıt is selected to use QR factorization while solving the minimization problem. Also it is very attractive if the system involves Hessenberg matrices which is the main attribute of GMRES method

QR factorization can be formed by different methods such as Gram-Schmit, Householder or Givens Rotation. As upper H_m matrix is close to a upper triangular matrix, it is very efficient to use Givens Rotation as a solution technique. Upper Hessenberg matrices can be classified as sparse systems which mean they include significant number of zeros as entries. While applying Householder procedure temporary nonzero elements arise in intermediate steps, which means need of extra temporary storage. Hence it is recommended to apply Givens rotation in GMRES since the matrix formed in minimization step is composed of an upper Hessenberg matrix.

$$H_m y_m = \beta_k e_1 \tag{44}$$

While applying QR factorization, Hessenberg matrix is written as multiplication of a triangular matrix, R, and an orthonormal matrix, Q.

$H_m = QR$ where $Q^T Q = I$

By substituting above equalities into left hand side of Equation (44) and simplifying, system becomes:

$$\begin{aligned} QRy_m &= \beta_k e_1 \\ Q^T QRy_m &= Q^T \beta_k e_1 \\ Ry_m &= Q^T \beta_k e \end{aligned} \tag{45}$$

3.2.3 Givens Rotation

As a note, it was given that, in order to solve the least square system Givens Rotation can be applied to the Hessenberg matrix that was formed. Givens transformation is a method that composes of series of rotations in order to zero the lower triangular of a given matrix.

The idea behind Given's rotation is to make $y_k = 0$ where

$$y = G^T(i,k,\theta)x$$

Since y_i, y_j, y_k is given as below, $s = \sin\theta,\ c = \cos\theta$ should be computed as in Equation (46)

$$\begin{aligned} y_i &= cx_i - sx_i \\ y_k &= cx_k + sx_k \\ y_j &= x_j \end{aligned}$$

$$\begin{aligned} c &= x_i / t \\ s &= -x_k / t \end{aligned} \quad \text{where } t = \sqrt{x_i^{\,2} + x_k^{\,2}} \tag{46}$$

A Givens rotation matrix can be represented as

$$G(i,k,\theta) = \begin{bmatrix} 1 & 0 & . & . & 0 & . & . & 0 & . & . & 0 \\ 0 & 1 & & & & & & & & & . \\ . & & . & & & & & & & & . \\ . & & & . & & & & & & & 0 \\ 0 & & & & c & & & s & & & . \\ . & & & & -s & & & c & & & . \\ . & & & & & & & & & & 0 \\ 0 & & & & & & & . & & & . \\ . & & & & & & & & . & & . \\ . & & & & & & & & & 1 & 0 \\ 0 & . & . & 0 & . & . & 0 & . & . & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ i \\ \\ \\ \\ \\ \\ k \end{matrix} \tag{47}$$

$s = \sin\theta,\ c = \cos\theta$

3.3 Preconditioning

Preconditioning refers to transform a given linear system into a form that is easy to solve with an iterative solver. This becomes applicable with constructing a preconditioning matrix M. There are different kinds of forming a preconditioned system with using this M matrix. First of all one should find this preconditioning matrix. There are few requirements that M have to satisfy. Preconditioned system solution should be inexpensive and close to original matrix A and must be invertable [22]. The ideal case of M is being equal to A. Consequently, multiplication of M^{-1}

and A gives identity matrix. However, this causes high computational cost and system becomes explicitly solved. Hence an approximation of A matrix is a better choice to compose M matrix. M can be applied from left, right or in a splitted form to system $Ax = b$.

Preconditioning from left : $M^{-1}Ax = M^{-1}b$

Preconditioning from right: $M^{-1}Ax = M^{-1}b$

Splitted preconditioning : $M_L^{-1}AM_R^{-1}x = M_L^{-1}bM_R^{-1}$ (in special cases)

Left preconditioning is an easier process with respect to right preconditioning. That does not need any extra computation in the algorithm except multiplying the residual with M matrix at the beginning of inner loop. On the other hand, in case of using right preconditioner, one should perform a multiplication of M^{-1} with x while updating their values at outer loop.

There are various techniques of preconditioning. Most commons applied to GMRES are sorted as: diagonal preconditioner ILU(0), ILU(2) which are ILU Factorization Preconditioners.

Diagonal preconditioner is also named as Jacobi Preconditioner. One can form it easily. It is composed of the diagonal element of Jacobi matrix. But as long as it is not convenient to construct the exact Jacobian because of its expenses, an approximation of the Jacobi is selected to form a diagonal preconditioner. It does not classified as most effective but in some kinds of problems it works well [23]. Block Jacobi preconditioners also depend on same principle. The only difference between them is to zero in the blocks except at the diagonal instead of zero in the entries except the ones on the diagonal. Block Jacobi is expected to be more efficient than diagonal preconditioning.

It is stated that ILU type preconditioners are more attractive in many problems while application method is still simple [24]. ILU(0) is the cheapest case with zero fill in value. They require sparse form of the original matrices

CHAPTER 4

RESULTS

Chapter 4 is composed of four sections. In the first part, a preliminary study is presented whose results are promotive to work with larger systems. Second section includes the test cases. Comparisons are held on six different test cases. Grid sizes and discretization types are changed in order to observe the act of Newton-GMRES method while the system is getting larger. CPU times and convergences histories of different cases for both Newton and Newton-GMRES method are given. Parameters that affects Newton-GMRES method's convergence are analysed. Also a mesh independence study is given in section 4.4

The code of Newton-GMRES is developed by using Fortran77 by manipulating the code used in [25] . The analysis is conducted on 2.3 GHz, AMD Opteron6227 processor.

4.1 Solution of Three Nonlinear Equations

In order to test the algorithm formed is working well or not, a system composed of three nonlinear equations was constructed.

$$\textit{The system}: \begin{aligned} x^3 - 2y^2 - 2 &= 0 \\ x^3 - 5z^2 + 7 &= 0 \\ yz^2 - 1 &= 0 \end{aligned}$$

$\textit{Initial values}: x = 1, y = 1, z = 1$

Solution by Newton's Method

$x = 1.4422500,\ y = 0.500000,\ z = 1.414214$

Solution by Newton - GMRES Method

$x = 1.4422425,\ y = 0.499999,\ z = 1.414207$

The system is both solved with Newton and Newton-GMRES method. Results were pretty sufficient. Same accuracy was provided with Newton-GMRES method. As long as the system is so small, CPU calculations couldn't been held.

Success that is achieved while solving three nonlinear equations can be seen on convergence history comparison of preliminary study in Figure 4.1 given below

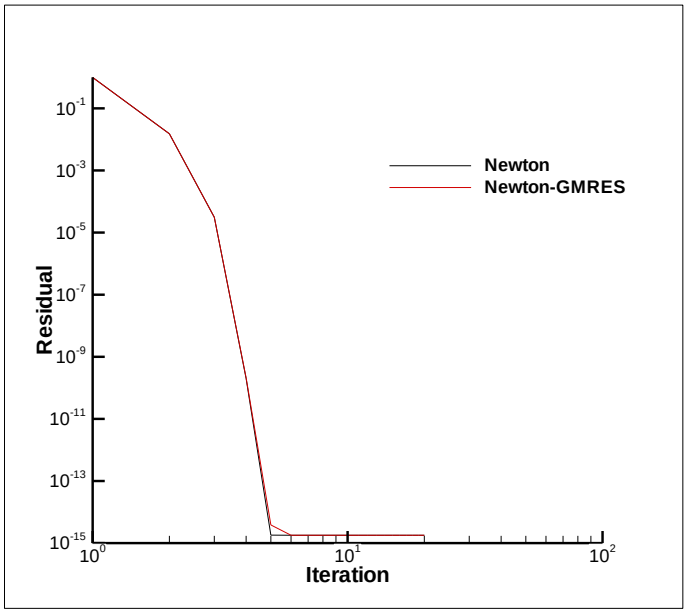

Figure 4.1 Comparison of Newton and Newton-GMRES Methods on Three Equations

4.2 Solution of 3-D Euler Equations in Nozzles

CPU times are calculated with respect to grid sizes and order of construction. They are formed to see the effect of grid size to the convergence and CPU time. Grid sizes were selected as they are tabulated below. They are formed in-house

Table 4.1 Sizes of the Grids Generated

Coarse Grid	17x5x5
Medium Grid	33x9x9
Fine Grid	65x17x17

As it was mentioned in flow analysis, spatial discretization is an important process on improving the accuracy. If higher order discretization is selected CPU time is getting higher in return of more accurate solution.

Table 4.2 Test Cases

Test Case	**Grid Size**	**Order**
C1	17x5x5	1
C2	17x5x5	2
M1	33x9x9	1
M2	33x9x9	2
F1	65x17x17	1
F2	65x17x17	2

Test cases are tabulated above with respect to order of discretization and grid sizes. Abbreviations C, M and F denote coarse, medium and fine grids. 1 and 2 show the order of discretization.

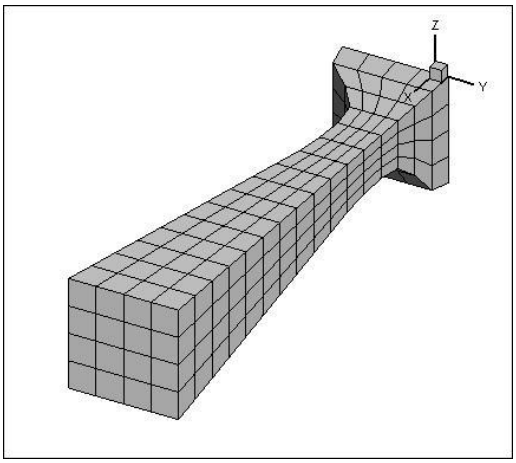

Figure 4.2 Generation of Coarse Grid

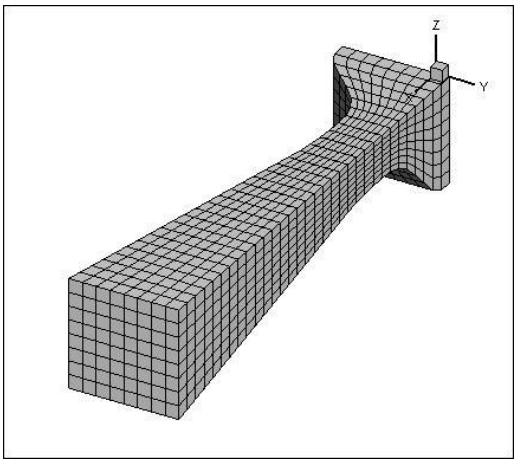

Figure 4.3 Generation of Medium Grid

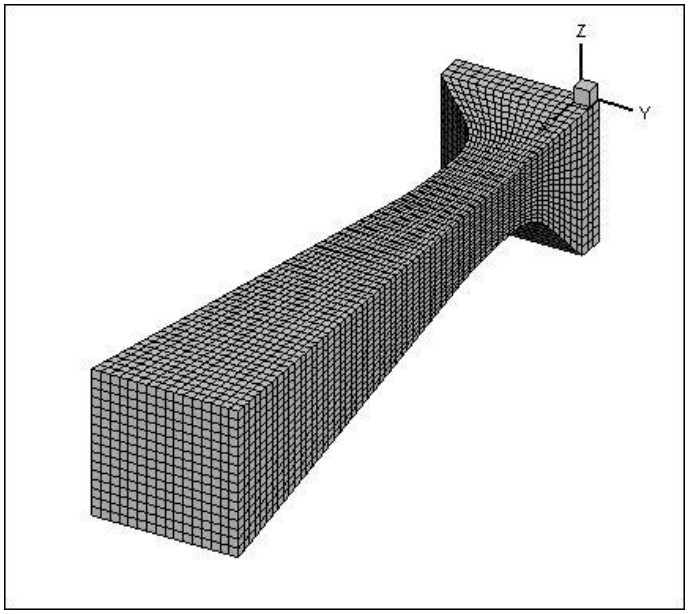

Figure 4.4 Generation of Fine Grid

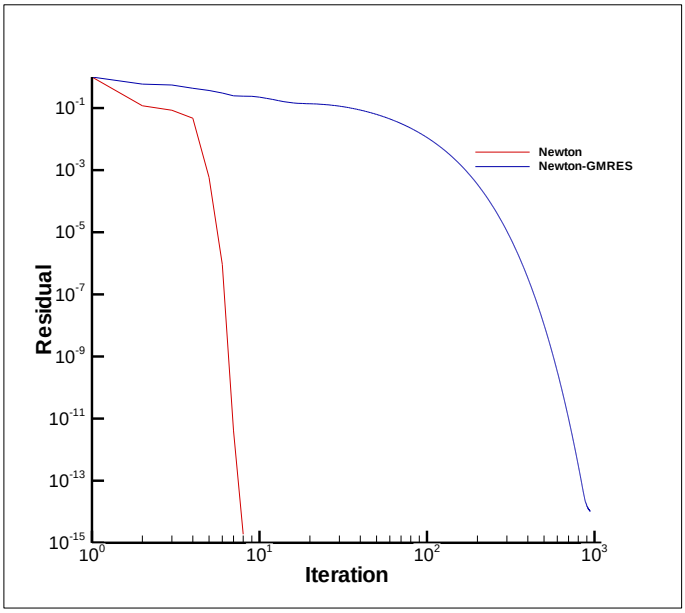

Figure 4.5 Convergence Histories Comparison for C1

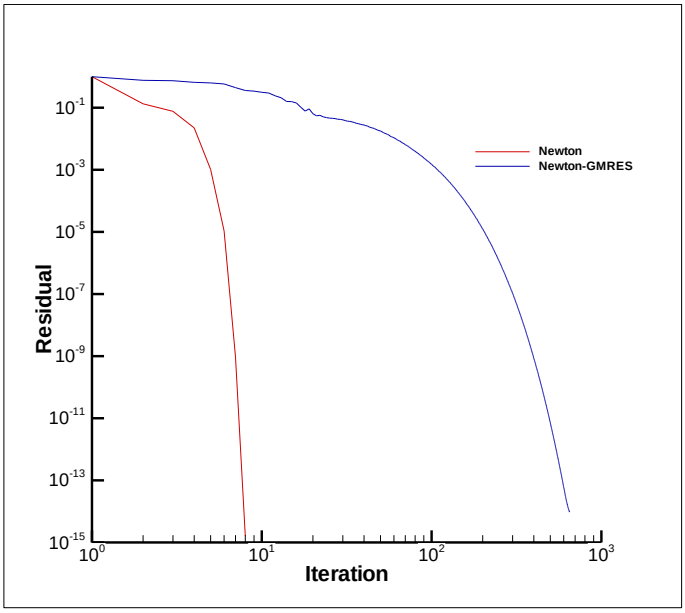

Figure 4.6 Convergence Histories Comparison for C2

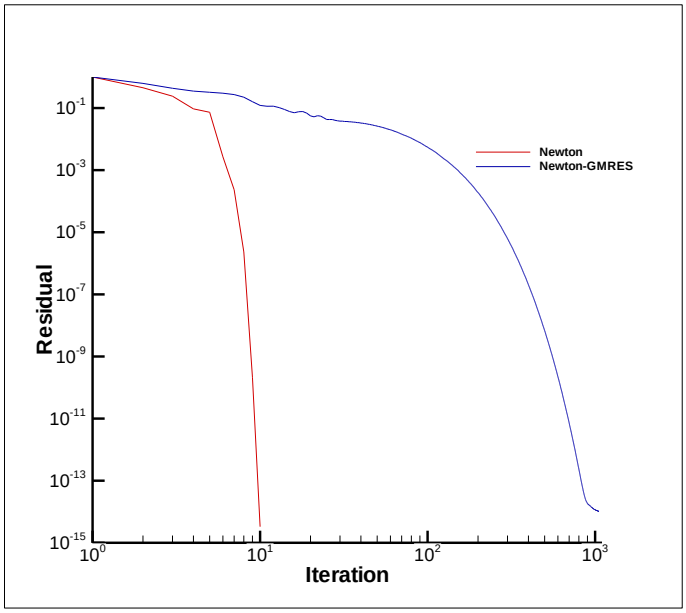

Figure 4.7 Convergence Histories Comparison for M1

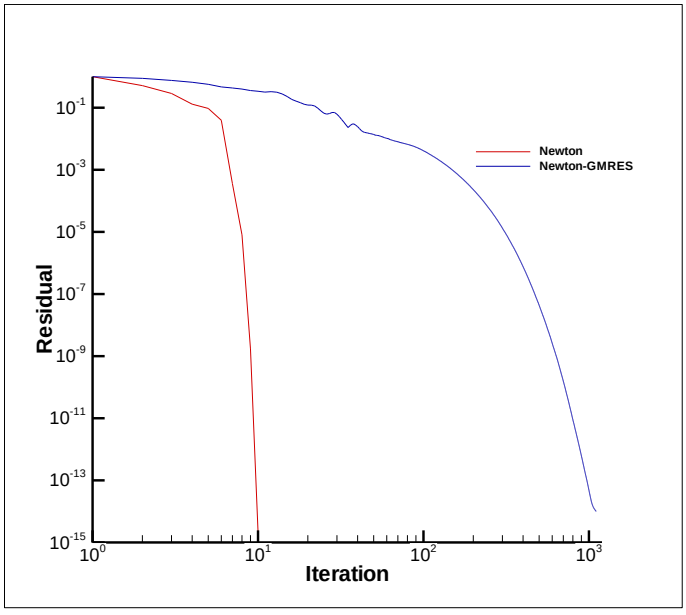

Figure 4.8 Convergence Histories Comparison for M2

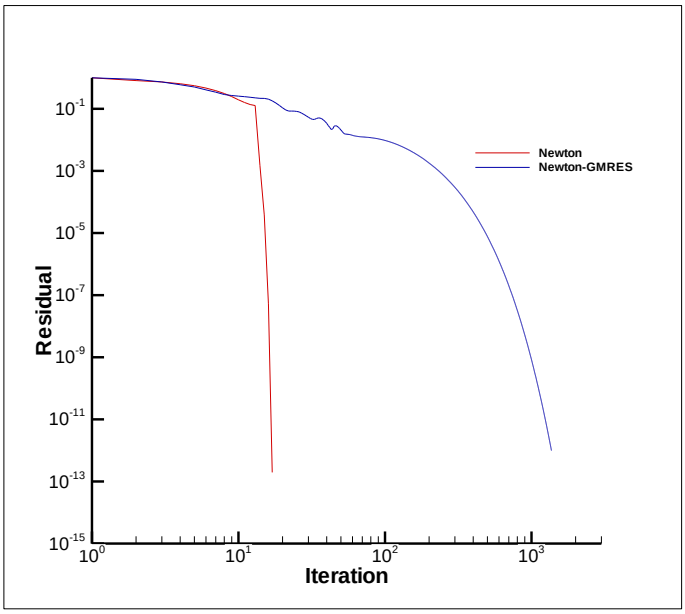

Figure 4.9 Convergence Histories Comparison for F1

Newton-GMRES method is a combination of an inner and outer iteration as it was mentioned before. While second order discretization is applied, it is expected to have longer convergence histories. However it is seen that, the inner loop converges in further iterations. This is not observed in convergence histories figures as they only show the Newton iteration counts. Also it should be noted that, for first and second order discretizations, different η_k values are selected in order to shorten the CPU time elapsed. This also changes the count of iterations.

Implementation of Newton's method into F2 case is not carried out due to UMFPACK performance. UMFPACK gives error message related to insufficient memory and which is faced by other researcehers very common. It is recommended to switch to another matrix solver as PETSc or MUMPS

In convergence histories, Newton method converges around 10^{th} iteration for each case. On the other hand, Newton-GMRES method converges at least around 700^{th} cycle. Although the number of iterations is approximately two order of magnitude smaller in Newton method, the CPU time is less in Newton-GMRES method.

Table 4.3 CPU Time Comparisons

Test Cases		CPU Time Elapsed (sec)
M1	*Newton-GMRES*	15.480
	Newton	80.195
C1	*Newton-GMRES*	1.548
	Newton	2.299
M2	*Newton-GMRES*	20.78
	Newton	242.757
C2	*Newton-GMRES*	1.985
	Newton	3.192
F1	*Newton-GMRES*	255.770
	Newton	3584.438
F2	*Newton-GMRES*	731.201

The difference between the CPU times increases as the mesh size is getting larger. This is because of the reason that Newton method requires the solution of very large matrix updated at each iteration. As far as the CPU time is concerned, the performance of Newton- GMRES method is getting better as the mesh size increases.

Due to the quadratic convergence property of Newton's method, a residual value between 10^{-9} and 10^{-14} can't be selected as stopping criteria for many of the cases. On the other hand, while applying Newton-GMRES method, after 10^{-12} solution slows down.. Also the accuracy of solutions with residual 10^{-14} and 10^{-12} are pretty similar; solving Newton-GMRES with residual of 10^{-14} can be defined as loss of time for this specific purpose of solving the system generated for this study. The CPU time elapsed in F1 with Newton-GMRES is 498.7 sec if residual is set to 10^{-12} where for 10^{-14} 731 sec. Same problem is encountered at each of the test cases.

GMRES method is getting better as the mesh size increases. The lowest iteration was achieved in small grid with higher order discretization. However when CPU time calculations are taken into account, this dramatic difference in iteration numbers lost their importance.

Mach and pressure contours of 3-D Supersonic Nozzle with medium grid and first order of discretization are plotted below for both of the solution methods. Also

velocity vector distributions are presented. Since the residual level reduced to same order of magnitude, the similarity between the contour plots of Newton-GMRES and Newton's Method is shown below.

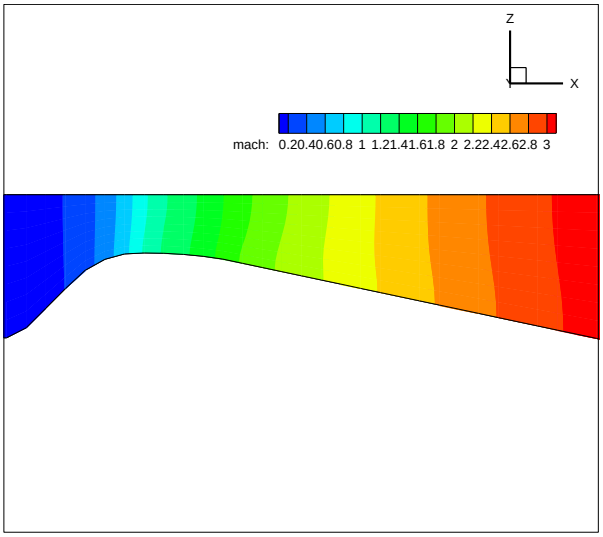

Figure 4.10 Mach Contours Obtained by Newton-GMRES Method

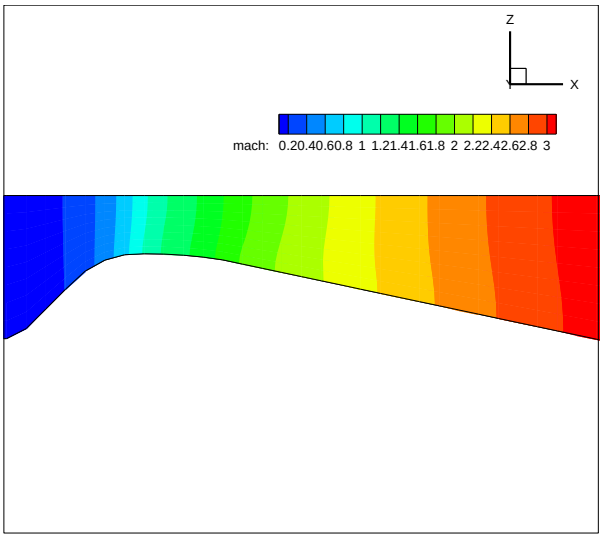

Figure 4.11 Mach Contours Obtained by Newton Method

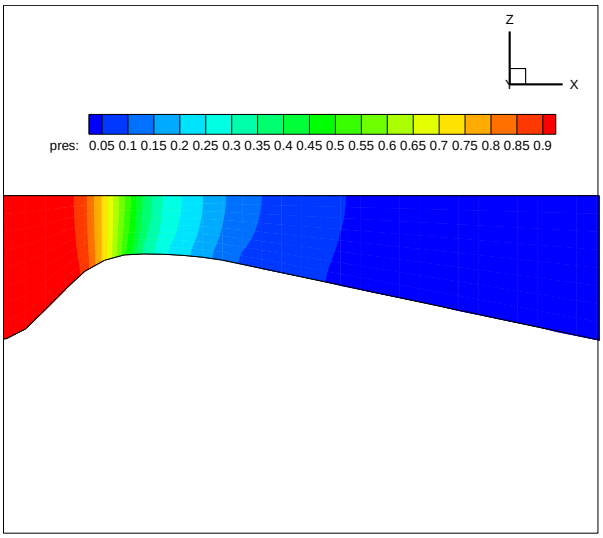

Figure 4.12 Pressure Distribution Obtained by Newton-GMRES Method

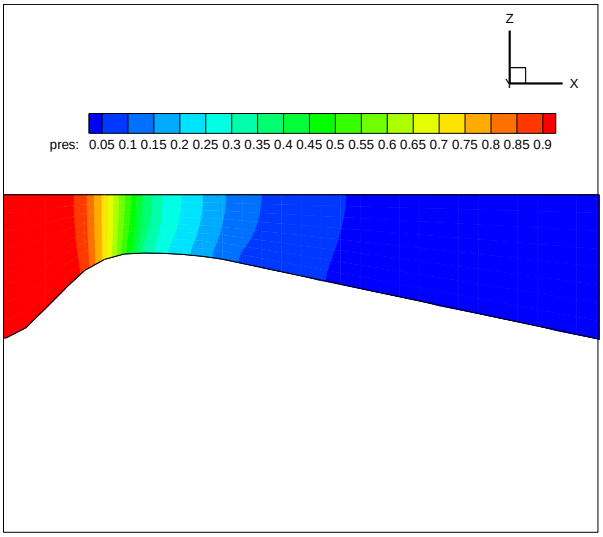

Figure 4.13 Pressure Distribution Obtained by Newton Method

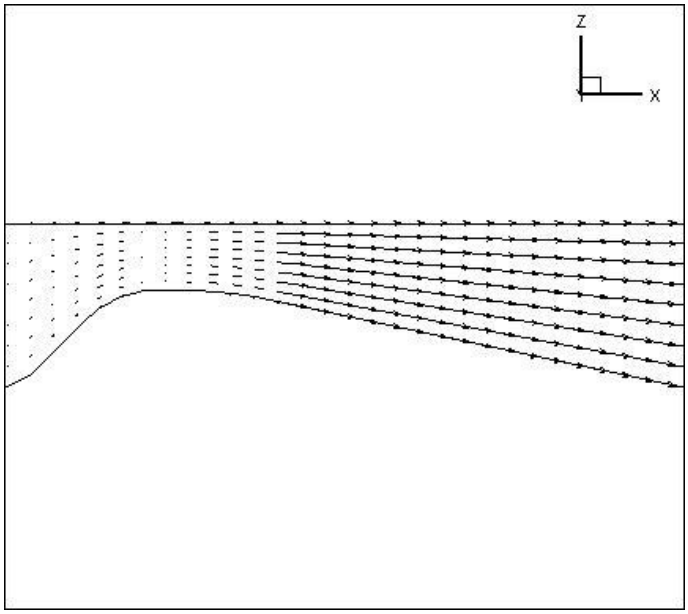

Figure 4.14 Velocity Vectors Obtained By Newton-GMRES Method

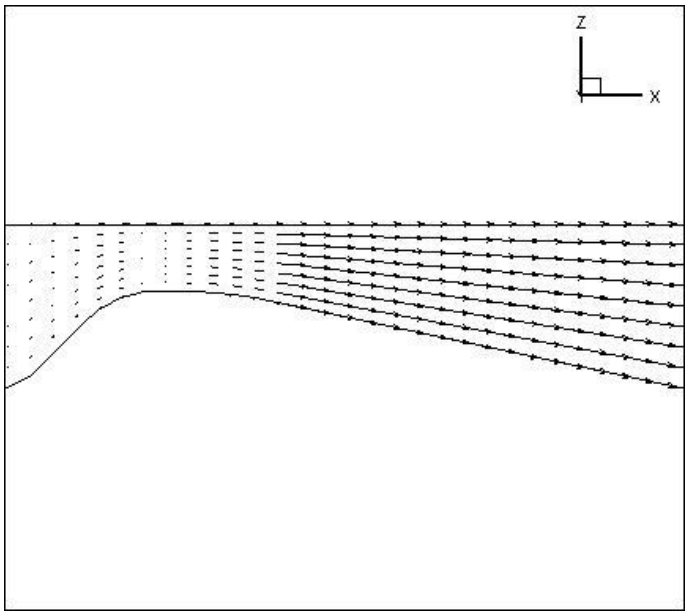

Figure 4.15 Velocity Vectors Obtained By Newton Method

4.3 Parametric Study

At the beginning of Newton-GMRES process, one should identify the parameters $\Delta Q_k^0, \eta_k, \varepsilon$. Computation of ε is given in Chapter 2. It is used to perturb the system and defining it as close to zero as possible is the main purpose of given techniques of calculations. By decreasing the value of ε, residual is also approaches much more smaller magnitudes.

Both of the solution techniques emphasized are locally convergent methods. Local convergence implies that, Newton and Newton-GMRES methods are successful providing good initial approximations. To put it in different words, convergence can only be achieved if the initial selection of flow variables done well.

Apart from initializing the flow variables, in Newton-GMRES method one also defines a step size in the inner loop of the algorithm, ΔQ_k^0. Selection of ΔQ_k^0 also evidently affects the convergence of the process. It is observed that initializing ΔQ_k^0 with 0 accelerates the convergence of the method as a result of this local convergence property.

While conducting the study, it is seen that the solution is sensitive to selection of forcing term, η_k. As they were mentioned above, there are various strategies to choose η_k. With different selections of α, γ Equation (36) is applied to the code in order to find optimum value of this forcing term. Medium grid is chosen to compare the efficiencies provided by different selections of η_k. In Eisenstat's suggestion given in Equation (36), it depends on both residual of the previous iteration and also the updated version in addition to α, γ values. A η_k value is selected arbitrarily for the first iteration.

It is noted that the effect of α is limited with respect to the effect of γ. However, the order of their importance is the same if one goes out of the interval that was

defined for these to variable. System does not converge in situation of selection of arbitrary numbers.

$$\eta_k = 0,5$$

Table 4.4 CPU Time Comparison with Eisenstat's Formulation (sec)

α \ γ	0.1	0.4	0.5	0.6	0.9
1.1	26.465	14.794	12.427	13.561	14.022
1.3	27.907	16.783	15.296	15.273	15.343
1.5	28.245	17.460	15.978	15.363	15.314
1.8	28.1647	19.569	16.595	16.219	15.049
1.9	27.930	18.192	17.004	16.447	14.358

Pueyo and Zingg suggested to use a higher η_k for first few iterations and then lower the value in order to prevent oversolving problems. Idea is implemented and seen that using a fixed number is giving a better CPU time results if the selection made carefully.

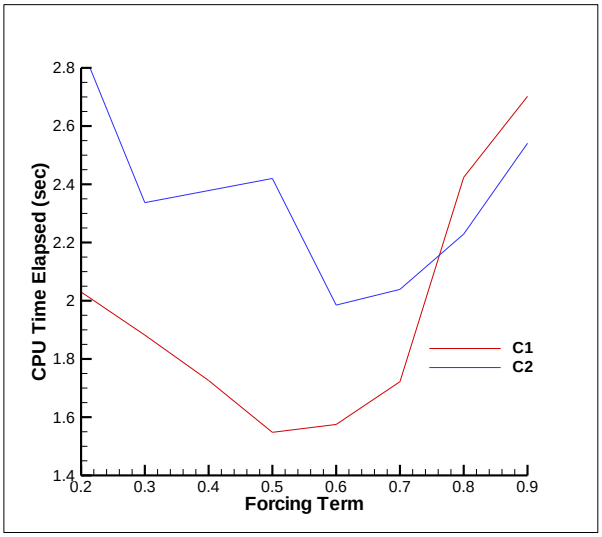

Figure 4.16 Effect of η_k on Coarse Grid

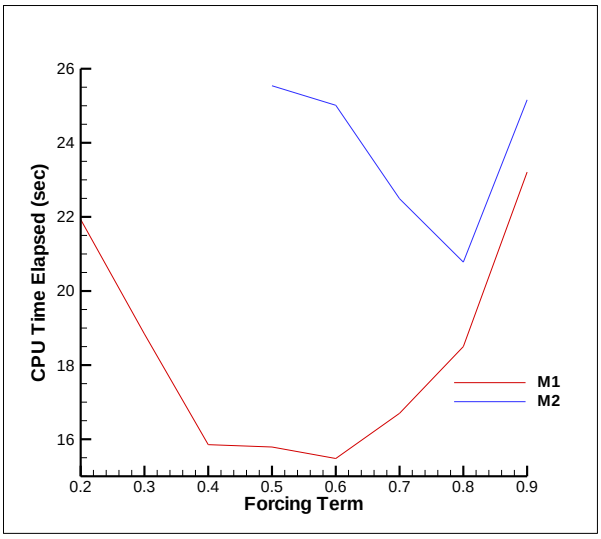

Figure 4.17 Effect of η_k on Medium Grid

With trial and error it is examined that, the best selection can be done by using η_k as 0,6 for Case M1, 0,8 for Case M2. On the other hand, setting η_k to 0,5 would be a better preference while working on case C1 and 0,6 is an appropriate choice of η_k for case C2. This study implies that, for different systems, it is convenient to make an analysis of η_k with respect to CPU time. A tendency to higher values of forcing terms in larger systems is observed.

η_k selection is a vital process while solving the system. In case of selecting either values closer to 0 or close to 1, a convergence couldn't be achieved in most of the situations. On the other hand, Eisenstat formula still works in such situations. However, it can't be concluded that the results would be optimum.

4.4 Mesh Independence Analysis

As the mesh gets finer, the effect of it to the solution decreases. After a point they converge to a same value. The effect of meshes generated to the system defined is analysed below for 1st and 2nd order of discretization.

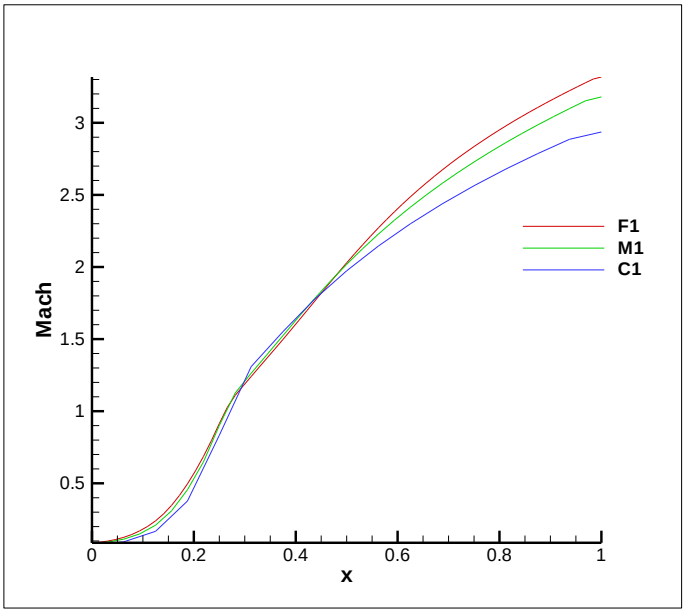

Figure 4.18 Mesh Inpedepence with Newton-GMRES Method on 1st Order Discretization

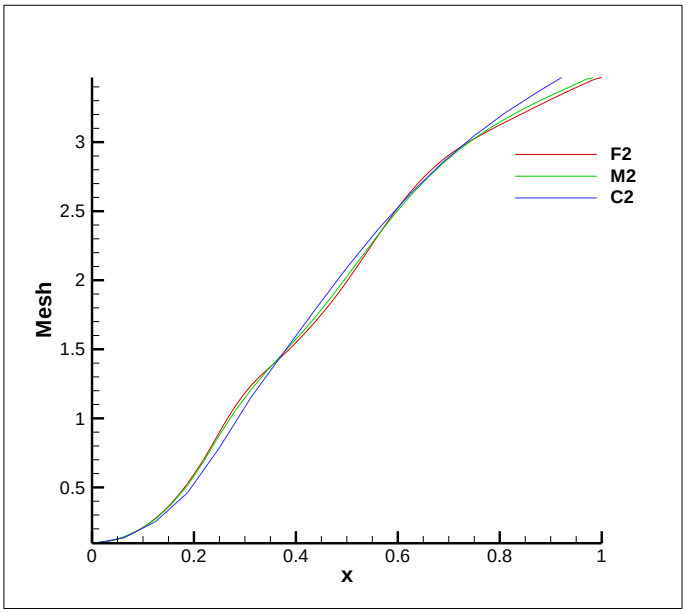

Figure 4.19 Mesh Inpedepence with Newton-GMRES Method on 2nd Order Discretization

While analyzing the first plot, it can be concluded that, the similarity of the solutions between fine and medium grid is more than the similarity between coarse and medium grid. The difference decreases rapidly while going from coarse to fine meshes. Hence, it is predicted that a fourth grid finer than the ones generated before coincides with the results received with fine grid. Due to memory problems it could not been performed. It is seen that for second order of discretization, the sensitivity to the mesh becomes insignificant after the medium grid. Fine and medium grids almost overlap.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In this thesis, both Newton's method and an iterative method are applied to solve 3-D Euler equations respectively. In Newton's method, Van Leer Upwind scheme is used. Two different discretization techniques are implemented in order to make a detailed comparison. A second order discretization: MUSCLE and first order discretization are tried before implementing Newton's and Newton-GMRES methods.

While solution is conducting with Newton's Method, the necessity of calculation of Jacobian matrix has arise. This calculation is carried out with Analytical approach. UMFPACK based on converting a full matrix into a sparse matrix and applying LU decomposition is used in Newton's method to solve the system. A 3-D supersonic nozzle is selected as test case. Different grid sizes are also examined.

The main objective of this study is to implement Newton-GMRES method algorithm into the flow analysis of a 3-D supersonic nozzle. A code is developed by combining Arnoldi iteration Givens rotation and QR factorization. The effect of forcing term and ε are observed. Their optimum values are found either by using well known procedures or trial and error techniques. With these modifications, residual can be decreased up to 10^{-14} besides its significantly small CPU times. Despite of increasing CPU time, a 2^{nd} order discretization converges apparently earlier iterations while combining with Newton-GMRES method

Finally, these two methods are compared on a 3-D supersonic nozzle geometry with various grids. Flow analysis are carried out and observed on Mach contours and pressure distribution. With respect to Newton's method, approximately same results are obtained. Unignorable CPU time saving is provided by Newton-GMRES method for all test cases that were tried out.

As it is explained, Jacobi preconditioner is the basic preconditioner. As a starting point to preconditioning procedure, implementation of Jacobi preconditioner is experienced. Unfortunately, the system diverges after application of Jacobi preconditioning. This type of preconditioning does not guarantee to work effectively. However divergence causes to finalize the study without preconditioning with regarding to the prosperous results in terms of CPU time obtained before applying preconditioner.

5.2 Future Works

It is noticed in the literature research that implementing a preconditioner to the system before starting to the Newton-GMRES method is also a good suggestion in order to enhance the accuracy more than it was provided. However this approach will break down the matrix-free structure of the method which is one of the most important feature of this method. As long as this step requires an approximate Jacobian matrix, computation of it would increase the CPU time. A matrix free preconditioner implementation can be a possibility to improve both accuracy and CPU time together.

Due to being locally convergent methods, initial approximations come into prominence. Globalization procedures are applicable in order to insure convergence with an arbitrary initial selection.

.

Flux calculations for 3-D Euler equations are only conducted for Van Leer flux scheme. Different upwind schemes like Steger Warming or Roe Flux splitting methods can also be tried to see their effect on the solver's performance.

REFERENCES

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

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

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

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

[5] Saad, Y., and Schultz, M.H., "GMRES: A Generalized Minimal Residual Algorithm For Solving Non-Symmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, July 1986.

[6] Wong, Y.S., and Hafez M., "Application of Conjugate Gradient Methods to Transonic Finite Difference and Finite Element Calculations", *AIAA Paper* 81-1032

[7] Elman, H.C.,"Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations", Yale University Department of Computer Science Research Report 229, April 1982.

[8] Wigton, L.B., "GMRES Acceleration of Computational Fluid Dynamics Codes" *AIAA Paper* A85-40933, 1984.

[9] Gear C.W., Saad Y., "Iterative solution of linear equations in ODE codes", *SIAM Journal on Scientific and Statistical Computing, Vol. 4,* 1983, *pp.*583–601,

[10] Brown P.N., Hindmarsh A.C., "Matrix-free methods for stiff systems of ODEs", *SIAM Journal on Numerical Analysis* Vol. 23, 1986, pp. 610–638.

[11] Chan T.F., Jackson K.R.," Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms", *SIAM Journal on Scientific and Statistical Computing, Vol.5,* 1984, pp. 533–542

[12] Brown P.N. and Saad Y., "Hybrid Krylov Methods for Nonlinear Systems of Equations", *SIAM Journal on Scientific and Statistical Computing,* Vol.3, May 1990, pp.450-481.

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

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

[15] 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

[16] Eisenstat, S.C., and Walker, H.F., "Choosing the Forcing Terms in an Inexact Newton Method", *SIAM Journal on Scientific Computation.,* Vol.17, pp.16-32, January 1996.

[17] Pueyo, A., Zingg, D.W., "An Efficient Newton-GMRES Solver for Aerodynamic Computations", *13th AIAA Paper,* A97-32464, 1997.

[18] C. T. Kelly, “Iterative Methods for Linear and Nonlinear Equations,” *Frontiers Appl. Math. 18*, SIAM, Philadelphia, 1999.

[19] Nielsen, E.,J., Anderson, W.K., Walters, R.W., and Keyes, D.E., “Application of Newton-Krylov Methodology to a Three Dimensional Unstructured Euler Code”, AIAA Paper 95-1733-CP, June 1995.

[20] Bellevia, S., and Morini B., “A Globally Convergent Newton-GMRES Subspace Method for Systems of Nonlinear Equations,” *SIAM J. Sci. Comput.*, Vol. 23, No.3, pp.940-960, 2001.

[21] W. E. Arnoldi, “The Principal of Minimized Iteration in the Solution of the Matrix Eigenvalue Problem,” *Quart. Appl. Math.*, pp.17-29, 1951.

[22] Saad, Y., “Iterative Methods for Sparse Linear Systems (Ed.2)”, *SIAM*, 2001.

[23] Auzinger W., “Iterative Solution of Large Linear Systems”, Retrieved January 15, 2014, from http://www.asc.tuwien.ac.at/~winfried/teaching/106.079/SS2011/downloads/script-p-106-122.pdf, 2014.

[24] Persson, P.O., and Peraire, J. “Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations.” *SIAM Journal on Scientific Computing*, Vol 23, No.6, 2008, pp. 2709-2733

[25] Onur, Ö., “Effect of Jacobian Evaluation on Direct Solution of the Euler Equations,” Master thesis, Middle East Technical University, 2005