12-2004

Convergence and Robustness Issues in Computational Fluids

Xiaoqiang Zeng
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Xiaoqiang Zeng entitled "Convergence and Robustness Issues in Computational Fluids." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Aerospace Engineering.

Charles L. Merkle, Major Professor

We have read this dissertation and recommend its acceptance:

Basil Antar, Kenneth R. Kimble, Joe Majdalani

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Xiaoqiang Zeng entitled "Convergence and Robustness Issues in Computational Fluids." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Aerospace Engineering.

Charles L. Merkle

Major Professor

We have read this dissertation
and recommend its acceptance:

Basil Antar

Kenneth R. Kimble

Joe Majdalani

Accepted for the Council:

Anne Mayhew

Vice Chancellor and Dean of
Graduate Studies

(Original Signatures are on file with official student records.)

Convergence and Robustness Issues in Computational Fluids

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Xiaoqiang Zeng

December 2004

Acknowledgements

I sincerely appreciate my thesis advisor Professor Charles L. Merkle for his expert guidance, support and encouragement during the course of my Ph.D study at UTSI, without which nothing could have been achieved.

I would like to express my gratitude to Dr. Majdalani, Dr. Antar and Dr. Kimble for serving in my thesis committee and giving advises to improve my thesis.

Thanks also go to our group members Dr. S. Venkateswaran, Dr. D. Li and Dr. G. Xia for their enlightening discussion, continual encouragement and valuable suggestions during the whole period of my Ph.D study.

I would also like to thank my friends, Kanwai Tong, Shenghong Qiu, Yafang Tao, Wei Su, Ming Xiao, Xiaohai Chen, Lesong Wang, Fan Chen, Men Fan and many other Chinese families in Tullahoma of Tennessee for their considerate care in my life during my study at UTSI.

No words can express my deepest gratitude to my family, including my dear parents, my grandfather, my sister and my brothers for their sustained support and encouragement.

Abstract

The present research develops two methods to improve the convergence and robustness of CFL algorithm, the triple time method and error-limited time step ramping method.

A general formulation of the triple time scheme is developed by introducing three pseudo time-marching steps to control three preconditionings for artificial dissipation, non-linear equation iteration convergence and linear equation iteration convergence separately. It is proven that the triple time method can be degenerated to the single time method and the multiple DDLGS iteration method at special cases.

Stability analysis is used to choose the optimum combination of three preconditionings from the steady preconditioning, the physical and the unsteady preconditioning matrices, and show that the system with unsteady preconditioning for artificial dissipation and linear equation convergence, and physical Jacobian matrix for the non-preconditioning (UPU) gives slightly better stability results than the other systems. The stability results for the 'UPU' triple time system are presented. Some computation results for the linear problem of straight duct flow are given and show a good match with the stability results.

The CPU time saving and the storage cost of triple time method over the single time method are analyzed. The analytical results show that the CPU time per inner iteration is proportional to the square of the number of equations of the system while the CPU time per outer iteration is proportional the cube of the number of equations, and the storage of the triple time costs about four times more than the single time. Some computational results are presented to support the analytical results. The computational results show that the triple time method gains a factor between two and three over the single time in CPU

time.

The robustness of the triple time method is tested and compared with the single time method for the straight duct flow, choked nozzle flow and non-choked nozzle flow. The results show a good improvement of triple time scheme over the single time scheme in robustness for all three cases.

Finally, the error-limited time step ramping method is used to improve the convergence and robustness. A detailed overview of this method is introduced. Some analytical and computational results are provided to prove the feasibility of this method by showing that the implicit error is always less than or equal to the explicit error. Some computational results for the straight duct uniform flow show that the error-limited time step ramping method has improvement in both convergence and robustness.

Contents

List of Tables

List of Figures

Nomenclature

English Symbols

A	Jacobian matrix of flux in x-direction
B	Jacobian matrix of flux in y-direction
CFL	Courant-Friendrichs-Lewy number
D	Diagonally-dominated matrix
E	Flux in x-direction
Err	Approximate factorization error
F	Flux in y-direction or flux
G	Flux in z-direction or amplification matrix
I	Identity matrix
J	Jacobian matrix coefficient
K	Number of faces of each cell
L	Exact operator at left hand side
M	Mach number of the total number of inner iterations
m	Index for the inner iteration
N	Total number of outer iterations or the number of some quantity
n	Index for the outer iteration
p	Static pressure
Q	Primary variable
R	Residual or Ratio
r	Row of a matrix

T	Temperature
U	Magnitude of velocity vector
u	x-component velocity
v	y-component velocity
w	z-component velocity
x	x coordinate drection
y	y coordinate drection
z	z coordinate drection

Greek Symbols

a	Angle of a vector in x-direction of Cartesian coordinate, or error tolerance
b	Angle of a vector in z-direction of Cartesian coordinate
Λ	Diagonal matrix of the eigenvalues
l	Eigenvalue
m	Molecular viscosity
r	Density
Γ	Jacobian matrix of the vector Q with respect to Q_p
t	Pseudo time
Δ	Amount of change of a quantity
∇	Gradient operator
Ω	Volume of a geometry
Σ	Summation

Subscripts

1	Artificial dissipation or the first pseudo time level
2	Non-linear level or the second pseudo time level
3	Linear level or the third pseudo time level
c	cell
dim	dimension
eq	equation
f	face
i	Index of grid points in x-direction
j	Index of grid points in y-direction
k	Index of a face
p	Quantity using the primary variable of Q_p

Superscripts

'	Artificial property
~	Quantity at the third pseudo time level
m	Linear level or the third pseudo time level
n	Non-linear level or the second pseudo time level

Chapter 1
Introduction

1.1 The Convergence and Robustness Issues of CFD

Fluid flow is one of the most common phenomena in the world, but although it has been studied for several centuries it is not completely understood yet. The most popular mathematical decription for the fluid flow problem is the Navier-Stokes equations. Analytical solutions for these equations are seldom accomplished except for several very simple problems because of the complexity of the equations. Numerical methods must be used for most problems. Computation Fluid Dynamics (CFD) is the application of numerical methods to solve the fluid flow conservation equations. With the fast development of computer technology in the last twenty years, CFD technology has greatly progressed and has been applied to many engineering problems.

All CFD algorithms use some kind of iterative method to solve the partial differential equations of fluid flow numerically. Iterative methods typically start with an estimated initial condition and then iterate until some convergence criteria is satisfied. It is useful to plot the results of this iteration process in a figure like figure 1.1 which show a typical plot for a case that converges and a typical plot for one that diverges. The solid line represents a case for which the solution change gets smaller and smaller until convergence is reached. We can artificially divide the solid line into a “non-linear” part where the convergence rate is irregular, and, a “linear part” where the convergence rate is constant. The total number of iterations represented by the solid line, or the speed (efficiency) of the convergence process is an important issue in CFD. The goal of a convergence study is to minimize the total number of iterations. The dashed line in figure

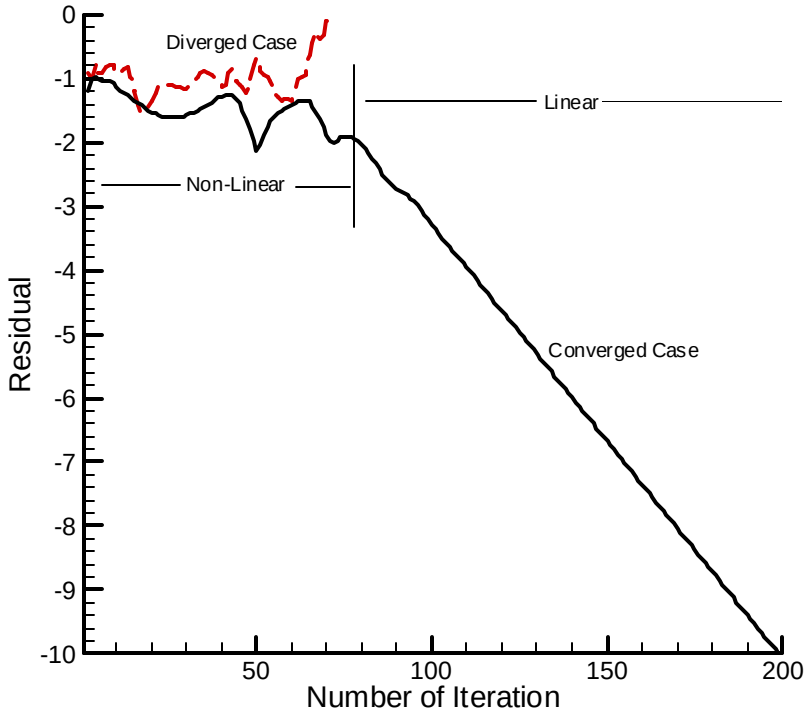

Figure 1.1 A typical converged and diverged convergence for CFD iteration

1.1 represents a diverged case in which the error eventually becomes unbounded. This leads to another very important issue in CFD, what steps are necessary to ensure an iteration converges? In the present thesis we will refer to the boundary between convergence and non-convergence as the robustness problem.

Convergence and Robustness are two major aspects of any iteration method and of CFD. Generally speaking, these two issues depend on the problem, the initial condition and the iterative method used. Also, the convergence and the robustness are not independent of each other. During the non-linear part of the convergence, the non-linear characteristics of the problem dominate the iteration process and create difficulties in both convergence and robustness. The convergence and robustness at this region are mainly related to the initial condition and large changes in the solution at every step. Given a problem and an initial condition, there is a maximum change of the solution the

iteration can bear. Any scheme that gives a solution change larger than this maximum will diverge, and vice versa, but on the other hand, to achieve fast convergence requires a large change of the solution every step. So these two issues act against each other. Almost every iterative method trades one for the other. Once the linear part of the convergence process has been reached, the robustness issue generally no longer exists and the time step can be chosen large to achieve very fast convergence. A good scheme should be able to provide the maximum change of the solution that the robustness can bear during the non-linear portion of the convergence process and use a very large time step or a Newton iteration in the linear part of the convergence.

1.2 Review of Convergence and Robustness Improvement Methods

The most well-known and fastest iteration method is Newton's method. Newton's method is famous for its fast convergence but notorious for its poor robustness. Its convergence is highly dependent on the initial condition. If a good initial condition is given, the iteration can converge very rapidly [20].

In order to improve the robustness of Newton's method, a relaxation iteration method is often introduced [20,27]. In the relaxation method, the new value is updated by weighting of the old value and the new value in the Newton's method. The relaxation is more robust than Newton's method but converges more slowly because the change of the solution at every step is smaller.

One of the very important iteration methods is time marching method [20, 27] which uses a pseudo time to mimic the process of a physical unsteady flow approaching a steady state, but it is not restricted to steady state and can be also applied to transient problems. Actually, the Euler implicit time marching method can be proven to be a special kind of Newton's method where a pseudo time step is introduced to control the

change of the solution every step. When the time step goes to infinity, the time marching becomes exactly the Newton's method.

Time-accurate time marching method works well in transonic and supersonic flow since the particle speed is the same order as the sound speed, but problems arise when it is applied to the incompressible flow and low Mach number compressible flows. For incompressible flow, the time derivative of the continuity equation is zero because the density is constant and the system becomes singular indicating that we can not obtain an update equation from the continuity equation. For low Mach number flow, the waves of the system propagate at very disparate speeds. For implicit scheme, since generally the CFL of the fastest wave should meet some limit requirement, the CFL of the slowest wave becomes very small, which will dominate the convergence and make the whole convergence very slow. In order to solve the above difficulties, the preconditioning method was developed.

The preconditioning method actually is developed by two different philosophies [13,38,11,39], but the results are similar. The first one accelerates convergence by altering the pseudo- time derivative to control the eigenvalues. The second one is based on the small perturbation analysis.

The pioneer work of the first philosophy was done by Chorin, who introduced a properly defined artificial pseudo time derivative in the continuity equation to solve incompressible viscous flows [13]. He referred to it as 'artificial compressibility' instead of 'preconditioning' in that paper. By adding an artificial time derivative, he eliminated the singularity in the time marching method for the incompressible equation system and obtained good convergence.

The extension of this concept to the whole equation system and low speed compressible flow was achieved by Turkel [38], Van Leer [39], Viviand [50] and Briley [3]. In 1983, Briley, McDonald and Shamroth applied the preconditioning method in

isoenergetic flow and the results shows that the method is effective for a Mach number of 0.05 [3]. Turkel was the first one who apply Chorin's artificial compressibility to the whole flow control equation system and all Mach number range [38]. None of the above preconditioners took into account the stiffness at the sonic point. This work was done by Van leer and D. Lee who used the characteristic re-scaling to develop the preconditioner [39]. Their preconditoner removes the stiffness at the sonic point, but the stiffness in the sonic point is not important in most problems.

The work of developing preconditioning by small perturbation analysis is first done by Rehm and Baum [33], followed by Guerra and Gustafsson [19] and by Merkle and Choi [11]. In [19], Guerra and Gustafsson uses the perturbation pressure instead of the thermodynamic pressure as the primary variable in the continuity equation to solve the incompressible two dimensional Euler equation, the small perturbation in their work is proportional to the first power of Mach number. Merkle and Choi further developed this method by using a small perturbation with an order of Mach number squared [11]. They also applied the same philosophy to the energy equation and finally developed a preconditioning system to be used in all speed flow. This form of preconditioning is proven to be Mach number independent for Mach number down to 10^{-5} [18]. In transonic and supersonic regions, the preconditioning system recovers the non-preconditioned system.

In [42], Venkateswaran shows that Merkle-Choi system, Turkel system and Van Leer system are very similar. Compared with the Van Leer system, the Merkle-Choi and Turkel systems are more popular for their simplicity.

Another kind of preconditioning method, block Jacobi preconditioner is generally used for multigrid method. The full coarsening multigrid method encounters difficulty in damping the high frequency modes. Motivated by this problem, Mulder [29] and Allmaras [1] followed by Pierce and Giles [32] developed a block Jacobi preconditioner

to damp the high frequency modes. This method is used for the highly stretched meshes by combination with multigrid method, but it can only improve the convergence at high frequency modes and has nothing to do with the convergence acceleration at low Mach number.

Most of these preconditioners we have discussed were originally developed by only considering the inviscid Euler equation. Problems arose when they are applied to the low speed and low Reynolds number viscous flows. After realizing this problem, Merkle and Choi improved their preconditioner by including the viscous effect using small perturbation analysis [12]. The reason underlying this was found by Venkateswaran to be that the errors of a Navier-Stokes equation are controlled by both the wave propagation and the diffusion damping [42]. In the low Reynolds number flow, the diffusion error damping is dominant and the preconditioning should be based on the viscous term instead of the convection term.

The transient problem is traditionally solved by straight time marching method. Although most of the time the traditional steady preconditioning works well for unsteady problems, under some limits, the unsteady effect will cause some problems and need to be preconditioned. The unsteady preconditioning method is contributed by Venkateswaran [42]. The extension of Merkle-Choi preconditioning method to more advanced systems including chemical reaction is addressed by Venkateswaran, Deshpande, Merkle [45,48], Shuen and Chen [36, 48].

In addition to convergence enhencement, another unexpected advantage of the preconditioning method is an accuracy improvement. Part of the accuracy of a scheme is closely related to the artificial dissipation which is generally controlled by the eigenvalues of the system. In a non-preconditioned system of low Mach number flow, the disparity of eigenvalues results in a oversized or (and) undersized artificial dissipation which will make the solution inaccurate. By balancing all the eigenvalues of the system,

the preconditioning obtains a proper artificial dissipation and makes the solution more accurate.

The preconditioning technology provides a method to solve flow problems for all Mach numbers. Otherwise, we have to solve high Mach number flow by time marching and solve low Mach number flow by some other iterative method, for example, the SIMPLE method [30,31]. Although the preconditioning method has shown great success in enhancing the convergence in many low speed flow applications. There are still some problems to be tackled. The most important disadvantage of the preconditioning method is its lack of robustness. So far, the robustness issue has not been completely understood. The most popular explanation for this issue is the following four points. First, the modification of the eigenvalues of the equation system in low Mach number flow leads to a much larger time step, which lead to the robustness difficulty; Second, the Merkle-Choi preconditioner is derived from the small perturbation analysis, during the derivation, we require the pressure perturbation should be of the order ru^2. If the pressure perturbation does not meet that requirement, the derivation is invalid and preconditioning breaks down. Third, the non-orthogonality of the eigenvectors of the preconditioning system could result in the transient amplification of the errors and consequently lead to the blow up of the convergence. In [14], Darmofal and Schmid stated that 'Due to the lack of eigenvector orthogonality, small perturbations in a linearized evolution problem can be significantly amplified over short time scales while the long time or asymptotic behavior of the linearized system is governed by eigenvalues'. In that paper, they also point out that the Van Leer and Turkel preconditioned systems are highly non-orthogonal for low Mach numbers. Although they did not mention the Merkle-Choi system, we can expect the eigenvectors of this system are also not orthogonal for his similarity with the other two systems. The Block Jacobi preconditioner is the only one who does not suffer from the non-orthogonality at low Mach number. Fourth, our recent study shows that artificial

dissipation plays a very important role in robustness, even more important than the preconditioning in the pseudo time derivative, and our research shows that non-preconditioning in artificial dissipation at the start up of the convergence helps to improve the robustness in our selected computations.

Some representative problems that have difficulties in robustness are the high Mach number stagnant point flow, high area ratio nozzle flow with mass flow specified rather than stagnation pressure and some internal flows involving large pressure perturbation.

In the high area ratio nozzle flow with mass flow specified as one of the boundary conditions, a good initial condition generally is not available at the beginning of the computation, and the conservation of mass will result in a large pressure and velocity fluctuation. In an internal flow, the pressure perturbation is easier to build up for the confining wall than in an external flow where the errors can be convected out of the boundary. Another factor that causes the large pressure perturbation is involved in combustion. The huge heat release in a short time results in an extremely high pressure and temperature perturbations.

The preconditioning system encounters difficulties when applied to the stagnant point flow problem. The stagnant point problem in incompressible flow or low Mach compressible flow is relatively easy to solve because of the narrow range of Mach number and small pressure gradient. The only difficulty is the preconditioning at the stagnant point because of the zero velocity, which can be tackled by using the free stream velocity to define the preconditioning, but in a relatively high Mach number flow, the stagnant point problem becomes much more difficult to solve because of the mutiple Mach number regimes and the high pressure gradient in the vicinity of the stagnant point. The low Mach number region near the stagnant point forces us to use preconditioning at that region, but at the same time the preconditioning method suffers from the large pressure gradient in the vicinity of the stagnant region and results in a convergence

problem.

This problem has received considerable attention in recent years and a lot of work have been done to try to solve this prolem. Weiss, Maruszewski and Smith [52] use a cut off value based on the pressure gradient at the vicinity of the stagnant point to sense the pressure gradient. If the pressure gradient is very large, the preconditioning is switched to be close to the non-preconditioning and vice verse. This appears to be a good idea but is proven to be not general. Darmofal and Siu use a combination of the local preconditioner with a cut off value based on the pressure perturbation and the point block jacobi preconditioner to improve the robustness of their multigrid scheme[15]. Their results show a robustness gain. Based on the results, they believe that the gain is more from the point block jacobi preconditioner than from the new local preconditioner with pressure perturbation cut off because of the eigenvector orthogonality of point block jacobi preconditioner. Darmofal and Schmid modified the Turkel and Van Leer preconditioners by making their eigenvectors more orthogonal and showed an improvement in robustness [14]. Darmofal and Van Leer also improved the Van Leer preconditioner by requiring the eigenvector orthogonality [16]. Unfortunately, the improvement is limited to the streamwise direction because of some limitations.

Another avenue to get around the robustness issue is proposed by Venkateswaran, Merkle and Zeng [41, 42, 55], they referred to it as 'dual time method' in that notes and is changed to 'triple time method' later. We call it 'triple time method' in this thesis. The triple time method treats the accuracy, the non-linear robustness and convergence, the linear convergence separately by three pseudo times so that we can control these three issues independently. The preconditioning is used in the artificial dissipation control and the linear convergence control. The non-linear robustness and convergence is controlled by non-preconditioned time marching so that large pressure perturbation can be handled. In the linear solver level, we can store the decomposition of the LU solver to save CPU

time with the cost of more storage. Significant improvement in robustness and convergence are observed and published in [42], [55], [46] and [47].

Regarding the convergence issue, one of the most difficult convergence issues for any Mach number flow in modern CFD is the high Reynolds number flow applications. In a high Reynolds number flow, most of the region is convection-dominated. Only in a very thin region the viscous term plays an important role. That thin region is generally called the 'boundary layer'. In the boundary layer, a large velocity gradient exists and a highly stretched grid is required to resolve the accuracy of the large pressure gradient. The aspect ratio of the highly stretched grid could be 1000 or even higher. The difficulty of solving high Reynolds number flow does not lie in the addition of the viscous term in the equation but the highly stretched grid. Actually, the addition of the viscous term makes the equation easier to solve. Because of the highly stretched grid, the satisfaction of the CFL limit in the long direction results in an extremely small CFL in the short direction which will dominate the whole convergence.

So far not a lot of work has been done about the high aspect ratio problem. Briley, Govindan and Mcdonald [4] studied the effect of aspect ratio on the convergence of various schemes. A systematical study of the high aspect ratio problem is done by Beulow, Merkle and Venkateswaran [6,7,8]. The conventional opinion is that we should use the maximum of all the CFLs in different directions for explicit scheme, but in his Ph.D thesis [8], Beulow found that the minimum one should be used in the two dimensional implicit approximate factorization scheme for the Euler equation, which contradicts the conventional thought that the maximum one also should be used in implicit schemes. In the Navier-Stokes equations, a local time step should be based on the combination of the minimum CFL and the maximum VNN [9]. The conclusion becomes grid-dependent when it is extended to three dimension. For the type of grid that has only one short dimension, the time step based on the minimum CFL and the maximum VNN still works

well, but it does not work for the other types of high grid aspect ratio grids. The underlying reason can be found by examining the approximate factorization error.

Another remedy to tackle the high aspect ratio problem was suggested to be the multigrid method. For most iterative methods, the high frequency errors are very easy to damp while the low frequency errors are very stiff. Therefore, the low frequency errors become the bottleneck for convergence. To alleviate this prolem, the multigrid method is designed to remove the high frequency errors in the fine grids and the low frequency errors in the coarse grids. So, multiple grids are used in this method. The solutions and residuals in different grids are transferred to the next grid level. The accuracy of the final solution is guaranteed by finishing the iteration on the finest grid. Because its powerful capability of damping the low frequency errors, it is used for solving the high aspect ratio problem. Some of the research results can be found in [26, 52].

1.3 A Brief Introduction to the Triple Time Method and the Error-Limited Time Step Ramping

So far, we have reviewed the history of various robustness and convergence improvement methods. Although a lot of remedies have been suggested to solve these problems, they are proven to be problem-dependent and performed well for the problems they are designed for but generally fail for the other problems. A very promising solution appears to be to combine the triple time method with error-limited time step ramping technology.

The triple time scheme was originally proposed by Venkateswaran and Merkle [42]. This scheme uses three pseudo times to control the accuracy, the robustness and non-linear convergence, and linear convergence separately so that we can use different preconditioning according to different requirements. By solving the non-linear equation

exactly instead of approximately, we can use an infinite CFL at the linear part of the outer convergence and obtain an exact Newton solver. So no preconditioning is needed in the non-linear level because we can use an infinite CFL, but at the non-linear part of the outer convergence, we can not use an infinite CFL and have to start with a small CFL and ramp it to infinity. So at that part the outer convergence will be very slow if it is not preconditioned. This is especially true in complex problems, but this issue can be alleviated by using the error-limited ramping technology to optimize the CFL ramping at the non-linear part of convergence. The error-limited ramping technology is first introduced by Edluke (it is not published), who ramps the CFL by allowing the change of the solution by explicit method to be less than a specific amount.

1.4 The Present Research

In this thesis, we will study the convergence of the triple time scheme by the stability analysis and use some simple computational cases to test the convergence and robustness of our scheme. Also, an anlysis for the error-limited CFL ramping method is given and some computational results are presented.

The code we are going to use is an in-house code called GEMS (general equation and mesh solver) [24], which is a three dimensional, unstructured, parallel code for arbitrary fluids. It has been used for turbulent flow, steady and unsteady, combustion, and MHD applications. The primary linear solvers are Line Gauss Seidel [8] and GMRES [35, 53].

The thesis is organized as follows. In chapter two, we formulate the general triple time formulation and introduce the Diagonally Dominant Line Gauss Seidel linear solver. In chapter three, using stability analysis, we optimize the three preconditioners for artificial dissipation, non-linear robustness and convergence control, and linear convergence control. After that, we present the stability analysis result of our triple time

system. In chapter four, the CPU time and storage for linear iteration and non-linear iteration are analytically estimated and compared with the numerical computation. In chapter five, some robustness and convergence improvement results are presented and analyzed for selected cases. In chapter six, we discuss the error-limited time step ramping issue. A theoretical reason for the feasibility of this method is given and some numerical results of this method are presented. In chapter seven, we summarize some of the results we obtained and give a direction for future research.

Chapter 2
Triple Time Formulation

2.1 Introduction

In this chapter, we begin with the general equations of fluid mechanics to formulate the triple time formulation. The first pseudo time produces the artificial dissipation by following the standard approximate Riemman Solver. The second pseudo time is introduced to solve the discretized equation system by an implicit marching method. The third pseudo time is to solve the linear equation obtained in the second pseudo time iteratively. Finally, to solve the triple time equation, the approximate factorization solver of diagonally dominant line Gauss-Siedel (DDLGS) is presented.

2.2 Governing Equation

The unsteady Navier-Stokes equation can be written as the divergence of a four-dimensional vector:

$$\nabla \cdot F = 0 \tag{2.1}$$

where, the flux vector F is given by

$$F = Qe_t + (E - E_v)i + (F - F_v)j + (G - G_v)k \tag{2.2}$$

and the divergence operator is

$$\nabla = \frac{\partial}{\partial t}e_t + \frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial z}k$$

Q, E, F and G are the conservative flux vectors in x, y and z directions respectively. They are:

$$Q=\begin{pmatrix} r \\ ru \\ rv \\ rw \\ rh^0 - p \end{pmatrix},\ E=\begin{pmatrix} ru \\ ru^2 + p \\ ruv \\ ruw \\ ruh^0 \end{pmatrix}, F=\begin{pmatrix} rv \\ ruv \\ rv^2 + p \\ rvw \\ rvh^0 \end{pmatrix}, G=\begin{pmatrix} rw \\ ruw \\ rvw \\ rw^2 + p \\ rwh^0 \end{pmatrix} \tag{2.3}$$

E_v, F_v and G_v are the viscous flux vectors,

$$E_v=\begin{pmatrix} 0 \\ t_{xx} \\ t_{yx} \\ t_{zx} \\ ut_{xx} + vt_{yx} + wt_{zx} + kT_x \end{pmatrix},\ F_v=\begin{pmatrix} 0 \\ t_{xy} \\ t_{yy} \\ t_{zy} \\ ut_{xy} + vt_{yy} + wt_{zy} + kT_y \end{pmatrix},$$

$$G_v=\begin{pmatrix} 0 \\ t_{xz} \\ t_{yz} \\ t_{zz} \\ ut_{xz} + vt_{yz} + wt_{zz} + kT_z \end{pmatrix}$$

Where $t_{xx}, t_{xy}, \ldots$ are the nine components of the stress tensor $\vec{t}$ and are given as

$$t_{ij} = -\frac{2}{3}m(u_x + v_y + w_z)d_{ij} + m\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

In this thesis, only the Euler equations are considered. The methods defined extend directly to the full Navier-Stokes equation.

Equation 2.1 is the most general equation. It is unsteady, viscous and three-dimensional. It can be simplified to steady, inviscid, two-dimensional or one-dimensional equations very easily.

As witten, equation 2.1 contains 6 unknowns (p, r , h^0,u, v and w) but only 5 equations. To close this system, we must add two more equations, the state equation and the enthalpy equation. To enable the equation set to apply to general fluids including perfect gases, incompressible liquids, real gases, supercritical fluids and other generic fluids, we express the equation of state as an arbitrary function of pressure and temperature

$$\rho = \rho(p, T)$$

and choose an analogous relation for enthalpy,

$$h = h(p, T)$$

The stagnation enthalpy, h^0, can then be defined as,

$$h^0 = h + \frac{1}{2}\left(u^2 + v^2 + w^2\right)$$

The equation system is closed by adding these three relations to equation 2.1.

For completeness, we also define the internal energy, e, in terms of the enthalpy, pressure and density as,

$$h = \varepsilon + p/\rho$$

and define the total internal energy,

$$re = r\left[e + \left(u^2 + v^2 + w^2\right)/2\right]$$

Straightforward extensions of these equations along with the incorporation of additional conservation equations allow this system to be applied to multi-component and multi-phase fluids [25,40,48].

2.3 The Time Marching Method and Selection of the Primary Dependent Variables

The time marching method is an iterative method that adds a pseudo time to simulate the physical transition from unsteady to steady state. It uses the pseudo time step to control the convergence and robustness by analog with the physical time derivative. We add a pseudo time to equation 2.1, obtain:

$$\frac{\partial Q}{\partial t} + \nabla \bullet F = 0 \tag{2.4}$$

Note that all derivatives are in conservative form in the formulation 2.4. We must use the conservative form in the physical derivatives to keep accuracy (note that this statement applies to both physical time and space). But it is not necessary to do that in the pseudo time derivative since it just serves as a path to the final solution. Whether it is

conservative or not does not affect the final solution apart from a possible impact on the artificial dissipation as is noted below. By choosing any variable set Q_a as the primary dependent variable, we can write the pseudo time derivative in equation 2.4 in the non-conservative form:

$$\frac{\partial Q}{\partial Q_a}\frac{\partial Q_a}{\partial t} + \nabla \bullet F = 0 \tag{2.5}$$

In principle, the primary variable Q_a in equation 2.5 is arbitrary as long as each of the four flux vectors, Q, E, F and G can be expressed as a unique function of the primary dependent variable. Consequently there are numerous potential choices for the primary dependent variable. Several of the most commonly used variables are discussed below. The first choice is the conservative vector, Q defined in equation 2.3. The shortcoming of using this conservative variable is that it is not the variable that we are familiar with and can be easily measured; A second choice is a set containing entropy, for example, $Q_S = (p,u,v,w,s)^T$. Other sets of variables in this family can be obtained by replacing 'p' by 'r', 'T' or enthalpy. A third choice is the "non-conservative" variable vector, $Q_p = (r,u,v,w,p)^T$. The fourth choice is the primitive vectors, $Q_p = (p,u,v,w,T)^T$. The primitive variables appear to be the most appropriate choice because of the following,

1. They apply to both compressible and incompressible flow since all of the components of Q_p vary in both compressible and incompressible flow. Note choosing density as one of the primary variable precludes the computation of incompressible flow in which the density is a constant.
2. When pressure and temperature are used as the primary dependent variables, we can calculate the enthalpy directly rather than iteratively as when h is known and T is to be determined.
3. The primitive variables set allows us to compute directly the variables that are generally of most interest.

By choosing $Q_p = (p, u, v, w, T)^T$ as the primary dependent variable, we can write equation 2.5 as:

$$\frac{\partial Q}{\partial Q_p}\frac{\partial Q_p}{\partial t} + \nabla \bullet F = 0 \tag{2.6a}$$

For convenience, we can then define the matrix, Γ_p, as

$$\Gamma_p = \frac{\partial Q}{\partial Q_p} \tag{2.7}$$

so that equation 2.6a becomes

$$\Gamma_p \frac{\partial Q_p}{\partial t} + \nabla \bullet F = 0 \tag{2.6b}$$

Since the pseudo time term, $\Gamma_p \frac{\partial Q_p}{\partial t}$ has no effect on the final converged solution, the physical Jacobean, Γ_p, need not be used but can be replaced by another matrix. An important point to note here, is that the choice of this matrix can affect both the convergence and the robustness of the iteration. For now, we simply replace the physical Jacobean Γ_p by some "artificial" matrix Γ. Then equation 2.6b becomes

$$\Gamma \frac{\partial Q_p}{\partial t} + \nabla \bullet F = 0 \tag{2.6c}$$

In summary, it is important to emphasize that the sole purpose of the pseudo time is to obtain the solution to Equation 2.1. Thus, it is appropriate to choose the vector of primary dependent variables for convenience. Similarly, it is important to choose the coefficient matrix, Γ, so that it provides the most efficient convergence. This construction provides a very flexible 'iteration' procedure. We discuss methods for selecting Γ in later sections, but first digress to discuss the several functions that a pseudo time serves.

2.4 Triple Time Formulation

To solve equation 2.1, We introduce a triple time formulation. The first pseudo time is introduced to define the artificial dissipation. A discretized equation is obtained in this step. The second pseudo time is introduced to solve the discretized equation. The Euler implicit method is used and the equation is linearized and written in the delta form. The formulation for this step must be designed to deal with the robustness difficulty of the highly non-linear part of the convergence process. A third pseudo time is introduced to solve the linearized delta form equation by another time marching method and should be chosen to achieve optimum convergence of the linearized equation.

2.4.1 The First Pseudo Time: Defining the Artificial Dissipation

No matter how it is added, all CFD schemes contain artificial dissipation. Some add artificial dissipation by means of an overt action while some incorporate it inherently through the discretization. No matter how it is added, artificial dissipation plays a very important role in CFD and can affect both the rate of convergence and the accuracy of the final solution. Depending upon the manner in which it is formulated, artificial dissipation can improve, have no effect or detract from the rate of convergence of a computational algorithm. Similarly, it can improve the accuracy of the final solution by eliminating "wiggles" and detract from the accuracy by smoothing out steep gradients and shock features. Consequently, it is very important to keep a proper amount of artificial dissipation in the scheme. An important characteristic of an artificial dissipation criteria is the rate at which it decreases as the grid is refined.

In the present thesis, we obtain the artificial dissipation by a modification of the standard Godunov/Riemann procedure [20, 27]. The issue regarding the definition the proper amount of artificial dissipation is presented later.

Traditionally, the Godunov/Riemann method uses a physical time as an aid to define the artificial dissipation. In the present approach, we replace this physical time by an artificial time. Because we anticipate using other pseudo times, we use the subscript

'1' to denote the present pseudo time. Consequently, by replacing Γ by Γ_1 and t by t_1 in equation 2.6c, we obtain

$$\Gamma_1 \frac{\partial Q_p}{\partial t_1} + \nabla \cdot F = 0 \tag{2.6d}$$

To solve equation 2.6d numerically, we integrate this equation over a space-time volume Ω, obtain:

$$\oint_\Omega \Gamma_1 \frac{\partial Q_p}{\partial t_1} d\Omega + \oint_\Omega \nabla \bullet F d\Omega = 0 \tag{2.8a}$$

The integrand $\Gamma_1 \frac{\partial Q_p}{\partial t_1}$ can be converted to a perfect differential by defining a new variable, Q_1' as,

$$\Gamma_1 \frac{\partial Q_p}{\partial t_1} = \frac{\partial Q_1'}{\partial t_1}$$

where we have taken the coefficient matrix, Γ_1, as the Jacobian,

$$\Gamma_1 = \frac{\partial Q_1'}{\partial Q_p} \tag{2.8b}$$

Using this new variable, Q_1', we can now compute the average integral of the pseudo time derivative over the volume, Ω, by defining an average value over the control volume,

$$\overline{Q_1'} = \frac{1}{\Omega} \int_\Omega Q_1' d\Omega$$

The integral of the pseudo time derivative then becomes:

$$\oint_\Omega \Gamma_1 \frac{\partial Q_p}{\partial t_1} d\Omega = \oint_\Omega \frac{\partial Q_1'}{\partial t_1} d\Omega = \frac{\partial}{\partial t_1} \left\{ \oint_\Omega Q_1' d\Omega \right\} = \frac{\partial \overline{Q_1}'}{\partial t_1} \Omega \tag{2.9}$$

The second term, the convective integral, can be converted to a surface integral by using Green's theorem as shown in the following

$$\oint_\Omega \nabla \bullet F d\Omega = \oint_{\partial\Omega} F \bullet \mathbf{n} dS \tag{2.10}$$

Where n is the unit normal vector of the surface and $\partial\Omega$ is the surface area of volume Ω in a four dimensional space (space-time).

Substituting equation 2.9 and equation 2.10 into equation 2.8a, gives,

$$\frac{\partial \overline{Q_1}'}{\partial t_1}\Omega + \oint_{\partial\Omega} F \bullet \mathbf{n} dS = 0$$

We now divide the surface of the volume into *K* distinct faces of finite area so that this equation becomes:

$$\frac{\partial \overline{Q_1}'}{\partial t_1}\Omega + \sum_{k=1}^{K} \oint_{\partial\Omega} F_k \bullet \mathbf{n}_k \, dS_k = 0$$

where subscript 'k' is the face label. If every surface k is a planar surface and F_k is constant on surface k or if F_k is viewed as an average, we can write the last equation as:

$$\frac{\partial \overline{Q_1}'}{\partial t_1}\Omega + \sum_{k=1}^{K} F_k \bullet \mathbf{n}_k S_k = 0 \tag{2.11}$$

This equation is the standard form for any approximate Riemann solver except that we have extended the space flux to a space-time flux, we have used a pseudo time, and the primary variables are not in the standard form. Using the standard high resolution procedures [27, 20], the fluxes can be evaluated by using an approximate Riemann solver in which the normal flux, F_{nk}, through the face *k* is calculated from the numerical flux vector,

$$\begin{aligned} F_{nk} &\equiv \frac{1}{2}\left(F_{nL} + F_{nR}\right)_k - \frac{1}{2}\left|\frac{\partial F_n}{\partial Q_1'}\right|_k \left(dQ_1'\right)_k \\ &= \frac{1}{2}\left(F_{nL} + F_{nR}\right)_k - \frac{1}{2}\left|\frac{\partial F_n}{\partial Q_p}\frac{\partial Q_p}{\partial Q_1'}\right|_k \left(\frac{\partial Q_1'}{\partial Q_p}\right)_k \left(Q_{pR} - Q_{pL}\right)_k \end{aligned} \tag{2.12}$$

Where $F_n = F \bullet n$ and the subscripts 'L' and 'R' on F_n and Q_p indicate the left and right side of the surface 'k' respectively. Note that we must retain the variable, Q_1', in computing this flux since it is the conservative variable that appears in the time derivative. The second term in the numerical flux in Equation 2.12 is the traditional artificial dissipation term that is introduced in the approximate Riemann procedure. Accordingly, we also see that Q_1' is the variable with which we define the artificial dissipation.

Defining the flux Jacobian in the normal direction,

$$A_{pn} = \frac{\partial F_n}{\partial Q_p}$$

and using equation 2.8b to replace the pseudo time Jacobian $\frac{\partial Q_1'}{\partial Q_p}$ by its equivalent matrix, Γ_1, we have,

$$F_{nk} = \frac{1}{2}\left(F_{nL} + F_{nR}\right)_k - \frac{1}{2}\left|A_{pn}\Gamma_1^{-1}\right|_k \left(\Gamma_1\right)_k \left(Q_{pR} - Q_{pL}\right)_k$$

Using the identity

$$\left|A_{pn}\Gamma_1^{-1}\right|_k \left(\Gamma_1\right)_k = \left(\Gamma_1\right)_k \left|\Gamma_1^{-1}A_{pn}\right|_k$$

and dropping the subscript n on the flux vector and the Jacobian by assuming that the fluxes and Jacobians all represent the flux in the normal direction on the given face area, we get the final form for the normal flux,

$$F_k = \frac{1}{2}\left(F_L + F_R\right)_k - \frac{1}{2}\left(\Gamma_1\right)_k \left|\Gamma_1^{-1}A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k \tag{2.13}$$

The last term in equation 2.13 is the artificial dissipation which is related by the pseudo time, t_1 and the matrix, Γ_1.

Substituting equation 2.13 into equation 2.11, gives,

$$\frac{\partial \overline{Q_1}'}{\partial t_1}\Omega + \sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right)_k - \frac{1}{2}\left(\Gamma_1\right)_k \left|\Gamma_1^{-1}A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k\right]S_k = 0 \tag{2.14}$$

To de-couple the artificial dissipation from the iteration process, we discretize the t_1 time derivative, set Δt_1 to be infinity, and divide by Ω in equation 2.14 to obtain the algebraic flux conservative relation,

$$\frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right)_k - \frac{1}{2}\left(\Gamma_1\right)_k \left|\Gamma_1^{-1}A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k\right]S_k = 0 \tag{2.15a}$$

By pulling all subscripts 'k' out of the bracket, we define this algebraic flux conservative vector as the discretized divergence operator

$$\nabla_D \bullet F = \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(Q_{pR} - Q_{pL}\right)\right]_k S_k \tag{2.15b}$$

Substituting 2.15b into equation 2.15a, we obtain a more compact form for equation 2.15a as

$$\nabla_D \bullet F = 0 \tag{2.15c}$$

Note that this divergence operator is algebraic and no longer differential. In addition, note that the Jacobian matrices Γ_1 and A_p now appear in the artificial dissipation term. Accordingly, the matrix $\Gamma_1\left|\Gamma_1^{-1}A_p\right|$ should be defined very carefully to ensure both a proper amount artificial dissipation and that the jump in the flux vector is equal to the mean Jacobian times the jump in the primary unknown variable. The preconditioning Γ_1 developed by Merkle and Choi [11] can ensure the proper amount of artificial dissipation. These issues will be discussed in detail later.

2.4.2 The Second Pseudo Time: Solving the Non-Linear Equation

To solve the discretized non-linear equation 2.15a, we add a second pseudo time t_2 and introduce a corresponding matrix Γ_2 to give another time-marching equation.

$$\Gamma_2 \frac{\partial Q_p}{\partial t_2} + \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right)_k - \frac{1}{2}\left(\Gamma_1\right)_k\left|\Gamma_1^{-1}A_p\right|_k\left(Q_{pR} - Q_{pL}\right)_k\right]S_k = 0 \tag{2.16}$$

Note that we allow the new matrix Γ_2 to be distinct from the artificial dissipation matrix Γ_1 if this should prove convenient.

After discretizing by the Euler implicit method, this equation becomes

$$\frac{\Gamma_2}{\Delta t_2}(Q_p^{n+1} - Q_p^n) + \frac{1}{\Omega}\left\{\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right)_k - \frac{1}{2}\left(\Gamma_1\right)_k\left|\Gamma_1^{-1}A_p\right|_k\left(Q_{pR} - Q_{pL}\right)_k\right]S_k\right\}^{n+1} = 0 \tag{2.17}$$

Where the superscript ‘n’ indicates the t_2 time level.

Next we use the standard linearization procedure to linearize equation 2.17. According to the Taylor series expansion, we have,

$$F^{n+1} = F^n + \left(\frac{\partial F}{\partial t_2}\right)^n \Delta t_2 + o\left(\Delta t_2^{\,2}\right) = F^n + \left(\frac{\partial F}{\partial Q_p}\right)^n \frac{\partial Q_p}{\partial t_2}\Delta t_2 + o\left(\Delta t_2^{\,2}\right)$$

$$= F^n + \left(\frac{\partial F}{\partial Q_p}\right)^n \Delta Q_p + o\left(\Delta t_2^{\,2}\right)$$

where $\Delta Q_p = Q_p^{n+1} - Q_p^n$.

Ignoring the higher order term, the last equation becomes,

$$F^{n+1} = F^n + \left(\frac{\partial F}{\partial Q_p}\right)^n \Delta Q_p$$

Applying the last equation to F_L and F_R, since the flux at the left (right) hand side is a function of the variable at the left (right) side of the face

$$F_L = F_L\left(Q_{pL}\right)$$

and

$$F_R = F_R\left(Q_{pR}\right)$$

we obtain,

$$F_L^{\,n+1} = F_L^{\,n} + \left(\frac{\partial F_L}{\partial Q_{pL}}\right)^n \Delta Q_{pL} \qquad (2.18a)$$

and

$$F_R^{\,n+1} = F_R^{\,n} + \left(\frac{\partial F_R}{\partial Q_{pR}}\right)^n \Delta Q_{pR} \qquad (2.18b)$$

Note that for first order system, the variables at the face, Q_{pL} and Q_{pR} are equal to the value at the left and right cells of the face respectively, but for second order system are computed by the reconstruction procedure (see subsection 2.5) and are functions of the corresponding variable at all adjacent cells. We define

$$A_{pL} = \frac{\partial F_L}{\partial Q_{pL}}$$

and

$$A_{pR} = \frac{\partial F_R}{\partial Q_{pR}}$$

Substituting these two relations into equations 2.18a and b respectively, we obtain

$$F_L^{\,n+1} = F_L^{\,n} + A_{pL}^{\,n} \Delta Q_{pL} \tag{2.18c}$$

and

$$F_R^{\,n+1} = F_R^{\,n} + A_{pR}^{\,n} \Delta Q_{pR} \tag{2.18d}$$

The artificial dissipation term in equation 2.17 can be linearized in a similar way. Before linearizing it, we express it in a simpler form by defining

$$O = \frac{1}{2}(\Gamma_1)_k \left|\Gamma_1^{-1} A_p\right|_k \tag{2.18e}$$

Then the artificial dissipation term becomes

$$\frac{1}{2}(\Gamma_1)_k \left|\Gamma_1^{-1} A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k = O\left(Q_{pR} - Q_{pL}\right)_k$$

The linearization of the artificial dissipation becomes

$$\begin{aligned}
&\left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n+1} \\
&= \left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \Delta t_2 \left\{\frac{\partial}{\partial t_2}\left[O\left(Q_{pR} - Q_{pL}\right)_k\right]\right\}^{n} \\
&= \left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \Delta t_2 \left\{O\frac{\partial}{\partial t_2}\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \Delta t_2 \left\{\left(Q_{pR} - Q_{pL}\right)_k \frac{\partial O}{\partial t_2}\right\}^{n} \\
&= \left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \left\{O\left(\Delta Q_{pR} - \Delta Q_{pL}\right)_k\right\}^{n} + \Delta t_2 \left\{\left(Q_{pR} - Q_{pL}\right)_k \frac{\partial O}{\partial t_2}\right\}^{n}
\end{aligned}$$

Ignoring last term at the right hand side of the last equation, we obtain,

$$\left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n+1} = \left\{O\left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \left\{O\left(\Delta Q_{pR} - \Delta Q_{pL}\right)_k\right\}^{n}$$

Substituting equation 2.18e into the last equation, gives

$$\left\{\frac{1}{2}(\Gamma_1)_k \left|\Gamma_1^{-1} A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k\right\}^{n+1} = \left\{\frac{1}{2}(\Gamma_1)_k \left|\Gamma_1^{-1} A_p\right|_k \left(Q_{pR} - Q_{pL}\right)_k\right\}^{n} + \left\{\left[\frac{1}{2}(\Gamma_1)_k \left|\Gamma_1^{-1} A_p\right|_k\right]\left(\Delta Q_{pR} - \Delta Q_{pL}\right)_k\right\}^{n} \tag{2.18f}$$

Substituting equations 2.18c, d and f into equation 2.17 and pulling all the subscripts 'k' out of the bracket, we obtain,

$$\begin{aligned}&\left\{\frac{\Gamma_2}{\Delta t_2}+\frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL}\bullet_L+A_{pR}\bullet_R\right)-\frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(\bullet_R-\bullet_L\right)\right]_k S_k\right\}^n \Delta Q_p \\ &=-\left\{\frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L+F_R\right)-\frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(Q_{pR}-Q_{pL}\right)\right]_k S_k\right\}^n\end{aligned} \tag{2.19a}$$

The dot '$\bullet$' implies the term ΔQ_p should be put in the dot position. For example,

$$\begin{aligned}&\left\{\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL}\bullet_L+A_{pR}\bullet_R\right)-\frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(\bullet_R-\bullet_L\right)\right]_k S_k\right\}^n \Delta Q_p \\ &=\left\{\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL}\Delta Q_{pL}+A_{pR}\Delta Q_{pR}\right)-\frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(\Delta Q_{pR}-\Delta Q_{pL}\right)\right]_k S_k\right\}^n\end{aligned} \tag{2.20a}$$

Note that the quantities, ΔQ_p that appear in this expression are not the unknowns at the cell center, but rather the reconstructed values at the cell faces (see section 2.5).

To simplify the notation, we define the discretized Jacobian divergence operator as

$$\nabla_D\cdot A_p=\frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL}\bullet_L+A_{pR}\bullet_R\right)-\frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(\bullet_R-\bullet_L\right)\right]_k S_k \tag{2.20b}$$

Equation 2.20b is a general expression for the Jacobian divergence. Later on, we will use different subscripts on the operator $\nabla_D\cdot A_p$ to indicate the application of equation 2.20b to the specific conditions. The subscripts '2' and '3' will be used to indicate the second and the third pseudo times respectively, and the subscripts 'I' and 'II' will be used for the first and second order discretization respectively.

Substituting equation 2.20b and 2.15b into equation 2.19a and use a subscript '2' as the operator $\nabla_D\cdot A_p$ to indicate the second pseudo time, we obtain

$$\left[\frac{\Gamma_2}{\Delta t_2}+\left(\nabla_D\cdot A_p\right)_2\right]^n\Delta Q_p=-\left(\nabla_D\cdot F\right)^n \tag{2.19b}$$

For later convenience, we define the whole operator on the left hand side and the residual of equation 2.19b as

$$L_2 = \left[\frac{\Gamma_2}{\Delta t_2} + \left(\nabla_D \cdot A_p \right)_2 \right]^n \tag{2.20c}$$

and

$$R_2 = \left(\nabla_D \cdot F \right)^n \tag{2.20d}$$

so that equation 2.19b can be written as the very simple form

$$L_2 \Delta Q_p = -R_2 \tag{2.19c}$$

The Euler implicit formulation in equation 2.19b, in theory, provides an iterative path for updating the variable, Q_p. Our goal is to perform the iteration in an optimum manner. The convergence rate of a fully implicit time-marching algorithm increases monotonically as the time step is increased if the initial condition is sufficiently close to the final solution) and duplicates Newton's method in the limit of an infinite time step. The deficiency is that Newton's method is highly efficient near convergence, but is sensitive to initial conditions and will diverge when the initial condition lies too far from the solution. Euler-implicit time-marching methods provide a natural path around this difficulty. A small time step can be used to limit the changes in the solution at the outset of the computation (the 'inexact Newton' problem) and then increase as the solution nears convergence.

Although equation 2.19b is linear, it is expensive to solve because it is wide banded for multi-dimensional systems. Conventionally it is solved by some approximate factorization method, such as ADI, LU, line Gauss-Siedel [8,9] or GMRES [35,53].

Approximate factorization methods generally use a first order discretization for the t_2 Jacobian divergence operator, $\left(\nabla_D \cdot A_p \right)_2$, which will slow down the convergence rate for higher order discretization of the right hand side, but will not impact the resolution of the final solution because equation 2.19b is a delta form and the left hand side of this equation does not affect the solution as long as the iteration converges. Consequently, we obtain an inconsistent discretization system because of the inconsistent discretization accuracy on the left hand side and the right hand side of equation 2.19b.

The convergence of the approximate factorization is not only slowed down by the inconsistent discretization, but also by the CFL limitation. Approximate factorization

introduces an optimum CFL beyond which the convergence rate slows down. The optimum CFL forces us to use steady preconditioning in Γ_1 and Γ_2 to ensure all errors decay in the same order of speed. But the steady preconditioning is much less robust than the physical Γ, Γ_p.

A way to get around these two dilemmas is to solve the linear equation 2.19b exactly, which is introduced in the next subsection.

2.4.3 The Third Pseudo Time: Solving the Linear Equation

In this subsection, we solve equation 2.19b by introducing a third pseudo time. We first define the new variable $\tilde{Q}_p = Q_p^{n+1}$ and add another pseudo time t_3 to solve equation 2.19b for $\tilde{Q}_p$. Upon adding the third pseudo time, we have

$$\Gamma_3 \frac{\partial \tilde{Q}_p}{\partial t_3} + \frac{\Gamma_2}{\Delta t_2}(\tilde{Q}_p - Q_p^n) + \left(\nabla_D \cdot A_p\right)_2^n (\tilde{Q}_p - Q_p^n) = -(\nabla_D \cdot F)^n$$

In the limit as Δt_3 goes to infinity, the last equation reduces to the desired solution of equation 2.19b.

To solve the last equation, we discretize it in pseudo time t_3. Using an Euler implicit method and writing in delta form, we obtain,

$$\left[\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2}\right) + \left(\nabla_D \cdot A_p\right)_3\right]^n \Delta\tilde{Q}_p = -(\nabla_D \cdot F)^n - \left[\frac{\Gamma_2}{\Delta t_2} + \left(\nabla_D \cdot A_p\right)_2\right]^n (\tilde{Q}_p^m - Q_p^n) \tag{2.21a}$$

Multiplying equation 2.21a by Δt_3, we obtain

$$\left[\left(\Gamma_3 + \frac{\Delta t_3}{\Delta t_2}\Gamma_2\right) + \Delta t_3\left(\nabla_D \cdot A_p\right)_3\right]^n \Delta\tilde{Q}_p = -\Delta t_3(\nabla_D \cdot F)^n - \left[\frac{\Delta t_3}{\Delta t_2}\Gamma_2 + \Delta t_3\left(\nabla_D \cdot A_p\right)_2\right]^n (\tilde{Q}_p^m - Q_p^n) \tag{2.21b}$$

where $\Delta\tilde{Q}_p = \tilde{Q}_p^{m+1} - \tilde{Q}_p^m$, and the superscript 'm' indicates the t_3 time level. The term, $\Gamma_2 \Delta t_3 / \Delta t_2$ acts as a sink term, which is helpful for the convergence. The subscript '3'

on $(\nabla_D \cdot A_p)_3$ indicates the discretized Jacobian divergence, $(\nabla_D \cdot A_p)$ at the t_3 time level. $(\nabla_D \cdot A_p)_3$ is traditionally discretized as first order while $(\nabla_D \cdot A_p)_2$ can be discretized as any order of accuracy in an approximate factorization solver.

For later convenience, we define the triple time residual at the right hand side of equation 2.21a as

$$R_3 = (\nabla_D \cdot F)^n + \left[\frac{\Gamma_2}{\Delta t_2} + (\nabla_D \cdot A_p)_2^n\right](\tilde{Q}_p^m - Q_p^n) \tag{2.22a}$$

and the left hand side operator as

$$L_3 = \left[\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + (\nabla_D \cdot A_p)_3\right] \tag{2.22b}$$

where the subscript '3' on the operators, 'L' and 'R', indicates that it is the operator corresponding to the third pseudo time. Consequently equation 2.21a can alternately be written as

$$L_3 \Delta\tilde{Q}_p = -R_3 \tag{2.23}$$

The triple time formulation is not finished until the three Γ's are determined. Obviously the three Γ's are very important and not arbitrary. We delay their definition to the next chapter, and for now, simply use them as parameters.

For convenience, we will refer to the convergence of equation 2.21a as the 'inner' convergence, as compared with the convergence of equation 2.19b, which we will call the 'outer' convergence.

The direct solution of Equation 2.21a is expensive for the same reason as solving equation 2.19b. Accordingly, we use an approximate factorization method to solve equation 2.21a. In this thesis, we use the diagonally dominant line Gauss-Sediel (DDLGS) method [8,9]. Since all Jacobian matrices are at time level 'n', equation 2.21a is a completely linear equation and therefore easier to solve than equation 2.19b. Also, we can store the LU decomposition of each tri-diagonal solver in the DDLGS method to minimize CPU time.

2.5 Green-Gauss Reconstruction

Before introducing the DDLGS approximate factorization method, we discuss the reconstruction procedure for the higher order scheme. For the first order discretization, we simply set the variable value at a face equal to the corresponding cell [27], which will introduce a large error. For a higher order system, we use some reconstruction procedure to compute the variable value at the face from the variable values at the surrounding cells [27] and consequently make the scheme more accurate than the first order. The recontruction methods compute the face value by a Taylor series expansion as

$$Q_p = Q_{pi,j} + \left(\nabla Q_p\right)_{ij} \cdot \Delta\vec{r} + \cdots \tag{2.24}$$

where $\Delta\vec{r}$ is the distance vector from the centroid of the cell 'i,j' to the face and the gradient term $\left(\nabla Q_p\right)_{ij}$ will be computed by the least square method [27] or Green-Gauss method [27]. For a structured rectangular grid, it can be shown that these two methods give the same result. Hence, only the Green-Gauss method is presented.

Consider an equal-spaced rectangular grid with size Δx in the x-direction and Δy in the y-direction in figure 2.1a. 'L' and 'R' in this figure are the labels used to identify the two sides of a face and the numbers, 1, 2, 3 and 4, are the face numbers. The goal of the Green-Gauss method is to compute the gradient at the cell center 'i,j' from the values of the surrounding cells. Therefore, we construct the Green-Gauss control volume for cell 'i,j' by connecting the centroids of the surrounding cells as shown in figure 2.1b. The volume of the Green-Gauss control volume, Ω', is then clearly $2\Delta x\Delta y$. Note that here we use a prime to distinguish the control volume of Green-Gauss from the control volume of discretization.

The average gradient of this control volume is,

$$\nabla\overline{Q}_{pi,j} = \frac{1}{\Omega'}\int_{\Omega'}\nabla Q_p\, d\Omega' = \frac{1}{\Omega'}\int_{\Omega'} Q_p\vec{n}\, dS = \sum_{k=1}^{K} Q_{pk}\,\vec{n}_k S_k \tag{2.25a}$$

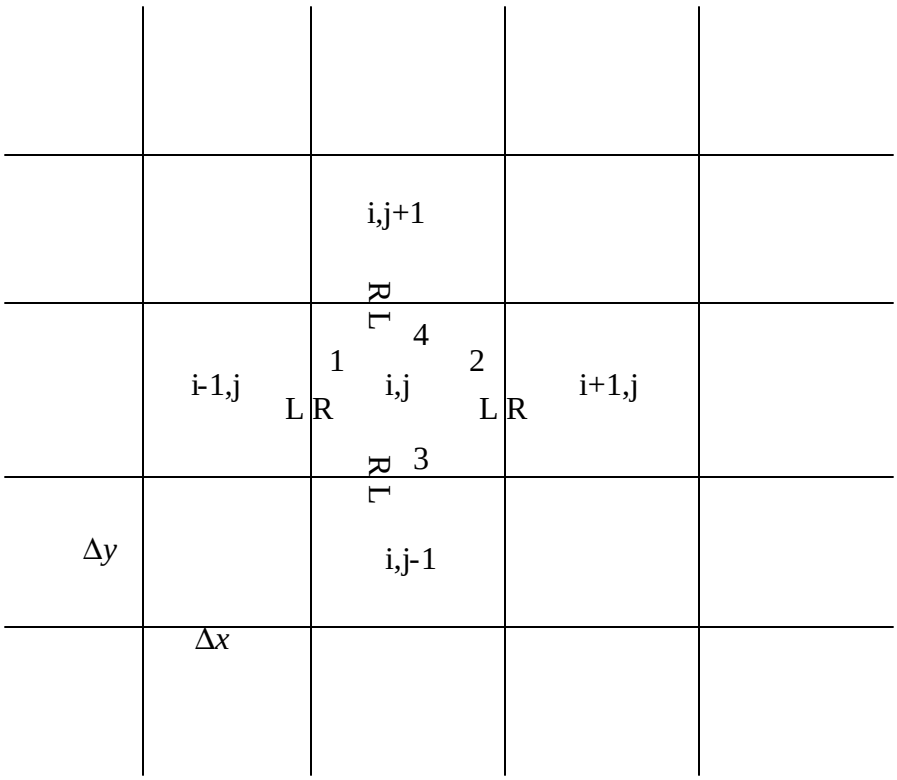

Figure 2.1a Equal-spaced rectangular grids

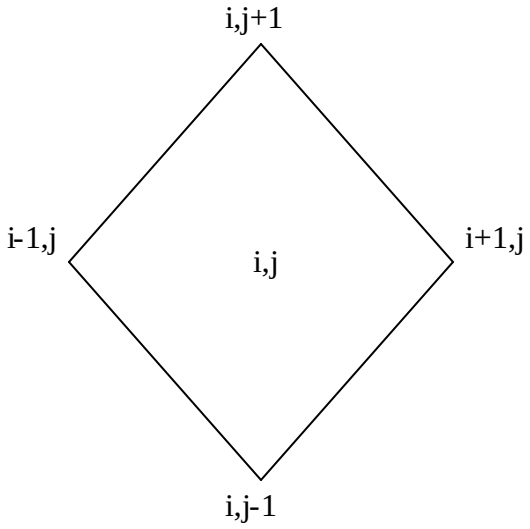

Figure 2.1b The Green-Gauss control volume for cell 'i,j' in figure 2.1a

In equation 2.25a, we have applied the Green-Gauss theorem to transform the volume integral to the face integral.

Evaluate each face value, Q_{pk}, by the average of the two end points to obtain,

$$\nabla\overline{Q}_{pi,j} = \frac{1}{2\Delta x\Delta y}\left[\frac{Q_{pi-1,j}+Q_{pi,j+1}}{2}\left(-\Delta y\vec{i}+\Delta x\vec{j}\right)+\frac{Q_{pi,j-1}+Q_{pi+1,j}}{2}\left(\Delta y\vec{i}-\Delta x\vec{j}\right)+\right.$$
$$\left.\frac{Q_{pi-1,j}+Q_{pi,j-1}}{2}\left(-\Delta y\vec{i}-\Delta x\vec{j}\right)+\frac{Q_{pi,j+1}+Q_{pi+1,j}}{2}\left(\Delta y\vec{i}+\Delta x\vec{j}\right)\right] \quad (2.25b)$$
$$=\frac{1}{2\Delta x\Delta y}\left[\left(Q_{pi+1,j}-Q_{pi-1,j}\right)\Delta y\vec{i}+\left(Q_{pi,j+1}-Q_{pi,j-1}\right)\Delta x\vec{j}\right]$$

The distance vectors from the center of cell 'i,j' to the four faces of the cell are the following

$$\Delta\vec{r}_{1R} = -\frac{\Delta x}{2}\vec{i}, \quad (2.26a)$$

$$\Delta\vec{r}_{3R} = -\frac{\Delta y}{2}\vec{j}, \quad (2.26b)$$

$$\Delta\vec{r}_{2L} = \frac{\Delta x}{2}\vec{i}, \quad (2.26c)$$

$$\Delta\vec{r}_{4L} = \frac{\Delta y}{2}\vec{j} \quad (2.26d)$$

where the numbered subscripts are the face numbers, and 'L' and 'R' indicate the sides of the face.

Perform the dot product of the average gradient in equation 2.25b and the distance vectors to get

$$\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{1R} = -\frac{1}{4}\left(Q_{p\,i+1,j}-Q_{p\,i-1,j}\right) \quad (2.27a)$$

$$\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{3R} = -\frac{1}{4}\left(Q_{p\,i,j+1}-Q_{pi,j-1}\right) \quad (2.27b)$$

$$\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{2L} = \frac{1}{4}\left(Q_{pi+1,j}-Q_{pi-1,j}\right) \quad (2.27c)$$

$$\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{4L} = \frac{1}{4}\left(Q_{pi,j+1}-Q_{pi,j-1}\right) \quad (2.27d)$$

where the subscript ‘1R’ indicates the right side of face 1, and etc.. By the same token, we can obtain

$$\left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{1L} = \frac{1}{4}\left(Q_{pi,j} - Q_{pi-2,j}\right) \tag{2.28a}$$

$$\left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{3L} = \frac{1}{4}\left(Q_{pi,j} - Q_{pi,j-2}\right) \tag{2.28b}$$

$$\left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{2R} = -\frac{1}{4}\left(Q_{pi+2,j} - Q_{pi,j}\right) \tag{2.28c}$$

$$\left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{4R} = -\frac{1}{4}\left(Q_{pi,j+2} - Q_{pi,j}\right) \tag{2.28d}$$

Substituting equation sets 2.27 and 2.28 into equation 2.24 and ignoring the higher order terms than the gradient, we obtain all face values at the surrounding faces of cell ‘i,j’ as

$$\left(Q_p\right)_{1R} = Q_{pij} + \left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{1R} = Q_{pij} - \frac{1}{4}\left(Q_{p\,i+1,j} - Q_{p\,i-1,j}\right) \tag{2.29a}$$

$$\vdots$$

$$\left(Q_p\right)_{4R} = Q_{pi,j+1} + \left(\nabla \overline{Q}_p \cdot \Delta \vec{r}\right)_{4R} = Q_{pi,j+1} - \frac{1}{4}\left(Q_{p\,i,j+2} - Q_{p\,i,j}\right) \tag{2.29h}$$

Equation set 2.29 gives a second order accurate system by using the Green-Gauss reconstruction.

2.6 Four-Sweep DDLGS Approximate Factorization

Now, we discuss the DDLGS approximate factorization method. The line Gauss-Sediel (LGS) approximate factorization method is a popular approximate method for solving linear equations. There are several variants for this method, such as the diagonally-dominant line Gauss-Sediel (DDLGS) method, the non-diagonally-dominant line Gauss-Sediel (LGS) method, the two sweep line Gauss-Sediel method, and the four sweep line Gauss-Sediel method [8,9], and etc.

In this thesis we use the four sweep DDLGS approximate factorization method to solve equation 2.21a because of its faster convergence compared with the non-diagonally-dominant line Gauss-Sediel (LGS) method [8]. DDLGS method was

originally developed for structured grids, but it can also be extended to the unstructured grids [24]. For unstructured grids it is difficult to write down and to be understood. Therefore, only a rectangular grid is used to illustrate how the DDLGS approximate factorization method works. The four sweep LGS approximation factorization for unstructured grid follows that similar way [24]. For simplicity, only the two-sweep DDLGS with both forward and backward sweeps in the X-direction is derived in detail. The results for four sweep DDLGS are then given without derivation but their derivation follows in a similar way as the two sweep DDLGS.

Consider a two dimensional, steady flow and computational grids as shown in figure 2.1. We begin by writing the first order expression for the discretized Jacobian divergence operator, $\nabla_D \cdot A_p$, in terms of the cell values. In the $\nabla_D \cdot A_p$ expression of equation 2.20b, $\nabla_D \cdot A_p$ is expressed in terms of the face values. Since for the first order the "left" and "right" face values equal the corresponding cell values in figure 2.1, equation 2.20b becomes

$$
\begin{aligned}
(\nabla_D \cdot A_p)_I = & -\frac{1}{2}\frac{1}{\Delta x}\left[A_{pi}\bullet + A_{pi-1}\bullet - \left(\Gamma_1\left|\Gamma_1^{-1}A_p\right|_i \bullet - \Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i-1}\bullet\right)\right] \\
& + \frac{1}{2}\frac{1}{\Delta x}\left[A_{pi}\bullet + A_{pi+1}\bullet - \left(\Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i+1}\bullet - \Gamma_1\left|\Gamma_1^{-1}A_p\right|_i \bullet\right)\right] \\
& - \frac{1}{2}\frac{1}{\Delta y}\left[B_{pj}\bullet + B_{pj-1}\bullet - \left(\Gamma_1\left|\Gamma_1^{-1}B_p\right|_j \bullet - \Gamma_1\left|\Gamma_1^{-1}B_p\right|_{j-1}\bullet\right)\right] \\
& + \frac{1}{2}\frac{1}{\Delta y}\left[B_{pj}\bullet + B_{pj+1}\bullet - \left(\Gamma_1\left|\Gamma_1^{-1}B_p\right|_{j+1}\bullet - \Gamma_1\left|\Gamma_1^{-1}B_p\right|_j \bullet\right)\right]
\end{aligned}
$$

where the subscript 'I' on $\left(\nabla_D \cdot A_p\right)_I$ indicates the first order upwind.

This equation can be simplified by putting all terms at the same grid point together. It becomes

$$
\begin{aligned}
\left(\nabla_D \cdot A_p\right)_I = & \left(\frac{1}{\Delta x}\Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i,j}\bullet + \frac{1}{\Delta y}\Gamma_1\left|\Gamma_1^{-1}B_p\right|_{ij}\bullet\right) \\
& + \frac{1}{2}\frac{1}{\Delta x}\left(A_{pi+1}\bullet - \Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i+1}\bullet\right) - \frac{1}{2}\frac{1}{\Delta x}\left(A_{pi-1}\bullet + \Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i-1}\bullet\right) \\
& + \frac{1}{2}\frac{1}{\Delta y}\left(B_{pj+1}\bullet - \Gamma_1\left|\Gamma_1^{-1}B_p\right|_{j+1}\bullet\right) - \frac{1}{2}\frac{1}{\Delta y}\left(B_{pj-1}\bullet + \Gamma_1\left|\Gamma_1^{-1}B_p\right|_{j-1}\bullet\right) \qquad (2.30)
\end{aligned}
$$

To simplify the notation, we define the coefficients of ΔQ_p at all five grid points as

$$J_{ij} = \frac{1}{\Delta x}\Gamma_1 \left|\Gamma_1^{-1} A_p\right|_{i,j} \bullet + \frac{1}{\Delta y}\Gamma_1 \left|\Gamma_1^{-1} B_p\right|_{ij} \bullet \tag{2.31a}$$

$$J_{i+1,j} = \frac{1}{2}\frac{1}{\Delta x}\left(A_{pi+1,j} \bullet - \Gamma_1 \left|\Gamma_1^{-1} A_p\right|_{i+1,j} \bullet\right) \tag{2.31b}$$

$$J_{i-1,j} = -\frac{1}{2}\frac{1}{\Delta x}\left(A_{pi-1,j} \bullet + \Gamma_1 \left|\Gamma_1^{-1} A_p\right|_{i-1,j} \bullet\right) \tag{2.31c}$$

$$J_{i,j+1} = \frac{1}{2}\frac{1}{\Delta y}\left(B_{pi,j+1} \bullet - \Gamma_1 \left|\Gamma_1^{-1} B_p\right|_{i,j+1} \bullet\right) \tag{2.31d}$$

$$J_{i,j-1} = -\frac{1}{2}\frac{1}{\Delta y}\left(B_{pi,j-1} \bullet + \Gamma_1 \left|\Gamma_1^{-1} B_p\right|_{i,j-1} \bullet\right) \tag{2.31e}$$

Substituting equation set 2.31 into equation 2.30, gives

$$(\nabla_D \cdot A_p)_I = J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{2.32a}$$

For the first order discretization of $\left(\nabla_D \cdot A_p\right)_3$, we obtain the Jacobian divergence operator for the t_3 time level to be

$$(\nabla_D \cdot A_p)_3 = (\nabla_D \cdot A_p)_I = J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{2.32b}$$

Substituting equations 2.32b and 2.22a into equation 2.21a, we can re-write equation 2.21a as

$$\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.33}$$

And the left hand side operator L_3 in equation 2.22b becomes

$$L_3 = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{2.34}$$

To solve equation 2.33, we approximately factor it into the matrices that can be solved simply. Following the sketch in figure 2.2, we designate all points on the line i=constant (j=1,J) as an 'x-diagonal' term, D_x, that is given by

$$D_x = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i,j+1} + J_{i,j-1} \tag{2.35}$$

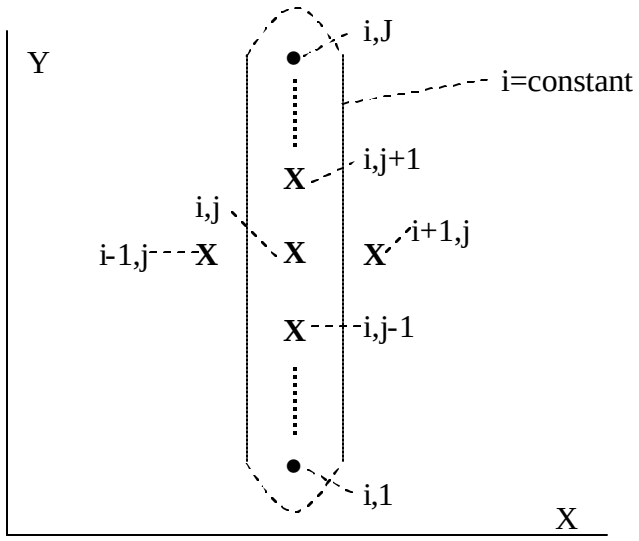

Figure 2.2 Definition of the 'X-diagonal' operator

Note that all terms in D_x lie on the line i=constant.

Using equation 2.35, equation 2.33 becomes

$$\left(D_x + J_{i+1,j} + J_{i-1,j}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.36}$$

A simple 'approximate factorization' of the whole Jacobian operator on the left hand side is

$$\begin{aligned} D_x + J_{i+1,j} + J_{i-1,j} &= D_x\left(I + D_x^{-1}J_{i+1,j} + D_x^{-1}J_{i-1,j}\right) \\ &= D_x\left(I + D_x^{-1}J_{i-1,j}\right)\left(I + D_x^{-1}J_{i+1,j}\right) \end{aligned} \tag{2.37}$$

Where 'I' is an identity matrix. This factorization is clearly a good approximation to the original matrix, L_3 in equation 2.34, if the quantities, $D_x^{-1}J_{i-1,j}$ and $D_x^{-1}J_{i+1,j}$, are small with respect to the identity matrix (without going into detail we note that these two terms are small for small enough values of Δt_3 since D_x^{-1} goes to zero as Δt_3 goes to zero).

Upon symmetrizing equation 2.37 and substituting into equation 2.36, we have the approximately factorized version of equation 2.36

$$\left(D_x + J_{i-1,j}\right)D_x^{-1}\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.38}$$

Reference to the definition of D_x in equation 2.35 shows that each of the three operators in equation 2.38 is a tri-diagonal operator that can be solved efficiently. Equation 2.38 can be solved by a series of sweeps. If we replace the last two operators in equation 2.38 by the interim vector $\Delta\tilde{Q}_p{}^*$ as

$$D_x^{-1}\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p = \Delta\tilde{Q}_p{}^* \tag{2.39a}$$

Equation 2.38 becomes

$$\left(D_x + J_{i-1,j}\right)\Delta\tilde{Q}_p{}^* = -R_3 \tag{2.39b}$$

This is a tri-diagonal linear equation and can easily be solved for $\Delta\tilde{Q}_p{}^*$ for the first step. We then introduce a second interim variable, $\Delta\tilde{Q}_p{}^{**}$ as

$$\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p = \Delta\tilde{Q}_p{}^{**} \tag{2.39c}$$

and by combining equation 2.39a and equation 2.39c, we have

$$D_x^{-1}\Delta\tilde{Q}_p{}^{**} = \Delta\tilde{Q}_p{}^* \qquad \text{or} \qquad \Delta\tilde{Q}_p{}^{**} = D_x\Delta\tilde{Q}_p{}^* \tag{2.39d}$$

which shows we can compute $\Delta\tilde{Q}_p{}^{**}$ from the unknown vector $\Delta\tilde{Q}_p{}^*$ by a matrix multiply.

Finally, we can solve for $\Delta\tilde{Q}_p$ from equation 2.39c by a second tri-diagonal matrix.

Equations 2.39c and 2.39d could be combined to be one equation as

$$\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p = D_x\Delta\tilde{Q}_p{}^* \tag{2.39e}$$

From figure 2.3 we can see that starting from the left, we can march across the entire field solving for $\Delta\tilde{Q}_p{}^*$ (and $\Delta\tilde{Q}_p{}^{**}$) on each i-line from i=1 to I using equations 2.39b. Then we can sweep from i=I to 1 to solve for $\Delta\tilde{Q}_p$ on each i-line using equation 2.39e, thus completing one update of the approximate factorization solution.

Upon multiplying out the approximately factored expression in equation 2.38, we see that the actual equation we are solving is

$$\left(D_x + J_{i-1,j} + J_{i+1,j} + J_{i-1,j}D_x^{-1}J_{i+1,j}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.40a}$$

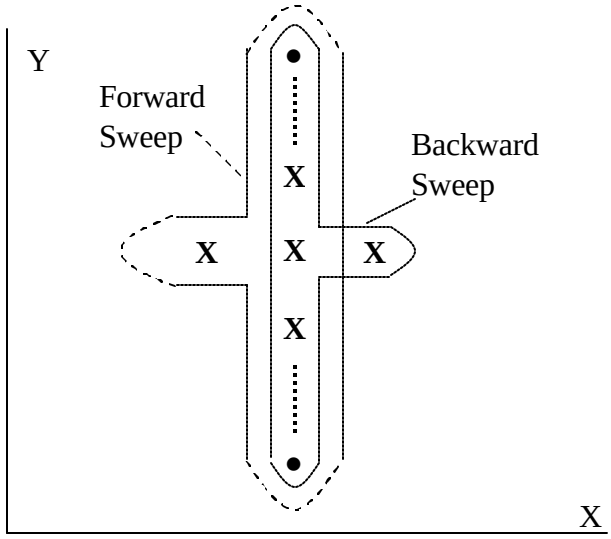

Figure 2.3 The forward sweep and backward sweep of two-sweep LGS

Thus, instead of solving the exact equation which is shown in equation 2.33 we are solving an approximate equation. Using the definition of the operator L_3 in equation 2.34, and equation 2.35 we can write equation 2.40a as

$$\left(L_3 + Err_{2X}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.40b}$$

where the error of the approximate factorization, Err_{2X} (the subscript '2X' indicates the error of the two-sweep DDLGS in the X-direction), is

$$Err_{2X} = J_{i-1,j} D_x^{-1} J_{i+1,j} \tag{2.41}$$

It is also useful to re-write the approximately factored version, equations 2.39b and e in an alternate form. By comparing equations 2.34 and 2.35, we see that the operator in equation 2.39b can be written as the equivalent expression in term of the exact operator, L_3, minus a 'corrector' as

$$D_x + J_{i-1,j} = L_3 - J_{i+1,j}$$

Substituting this relationship into equation 2.39b, we obtain

$$\left(L_3 - J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.42a}$$

Similarly, we can write equation 2.39e as

$$\left(L_3 - J_{i-1,j}\right)\Delta\tilde{Q}_p = -R_3 - J_{i-1,j}\Delta\tilde{Q}_p^{\,*} \tag{2.42b}$$

Equations 2.39b and e or equations 2.42a and b, give the identical equations for the two-sweep DDLGS approximate factorization method. Although the equation set 2.42 appears to be an arbitrary replacement for the exact operator, the more logical approximate factorization approach shows that it is factored upon a rational analysis. Equation 2.39b or 2.42a is the forward sweep in the x-direction and equation 2.39e or equation 2.42b is the backward sweep in the x-direction.

Investigation of equations 2.33, 2.35 and 2.37 immediately suggests we could do a similar two-sweep factorization in the y-direction by defining

$$D_y = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} \tag{2.43}$$

This 'diagonal' operator is clearly a tri-diagonal operator along a j=constant line. Accordingly we could express equation 2.33 as

$$\left(D_y + J_{i,j+1} + J_{i,j-1}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.44}$$

and do a similar factorization in the y-direction as we did in equation 2.37.

By combining these two two-sweep operators together we can after substantial algebra obtain a four-sweep approximate factorization (see appendix B) as

$$\left(D_x + J_{i-1,j}\right)D_x^{-1}\left(D_x + J_{i+1,j}\right)D^{-1}\left(D_y + J_{i,j-1}\right)D_y^{-1}\left(D_y + J_{i,j+1}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.45}$$

where the new diagonal term, D, is

$$D = L_3 + J_{i-1,j}D_x^{-1}J_{i+1,j} + J_{i,j-1}D_y^{-1}J_{i,j+1} \tag{2.46}$$

Again, we expand the operator at the left hand side of equation 2.45 and write equation 2.45 as the form of the exact operator plus an error as the following (see appendix B),

$$\left(L_3 + Err_{4S}\right)\Delta\tilde{Q}_p = -R_3 \tag{2.47}$$

where the subscript '4S' on the error of the approximate factorization, Err_{4S}, indicates the four-sweep DDLGS. Err_{4S} is

$$Err_{4S} = J_{i,j-1}D_y^{-1}J_{i,j+1}D^{-1}J_{i-1,j}D_x^{-1}J_{i+1,j} \tag{2.48}$$

Also, this equation can be written in four steps, which correspond to the four sweeps in the four-sweep DDLGS. The derivation is given in appendix B and not presented here for complexity and only the results are given in equation set 2.49 and 2.50.

$$\left(D_x + J_{i-1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.49a}$$

$$\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,**} = D_x\Delta\tilde{Q}_p^{\,*} \tag{2.49b}$$

$$\left(D_y + J_{i,j-1}\right)\Delta\tilde{Q}_p^{\,***} = D\Delta\tilde{Q}_p^{\,**} \tag{2.49c}$$

$$\left(D_y + J_{i,j+1}\right)\Delta\tilde{Q}_p = D_y\Delta\tilde{Q}_p^{\,***} \tag{2.49d}$$

Again, we can write equations 2.49a, b, c and d as the equivalent expression in term of the exact operator, L_3, minus a 'corrector' as

$$\left(L_3 - J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.50a}$$

$$\left(L_3 - J_{i-1,j}\right)\Delta\tilde{Q}_p^{\,**} = -R_3 - J_{i-1,j}\Delta\tilde{Q}_p^{\,*} \tag{2.50b}$$

$$\left(L_3 - J_{i,j+1}\right)\Delta\tilde{Q}_p^{\,***} = -R_3 - J_{i,j+1}\Delta\tilde{Q}_p^{\,**} \tag{2.50c}$$

$$\left(L_3 - J_{i,j-1}\right)\Delta\tilde{Q}_p = -R_3 - J_{i,j-1}\Delta\tilde{Q}_p^{\,***} \tag{2.50d}$$

The equation set 2.50a,b, c and d or equation set 2.49 is the four-sweep LGS approximation method for two dimensions. Equation 2.50a or 2.49a is the forward sweep in the x-direction; Equation 2.50b or 2.49b is the backward sweep in the x-direction; Equation 2.50c or 2.49c is the forward sweep in the y-direction; Equation 2.50d or 2.49d is the backward sweep in the y-direction. Every iteration step includes all four sweeps. Since every equation in equation set 2.50 or 2.49 is a tri-diagonal linear equation, it can be solved very easily.

Equation set 2.50 is to remember and code than equation set 2.49. But equation set 2.49 is very useful for the unstanding.

A similar six-sweep approximate factorization scheme can be written for three dimensions.

2.7 Summary and Implementation for the Triple Time Method

For convenience, we summarize the triple time formulation for the four-sweep DDLGS approximate factorization (rectangular grids) here. The triple time equations are equation set 2.49 and are repeated here as equation set 2.51.

$$\left(D_x + J_{i-1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.51a}$$

$$\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,**} = D_x\Delta\tilde{Q}_p^{\,*} \tag{2.51b}$$

$$\left(D_y + J_{i,j-1}\right)\Delta\tilde{Q}_p^{\,***} = D\Delta\tilde{Q}_p^{\,**} \tag{2.51c}$$

$$\left(D_y + J_{i,j+1}\right)\Delta\tilde{Q}_p = D_y\Delta\tilde{Q}_p^{\,***} \tag{2.51d}$$

where D_x, D_y and D are defined in equations 2.35, 2.43 and 2.46, and are repeated here as equations 2.52a, b and c respectively

$$D_x = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i,j+1} + J_{i,j-1} \tag{2.52a}$$

$$D_y = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} \tag{2.52b}$$

$$D = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} + J_{i-1,j}D_x^{-1}J_{i+1,j} + J_{i,j-1}D_y^{-1}J_{i,j+1} \tag{2.52c}$$

To obtain equation 2.52c, we have substituted equation 2.34 into equation 2.46.

Where the Jacobian matrices, J's, in equation set 2.52 are given by equation set 2.31 and are repeated here as equation set 2.53

$$J_{ij} = \frac{1}{\Delta x}\Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i,j} \bullet + \frac{1}{\Delta y}\Gamma_1\left|\Gamma_1^{-1}B_p\right|_{ij} \bullet \tag{2.53a}$$

$$J_{i+1,j} = \frac{1}{2}\frac{1}{\Delta x}\left(A_{pi+1,j} \bullet -\Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i+1,j} \bullet\right) \tag{2.53b}$$

$$J_{i-1,j} = -\frac{1}{2}\frac{1}{\Delta x}\left(A_{pi-1,j} \bullet +\Gamma_1\left|\Gamma_1^{-1}A_p\right|_{i-1,j} \bullet\right) \tag{2.53c}$$

$$J_{i,j+1} = \frac{1}{2}\frac{1}{\Delta y}\left(B_{pi,j+1} \bullet - \Gamma_1 \left|\Gamma_1^{-1} B_p\right|_{i,j+1} \bullet\right) \tag{2.53d}$$

$$J_{i,j-1} = -\frac{1}{2}\frac{1}{\Delta y}\left(B_{pi,j-1} \bullet + \Gamma_1 \left|\Gamma_1^{-1} B_p\right|_{i,j-1} \bullet\right) \tag{2.53e}$$

And the triple time residual is defined in equation 2.22a and is repeated here as equation 2.54

$$R_3 = \left(\nabla_D \cdot F\right)^n + \left[\frac{\Gamma_2}{\Delta t_2} + \left(\nabla_D \cdot A_p\right)_2^n\right]\left(\tilde{Q}_p^m - Q_p^n\right) \tag{2.54}$$

The flux divergence is given by equation 2.15b and is repeated here as equation 2.55

$$\left(\nabla_D \bullet F\right)^n = \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(F_L + F_R\right) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1} A_p\right|\left(Q_{pR} - Q_{pL}\right)\right]_k^n S_k \tag{2.55}$$

and the Jacobian divergence is given by equation 2.20b and is repeated here as equation 2.56

$$\left(\nabla_D \cdot A_p\right)^n = \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL} \bullet_L + A_{pR} \bullet_R\right) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1} A_p\right|\left(\bullet_R - \bullet_L\right)\right]_k^n S_k \tag{2.56}$$

The following is the procedure to implement the triple time method, we start from the solution Q_p^n at the old time step

1. Compute values of Q_L and Q_R at all surrounding faces of each cell from equation set 2.29.
2. Compute the fluxes F_L and F_R, and the Jacobian matrices Γ_1 and $\Gamma_1^{-1} A_p$ from Q_L and Q_R.
3. Calculate the residual in t_2, $\nabla_D \cdot F$ from equation 2.55.
4. Compute the discretized divergence operator $\nabla_D \cdot A_p$ from equation 2.56.
5. Compute the matrix coefficients J's from equation set 2.53 for all cells using Γ_1 and $\Gamma_1^{-1} A_p$ from step 2.

6. Using the last solution $\tilde{Q}_p{}^m$ (or the initial condition of $\tilde{Q}_p{}^0 = Q_p{}^n$) and the divergence operator from steps 3 and 4, compute the triple time residual R_3 from equation 2.54.
7. Compute a new update $\Delta\tilde{Q}_p$ from the DDLGS systems of equation 2.51 using the definitions in equations 2.52a and b.
8. Compute the new solution as $\tilde{Q}_p{}^{m+1} = \tilde{Q}_p{}^m + \Delta\tilde{Q}_p$ and repeat steps 6 through 8 until the Mth step or $\Delta\tilde{Q}_p$ is less than some specified tolerance.
9. Set $Q_p{}^{n+1} = \tilde{Q}_p{}^m$ to complete the time step.
10. Repeat steps 1 through 9 until the error $Q_p{}^{n+1} - Q_p{}^n$ less than some specified tolerance or running out of the specified number of iterations to finish the whole iteration.
11. Obtain the final solution

2.8 The Degeneration of Triple Time to Single Time

The triple time formulation can be degenerated to the single time method [18,24,42] and dual time method by appropriate simplifications. Thus these more familiar systems represent subsets of the more general triple time system. In the following, we will show how the triple time scheme degenerates to the single time and dual time schemes.

To degenerate the triple time scheme to the single time scheme, we first set $\Gamma_1 = \Gamma_2$ in the triple time equation 2.51a, b, c and d, so the artificial dissipation is determined by the time marching procedure, then use only one inner iteration ($\tilde{Q}_p{}^1 = Q_p{}^{n+1}$), and use the previous outer iteration solution as the initial condition for $\tilde{Q}_p$ ($\tilde{Q}_p{}^0 = Q_p{}^n$), then

$$\Delta\tilde{Q}_p = \tilde{Q}_p^1 - \tilde{Q}_p^0 = Q_p^{n+1} - Q_p^n = \Delta Q_p \tag{2.57}$$

Using the assumption that $\tilde{Q}_p{}^0 = Q_p{}^n$, the second term on the right side of equation 2.54 vanishes, so that the residual of the third pseudo time R_3 becomes equal to the second pseudo time residual R_2 (see equation 2.22a), obtain

$$R_3 = \left(\nabla_D \cdot F\right)^n = R_2 \tag{2.58}$$

Setting Δt_3 to infinity in equations 2.52a, b and c, and re-defining the 'x-diagonal' term as D_x', the 'y-diagonal' term as D_y' and D as D', we have

$$D_x' = \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i,j+1} + J_{i,j-1} \tag{2.59a}$$

$$D_y' = \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} \tag{2.59b}$$

$$D' = \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} + J_{i-1,j} D_x^{-1} J_{i+1,j} + J_{i,j-1} D_y^{-1} J_{i,j+1} \tag{2.59c}$$

Substituting equation 2.58, 2.59a, b and c into equation set 2.41, and using $\Delta Q_p{}^*$, $\Delta Q_p{}^{**}$ and $\Delta Q_p{}^{***}$ to replace $\Delta\tilde{Q}_p{}^*$, $\Delta\tilde{Q}_p{}^{**}$ and $\Delta\tilde{Q}_p{}^{***}$ to indicate that it is for the dual time instead of triple time, we obtain

$$\left(D_x' + J_{i-1,j}\right)\Delta Q_p{}^* = -R_2 \tag{2.60a}$$

$$\left(D_x' + J_{i+1,j}\right)\Delta Q_p{}^{**} = D_x'\,\Delta Q_p{}^* \tag{2.60b}$$

$$\left(D_y' + J_{i,j-1}\right)\Delta Q_p{}^{***} = D'\Delta Q_p{}^{**} \tag{2.60c}$$

$$\left(D_y' + J_{i,j+1}\right)\Delta Q_p = D_y'\,\Delta Q_p{}^{***} \tag{2.60d}$$

We call equation set 2.60 the single time scheme because the third pseudo time vanishes and only one pseudo time exists. A variation of the single time method would be to leave Γ_1 different from Γ_2 so that the preconditioning for the time marching is different from the artificial dissipation and consequently provide us more freedom to implement the scheme. This could be considered as a 'dual' time.

In summary, the triple time scheme is a general scheme and contains the single time scheme as a special case. If we use only one inner iteration, use the previous outer solution as the initial condition of inner iteration and set Δt_3 to be infinity in the triple time equation, we get a new single time formulation in which the preconditioning in time marching is different from the preconditioning in artificial dissipation. If we further set time marching Γ_2 equal to the artificial dissipation Γ_1, we obtain the traditional single time scheme.

2.9 The Degeneration of the Triple Time Method to the Multiple DDLGS Iteration Method

Besides the triple time method, there are other iteration methods to solve equation 2.19b. One typical way is the multiple DDLGS iteration method [24]. We prove in this subsection that the multiple DDLGS iteration method is just a special case of the triple time method.

This time we use equation set 2.50 instead of equation set 2.51 because equation set 2.50 is more similar to the multiple DDLGS iteration method. We start with the first sweep of the four-sweep DDLGS approximate solver, equation 2.50a (re-labeled as 2.61),

$$\left(L_3 - J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.61}$$

We assume that the Jacobian divergence operator $\left(\nabla_D \cdot A_p\right)_2$ in R_3 is discretized to be the first order (see equation 2.32a) so that

$$\left(\nabla_D \cdot A_p\right)_2 = \left(\nabla_D \cdot A_p\right)_I = J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{2.62}$$

Letting Δt_3 go to infinity in equation 2.34 and re-defining the exact operator L_3 as L_2 for the vanish of the third pseudo time, we obtain

$$L_3 = L_2 = \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{2.63}$$

Substituting equation 2.62 into equations 2.54 and applying equation 2.63, we obtain

$$R_3 = (\nabla_D \cdot F)^n + \left(\frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \right) (\tilde{Q}_p^m - Q_p^n)$$
$$= R_2 + L_2 (\tilde{Q}_p^m - Q_p^n) \tag{2.64}$$

Substituting equations 2.63 and 2.64 into equation 2.61, after arrangement we obtain

$$(L_2 - J_{i+1,j}) \Delta \tilde{Q}_p^{\,*} = -R_2 - L_2 (\tilde{Q}_p^m - Q_p^n) \tag{2.65}$$

Equation 2.65 can be re-arranged as

$$(L_2 - J_{i+1,j}) (\Delta \tilde{Q}_p^{\,*} + \tilde{Q}_p^m - Q_p^n) = -R_2 - J_{i+1,j} (\tilde{Q}_p^m - Q_p^n) \tag{2.66a}$$

Substituting equations 2.63 and 2.64 into the other three equations of DDLGS, 2.50b, c and d, after arrangement, we obtain

$$(L_2 - J_{i-1,j}) (\Delta \tilde{Q}_p^{\,**} + \tilde{Q}_p^m - Q_p^n) = -R_2 - J_{i-1,j} (\Delta \tilde{Q}_p^{\,*} + \tilde{Q}_p^m - Q_p^n) \tag{2.66b}$$

$$(L_2 - J_{i,j+1}) (\Delta \tilde{Q}_p^{\,***} + \tilde{Q}_p^m - Q_p^n) = -R_2 - J_{i,j+1} (\Delta \tilde{Q}_p^{\,**} + \tilde{Q}_p^m - Q_p^n) \tag{2.66c}$$

$$(L_2 - J_{i,j-1}) (\Delta \tilde{Q}_p + \tilde{Q}_p^m - Q_p^n) = -R_2 - J_{i,j-1} (\Delta \tilde{Q}_p^{\,****} + \tilde{Q}_p^m - Q_p^n) \tag{2.66d}$$

Obviously, we can re-define the intermediate variables as

$$\Delta Q_p^{\,*} = \Delta \tilde{Q}_p^{\,*} + \tilde{Q}_p^m - Q_p^n \tag{2.67a}$$

$$\Delta Q_p^{\,**} = \Delta \tilde{Q}_p^{\,**} + \tilde{Q}_p^m - Q_p^n \tag{2.67b}$$

$$\Delta Q_p^{\,***} = \Delta \tilde{Q}_p^{\,***} + \tilde{Q}_p^m - Q_p^n \tag{2.67c}$$

After substituting equation set 2.67 into equation set 2.66 and applying $\Delta \tilde{Q}_p = \tilde{Q}_p^{m+1} - \tilde{Q}_p^m$ to equation 2.66d, we obtain

$$(L_2 - J_{i+1,j}) \Delta Q_p^{\,*} = -R_2 - J_{i+1,j} (\tilde{Q}_p^{\,m} - Q_p^{\,n}) \tag{2.68a}$$

$$(L_2 - J_{i-1,j}) \Delta Q_p^{\,**} = -R_2 - J_{i-1,j} \Delta Q_p^{\,*} \tag{2.68b}$$

$$(L_2 - J_{i,j+1}) \Delta Q_p^{\,***} = -R_2 - J_{i,j+1} \Delta Q_p^{\,**} \tag{2.68c}$$

$$(L_2 - J_{i,j-1}) (\tilde{Q}_p^{\,m+1} - Q_p^{\,n}) = -R_2 - J_{i,j-1} \Delta Q_p^{\,***} \tag{2.68d}$$

Equation set 2.68a, b, c and d is the multiple DDLGS iteration method for the non-linear equation 2.19d, which is similar to the DDLGS approximate factorization method except we add the previous iteration solution at the right hand side of the equation 2.68a.

In summary, the multiple LGS iteration method is just a special case of triple time method, in which Δt_3 goes to infinity and $\left(\nabla_D \cdot A_p\right)_2$ is discretized to be first order. Results shown later indicate that infinity is not always the most optimum value for Δt_3. Further, using consistent discretization in the t_1 and t_2 operators (as opposed to using an inconsistent first order operator as in the iterative method) gives a substantial gain in efficiency as is shown later.

2.10 Boundary Conditions

2.10.1 General Formulation

The solution for any differential equation requires the specification of appropriate boundary conditions. What the boundary conditions are depends on the specific physical problem and should be physically realistic. For example, for the flow of air drawn from a large plenum into a channel, the flow is designed by specifying the total pressure and total temperature at the inlet of the channel, but for flow from a choked nozzle, the mass flow rate and the total temperature must be specified. Triple time boundary conditions are exactly the same as dual time or single time boundary conditions because the spatial derivatives in all these three methods are exactly the same.

Boundary conditions can be implemented explicitly or implicitly. The explicit boundary condition is straightforward and only the implicit one is presented. In this subsection, we will start with a specific example of a boundary condition procedure to develop a formulation that can be applied to arbitrary boundary conditions.

Consider the following cells,

g	b	b+1

Where the symbols ‘g’, ‘b’ and ‘b+1’ indicate the ghost cell, the boundary cell and the internal cell that is adjacent to the boundary cell. The ghost cell ‘g’ is artificially added for the convenience of treating the boundary cell in the same way as internal cells. We

force the ghost cell center instead of the boundary cell center to satisfy the boundary condition.

A typical subsonic inlet boundary condition is to specify the total enthalpy, h^0, the entropy, s, and the flow directions, and obtain the remaining boundary variable, the velocity magnitude from the internal flow field (the boundary cell in this thesis). For an ideal gas, the total enthalpy and entropy become total pressure and total temperature. The flow directions are defined with two angles, a (the angle of the projected velocity vector on the X-Y plane with respect to the X-axis) and b (the angle of the velocity with respect to the X-Y plane), in a Cartesian coordinate shown in figure 2.4. From figure 2.4, we can easily express the two angles, a and b, in terms of the three velocity components, u,v and w, as

$$a = \arctan(v/u) \tag{2.69a}$$

and

$$b = \arctan\left(w/\sqrt{u^2+v^2}\right) \tag{2.69b}$$

To implement the boundary conditions, we define the solution vector of the ghost cell as the vector Ω_g given by

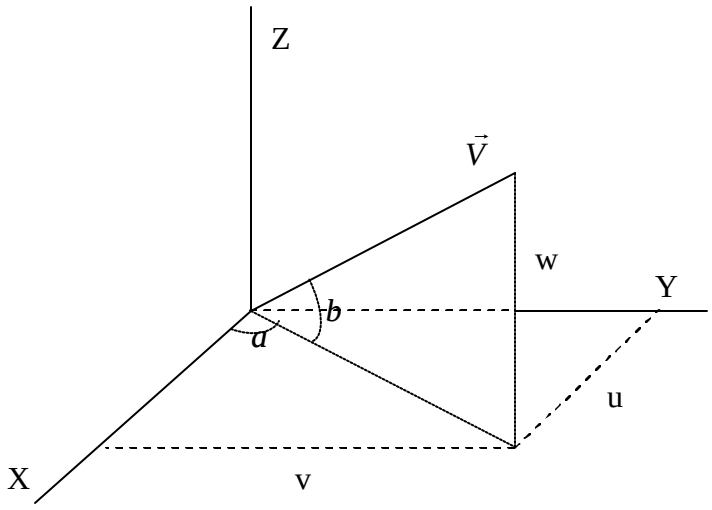

Figure 2.4 Direction angles for velocity

$$\Omega_g = (s_g, a_g, b_g, U_g, h_g^0) \tag{2.70}$$

where the subcript 'g' indicates the ghost cell and the variable, U_g is the magnitude of the velocity of the ghost cell given by

$$U_g = \sqrt{u_g^2 + v_g^2 + w_g^2} \tag{2.71}$$

We define Q_g, as the primary variable Q_p evaluated at the ghost cell,

$$Q_g = \left(P_g, u_g, v_g, w_g, T_g\right)^T \tag{2.72}$$

Clearly, the vector Ω_g is a function of the primary variable Q_g.

We use the symbol Ω_{ref} to represent the specified boundary condition vector as

$$\Omega_{ref} = (s_{ref}, a_{ref}, b_{ref}, 0, h_{ref}^0) \tag{2.73a}$$

and the symbol Ω_b to represent the solution from the flow field as

$$\Omega_b = (0,0,0,U_b,0) \tag{2.73b}$$

where the subcripts 'ref' in equation 2.73a and 'b' in equation 2.73b indicate the boundary condition coming from the boundary cell and the specified boundary condition respectively, and the variable, U_b is the magnitude of the velocity of the boundary cell given by

$$U_b = \sqrt{u_b^2 + v_b^2 + w_b^2} \tag{2.74}$$

Clearly, the vector Ω_b is a function of Q_b, the primary variable Q_p evaluated at the boundary cell

$$Q_b = \left(P_b, u_b, v_b, w_b, T_b\right)^T \tag{2.75}$$

The solution vector of the ghost cell, Ω_g comes from two parts, the boundary conditions and the velocity magnitude from the adjacent cell of the flow field. Accordingly, we require

$$\Omega_g = \Omega_{ref} + \Omega_b \tag{2.76a}$$

Evaluating equation 2.76a implicitly at the new time level, n+1, gives,

$$\Omega_g^{n+1} = \Omega_{ref}^{n+1} + \Omega_b^{n+1} \tag{2.76b}$$

Expanding Ω_g^{n+1} and Ω_b^{n+1} in Taylor series and ignoring the orders higher than Δt gives,

$$\Omega_g^{n+1} = \Omega_g^n + \frac{\partial \Omega_g}{\partial t}\Delta t = \Omega_g^n + \frac{\partial \Omega_g}{\partial Q_g}\frac{\partial Q_g}{\partial t}\Delta t = \Omega_g^n + \frac{\partial \Omega_g}{\partial Q_g}\Delta Q_g \tag{2.77a}$$

or

$$\Omega_b^{n+1} = \Omega_b^n + \frac{\partial \Omega_b}{\partial t}\Delta t = \Omega_b^n + \frac{\partial \Omega_b}{\partial Q_b}\Delta Q_b \tag{2.77b}$$

Now, for steady state computation, Ω_{ref} is independent of time, so we have

$$\Omega_{ref}^{n+1} = \Omega_{ref}^n \tag{2.77c}$$

Substituting equations 2.77a, b and c into equation 2.76b, gives

$$\Omega_g^n + \frac{\partial \Omega_g}{\partial Q_g}\Delta Q_g = \Omega_{ref} + \Omega_b^n + \frac{\partial \Omega_b}{\partial Q_b}\Delta Q_b \tag{2.78}$$

This represents an equation for ΔQ_g, the unknown results in the ghost cell in terms of ΔQ_b, the unknown results in the boundary cell. Solving equation 2.78 for ΔQ_g, we obtain,

$$\Delta Q_g = \left(\frac{\partial \Omega_g}{\partial Q_g}\right)^{-1}\frac{\partial \Omega_b}{\partial Q_b}\Delta Q_b - \left(\frac{\partial \Omega_g}{\partial Q_g}\right)^{-1}\left(\Omega_g^n - \Omega_{ref} - \Omega_b^n\right) \tag{2.79}$$

Equation 2.79 is an implicit equation for the variables in the ghost cell in terms of the variables in the boundary cell. This general form is clearly not limited to the specific inlet boundary condition we discussed here. Instead, it can be applied to any boundary condition as long as we adjust the definition of the three vectors, Ω_g, Ω_{ref} and Ω_b to that boundary condition. In the following, we will discuss several commonly used types of boundary conditions and derive the expressions for the two Jacobians, $\frac{\partial \Omega_g}{\partial Q_g}$ and $\frac{\partial \Omega_b}{\partial Q_b}$, in equation 2.79.

2.10.2 Specify Entropy, Total Enthalpy and Flow Angle at the Inlet

We have discussed this type of boundary condition in subsection 2.10.1. The three boundary condition vectors, Ω_g, Ω_{ref} and Ω_b, have already been defined in equations 2.70, 2.73a and b. To calculate these two Jacobians, $\frac{\partial \Omega_g}{\partial Q_g}$ and $\frac{\partial \Omega_b}{\partial Q_b}$, we need to relate the total enthalpy, entropy, flow angles and the magnitude of the velocity to the primitive variable, Q_p.

For an ideal gas, the entropy can be computed by

$$\Delta s_g = c_p \ln \frac{T_g}{T_1} - R \ln \frac{P_g}{P_1} \tag{2.80}$$

where R is the universal gas constant, and P_1 and T_1 are the some reference pressure and temperature.

The stagnant enthalpy relation is

$$h_g^0 = c_p T_g + \frac{1}{2}\left(U_g\right)^2 \tag{2.81}$$

From equations 2.69a, 2.69b, 2.80 and 2.81, we can easily compute the two Jacobians, $\frac{\partial \Omega_b}{\partial Q_b} = \frac{\partial (0,0,0,U_b,0)}{\partial (P_b, u_b, v_b, w_b, T_b)}$ and $\frac{\partial \Omega_g}{\partial Q_g} = \frac{\partial (s_g, a_g, b_g, U_g, h_g^0)}{\partial (P_g, u_g, v_g, w_g, T_g)}$ as

$$\frac{\partial \Omega_b}{\partial Q_b} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{u_b}{u_b} & \frac{v_b}{u_b} & \frac{w_b}{u_b} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.82}$$

and

$$\frac{\partial \Omega_g}{\partial Q_g} = \begin{pmatrix} -\frac{R}{P_g} & 0 & 0 & 0 & \frac{c_p}{T_g} \\ 0 & -\frac{v_g}{u_g^2 + v_g^2} & \frac{u_g}{u_g^2 + v_g^2} & 0 & 0 \\ 0 & -\frac{u_g w_g}{(U_g)^2 \sqrt{u_g^2 + v_g^2}} & -\frac{v_g w_g}{(U_g)^2 \sqrt{u_g^2 + v_g^2}} & \frac{\sqrt{u_g^2 + v_g^2}}{(U_g)^2} & 0 \\ 0 & \frac{u_g}{U_g} & \frac{v_g}{U_g} & \frac{w_g}{U_g} & 0 \\ 0 & u_g & v_g & w_g & c_p \end{pmatrix} \tag{2.83}$$

Equations 2.82 and 2.83 combined with equation 2.79 give the implicit inlet boundary condition of specifying the entropy, total enthalpy and flow angle for an ideal gas.

2.10.3 Outlet Boundary Condition

At the outlet, we need only one boundary condition for subsonic flow. Conventionally, we specify the back pressure, P_{back} . Consequently, the specified boundary condition vector Ω_{ref} is

$$\Omega_{ref} = (P_{back}, 0, 0, 0, 0) \tag{2.84a}$$

The rest of components of the solution at the ghost cell come from the internal flow field (the boundary cell). So the vector Ω_b becomes

$$\Omega_b = (0, u_b, v_b, w_b, T_b) \tag{2.84b}$$

Accordingly, the ghost cell boundary condition vector, Ω_g , is

$$\Omega_g = (P_g, u_g, v_g, w_g, T_g) \tag{2.84c}$$

From equations 2.84a, b, c and d, we can easily compute the two Jacobians, $\frac{\partial \Omega_b}{\partial Q_b} = \frac{\partial (0, u_b, v_b, w_b, T_b)}{\partial (P_b, u_b, v_b, w_b, T_b)}$ and

$$\frac{\partial \Omega_g}{\partial Q_g} = \frac{\partial (P_g, u_g, v_g, w_g, T_g)}{\partial (P_g, u_g, v_g, w_g, T_g)} = I \tag{2.85}$$

where the symbol, I, is the unit matrix. The first Jacobian is

$$\frac{\partial \Omega_b}{\partial Q_b} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.86}$$

Equations 2.85 and 2.86 combined with equation 2.79 give the implicit outlet boundary condition of specifying the back pressure.

2.10.4 Inviscid Wall Boundary Condition

For inviscid internal flows, we need to use inviscid boundary conditions at the wall. The inviscid wall boundary condition requires the velocity at the wall be tangential to the wall geometry. This requirement can be realized by setting the velocity components of the boundary cell and ghost cell that is normal to the wall equal to each other but has opposite direction, and setting the velocity components of the boundary cell and the ghost cell that is tangential to the wall have the same magnitude and direction. This relation can be written in an equation as

$$\vec{V}_b - \vec{V}_g = 2\left(\vec{V}_b \cdot \vec{n}_w\right)\vec{n}_w$$

or

$$\begin{aligned} \vec{V}_g &= \vec{V}_b - 2\left(\vec{V}_b \cdot \vec{n}_w\right)\vec{n}_w \\ &= \vec{V}_b - 2\left(\vec{V}_b \cdot \vec{n}_w\right)\vec{n}_w \end{aligned} \tag{2.87}$$

where $\vec{n}_w$ is the unit normal vector of the wall pointing to the flow field. The other variables of the ghost cell are set equal to the values of the adjacent cell of the flow field. Consequently, boundary condition vector Ω_{ref} is zero.

Accordingly, we can define the boundary condition vector Ω_b from equation 2.87 as

$$\begin{aligned} \Omega_b = (P_b, u_b - 2(u_b n_x + v_b n_y + w_b n_z)n_x, v_b - 2(u_b n_x + v_b n_y + w_b n_z)n_y, \\ w_b - 2(u_b n_x + v_b n_y + w_b n_z)n_z, T_b) \end{aligned} \tag{2.88a}$$

where n_x, n_y and n_z are three components of the unit wall normal vector, $\vec{n}_w$.

The ghost cell vector Ω_g is

$$\Omega_g = (P_g, u_g, v_g, w_g, T_g) \tag{2.88b}$$

Consequently, from equations 2.88a and b, we obtain

$$\frac{\partial \Omega_g}{\partial Q_g} = \frac{\partial (P_g, u_g, v_g, w_g, T_g)}{\partial (P_g, u_g, v_g, w_g, T_g)} = I \tag{2.89}$$

and

$$\frac{\partial \Omega_b}{\partial Q_b} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1-2n_x^2 & -2n_x n_y & -2n_x n_z & 0 \\ 0 & -2n_x n_y & 1-2n_y^2 & -2n_y n_z & 0 \\ 0 & -2n_x n_z & -2n_y n_z & 1-2n_z^2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.90}$$

Equations 2.89 and 2.90 combined with equation 2.79 give the implicit inviscid boundary condition.

2.10.5 Viscous Wall Boundary Condition

For a viscous internal flow computation, we need to use viscous wall boundary condition. The viscous wall boundary condition requires the velocity at the wall be zero. This requirement can be written as

$$\vec{V}_b + \vec{V}_g = 0$$

or

$$\vec{V}_g = -\vec{V}_b$$

Accordingly, we can define the boundary condition vector Ω_b as

$$\Omega_b = (P_b, -u_b, -v_b, -w_b, T_b) \tag{2.91a}$$

Also, define Ω_g as

$$\Omega_g = (P_g, u_g, v_g, w_g, T_g) \tag{2.91b}$$

Consequently, from equations 2.91a and b, we obtain

$$\frac{\partial \Omega_g}{\partial Q_g} = \frac{\partial (P_g, u_g, v_g, w_g, T_g)}{\partial (P_g, u_g, v_g, w_g, T_g)} = I \tag{2.92}$$

and

$$\frac{\partial \Omega_b}{\partial Q_b} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.93}$$

Equations 2.92 and 2.93 combined with equation 2.79 give the implicit viscous boundary condition.

2.11 Conclusion

In this chapter, we use three pseudo time marching methods to develop the triple time scheme. Green-Gauss reconstruction method is introduced to give second order accuracy. The DDLGS approximate factorization method is introduced to solve this scheme approximately. We also show that the triple time scheme degenerates to the single time schemes and multiple LGS iteration method in special cases. The implicit boundary conditions are discussed.

Chapter 3
Stability Analysis

3.1 Introduction

In the triple time equation 2.21a and its approximately factored version in equation 2.51, there are three different Γ's, Γ_1, Γ_2 and Γ_3, which correspond to three different pseudo times. Obviously, the selection of these three Γ's is not arbitrary and it is very important to select a proper set to make the algorithm efficient and robust.

In the present formulation, we use the first time coefficient matrix, Γ_1, to control the artificial dissipation. It is well known that a proper artificial dissipation is crucial to any scheme. Artificial dissipation can impact the accuracy of the final solution, the robustness of the computation and the convergence rate. An overly large artificial dissipation can lead to an incorrect solution while a dissipation that is too small will result in a solution with odd-even splitting. Consequently, it is very important to select a proper Γ_1 to insure a suitable artificial dissipation.

We will use the second temporal matrix, Γ_2, to solve the non-linear portion of the equation. The non-linear characteristics of the iteration cause robustness to be an important issue. Accordingly, Γ_2 will be chosen to deal with the highly non-linear portion of the problem that is crucial during the start up. Our results also show that Γ_2 affects convergence. Since we can choose an infinite time step at the linear portion of the outer convergence, the effect of Γ_2 on convergence shows up only in the initial non-linear part of the outer convergence.

The third matrix, Γ_3, will be used to solve the linear equation. It should be selected to make the inner convergence as fast as possible.

Stability analysis is very useful to predict the performance of a CFD scheme. In this chapter, we use stability analysis to choose the three matrices for the three pseudo times

to try to give optimum convergence. Some computational results of a linear problem are presented to support our stability results

3.2 Three Candidates of Γs

We will consider three potential forms of the matrix, Γ, as candidates for Γ_1, Γ_2 and Γ_3. The most obvious one is the physical Jacobian matrix of the conservative variable Q with respect to the primary dependent variable Q_p. The three dimensional form of this Jacobian matrix is

$$\Gamma_p = \frac{\partial Q}{\partial Q_p} = \begin{pmatrix} \rho_p & 0 & 0 & 0 & \rho_T \\ \rho_p u & \rho & 0 & 0 & \rho_T u \\ \rho_p v & 0 & \rho & 0 & \rho_T v \\ \rho_p w & 0 & 0 & \rho & \rho_T w \\ \rho_p h_0 + \rho h_p - 1 & \rho u & \rho v & \rho w & \rho_T h_0 + \rho h_T \end{pmatrix} \tag{3.1a}$$

We will call this matrix, Γ_p, the physical matrix since it is the coefficient matrix of the physical time derivative.

For reference in preconditioning, ρ_p can be written in terms of the sound speed, c, which, for a general fluid, is

$$c^2 = \frac{\rho h_T}{\rho \rho_p h_T + \rho_T (1 - \rho h_p)} \tag{3.2a}$$

Solving equation 3.2a for ρ_p, gives

$$\rho_p = \frac{1}{c^2} - \frac{\rho_T (1 - \rho h_p)}{\rho h_T} \tag{3.3a}$$

The second potential choice is the steady preconditioning matrix, Γ, which is developed to deal with the slow convergence problem at low Mach numbers in steady flow [42]. There are several different preconditioning matrices developed by different people [11,38,39,42]. Although in general these are similar in form, we use the one developed by Merkle and Choi [11,42], which is obtained by replacing the physical

property, r_p, in Γ_p by an artificial property, r'_{ps}, where the prime indicates it is artificial and the subscript 's' indicates steady preconditioning in order to distinguish it from the unsteady preconditioner that will be discussed later. We define this 'steady' preconditioning matrix as Γ_s. It is given by

$$\Gamma_s = \begin{pmatrix} r'_{ps} & 0 & 0 & 0 & r_T \\ r'_{ps}u & r & 0 & 0 & r_T u \\ r'_{ps}v & 0 & r & 0 & r_T v \\ r'_{ps}w & 0 & 0 & r & r_T w \\ r'_{ps}h_0 + rh_p - 1 & ru & rv & rw & r_T h_0 + rh_T \end{pmatrix} \tag{3.1b}$$

where r'_{ps} is obtained by replacing the physical sound speed, c in equation 3.3a by a steady preconditioned artificial sound speed, c'_s,

$$r'_{ps} = \frac{1}{(c'_s)^2} - \frac{r_T(1 - rh_p)}{rh_T} \tag{3.3b}$$

For inviscid flow [11,42],

$$c'_s = \min(u, c) \tag{3.2b}$$

where 'u' is the velocity magnitude and 'c' is the sound speed defined in equation 3.2a.

The third time derivative matrix is the "unsteady" preconditioning matrix, Γ_u, developed by Venkateswaran [42] for time accurate computation. This matrix takes into account high frequency unsteady effects during the computation. The unsteady preconditioning matrix is obtained by using an unsteady sound speed instead of the physical sound speed in equation 3.3a. The unsteady artificial sound speed, c'_u is,

$$c'_u = \min\left[\max(u, V_u), c\right] \tag{3.2c}$$

Where $V_u = l/(p\Delta t)$ is the unsteady speed, 'l' is a characteristic length of the problem and Δt is the unsteady time scale, which can be the physical time or a pseudo time [42]. We use the symbol c'_u for the unsteady artificial sound speed to distinguish it from the physical sound speed, c and the steady artificial sound speed, c'_s. To define the unsteady speed in a discretized space, we divide the computational characteristic length 'l' into N equally spaced grids so that $l = N\Delta x$,

$$V_u = \frac{N\Delta x}{p\Delta t}$$

Accordingly, the unsteady property, r'_{pu} becomes

$$r'_{pu} = \frac{1}{(c'_u)^2} - \frac{r_T(1 - rh_p)}{rh_T} \tag{3.4}$$

and the unsteady matrix, Γ_u becomes

$$\Gamma_u = \begin{pmatrix} r'_{pu} & 0 & 0 & 0 & r_T \\ r'_{pu}u & r & 0 & 0 & r_T u \\ r'_{pu}v & 0 & r & 0 & r_T v \\ r'_{pu}w & 0 & 0 & r & r_T w \\ r'_{pu}h_0 + rh_p - 1 & ru & rv & rw & r_T h_0 + rh_T \end{pmatrix} \tag{3.5}$$

From these definitions, we can see that the magnitudes of the three sound speeds and the three r_p's are ordered according to the following,

$$c'_s \le c'_u \le c$$

and

$$r_p \le r'_{pu} \le r'_{ps}$$

In the limit of small pseudo times step (small Δt), $c'_u = c$ and $r'_{pu} = r_p$. While for large pseudo times step (large Δt), $c'_u = c'_s$ and $r'_{pu} = r'_{ps}$. For moderate pseudo time step (moderate Δt), c'_u is between c'_s and c, r'_{pu} is between r_p and r'_{ps}.

The transition of unsteady sound speed c'_u from the physical sound speed to the steady preconditioning sound speed can be obtained by taking the ratio of the unsteady sound speed to the physical sound speed. By doing so, we have

$$\begin{aligned} \frac{c'_u}{c} &= \frac{\min(\max(u, V_u), c)}{c} = \min\left[\max\left(M, \frac{V_u}{c}\right), 1\right] \\ &= \min\left[\max\left(M, \frac{l}{pc\Delta t}\right), 1\right] = \min\left[\max\left(M, M\frac{N}{pCFL_u}\right), 1\right] \end{aligned} \tag{3.6}$$

In equation 3.6, $M = u/c$ is the Mach number, $CFL_u = u\Delta t / \Delta x$ is the CFL based on the particle velocity and the time Δt.

Equation 3.6 is plotted in figure 3.1 for different Mach numbers (M=0.1,0.5 and 0.7) and numbers of grids (N=10,100 and 1000). Both the x-axis and the y-axis are in logarithmic scale. From the figure we can see that the transition of the unsteady artificial sound speed from the physical sound speed to the steady, artificial sound speed is linear on the log-log scale with a slope of minus one for every case. As the Mach number increases, the transition region becomes smaller and disappears as Mach number becomes equal to or greater than one. As the number of grid points increases, the transition region moves to higher value of CFL_u and the physical speed of sound is used over a larger and larger CFL_u region.

In the following we will limit the definition of the three matrices, Γ_1, Γ_2 and Γ_3 to these three matrices, Γ_p, Γ_s and Γ_u. Since the three matrices, Γ_p, Γ_s and Γ_u, are

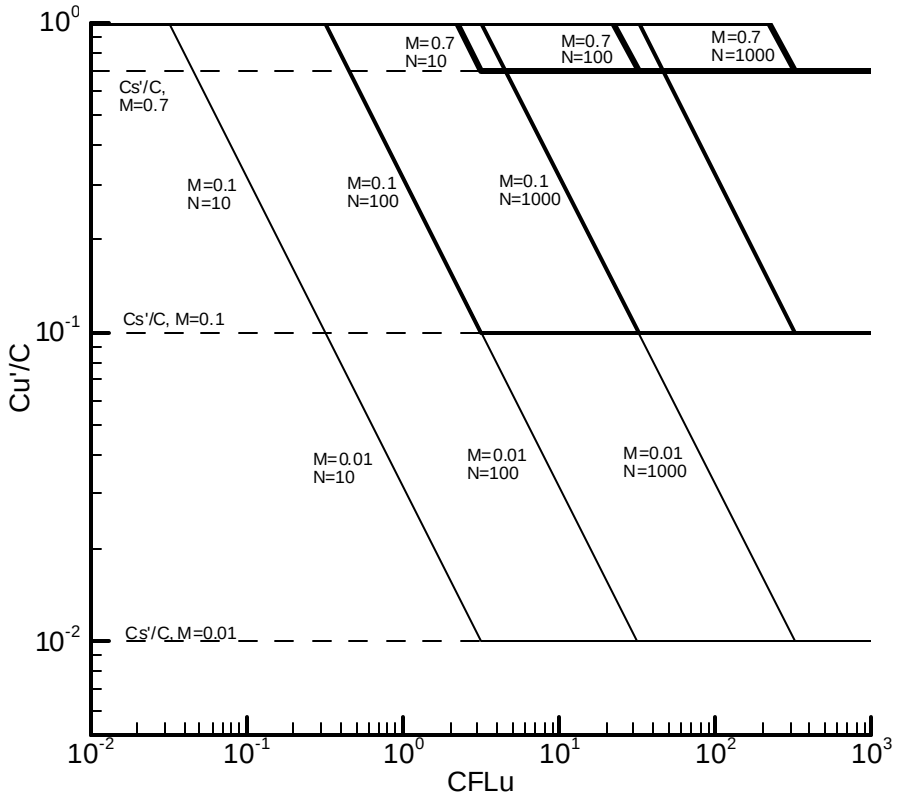

Fig. 3.1 The ratio of preconditioned artificial sound speed and physical sound speed versus CFL_u for M=0.01, 0.1 and 0.7, N=10,100 and 100.

close to each other at high Mach number, all of the following discussion is for the low Mach number case. The purpose is to decide which choice is the best for each time level. We start by choosing an appropriate matrix for Γ_1, which is used to control the artificial dissipation. Our research shows that Γ_s is a very good choice for Γ_1 because it gives an accurate solution upon convergence [42,43]. But Γ_u is also possibly a good choice because $\Gamma_u = \Gamma_s$ at large CFL_u's, which is always true if CFL_u is large enough at the end of the convergence.

Second, to provide robustness during the non-linear part of the outer convergence, Γ_2 should be chosen to be physical because our experience shows that the robustness limit of Γ_p is of the order of the static pressure and that of Γ_s is of the order of the dynamic pressure [42,47]. But Γ_u could also be good because $\Gamma_u = \Gamma_p$ at small CFL_{t2}, which is always true at the non-linear part of convergence. Last, Γ_3 should be Γ_u to achieve fast inner convergence. Based on the above analysis, we can limit our selection to the following four cases in Table 3.1.

In Table 3.1, for the symbol 'UPS', the first letter 'U' indicates the third matrix Γ_3 equals to the unsteady preconditioning matrix Γ_u. The second letter 'P' indicates the second matrix Γ_2 equals to the physical matrix Γ_p and the third letter 'S' indicates the first matrix Γ_1 equals to the steady preconditioning matrix Γ_s. By the same token, we

Table 3.1 The four potential choices for the three matrices, Γ_1, Γ_2 and Γ_3.

cases	Γ_3	Γ_2	Γ_1
UPS	Γ_u	Γ_p	Γ_s
UUS	Γ_u	Γ_u	Γ_s
UUU	Γ_u	Γ_u	Γ_u
UPU	Γ_u	Γ_p	Γ_u

can define ‘UUS’, ‘UUU’ and ‘UPU’.

In the following, we use stability analysis to compare these 4 cases and pick the most appropriate of the four. Before we do that, we define the symbol, */*/* for the discretization accuracy of the system that is going to be used very often in this thesis as: the first star is the order of accuracy of $(\nabla_D \cdot A_p)_3$ or the order of accuracy in t_3, hich is traditionally first order and represented by ‘I’; the second star is the order of accuracy of $(\nabla_D \cdot A_p)_2$ or the order of accuracy in t_2; the third star is the order of accuracy of $(\nabla_D \cdot F)$ or the order of accuracy in t_1. For example, in a I/II/II order system, $(\nabla_D \cdot A_p)_3$ is first order, $(\nabla_D \cdot A_p)_2$ is second order and $(\nabla_D \cdot F)$ is second order. Another way to say this is that it is first order in t_3 and second order in both t_2 and t_1.

Finally we point out that the order of the symbol for the three choices of matrices is the same as the order of the discretization accuracy with the first letter indicating the third pseudo time, the second letter indicating the second pseudo and the third letter indicating the first pseudo time.

3.3 Inner t_3 Stability Analysis

Stability analysis is an important tool that gives a good prediction of the convergence of a scheme. It is always helpful to do a stability analysis before we go to real computations.

First, we do the stability analysis for the direct inversion of the triple time equation The direction inversion procedure is very useful for understanding the stability characteristics of the approximate factorization of four-sweep line Gauss-Siedel procedure that will be done next.

3.3.1 The Direct Inversion of Inner t_3 Stability Analysis

3.3.1.1 General Formulation

The direct inversion stability analysis is the stability analysis for directly solving equation 2.21a without the use of approximate factorization.

Starting from the triple time equation 2.21a,

$$\left[\left(\frac{\Gamma_3}{\Delta t_3}+\frac{\Gamma_2}{\Delta t_2}\right)+\left(\nabla_D\cdot A_p\right)_3\right]^n\Delta\tilde{Q}_p=-\left(\nabla_D\cdot F\right)^n-\left[\frac{\Gamma_2}{\Delta t_2}+\left(\nabla_D\cdot A_p\right)_2\right]^n(\tilde{Q}_p^m-Q_p^n) \qquad (2.21a)$$

we compute the inner stability characteristics of $\tilde{Q}_p$. The stability of $\tilde{Q}_p$ is independent of terms such as, $\left(\nabla_D\cdot F\right)^n$ and $\left[\frac{\Gamma_2}{\Delta t_2}+\left(\nabla_D\cdot A_p\right)_2\right]^n Q_p^n$ that do not contain the variable $\tilde{Q}_p$, because such terms are constant during the iteration and do not contribute to error propagation. These terms can therefore be omitted and the stability form of equation 2.21a becomes

$$\left[\left(\frac{\Gamma_3}{\Delta t_3}+\frac{\Gamma_2}{\Delta t_2}\right)+\left(\nabla_D\cdot A_p\right)_3\right]^n\Delta\tilde{Q}_p=-\left[\frac{\Gamma_2}{\Delta t_2}+\left(\nabla_D\cdot A_p\right)_2\right]^n\tilde{Q}_p{}^m \qquad (3.7)$$

To perform the stability analysis, we define the amplification matrix $\tilde{G}$

$$\tilde{Q}_p^{m+1}=\tilde{G}\tilde{Q}_p^m$$

and substitute this relation into equation 3.7, cancel out the common factor $\tilde{Q}_p{}^m$, and use Fourier transformation to obtain:

$$\left[\left(\frac{\Gamma_3}{\Delta t_3}+\frac{\Gamma_2}{\Delta t_2}\right)+F\left[\left(\nabla_D\cdot A_p\right)_3\right]\right]^n\left(\tilde{G}-I\right)=-\left[\frac{\Gamma_2}{\Delta t_2}+F\left[\left(\nabla_D\cdot A_p\right)_2\right]\right]^n$$

where $F\left[\left(\nabla_D\cdot A_p\right)_3\right]$ and $F\left[\left(\nabla_D\cdot A_p\right)_2\right]$ are the Fourier transformations of $\left(\nabla_D\cdot A_p\right)_3$ and $\left(\nabla_D\cdot A_p\right)_2$ respectively. Their expressions are presented later.

Solving for $\tilde{G}$, we get

$$\tilde{\tilde{G}} = I - \left\{ \left[\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} \right) + F\left[\left(\nabla_D \cdot A_p \right)_3 \right] \right]^n \right\}^{-1} \left[\frac{\Gamma_2}{\Delta t_2} + F\left[\left(\nabla_D \cdot A_p \right)_2 \right] \right]^n \qquad (3.8a)$$

For the I/I/* system, $\left(\nabla_D \cdot A_p \right)_2 = \left(\nabla_D \cdot A_p \right)_3 = \left(\nabla_D \cdot A_p \right)_I$, equation 3.8a can be simplified to give

$$\tilde{\tilde{G}} = I - \left\{ \left[\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} \right) + F\left[\left(\nabla_D \cdot A_p \right)_I \right] \right]^n \right\}^{-1} \left[\frac{\Gamma_2}{\Delta t_2} + F\left[\left(\nabla_D \cdot A_p \right)_I \right] \right]^n$$

or upon simplifying

$$\tilde{\tilde{G}} = \left\{ \left[\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} \right) + F\left[\left(\nabla_D \cdot A_p \right)_I \right] \right]^n \right\}^{-1} \left(\frac{\Gamma_3}{\Delta t_3} \right)$$

Taking the inverse of this equation, we obtain,

$$\tilde{\tilde{G}}^{-1} = \Gamma_3^{-1} \left(\Gamma_3 + \frac{\Delta t_3}{\Delta t_2} \Gamma_2 + \Delta t_3 F\left[\left(\nabla_D \cdot A_p \right)_I^n \right] \right)$$

and upon multiplying by Γ_3

$$\tilde{\tilde{G}}^{-1} = I + \frac{\Delta t_3}{\Delta t_2} \Gamma_3^{-1} \Gamma_2 + \Delta t_3 \Gamma_3^{-1} F\left[\left(\nabla_D \cdot A_p \right)_I^n \right] \qquad (3.8b)$$

The stability analysis calculates all eigenvalues of the amplification matrix, $\tilde{\tilde{G}}$, but the largest eigenvalue decides the convergence rate of the scheme. The amplification factor matrix, $\tilde{\tilde{G}}$, is generally computed numerically because of the complexity, but for the one dimensional case, we can compute $\tilde{\tilde{G}}$ analytically, which will give us a good understanding for the two and three dimensional cases and a tool to check the correctness of our two and three dimensional numerical results.

3.3.1.2 One Dimensional Analytical Stability Analysis of t_3 Step with Direct Inversion

In this subsection, we compute the eigenvalues of one dimensional amplification matrix $\tilde{\tilde{G}}$ analytically for direct inversion of the t_3 time step with first order upwind

differencing. First, we must obtain the Fourier transformation of the discretized Jacobian divergence operator. From equation 2.20b, for one dimension, the discretized Jacobian divergence is

$$\left(\nabla_D \cdot A_p\right)^n = \sum_{k=1}^{2}\left[\frac{1}{2}\left(A_{pL}\bullet_L + A_{pR}\bullet_R\right) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|_k\left(\bullet_R - \bullet_L\right)\right]_k \frac{1}{\Delta x}$$

When applied to the first order upwind in space, this divergence operator becomes

$$\left(\nabla_D \cdot A_p\right)_I^n = \frac{1}{\Delta x}\Gamma_1\left[\left|\Gamma_1^{-1}A_p\right|_i + \left(\Gamma_1^{-1}A_p\right)_{i+1}^- - \left(\Gamma_1^{-1}A_p\right)_{i-1}^+\right]$$

Ignoring the variations in the Jacobian coefficient matrix between grid points, we get the Fourier transformation of this equation as

$$F\left[\left(\nabla_D \cdot A_p\right)_I^n\right] = \frac{1}{\Delta x}\Gamma_1\left[\left|\Gamma_1^{-1}A_p\right|\left(1 - C_x\right) + iS_x\Gamma_1^{-1}A_p\right] \tag{3.9a}$$

where S_x and C_x are the sine and cosine of the wave number in the x-direction as the following

$$S_x = \sin f \quad \text{and} \quad C_x = \cos f \tag{3.9b}$$

f is the wave number in x-direction, and $f = 2pn/N$, $n = 0, 1, \cdots,$ N (N is the number of grids in x-direction).

Upon substituting equation 3.9a for the Fourier transformation of the divergence operator into equation 3.8b and arrangement, we obtain

$$\tilde{G}^{-1} = I + \frac{\Delta t_3}{\Delta t_2}\Gamma_3^{-1}\Gamma_2 + \frac{\Delta t_3}{\Delta x}\left[\Gamma_3^{-1}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(1 - C_x\right) + iS_x\Gamma_3^{-1}A_p\right] \tag{3.10}$$

To compute the inverse of the amplification factor $\tilde{G}^{-1}$, we need to know the matrices, $\Gamma_3^{-1}\Gamma_2$, $\Gamma_3^{-1}\Gamma_1\left|\Gamma_1^{-1}A_p\right|$ and $\Gamma_3^{-1}A_p$. In the following, we compute all these matrices.

We begin with the one dimensional form of the three Γ's in equation 3.1a,b and 3.5, which can be written in a generic form as

$$\Gamma_k = \begin{pmatrix} r_{pk} & 0 & r_T \\ r_{pk}u & r & r_T u \\ r_{pk}h_0 + rh_p - 1 & ru & r_T h_0 + rh_T \end{pmatrix} \tag{3.11a}$$

where subscript k runs from 1 to 3 to represent the 3 different Γ's.

The determinant of Γ_k can be easily computed to be

$$\det(\Gamma_k) = r\left[r_T(1-rh_p)+rh_T r_{pk}\right] = rd_k$$

where

$$d_k = r_T(1-rh_p)+rh_T r_{pk}$$

The inverse of Γ_k is

$$\Gamma_k^{-1} = \begin{pmatrix} \frac{1}{d_k}\left[rh_T + r_T\left(h-\frac{1}{2}u^2\right)\right] & \frac{r_T u}{d_k} & -\frac{r_T}{d_k} \\ -\frac{u}{r} & \frac{1}{r} & 0 \\ \frac{1}{d_k}\left[(1-rh_p)-r_{pk}\left(h-\frac{1}{2}u^2\right)\right] & -\frac{r_{pk}u}{d_k} & \frac{r_{pk}}{d_k} \end{pmatrix} \qquad (3.11b)$$

The Jacobian matrix A_p is

$$A_p = \frac{\partial E}{\partial Q_p} = \frac{\partial(ru,\, ru^2+p,\, ruh_o)}{\partial(p,u,T)}$$

$$= \begin{pmatrix} r_p u & r & r_T u \\ r_p u^2+1 & 2ru & r_T u^2 \\ (r_p h_0 + rh_p)u & r(h_0+u^2) & (r_T h_0 + rh_T)u \end{pmatrix}$$

Define $d = r_T(1-rh_p)+rh_T r_p$, for the physical value of r_p. Also define the ratio of 'd' and 'd_k' to be

$$e_k = d/d_k$$

and the artificial sound speed,

$$c_k^2 = \frac{rh_T}{d_k}$$

We get

$$\Gamma_k^{-1}A_p = \begin{pmatrix} ue_k & rc_k^2 & 0 \\ \frac{1}{r} & u & 0 \\ \frac{1}{d_k}u\left(r_p - r_{pk}\right)\left(1-rh_p\right) & \frac{1}{d_k}r\left(1-rh_p\right) & u \end{pmatrix} \tag{3.12}$$

and

$$\Gamma_j^{-1}\Gamma_k = \begin{pmatrix} \frac{d_k}{d_j} & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{d_j}\left(1-rh_p\right)\left(r_{pk} - r_{pj}\right) & 0 & 1 \end{pmatrix} \tag{3.13}$$

where the subscripts j and k run from 1 to 3.

The eigenvalues of the matrix $\Gamma_k^{-1}A_p$ are

$$l_1 = \frac{1}{2}u(1+e_k) + \frac{1}{2}s_k \tag{3.14a}$$

$$l_2 = \frac{1}{2}u(1+e_k) - \frac{1}{2}s_k \tag{3.14b}$$

$$l_3 = u \tag{3.14c}$$

where,

$$s_k = \sqrt{u^2\left(1-e_k\right)^2 + 4c_k^2} \tag{3.15}$$

It can be easily proven that $l_1 > 0$ and $l_2 \le 0$ for $u \le c$.

The right eigenvectors of the matrix $\Gamma_k^{-1}A_p$ are

$$M_k = \begin{pmatrix} \frac{s_k - u\left(1-e_k\right)}{2s_k} & \frac{s_k + u\left(1-e_k\right)}{2s_k} & 0 \\ \frac{1}{rs_k} & -\frac{1}{rs_k} & 0 \\ \frac{1-rh_p}{rh_T}\frac{s_k - u\left(1-e_k\right)}{2s_k} & \frac{1-rh_p}{rh_T}\frac{s_k + u\left(1-e_k\right)}{2s_k} & -\frac{1-rh_p}{rh_T} \end{pmatrix}$$

Upon introducing the eigenvalue equation set 3.14, we obtain

$$M_k = \begin{pmatrix} \dfrac{l_2 - e_k u}{l_2 - l_1} & \dfrac{l_1 - e_k u}{l_1 - l_2} & 0 \\ \dfrac{1}{r(l_1 - l_2)} & -\dfrac{1}{r(l_1 - l_2)} & 0 \\ \dfrac{1 - rh_p}{rh_T}\dfrac{l_2 - e_k u}{l_2 - l_1} & \dfrac{1 - rh_p}{rh_T}\dfrac{l_1 - e_k u}{l_1 - l_2} & -\dfrac{1 - rh_p}{rh_T} \end{pmatrix}$$

The left eigenvectors are obtained from the inverse of the matrix M_k as

$$M_k^{-1} = \begin{pmatrix} 1 & r\dfrac{s_k + u(1 - e_k)}{2} & 0 \\ 1 & r\dfrac{-s_k + u(1 - e_k)}{2} & 0 \\ 1 & 0 & -\dfrac{rh_T}{1 - rh_p} \end{pmatrix}$$

or, written in term of the eigenvalues

$$M_k^{-1} = \begin{pmatrix} 1 & r(l_1 - e_k u) & 0 \\ 1 & r(l_2 - e_k u) & 0 \\ 1 & 0 & -\dfrac{rh_T}{1 - rh_p} \end{pmatrix}$$

So, the diagonalization of the matrix $\Gamma_k^{-1} A_p$ becomes

$$M_k^{-1}\left(\Gamma_k^{-1} A_p\right) M_k = \begin{pmatrix} l_1 & 0 & 0 \\ 0 & l_2 & 0 \\ 0 & 0 & l_3 \end{pmatrix}$$

and the matrix $\left|\Gamma_k^{-1} A_p\right|$ is defined by multiplying the right eigenvector matrix M_k by a eigenvalue matrix where all eigenvalues are the magnitudes of the eigenvalues of matrix $\Gamma_k^{-1} A_p$ and then multiplying by the left eigenvector matrix M_k^{-1} as the following

$$\left|\Gamma_k^{-1} A_p\right| = M_k \begin{pmatrix} |l_1| & 0 & 0 \\ 0 & |l_2| & 0 \\ 0 & 0 & |l_3| \end{pmatrix} M_k^{-1}$$

For supersonic flow, all eigenvalues are greater than or equal to zero, we have

$\left|\Gamma_k^{-1}A_p\right| = \Gamma_k^{-1}A_p$. For subsonic flow, $l_1, l_3 \geq 0$ and $l_2 \leq 0$, we obtain

$$\left|\Gamma_k^{-1}A_p\right| = M_k \begin{pmatrix} l_1 & 0 & 0 \\ 0 & -l_2 & 0 \\ 0 & 0 & l_3 \end{pmatrix} M_k^{-1}$$

Performing this matrix multiplication gives

$$\left|\Gamma_k^{-1}A_p\right| = \begin{pmatrix} \dfrac{e_k u(l_1+l_2)-2l_1l_2}{l_1-l_2} & -\dfrac{r(l_1+l_2)(l_1-e_k u)(l_2-e_k u)}{l_1-l_2} & 0 \\ \dfrac{1}{r}\dfrac{l_1+l_2}{l_1-l_2} & \dfrac{{l_1}^2+{l_2}^2-e_k u(l_1+l_2)}{l_1-l_2} & 0 \\ \left[\dfrac{e_k u(l_1+l_2)-2l_1l_2}{l_1-l_2}-u\right]\dfrac{1-rh_p}{rh_T} & -\dfrac{r(l_1+l_2)(l_1-e_k u)(l_2-e_k u)}{l_1-l_2}\dfrac{1-rh_p}{rh_T} & l_3 \end{pmatrix}$$

Substituting the eigenvalue equation set 3.14, we obtain

$$\left|\Gamma_k^{-1}A_p\right| = \begin{pmatrix} \dfrac{1}{s_k}\left[u^2e_k(e_k-1)+2c_k^2\right] & \dfrac{1}{s_k}ru(1+e_k)c_k^2 & 0 \\ \dfrac{u(1+e_k)}{rs_k} & \dfrac{1}{s_k}\left[u^2(1-e_k)+2c_k^2\right] & 0 \\ \left\{\dfrac{1}{s_k}\left[u^2e_k(e_k-1)+2c_k^2\right]-u\right\}\dfrac{1-rh_p}{rh_T} & \dfrac{1}{s_k}ru(1+e_k)c_k^2\dfrac{1-rh_p}{rh_T} & u \end{pmatrix} \qquad (3.16)$$

In the last matrix, if we set $k=1$, we obtain the expression for $\left|\Gamma_1^{-1}A_p\right|$. In equation 3.13, we can set j=3 and k=1 to obtain $\Gamma_3^{-1}\Gamma_1$. Multiplying these two matrices, gives

$$\Gamma_3^{-1}\Gamma_1\left|\Gamma_1^{-1}A_p\right| = \begin{pmatrix} \dfrac{d_1}{d_3}\dfrac{1}{s_1}\left[u^2e_1(e_1-1)+2c_1^2\right] & \dfrac{d_1}{d_3}\dfrac{1}{s_1}ru(1+e_1)c_1^2 & 0 \\ \dfrac{u(1+e_1)}{rs_1} & \dfrac{1}{s_1}\left[u^2(1-e_1)+2c_1^2\right] & 0 \\ \left\{\dfrac{1}{s_1}\left[u^2e_1(e_1-1)+2c_1^2\right]\left(1+rh_T\dfrac{r_{p1}-r_{p3}}{d_3}\right)-u\right\}\dfrac{1-rh_p}{rh_T} & \dfrac{1}{s_1}ru(1+e_1)c_1^2\dfrac{1-rh_p}{rh_T}\left(1+rh_T\dfrac{r_{p1}-r_{p3}}{d_3}\right) & u \end{pmatrix}$$

which is the middle term in equation 3.10. In equation 3.13, set j=3 and k=2 to obtain $\Gamma_3^{-1}\Gamma_2$, which is the first term in equation 3.10. In equation 3.12, set k=3 to obtain

$\Gamma_3^{-1} A_p$, which is the last term in equation 3.10.

Substitute $\Gamma_3^{-1}\Gamma_2$, $\Gamma_3^{-1}\Gamma_1\left|\Gamma_1^{-1}A_p\right|$ and $\Gamma_3^{-1}A_p$ into equation 3.10 to obtain

$$\tilde{G}^{-1} = I + \frac{\Delta t_3}{\Delta t_2}\Gamma_3^{-1}\Gamma_2 + \frac{\Delta t_3}{\Delta x}\Gamma_3^{-1}\Gamma_1\left[\left|\Gamma_1^{-1}A_p\right|(1-C_x) + iS_x\Gamma_1^{-1}A_p\right]$$

Separate the real and imaginary parts to obtain

$$\tilde{G}^{-1} = \left[I + \frac{\Delta t_3}{\Delta t_2}\Gamma_3^{-1}\Gamma_2 + \frac{\Delta t_3}{\Delta x}\Gamma_3^{-1}\Gamma_1\left|\Gamma_1^{-1}A_p\right|(1-C_x)\right] + iS_x\frac{\Delta t_3}{\Delta x}\Gamma_3^{-1}A_p$$

Writing this in matrix notation gives

$$\tilde{G}^{-1} = \begin{pmatrix} C_{11} & C_{12} & 0 \\ C_{21} & C_{22} & 0 \\ C_{31} & C_{32} & C_{33} \end{pmatrix}$$

where

$$C_{11} = 1 + \frac{\Delta t_3}{\Delta t_2}\frac{d_2}{d_3} + \frac{d_1}{d_3}\frac{1}{s_1}\left[u^2 e_1(e_1 - 1) + 2c_1^2\right]\frac{\Delta t_3}{\Delta x}(1 - C_x) + i\frac{\Delta t_3}{\Delta x}S_x u e_3 \qquad (3.17a)$$

$$C_{12} = \frac{d_1}{d_3}\frac{1}{s_1} r u(1 + e_1)c_1^2\frac{\Delta t_3}{\Delta x}(1 - C_x) + i\frac{\Delta t_3}{\Delta x}S_x r c_3^2 \qquad (3.17b)$$

$$C_{21} = \frac{\Delta t_3}{\Delta x}(1 - C_x)\frac{u(1 + e_1)}{r s_1} + i\frac{\Delta t_3}{\Delta x}S_x\frac{1}{r} \qquad (3.17c)$$

$$C_{22} = 1 + \frac{\Delta t_3}{\Delta t_2} + \frac{1}{s_1}\left[u^2(1 - e_1) + 2c_1^2\right]\frac{\Delta t_3}{\Delta x}(1 - C_x) + i\frac{\Delta t_3}{\Delta x}S_x u \qquad (3.17d)$$

$$C_{33} = 1 + \frac{\Delta t_3}{\Delta t_2} + u\frac{\Delta t_3}{\Delta x}(1 - C_x) + iu\frac{\Delta t_3}{\Delta x}S_x \qquad (3.17e)$$

Since the first two components in the third column are both zeros, the two terms C_{31} and C_{32} in the third row do not affect the convergence. Also, the expressions of C_{31} and C_{32} are very complex and accordingly are not given.

The eigenvalues of $\tilde{G}^{-1}$ are

$$l_1^{-1} = C_{33} = 1 + \frac{\Delta t_3}{\Delta t_2} + u\frac{\Delta t_3}{\Delta x}(1 - C_x) + iu\frac{\Delta t_3}{\Delta x}S_x$$

$$l_{2,3}{}^{-1} = \frac{1}{2}\left[C_{11} + C_{22} \pm \sqrt{(C_{11} - C_{22})^2 + 4C_{12}C_{21}}\right]$$

where $l_{1,2,3}$ are the eigenvalues of matrix $\tilde{G}$.

The eigenvalues of $\tilde{G}$ are the inverse of the eigenvalues of $\tilde{G}^{-1}$, which are

$$l_1 = \left(1 + \frac{\Delta t_3}{\Delta t_2} + u\frac{\Delta t_3}{\Delta x}(1 - C_x) + i\frac{\Delta t_3}{\Delta x}S_x u\right)^{-1} \tag{3.18a}$$

$$l_{2,3} = \left\{\frac{1}{2}\left[C_{11} + C_{22} \pm \sqrt{(C_{11} - C_{22})^2 + 4C_{12}C_{21}}\right]\right\}^{-1} \tag{3.18b}$$

For the zero wave number, $C_x = 1$ and $S_x = 0$, equations 3.17a, b, c and d give

$$C_{11} = 1 + \frac{\Delta t_3}{\Delta t_2}\frac{d_2}{d_3}$$

$$C_{12} = 0$$

$$C_{21} = 0$$

$$C_{22} = 1 + \frac{\Delta t_3}{\Delta t_2}$$

Substituting into equation 3.18a and b, we obtain

$$l_1 = \left(1 + \frac{\Delta t_3}{\Delta t_2}\right)^{-1} \tag{3.19a}$$

$$l_2 = \left(1 + \frac{\Delta t_3}{\Delta t_2}\frac{d_2}{d_3}\right)^{-1} \tag{3.19b}$$

$$l_3 = \left(1 + \frac{\Delta t_3}{\Delta t_2}\right)^{-1} \tag{3.19c}$$

From the eigenvalues equation 3.19a, b, c and d, we can see that the sink term, $\Delta t_3 / \Delta t_2$ (which is always positive) provides damping at the zero wave number. Since zero wave number generally is the most stiff point over all wave number space, we can expect that the larger the sink term the faster our scheme will converge. Results given later generally collaborate this statement.

The maximum of the three eigenvalues in equation set 3.18 are plotted over all wave

number space in figure 3.2 for the 'UPU' ($\Gamma_1 = \Gamma_u$, $\Gamma_2 = \Gamma_p$ and $\Gamma_3 = \Gamma_u$) case, which will be shown later to be the best choice. We use a pressure of 1.0e5 pa, a temperature of 300 Kelvin, a Mach Number of 0.01, a grid number N of 100, and a $CFL_{u3} = u\Delta t_3 / \Delta x$ of 10 and show results for a series of $\Delta t_3 / \Delta t_2$ values.

This plot shows how the sink term $\Delta t_3 / \Delta t_2$ helps the damping of our scheme. The uppermost line ($\Delta t_3 / \Delta t_2 = 0.1$) has a very small sink term. It is very similar to the single time result [27] which has no source term, and is very stiff in the small wave number region but gives strong damping in the large wave number region. As the sink term increases, the damping gets stronger and the figure shows that it acts over all wave numbers. When $\Delta t_3 / \Delta t_2 = 100$ or 1000, we get an extremely fast convergence for all wave numbers including the zero one. The discontinuous 'spike' at the $\Delta t_3 / \Delta t_2 = 10$ line is caused by the switch from one eigenvalue to another.

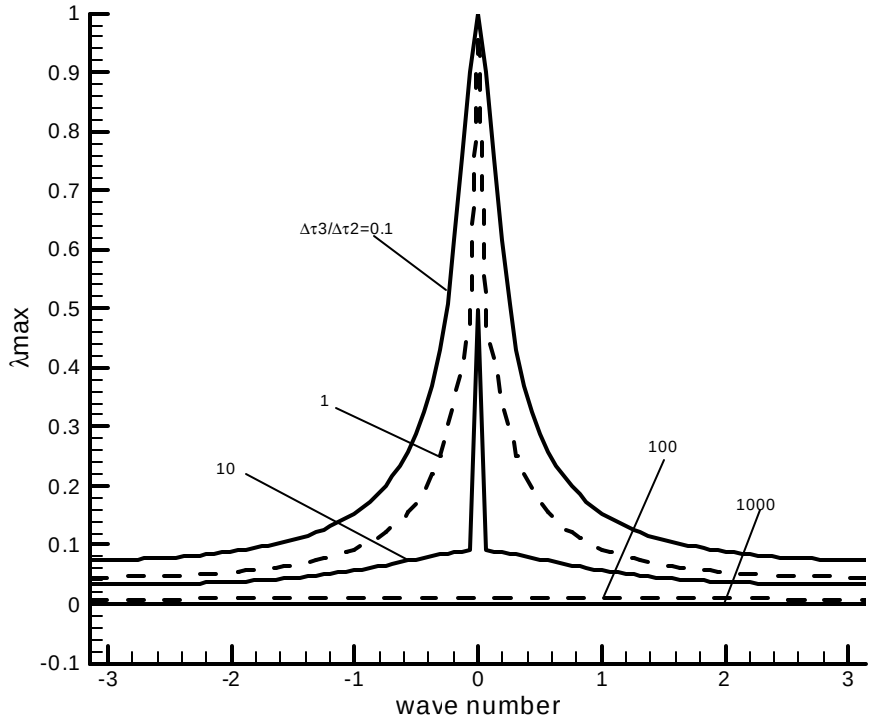

Figure 3.2 The maximum eigenvalue of direct inversion of one dimensional amplification factor for inner iteration (UPU, I/I/*, M=0.01, N=100 and $CFL_{u3} = 10$)

3.3.2 The Inner t_3 Stability of the Four-Sweep DDLGS

The inner t_3 stability analysis of the direct inversion of the one dimensional problem gives a good understanding of the convergence of the exact solution, but direct inversion is not practical in multi-dimensions. The linear equation is not solved exactly or directly but is solved approximately. Although the stability of the approximate solver shares something in common with that of the direct solver, many differences exist and sometimes these differences are very large. Consequently, it is necessary to conduct the stability analysis of the approximate solver. As noted before, we will use the diagonal dominated Line Gauss-Siedel (DDLGS) approximate factorization method.

Beginning with the four-sweep DDLGS equations 2.50a, b, c and d

$$\left(L_3 - J_{i+1,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{2.50a}$$

$$\left(L_3 - J_{i-1,j}\right)\Delta\tilde{Q}_p^{\,**} = -R_3 - J_{i-1,j}\Delta\tilde{Q}_p^{\,*} \tag{2.50b}$$

$$\left(L_3 - J_{i,j+1}\right)\Delta\tilde{Q}_p^{\,***} = -R_3 - J_{i,j+1}\Delta\tilde{Q}_p^{\,**} \tag{2.50c}$$

$$\left(L_3 - J_{i,j-1}\right)\Delta\tilde{Q}_p = -R_3 - J_{i,j-1}\Delta\tilde{Q}_p^{\,***} \tag{2.50d}$$

We define

$$\Delta\tilde{Q}_p^{*} = \tilde{G}^{*}\tilde{Q}_p^{m},\ \ \Delta\tilde{Q}_p^{**} = \tilde{G}^{**}\tilde{Q}_p^{m},\ \ \Delta\tilde{Q}_p^{***} = \tilde{G}^{***}\tilde{Q}_p^{m},\ \ \tilde{Q}_p^{m+1} = \tilde{G}\tilde{Q}_p^{m}$$

where $\tilde{G}$ is the amplification factor for the time step from Q_p^m to Q_p^{m+1} and $\tilde{G}^{*}, \tilde{G}^{**}, \tilde{G}^{***}$ are amplification factors for the intermediate variables.

Substituting all these amplification factors into equations 2.50a, b, c and d, and using Fourier transformation, after arrangement, we obtain,

$$\tilde{G}^{*} = -\left[F\left(L_3\right) - F\left(J_{i+1,j}\right)\right]^{-1} F\left(R_3\right) \tag{3.36a}$$

$$\tilde{G}^{**} = \left[F\left(L_3\right) - F\left(J_{i-1,j}\right)\right]^{-1}\left[-F\left(R_3\right) - F\left(J_{i-1,j}\right)\tilde{G}^{*}\right] \tag{3.36b}$$

$$\tilde{G}^{***} = \left[F\left(L_3\right) - F\left(J_{i,j+1}\right)\right]^{-1}\left[-F\left(R_3\right) - F\left(J_{i,j+1}\right)\tilde{G}^{**}\right] \tag{3.36c}$$

$$\tilde{G} = \left[F\left(L_3\right) - F\left(J_{i,j-1}\right)\right]^{-1}\left[-F\left(R_3\right) - F\left(J_{i,j-1}\right)\tilde{G}^{***}\right] \tag{3.36d}$$

where,

$$F(L_3)=\frac{\Gamma_3}{\Delta t_3}+\frac{\Gamma_2}{\Delta t_2}+F\left[(\nabla_D\cdot A_p)_3^n\right] \tag{3.37}$$

is the Fourier transformation of operator L_3 in equation 2.22b, and

$$F(R_3)=\frac{\Gamma_2}{\Delta t_2}+F\left[(\nabla_D\cdot A_p)_2^n\right] \tag{3.38}$$

is the Fourier transformation of operator R_3 in equation 2.22a.

The Fourier transformation of the residual R_3 requires the Fourier tansformation of the discretized divergence operator. Because we will use the first order upwind and second order upwind to calculate this term, we will give both Fourier transforms here.

The discretized divergence operator for the first order can be computed from equation 2.30

$$F\left[(\nabla_D\cdot A_p)_I^n\right]=\left(\frac{1-C_x}{\Delta x}\Gamma_1\left|\Gamma_1^{-1}A_p\right|+\frac{1-C_y}{\Delta y}\Gamma_1\left|\Gamma_1^{-1}B_p\right|\right)+i\left(\frac{S_x}{\Delta x}A_p+\frac{S_y}{\Delta y}B_p\right) \tag{3.39}$$

where S_x and C_x are the sine and cosine of the wave number in the x-direction as defined in equation 3.9b, and S_y and C_y are the sine and cosine of the wave number in the y-direction as

$$S_x=\sin j \quad \text{and} \quad C_y=\cos j \tag{3.40}$$

j is the wave number in the y-direction, and $j=2pm/M$, $m=0,1,\cdots,\mathrm{M}$ (M is the number of grids).

For second order discretization, we substitute equation 2.20b into equation 2.20a and rearrange the right hand side of equation 2.20b, then equation 2.20a becomes

$$(\nabla\cdot A_p)\Delta Q_p=\frac{1}{\Omega}\sum_{k=1}^{4}\left(\frac{A_{pL}+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\Delta Q_{pL}+\frac{A_{pR}-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\Delta Q_{pR}\right)_k S_k \tag{3.41}$$

where Q_{pL} and Q_{pR} are computed by equation set 2.29, and A_{pL}, A_{pR} and $\Gamma_1\left|\Gamma_1^{-1}A_p\right|$ are computed from Q_{pL} and Q_{pR}. For stability analysis, $A_{pL}, A_{pR}, \Gamma_1\left|\Gamma_1^{-1}A_p\right|$ are set equal at every face of the whole computational domain and can be treated as constants.

Consequently we can ignore the subscripts in the notation. Substituting equation set 2.29 into equation 3.41 to obtain the second order discretization,

$$\begin{aligned}
\left(\nabla\cdot A_p\right)_{II} Q_p = &\frac{A_p+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\left[Q_{pi-1,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{1L}\right]\frac{1}{\Delta x}-\frac{A_p-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\left[Q_{pi,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{1R}\right]\frac{1}{\Delta x}\\
&+\frac{A_p+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\left[Q_{pi,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{2L}\right]\frac{1}{\Delta x}+\frac{A_p-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2}\left[Q_{pi+1,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{2R}\right]\frac{1}{\Delta x}\\
&-\frac{B_p+\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2}\left[Q_{pi,j-1}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{3L}\right]\frac{1}{\Delta y}-\frac{B_p-\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2}\left[Q_{pi,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{3R}\right]\frac{1}{\Delta y}\\
&+\frac{B_p+\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2}\left[Q_{pi,j}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{4L}\right]\frac{1}{\Delta y}+\frac{B_p-\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2}\left[Q_{pi,j+1}+\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{4R}\right]\frac{1}{\Delta y}
\end{aligned}$$

where the subscript 'II' on $\left(\nabla\cdot A_p\right)_{II}$ indicates second order discretization.

By putting all the gradient terms together and all the Q_p's at the cells together, we can find that $\left(\nabla\cdot A_p\right)_{II} Q_p$ is equal to the first order expression, $\left(\nabla\cdot A_p\right)_I Q_p$ (equation 2.30), plus the gradient terms.

$$\begin{aligned}
&\left(\nabla\cdot A_p\right)_{II} Q_p = \left(\nabla\cdot A_p\right)_I Q_p\\
&-\frac{A_p+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{1L}-\frac{A_p-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{1R}+\frac{A_p+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{2L}+\frac{A_p-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{2R}\\
&-\frac{B_p+\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{3L}-\frac{B_p-\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{3R}+\frac{B_p+\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{4L}+\frac{B_p-\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}\left(\nabla\overline{Q}_p\cdot\Delta\vec{r}\right)_{4R}
\end{aligned}$$

(3.42a)

Substituting the gradient term equation sets 2.27 and 2.28 into equation 3.42a, after arrangement, we get

$$\begin{aligned}
&\left(\nabla\cdot A_p\right)_{II} Q_p = \left(\nabla\cdot A_p\right)_I Q_p\\
&-\frac{1}{4}\left(\frac{\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{\Delta x}Q_{pi,j}+\frac{A_p}{\Delta x}Q_{pi-1,j}-\frac{A_p}{\Delta x}Q_{pi+1,j}+\frac{A_p-\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}Q_{pi+2,j}-\frac{A_p+\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{2\Delta x}Q_{pi-2,j}\right)\\
&-\frac{1}{4}\left(\frac{\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{\Delta y}Q_{pi,j}+\frac{B_p}{\Delta y}Q_{pi,j-1}-\frac{B_p}{\Delta y}Q_{pi,j+1}+\frac{B_p-\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}Q_{pi,j+2}-\frac{B_p+\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{2\Delta y}Q_{pi,j-2}\right)
\end{aligned}$$

(3.42b)

The Fourier transformation of the Jacobian divergence for the second order can be computed from equation 3.42b as

$$F\left[(\nabla_D \cdot A_p)^n_{II}\right] = F\left[(\nabla_D \cdot A_p)^n_I\right] - \frac{1}{2}\left[\frac{\Gamma_1\left|\Gamma_1^{-1}A_p\right|}{\Delta x}(1-C_x^2)+\frac{\Gamma_1\left|\Gamma_1^{-1}B_p\right|}{\Delta y}(1-C_y^2)\right]+i\frac{1}{2}\left[\frac{A_p}{\Delta x}S_x(1-C_x)+\frac{B_p}{\Delta y}S_y(1-C_y)\right] \quad (3.43)$$

For the I/I/* (I/I/I and I/I/II) systems,

$$F\left[(\nabla_D \cdot A_p)^n_3\right] = F\left[(\nabla_D \cdot A_p)^n_2\right] = F\left[(\nabla_D \cdot A_p)^n_I\right] \quad (3.44a)$$

where the Fourier transformation of the first order divergence at the right side of equation 3.44a is given in equation 3.39.

For the I/II/II sytem,

$$F\left[(\nabla_D \cdot A_p)^n_3\right] = F\left[(\nabla_D \cdot A_p)^n_I\right] \quad (3.44b)$$

$$F\left[(\nabla_D \cdot A_p)^n_2\right] = F\left[(\nabla_D \cdot A_p)^n_{II}\right] \quad (3.44c)$$

Equation set 3.36, equations 3.37, 3.38 and equation set 3.44 give stability formulation for the four-sweep DDLGS systems of I/I/I, I/I/II and I/II/II accuracy. Numerical methods are used to compute the amplification matrix $\tilde{\tilde{G}}$.

3.3.3 Inner Stability Results

The amplification matrix $\tilde{\tilde{G}}$ in equation 3.36 can not be computed until we choose all three Γs. For now, let us choose $\Gamma_3 = \Gamma_u$, $\Gamma_2 = \Gamma_p$ and $\Gamma_1 = \Gamma_u$ (we will prove later that this is indeed the most appropriate choice). We choose $CFL_{u3} = u\Delta t_3 / \Delta x$ and $\Delta t_3 / \Delta t_2$ as the two parameters. Note that we use the particle speed, *u*, rather than the more common artificial maximum eigenvalue, $u + c'_u$ to define the Courant number for the inner step. At small t_2 time steps, the scheme is source term dominated and although $u + c'_u$ is large the effect of the convective term is small; at large t_2 time steps, the inner t_3 becomes steady preconditioned and c_u' is the same order as u. Either of u and $u + c'_u$ can be used for defining CFL. Consequently, we use

$CFL_{u3} = u\Delta t_3 / \Delta x$. We should mention that the differences between them are small. By computing the maximum eigenvalue of matrix $\tilde{G}$ in equation 3.36 numerically over all wave number space, we can obtain a 'maximum eigenvalue' stability plot for the four-sweep DDLGS approximate factorization. Some representative results are plotted in figures 3.3, 3.4 and 3.5.

Figures 3.3, 3.4 and 3.5 present the stability map of the inner iteration for the complete two-dimensional Fourier space for a flow field with a Mach number of 0.01, a number of grids of 100 in each direction and a flow angle of zero. Since the results are based on a preconditioned set of equations, Mach number effects have essentially been scaled out and do not affect the analysis. The flow angle causes significant local changes in the stability contours, but the results shown here are indicative of the characteristics of the DDLGS algorithm. The results are for a Courant number of 80, and a time step ratio, $\Delta t_3 / \Delta t_2$ of unity.

Figure 3.3 is for the I/I/*(we use '*' here because $\nabla \cdot F$ disappears in the stability analysis and the stability results are the same for all orders of accuracy in $\nabla \cdot F$) order system and figures 3.4 and 3.5 are for the I/II/* order system. The only difference between figure 3.4 and figure 3.5 is that the middle one is plotted in the whole wave number region and the lower one is for the small wave number region.

In figures 3.3 and 3.4, the maximum eigenvalue at the high x wave number region is very small over most of the domain but increases sharply in the small wave number region where it becomes highly unstable, which indicates fast convergence in large wave number region and slow convergence in the small wave number region. The key characteristic of these two maps is the strong unstable region in the second and fourth quadrants that is near the origin of Fourier space. This unstable region is characteristic of the conditionally stable DDLGS scheme. As the CFL number is reduced, this region of instability disappears and excellent convergence is observed. The maximum eigenvalues in figures 3.3 and 3.4 are both 1.47 by accident. Details of the unstable region in figure 3.4 are shown in figure 3.5.

Although figures 3.3 and 3.4 share the same characteristics as stated above, they do differ because of the discretization difference. The major difference is that the maximum

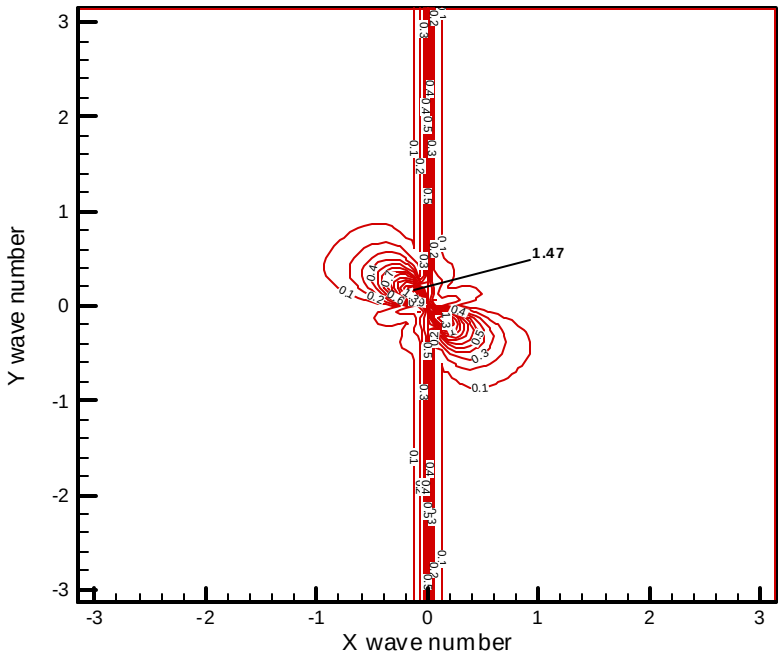

Figure 3.3 I/I/* system, the maximum eigenvalue of inner iteration for four-sweep DDLGS($CFL_{u3} = 80$, $\Delta t_3 / \Delta t_2 = 1$ and M=0.01)

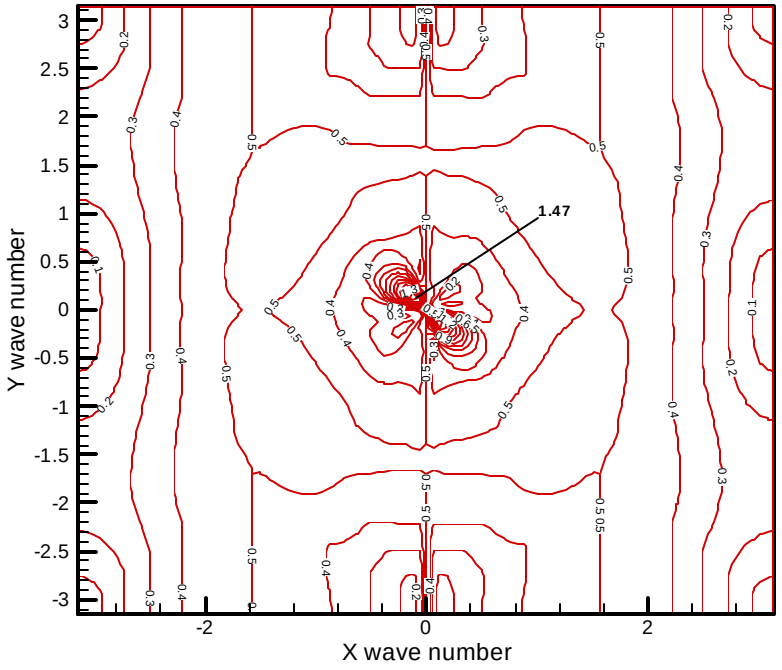

Figure 3.4 I/II/* system, the maximum eigenvalue of inner iteration for four-sweep DDLGS($CFL_{u3} = 80$, $\Delta t_3 / \Delta t_2 = 1$ and M=0.01)

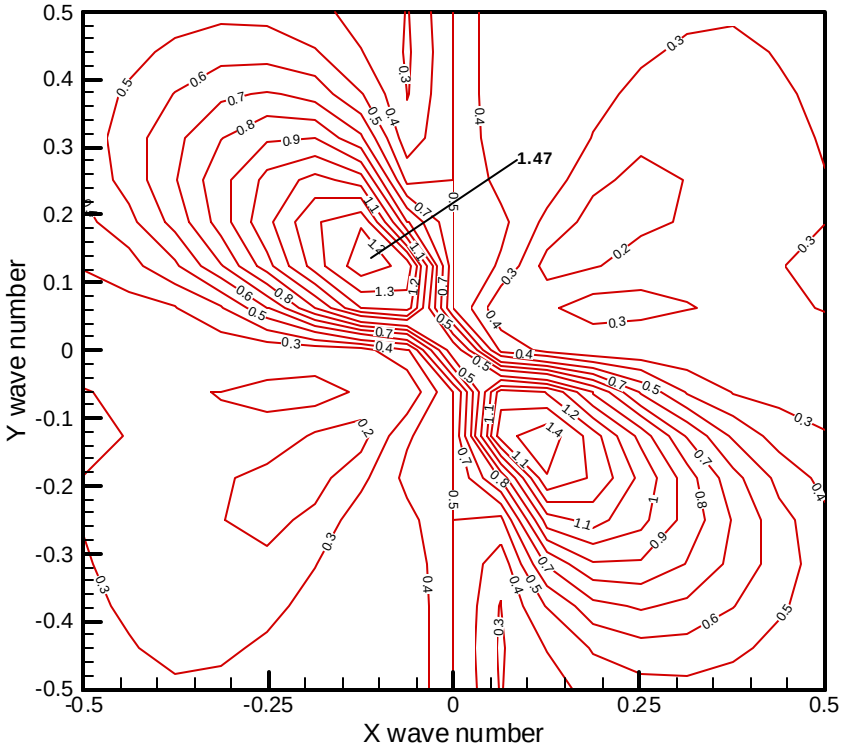

Figure 3.5 I/II/* system in small wave number region, the maximum eigenvalue of inner iteration for four-sweep DDLGS($CFL_{u3} = 80$, $\Delta t_3 / \Delta t_2 = 1$ and M=0.01)

eigenvalue in figure 3.3 is much less than the one in figure 3.4 in the large wave number region, which may mislead us to choose the I/I/II system instead of the I/II/II system. But do not forget, the inner convergence is just part of the convergence of the triple time system. We will show later that the whole convergence of I/II/II system is much better than that of the I/I/II system.

3.3.4 Comparison of Various Ways to Plot the Results

To provide a global overview of the convergence characteristics of the inner time step, we summarize the stability characteristics taken from a large number of calculations like those presented in figure 3.4. To construct these summary figures, we have computed the stability maps for a wide range of two appropriate parameters chosen from the three parameters, CFL_{u3}, CFL_{u2} and $\Delta t_3 / \Delta t_2$, for various orders of accuracy. Note here that the Courant number of the outer t_2 iteration (CFL_{u2}) is based on the particle

speed (the smallest eigenvalue at low Mach number) instead of the maximum eigenvalue. The reason for this is that we do not have a CFL limitation for the outer iteration of our triple time scheme because of the direct solution. At each condition, we have then searched the Fourier space (excluding points on the y-axis since the zero wave number in the x-direction is meaningless) to find the maximum eigenvalue in the entire domain which is defined to be $l_{\max}$, and plotted this maximum on the summary figures. For example, the maximum eigenvalue obtained from figure 3.4 is the 1.47 value noted at the peak of the unstable region. Since the maximum eigenvalue generally controls convergence, these summary plots should provide insight into the expected convergence rate.

Another question we should ask is: which two parameters should we choose to plot the results? Any two of the three parameters, CFL_{u3}, CFL_{u2} and $\Delta t_3 / \Delta t_2 = CFL_{u3} / CFL_{u2}$ could be chosen as the primary variables. From the results of one dimensional analysis, it appears that CFL_{u3} and $\Delta t_3 / \Delta t_2$ are the two right parameters to plot the results (actually they are not as shown later). The quantity, $\Delta t_3 / \Delta t_2$, is the sink term and CFL_{u3} is the CFL for the inner iteration. For now, we choose these two quantities.

Figures 3.6, 3.7 and 3.8 show the I/I/* order direct inversion, I/II/* order direct inversion and I/II/* order DDLGS stability results respectively. In these figures, we use a square symbol and a triangle symbol to identify the transition region, where the square symbol and the triangle symbol are the starting point and the ending point of the transition from the non-preconditioned to the steady preconditioning respectively. This transition region does not change but only moves toward the higher CFL_{u3} region as the sink term increases. The x-axis is the logarithmic value of CFL_{u3} and the Y axis is the largest value of all four eigenvalues of $\tilde{G}$ over the whole wave number space excluding the y-axis. $\Delta t_3 / \Delta t_2$ is an indicator of the magnitude of the sink term. A series of $\Delta t_3 / \Delta t_2$ is plotted, ranging from 0.01 to 100. For simplicity, let us start with the direct inversion.

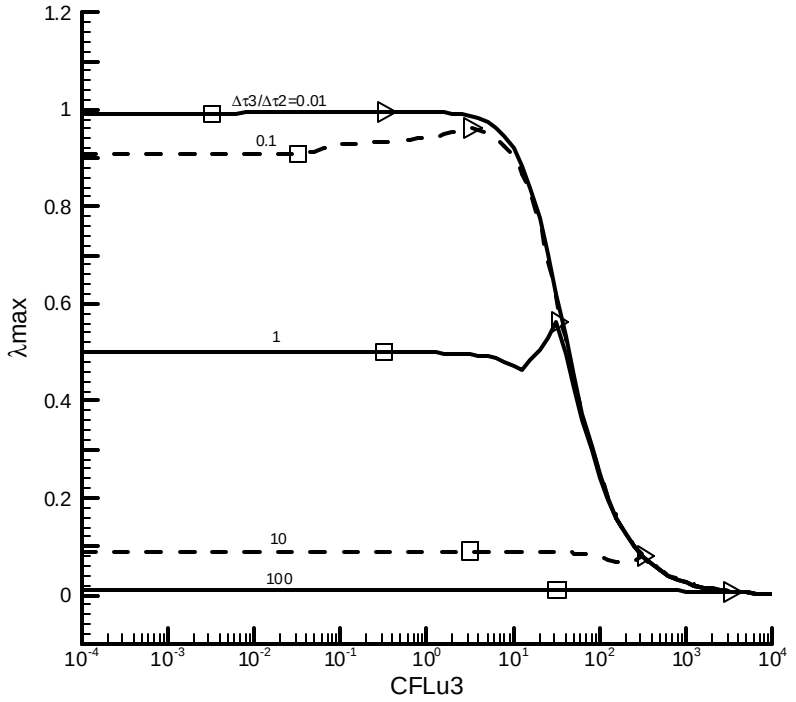

Figure 3.6 I/I/* order of direct inversion stability results of inner t_3 iteration in triple-time method (UPU, Mach number = 0.01, Flow angle = 0, Grids of 100x100)

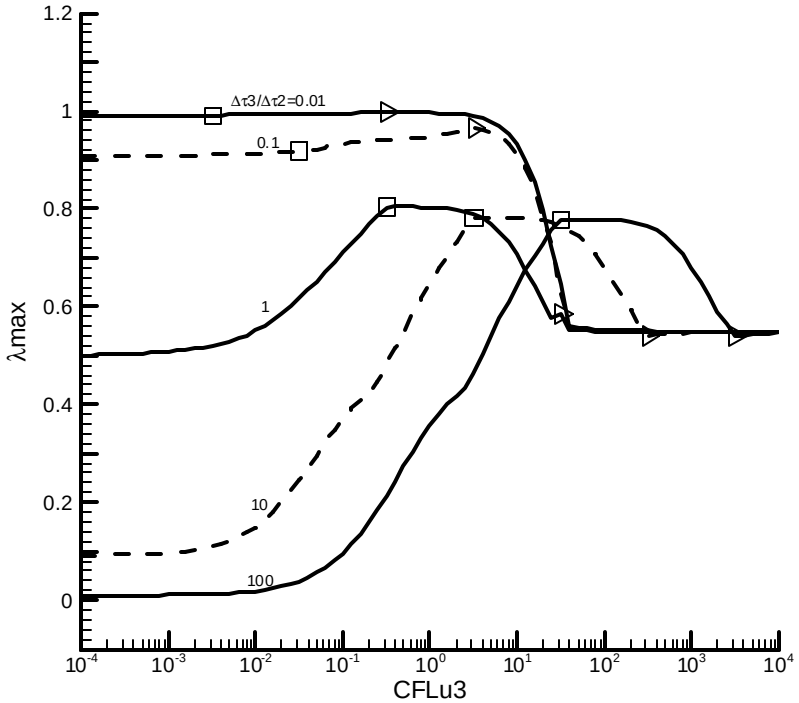

Figure 3.7 I/II/* order of direct inversion stability results of inner t_3 iteration in triple-time method (UPU, Mach number = 0.01, Flow angle = 0, Grids of 100x100)

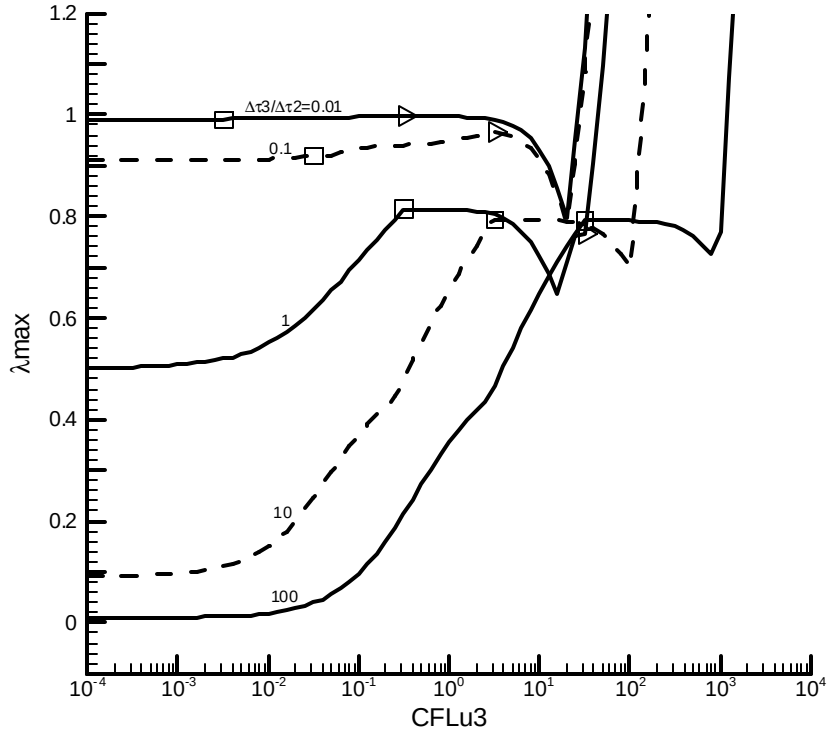

Figure 3.8 I/II/* order of four-sweep DDLGS stability results of inner t_3 iteration in triple-time method (UPU, Mach number = 0.01, Flow angle = 0, Grids of 100x100).

In figure 3.6, the I/I/* order of direct inversion, the uppermost line is the $\Delta t_3 / \Delta t_2 = 0.01$ case, has the same characteristics as the case where the source term introduced by the triple time is absent. As in any stability results, the maximum eigenvalue approaches unity as CFL_{u3} approaches zero, and since we are using a direct inversion and a consistant differencing (I/I/*) the maximum eigenvalue approaches zero as CFL_{u3} approaches infinity. When $\Delta t_3 / \Delta t_2$ is increased, the sink term becomes larger causing the maximum eigenvalue to decrease at small and moderate CFL_{u3}'s. There are bumps at the lines of middle $\Delta t_3 / \Delta t_2$ values ($\Delta t_3 / \Delta t_2$=1 and 0.1). They are caused by the transition of Γ_3 from physical preconditioning to steady preconditioning as can be easily seen from the marked transition region.

Figure 3.7 is I/II/* order of direct inversion and is quite different from the I/I/* order figure because of the inconsistent discretization. At the small CFL_{u3} limit, the

inconsistent I/II/* order results approach the consistent I/I/* order results because the divergence term, $\nabla_D \cdot A_p$ is negligible at the small CFL_{u3} limit, but at large CFL_{u3}, the maximum eigenvalue approaches a finite value instead of zero because of the inconsistent differencing. Also, the bumps caused by the preconditioning transition still exist and are much larger than the I/I/* order case. As the sink term ($\Delta t_3 / \Delta t_2$) increases, these bumps move toward the large CFL_{u3} direction.

Figure 3.8 shows the corresponding stability characteristics of the DDLGS method for the inconsistent I/II/* scheme, which share much in common with the direct inversion results in figure 3.7. At small CFL_{u3} region, the results are exactly the same as the corresponding direct inversion case (figure 3.7) because the approximate factorization error at this region is small. As CFL_{u3} increases, the error introduced by the approximate factorization gets larger so that at some point the scheme becomes unstable. At moderate and large CFL_{u3} region, the instability introduced by the approximate factorization that was seen in figure 3.4 dominates the high-CFL region to such an extent that the consistent and inconsistent DDLGS results are very similar, especially in the intermediate CFL regimes that are of interest for practical calculations.

The results for the DDLGS of the I/I/* order system is very similar to the I/II/* system because of the large approximate factorization error at large CFL_{u3} region. Accordingly, only results for the inconsistent system are given.

In figures 3.6, 3.7 and 3.8, we used CFL_{u3} and $\Delta t_3 / \Delta t_2$ as the two independent parameters. These quantities are very useful for visualizing the effect of the sink term $\Delta t_3 / \Delta t_2$, but are not practical because of the following two reasons. First, we expect that we will have to ramp CFL_{u2} because of the non-linearity at the beginning of the convergence process, but we do not have any knowledge of how to ramp $\Delta t_3 / \Delta t_2$ in the triple time method. Second, figure 3.8 shows that the optimum CFL_{u3} changes with $\Delta t_3 / \Delta t_2$ (excluding the very small CFL_{u3} region ($CFL_{u3} \leq 10^{-3}$)) and a small value beyond the optimum one could result in divergence.

For the first reason mentioned above, CFL_{u2} appears to be a better independent parameter since we have obtained some heuristic experience of how to ramp it from the single time method. The other parameter must then be either $\Delta t_3 / \Delta t_2$ or CFL_{u3}. In order to decide which is better, we compare plots of the stability results for the same approximate factorization case in each of these two cases. The results are shown in figures 3.9 and 3.10. Figure 3.9 uses CFL_{u2} and CFL_{u3} while figure 3.10 uses CFL_{u2} and $\Delta t_3 / \Delta t_2$. The transition region from physical to steady preconditioning is marked by two straight lines in term of CFL_{u2}. These two figures, combined with figure 3.8, show the same data in terms of three different sets of independent parameters.

In figure 3.9 CFL_{u2} and CFL_{u3} are the two independent parameters. At small CFL_{u2} region, the damping of the error is very fast and there is no difference between the different CFL_{u3}'s but these values of CFL_{u2} are clearly not practical for the outer iteration. As CFL_{u2} increases, the damping gets slower and the difference between different CFL_{u3}'s becomes bigger. We can see a corner at the beginning of the transition region caused by the transition from the non-preconditioned to the steady preconditioning. In the transition region, when CFL_{u3} is large ($CFL_{u3} \geq 10$), we can still achieve very good convergence. The convergence rate increases at the beginning but gets worse monotonically as CFL_{u2} increases. For smaller values of CFL_{u3}, the convergence rate gets worse monotonically as CFL_{u2} increases. In the steady preconditioning region, the damping decreases as CFL_{u2} increases ($\Delta t_3 / \Delta t_2$ is becoming smaller), and the scheme even becomes divergent for $CFL_{u3} = 30$. This is evidence of the well-known conditional stability of the DDLGS scheme as shown in figure 3.4. Over the whole CFL_{u2} region, we find that $CFL_{u3} = 20$ is close to an optimum choice and has a pretty good damping for any value of CFL_{u2}. That suggests that we can fix CFL_{u3} when ramping CFL_{u2}, reducing the two variables to only one.

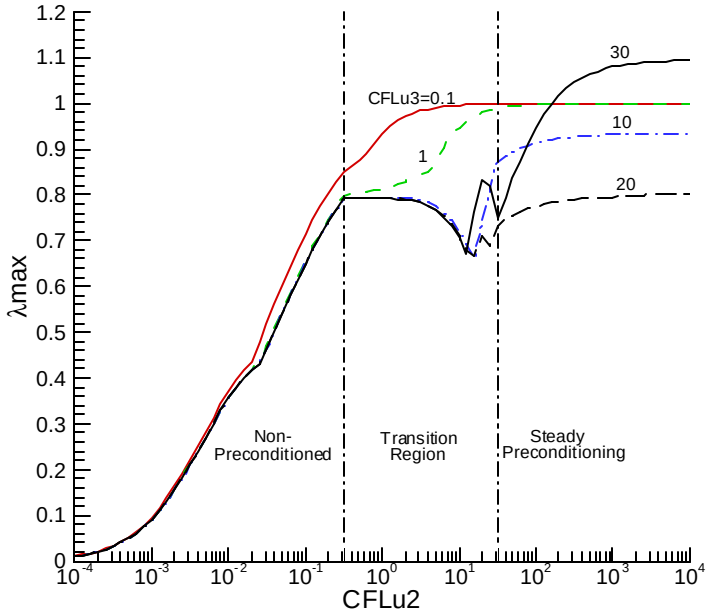

Figure 3.9 Stability of inner iteration in triple-time method with DDLGS of the t_3 operator. UPU, I/II/* order system, Mach number = 0.01, Flow angle = 0. CFL_{u2} and CFL_{u3} are the two independent parameters.

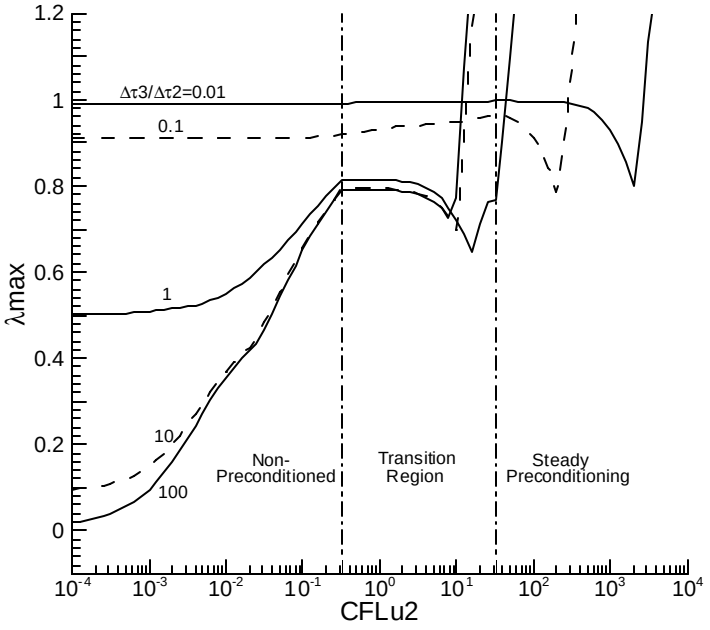

Figure 3.10 Stability of inner iteration in triple-time method with DDLGS) of the t_3 operator. UPU, I/II/* order system, Flow Mach number = 0.01, Flow angle = 0. CFL_{u2} and $\Delta t_3 / \Delta t_2$ are the two independent parameters.

In figure 3.10 we use CFL_{u2} and $\Delta t_3 / \Delta t_2$ as two parameters. This figure is very similar to figure 3.8. For every value of the sink term $\Delta t_3 / \Delta t_2$, the maximum eigenvalue $l_{\max}$ increases as CFL_{u2} or CFL_{u3} increases at the beginning, then decreases and finally increases extremely fast to the divergence. Although the sink term pushes the convergence and the CFL_{u3} limit for divergence in the right direction, the scheme finally diverges no matter how strong the sink term is. Thus, if we ramp CFL_{u2} at the beginning of a computation, we will have to use a different value of $\Delta t_3 / \Delta t_2$ for each CFL_{u2}. We can not find a constant $\Delta t_3 / \Delta t_2$ value to make the scheme stable over all CFL_{u2} value.

Consequently, CFL_{u2} and CFL_{u3} as shown in figure 3.9 appears to be the best set of independent parameters. In the rest of the thesis, we use CFL_{u2} and CFL_{u3} as the primary parameters instead of CFL_{u3} and $\Delta t_3 / \Delta t_2$, or CFL_{u2} and $\Delta t_3 / \Delta t_2$.

3.3.5 Comparison of the Different Sets of Γ's

In order to compare the stability characteristics of the four sets of Γ's, we do the inner t_3 stability analysis according to equation set 3.36 using the method of figure 3.9 to compare the results. The three matrices, Γ_p, Γ_s and Γ_u are different only at low Mach number and the I/II/* order system is of most interest for its expected fast outer convergence since the I/II/II order system, one member of the family of I/II/* order system provides consistent differencing for the outer time step and the residual. Consequently, only results for Mach=0.01 and I/II/* system are presented.

Figures 3.11 through 3.14 are the plots for these four sets of Γ's . The results are plotted in two different ways. One way is the CFL_{u2} versus $l_{\max}$ plot for a series of CFL_{u3}, the same way plotted in figure 3.9.This is shown as the upper plot in each figure.

The second way for presenting the results is to plot CFL_{u2} versus the number of iterations for ten orders of inner convergence which is obtained by the formulation

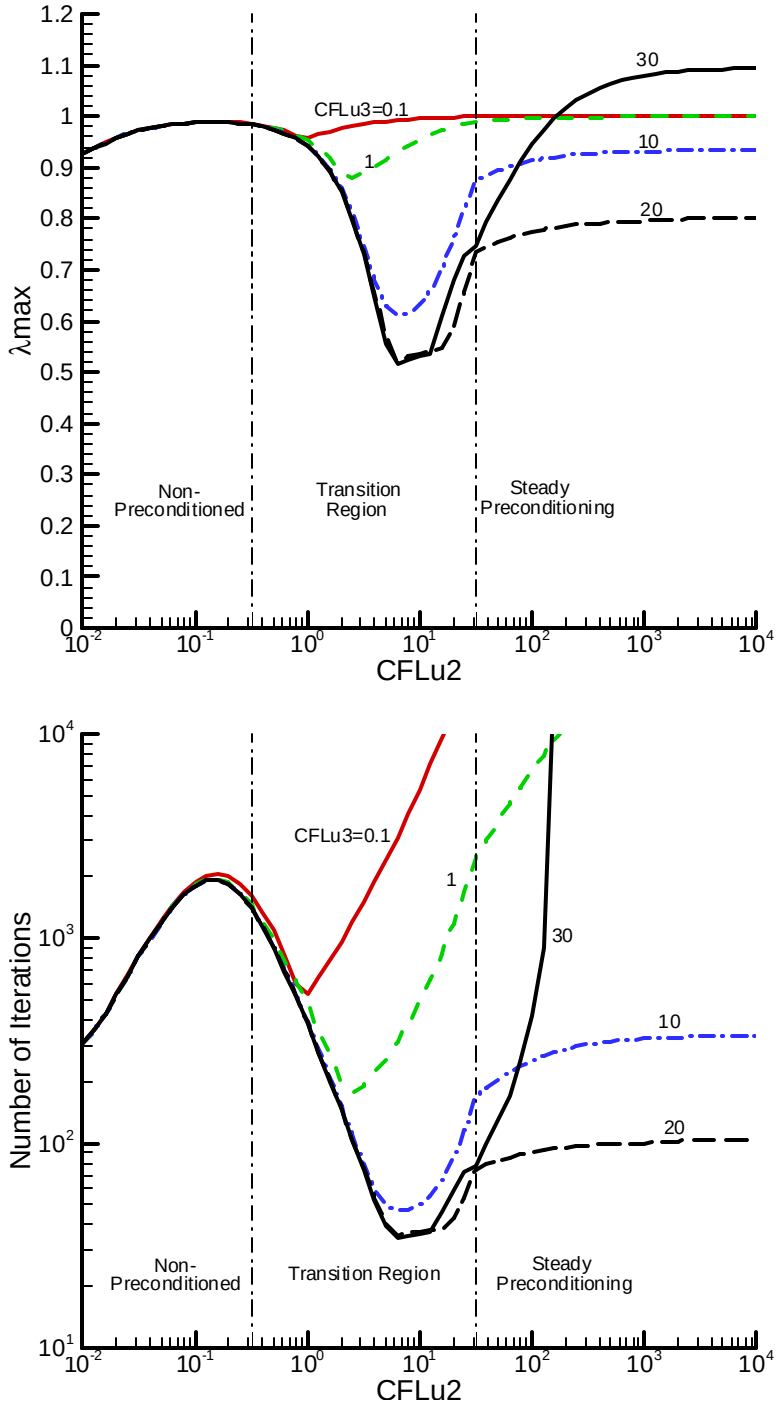

Figure 3.11 Stability of inner t_3 iteration for UPS, I/II/* order system, four-sweep DDLGS approximate factorization, Flow Mach number = 0.01, Flow angle = 0. The first plot has a Y axis of $l_{\max}$; The second plot has a Y axis of number of iterations for ten orders convergence.

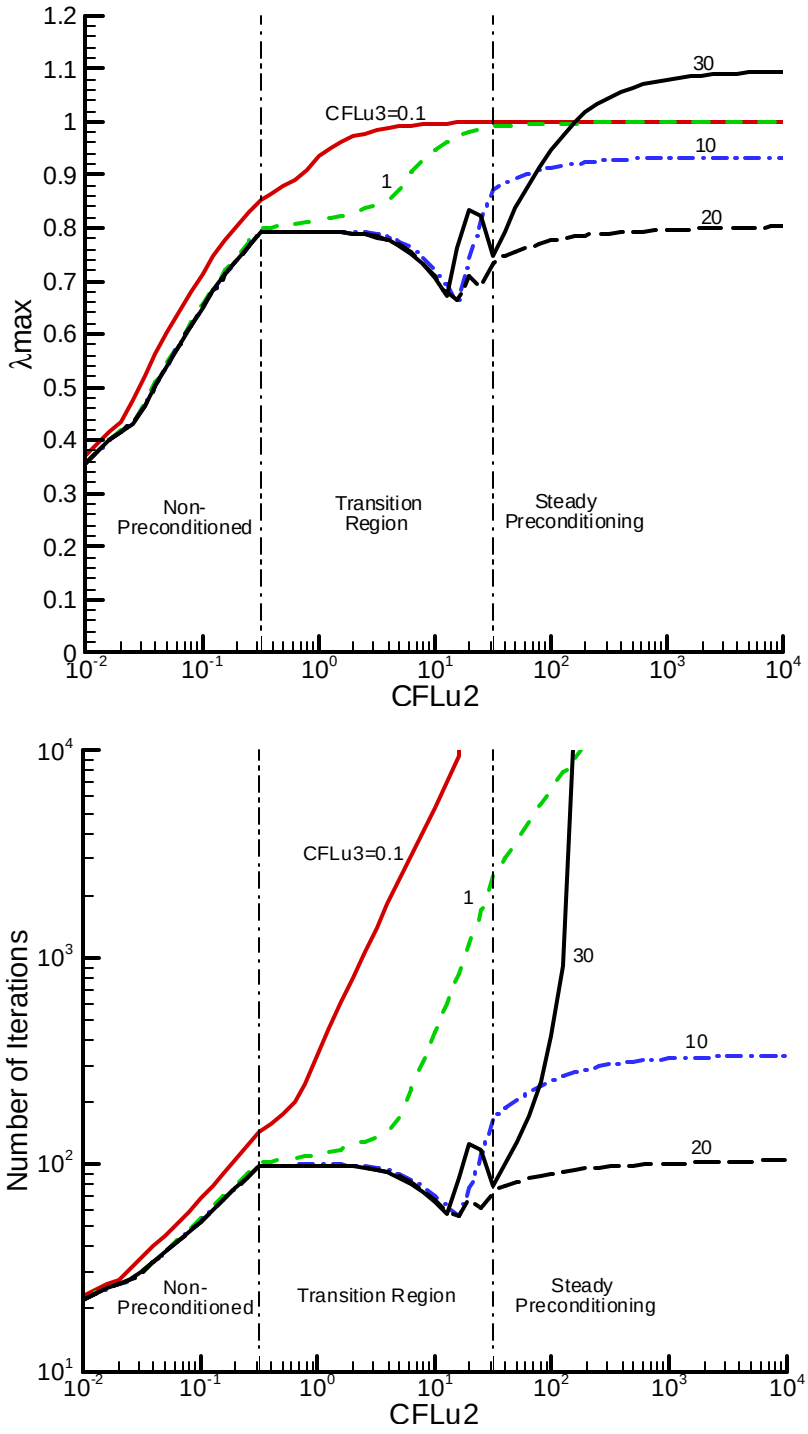

Figure 3.12 Stability of inner t_3 iteration for UPU, I/II/* order system, four-sweep DDLGS approximate factorization, Flow Mach number = 0.01, Flow angle = 0. The first plot has a Y axis of $l_{\max}$; The second plot has a Y axis of number of iterations for ten orders convergence

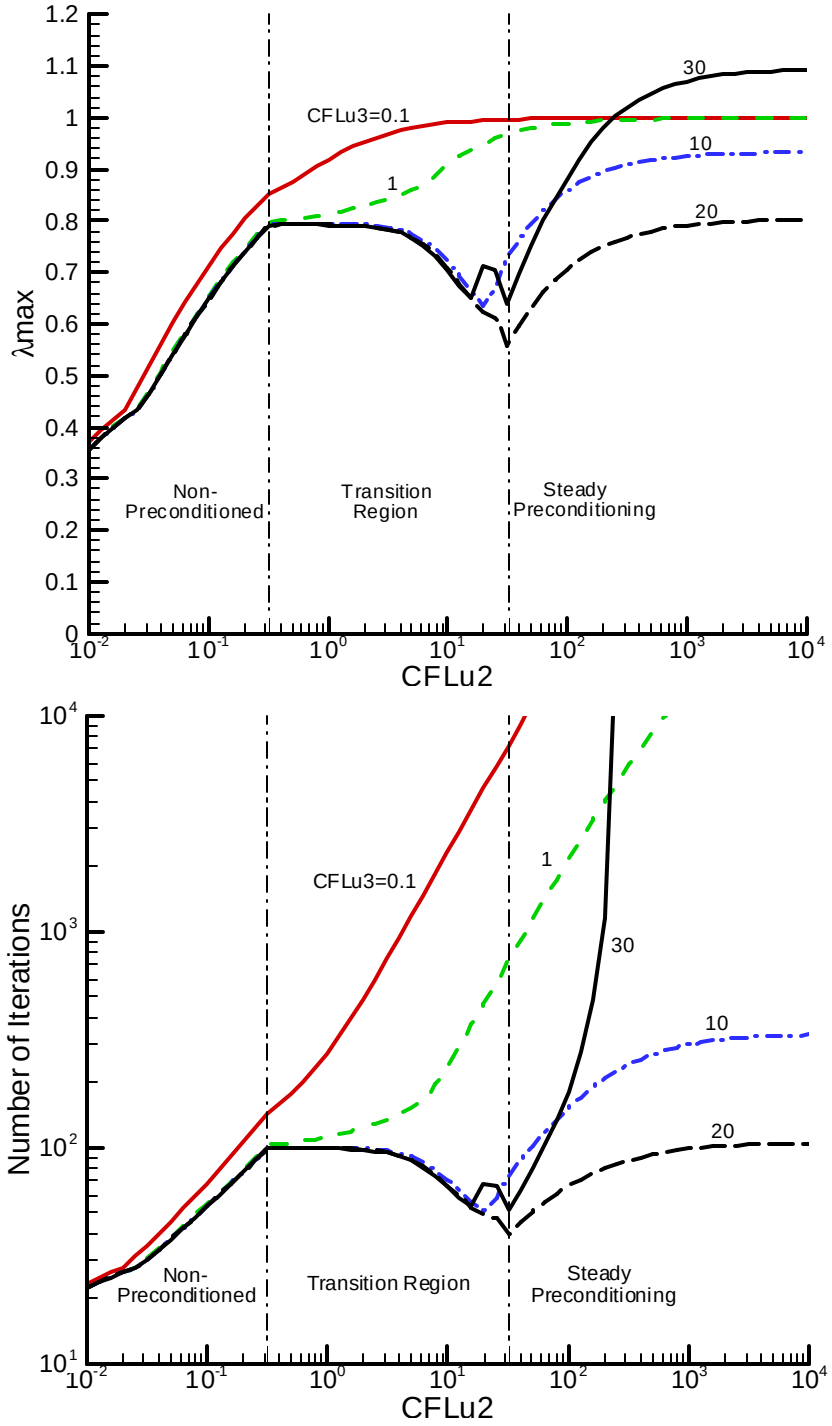

Figure 3.13 Stability of inner t_3 iteration for UUU, I/II/* order system, four-sweep DDLGS approximate factorization, Flow Mach number = 0.01, Flow angle = 0. The first plot has a Y axis of $l_{\max}$; The second plot has a Y axis of number of iterations for ten orders convergence

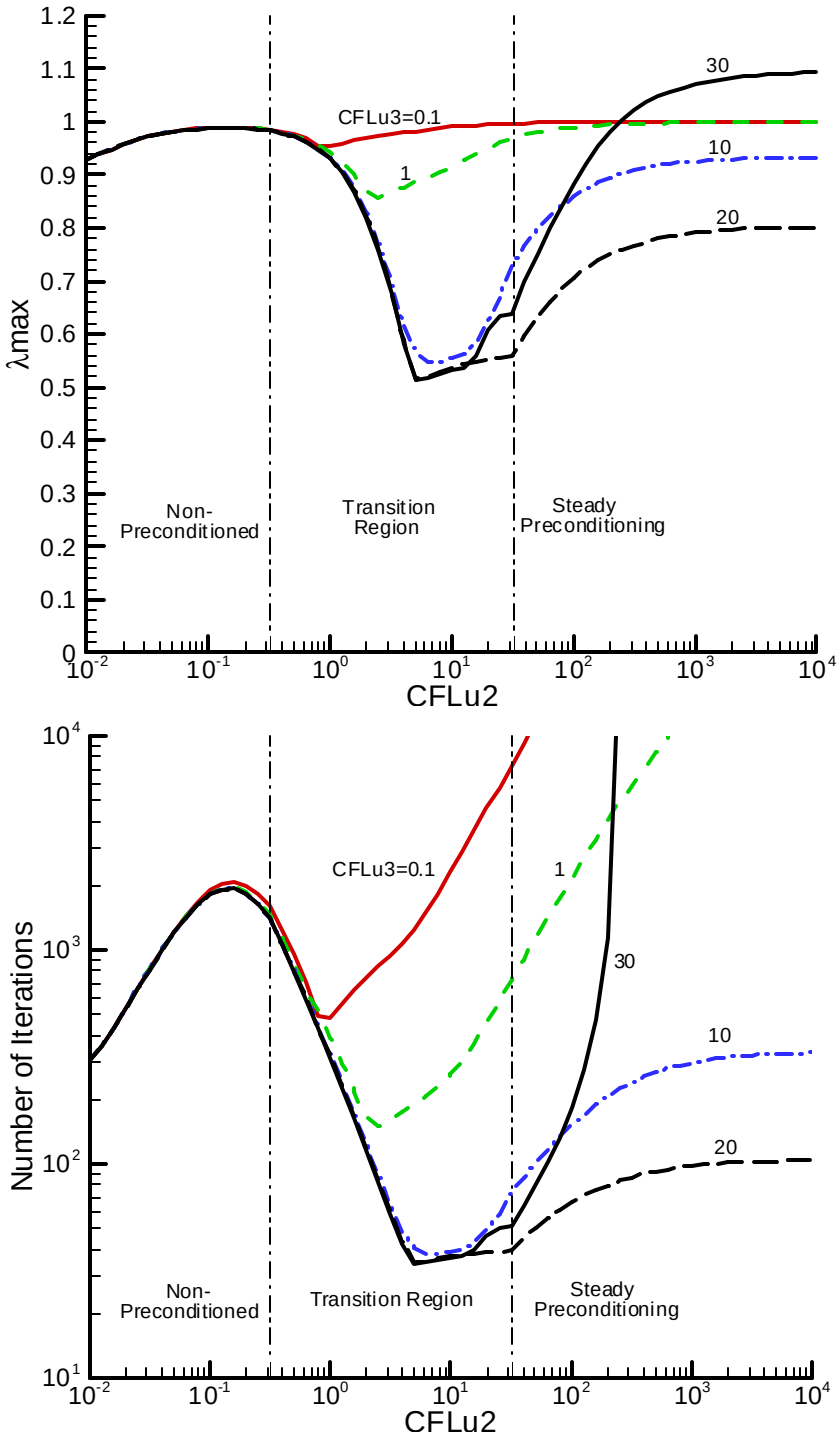

Figure 3.14 Stability of inner t_3 iteration for UUS, I/II/* order system, four-sweep DDLGS approximate factorization, Flow Mach number = 0.01, Flow angle = 0. The first plot has a Y axis of $l_{\max}$; The second plot has a Y axis of number of iterations for ten orders convergence

$-10/\log_{10}(l_{\max})$ ($l_{\max}$ is the $l_{\max}$ in the first way). The number of iterations is more meaningful in application to the real computations and in addition gives a clearer distinction between small (but very important) changes in the eigenvalue as its eigenvalue approaches unity. The second way is plotted in the lower plot of each figure. Again, the x-axis is the logarithmic scale of CFL_{u2} based on velocity u instead of the u+c. One difference in these plots from figure 9 where CFL_{u2} ranges from 10^{-5} to 10^{5} to give a clear picture in the very large and very small CFL_{u2} regions, the range of CFL_{u2} here will be narrowed to a more interesting region from 10^{-2} to 10^{4}. The y-axis is again the maximum value of the maximum eigenvalue of the amplification matrix $\tilde{G}$ over both X and Y wave number space excluding the value at Y axis.

The transition region from physical to steady preconditioning is marked by two straight lines in terms of CFL_{u2}. At the non-preconditioned region, $r'_{pu} = r_p$ and unsteady preconditioning becomes non-preconditioned, hence both 'UPS' and 'UUS' become 'PPS', and both 'UPU' and 'UUU' become 'PPP'. At the steady preconditioning region, $r'_{pu} = r'_{ps}$ and unsteady preconditioning becomes steady preconditioning, thus both 'UPS' and 'UPU' become 'SPS', and both 'UUS' and 'UUU' become 'SSS'.

In figure 3.11, the 'UPS' case, at the non-preconditioned region (small CFL_{u2}), the convergence rate is very slow because of the inconsistent preconditioning ('UPS' becomes 'PPS' for small values of CFL_{u2}) and the differences between different values of CFL_{u3} are too small to be seen from the upper plot but can be observed in the lower plot. After entering the transition region, the difference between different values of CFL_{u3} appears, the convergence rate increases very fast for large values of CFL_{u3} and decreases at the end of the transition region, and gets worse as CFL_{u3} decreases. In the low CFL_{u2} part of the steady preconditioning region, at the beginning, the number of iterations increases as CFL_{u2} increases and then becomes constant as CFL_{u2} becomes large enough. Again, the $CFL_{u3} = 30$ case diverges for very large values of CFL_{u2}. Also, we can see that the results of the $CFL_{u3} = 20$ line are almost optimum one over all

ranges of CFL_{u2}.

Figure 3.14, the 'UUS' case, is very similar to figure 3.11, the 'UPS' case.

In figure 3.12, the first plot uses the same data as figure 3.9. Accordingly the analysis will not be repeated here and can be referred to section 3.3.3. Comparison between figures 3.11 and 3.12 shows that, for small values of CFL_{u2} region, the 'UPU' converges much faster than 'UPS' because of its consistent preconditioning ('UPU' becomes 'PPP' for small values of CFL_{u2}), while for large values of CFL_{u2}, they are very similar. Again, Figure 3.13, the 'UUU' case, is very similar to the 'UPU' case in figure 3.12.

An important point to note for all four figures 3.11 through 3.14 is that in all cases the value $CFL_{u3} = 20$ is very close to the optimum value for all conditions. Consequently we will use $CFL_{u3} = 20$ as 'optimum'.

To compare the four sets of Γ's, we plot the lines of this optimum CFL_{u3} of twenty for all four upper plots together in figure 3.15. Again we show $l_{\max}$ in the upper plot and the number of iterations in the lower plot of figure 3.15. From figure 3.15, we can see that, in the small CFL_{u2} region (non-preconditioned), 'UPS' is exactly the same as 'UUS' because both 'UPS' and 'UUS' become 'PPS', and 'UPU' is exactly the same as 'UUU' because both 'UPU' and 'UUU' become 'PPP'. 'PPP' (UPU and UUU) has a better convergence than 'PPS' (UUS and UPS) because of its consistent preconditioning matrices. At large CFL_{u2} region (steady preconditioning region), 'UPS' is exactly same as 'UPU' because both 'UPS' and 'UPU' become 'SPS', and 'UUS' is exactly the same as 'UUU' because both 'UUS' and 'UUU' become 'SSS'. 'SSS'(UUS and UUU) has better convergence than 'SPS' because of its consistent precondition matrices. At extremely large CFL_{u2} region (greater than 1000), all four cases become identical. That is due to the vanishing effect of Γ_2 at very small values of $\Delta t_3 / \Delta t_2$ (large CFL_{u2}).

From both the upper plot and lower plot of figure 3.15, we can see that 'UUU' is the best except for a small region at moderate CFL_{u2} values where the convergence is worse than 'UUS' and 'UPS'. But the cost is small compared with its large gain from the

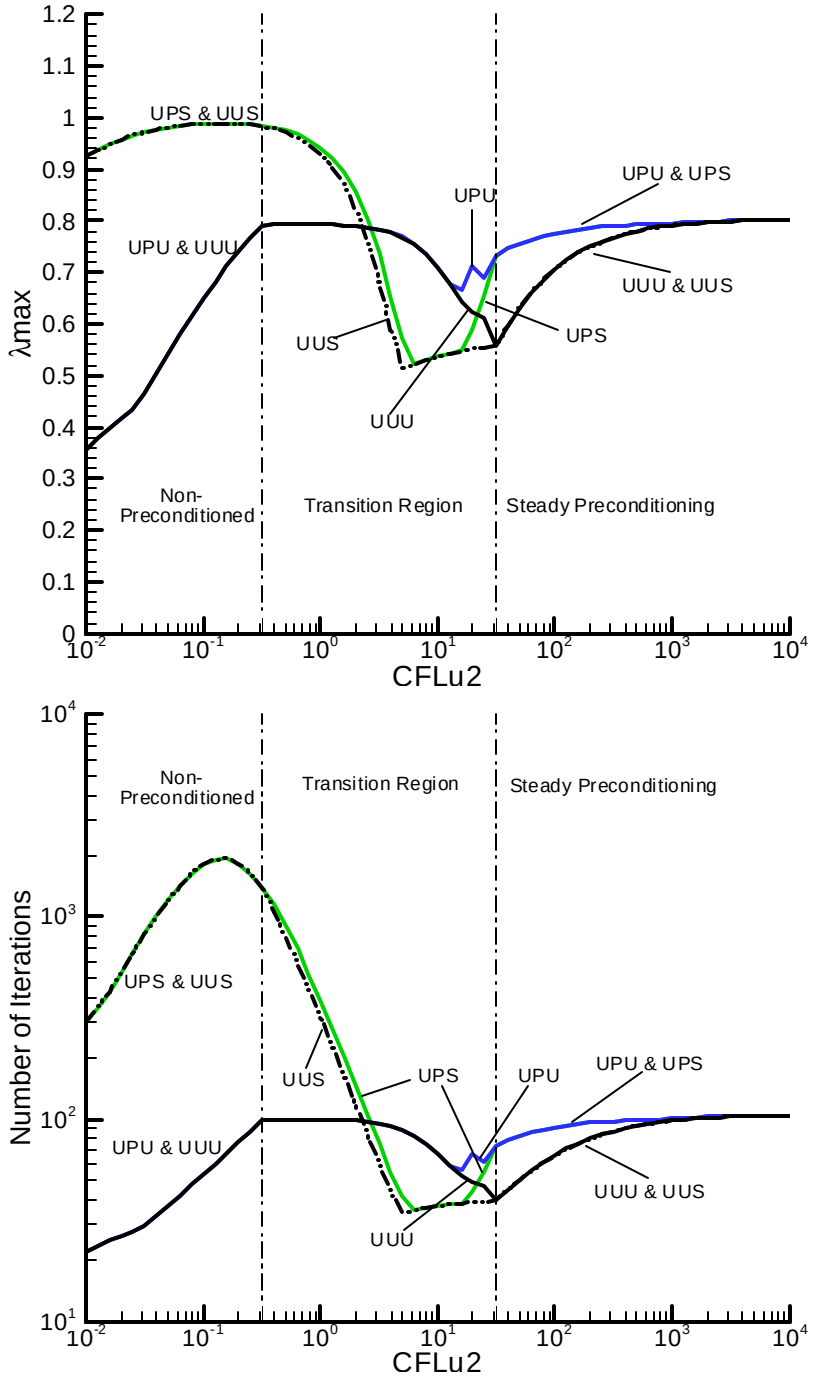

Figure 3.15 Comparison of the inner t_3 stability between the 4 sets of Γs for the optimum $CFL_{u3} = 20$, I/II/* order system, four-sweep DDLGS approximate actorization, Flow Mach number = 0.01, Flow angle = 0. The first plot has a Y axis of $l_{\max}$; The second plot has a Y axis of number of iterations for ten orders convergence.

small CFL region. ‘UPU’ requires slightly more iterations than ‘UUU’ and is also a possible choice. ‘UUS’ and ‘UPS’ are our last choice because of their extremely large number of iterations at small CFL_{u2}.

It is important to realize, however, that the inner convergence is not enough to evaluate the total convergence. We have to take into account the outer convergence besides the inner convergence.

3.4 Outer Stability Analysis for Direct Inversion

We have discussed the inner stability analysis in the last subsection and found that ‘UUU’ case has the best inner convergence over most CFL_{u2} values. The ‘UPU’ case is slightly worse than the ‘UUU’. The ‘UPS’ and ‘UUS’ methods have very poor convergence at small values of CFL_{u2}, but are the best in the middle CFL_{u2} between three and thirty. Since the triple time scheme includes both the inner convergence and the outer convergence, it is not complete if we only consider the inner convergence. In this and next subsection, we perform the stability analysis of the outer convergence. The outer convergence depends on the number of inner iterations. For an infinite number of inner iterations, the outer convergence becomes the convergence of the direction inversion of equation 2.19b, which is the best we can achieve for the outer convergence. In this subsection, we will discuss the stability analysis of the direct inversion (an infinite number of inner iterations) of equation 2.19b. The stability analysis for finite number of inner iterations will be discussed in the next subsection.

The stability analysis for the direct inversion can begin from the outer iteration equation 2.19b (re-labeled as 3.45)

$$\left[\frac{\Gamma_2}{\Delta t_2}+\left(\nabla_D\cdot A_p\right)_2\right]^n \Delta Q_p = -\left(\nabla_D\cdot F\right)^n \tag{3.45}$$

For the outer iteration, the time level ‘n’ remains in the stability formulation as it is the variable we are updating and we must evaluate the discretized divergence of flux $\left(\nabla_D\cdot F\right)^n$. For stability purposes, we again take the Jacobian, A_p, as a constant,

therefore allowing us to express the divergence of the flux as the divergence of the Jacobian times the primary variable

$$\left(\nabla_D \cdot F\right) = \left(\nabla_D \cdot A_p\right)_1 Q_p \tag{3.46}$$

Here we have used the subscript '1' on the discretized divergence to indicate the t_1 pseudo time level.

To perform the stability analysis, we define the outer amplification factor G as

$$Q_p^{n+1} = GQ_p^n \tag{3.47}$$

Substituting equations 3.46 and 3.47 into equation 3.45 and canceling out Q_p we obtain

$$\left[\frac{\Gamma_2}{\Delta t_2} + \left(\nabla_D \cdot A_p\right)_2\right]^n (G - I) = -\left(\nabla_D \cdot A_p\right)_1 \tag{3.48}$$

From equation 3.48 we can solve for G, and compute the maximum eigenvalue and number of iterations to converge ten orders of magnitude and plot versus CFL_{u2} in the same way as we have done for the inner stability analysis.

Since the outer iteration formulation only involves two pseudo times, t_1 and t_2, and does not depend on the third pseudo time, t_3, we use an '*' to indicate the absence of Γ_3 or t_3. For example, the outer iterations of the 'UPS', 'UUS', 'UUU' and 'UPU' cases are expressed as '*PS', '*US', '*UU', '*PU' respectively. Also, the I/I/I, I/I/II and I/II/II order of accuracy of the inner iteration become */I/I, */I/II and */II/II order of accuracy at outer iteration respectively.

For the same reason as for the inner stability analysis, only the results for M=0.01 are presented. Figure 3.16 shows the outer stability of the direction inversion for the '*PU' case. All three order systems, '*/I/I','*/I/II' and '*/II/II', are shown in the same plot. From the plot, we can see that the two consistent systems, '*/I/I' and '*/II/II', are exactly the same over the entire wave number space. This shows that consistent differencing gives similar convergence rate no matter what order of accuracy is used. They both show stiffness in the small CFL_{u2} region and extremely fast damping at large CFL_{u2}.

The inconsistent '*/I/II' system is exactly the same as the two consistent systems at

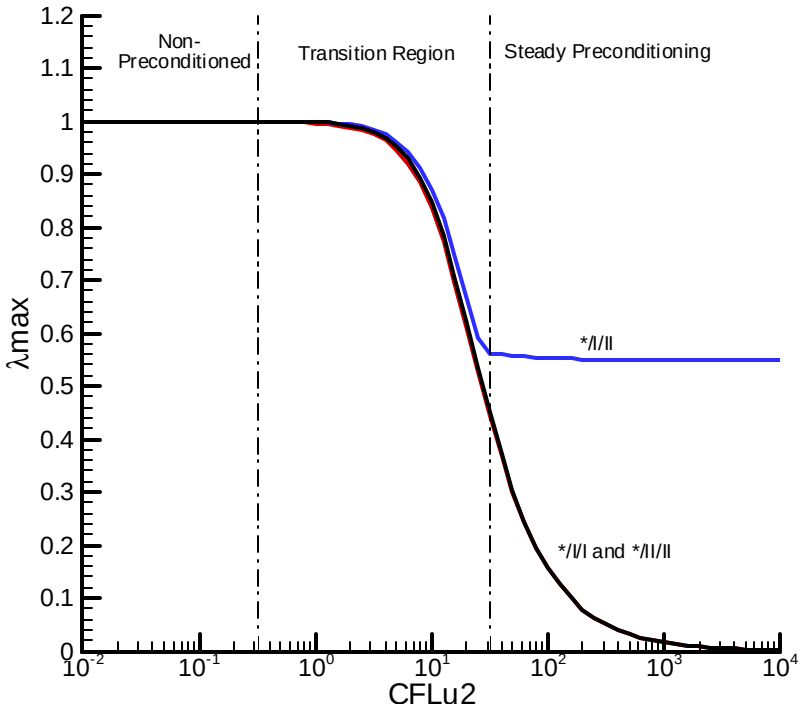

Figure 3.16 Outer Stability with Euler Implicit (Direct Inversion), *PU, Mach number = 0.01, Flow angle = 0.

small CFL_{u2}, but shows much smaller damping at large CFL_{u2} because the inconsistent discretization. This suggests a weakness of the inconsistent iterative DDLGS system discussed in section 2.9.

The results for '*PS', '*US' and '*UU' are very similar to that of '*PU' and hence are not presented. Instead, we plot the results of '*/II/II' system for all four sets of Γs in one plot in figure 17.

In figure 17, again the transition region is shown on the plot for reference. The 'U' quantity degenerates to the 'P' quantity at small CFL_{u2} and degenerates to the 'S' quantity at large CFL_{u2}. Accordingly, in the non-preconditioned region, both the system '*PU' and the system '*UU' are the same as the system '*PP'. Similarly both the system '*PS' and the system '*US' become the system '*PS'. In the steady preconditioning region, both the system '*US' and the system '*UU' become the system '*SS', and the '*PS' and the '*PU' systems become the '*PS' system.

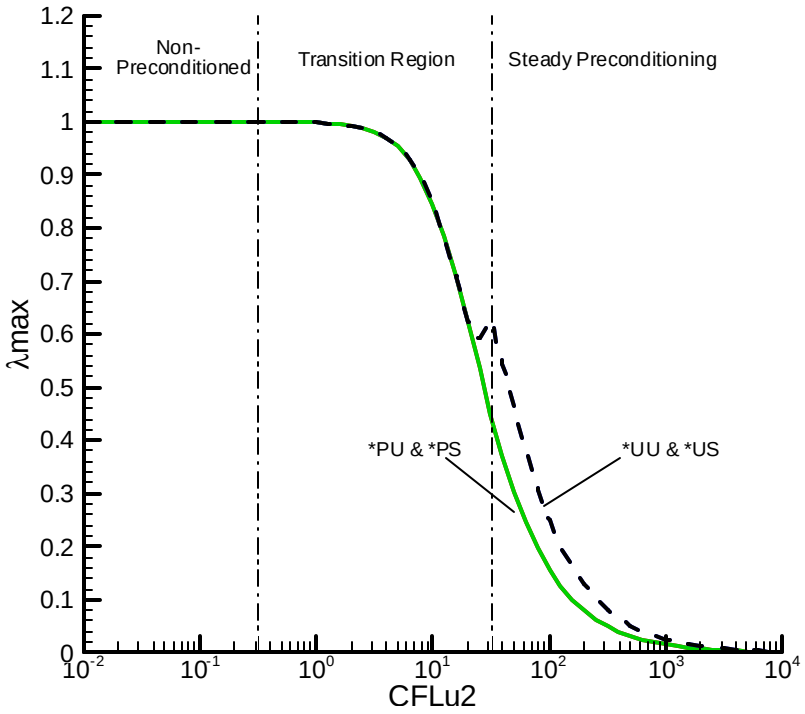

Figure 3.17 Comparison of outer stability with direct inversion between the four sets of Γ's , */II/II order (and */I/I), Mach number = 0.01, Flow angle = 0.

There is no difference between the four sets of Γs at small CFL_{u2} because they all approach to unity. In the steady preconditioning region at large CFL_{u2}'s, '*PU' is exactly the same as '*PS' and '*UU' are exactly the same as '*US' because the quantity 'U' degenerates to 'S' at this region. Also, the systems '*PU' and '*PS' have better damping than the systems '*UU' and '*US'. In the transition region, again, '*PU' is exactly the same as '*PS' and '*UU' are exactly the same as '*US', which indicates that there is no difference in convergence between the unsteady preconditioning and steady preconditioning.

Generally, we do not have to converge the inner iteration to machine accuracy. We want to use as few inner iterations as possible to save CPU time provided the outer convergence is not harmed. So it will be very helpful if we can provide the overall stability for finite number of inner iterations, which is presented in the next subsection.

3.5 Outer Stability with Finite Number of Iterations

The triple time scheme contains an inner iteration and an outer iteration. The number of inner iterations obviously affects the convergence of the outer iteration. In general, the smaller the number of inner iterations we use, the more outer iterations are needed. Also, if the number of inner iterations is increased beyond some limit, we can expect that the inner iteration will no longer affect the outer convergence. Large numbers of inner iterations are expensive and unnecessary, so there must be an optimum number of inner and outer iterations that provides the best global convergence. In this section, we will address the number of inner iterations by using stability analysis of the outer iteration for a finite number of inner iterations. The difference between the outer stability analysis with finite number of iterations and the inner stability analysis is that the inner stability amplification factor is based on the last inner iteration results while the outer stability amplification factor is based on the last inner iteration variable. In this subsection, we derive the formula for the outer stability amplification factor for a given number of inner iterations. The formulation is as follows.

Starting with the approximately factorized triple time equation 2.47 and substituting the triple time residual equation 2.22a into equation 2.47, we obtain

$$\left(L_3 + Err_{4S}\right)\Delta\tilde{Q}_p = -\left(\nabla_D \cdot F\right)^n - \left[\frac{\Gamma_2}{\Delta t_2}(\tilde{Q}_p^m - Q_p^n) + \left(\nabla_D \cdot A_p\right)_2^n(\tilde{Q}_p^m - Q_p^n)\right] \tag{3.49}$$

where $\Delta\tilde{Q}_p = \tilde{Q}_p^{m+1} - \tilde{Q}_p^m$. The approximate factorization error Err_{4S}, is given by equation 2.48

$$Err_{4S} = J_{i,j-1}D_y^{-1}J_{i,j+1}D^{-1}J_{i-1,j}D_x^{-1}J_{i+1,j} \tag{2.48}$$

and the Jacobian coefficients J's are given by equation set 2.53 (or equation set 2.31). For the combined inner and outer stability analysis, the quantities $\tilde{Q}_p^m$ and Q_p^n are variables that must be retained.

We define the amplification factor of the solution of mth inner iteration $\tilde{Q}_p^m$ with respect to the solution of the last outer iteration Q_p^n as $\tilde{G}^m$, where the superscript 'm'

on $\tilde{G}^m$ indicates a series of intermediate amplification factors instead of an exponent. Accordingly we have

$$\tilde{Q}_p^m = \tilde{G}^m Q_p^n \tag{3.50}$$

Again taking the Jacobian matrices as constants we express $(\nabla_D \cdot F)^n = (\nabla_D \cdot A_p)_1^n Q_p^n$, substitute this and equation 3.50 into equation 3.49 and use the Fourier transformation. Canceling out the common factor, Q_p^n, we obtain

$$[F(L_3) + F(Err_{4S})](\tilde{G}^{m+1} - \tilde{G}^m) = -F\left[(\nabla_D \cdot A_p)_1^n\right] - \left\{\frac{\Gamma_2}{\Delta t_2} + F\left[(\nabla_D \cdot A_p)_2^n\right]\right\}(\tilde{G}^m - I) \tag{3.51}$$

where the Fourier transformation operator $F(L_3)$ is given by equation 3.37. The operator $F\left[(\nabla_D \cdot A_p)_3^n\right]$ in equation 3.37 is discretized as first order and accordingly given by equation 3.39. For I/I/I system, the operators $F\left[(\nabla_D \cdot A_p)_1^n\right]$ and $F\left[(\nabla_D \cdot A_p)_2^n\right]$ are both first order and equal to $F\left[(\nabla_D \cdot A_p)_I^n\right]$ in equation 3.39. For I/II/II system, the operators $F\left[(\nabla_D \cdot A_p)_1^n\right]$ and $F\left[(\nabla_D \cdot A_p)_2^n\right]$ are both second order and equal to $F\left[(\nabla_D \cdot A_p)_{II}^n\right]$ in equation 3.43. For I/I/II order system, the operator $F\left[(\nabla_D \cdot A_p)_1^n\right]$ is second order and given by equation 3.43, but $F\left[(\nabla_D \cdot A_p)_2^n\right]$ is first order and given by equation 3.39. The Fourier transformation of the error term is given by taking the Fourier transformation of equation 2.48. This procedure is straightforward. The results are not given because it is two messy.

Moving the terms with $\tilde{G}^m$ on the left hand side of equation 3.51 to the right hand side, gives,

$$[F(L_3) + F(Err_{4S})]\tilde{G}^{m+1} = -F\left[(\nabla_D \cdot A_p)_1^n\right] + [F(L_3) + F(Err_{4S})]\tilde{G}^m + \left\{\frac{\Gamma_2}{\Delta t_2} + F\left[(\nabla_D \cdot A_p)_2^n\right]\right\}(I - \tilde{G}^m) \tag{3.52}$$

For generality, write the initial condition as an arbitrary function of the solution Q_p^n at the previous t_2 time step.

$$\tilde{Q}_p^0 = VQ_p^n \tag{3.53a}$$

where V is a vector function that is independent of Q_p^n. In the present analysis, we have taken the initial condition of the inner iteration as $\tilde{Q}_p^0 = Q_p^n$ so that V is the unit matrix. Substituting equation 3.53a into 3.50 for m=0 and canceling out Q_p^n, we obtain,

$$\tilde{G}^0 = V \tag{3.53b}$$

To find the results of $\tilde{G}^m$ for m inner iteration steps, m=0, 1, 2, …, m, we begin with the initial condition of the inner iteration where m=0 so that $\tilde{Q}_p^0 = VQ_p^n$ and $\tilde{G}^0 = V$, and proceed through the first inner iteration to find $\tilde{Q}_p^1$ (for m=0). Substituting $\tilde{G}^0 = V$ into equation 3.52 to obtain the first inner iteration relationship

$$[F(L_3) + F(Err_{4S})]\tilde{G}^1 = -F\left[\left(\nabla_D \cdot A_p\right)_1^n\right] + [F(L_3) + F(Err_{4S})]V + \left\{\frac{\Gamma_2}{\Delta t_2} + F\left[\left(\nabla_D \cdot A_p\right)_2^n\right]\right\}(I - V) \tag{3.54a}$$

Repeat this process for the second inner iteration step, m=1, and using the result for the first inner iteration step, m=0, we obtain the second inner iteration relationship,

$$[F(L_3) + F(Err_{4S})]\tilde{G}^2 = -F\left[\left(\nabla_D \cdot A_p\right)_1^n\right] + [F(L_3) + F(Err_{4S})]\tilde{G}^1 + \left\{\frac{\Gamma_2}{\Delta t_2} + F\left[\left(\nabla_D \cdot A_p\right)_2^n\right]\right\}\left(I - \tilde{G}^1\right) \tag{3.54b}$$

Extending the results to the last inner iteration step when $m = M - 1$, gives

$$[F(L_3) + F(Err_{4S})]\tilde{G}^M = -F\left[\left(\nabla_D \cdot A_p\right)_1^n\right] + [F(L_3) + F(Err_{4S})]\tilde{G}^{M-1} + \left\{\frac{\Gamma_2}{\Delta t_2} + F\left[\left(\nabla_D \cdot A_p\right)_2^n\right]\right\}\left(I - \tilde{G}^{M-1}\right) \tag{3.54c}$$

The matrix, $\tilde{G}^M$, then becomes the outer amplification factor for M inner iterations steps, i.e., $\tilde{G}^M = G$, where $Q_p^{n+1} = \tilde{Q}_p^M$ and $Q_p^{n+1} = GQ_p^n$.

One representative result for the complete inner-outer iteration process for the UPU system is plotted in figure 3.18 for a series of different values of M (number of inner iterations). As was done before, the results are plotted in two different ways, the maximum eigenvalue, $l_{\max}$ versus $\log 10(CFL_{u2})$ (the upper plot of figure 3.18) and

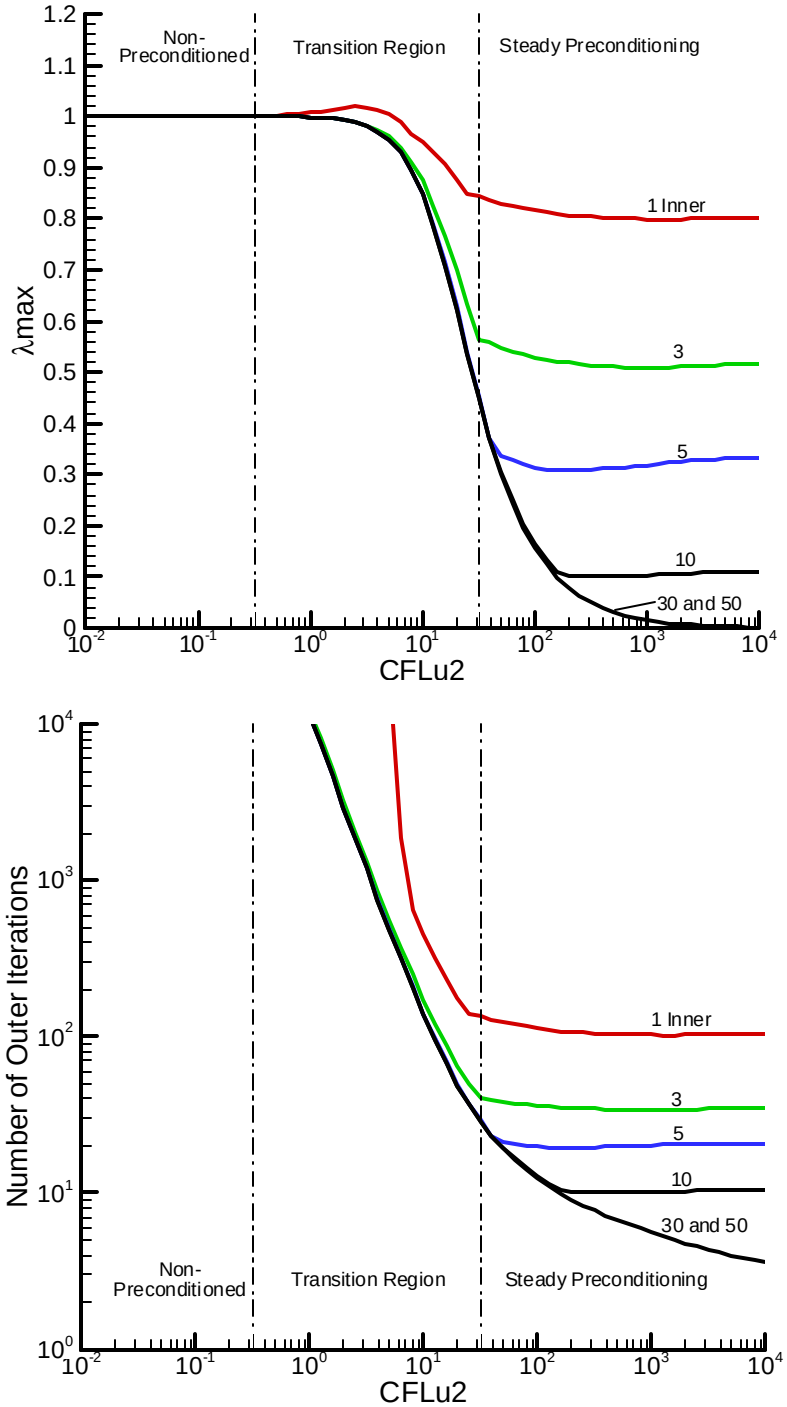

Figure 3.18 Combined inner-outer stability analysis for various number of inner iterations; UPU, I/II/II, Four sweep DDLGS, CFL_{u3}=20, Mach=0.01 and flow angle of zero. The upper plot is $l_{\max}$ versus CFL_{u2} and the lower plot is number of outer iterations to converge ten orders magnitude versus CFL_{u2}.

the number of outer iterations needed to converge ΔQ_p ten orders of magnitude versus $\log 10(CFLu2)$ (the lower plot of figure 3.18). For the coupled inner-outer outer iteration case, the maximum eigenvalue is the largest value of the amplification matrix $\tilde{G}^M$(G) over the whole Fourier domain and the number of iterations is the number of outer iterations needed to reduce ΔQ_p by ten orders of magnitude.

Looking first at the low CFL_{u2} region of figure 3.18, the number of outer iterations is too large to be seen no matter how many inner iterations are used because CFL_{u2} is simply too small. Inspection of the upper plot shows that if we do computations in this region, we do not have to use many inner iterations. In the transition CFL_{u2} region, one inner iteration leads to divergence because of the conditional stability of the DDLGS system and the insufficient inner convergence. After two or three inner iterations, there are very small differences between the different numbers of inner iterations. Again, the transition from non-preconditioned to steady preconditioning at the end of the transition region causes a corner. In the large CFL_{u2} region (steady preconditioning), the larger the number of inner iterations we use, the faster the convergence is. When the iteration number is large enough, the results are very similar to the direct inversion stability results (figure 3.17). Increasing the number of inner iteration does not help the outer convergence anymore. The results for 'UPS', 'UUU' and 'UUS' are similar and are not presented.

To compare the four sets of Γs, we plot the results for three, five and ten iterations for the UPS, UPU, UUU and UUS cases in figures 3.19, 3.20 and 3.21 respectively. Again the results are plotted in terms of $l_{\max}$ versus CFL_{u2} (the upper plot of each figure) and the number of outer iterations to converge ten orders of magnitude versus CFL_{u2} (the lower plot of each figure).

Figure 3.19 shows the outer convergence comparison of the four systems for the three inner iterations case. In this figure, as for the direct inversion results shown in figure 3.17, the 'UPS' system is exactly the same as 'UPU' and 'UUS' is exactly the same as 'UUU'. Actually, there is a tiny difference, but we can not see from the plot. Compared

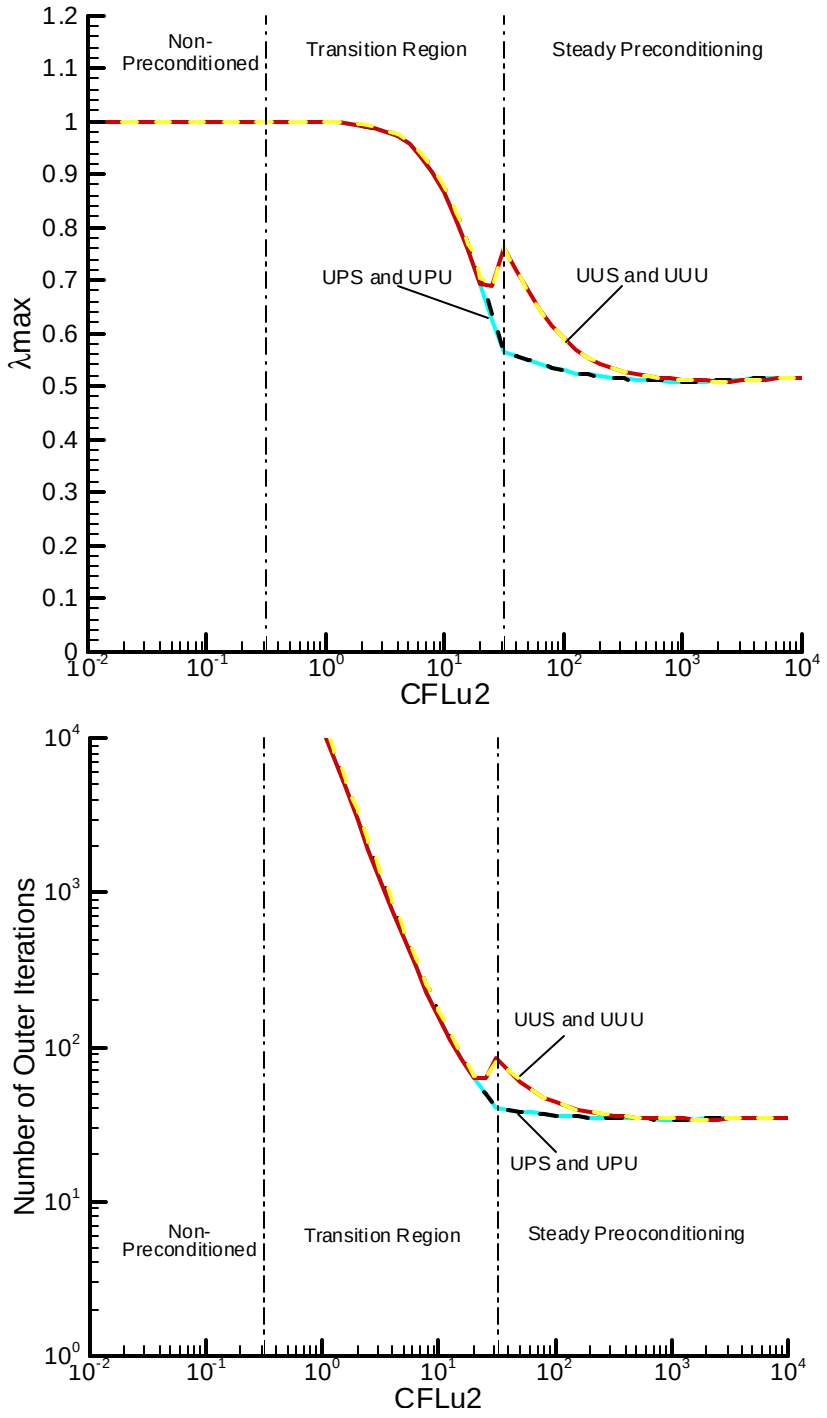

Figure 3.19 The comparison of outer stability for three inner iterations between the four sets of Γ's, I/II/II, four-sweep DDLGS, CFL_{u3}=20, Mach of 0.01 and flow angle of zero. The upper plot is $l_{\max}$ versus CFL_{u2} and the lower plot is the number of outer iterations to converge ten orders versus CFL_{u2}.

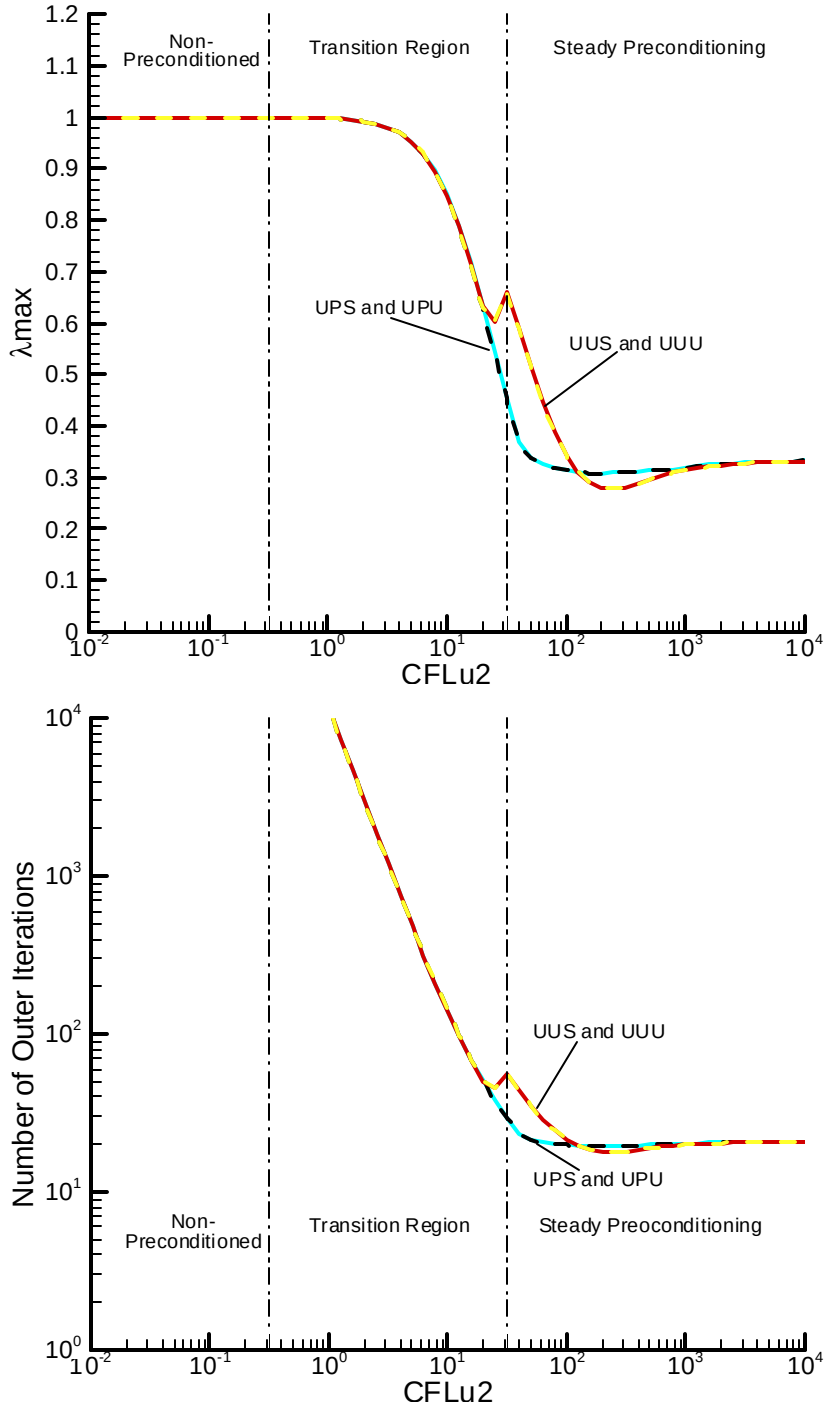

Figure 3.20 The comparison of outer stability for five inner iterations between the four sets of Γ's, I/II/II, four-sweep DDLGS, CFL_{u3}=20, Mach of 0.01 and flow angle of zero. The upper plot is $l_{\max}$ versus CFL_{u2} and the lower plot is the number of outer iterations to converge ten orders versus CFL_{u2}.

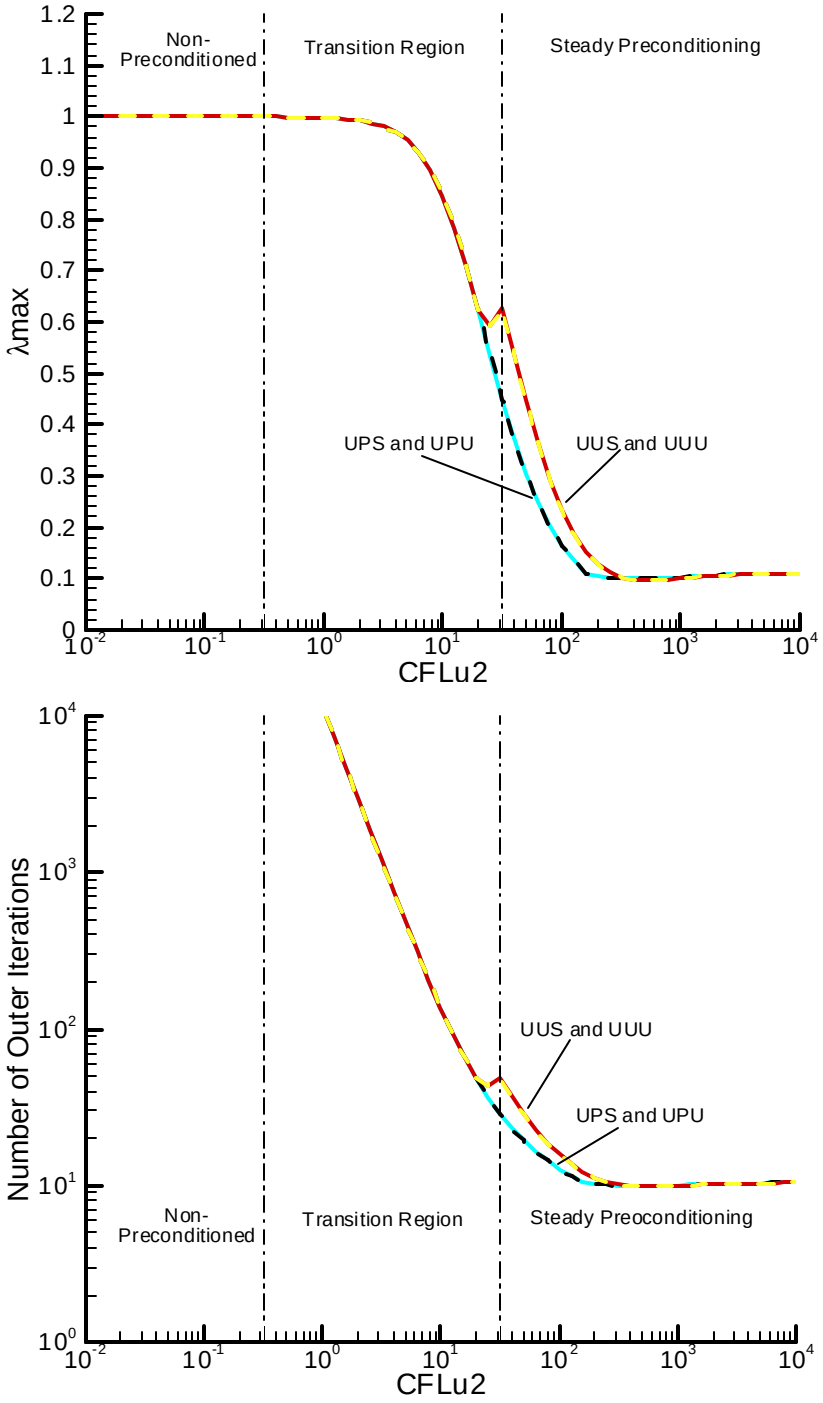

Figure 3.21 The comparison of outer stability for ten inner iterations between the four sets of Γ's, I/II/II, four-sweep DDLGS, CFL_{u3}=20, Mach of 0.01 and flow angle of zero. The upper plot is $l_{\max}$ versus CFL_{u2} and the lower plot is the number of outer iterations to converge ten orders versus CFL_{u2}.

with 'UUU' and 'UUS', 'UPU' and 'UPS' are exactly the same at small and large CFL_{u2} regions but marginally better at the middle CFL_{u2} region. So from this figure, we can conclude that there is no difference between 'UPU' and 'UPS'. But we should mention that 'UPU' is much better in inner convergence at small CFL_{u2} region as shown in figure 3.15, which can not be seen from figure 3.19 because of the stiffness of the outer convergence at small CFL_{u2} region. For this reason, we use 'UPU' instead of 'UPS' for the three Γ s in our triple time scheme.

Figures 3.20 and 3.21 which show the comparison of the outer convergence of the four systems for five and ten inner iterations are very similar to figure 3.19 except that the convergence rate at large CFL_{u2} region is faster.

In computing the inner iteration, there are two obvious choices for determining when to stop the inner iteration. The first choice is to specify the number of inner iterations (i.e, one, ten or fifty). An alternative is to continue the inner iteration until it has converged a certain number of orders of magnitude (say two or four orders). We next present the results for specifying the order of convergence. We only present the results for the 'UPU' system because of its similarity with other three sets of Γ's. The results are shown in figure 3.22. The upper plot is plotted as $l_{\max}$ versus CFL_{u2} and the lower plot is plotted as number of outer iterations to converge ten orders versus CFL_{u2}.

The results in figure 3.22 show that number of iterations decreases very rapidly from 10,000 to less than 100 as CFL_{u2} increases from one. At small and moderate CFL_{u2} region, there is no difference between the one, two and three orders of inner convergence. Only at the very large CFL_{u2} region, does a difference between the three values of inner convergence order appear. The outer convergence for two orders of inner convergence is better than that of the one order of inner convergence. When the inner convergence is more than two orders, we can only see a small gain in the outer damping at high CFL_{u2} number from the upper plot while the difference is more clear from the lower plot. For an inner convergence greater than three orders, any further increase in inner convergence does not improve the outer convergence anymore.

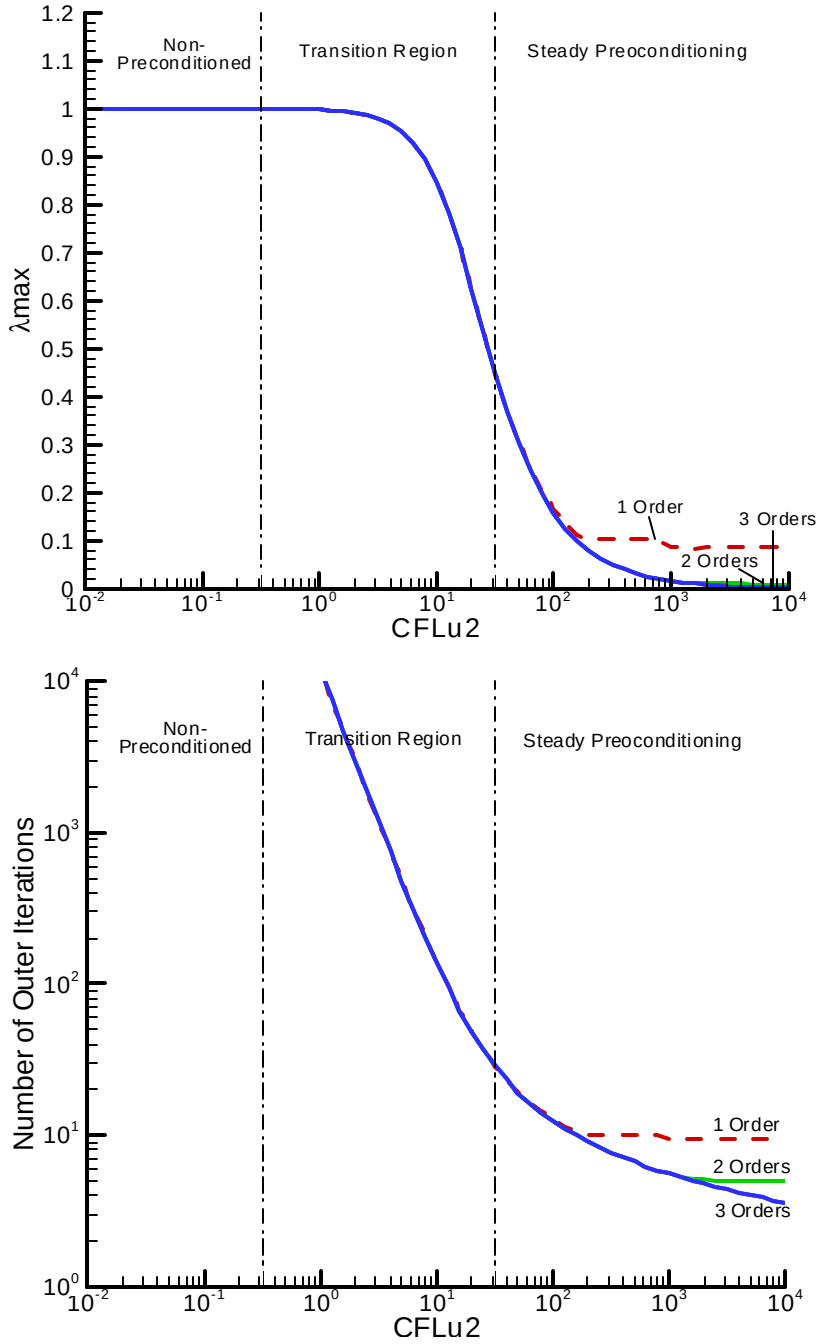

Figure 3.22 Combined inner/outer stability results for various orders of inner convergence, I/II/II, four-sweep DDLGS, CFL_{u3}=20, Mach of 0.01 and flow angle of zero. The upper plot is $l_{\max}$ versus CFL_{u2} and the lower plot is the number of outer iterations to converge 10 orders versus CFL_{u2}.

3.6 Inner Iteration Optimization by Stability Analysis

In section 3.5, we have shown that increasing the number of inner iterations decreases the number of outer iterations. The optimum CPU time, however, will depend upon the total number of inner plus outer iterations. In the following, we try to optimize the number of inner iterations by calculating the CPU time for converging ten orders magnitude for different numbers of inner iterations.

We assume the time for one outer iteration is three times that for one inner iteration (we will shown in the next chapter how this can be achieved). Consequently, we can convert the total number of inner and outer iterations to an equivalent number of iterations and obtain the CPU time for ten orders outer convergence in the overall solution in terms of the CPU time per iteration. By doing so, we re-plot the number of outer iterations in figure 3.18 as the equivalent number of inner iterations in figure 3.23.

Figure 3.23 shows that the equivalent number of inner iteration to obtain ten orders of magnitude convergence in the outer iteration decreases monotonically with increasing CFL_{u2} for any fixed number of inner iterations per outer time step. The optimum number of inner iteration per step changes depending upon the value of CFL_{u2}. As long as CFL_{u2} is less than 30, the optimum number is three, but as CFL_{u2} increases, five becomes more efficient to about CFL_{u2} equals 100. After that ten inner iterations are more efficient than three and five. As CFL_{u2} reaches 10^4, the 30 inner iterations case begins to excel over the ten case. The general conclusion is that the optimum number of inner iteration per step increases slowly with CFL_{u2}.

To see how the optimum number of inner iteration varies with CFL_{u2}, we plot the optimum number of inner iterations from figure 3.23 versus CFL_{u2}, in figure 3.24. The scale of the x-axis in figure 3.24 has been extended to 10^{-2} from the one in figure 3.23 to show the optimum value at small values of CFL_{u2}.

From figure 3.24, we can see that the optimum number of inner iteration increases monotonically with CFL_{u2} from one to thirty. In the CFL_{u2} region which is of most

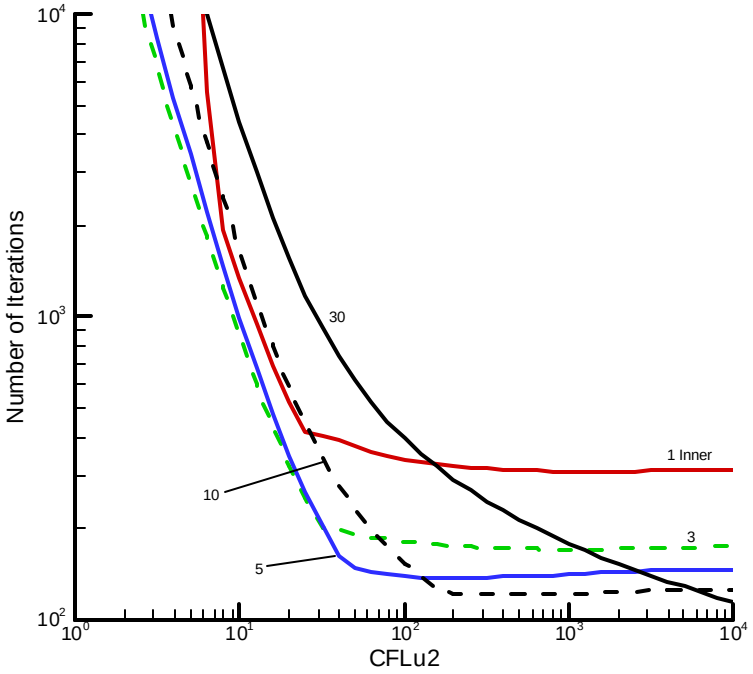

Figure 3.23 The equivalent number of inner iterations for 10 orders of convergence in the outer iteration for various number of inner iterations. UPU, I/II/II, four-sweep line Gauss-Sediel, CFL_{u3}=20, Mach of 0.01 and flow angle of zero.

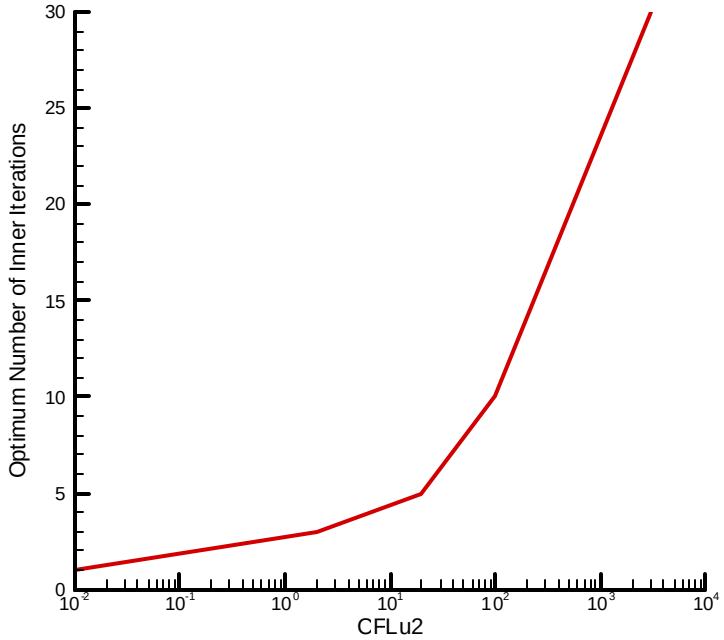

Figure 3.24 The optimum number of inner iterations, UPU, I/II/II, four-sweep line Gauss-Sediel, CFL_{u3}=20, Mach of 0.01 and flow angle of zero.

interest ($1 \le CFL_{u2} \le 100$**),** the optimum inner iteration number is less than 10. In the small CFL_{u2} region, the optimum number of inner iterations is only one.

3.7 The Effect of the Number of Grids on Stability

The number of grid points has an important effect on stability. For a single time scheme, the effect is generally attributed to the low wave number region which is notorious for its convergence stiffness or unstability, as shown in figure 3.4. Generally speaking, the finer the grid the more possible we can resolve the highest eigenvalue point and the stiffer or the more unstable the convergence is. For our triple time scheme, besides the above factor, the grid number also affects the unsteady preconditioning Γ as shown in figure 3.1. Increasing the number of grid points moves the transition region of unsteady preconditioning Γ from the physical to the steady region but does not change the magnitude of that transition region or the magnitude of the eigenvalue.

3.7.1 The Effect of Grid Number on the Inner Stability

First we examine the effect of grid number on the inner stability. There are two aspects for this issue. The first one is the optimum CFL_{u3} and the second one is the maximum eigenvalue. We will use the 'UPU' system, 'I/II/II' accuracy, a Mach number of 0.01 and a flow angle of zero in all of the following results.

The effect of the grid number on the optimum CFL_{u3} can be found by repeating figure 3.12 for a series of grid number N, but that is too much work because we have to compute all cases for a series values of CFL_{u3}, CFL_{u2} and number of grid points. Therefore, to reduce the work, instead of computing the results for a series of CFL_{u2} such as in figure 3.12, we only compute one representative value of CFL_{u2} and assume that the optimum value of CFL_{u3} for this representative CFL_{u2} value is the optimum CFL_{u3} value for all CFL_{u2} values. We choose the representative value of CFL_{u2} to be

infinity because it has the maximum eigenvalue along the optimimum CFL_{u3} line ($CFL_{u3} = 20$) and there is an obvious difference between different values of CFL_{u3} at that point as shown in figure 3.12.

We use this representative value of CFL_{u2}, infinity to compute the maximum eigenvalue of the inner iteration amplification matrix $\tilde{G}$ from equation 3.36 for a series of CFL_{u3} and search for the optimum (mininum) value. Repeat this procedure for a series of numbers of grid points, we can obtain the effect of the number of grid points on the optimum CFL_{u3} and their corresponding maximum eigenvalues. The results are shown in figures 3.25 and 3.26.

Figure 3.25 shows the effect of number of grid points on the optimum CFL_{u3}, we can see that the number of grid points has little effect on the optimum CFL_{u3} except at very small non-practical number of grid points of ten. In the practical number of grid points region, the optimum CFL_{u3} increases very slowly with the number of grid points.

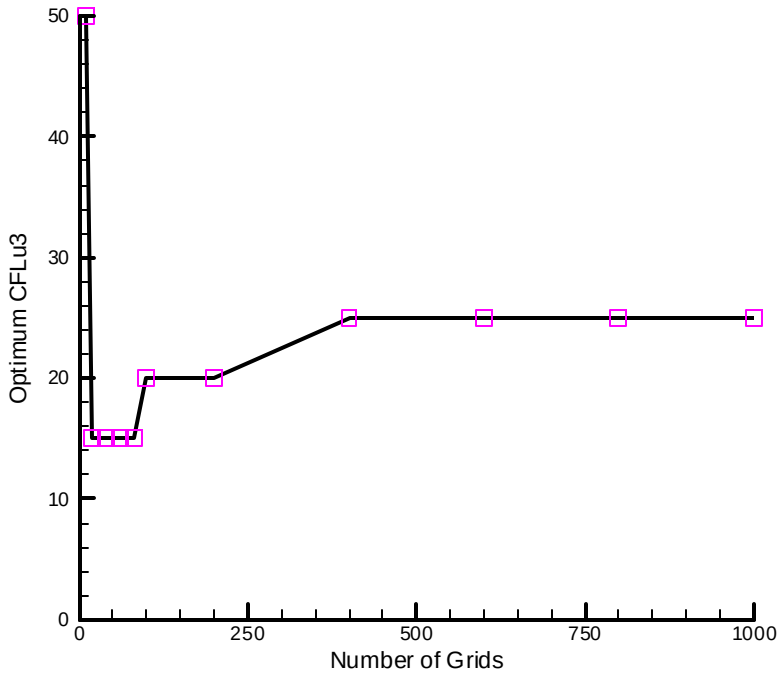

Figure 3.25 The effect of grid number on the optimum CFL_{u3} in the inner stability. UPU, I/II/II, Mach number of 0.01, flow angle of zero and CFL_{u2} of infinity.

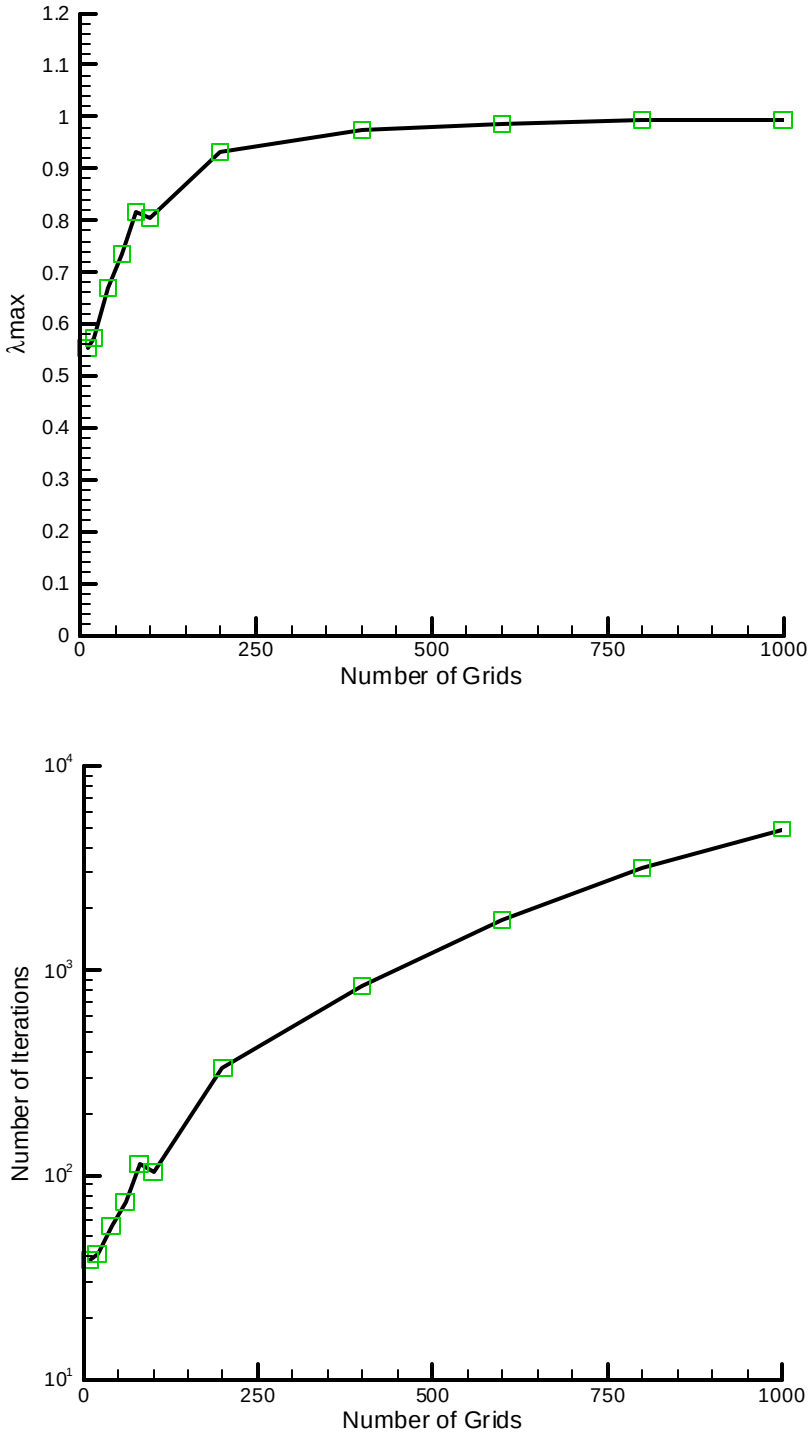

Figure 3.26 The effect of grid number on the convergence of inner stability. The upper plot: the effect of grid number on maximum eigenvalue; The lower plot: the effect of grid number on number of inner iterations for ten orders convergence; UPU, I/II/II, Mach number of 0.01, flow angle of zero and CFL_{u2} of infinity.

Figure 3.26 shows the variation of the convergence rate with the number of grid points. Again the upper plot is the maximum eigenvalue results while the lower plot is the number of inner iterations for ten orders of magnitude. The CFL_{u3} value is the optimum CFL_{u3} in figure 3.25. In this figure the convergence rate increases monotonically with the grid number as we expect and the maximum eigenvalue is less than unity for any number of grid points.

3.7.2 The Effect of Grid Number on the Outer Stability

The grid number also has an influence on outer stability. Using the optimum CFL_{u3} from figure 3.25, we compute the maximum eigenvalue of the outer stability for various numbers of inner iterations and a series of grid sizes. Two values of CFL_{u2} are considered, infinity and ten. The other parameters are the same as subsection 3.7.1. The results are plotted as the maximum eigenvalue $l_{\max}$ versus the grid number in figure 3.27. The upper plot is for an infinite value of CFL_{u2} and the lower plot is for a CFL_{u2} value of 10.

In figure 3.27 where CFL_{u2} is equal to infinity, the maximum eigenvalue increases monotonically with the increase in the number of grids but is less than one for all cases. The larger the number of inner iterations the faster the convergence of the outer iteration is. Figure 3.28 where a CFL_{u2} value of 10 is used follows the same pattern with the first plot, but the iteration diverges for a large grid number for small number of inner iterations. The reason for this divergence is the same as what causes the divergence of one inner iteration case in figure 3.18, which is the conditional stability of the DDLGS system and the insufficient inner convergence.

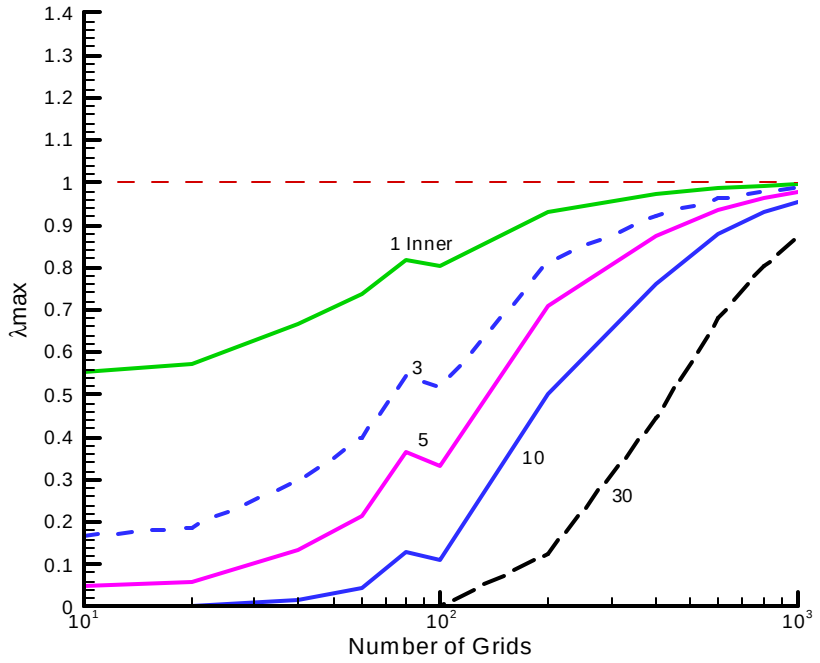

Figure 3.27 The effect of grid number on the outer stability for different number of inner iterations. UPU, I/II/II, Mach number of 0.01, flow angle of zero, CFL_{u3} from figure 3.25, CFL_{u2} of infinity.

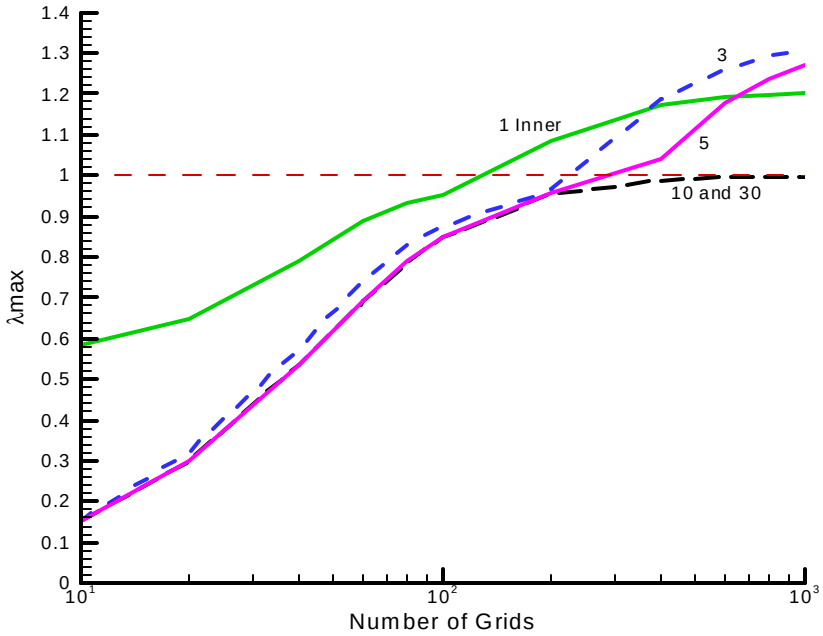

Figure 3.28 The effect of grid number on the outer stability for different number of inner iterations. UPU, I/II/II, Mach number of 0.01, flow angle of zero, CFL_{u3} from figure 3.25, CFL_{u2} of 10.

3.8 Comparison between Computation and Stability

To show the validity of our stability results, we present some computational results. The code we used is an in-house code called 'GEMS', whose brief introduction is given in chapter 5. We use a uniform flow, inviscid flow in a constant area duct as show in figure 3.29. We use exactly the same conditions as in the stability analysis. We present results for the 'UPU' system, the 'I/II/II' accuracy system, a Mach number of 0.01 and a flow angle of zero. The grids in the computational domain is equally spaced with a size of 101x101 and an aspect ratio of one.

The total pressure (P^0), total temperature (T^0) and flow direction (a) are given as the inlet boundary condition, and the back pressure (P_{back}) is given as the outlet boundary condition. Inviscid wall boundary condition is used at the wall boundaries.

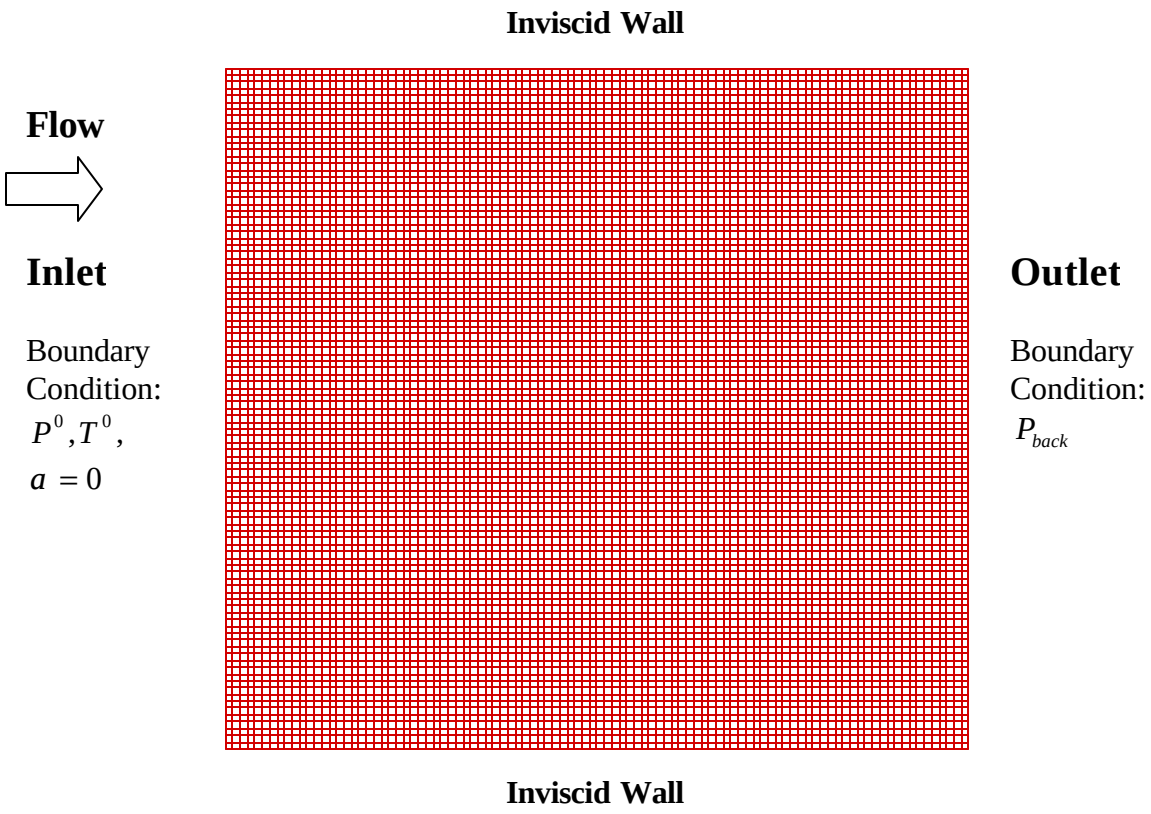

Figure 3.29 Computational grids and boundary conditions for the uniform flow.
Grids: 101x101

The initial condition is set up by introducing a perturbation at every grid point while keeping the total pressure and total temperature constant. The perturbation is introduced in the following way.

First, from the exact solution, we can compute the total pressure, P^0 and total temperature, T^0 from the Mach number and static pressure (back pressure). Then, for a perturbed static pressure less than the exact static pressure, we perturb the static pressure as the change from the initial pressure to the pressure of the exact solution, not the converse. For example, if the initial pressure is 0.01atm and the pressure of the exact solution is 1atm, this is a hundred times perturbation instead of 0.99 times perturbation. By doing so, we obtain

$$P_p = P_e /(1+eR_f) \tag{3.55}$$

Where P_p and P_e are the perturbed static pressure and the exact static pressure respectively, R_f is a random function between 0 and 1, e is a parameter to control the magnitude of the perturbation. As e goes to infinity, P_p goes to zero. Then the perturbed Mach number, M_p can be calculated from the total pressure and the perturbed pressure as

$$M_p = \sqrt{\frac{2}{g-1}\left[\left(\frac{P^0}{P_p}\right)^{\frac{g-1}{g}} - 1\right]} \tag{3.56}$$

and the perturbed static temperature becomes

$$T_p = T_0\left(1+\frac{g-1}{2}M_p{}^2\right)^{-1} \tag{3.57}$$

This ensures that the pressure and velocity in the field remain positive.

Equations 3.55, 56 and 57 give the perturbed flow field as the initial condition in our computation. In the present study, a very small e value of 10^{-5} is used to make the initial condition close to the exact solution.

Figure 3.30 shows the comparison of the inner convergence between the stability and

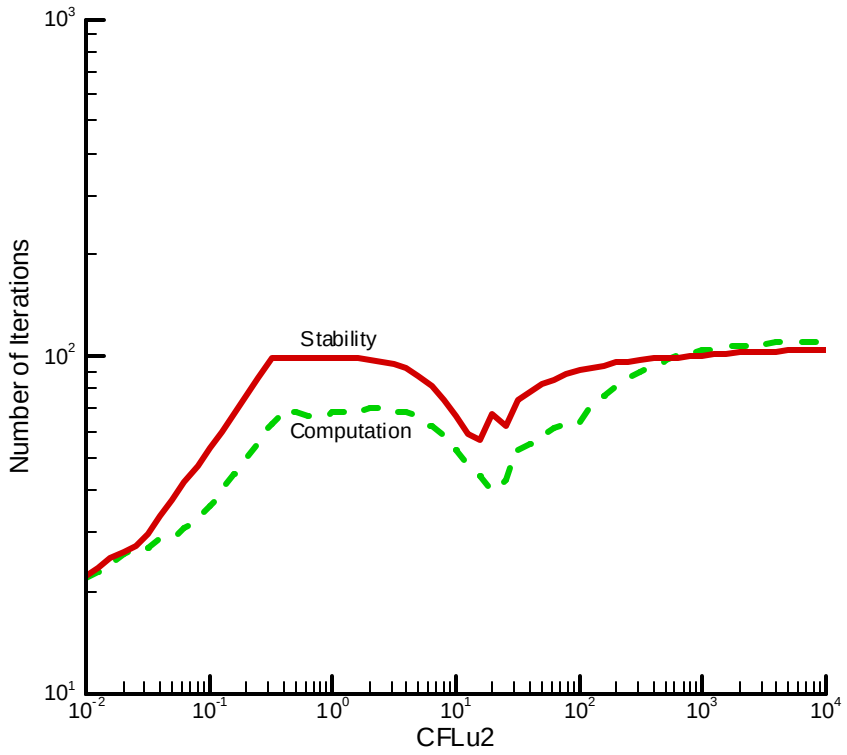

Figure 3.30 Inner iteration comparison between stability analysis and computations of an uniform flow in a straight duct. UPU, I/II/II, Mach number of 0.01, flow angle of zero, CFL_{u3} of 20 and grid number of 101 by 101.

the computation. The data used for the stability line of this figure is exactly the same as the data used for the $CFL_{u3} = 20$ line of figure 3.12. In this figure, the computational results follow a similar pattern as the stability results indicating that stability gives a reasonable prediction of convergence rate. In most of the CFL_{u2} region (except the very large value of CFL_{u2}'s), we need less iterations in the computation than in stability because of the boundary error convection.

Figure 3.31 shows the comparison between the outer stability results and computations of the same uniform flow in a straight duct. The dashed lines are the computational results and the solid lines are the stability results. Again, the stability results and the computational results are similar. This figure shows that the computation results converge slower than stability, which is kind of surprising. Also, the outer

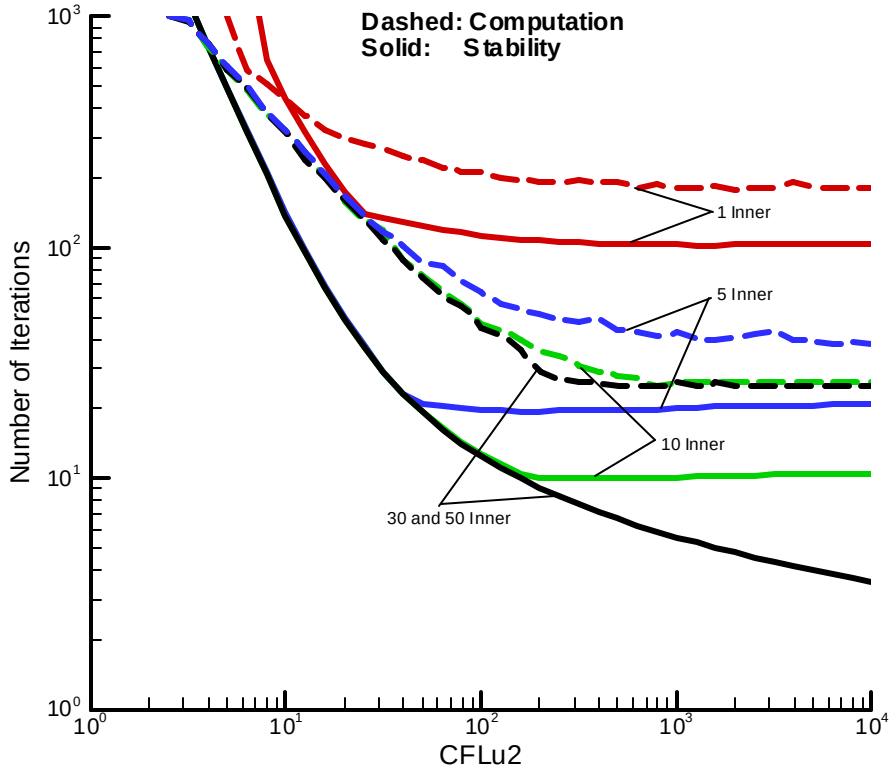

Figure 3.31 The comparison of the outer stability between stability analysis and computation for an uniform flow in a straight duct. Y axis: number of outer iterations for 10 orders outer convergence; UPU, I/II/II, Mach number of 0.01, flow angle of zero, CFL_{u3} of 20 and grid number of 101 by 101.

convergence stops improving after 10 inner iterations instead of 30 inner iterations, as indicated by the stability. Although the above differences exist, the general characteristics of both cases are similar.

Figure 3.32 shows the comparison between the stability analysis and the computational results for the effect of the number of grids on inner stability for the same uniform flow in a straight duct. The computational parameters are exactly the same as the stability parameters. The grids in both the x and y directions are equally spaced, the numbers of grids in both directions are equal and the grid aspect ratio is one. The x-axis of this figure, the grid number, is just the number of grids in the x-dimension. The upper plot is for the optimum CFL_{u3} and the lower plot is for the number of inner iterations

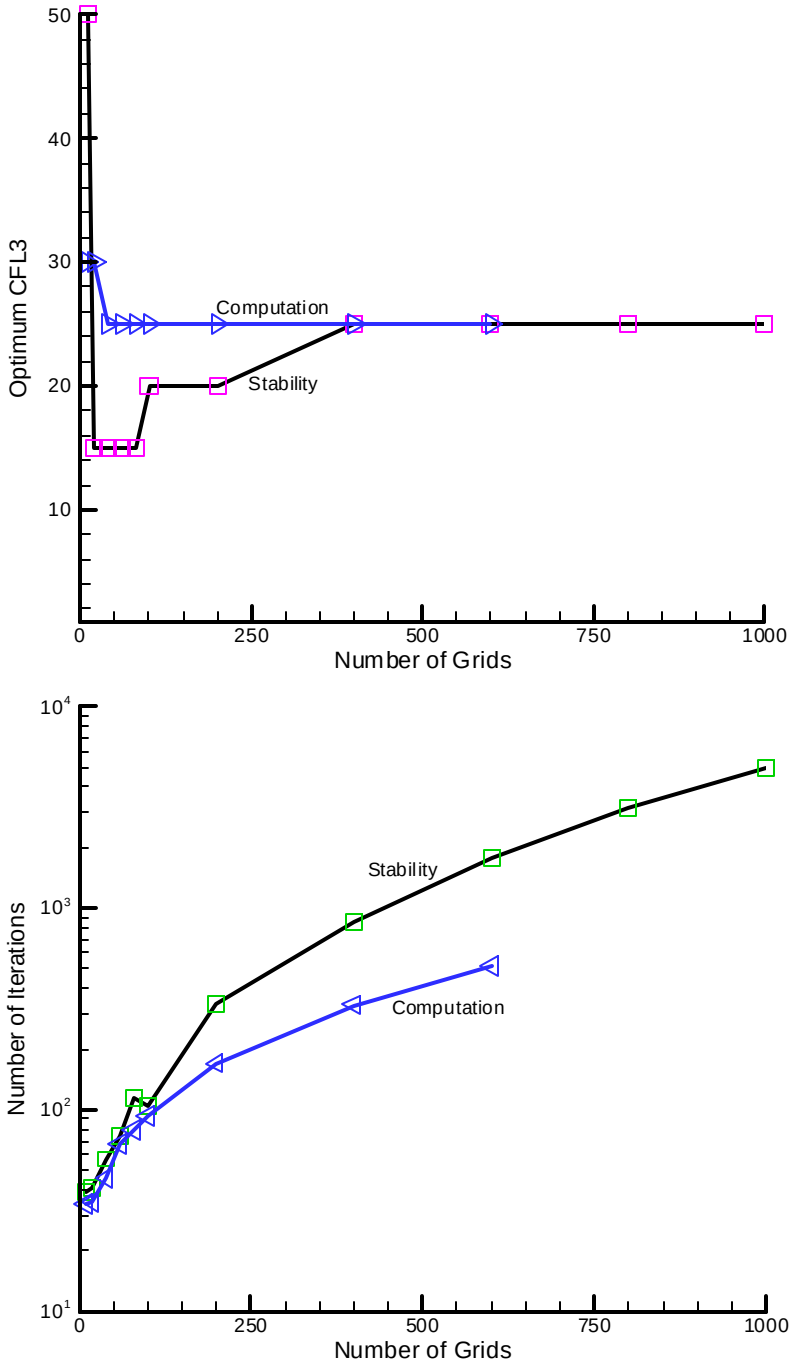

Figure 3.32 Inner stability comparison between the stability analysis and computation of an uniform flow in a straight duct. UPU, I/II/II, Mach number of 0.01, flow angle of zero and grid size of 101 by 101. Upper plot: the optimum CFL_{u3}; Lower plot, the number of iterations for 10 orders inner convergence, CFL_{u3} is the optimum CFL_{u3} from the upper plot.

for ten orders magnitude of convergence for the corresponding optimum CFL_{u3} in the upper plot. The computation results are only provided for grid number up to 600x600 because of the machine limitation (storage and CPU time issues).

From this figure, we can see that the computational results are again close to the stability results. In the first plot, the optimum value of CFL_{u3} for the computation is twenty five and independent of the grid number except for very small grid numbers, which is a little bit different from the stability results in the small grid number region and is exactly the same in the high grid number region. In the second plot, the computational results follow the same trend as the stability results as the number of grids increases. Again, the number of inner iterations for the computation is less than the number of inner iterations for the stability because of the convection of error out of the boundary.

In summary, by comparing the inner convergence, the outer convergence, the effect of grid number on the optimum CFL_{u3} and the number of inner iterations between the computation results and the stability results, we can see that our stability results gives a good prediction for the computations.

3.9 Single Time Stability Analysis and Comparison with Triple Time Scheme

To compare the triple time scheme with the single time scheme, we also perform a stability analysis for the single time. From section 2.9, we know that the single time method is just a special case of the triple time method in which we use only one inner iteration, use the previous outer solution as the initial condition of inner iteration, set Δt_3 to be infinite and $\Gamma_1 = \Gamma_2 = \Gamma_s$ in the triple time equation. Consequently, the stability of the triple time scheme can be obtained from the triple time stability formulation for the first inner iteration (equation 3.54a) by setting Δt_3 to be infinite and $\Gamma_1 = \Gamma_2 = \Gamma_s$.

As before, we compute the maximum eigenvalue of the amplification factor in

equation 3.54a and search over the whole wave number region for the largest value of the maximum eigenvalue which is defined to be $l_{\max}$. We then compute the corresponding number of iterations required for ten orders of convergence versus CFL_{u2} and plot both this number and the value of $l_{\max}$ to obtain figure 3.33.

In figure 3.33, the convergence is very stiff at small values of CFL_{u2}. As CFL_{u2} increases to a value less than twenty, the convergence rate increases. After that, the increase of CFL_{u2} leads to an rapid decrease in convergence rate and the scheme becomes unstable very rapidly. We can see that the optimum CFL_{u2} is 20 and corresponding number of iterations for 10 orders convergence is 100.

We compare the optimum convergence of single time method in figure 3.33 with that of the triple time method in figure 3.23 to give us an idea how much we gain from the triple time. From figure 3.33, the optimum number of outer iterations is 100, whose CPU time is equal to that of 300 inner iterations if one outer iteration takes three times more

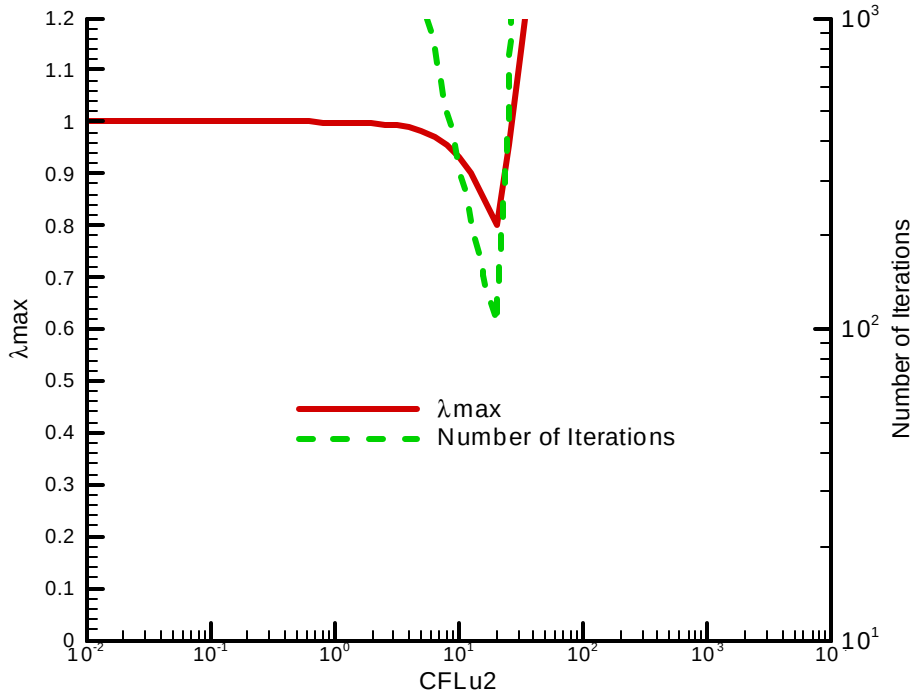

Figure 3.33 The stability analysis for single time. I/II order, Mach number of 0.01, flow angle of zero, grid number of 101x101.

than an inner iteration. From figure 3.23, we can see that the optimum for triple time is $CFL_{u2} = 10000$ and the number of inner iteration is 30, in which case the equivalent number of total inner iterations is 110. Then, from stability theory, the ratio of the optimum triple time convergence is 2.7 times faster than the optimum single time convergence.

3.10 Conclusion

In this chapter, we first discussed the general formulation for the stability analysis. Then we compared the different ways to plot the results. Our results show that the two variables, CFL_{u2} and CFL_{u3}, are the best set of independent parameters to plot the results because they indicate that a constant value of CFL_{u3} is nearly optimum over the entire range of CFL_{u2}. Then we discussed how to pick the best set for Γ_1, Γ_2 and Γ_3 by plotting the stability results by choosing CFL_{u2} and CFL_{u3} as the two independent variables. We compared the inner stability analysis, outer stability analysis with infinite iteration steps (direct inversion) and combined inner and outer stability analysis with limited inner iteration steps. The stability results show that the four systems, 'UPS', 'UPU', 'UUU' and 'UUS' give similar. The system that gives the best overall convergence is 'UPU'.

Second, we optimized the inner iteration. The stability results show that the optimum inner iteration step increases monotonically with CFL_{u2}. In the CFL_{u2} regions that we are of most interest, the optimum inner iteration number ranges between 3 and 30.

The effect of grid number on the convergence is also examined. The grid number has little effect on the optimum CFL_{u3}, but it has a great effect on convergence. As the grid number increases, the convergence rate decreases very rapidly.

Some computational results are given to show the validity of the stability results. The computational results show that our stability analysis gives a good prediction of the

computational performance.

At last, the comparison of the stability results of the triple time and single time indicates that, the optimum triple time convergence is about 2.7 times faster than the optimum single time convergence.

Note that all the computational results in this chapter are for a linear problem that is very close to the solution. Additional potential advantages of the triple time method over the single time method will be presented in chapter five later.

Chapter 4
CPU Time and Storage

4.1 Introduction

In the triple time method, we use an inner iteration to solve the non-linear equation. During each inner iteration, we need to solve the tri-diagonal linear equation at every sweep of the DDLGS approximate factorization method. All elements of the tri-diagonal matrix in the tri-diagonal linear equation are evaluated at the last outer iteration and do not change during the inner iteration. Only the residual of the triple time formulation need to be updated at each inner iteration. Consequently, it is possible to save CPU time by storing the inverse of some matrices in the solver of the tri-diagonal linear equation.

In this chapter, by going through the tri-diagonal linear equation solver procedure, we will analyze and approximately estimate what we should store, how much memory we need to store and how much CPU time we save by storing. We will also present some computational results to compare with our estimation results.

4.2 CPU Time Estimation

The CPU time per iteration for single time method includes the CPU time for Jacobian matrix setup, tri-diagonal linear equation solver and other trivial time costs. Compared with the single time method that has only one iteration, the triple time method contains two iterations, the inner iteration and the outer iteration. Each outer iteration is composed of many inner iterations. The first inner iteration in which we have to setup the matrices and solve the tri-diagonal linear equations is similar to the single time iteration

except that we store some necessary matrices in the tri-diagonal linear equation solver in the first inner iteration of triple time but not in the single time iteration. Consequently, the first inner iteration costs about the same CPU time as one iteration in the single time method. In the other inner iterations, we only need to update the triple time residual and solve the tri-diagonal linear equation by back substitution (as will be discussed later) using the matrices stored in the first inner iteration.

In this thesis, we define the CPU time for solving the tri-diagonal linear equation in all sweeps of DDLGS in each iteration of the single time method or the first inner iteration of the triple time method as the non-linear iteration, and the CPU time for all sweeps of the DDLGS method in the inner iteration (except the first inner iteration) as linear iteration. Thus, the linear iteration includes updating the triple time residual and solving the tri-diagonal linear equation by back substitution.

Although the matrix setup before the iteration requires about half of the total iteration time, its operation count is complex to compute analytically. We do not consider that part in our thesis and simply treat the non-linear iteration as the CPU time for each iteration of the single time or the first inner iteration of the triple time.

In this section, by going through the tri-diagonal linear equation solver, we will analyze the relative costs of the linear iteration and the non-linear iteration in term of the scalar operation count.

4.2.1 Operation Count for Non-Linear Iteration

We start by estimating the number of operations needed to solve the tri-diagonal equation that occurs in the non-linear iteration procedure. The equation can be expressed in the following

$$\begin{pmatrix} D_1 & U_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ L_1 & D_2 & U_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & L_2 & D_3 & U_3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & L_{n-2} & D_{n-1} & U_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & L_{n-1} & D_n \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \Delta Q_{p3} \\ \vdots \\ \Delta Q_{pn-1} \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{n-1} \\ R_n \end{pmatrix} \tag{4.1}$$

This tri-diagonal equation is solved in two steps. The first step is the elimination of the lower diagonal, in which we transform the left hand tri-diagonal matrix to an upper diagonal matrix with all identity matrices on the main diagonal. The procedure is shown in the following,

Assume we have eliminated the lower diagonal components at the first 'i' rows and the tri-diagonal linear equation becomes,

$$\begin{pmatrix} I & U_1' & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & U_2' & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & I & U_i' & 0 & \cdots & 0 \\ 0 & \cdots & 0 & L_i & D_{i+1} & U_{i+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & L_{n-1} & D_n \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \vdots \\ \Delta Q_{pi} \\ \Delta Q_{pi+1} \\ \vdots \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1' \\ R_2' \\ \vdots \\ R_i' \\ R_{i+1} \\ \vdots \\ R_n \end{pmatrix} \tag{4.2}$$

Note that $U_1, U_2, \cdots, U_i$ and $R_1, R_2, \cdots, R_i$ become $U_1', U_2', \cdots, U_i'$ and $R_1', R_2', \cdots, R_i'$ because of the transformation. The expressions for $U_1', U_2', \cdots, U_i'$ and $R_1', R_2', \cdots, R_i'$ are presented as equations 4.7 and 4.8 later.

Now we need to eliminate L_i at the 'i+1' row. In equation 4.2, we multiply the 'i' row by the matrix L_i at the 'i+1' row, subtract it from the 'i+1' row on both the left and right sides of this equation and replacing the 'i+1' row by the computed row to obtain,

$$\begin{pmatrix} I & U_1' & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & U_2' & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & I & U_i' & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & D_{i+1} - L_i U_i' & U_{i+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & L_{n-1} & D_n \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \vdots \\ \Delta Q_{pi} \\ \Delta Q_{pi+1} \\ \vdots \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1' \\ R_2' \\ \vdots \\ R_i' \\ R_{i+1} - L_i R_i' \\ \vdots \\ R_n \end{pmatrix} \tag{4.3}$$

In this step of eliminating the lower diagonal element, the operations for the left hand side of the equation are a matrix-matrix multiplication and a matrix-matrix addition, and the operations for the right hand side of the equation are a matrix-vector multiplication and a vector-vector addition. So the total operations are the addition of the operations at the left and right hand side of equation 4.3

$$1(M \times M) + 1(M + M) + 1(M \times V) + 1(V + V) \tag{4.4}$$

where the symbols, $M \times M, M + M, M \times V, V + V$ indicate the operations for matrix-matrix multiplication, matrix-matrix addition, matrix-vector multiplication and vector-vector addition respectively.

In equation 4.3, multiplying the 'i+1' row by the inverse of the diagonal element at the 'i+1' row, $D_{i+1} - L_i U_i'$, to make the diagonal element at this row to be identity matrix, we obtain

$$\begin{pmatrix} I & U_1' & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & U_2' & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & I & U_i' & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & I & (D_{i+1} - L_i U_i')^{-1} U_{i+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & L_{n-1} & D_n \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \vdots \\ \Delta Q_{pi} \\ \Delta Q_{pi+1} \\ \vdots \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1' \\ R_2' \\ \vdots \\ R_i' \\ (D_{i+1} - L_i U_i')^{-1} (R_{i+1} - L_i R_i') \\ \vdots \\ R_n \end{pmatrix}$$

(4.5)

The operations for this step are

$$1(M^{-1}) + 1(M \times M) + 1(M \times V) \tag{4.6}$$

where M^{-1} indicates the operation for a matrix inversion.

We define the new upper diagonal term in equations 4.2, 4,3 and 4.5 after decomposition to be

$$U_1' = D_1^{-1} U_1 \tag{4.7a}$$

$$U_{i+1}' = \left(D_{i+1} - L_i U_i'\right)^{-1} U_{i+1}, i = 1,2,\cdots,n-2 \tag{4.7b}$$

and the new residual in equations 4.2, 4,3 and 4.5 after decomposition to be

$$R_1' = D_1^{-1} R_1 \tag{4.8a}$$

$$R_{i+1}' = \left(D_{i+1} - L_i U_i'\right)^{-1} \left(R_{i+1} - L_i R_i'\right), i = 1,2,\cdots,n-1 \tag{4.8b}$$

Substitute equation 4.7b and 4.8b into equation 4.5 to obtain,

$$\begin{pmatrix} I & U_1' & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & U_2' & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & I & U_i' & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & I & U_{i+1}' & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & L_{n-1} & D_n \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \vdots \\ \Delta Q_{pi} \\ \Delta Q_{pi+1} \\ \vdots \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1' \\ R_2' \\ \vdots \\ R_i' \\ R_{i+1}' \\ \vdots \\ R_n \end{pmatrix} \tag{4.9}$$

which is the exact same form as equation 4.2 with an additional row transformed from tri-diagonal to upper diagonal.

Repeating these last two decomposition steps until i=n, then completes the lower diagonal decomposition. The tri-diagonal equation then becomes

$$\begin{pmatrix} I & U_1' & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & U_2' & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & I & U_i' & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & I & U_{n-1}' \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} \Delta Q_{p1} \\ \Delta Q_{p2} \\ \vdots \\ \Delta Q_{pi} \\ \vdots \\ \Delta Q_{pn-1} \\ \Delta Q_{pn} \end{pmatrix} = \begin{pmatrix} R_1' \\ R_2' \\ \vdots \\ R_i' \\ \vdots \\ R_{n-1}' \\ R_n' \end{pmatrix} \tag{4.10}$$

We can easily solve this equation by backward substitution. From the last row, we immediately obtain $\Delta Q_{pn} = R_n'$. In the row before the last row, we obtain $\Delta Q_{pn-1} = R_{n-1}' - U_{n-1}' \Delta Q_{pn}$. By the same token, we can solve the rest rows to complete the backward substitution. Consequently, we obtain the solution as

$$\begin{aligned} \Delta Q_{pn} &= R_n' \\ \Delta Q_{pn-1} &= R_{n-1}' - U_{n-1}' \Delta Q_{pn} \\ &\vdots \\ \Delta Q_{pi} &= R_i' - U_i' \Delta Q_{pi+1} \\ &\vdots \\ \Delta Q_{p1} &= R_1' - U_1' \Delta Q_{p2} \end{aligned} \tag{4.11}$$

From equation 4.11, we can see that the operations for the backward substitution are

$$1(M \times V) + 1(V + V) \tag{4.12}$$

The operations for solving a tri-diagonal equation are then determined by the dimension of the tri-diagonal matrix, n. To distinguish from the notation of the second pseudo time level, 'n', we will use N_c as the dimension of the tri-diagonal matrix. For the present problem, N_c is the number of grid cells in the computational domain. Consequently, the operations for solving one tri-diagonal matrix is obtained by multiplying N_c on the

summation of equations 4.4, 4.6 and 4.12 as

$$N_c\left[1(M^{-1})+2(M\times M)+3(M\times V)+1(M+M)+2(V+V)\right] \tag{4.13a}$$

For a problem with the dimension number of $N_{\dim}$, we generally need $2N_{\dim}$ sweeps in the DDLGS approximate solver and accordingly need $2N_{\dim}$ sweeps to compute one non-linear iteration. Thus we need to solve $2N_{\dim}$ such tri-diagonal equations. So the total operations in one non-linear iteration are obtained by multiplying equation 4.13a by $2N_{\dim}$ as

$$2N_{\dim}N_c\left[1(M^{-1})+2(M\times M)+3(M\times V)+1(M+M)+2(V+V)\right] \tag{4.13b}$$

Equation 4.13b is composed of a series of operations, M^{-1}, $M\times M$, $M\times V$, $M+M$ and $V+V$. Next, we estimate the time for these matrix or vector operations in term of the time for scalar operations.

There are many methods to compute the inverse of a matrix. The commonly used are Gaussian Elimination, Gaussian-Jordan Elimination and Crout's method [56]. The cost for solving a matrix inversion by Gaussian Elimination [56] is,

$$\left(M^{-1}\right)=\left(\frac{5}{6}N_{eq}^3+2N_{eq}^2-\frac{5}{6}N_{eq}\right)^{\times}+\left(\frac{4}{3}N_{eq}^3-\frac{1}{2}N_{eq}^2-\frac{5}{6}N_{eq}\right)^{+} \tag{4.14a}$$

where N_{eq} is the dimension of the matrix or the number of equations of our system, and the superscripts '×' and '+' indicate the operations for scalar multiplication and addition respectively. The Gaussian-Jordam Elimination and Crout's method cost about the same amount time as Gaussian Elimination [56].

The cost of a matrix-matrix multiplication is

$$\left(M\times M\right)=N_{eq}^2\left[\left(N_{eq}\right)^{\times}+(N_{eq}-1)^{+}\right] \tag{4.14b}$$

The cost of a matrix-vector multiplication is

$$(M \times V) = N_{eq}\left[(N_{eq})^{\times} + (N_{eq} - 1)^{+}\right] \tag{4.14c}$$

The cost of a matrix-matrix addition is

$$(M + M) = (N_{eq}^2)^{+} \tag{4.14d}$$

and the cost of the vector-vector addition is

$$(V + V) = (N_{eq})^{+} \tag{4.14e}$$

Substituting equation set 4.14 into equation 4.13b to obtain the operation count for non-linear iteration as

$$2N_{\dim}N_c\left[\left(\frac{17}{6}N_{eq}^3 + 5N_{eq}^2 - \frac{5}{6}N_{eq}\right)^{\times} + \left(\frac{10}{3}N_{eq}^3 - \frac{3}{2}N_{eq}^2 - \frac{11}{6}N_{eq}\right)^{+}\right] \tag{4.15}$$

From equation 4.15, we can see that the operation count for non-linear iteration is a polynomial function of the number of equations. As the number of equations increases, the operation count for non-linear iteration increases very rapidly. For a large enough number of equations, the operation count for non-linear iteration is proportional to the cube of the number of equations.

4.2.2 Operation Count for Linear Iteration

The linear iteration requires time for updating the residual and time for solving the back substitution of the tri-diagonal equation from the stored matrices.

First we compute the operations for updating the triple time residual which is given by the triple time residual equation 2.22a,

$$R_3 = (\nabla_D \cdot F)^n + \left[\frac{\Gamma_2}{\Delta t_2} + (\nabla_D \cdot A_p)_2^n\right](\tilde{Q}_p^m - Q_p^n) \tag{2.22a}$$

The corresponding Jacobian divergence operator is defined in equation 2.20 and is repeated here,

$$\nabla_D \cdot A_p = \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL}\bullet_L + A_{pR}\bullet_R\right) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}A_p\right|\left(\bullet_R - \bullet_L\right)\right]_k S_k \tag{2.20}$$

In equation 2.22a, we add a subscript '2' on the Jacobian divergence operator $\nabla_D \cdot A_p$ in equation 2.20 to indicate that it is for the second pseudo time.

For convenience we group the coefficients for $\bullet_L$ and $\bullet_R$ as

$$\nabla_D \cdot A_p = \frac{1}{\Omega}\sum_{k=1}^{K}\left[\frac{1}{2}\left(A_{pL} + \Gamma_1\left|\Gamma_1^{-1}A_p\right|\right)\bullet_L + \frac{1}{2}\left(A_{pR} - \Gamma_1\left|\Gamma_1^{-1}A_p\right|\right)\bullet_R\right]_k S_k \tag{4.16a}$$

By defining

$$J_L = \frac{1}{2}\left(A_{pL} + \Gamma_1\left|\Gamma_1^{-1}A_p\right|\right) \tag{4.17a}$$

$$J_R = \frac{1}{2}\left(A_{pR} - \Gamma_1\left|\Gamma_1^{-1}A_p\right|\right) \tag{4.17b}$$

We can re-write equation 4.16a as

$$\nabla_D \cdot A_p == \frac{1}{\Omega}\sum_{k=1}^{K}\left[J_L\bullet_L + J_R\bullet_R\right]_k S_k \tag{4.16b}$$

The discretized Jacobian divergence in equation 2.22a, $\left(\nabla_D \cdot A_p\right)_2$ can be discretized to be any order and generally first or second order. The operation cost is different between first order and second order, and consequently we compute them separately.

For the first order, the Jacobians, A_{pL}, A_{pR} and $\left|\Gamma_1^{-1}A_p\right|_k$ at the face are equal to the value at the corresponding cells. Therefore, the Jacobian divergence operator $\nabla_D \cdot A_p$ can be written as the summation of the Jacobians of the control cell 'i' and all surrounding cells. Accordingly, equation 4.16b becomes

$$\nabla_D \cdot A_p = J_i + \sum_{k=1}^{K} J_k \tag{4.18}$$

where matrices J_i and J_k can be expressed in terms of the matrices, J_L and J_R at cell 'i' and its surrounding cells, and K is the number of surrounding cells of cell 'i'

which equals the number of faces of cell 'i'. For a structured grid like that in figure 2.1, these Jacobians are obtained from equation set 2.31.

Substituting equation 4.18 into the right hand side of equation 2.22a provides the residual for the triple time, $R_3^{\ 1}$, where the superscript '1' indicates the first order,

$$R_3^{\ 1} = -(\nabla_D \cdot F)^n - \left[\left(\frac{\Gamma_2}{\Delta t_2} + J_i^{\ l} \right) (\tilde{Q}_p^m - Q_p^n)_i + \sum_{k=1}^{K} J_k^{\ l} (\tilde{Q}_p^m - Q_p^n)_k \right] \tag{4.19}$$

From equation 4.19, the operations for updating $R_3^{\ 1}$ are,

$$1(M \times s) + (1+K)(M \times V) + 1(M+M) + 2(K+1)(V+V) \tag{4.20}$$

Where $M \times s$ is the CPU time cost of matrix-scalar multiply.

For higher order discretization, the Jacobians, $A_L, A_R, \left|\Gamma_1^{-1} A_p\right|_k, \Delta\tilde{Q}_{pR}$ and $\Delta\tilde{Q}_{pL}$ in the operator $\nabla_D \cdot A_p$ of equation 4.16 can not be evaluated at cells and must be evaluated at the left and right sides of face k in the outer iteration. So equation 4.16b can not lead to equation 4.18 and have to keep the original form.

We substitute equation 4.16b into the right hand side of equation 2.22a to obtain the residual for second order, $R_3^{\ 2}$, where the superscript '2' indicates second order

$$R_3^{\ 2} = -(\nabla_D \cdot F)^n - \left\{ \frac{\Gamma_2}{\Delta t_2} (\tilde{Q}_p^m - Q_p^n) + \frac{1}{\Omega} \sum_{k=1}^{K} \left[J_L (\tilde{Q}_{pL}^m - Q_{pL}^n) + J_R (\tilde{Q}_{pR}^m - Q_{pR}^n) \right]_k S_k \right\} \tag{4.21}$$

From equation 4.21, the operations for updating $R_3^{\ 2}$ are,

$$(2K+1)(M \times V) + 3(K+1)(V+V) \tag{4.22}$$

After computing the number of operations for the residual update, we next estimate another part of the linear iteration, the time for back substitution in the tri-diagonal equation. The back substitution is given by equation 4.11, where $U_i', i = 1,2,\cdots,n-1$ is

provided by equation 4.7a and b. The matrices $U_i', i = 1,2,\cdots,n-1$ can be stored because they do not change during the inner iterations. The right hand side vectors $R_i', i = 1,2,\cdots,n$ in equation 4.11 are computed by equation 4.8. These vectors can not be stored because the triple time residuals $R_i, i = 1,2,\cdots,n$ are updated at every inner iteration. Consequently we could store the matrices D_1^{-1} and $\left(D_{i+1} - L_i U_i'\right)^{-1}, i = 1,2,\cdots,n-1$ in equations 4.7 and 4.8. The last term we need to store is $L_i, i = 1,2,\cdots,n-1$, which is already stored in our traditional single time method.

. Having stored all these matrices, we only need to use equation 4.8 and equation 4.11 to solve the tri-diagonal equation in the inner iteration. The operations in equation 4.8 are

$$2(M \times V) + 1(V + V) \tag{4.23}$$

The operations in equation 4.11 are

$$1(M \times V) + 1(V + V) \tag{4.24}$$

Adding equations 4.23 and 4.24 together, we get the total operations for back-substitution of a tri-diagonal equation as the following,

$$3(M \times V) + 2(V + V) \tag{4.25}$$

For a DDLGS approximate factorization with $2N_{\dim}$ sweeps, we need to solve $2N_{\dim}$ tri-diagonal equations. Multiply equation 4.25 by $2N_{\dim}$, obtain,

$$2N_{\dim}\left[3(M \times V) + 2(V + V)\right] \tag{4.26}$$

The total operations for each inner iteration include the operations of updating the residual and the operations for back substitution of the DDLGS approximate solver. For the I/I/* system, it is the sum of equation 4.21 and equation 4.26, which is,

$$\begin{aligned} &1(M \times s) + (1 + K)(M \times V) + 1(M + M) + 2(K + 1)(V + V) + 2N_{\dim}\left[3(M \times V) + 2(V + V)\right] \\ &= 1(M \times s) + (1 + K + 6N_{\dim})(M \times V) + 1(M + M) + 2(1 + K + 2N_{\dim})(V + V) \end{aligned}$$

(4.27)

For the I/II/* system, it is the summation of equation 4.22 and equation 4.26, which is,

$$\begin{aligned}&(2K+1)(M\times V)+3(K+1)(V+V)+2N_{\dim}\left[3(M\times V)+2(V+V)\right]\\&=(2K+1+6N_{\dim})(M\times V)+(3K+3+4N_{\dim})(V+V)\end{aligned} \tag{4.28}$$

The matrix-matrix and matrix-vector operations in equations 4.27 and 4.28 can be written in terms of scalar operations. For a fluid dynamics system composed of N_{eq} equations the matrix size is $N_{eq}\times N_{eq}$ and the vector size is N_{eq}. Consequently, by writing the matrix-matrix and matrix-vector operations in equation 4.27 in terms of the scalar operations, we get the total operation count for linear iteration for the I/I/* order system is

$$\begin{aligned}&(N_{eq}^2)^{\times}+(1+K+6N_{\dim})\left[(N_{eq}^2)^{\times}+N_{eq}\left(N_{eq}-1\right)^{+}\right]+1(N_{eq}^2)^{+}+(2K+2+4N_{\dim})(N_{eq})^{+}\\&=\left[(2+K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(2+K+6N_{\dim})N_{eq}^2+(K-2N_{\dim}+1)N_{eq}\right]^{+}\end{aligned} \tag{4.29a}$$

Similarly, from equation 4.28, we obtain the operation count for linear iteration for the I/II/* order as

$$\begin{aligned}&(1+2K+6N_{\dim})\left[\left(N_{eq}^2\right)^{\times}+N_{eq}\left(N_{eq}-1\right)^{+}\right]+\left[(3K+3+4N_{\dim})N_{eq}\right]^{+}\\&=\left[(1+2K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(1+2K+6N_{\dim})N_{eq}^2+(2+K-2N_{\dim})N_{eq}\right]^{+}\end{aligned} \tag{4.30a}$$

Multiply equations 4.29a and 4.30a by the number of cells, N_c to obtain the operations of linear iteration for a single linear iteration of the I/I/* system in the whole flow field as

$$N_c\left\{\left[(2+K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(2+K+6N_{\dim})N_{eq}^2+(K-2N_{\dim}+1)N_{eq}\right]^{+}\right\} \tag{4.29b}$$

and the operation count of linear iteration for the I/II/* system in the whole flow field as

$$N_c\left\{\left[(1+2K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(1+2K+6N_{\dim})N_{eq}^2+(K-2N_{\dim}+2)N_{eq}\right]^{+}\right\} \tag{4.30b}$$

For convenience, we put equations 4.15, 4.29b and 4.30b in table 4.1. Note that, to obtain this table, we have ignored the operations involved in matrix setup and location. Note that the variable, N_{eq} in three-dimensions is one more than the variable, N_{eq} at two dimensions because three-dimensional problem has one more momentum equation

Table 4.1 Operation count for non-linear iteration and linear iteration

	Operation count
Non-linear iteration	$2N_{\dim}N_c\left[\left(\frac{17}{6}N_{eq}^3+5N_{eq}^2-\frac{5}{6}N_{eq}\right)^{\times}+\left(\frac{10}{3}N_{eq}^3-\frac{3}{2}N_{eq}^2-\frac{11}{6}N_{eq}\right)^{+}\right]$
Linear iteration (I/I/*)	$N_c\left\{\left[(2+K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(2+K+6N_{\dim})N_{eq}^2+(K-2N_{\dim}+1)N_{eq}\right]^{+}\right\}$
Linear iteration (I/II/*)	$N_c\left\{\left[(1+2K+6N_{\dim})N_{eq}^2\right]^{\times}+\left[(1+2K+6N_{\dim})N_{eq}^2+(K-2N_{\dim}+2)N_{eq}\right]^{+}\right\}$

than the two dimensional problem.

From table 4.1, we can see that the operation count of non-linear iteration is proportional to the cube of the number of equations while the operation count of linear iteration is proportional to the square of the number of the equations. This suggests that we should expect a monotonic increase in the operation count ratio of the non-linear iteration to the linear iteration as the number of equations in the fluid dynamics system increases. Also, we need more operations for the I/II/* system than the I/I/* system. For a two-dimensional quadralateral grid with four faces, K=4 and $N_{\dim}=2$, table 4.1 simplifies to table 4.2.

From table 4.2, we can compute the operation count ratio (operation count ratio is equal to CPU time ratio) of non-linear iteration to linear iteration of the I/II/* system approximately as

$$\frac{4\left(\frac{17}{6}N_{eq}^3+5N_{eq}^2-\frac{5}{6}N_{eq}\right)^{\times}+4\left(\frac{10}{3}N_{eq}^3-\frac{3}{2}N_{eq}^2-\frac{11}{6}N_{eq}\right)^{+}}{\left(21N_{eq}^2\right)^{\times}+\left(21N_{eq}^2+2N_{eq}\right)^{+}}$$

Table 4.2 Operation count for non-linear iteration and linear iteration with K=4 and $N_{\dim} = 2$

	2D	**3D**
Non-linear iteration	$4N_c\left\{\left(\frac{17}{6}N_{eq}^3 + 5N_{eq}^2 - \frac{5}{6}N_{eq}\right)^{\times} + \left(\frac{10}{3}N_{eq}^3 - \frac{3}{2}N_{eq}^2 - \frac{11}{6}N_{eq}\right)^{+}\right\}$	$6N_c\left[\left(\frac{17}{6}N_{eq}^3 + 5N_{eq}^2 - \frac{5}{6}N_{eq}\right)^{\times} + \left(\frac{10}{3}N_{eq}^3 - \frac{3}{2}N_{eq}^2 - \frac{11}{6}N_{eq}\right)^{+}\right]$
linear iteration (I/I/*)	$N_c\left[\left(18N_{eq}^2\right)^{\times} + \left(18N_{eq}^2 + N_{eq}\right)^{+}\right]$	$N_c\left[\left(26N_{eq}^2\right)^{\times} + \left(26N_{eq}^2 + N_{eq}\right)^{+}\right]$
linear iteration (I/II/*)	$N_c\left[\left(21N_{eq}^2\right)^{\times} + \left(21N_{eq}^2 + 2N_{eq}\right)^{+}\right]$	$N_c\left[\left(31N_{eq}^2\right)^{\times} + \left(31N_{eq}^2 + 2N_{eq}\right)^{+}\right]$

Simply canceling out the addition operations because of the coefficients of the multiplication operator and the coefficients of the addition operator are very close, we obtain

$$\frac{4\left(\frac{17}{6}N_{eq}^3 + 5N_{eq}^2 - \frac{5}{6}N_{eq}\right)^{\times}}{\left(21N_{eq}^2\right)^{\times}} = \frac{34}{63}N_{eq} + \frac{20}{21} - \frac{10}{63N_{eq}} \tag{4.31}$$

In equation 4.31, the CPU time ratio of the non-linear iteration to the linear iteration is almost linear to the number of equations. As the number of equations increases as for a complex multiple species system, the saving should increase rapidly. For a four-equation system, $N_{eq} = 4$, from equation 4.31, the CPU time ratio of the non-linear iteration to the I/II/II order linear iteration is approximately 3.07.

4.3 Storage Requirement

The price of CPU time saving we have to pay is more storage cost. We should know how much we pay for our benefit. In this section, we will estimate the amount of storage required for the single time and triple time methods, then we compare them.

4.3.1 Single Time Storage

In a CFD code, matrix storage takes most of the total storage and other storages are negligible. So in our study, we only consider the matrix storage. The matrix storage needed in a single time method is the Jacobian coefficients for the left hand side of the single time equation. For a cell with K faces, we have K+1 Jacobian matrix coefficients to be stored. The storage of these matrices for an equation system with N_{eq} equations is,

$$(1+K)*N_{eq}{}^{2} \tag{4.32}$$

where $N_{eq}{}^{2}$, is the storage of a matrix

If the computational field contains N_c cells, the total storage will be

$$(1+K)*N_c*N_{eq}{}^{2} \tag{4.33}$$

4.3.2 Triple Time Storage

The triple time uses more storage than the single time. The storage increase is ascribed to two parts. The first part is for the linear solver in which the back substitution of tri-diagonal matrix is stored. The second part is the storage required for the higher order Jacobian matrices at the right hand side of the equation where we use consistent send order differencing rather than the inconsistent first order differencing as in the single time method.

In the tri-diagonal equation solver of the first inner iteration of triple time, we need to store the matrices, $U_i^{'}, i=1,2,\cdots,n-1$, D_1^{-1} and $\left(D_{i+1}-L_iU_i^{'}\right)^{-1}, i=1,2,\cdots,n-1$ in equation 4.5 during the lower diagonal decomposition procedure. Since we have $2N_{\dim}$ sweeps and for every sweep we have to store those matrices. So the total additional storage for the linear solver is

$$2N_{\dim}(2N_c-1){N_{eq}}^2 \approx 4N_{\dim}N_c{N_{eq}}^2 \tag{4.34}$$

So the total storage of I/I/* triple time scheme is the sum of equations 4.34 and 4.33, which is,

$$\left[(1+K)+4N_{\dim}\right]*N_c*{N_{eq}}^2 \tag{4.35}$$

For the higher order system, in addition to the extra storage in the linear solver, we also need to store the Jacobi matrices in the residual of triple time, ${R_3}^2$. From equation 4.20, the additional storage of these matrices is

$$2KN_c{N_{eq}}^2 \tag{4.36}$$

The total storage cost of I/II/* triple time scheme is the sum of equation 4.35 and 4.36.

$$\left[(1+3K)+4N_{\dim}\right]*N_c*{N_{eq}}^2 \tag{4.37}$$

For convenience, we summarize the storage requirement of the single time method (equation 4.33), the I/I/* triple time method (equation 4.35) and the I/II/* triple time method (equation 4.37) in table 4.3

To compute the storage ratio of the I/I/* triple time method to the single time method, R_{s1}, we take the ratio of equation 4.35 to equation 4.33 to obtain,

$$R_{s1}=\frac{\left[(1+K)+4N_{\dim}\right]*{N_{eq}}^2*N_c}{(1+K)*{N_{eq}}^2*N_c}=1+\frac{4N_{\dim}}{1+K} \tag{4.38}$$

To compute the storage ratio of I/II/* triple time to single time, R_{s2}, we take the ratio of

Table 4.3 Storage formulation for single time method and triple time method (K: the number of faces. N_{eq}: the number of equations. N_c: the number of cells.)

Method	Storage(words)
Single Time	$(1+K)*N_{eq}{}^2*N_c$
Triple Time (I/I/*)	$[(1+K)+4N_{\dim}]*N_{eq}{}^2*N_c$
Triple Time (I/II/*)	$[(1+3K)+4N_{\dim}]*N_{eq}{}^2*N_c$

equation 4.37 to equation 4.33 to obtain,

$$R_{s2}=\frac{[(1+3K)+4N_{\dim}]*N_{eq}{}^2*N_c}{(1+K)*N_{eq}{}^2*N_c}=3+\frac{4N_{\dim}-2}{1+K} \tag{4.39}$$

Note that the storage ratios are independent of the number of cells, N_c and the number of equations, N_{eq}. For a two dimensional, quadrilateral grid, $N_{\dim}=2$ and $K=4$, the storage ratio of the I/I/* triple time method to the single time method is 2.6 from equation 4.38 and the storage ratio of the I/II/* triple time method to the single time method is 4.2 from equation 4.39.

4.4 Operation Count and Storage Comparisons between Estimation and Computation

To check the results from our analysis, we consider exactly the same uniform straight duct flow introduced in section 3.8 except here we use a grid size of 51x51 instead of 101x101. Both the triple time method and the single time method are considered.

For the triple time method, the UPU, I/II/II system was computed using the optimum

CFL_{u3} of twenty which was found from stability analysis. Because the initial condition is so close to the exact solution we use a CFL_{u2} of infinity. The inner iterations were converged one half order (we will prove later that this is the optimum order of inner convergence for these conditions). For the single time method, the inconsistent I/II system and the optimum CFL_{u2} of twenty from stability analysis was used.

The comparison between the convergence of the single time method and that of the triple time method are shown in figure 4.1. In figure 4.1, the total number of iterations for converging to 10^{-10} for the triple time method is 54 including 18 outer iterations and 36

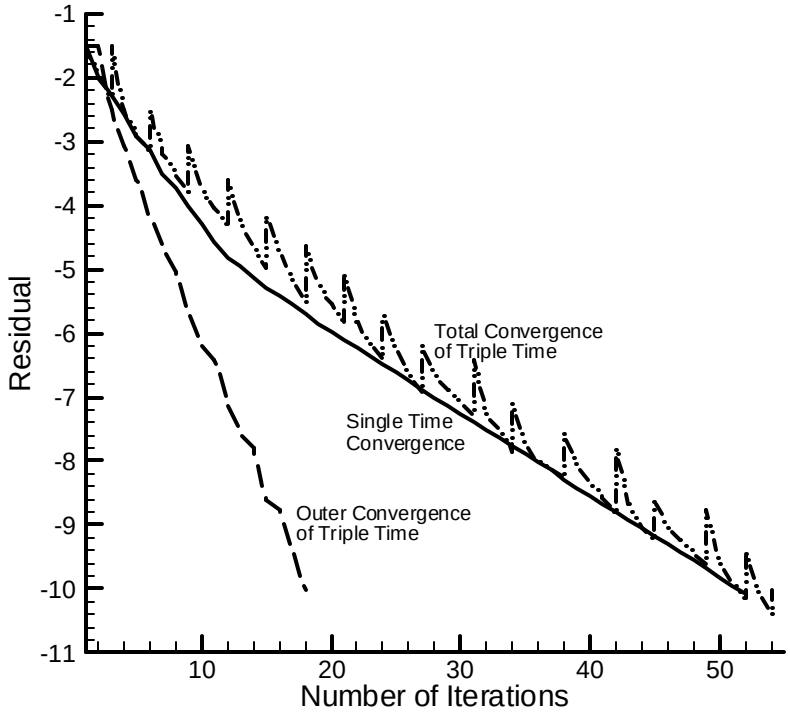

Figure 4.1 The comparison of convergence between the triple time method and the single time method for linear problem (uniform straight duct flow, Mach number of 0.01, grid number of 51x51 and grid aspect ratio of one). Triple time: UPU, I/II/II, CFL_{u3} of 20, CFL_{u2} of infinity and 0.5 order of inner convergence. Single Time: I/II, CFL_{u2} of 20.

inner iterations, while the number of iterations for the single time method is also 54. Although the total number of iterations for the triple time method is the same as the total number of iterations for the single time method, the triple time method is faster since the inner iteration in the triple time method is faster than the iteration in the single time method. Overall, the CPU time for the triple time case is about half that of the single time method. Additional details on the CPU timing are given later.

Now we try to optimize the tolerance for the inner iteration convergence. Although we have used stability theory to estimate an optimum inner convergence in chapter three (figure 3.18), we must check the validity of these results by actual computations. Accordingly, we run a series of calculations in which we vary the tolerance of the inner iteration convergence for four different values of CFL_{u2} (infinity, 100, 10 and 1). The case with the minimum CPU time will give the optimum value of the order of inner convergence. The results are summarized in table 4.4 which show the total number of iterations, the outer number of iterations and the CPU time. The last row, CFL_{u2} of infinity, one inner iteration, degenerates to the single time method.

For each of the four values of CFL_{u2}, as the order of inner convergence decreases, the number of outer iteration remains almost constant suggesting that the lowest inner convergence tolerance (0.5 order of magnitude) is enough for this problem. A value of 0.5 order inner convergence gives the fastest convergence for this uniform straight duct computation with a small perturbation from the exact solution as the initial condition. The number of outer iterations and total iterations of the triple time increases as the value of CFL_{u2} decreases, Also, comparing the optimum CPU time of the triple time method (0.5 order of inner convergence and infinite value of CFL_{u2}, the row next to the last row) with that of the single time (the last row), we can see that we save a factor of two.

After choosing the optimum tolerance for the inner convergence, to test the effect of

Table 4.4 The number of iterations for different orders of inner convergence of a uniform straight duct flow (Total: total number of iteration; Outer: number of outer iterations; Inner order: order of inner convergence), 51x51 grids.

CFL_{u2}	**Infinity**			**100**			**10**			**1**		
Inner order	**Total**	**Outer**	**CPU (s)**	**Total**	**Outer**	**CPU (s)**	**Total**	**Outer**	**CPU (s)**	**Total**	**Outer**	**CPU (s)**
-10	277	*18*	**26.8**	244	*18*	**24.0**	516	*66*	**58.4**	2331	331	**274.7**
-6	267	*18*	**26.2**	228	*18*	**22.6**	499	*66*	**57.2**	2309	331	**273.8**
-4	215	*18*	**21.6**	185	*18*	**19.4**	453	*66*	**55.3**	2201	331	**265.9**
-2	121	*18*	**14.2**	113	*18*	**13.4**	363	*66*	**45.9**	1740	333	**225.7**
-1	71	*17*	**9.9**	77	*17*	**10.5**	285	*68*	**40.3**	1322	330	**191.6**
-0.5	54	*18*	**8.7**	79	*22*	**11.9**	199	*70*	**33.4**	880	296	**146.2**
1 step	53	53	**17.4**									

number of equations on the non-linear time and the linear time, we repeat the case of 0.5 order of inner convergence and infinite value of CFL_{u2} in table 4.4 by arbitrarily adding more species. The number of species is added in such a way that the numbers of equations of our system becomes 4, 8, 12, 16, 20, 25, 30, 35 and 40. The results of the time and the storage ratio are plotted in figure 4.2. Since we simply add more species with the same property, the convergence for multiple species is exactly the same as the single species case (figure 4.1). The computational results for the CPU time and the storage are plotted in figures 4.3 and 4.4 along with the estimations respectively.

Figure 4.2 shows a comparison between the normalized time of non-linear iteration and the normalized time of the linear iteration for both the computations and the estimations. The upper plot is based on a regular scale coordinate and the lower plot is based on a log-log scale coordinate. The normalization of the CPU time for the computational results is based on the computational time of the four-equation case, and the normalization of the time for the estimations is based on the estimated operation count of the four-equation case. In the lower plot, to illustrate how far the accelerate rate

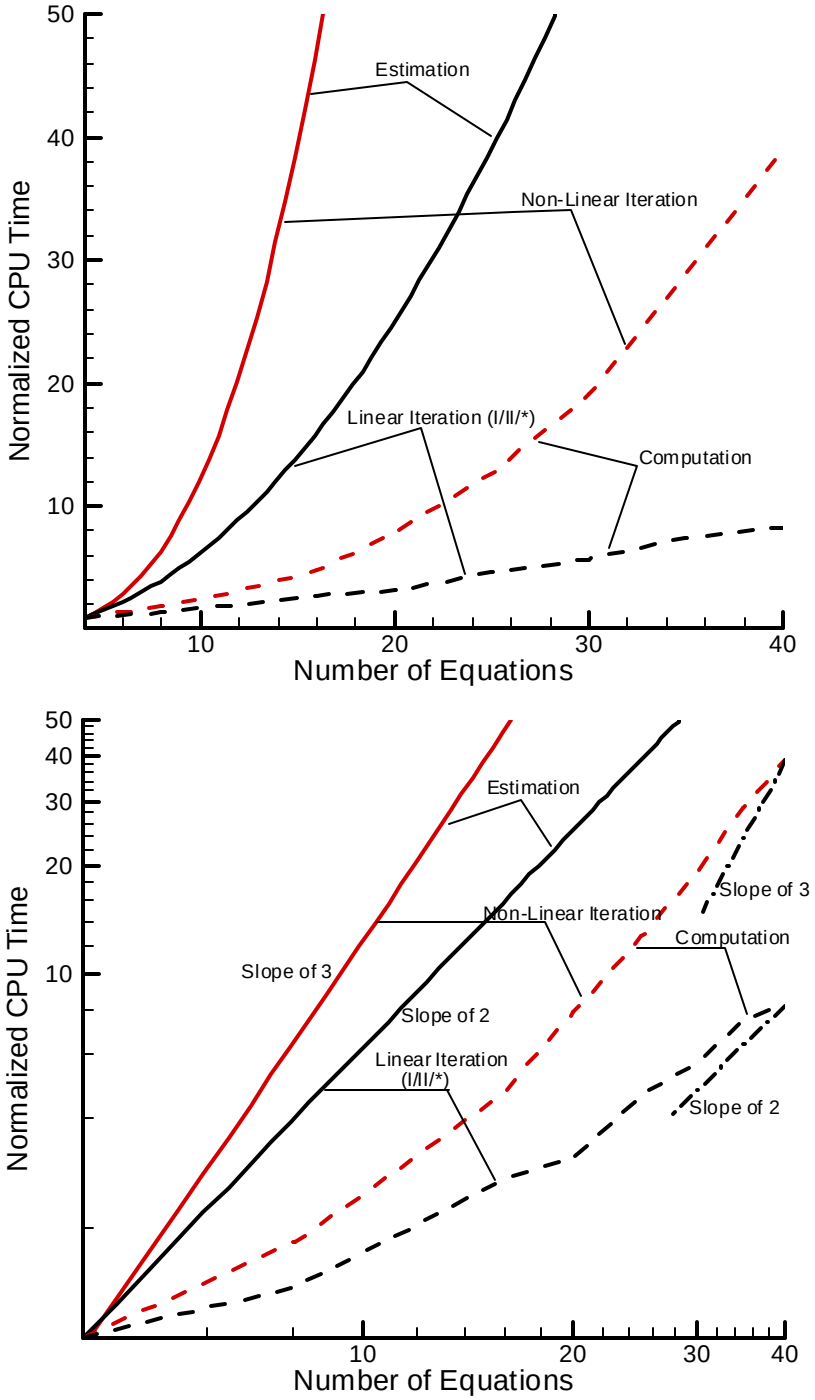

Figure 4.2 Time comparison between non-linear iteration and linear iteration for the computation and the estimation. Upper plot: regular scale; Lower plot: log-log scale.

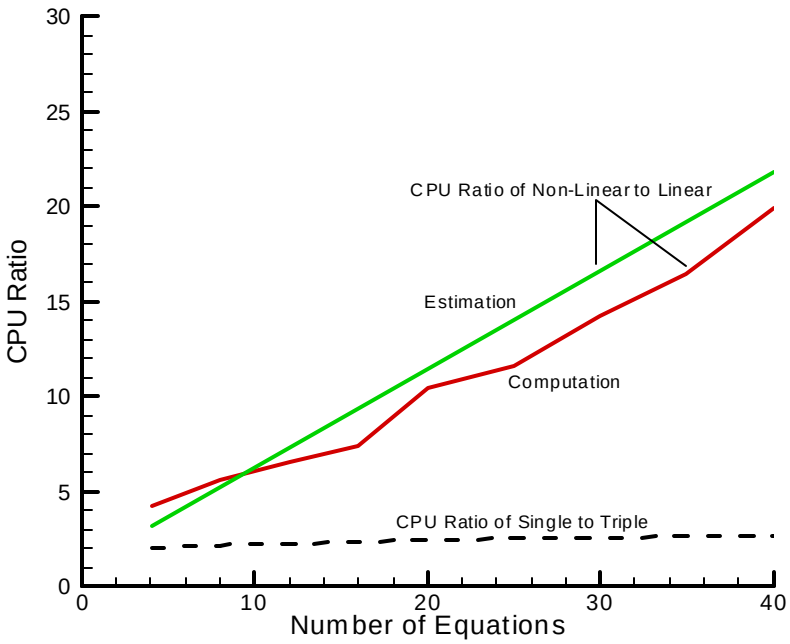

Figure 4.3 Comparison of the CPU time ratio of the non-linear iteration to linear iteration for both the computation and the estimation, and the CPU time ratio of the single time method to the triple time method, I/II/II order.

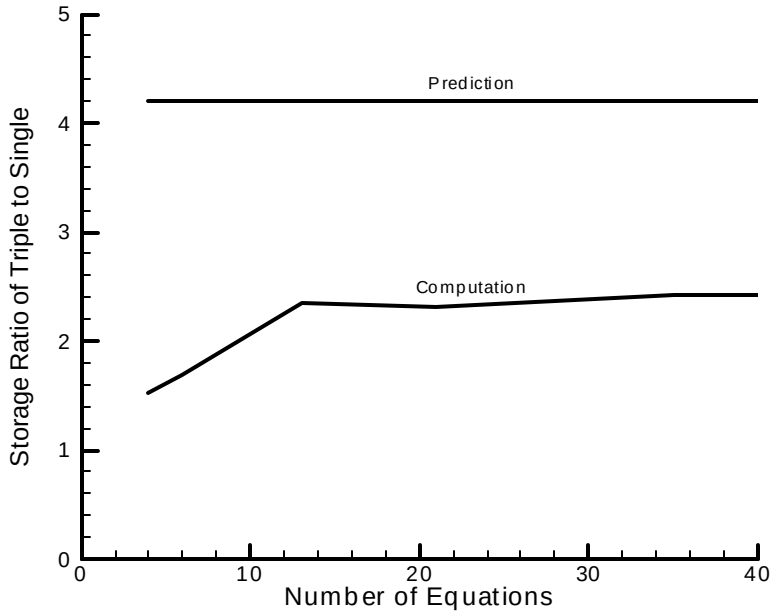

Figure 4.4 Comparison of the storage ratio of the triple time to single time between computation and estimation.

of the computational results is off our estimated results, we plot a line with a slope of 2 at the end of the computational results line of the linear iteration (I/II/* accuracy) and a line with a slope of 3 at the end of the computational results line of the non-linear iteration.
In this figure, the normalized CPU time increases monotonically with the number of equations. For the estimation results, on a logarithmic scale, the time of linear iteration is linearly proportional to the number of equations with a slope of two while the time of non-linear iteration is linearly proportional to the number of equations with a slope of three, suggesting that the CPU time saving of the linear iteration over the non-linear iteration is increased with the number of equations. The computational results increases slower than the estimation results and the acceleration of the computational results is not constant but increases with the number of equations. This is because our estimation results do not include the cost of matrix setup and many other costs. As the number of equations increases, the CPU time beyond our estimation becomes relatively less important and the CPU time acceleration rate of the computation becomes closer to the estimation results. For a number of equations of 40, the acceleration rate of the computational results is very close to the estimated results which have a slope of two for the linear iteration (I/II/*) and a slope of three for the non-linear iteration as can be seen from the lower plot.

Figure 4.3 shows the results of time ratio of the non-linear iteration to the linear iteration, and the time ratio of the single time method to the triple time method. Both computational results and estimation results are shown for the time ratio of non-linear iteration to the linear iteration. The estimation results come from equations 4.31.

From figure 4.3, we can see that for a small number of equations, the computational results are larger than the estimation in CPU time ratio. That is because the computational time of non-linear iteration includes the CPU time for matrices setup which takes

considerable time for a small number of equations and some other CPU time cost (looping, searching, and etc.), both of which are not contained in the estimation. As the number of equations increases, these two parts become less important. For large numbers of equations, our estimation of time ratio is larger than the computational results.

Surprisingly, although there is a large increase in the time ratio of non-linear iteration to linear iteration, the CPU time ratio of the single time method to the triple time method increases very slowly. This is because the ratio of the number of single time iteration to the number of the outer iterations in the triple time scheme is only three. Therefore the CPU ratio of the triple time method to the single time method can never be larger than three.

Our experiences, however, show that for more complex problem or problems with more grid points, the number of iterations for the single time method increases faster than the number of outer iterations of the triple time method. Accordingly, the CPU ratio of the triple time method to the single time method also increases with the complexity and number of grids in the problem.

To check the validity of this issue, we consider exactly the same uniform flow case but use a grid of 401x401 for both the single time method and the triple time method. The number of iterations of the single time is increased by a factor of 5.2 from 54 to 281 while the number of outer iterations of the triple time method is increased by a factor of 1.6 from 18 to 29. Consequently, the ratio of the CPU time for the single time method to that of the triple time method increases from 2 to 2.81.

Figure 4.4 shows the storage comparison between the computation and the estimation. The estimation results come from equations 4.39. We can see that the storage ratios for both the computation and the estimation are almost constant and do not change with the number of equations. The estimation is larger than the computation probably because the storage of our single code is not optimized and uses more storage than

necessary.

4.5 Conclusion

In this chapter, we estimate and compare both the CPU time saving and storage cost of the triple time and single time methods. Our estimation indicates that the linear CPU time is proportional to the square of the number of equations while the non-linear CPU time is proportional to the cube of the number of equations when the number of equations is large. As the number of equations increase, the ratio of the non-linear time to the linear time increases rapidly, but the CPU time ratio of the single time to the triple time increases very slowly and is limited to a factor of three for the single case computed.

The price to pay for the CPU time saving is more storage. Our estimation indicates the triple time method requires about four times more storage than the single time scheme for the I/II/* system. Computational results show an increase of approximately two. Importantly, both these ratios are independent of the number of equations.

In conclusion, compared with the single time scheme, the triple time scheme saves a factor between two and three in CPU time and cost 4.2 times more storage for a linear problem of the uniform straight duct flow. The memory price is getting cheaper and 4.2 times increase in the memory is generally not an issue for most of our computations, especially in a cluster. Also, we expect an improvement in the robustness of the triple time method which will be discussed in the next chapter. Consequently, the triple time method is feasible and efficient for most of our computations.

Chapter 5
Robustness Results

5.1 Introduction

So far, we have finished the stability analysis for our triple time scheme. We picked the best set of Γ's and chose the optimum number of inner iterations to achieve the optimum robustness and convergence. Also, we give an estimation of the CPU time saving and storage cost for both single time method and triple time method. At the same time, we presented some computational results for the linear problem to compare our analysis with the computational results. The results show a good match between the analysis and computation for problems that are essentially linear. In this chapter, we will discuss the non-linear effect on the triple time method.

For a non-linear problem, the convergence process can be divided into two parts, the non-linear part and the linear part. The linear part of the convergence for the non-linear problem is similar to the whole convergence process of the linear problem, which has been discussed in Chapters Three and Four, and accordingly will not be discussed in this chapter. The results of the linear problem in Chapters Three and Four show a very fast convergence. The non-linear portion is the place where there are most difficulties in CFD applications. There are two major issues regarding the difficulties of the non-linear portion, the convergence rate and the robustness. These two issues act against each other and are not completely independent. The robustness requires a small value of CFL_{u2} which will slow down the convergence rate and vice versa.

In this chapter, we will assess the robustness of the triple time for fixed value of CFL_{u2}. The convergence and robustness improvement of the non-linear part by using an

error-limited CFL ramping method will be discussed in the next chapter,.

We will start with a very simple case, the uniform straight duct flow, to test the robustness of our triple time scheme and compare with the traditional single time method. Then, we will apply our method to the more complex problem, converging-diverging nozzle flow.

5.2 The Robustness Results

5.2.1 Straight Duct Uniform Flow

One major purpose of designing the triple time scheme is to improve robustness. To test the robustness of the triple time scheme, we run a straight duct uniform flow case with a random perturbation over the whole flow field. To incorporate non-linear effects, we allow this perturbation to be large. The grids and the boundary conditions are the same as the uniform flow case in Chapter Three. Three representative Mach numbers, 0.01, 0.1 and 0.5 are tested. We use enough inner iterations to eliminate the effect of inner iterations so that the only issue is the effect of the outer iteration. The value of CFL_{u2} is fixed in both single time and triple time cases. An inner iteration CFL (CFL_{u3}) of 20 and I/II/II order system is used in the computation. For reference of later discussion, two triple time systems, 'UPU' and 'UPS' are considered in this case. The initial condition is set up in exactly the same way as the initial condition introduced in section 3.8.

A series of CFL_{u2}'s and the perturbation parameter e's are considered in both triple time and the traditional single time schemes. Given a perturbation magnitude e, the perturbation in the static pressure will be same for any Mach number flow, but the perturbation in velocity is much larger in low Mach flow than that in high Mach number flow. Accordingly, a small value of e in high Mach number flow is a small perturbation,

but that in a low Mach number flow is a large perturbation.

Before discussing the robustness results, we first look at a representative convergence result of the 'UPU' triple time method shown in figure 5.1 for Mach number of 0.01, $e = 0.0001$ and $CFL_{u2} = 10$. From this figure, we can clearly see the non-linear convergence region and the linear convergence region. In the non-linear part of convergence, the convergence rate is irregular and there are some wiggles. In the linear region, since the problem is linear and we use a constant value of CFL_{u2}, the convergence rate is constant.

Another representative convergence result we present is the outer convergence of the case with Mach number of 0.01, $e = 0.01$ and $CFL_{u2} = 0.5$ (0.5 is the largest allowable CFL_{u2} value for $e = 0.01$ as shown later) as shown in figure 5.2. The

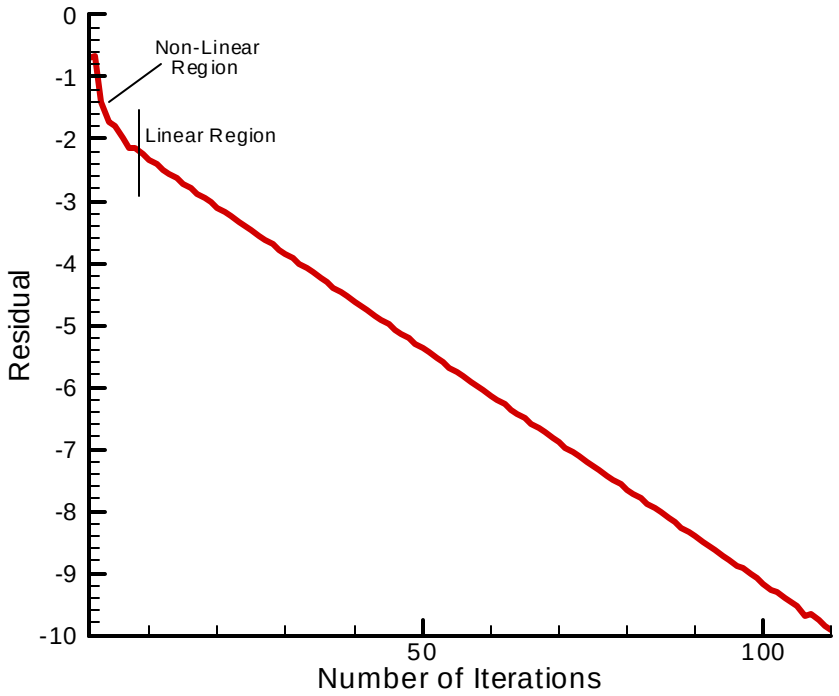

Figure 5.1 The outer convergence for the uniform straight duct flow. Mach number of 0.01, $e = 0.0001$, $CFL_{u2} = 10$, UPU, I/II/II.

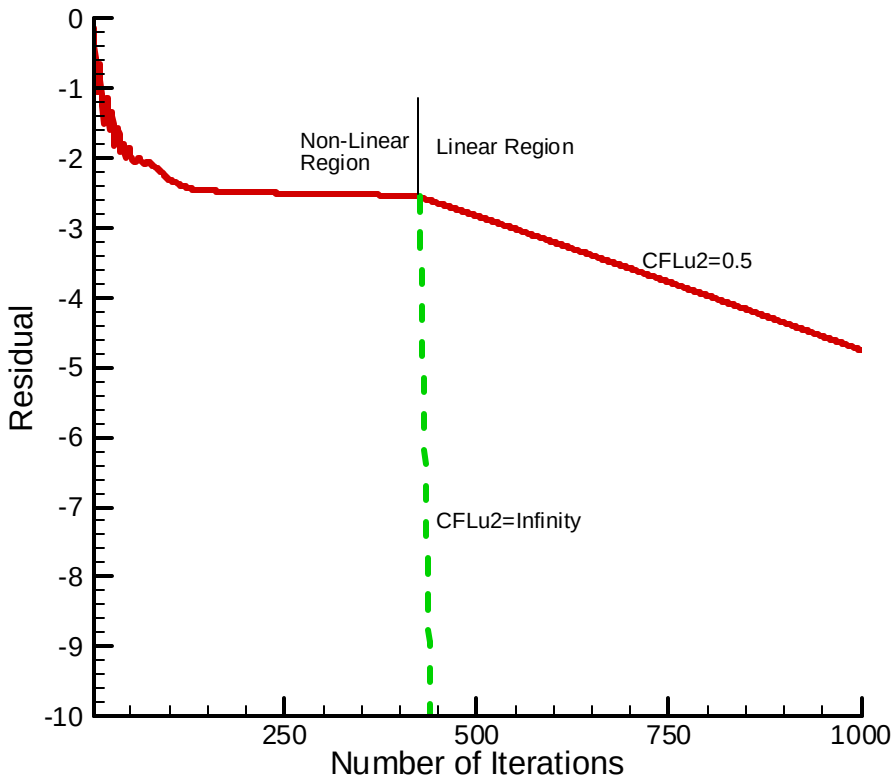

Figure 5.2 The outer convergence for the uniform straight duct flow. Mach number of 0.01, $e = 0.01$, $CFL_{u2} = 0.5$, UPU, I/II/II.

dashed line is a hypothetical convergence for the linear part if we use an infinite value of CFL_{u2}. From this figure, again we can clearly see the non-linear convergence region and the linear convergence region. In the non-linear part of convergence, the convergence rate is irregular, there are some wiggles at the beginning of the convergence and then the convergence is almost constant. In the linear region, since we use a constant value of CFL_{u2} and a very small value of $CFL_{u2} = 0.5$, we need much more iterations than the optimum number of iterations (the dashed line). Actually we can achieve the optimum number of outer iterations at the linear part of convergence by ramping that will be discussed in the next chapter. The non-linear part of the convergence takes twenty times more outer iterations than the linear part if the linear part is optimized (the dashed line).

Consequently, we need to find a way to accelerate the convergence of the non-linear portion of the convergence. This issue will be discussed in the next chapter.

Note that there is a corner at the boundary of non-linear region and the linear region, which is seldom seen in a common convergence. This might be caused by the unsteady preconditioning in the artificial dissipation (the unsteady preconditioning in this case is very close to the physical preconditioning because the CFL_{u2} is small). Since the physical preconditioning for low Mach number flow generally leads to an inaccurate solution, accordingly, a ramping method that increases CFL_{u2} value from small to large have to be used in the UPU system to provide an accurate final solution. Again, this issue will be discussed in the next chapter.

To demonstrate how the generality of the initial condition affects convergence, we show the results of a sequence of cases in figure 5.3 for the 'UPU' triple time system. All the cases are indicated by square or delta symbols depending on whether they converge or diverge. The converged cases are represented by the square symbols and the diverged cases are represented by the delta symbols. From these sets of successful and unsuccessful cases we draw a boundary between the converged zone and diverged zone. The solid line is the boundary of the converged zone and the region at the left side of this solid line is the converged zone. The dashed line is the boundary of the diverged zone and the region at the right side of this dashed line is the diverged zone. Between the solid line and dashed line is an uncertainty zone. For $e \le 10^{-4}$, the results of using very large value of CFL_{u2} (10^6) is converged but is not shown in figure 5.3. Consequently, the converged boundary is open at small values of e.

In general, the results in figure 5.3 show that larger initial errors require smaller value of CFL_{u2}.

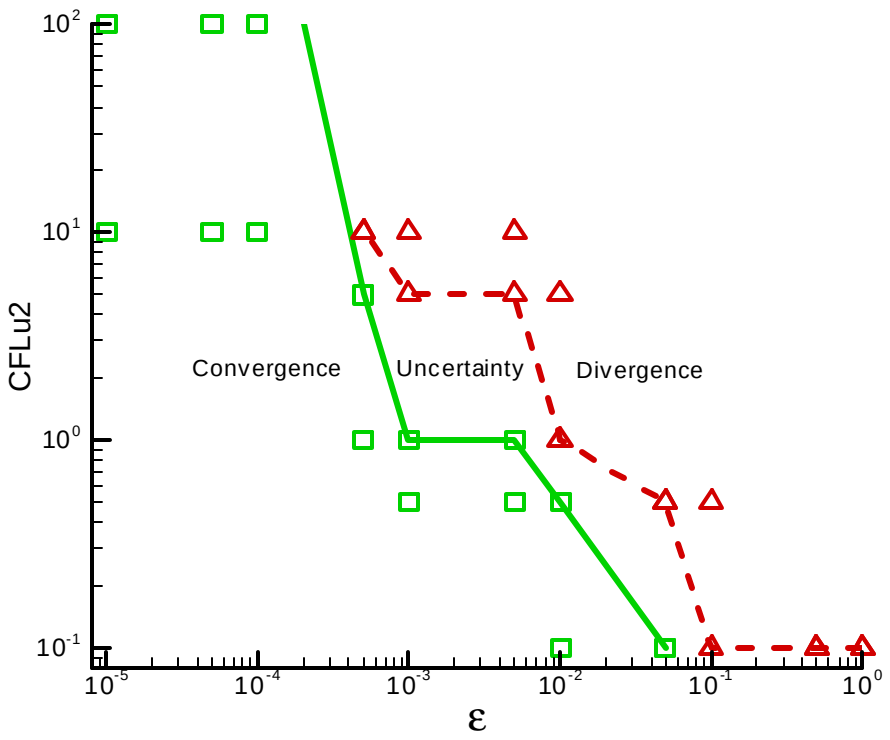

Figure 5.3 The converged and diverged zones for uniform straight duct flow. Mach number of 0.01, Triple time, UPU, I/II/II.

A comparison of the single time method and the triple time method for Mach number of 0.01 is shown in figure 5.4. To be conservative, in figure 5.4, we pick the solid line in figure 5.3 instead of the dashed line or some intermediate line as the boundary for the converged zone. The boundary line of the converged zone for the single time method and the 'UPS' triple time method is obtained in a similar way. Repeat this procedure for Mach number of 0.1 and 0.5, we obtain figures 5.5 and 5.6.

In figure 5.4 where the Mach number is 0.01, the 'UPU' triple time scheme has a larger convergence range of robustness than the single time scheme both in the e and CFL_{u2} directions. For small values of CFL_{u2}, the 'UPU' triple time scheme is even stable for a 10% perturbation in static pressure while the single time is only stable for a 0.05% perturbation of static pressure, the same order as the dynamic pressure. At the

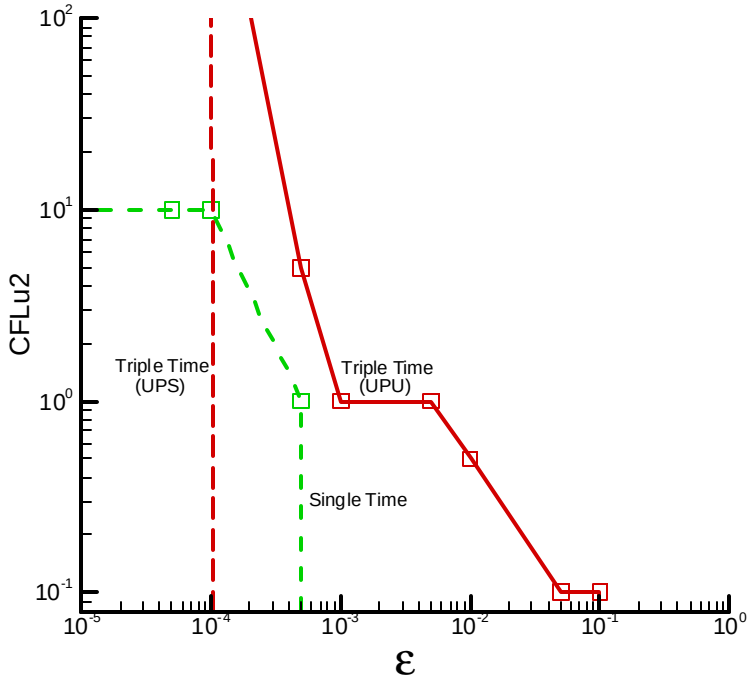

Figure 5.4 Mach number of 0.01, Robustness comparison between the single time and the triple time for the uniform flow.

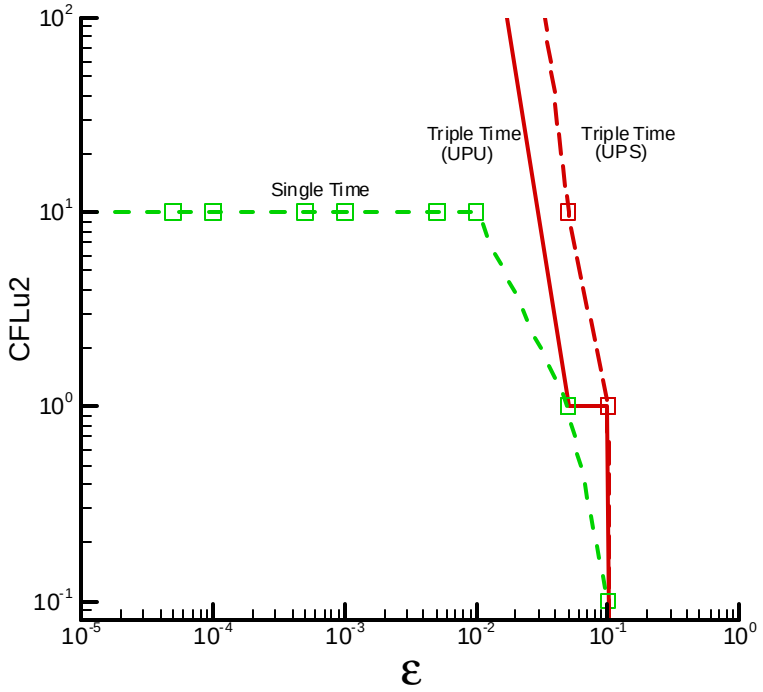

Figure 5.5 Mach number of 0.1, Robustness comparison between the single time and the triple time for the uniform flow.

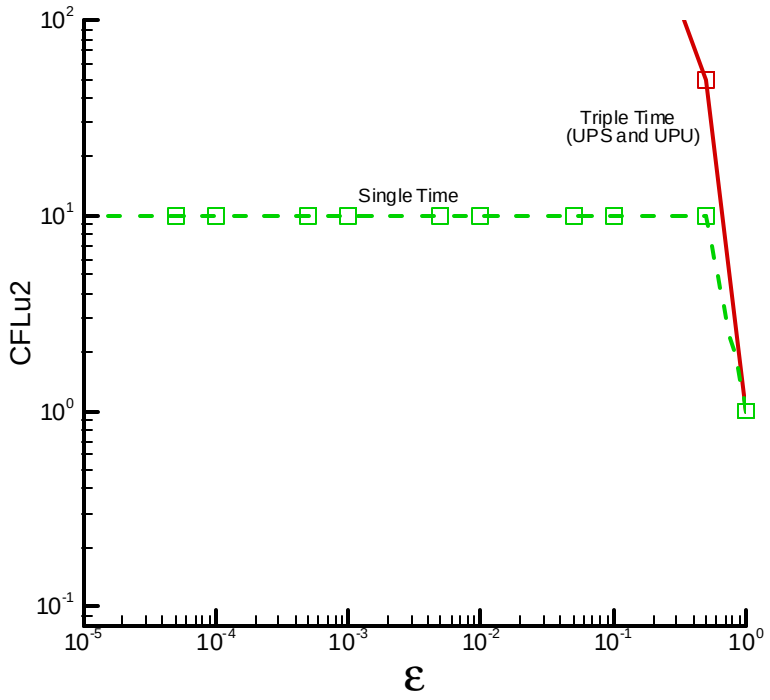

Figure 5.6 Mach number of 0.5, Robustness comparison between the single time and the triple time for the uniform flow.

small perturbation region, the 'UPU' triple time scheme is stable for infinite CFL_{u2} for its direct solution of the equation in t_2 time level and the single time is stable only for CFL_{u2} less or equal to10 for the approximate factorization.

The 'UPS' triple time system is much worse in robustness than the 'UPU' system. Because the only difference between the 'UPS' system and the 'UPU' system is the artificial dissipation, this suggest that artificial dissipation is very important for the robustness of low Mach number flow.

Results for larger Mach number are shown in figure 5.5 and 5.6. As the Mach number is increased, the robustness advantage of triple time scheme over the single time scheme decreases because of the reducing effect of the preconditioning. There are, however, still gains of triple time over single time in the large CFL_{u2} region. As Mach

number increases from 0.01 to 0.5, the largest allowable perturbation e of the triple time method only has a small increment from 0.05 to 1.0 while that of the single time method has a much larger increment from 0.0005 to 1.0. This is very little difference between the 'UPS' system and the 'UPU' system for Mach number of 0.1 and no difference for Mach number of 0.5 because of the diminishing effect of preconditioning on artificial dissipation.

5.2.2 Unchoked Nozzle Flow

As a second case for comparing the robustness of the triple time scheme with that of the single time scheme, we use the flow through an unchoked nozzle, but only 'UPU' triple time method is considered. The nozzle profile is defined by the cubic equation,

$$y = 2(1-a)x^3 - 3(1-a)x^2 + 1, \qquad 0 \le x \le 1.1 \tag{5.1}$$

where the inlet height of the nozzle is one and a is the throat area. The inlet of this nozzle is located at $x = 0$ while the throat is located at $x = 1$. A straight section with a length of 0.1 is added upstream of the converging section. A 61x51 straight grid is used for the whole computational domain and shown in figure 5.7. We use total pressure (p^0), total temperature (T^0), flow direction of zero ($a = 0$) at the inlet and back pressure at the outlet as the boundary conditions. The values are chosen to make the outlet Mach number to be 0.1 for all area ratio cases, which guarantees that it is non-choked for all area ratios. Four different area ratios, 1.0, 0.5, 0.1, 0.05 and 0.01 are tested. For the area ratio of one, the nozzle becomes a straight duct, which is similar to our previous straight duct case except the length in this case is twenty percent longer.

The initial condition is set up by introducing the same perturbation as that in the straight duct flow in Chapter Three on the exact solution of the two dimensional nozzle flow.

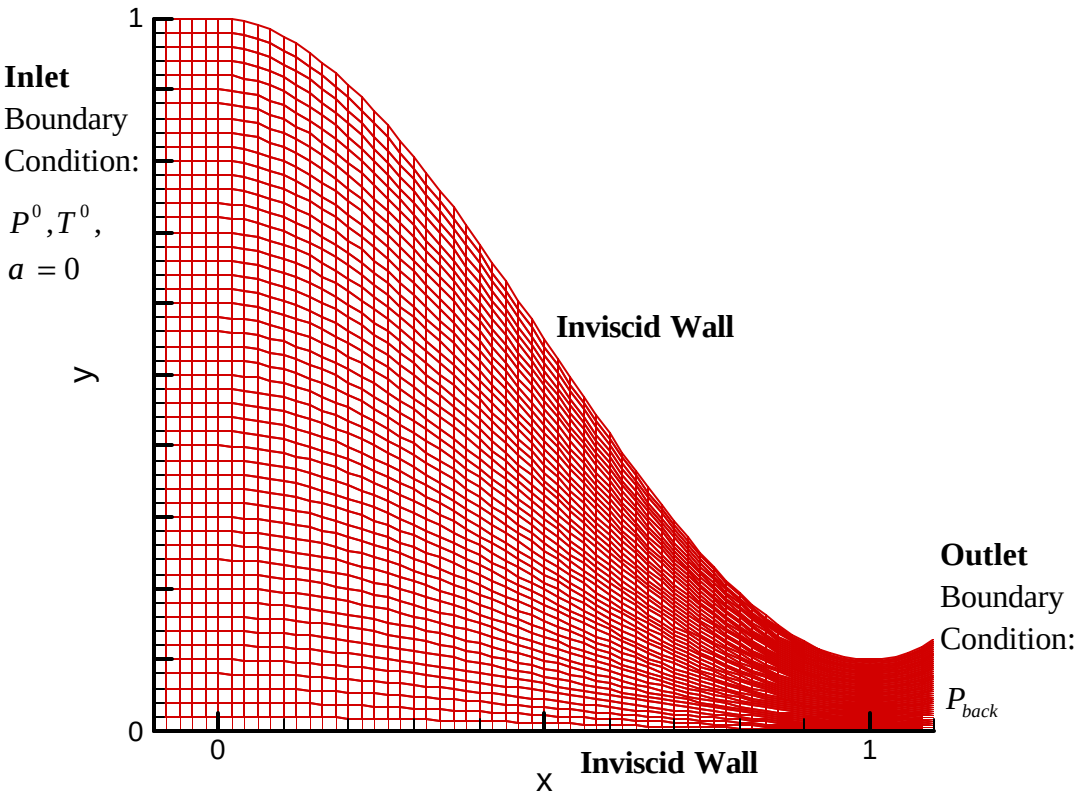

Figure 5.7 Grid (61x51) for the nozzle with throat area of 0.1

For each throat area case, we plot the results in the same way as the uniform flow case. The results are shown in figure 5.8. The upper plot is for throat areas of 1.0, 0.1 and 0.01, and the lower plot is for throat areas of 0.5 and 0.05. In the upper plot, for throat area of one, the triple time method is robust for $e \le 1$ but triple time method is only robust for $e \le 0.05$. For $e \le 0.01$, the triple time method could converge for any values while the single time method converges only for $CFL_{u2} \le 10$ because of the approximate errors. The robustness result is close to that in figure 5.5 except that the results in figure 5.5 have a larger robustness region in high perturbation region. The difference exists because the present duct is 20% longer than that in figure 5.5.

The results for the other throat areas are similar to that for throat area of one. As throat area decreases, the robustness region moves toward the small perturbation region

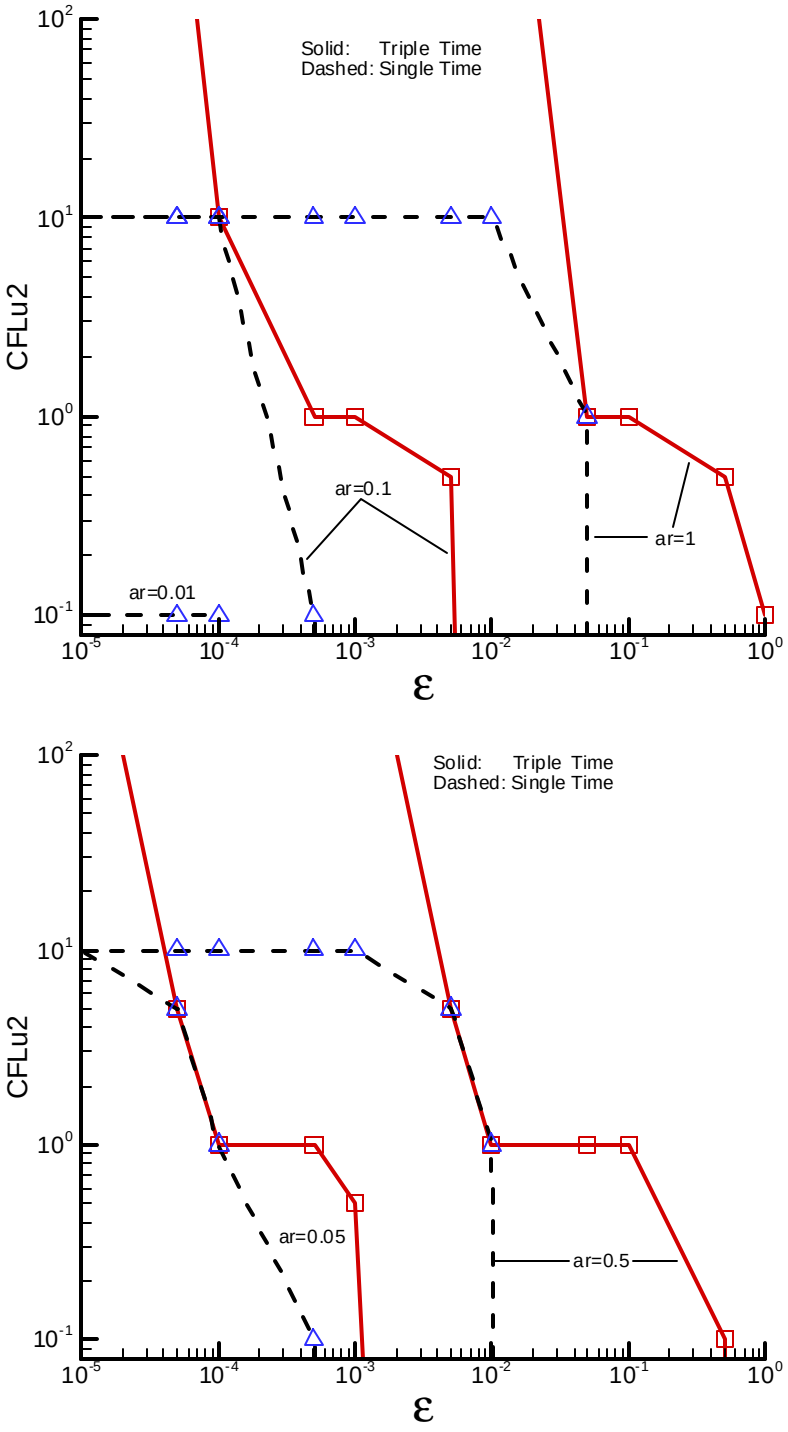

Figure 5.8 Robustness comparison between the single time method and the triple time method for the non-choked nozzle flow. 'ar' is the throat area. The upper plot: area ratio of 0.01, 0.1 and 1; The lower plot: area ratio of 0.05, and 0.5.

because the problem becomes more difficult as the throat area becomes smaller. For throat area of 0.01, the triple time method fails for all cases while the single time method has a very small robust region.

5.2.3 Choked Nozzle Flow

Another case for testing the robustness of the triple time scheme is the choked nozzle flow. By setting a higher total pressure and total temperature in the same nozzle as in the subsection 5.2.2, we obtain a choked flow. The exit Mach number can be computed from the area ratio-Mach number relationship of the nozzle. Given a back pressure (1.0e5 Pa) and temperature (300k), we can compute the total pressure (p^0) and total temperature (T^0). p^0, T^0, $\dot{m}$ (mass flow rate) and M_e (exit Mach number) are shown in table 5.1. For a throat area ratio of one, the nozzle becomes a straight duct where a Mach number of 1.2 is used. Everything else is the same as the unchoked nozzle flow case.

The robustness results are shown in figures 5.9. Again, the upper plot is for throat area of 1.0, 0.1 and 0.01, and the lower plot is for throat area of 0.5 and 0.05. For the case of throat area of one, the robustness of the triple time and the single time are about the

Table 5.1 One-dimensional results of the choked nozzle flow for different throat area's

Throat Area	$p^0\left(\times 10^5\right)$	T^0	$\dot{m}$	M_e
0.5	2.44	387	250.8	1.21
0.1	4.55	462	85.4	1.65
0.05	7.14	526	62.9	1.94
0.01	35.83	834	50.1	2.98

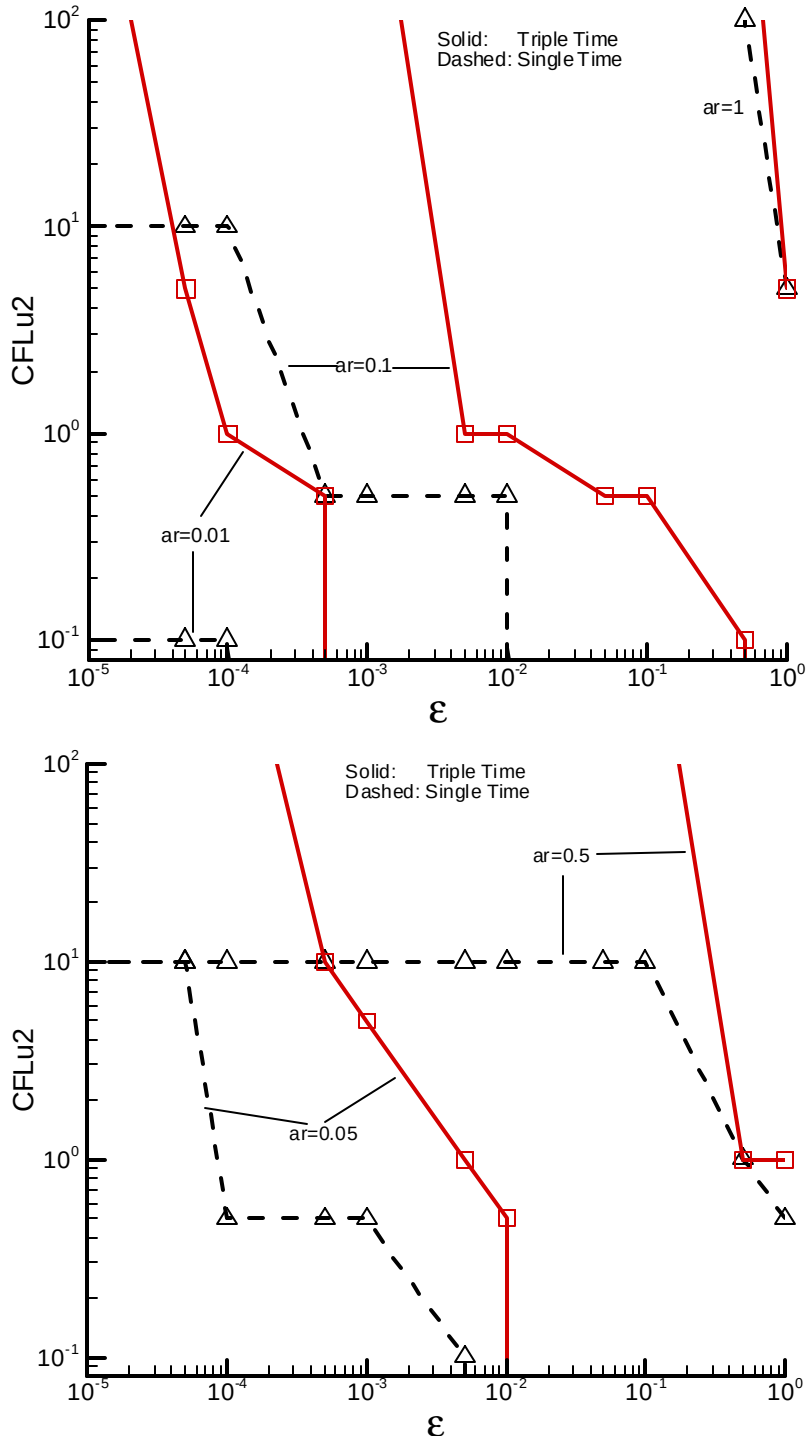

Figure 5.9 Robustness comparison between the single time and the triple time for the choked nozzle flow. 'ar' is the throat area. The upper plot: area ratio of 0.01, 0.1 and 1; The lower plot: area ratio of 0.05, and 0.5.

same because we use a Mach number of 1.2 and there is no difference between the preconditioning and non-preconditioning for supersonic flow. The small difference exists because the triple time scheme solves the non-linear equation exactly while the single time scheme solves the non-linear equation approximately. We can use an infinite CFL_{u2} in the single time case, which is kind of surprising. As the throat area decreases, the difference between the single time and the triple time appears and we can see an obvious improvement in the robustness of triple time scheme. For throat area of 0.5, the gain of the triple time over the single time is mainly located in the high CFL_{u2} region. For throat area of 0.1 and 0.01, the triple time method improves the robustness in both the magnitude of the perturbation and the CFL_{u2} value.

5.3 Conclusion

In this chapter, we compared the robustness of the triple time with that of the single time by running a uniform straight duct flow case for different Mach numbers and the nozzle flow for a series of throat areas and Mach numbers. The results of all these cases show a good improvement of the 'UPU' triple time method over the single time method in robustness. Also, the robustness comparison between the 'UPS' and 'UPU' triple time systems shows that 'UPU' system has a better robustness than the 'UPS' system suggesting the unsteady artificial dissipation has a better robustness than the steady artificial dissipation.

Chapter 6
Time Step Ramping

6.1 Introduction

So far, we have discussed the convergence of the linear problem and the robustness of the non-linear problem for fixed values of CFL. The triple time scheme shows an increase of a factor between two and three in speed for the convergence of linear problems and a substantial improvement in the robustness for non-linear problems. For a highly non-linear problem, the convergence rate is very slow for fixed CFL (figure 5.2) because a small value of CFL is required by the robustness of the non-linear portion of convergence but used in the whole convergence process. This is very inefficient because in the linear portion of convergence we could use a very large value of CFL to achieve fast convergence. Accordingly, for highly non-linear problems, the fixed CFL method is not practical and a method that ramps the CFL value from small to large should be used to provide needed robustness as well as fast convergence.

The underlying reason for ramping CFL in the convergence process is to limit the magnitude of the solution change. Often, this is done by arbitrarily 'scheduling' the increase in CFL as the solution converges. Here we look at a way for choosing the CFL so we can control the amount of the solution change directly. The reason is that the solution change limit is less problem-dependent than the CFL scheduling. Therefore, we can specify a tolerance for the solution change and allow the code to ramp the CFL in such a manner that this tolerance is not exceeded. By doing this, we can use a large CFL in some local computational regions and a small CFL in some highly non-linear computational regions. We call this ramping method the error-limited CFL ramping

method. Our goal is to understand the characteristics of this automatic CFL ramping method and its potential effects on robustness. We should mention that the error-limited CFL ramping is not restricted to the triple time method and can also be applied to any implicit method.

In this chapter, first we give a detailed overview of the error-limited CFL ramping method. Then we compare the solutions of the explicit method and implicit method, which is very important for the success of the error-limited CFL ramping method. After that we explain why ramping is more difficult for the 'UPU' system than the 'UPS' system. Finally we present some computational results for the automatic ramping method in the 'UPS' system for a uniform duct flow and compare with the triple time method without ramping.

6.2 The Error-Limited CFL Ramping

The purpose of the error-limited CFL ramping method is to choose a value of CFL that will ensure that the solution change, ΔQ_p, is less than or equal to a specified tolerance function, a of some reference value, Q_{pref}. Thus, we wish to find a time step, Δt_2, such that $\Delta Q_p \le a\, Q_{pref}$. To do this by successively computing a series of implicit time steps is computationally prohibitive. Consequently, we approximate the procedure by defining an explicit solution change, ΔQ_p^{ex} that corresponds to the change that would be computed from an explicit computation. In our triple time scheme, the CFL that is to be ramped is CFL_{u2}. Then, the detailed implementation is the following,

1. Choose an appropriate reference variable Q_{pref} and tolerance a. For example, use the local value of the primary variable as the reference variable and a constant reference a.
2. Compute time step Δt_2 from the following equation by forcing the explicit solution to equal the tolerance for every component and pick the minimum Δt_2 of all components.

   $$\Delta Q_p^{ex} = -\Delta t_2 \Gamma_2^{-1} (\nabla_D \cdot F) = a Q_{pref}$$

   where the superscript 'ex' on ΔQ_p^{ex} indicates the explicit solution computed by the explicit method, the discretized divergence $\nabla_D \cdot F$ is the residual, and Q_{pref} is an appropriate reference value of the solution (i.e., its current value) and a is a user specified tolerance. This equation is the same equation as equation 6.3b that will be derived and shown in subsection 6.2.1 later.
3. Use the time step Δt_2 to compute the implicit ΔQ_p^{im} (the superscript 'im' indicates the implicit solution) and complete the time step computation.

In this procedure, we require the explicit error ΔQ_p^{ex} instead of the implicit error ΔQ_p^{im} be equal to the tolerance of the solution change $a Q_{pref}$. The reason is that the residual $\nabla_D \cdot F$ is already computed in the implicit solution and the extra computation for the explicit solution is just a multiplication of the matrix $-\Delta t_2 \Gamma_2^{-1}$ and the residual matrix $\nabla_D \cdot F$. This provides a simple overview of the ramping procedure. The remainder of this chapter discussion adds more details.

The first question that arises is, does ensuring that the explicit error is less than the tolerance guarantee the implicit error will also be less than the tolerance? In the following,

we start by giving a detailed analysis to answer this question, and then go on to other issues.

6.2.1 Analytical Comparison between Explicit ΔQ_p^{ex} and Implicit ΔQ_p^{im}

In this section, we compare the explicit solution, ΔQ_p^{ex}, with the implicit solution ΔQ_p^{im}. First, we formulate the explicit method. Adding a pseudo time derivative to equation 2.15c as we did in chapter 2, we have,

$$\Gamma_2 \frac{\partial Q_p}{\partial t_2} + \nabla_D \cdot F = 0 \tag{6.2}$$

where the discretized flux divergence $\nabla_D \cdot F$ is already defined in equation 2.15b and is repeated here for convenience.

$$\nabla_D \cdot F = \frac{1}{\Omega} \sum_{k=1}^{K} \left[\frac{1}{2}\left(F_L + F_R\right) - \frac{1}{2}\Gamma_1 \left| \Gamma_1^{-1} A_p \right| \left(Q_{pR} - Q_{pL}\right) \right]_k S_k \tag{2.15b}$$

In equation 2.15b, it is important to note that, if $\Gamma_1 = \Gamma_u$, then the residual, $\nabla_D \cdot F$, is a function of the time step, Δt_2, because the matrix, Γ_u, depends on the time step Δt_2. The dependence of the residual, $\nabla_D \cdot F$, on the time step, Δt_2, will make the application of the ramping method more difficult for the 'UPU' system as will be discussed in subsection 6.2.2 later.

Discretizing equation 6.2 explicitly in time gives

$$\frac{\Gamma_2}{\Delta t_2} \Delta Q_p^{ex} = -\left(\nabla_D \cdot F\right)^n \tag{6.3a}$$

Solving equation 6.3a for ΔQ_p^{ex}, we obtain,

$$\Delta Q_p^{ex} = -\Delta t_2 \Gamma_2^{-1} \left(\nabla_D \cdot F\right)^n \tag{6.3b}$$

From equation 6.3b, we can see that ΔQ_p^{ex} depends on Δt_2, Γ_2 and the residual $(\nabla_D \cdot F)^n$. The explicit solution change, ΔQ_p^{ex}, is linearly proportional to Δt_2 if the matrices Γ_1 (which appears in the residual) and Γ_2 are independent of Δt_2. If either of these two matrices is a function of the time step, the dependence is more complicated. Of the three preconditioning matrices we have defined so far, only Γ_u varies with Δt_2.

To calculate the implicit solution, we use the Euler implicit method to discretize equation 6.2, linearize the discretized equation and write the linearized equation in delta form to obtain the equation 2.19b (refer to section 2.4.2)

$$\left[\frac{\Gamma_2}{\Delta t_2} + (\nabla_D \cdot A_p)_2\right]^n \Delta Q_p^{im} = -(\nabla_D \cdot F)^n \qquad (2.19b)$$

It is important to note that the right hand side (the residual) of the implicit discretization in this equation is identical to that for the explicit solution in equation 6.3a. Consequently, as we mentioned earlier in this section, having calculated the residual we only need to multiply by $-\Delta t_2 \Gamma_2^{-1}$ to compute the explicit solution change as a part of the implicit step. Solving this implicit equation symbolically for ΔQ_p^{im}, we obtain

$$\Delta Q_p^{im} = -\left\{\left[\frac{\Gamma_2}{\Delta t_2} + (\nabla_D \cdot A_p)_2\right]^n\right\}^{-1} (\nabla_D \cdot F)^n \qquad (6.4)$$

For very small values of the time step Δt_2, $(\nabla_D \cdot A_p)_2$ is small compared with $\frac{\Gamma_2}{\Delta t_2}$ and can be neglected. Consequently, equation 6.4 becomes

$$\Delta Q_p^{im} \approx -\Delta t_2 \Gamma_2^{-1} (\nabla_D \cdot F)^n \quad \text{for } \Delta t_2 \text{ small}$$

Comparing this equation with equation 6.3b, we obtain

$$\Delta Q_p^{im} \approx \Delta Q_p^{ex} \qquad \text{for } \Delta t_2 \text{ small} \qquad (6.5)$$

This relation indicates that, at very small Δt_2, the explicit solution is approximately equal to the implicit solution.

For large Δt_2, $\Gamma_2 / \Delta t_2$ is very small compared with $\left(\nabla_D \cdot A_p\right)_2$ and can be neglected. Then equation 6.4 becomes

$$\Delta Q_p^{\ im} = -\left\{\left(\nabla_D \cdot A_p\right)_2^{\ n}\right\}^{-1}\left(\nabla_D \cdot F\right)^n \qquad \text{for large } \Delta t_2 \tag{6.6}$$

which is independent of Δt_2 so that the $\Delta Q_p^{\ im}$ approaches a constant as Δt_2 goes to infinity (By contrast, note that $\Delta Q_p^{\ ex}$ goes to infinity as Δt_2 goes to infinity). Note that for large Δt_2, Γ_1 is independent of Δt_2 even for unsteady preconditioning because Γ_u is independent of Δt_2 for large Δt_2. Comparing this relation with equation 6.3b which is proportional to Δt_2 and hence goes to infinity as Δt_2 goes to infinity, we conclude,

$$\left|\Delta Q_p^{\ im}\right| < \left|\Delta Q_p^{\ ex}\right| \qquad \text{for large } \Delta t_2 \tag{6.7}$$

This relation indicates that, at large Δt_2, the implicit solution is smaller than the explicit solution.

So, we analytically conclude that, for small Δt_2 values, the implicit solution is approximately equal to the explicit solution while for large Δt_2, it is smaller than the explicit solution. This suggests that an explicit solution that is less than the error tolerance will guarantee that the implicit solution is also less than the error tolerance. For intermediate values of Δt_2, an approximate analysis is too difficult to conduct, but we will give some numerical results to prove that this argument is also true for the middle Δt_2 region.

6.2.2 The Effect of the Time Step on the Residual

In this subsection, we discuss the impact of time step ramping on the residual if the artificial dissipation matrix Γ_1 is based upon unsteady preconditioning matrix, Γ_u. We begin with the explicit solution equation 6.3b. Substituting equation 2.15b into equation 6.3b, setting $\Gamma_1 = \Gamma_u$, we obtain

$$\Delta Q_p^{ex} = -\Delta t_2 \Gamma_2^{-1} \frac{1}{\Omega} \sum_{k=1}^{K} \left[\frac{1}{2}(F_L + F_R) - \frac{1}{2} \Gamma_u \left| \Gamma_u^{-1} A_p \right| (Q_{pR} - Q_{pL}) \right]_k S_k \qquad (6.8)$$

Since the artificial dissipation matrix Γ_u is a function of the time step Δt_2 in both the local cells and the surrounding cells, ΔQ_p^{ex} has an explicit dependence on the Δt_2 of the local control volume as well as an implicit dependence on the Δt_2's of all the surrounding faces of the control cell. Consequently, ΔQ_p^{ex} is a complicated function of Δt_2 and finding an appropriate value of Δt_2 that will restrict ΔQ_p^{ex} to be less than or equal to aQ_{pref} requires a coupled solution of multiple cells, not just a single cell as in the case where the Γ_1 in the artificial dissipation is not dependent on Δt_2 (as for example if the steady preconditioning matrix is used for Γ_1). We note that, although the artificial dissipation appears to have a small effect on the residual, our later results show that it is not true. Generally an iteration procedure needs to be used to solve equation 6.8 for Δt_2. Although we have tried some simple iteration methods for obtaining this coupled time step, they have not worked well and additional effort is necessary to obtain a reliable method.

As an example of how the residual changes with the time step where unsteady preconditioning is used in the artificial dissipation of the residual, we consider the same

straight duct uniform flow as before with a perturbation of $e = 10^{-5}$ and plot the first component, r_1, of the residual vector $R = (r_1, r_2, r_3, r_4)$ at one point in the domain as a function of CFL_{u2}, in figure 6.1. In this figure, in the non-preconditioning and steady preconditioning regions of CFL_{u2}, r_1 are constant because the matrix Γ_u is not independent of time step in these two regions. The most characteristic feature of this plot is that the residual (and hence the solution) changes sign as CFL_{u2} is increased. For small values of CFL_{u2} where Γ_u is equal to the non-preconditioned matrix, r_1 is negative, indicating that ΔQ_p should be increased to reach convergence. For large values of CFL_{u2} where Γ_u is equal to the steady-preconditioned matrix, the residual is positive indicating that ΔQ_p should be decreased to reach convergence. At one

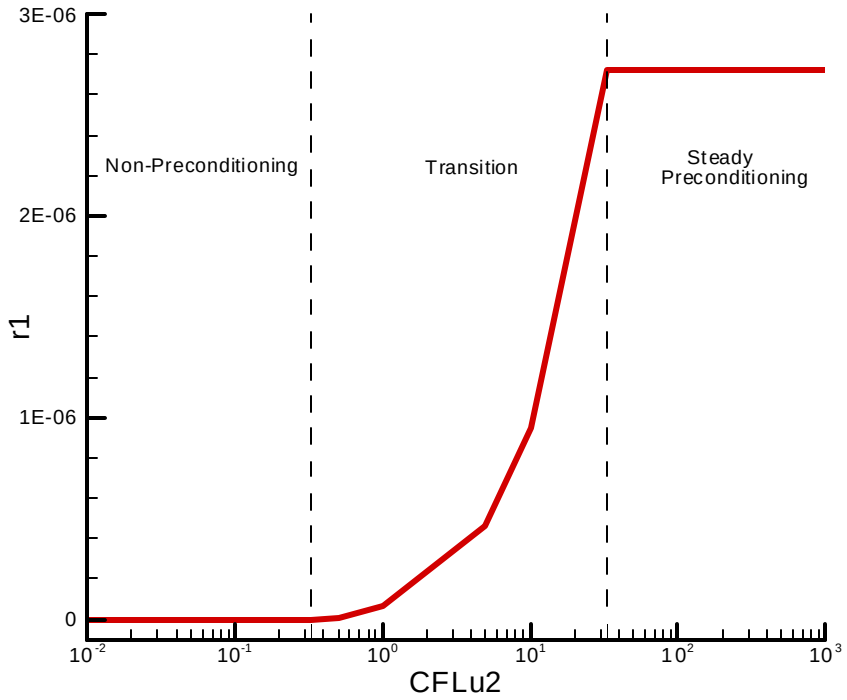

Figure 6.1 The variation of the first component of the residual with $CFLu2$

particular intermediate value of CFL_{u2}, the residual becomes zero indicating that the solution (at this point) is identically converged. The results for other components are similar and hence are not shown.

Now, we discuss the difficulties in an iteration for finding a solution for the time step. First, we look at the plot of one component of ΔQ_p^{ex}, Δq^{ex}, versus time step Δt_2 as shown in figure 6.2. The two horizontal dotted lines are the positive and negative solution tolerance (given by $a\,q_{ref}$) and the two vertical dotted lines are the two transition demarcations. The symbols, Δt_p and Δt_s, are time steps for the physical and steady preconditioning demarcation respectively. Three typical profiles for the function Δq^{ex}, case 1, case 2 and case 3, are shown together in figure 6.2. These three cases all start from zero at the origin, decrease linearly up to Δt_p, increase across zero until Δt_s and finally increase linearly to infinity. The difference between these three cases lies only in whether they cross the $-a\,q_{ref}$ horizontal tolerance line or not. The possible solution points are marked by stars.

Case 1 does not cross the $-a\,q_{ref}$ horizontal tolerance line and has only one solution, the crossing point with the $a\,q_{ref}$ horizontal tolerance line marked by a star. In this case, we can start from the initial condition of Δt_p and use Newton's iteration to obtain the solution very easily.

Case 3 crosses the $-a\,q_{ref}$ horizontal tolerance line. There are three solutions, one is positive and two are negative. Clearly, the smallest of these possibilities guarantees the solution will never exceed the restricting limits and hence is the best choice. This solution can also be found very easily.

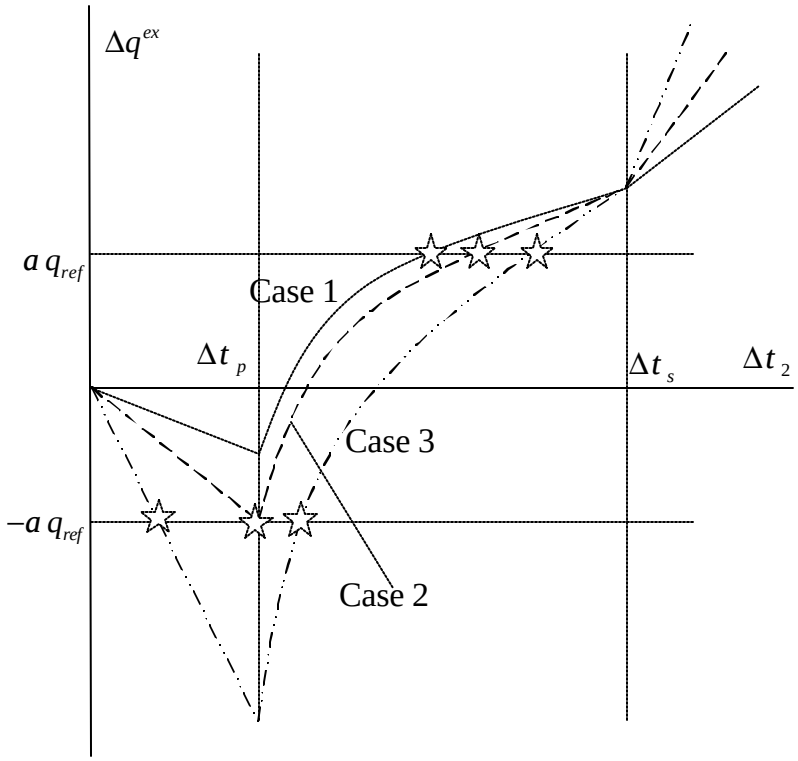

Figure 6.2 The variation of the explicit solution Δq^{ex} with time step

Case 2 acrosses the $-a\,q_{ref}$ horizontal tolerance line at the physical transition point Δt_p. In this case, before reaching the final converged solution, the value of Δq^{ex} very possibly will oscillate around the $-a\,q_{ref}$ horizontal tolerance line because of the dependence of Δq^{ex} on the time step of all surrounding cells. Part of the time it will correspond to case 1 while the rest of the time will correspond to case 3. Unfortunately, the solution of case 1 is orders of magnitude larger than the solution of case 3, and will cause a major disruption during the iteration procedure. Accordingly, it is difficult to find a way to get a converged solution. We should mention that those cases for which the value of Δq^{ex} at Δt_p is close to the tolerance $-a\,q_{ref}$ could also fall into this

category.

Besides the difficulty in case 2, probably the most difficult thing for the convergence of this equation 6.8 is that we don't know what category it belongs to until we get the converged solution. Also, because of the coupling problem, the function could change from one category to another during the iteration process. There are so many grids in the computational domain that it is difficult to develop an algorithm to deal with these problems automatically.

We have tried some simple Newton iteration methods. For each iteration, we update all the variables in the computational domain. We can get a converged solution for grid points of cases 1 and 3, but have difficulties in case 2. The following method might be useful to find a solution, but we have not tried it. The suggestion is that, instead of iterating and updating the whole computation simultaneously, iterate one grid point while keeping the other grid points fixed, after the full convergence of this grid, then go to the other grid points.

Because of the dependence of residual on the time step and the complexity of finding an iteration method to solve for the time step of the case with unsteady preconditioning in artificial dissipation, we switch from the 'UPU' system to the 'UPS' system in this chapter so that steady preconditioning is used for the artificial dissipation. Note that this represents a major limitation on the 'UPU' system for flow in low speed regimes. For transonic and supersonic flow, the preconditioning matrix is always equal to the physical Jacobian matrix so that this is not an issue.

6.2.3 The Normalization Issue

In the ramping procedure, we require that the normalized solution change be equal to some specified tolerance. Accordingly, the reference variable for the normalization must be chosen carefully. An improper normalization will lead to a solution that is too small or

too large. There are some requirements on the normalization variables we pick. First, we require that the normalization variable set be similar or equal to the primary variable set. Second, the normalization of some variables should be allowed to change sign while others must remain positive or negative. Specifically, the velocity component must be allowed to change sign. If we use the local velocity as the reference velocity and limit the solution change to be less than or equal to a specified fraction of this local velocity, the velocity can never change sign. Accordingly, some global velocity or maximum velocity should be used along with appropriate judgement to determine the proper reference variable.

Third, the normalization of some other variables such as temperature and pressure should not be allowed to change sign. For example, both pressure and temperature are always positive. In a flow with large pressure gradient, if we choose the maximum pressure as the reference pressure, the pressure could go negative, but choosing the local pressure as the reference pressure will never allow the pressure to go negative. By the same token, the local temperature should be chosen as the reference temperature.

Last, since the explicit solution depends on the artificial matrix Γ_2, the normalization variable set is also dependent on the matrix Γ_2. For high Mach number flows, there is no difference between the preconditioned Γ_2 and the non-preconditioned Γ_2. While for low Mach number flows, the preconditioned Γ_2 and the non-preconditioned Γ_2 are different and hence give quite different solutions. Accordingly, it might prove useful to use different normalization variables for the preconditioned Γ_2 and non-preconditioned Γ_2 in low Mach number flow.

To deduce an appropriate set of reference variables, we start with the one dimensional continuity equation

$$\frac{\partial r}{\partial t_2} + \frac{\partial ru}{\partial x} = 0$$

Use the static pressure 'p' as the primary variable, this equation becomes

$$r_p \frac{\partial p}{\partial t_2} + \frac{\partial ru}{\partial x} = 0$$

Replacing the quantity r_p by an artificial term, r'_p to obtain

$$r'_p \frac{\partial p}{\partial t_2} + \frac{\partial ru}{\partial x} = 0$$

Discretize this equation explicitly and solve for Δp to get

$$\Delta p = -\Delta t_2 \frac{1}{r'_p} \left(\frac{\partial ru}{\partial x} \right)^n$$

For non-preconditioned case, $r'_p = r_p$ and r / r'_p is of the order of static pressure. While for the preconditioned case, $r'_p \approx r_p / M^2$ and r / r'_p is of the order of the dynamic pressure. This suggests that, in a low Mach number flow, perhaps the dynamic pressure should be used as the reference pressure if Γ_2 is preconditioned while the static pressure should be used as the reference pressure if Γ_2 is non-preconditioned.

6.2.4 Numerical Comparison between Explicit ΔQ_p^{ex} and Implicit ΔQ_p^{im}

In subsection 6.2.1, we have analytically compared ΔQ_p^{ex} with ΔQ_p^{im}, and showed that ΔQ_p^{ex} is approximately equal to ΔQ_p^{im} at very small CFL_{u2} and is larger than ΔQ_p^{im} at very large CFL_{u2}, but we do not know the behavior at the intermediate CFL_{u2} region. In this subsection, we show some numerical results for the whole CFL_{u2} region.

We consider the same uniform flow problem as in chapter 5. In addition to the

perturbation parameter based on the static pressure, e, another perturbation parameter based on the dynamic pressure, e' is also provided to give a better understanding of the results. The relationship between e and e' can be easily computed as $e = g M^2 e'/2$. A perturbation parameter of $e' = 0.1$ is used in this case. We compute one outer iteration by using enough inner iterations to solve the non-linear equation exactly for the implicit solution ΔQ_p^{im}. We then compute the explicit solution ΔQ_p^{ex} from equation 6.3. The 'UPS' system is used in the computation so that the artificial dissipation does not depend on the time step in the low Mach number regimes.

First, to eliminate the effect of preconditioning, we consider the flow with Mach number of 0.7. The reference variable of $Q_{pref} = (p, U, U, T)$ (the quantity, U, is the magnitude of the velocity) based on the static pressure is used. The results are shown in figure 6.3. In this figure, all four components of the error are close to each other for both explicit and implicit solutions because the eigenvalues are of the same order for high Mach number flow. The explicit solution is linearly proportional to CFL_{u2}. The implicit solution is equal to the explicit solution for small CFL_{u2}, but starts to deviate from the explicit solution at a CFL_{u2} of about 10 and approaches a constant for large values of CFL_{u2}. The observations at small and large CFL_{u2} regions agree with our previous analysis, but also indicate that $\Delta Q_p^{im} < \Delta Q_p^{ex}$ in the intermediate region as well.

Second, we consider the same problem with low speed flow of Mach number of 0.01 without preconditioning in Γ_2. The same reference variable of $Q_{pref} = (p, U, U, T)$ is used. The results are shown in figure 6.4. The results in figure 6.4 are very similar to the results in figure 6.3 except that the implicit solution for the pressure and temperature components in the high CFL_{u2} region are about four orders of magnitude less than those

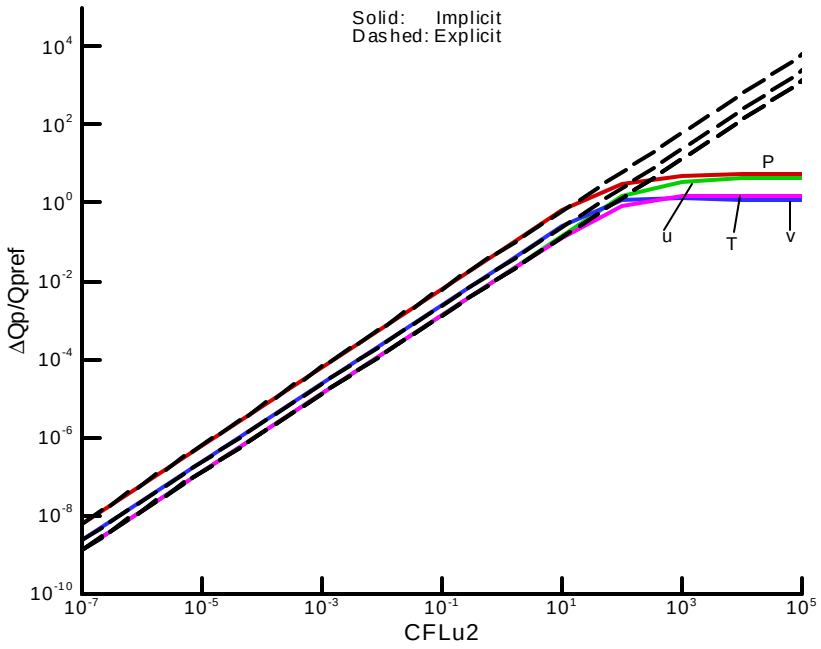

Figure 6.3 The comparison between ΔQ_p^{ex} and ΔQ_p^{im} for M=0.7, $Q_{pref} = (p, U, U, T)$, UPS, I/II/II, $e' = 0.1$ ($e = 0.0343$).

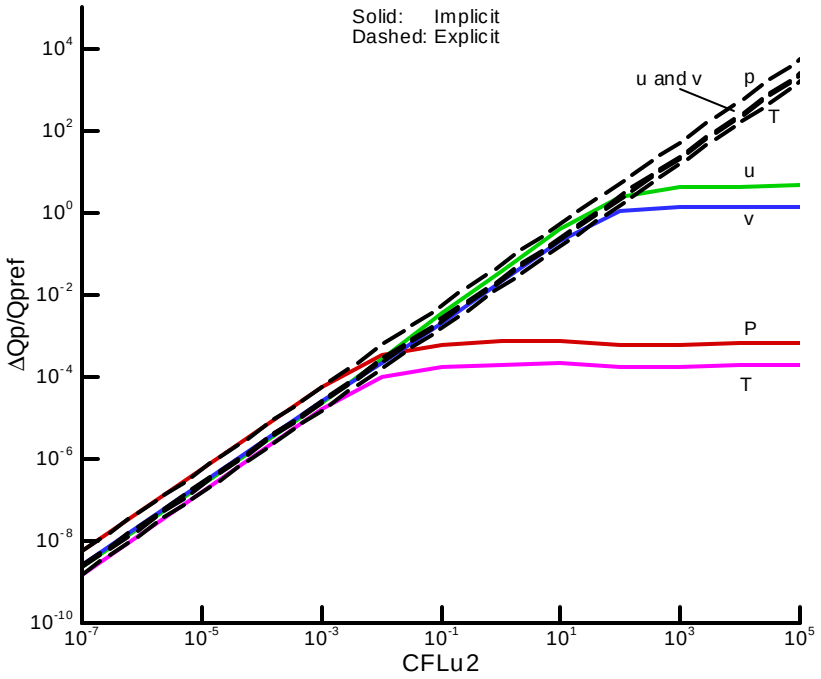

Figure 6.4 The comparison between ΔQ_p^{ex} and ΔQ_p^{im} for M=0.01 and non-preconditioned Γ_2, $Q_{pref} = (p, U, U, T)$, UPS, I/II/II, $e' = 0.1$ ($e = 7e^{-6}$).

of the velocity components. The reason is that the implicit solution of pressure should be normalized by the dynamic pressure instead of the static pressure in large CFL_{u2} region and similarly the temperature should be normalized by M^2T instead of T. We can not use the dynamic pressure as the reference pressure. Otherwise, the pressure component curves move four orders of magnitude up, and for a specified value of a ($\Delta Q_p / Q_{pref}$), the time step limited by the pressure (explicit solution) will be much less than the time steps limited by other components, which is not efficient.

The implicit solution of the temperature and pressure components start to deviate from the explicit solution at very small CFL_{u2} of about 0.001, which is about four orders less than the deviation starting points of the velocity components, about $CFL_{u2} = 10$. Although we expect a two orders difference (linear with the Mach number) between the deviation starting point of the temperature (or pressure) component and the velocity components for the disparity of the eigenvalues, four orders (Mach number square) difference is more than our expectation and the reason is not clear yet. Clearly, in this case, the time step will be only limited by the velocity components and not by the pressure and temperature components.

Third, we consider the same low Mach number problem with steady preconditioning in Γ_2. Based on the analysis in subsection 6.2.3, we use the reference variable, $Q_{pref} = (1/2\,rU^2, U, U, T)$, where the reference pressure is based on the dynamic pressure. The Mach number is 0.01 and the results are shown in figure 6.5. Compared with the non-preconditioned case in figure 6.4, the implicit pressure component at large CFL_{u2} moves up and is similar to the velocity component because the reference pressure is the dynamic pressure while the implicit solution of pressure at large CFL_{u2} does not change, and both the implicit and explicit temperature components at small and middle values of

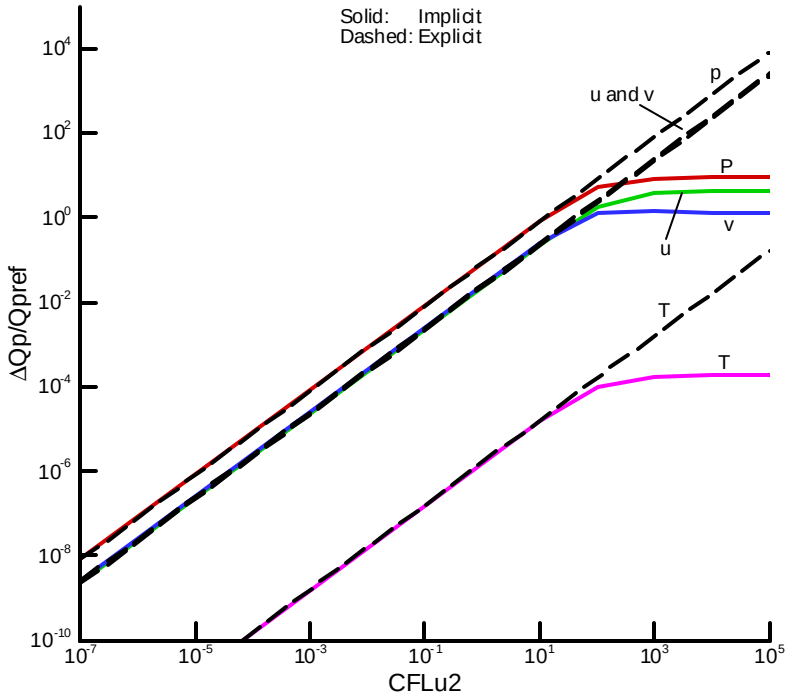

Figure 6.5 The comparison between $\Delta Q_p^{\ ex}$ and $\Delta Q_p^{\ im}$ for M=0.01 and preconditioning Γ_2, $Q_{pref} = (1/2\,rU^2, U, U, T)$, UPS, I/II/II, $e' = 0.1\,(e = 7\times10^{-6})$.

CFL_{u2} moves four orders of magnitude down indicating that the change in temperature component will be small no matter what time step is used. The reason for smaller temperature component is that, for an adiabatic flow, the temperature change is only of the order of M^2T. Accordingly, for an adiabatic flow, we can also use M^2T as the reference temperature. Here we choose the static temperature T instead of M^2T as the reference temperature so that our system can also be applied to those problems with heat addition where the temperature change might be as large as the static temperature.

For the low Mach number flow, the results in figures 6.4 and 6.5 show that, for the explicit solution, the reference variable based on the static pressure, $Q_{pref} = (p, U, U, T)$, should be used in the non-preconditioned Γ_2 case while the reference variable based on

the dynamic pressure, $Q_{pref} = \left(1/2\,rU^2, U, U, T\right)$, should be used in the preconditioning Γ_2 case. This confirms that our prediction of the normalization variable in subsection 6.2.3 is appropriate. For high Mach number flow, either dynamic pressure or static pressure can be used as reference pressure because they are of the same order of magnitude.

To analyze how the perturbation parameter e' affects the solution, we consider three perturbations ($e' = 1$, 0.1 and 0.01, or $e = 7\times10^{-5}$, 7×10^{-6} and 7×10^{-7}) to check the influence of the perturbation on $\Delta Q_p{}^{im} / Q_{pref}$ and $\Delta Q_p{}^{ex} / Q_{pref}$. The Mach number is 0.01 and Γ_2 is non-preconditioned. The results are presented in figure 6.6. In this figure,

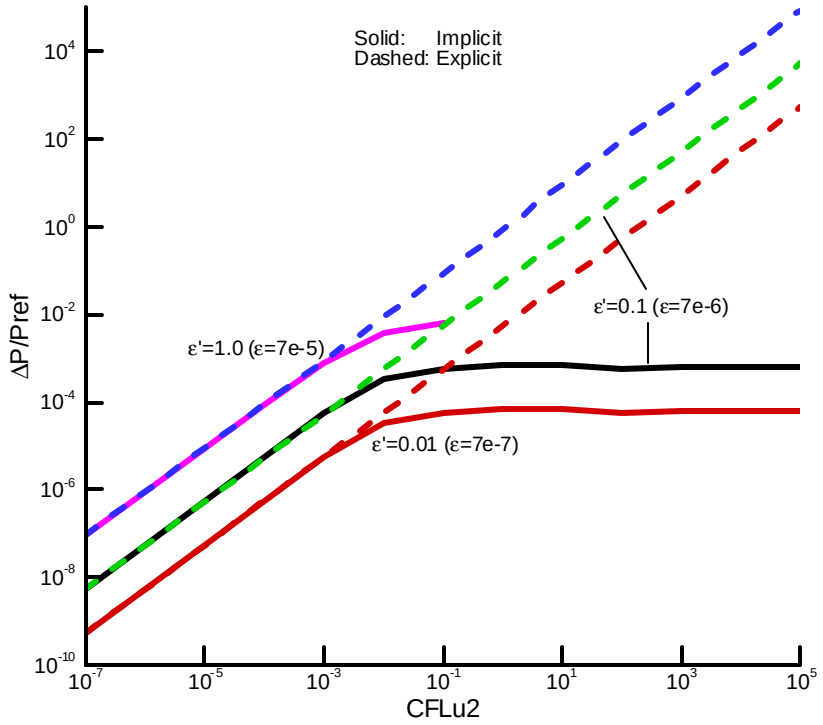

Figure 6.6 The effect of perturbation on the solutions of explicit pressure and implicit pressure. Uniform flow, Mach number of 0.01, I/II/II, non-preconditioned Γ_2, UPS.

for $e' = 1$ ($e = 7 \times 10^{-5}$), the results for middle and large values of CFL_{u2} diverge and consequently only the results for small values of CFL_{u2} are plotted. We can see that the perturbation only moves the results up or down and does not change the shape. As the perturbation increases, the error $\Delta p / p_{ref}$ also increases. Again the implicit solution is always less than or equal to the explicit solution. The results for the other three components are similar and hence are not shown.

The results for the case of preconditioning Γ_2 and Mach number of 0.01, and the case of Mach number of 0.7 are similar to the case of non-preconditioned Γ_2 of 0.01 Mach number. Accordingly, they are not shown.

6.3 Computational Results

Having chosen the reference variable, we run the same straight duct uniform flow in chapter 5 with the error-limited ramping method. A series of error tolerances $a = 0.01$, 0.1, 0.5, 1, 10 and 100 and perturbation parameters are tested. Also, three Mach numbers, M=0.5, 0.1 and 0.01, are considered. Since the error-limited ramping method only changes the outer convergence, only the number of outer iterations is presented.

We first look at the outer convergence for one representative case, a high Mach number of 0.5 with an allowable solution change of 50%, $a = 0.5$. The outer convergence and the corresponding maximum CFL_{u2}, the minimum CFL_{u2} and the average CFL_{u2} are shown in one plot, figure 6.7. All three values of CFL_{u2} increase monotonically (except for some wiggles) with the number of outer iterations as the residual decreases. The increase of CFL_{u2} is approximately exponentially linear with

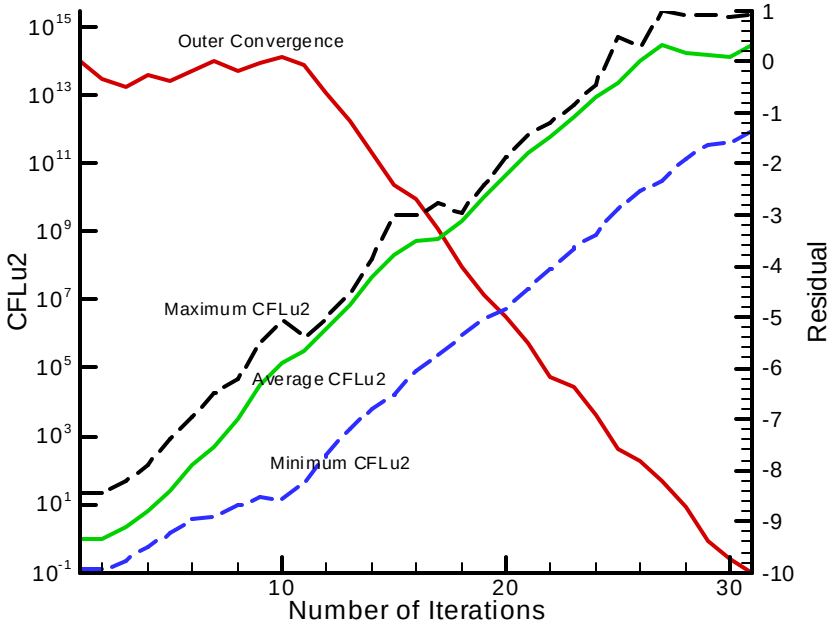

Figure 6.7 The outer convergence and CFL_{u2} value of triple time method with error-limited time step ramping for uniform straight duct flow, Mach number of 0.5, $a = 0.5$, e'=5.7 (e=1), UPS, I/II/II order.

the number of iterations. Throughout the computation process, the maximum CFL_{u2} is from two to five order of magnitude larger than the minimum CFL_{u2}. For the first outer iteration, the mininum CFL_{u2} is only 0.1 while the maximum one is as large as about 20. As convergence is approached, the minimum CFL_{u2} is increased to 10^{15} while the minimum CFL_{u2} is increased to 10^{11}.

The outer convergence takes 31 iterations to converge about ten orders of magnitude from one to 10^{-10}. The convergence is slow at the beginning because of the non-linearity of the problem and the small value of minimum CFL_{u2} which dominates the whole convergence in spite of the large values of the maximum CFL_{u2} and the average CFL_{u2}.

After about ten iterations, the non-linear effects in the outer convergence diminish and the solution converges rapidly.

The corresponding outer convergence of the triple time scheme with constant CFL_{u2} (without any ramping) is shown in figure 6.8 for a Mach number of 0.5. The CFL_{u2} value here is chosen as one, the largest value for which the initial condition with e'=5.7 (e =1) will converge as shown in figure 5.3. For the constant CFL_{u2} case, the number of iterations in the non-linear part is about 500 and that of the linear part is about 1000. Comparing the convergence of the triple time scheme with error-limited ramping in figure 6.7 with that of the triple time scheme with fixed $CFL_{u2}=1$ in figure 6.8, we can see that the convergence is improved by nearly a factor of 50. Clearly, time-step ramping preserves robustness while improving efficiency in this transonic Mach number case.

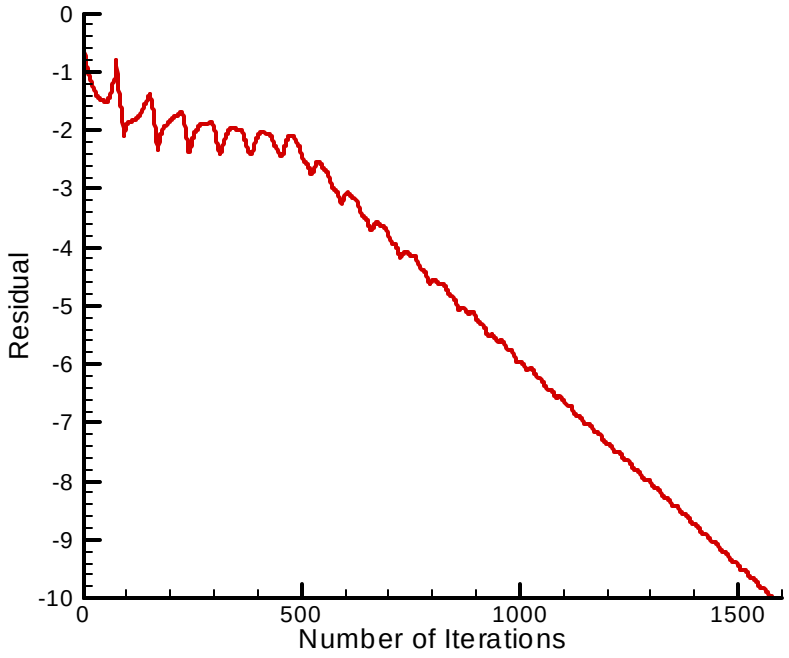

Figure 6.8 The outer convergence of triple time method with constant $CFL_{u2}=1$ for uniform straight duct flow, Mach number of 0.5, e'=5.7 (e =1), UPS, I/II/II order.

A second representative case is the low Mach number of 0.01 and $a = 0.5$ which is shown in figure 6.9. The results are plotted in the same way as the high Mach number, 0.5, case in figure 6.7. The whole convergence and CFL_{u2} results in figure 6.9 are similar to the results in the linear part of the high Mach number case in figure 6.7. The numbers of outer iterations for Mach number of 0.5 are summarized in Table 6.1. The diverged cases are indicated by the symbol, 'Div'. The first row in Table 6.1, for e' =0.057(e =0.01) corresponds to a perturbation that is small enough that the convergence process is almost completely linear. Accordingly as a increases, the number of iterations decreases because larger a allows the use of larger values of

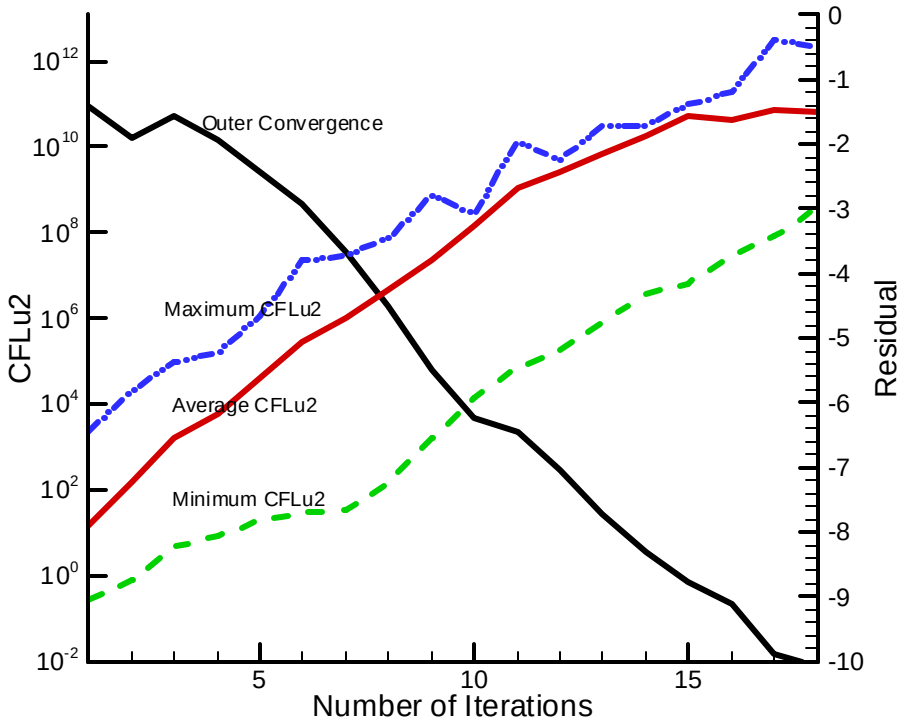

Figure 6.9 The outer convergence and CFL_{u2} values of triple time method with error-limited time step ramping for uniform straight duct flow, Mach number of 0.01, $a = 0.5$, e'=5.7 ($e = 4e^{-4}$),, UPS, I/II/II order.

Table 6.1 Mach number of 0.5, the number of outer iterations of the triple time method with the error-limited ramping for uniform straight duct flow, I/II/II order.

		$a = 0.01$	**0.1**	**0.5**	**1**	**10**	**100**
$e' =$ **0.057**	$e =$ **0.01**	110	32	26	26	24	24
0.57	**0.1**	109	32	28	27	26	27
2.85	**0.5**	130	36	30	29	Div	Div
5.7	**1**	185	43	31	30	Div	Div
28.5	**5**	Div	Div	Div	Div	Div	Div

CFL_{u2}, but the number of iterations does not change after $a \geq 0.5$ because increasing CFL_{u2} does not improve the convergence after some large value. The problem continues to converge even for $a = 100$. It is reasonable to conclude that the case of e'=0.057(e=0.01) converges for an infinite value of a or infinite value of CFL_{u2}, which agrees with the results in figure 5.6.

As e' increases, the problem becomes more difficult, and the number of iterations increases for a fixed value of a. In addition, the maximum value of a for converged cases decreases slightly until e' gets larger than 5.7 at which point the problem becomes divergent for all values of a.

To understand how the error-limiting parameter a affects the initial CFL_{u2} value, we look at the e'=5.7 (e=1) case and compute the minimum, maximum and average values of CFL_{u2} for every first outer iteration for $a = 0.01$, 0.1, 0.5, 1.0, 10 and 100.

The results are plotted in figure 6.10. For each vertical line, the symbols from the top to the bottom indicate the maximum, average and minimum values of CFL_{u2} respectively. The maximum allowable value of CFL_{u2} in the fixed CFL_{u2} case can be obtained from figure 5.6 which shows it is unity. This maximum allowable value of CFL_{u2} of unity is drawn in figure 6.10 as a horizontal line. The converged cases are

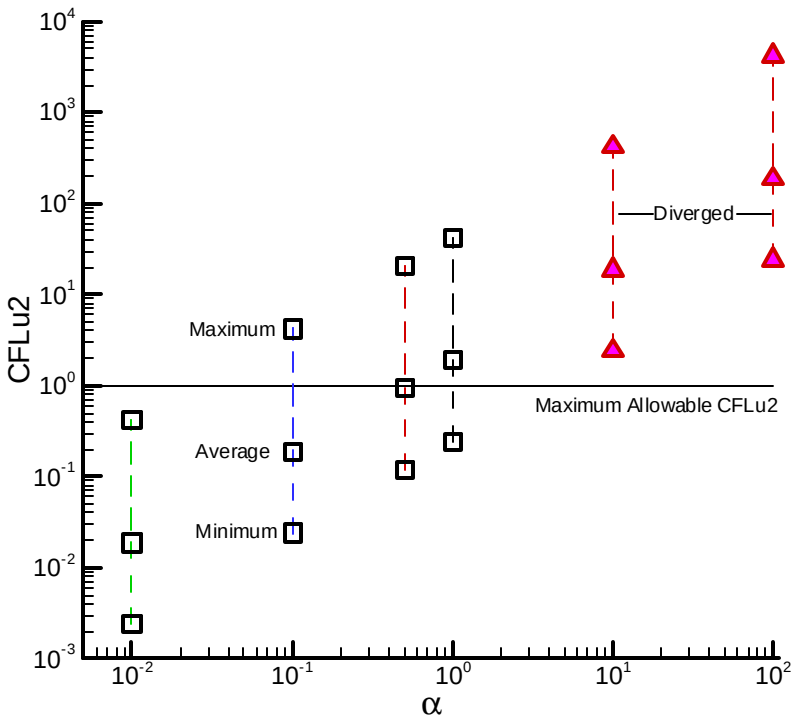

Figure 6.10 The variation of CFL_{u2} with the error-limited parameter a. Mach number of 0.5, e'=5.7 (e =1), UPS, I/II/II order.

represented by blank square symbols while the diverged cases are represented by filled triangle symbols.

Figure 6.10 shows that the three CFL_{u2} values vary exponentially with the error-limiting parameter a. For a = 0.01, all three CFL_{u2} values are less than the maximum allowable CFL_{u2}, one. Obviously, this case converges. When a is increased from 0.1 to 1.0, the maximum CFL_{u2} becomes larger than the constant CFL_{u2} limit while the minimum CFL_{u2} remains below one. From Table 6.1, these three cases also converge. As a is increased to 10 and 100, even the minimum CFL_{u2} is larger than the maximum allowable CFL_{u2}, one. Consequently, these two cases diverge as shown in

Table 6.1.

The suggestion from this one example is that only the minimum value of CFL_{u2} needs to be less than the constant CFL_{u2} limit and it is the point with the strongest non-linearity (minimum CFL_{u2}) that dominates the robustness and convergence in the computational domain. It is not efficient to use the same minimum CFL_{u2} in every point of the computational domain and throughout the entire computation as we did in Chapter Five. By using an error-limiter to automatically find an 'optimum' CFL_{u2} for each point of the computational domain in every outer iteration, we dramatically improve the convergence.

The numbers of outer iterations for Mach number of 0.1 and 0.01 are summarized in Tables 6.2 and 6.3 respectively in the same way as in Table 6.1.

The results in Table 6.2 only include perturbations of more than 100%, and so represent a very severe test of the ramping method. For a perturbation of order one, the maximum allowable tolerance, a can be as high as 100. For very large initial perturbations (e'=14.3 and 143), convergence can be obtained by reducing the allowable change in the solution to a =0.1 and 0.01. Thus the ramping method provides a dramatic improvement in robustness for these extremely large initial perturbations.

Table 6.3 is for Mach number of 0.01. Again, in this table, we only include perturbations of more than 100%. For $e' = 1.43$, the iterations converge for a as high as 10. As a increases, the number of iterations decreases first and then is fixed because the solution change is already very large for $a = 0.5$ and any further increase of a does not help the convergence anymore. The results for $e' = 7.15$ and 14.3 are similar to that for $e' = 1.43$. Surprisingly, even for $e' = 71.5$ and 143, the iteration still converges if $a = 0.1$.

In order to compare the convergence and robustness between the triple time method

Table 6.2 Mach number of 0.1, the number of outer iterations of the triple time method with the error-limited ramping for uniform straight duct flow, I/II/II order.

e'	e	$a = 0.01$	**0.1**	**0.5**	**1**	**10**	**100**
1.43	**0.01**	99	28	24	24	24	24
14.3	**0.1**	391	59	Div	Div	Div	Div
71.5	**0.5**	919	Div	Div	Div	Div	Div
143	**1**	1544	Div	Div	Div	Div	Div
1430	**10**	Div	Div	Div	Div	Div	Div

Table 6.3 Mach number of 0.01, the number of outer iterations of the triple time method with the error-limited ramping for uniform straight duct flow, I/II/II order.

e'	e	$a = 0.01$	**0.1**	**0.5**	**1**	**10**	**100**
1.43	**0.0001**	128	25	21	20	20	Div
7.15	**0.0005**	260	103	24	23	Div	Div
14.3	**0.001**	409	301	27	25	Div	Div
71.5	**0.005**	Div	270	Div	Div	Div	Div
143	**0.01**	Div	193	Div	Div	Div	Div
1430	**0.1**	Div	Div	Div	Div	Div	Div

with ramping and the triple time method without ramping, for each perturbation parameter e', we plot the number of outer iteration for these two cases in figures 6.11 and 6.12. The number of outer iterations without ramping is the optimum one from the computations of the straight duct uniform flow in chapter 5. The number of outer iterations with ramping is the minimum number for each e' from Tables 6.1, 6.2 and 6.3. The results for Mach numbers of 0.5 are plotted in figure 6.11 while that for Mach numbers of 0.1 and 0.01 are plotted in figure 6.12. The dashed lines and the triangle symbols are for the cases without ramping, and the solid lines and the square symbols are for the cases with ramping.

In figure 6.11 for Mach number of 0.5, both ramping and non-ramping cases converge for all values of e'. The convergence rate of the ramping case is very close to that of the non-ramping case at small perturbations, but is one hundred times faster than the non-ramping case for large perturbation of $e' = 5.7$.

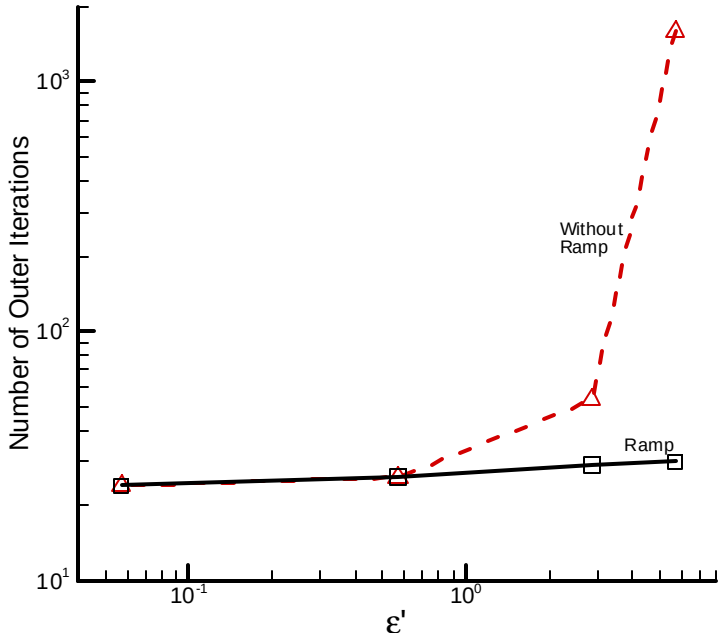

Figure 6.11 The comparison of the number of outer iterations between the triple time method without error-limited ramping and that with error-limited ramping for the uniform straight duct flow, Ma= 0.5.

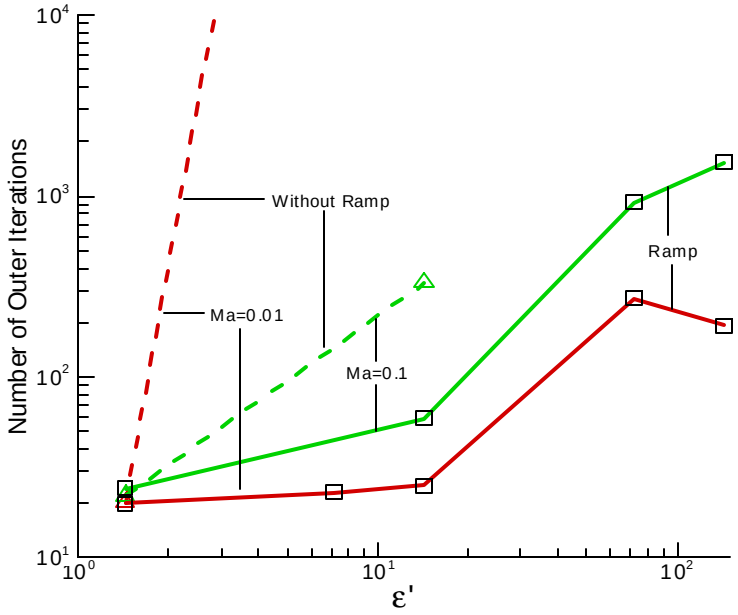

Figure 6.12 The comparison of the number of outer iterations between the triple time method without error-limited ramping and that with error-limited ramping for the uniform straight duct flow, Ma=0.1 and 0.5.

Figure 6.12 is for Mach numbers of 0.1 and 0.01. For Mach number of 0.1, the case with ramping converges for a perturbation e' as high as 143 while the case without ramping converges only for e' up to 14.3. Again for a perturbation of $e' = 1.43$, the number of outer iterations for the ramping and non-ramping cases are very close. For a large perturbation of $e' = 14.3$, the ramping method converges about five times faster than the non-ramping method. The case of Mach number equals to 0.01, is similar to the case of Mach number of 0.1.

In summary, in figures 6.11 and 6.12, the ramping method shows a good improvement in convergence for large perturbation in high Mach number flow. For low Mach number flow, the results show improvement in both convergence and robustness.

6.4 Conclusion

In this chapter, we discussed the error-limited time step ramping method. We perform some analytical analysis about this issue and show that the implicit solution ΔQ_p^{im} is always less than or equal to the explicit solution ΔQ_p^{ex}, which indicates that we can control the implicit solution change by controlling the explicit solution change. The normalization issue is discussed to provide a good normalization for the explicit solution ΔQ_p^{ex} and implicit solution ΔQ_p^{im}.

Some computational results for low and high Mach number uniform flows for the error-limited ramping method are presented and compared with the case without the error-limited ramping method. For high Mach number flow, the results show a good improvement in convergence for large perturbations. For low Mach number flows, the results show improvement in both convergence and robustness.

Chapter 7

Conclusion

7.1 Summary

The work in this thesis is dedicated at assessing the robustness and convergence of CFD algorithms. Two concepts are developed to achieve this goal, the triple time scheme and the error-limited CFL ramping method.

First, we develop the general triple time scheme by introducing three time-marching steps, each of them are designed for different purposes and relatively independent of each other. In the first step, we introduce the first time marching to define the artificial dissipation. In the second step, we introduce the second time marching to solve the non-linear discretized equation obtained in the first step. In the third step, we introduce the third time marching to solve the linear equation obtained in the second step. Then, a diagonally dominant line Gauss-Sediel approximate factorization method is introduced to solve the triple time equation.

Second, the preconditioning for the three Jacobian matrices introduced in three time marching is studied to satisfy their individual requirements by stability analyses. These three Jacobian matrices are chosen from the physical, the steady preconditioning and the unsteady preconditioning Jacobian matrices. After some analysis, we obtain four combinations of the three preconditioning schemes, 'UPS', 'UPU', 'UUU' and 'UUS'. Three levels of stability analysis, the inner t_3 stability, the outer t_2 stability with enough inner iterations (direct inversion), and the outer t_2 stability with a finite number of inner iterations, are studied to pick the optimum one from these four systems. According to the stability analysis, there is no major difference between these four

systems, but the 'UPU' system has a marginal advantage over the other three. Consequently, we use the 'UPU' system.

After choosing the preconditioning of the system, we perform the stability analysis for this system. An optimum value of CFL (CFL_{u3}=20) for the inner convergence is obtained from the inner stability analysis and the optimum number of inner iterations is obtained from the outer stability with a finite number of inner iterations. The optimum number of inner iterations is dependent on the CFL of the outer iteration. Since our unsteady preconditioning depends on the number of grids, we studied the effect of the grid number on the triple time scheme. The results show that the grid number has little effect on the optimum value of CFL for the inner iteration (CFL_{u3}), but slows down the inner convergence greatly and can even make the outer iteration diverge for a small number of inner iterations at $CFL_{u2} = 10$ (figure 3.28). At last, the stability comparison between the triple time and single time is performed and the results show a factor of almost 2.7 improvement of the triple time method over the single time method in CPU time for low Mach number of 0.01. All these stability results agree well with our computational results for a linear problem of straight duct uniform flow.

Third, since the triple time method requires more storage to save CPU time by storing the matrix inversion in solving the linear system, an estimation of the CPU time saving for the linear time and non-linear time, and the storage cost for the triple time and the single time are analyzed. The analysis shows that the operation count of the linear iteration is proportional to the square of the number of equations while the operation count of the non-linear iteration is proportional to the cube of the number of equation. Accordingly, the operation count ratio of the linear iteration to the non-linear iteration is proportional to the number of equations. For a four equation system, the operation count of the linear iteration is about 3.5 faster than that of the non-linear iteration. The storage

comparison shows that the storage for the triple time method costs about four times more than that of the single time method.

Fourth, some computational results are presented to support the improvement of the triple time method over the single time method in convergence and robustness. For a problem with very small perturbation, our computational results show that the triple time method is about two times faster than the single time method. To test the robustness in the non-linear convergence portion, we consider two problems with large perturbations for fixed values of CFL in outer iteration, a straight duct uniform flow and a nozzle flow whose non-linearity is controlled by a perturbation parameter. Our results show a good improvement in robustness in both the magnitude of the perturbation and CFL, but the convergence is very slow because we use a small CFL in the outer iteration as required by the non-linearity of the problem.

At last, to improve the convergence for the non-linear problems, an error-limited ramping method is introduced. The error-limited ramping method ramps the CFL by enforcing the change of the local solution computed by an explicit method less than a specified tolerance. Consequently, we use a local CFL instead of a global one. We begin this study by proving that the implicit solution is always less or equal to the explicit solution so that the implicit solution is less or equal to the specified tolerance as long as the explicit solution is less or equal to the specified tolerance. Then we discuss the normalization issue for the explicit solution. The normalization reference variable (local or global variable) should be picked very carefully to avoid non-physical results. Our analysis shows that, for low Mach number flow, the pressure component should be normalized by the static pressure if Γ_2 is non-preconditioned while the pressure component should be normalized by dynamic pressure if Γ_2 is steady preconditioned. Some computational results are presented to show that our analysis is correct.

Our study shows that time step ramping changes the residual during the iteration

process if unsteady preconditioning is used in the artificial dissipation. This will lead to difficulties in computing the time step from the explicit solution. These difficulties are analyzed briefly. So far we have not found a good way to solve for the time step from the explicit solution. Using steady preconditioning for the artificial dissipation avoids this difficulty so that the time step is easily found.

Some computational results for the ramping method are presented for low and high Mach number flows in a straight duct. For high Mach number flow, the results show a good improvement in convergence for large perturbations. For low Mach number flow, the results show improvement in both convergence and robustness.

7.2 Future Research

The robustness comparison of 'UPS' and 'UPU' systems show that 'UPU' is better than 'UPS' suggesting that artificial dissipation is very important to the robustness. Accordingly, a further systematic study of the effect of artificial dissipation on robustness is highly desirable.

Regarding the error-limited ramping method, more applications should be run to check the validity of this method and provide suggestions for how to pick the reference variable and tolerance value.

References

References

1. Allmaras S., Analysis of semi-implicit preconditioners for multigrid solution of the 2-D compressible Navier-Stokes equations. AIAA Paper 95-1651-CP, 12th Computational Fluid Dynamics Conference, San Diego, CA, 1995.

2. Brandt, A., Multigrid techniques: 1984 guide with applications to fluid dynamics, VKI Lecture Series 1984-04

3. Briley, W. R., McDonald, H. and Shamroth, S.J., A low Mach number Euler Formulation and Application to Time-Iterative LBI Schemes, AIAA Journal 21(24): 1467-1469, 1983.

4. Briley, W. R., Govindan, T. R. and McDonald, H., Efficient Navier-Stokes flow prediction algorithms, Contractor Report, NAS8-37340

5. Briley, W. R. and McDonald, H., Solution of the multidimensional compressible Navier-Stokes equations by a generalized implicit method, Journal of Computational Physics, 24:372-397

6. Buelow, P. E. O., Venkateswaran, S., and Merkle, C.L., The effect of grid aspect ratio on convergence, AIAA Journal, 32: 2401-2408, 1994.

7. Buelow, P .E. O., Venkateswaran, S., and Merkle, C.L., Grid aspect ratio effects on the convergence of upwind schemes, AIAA Paper 95-0565, 33rd Aerospace Sciences Meeting,

Reno, NV, 1995.

8. Buelow, P. E. O., Venkateswaran, S. and Merkle C. L., Stability and convergence analysis of implicit upwind schemes, Computers and Fluids, 30: 961-988, 2001.

9. Buelow, P. E. O., Convergence enhancement of Euler and Navier-Stokes algorithms, Ph.D. thesis , The Pennsylvania State University, 1995

10. Chakravarthy, S., Relaxation methods for unfactored implicit upwind schemes, AIAA paper 84-0165

11. Choi, D. and Merkle, C. L., Application of time-iterative schemes to incompressible flow, AIAA Journal, 23: 1518-1524, 1985.

12. Choi, Y. H. and Merkle, C. L., The application of preconditioning to viscous flows, Journal of Computational Physics, 105: 207-223, 1993.

13. Chorin, A. J., A numerical method for solving incompressible viscous flow problems, Journal of computational physics, 2: 12-26, 1967.

14. Darmofal, D. and Schmid, P., The importance of eigenvectors for local preconditioners of the Euler equations, Journal of Computational Physics, 127: 346-362, 1996.

15. Darmofal, D. and Siu, K., A robust locally preconditioned multigrid algorithm for the Euler equations, AIAA Paper 98-2428, 1998.

16. D. Darmofal and B. van Leer, Local preconditioning: Manipulating Mother Nature to fool Father Time, in Computing the Future II: Advances and Prospects in Computational Aerodynamics, M. Hafez and D. Caughey, eds., John Wiley and Sons, 1998

17. Douglas, J. and Gunn, J. E., A general formulation of alternating direction method – part I – parabolic and hyperbolic problem, Numerische Mathematik, 82: 428-453

18. Feng, J. and Merkle, C. L., Evaluation of preconditioning methods for time marching systems, AIAA Paper 90-0016, AIAA 28th Aerospace Sciences Meeting, Reno, NV

19. Guerra, J. and Gustafsson, B., A numerical method for incompressible and compressible flow problems with smooth solutions, Journal of Computational Physics, 63: 377, 1986

20. Hirsch C., Numerical computation of internal and external flows, Vols. 1 and 2, John Wiley and Sons, 1990.

21. Issa, R. I., Solution of the implicitly discretized fluid flow equations by operator-splitting, Journal of Computational physics, 62: 40-65, 1986

22. Jameson, A., Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method, Proceedings of the AIAA 4th CFD Conference pp. 122-146

23. Jameson, A. and Turkel, E., Implicit schemes and LU decompositions, Mathematics of Computation, 37(156): 385-397

24. Li, D., Venkateswaran, S., Fakhari, K. and Merkle C. L., Convergence assessment of general fluid equations on unstructured hybrid grids, AIAA Paper 2001-2557

25. Li, D., Venkateswaran, Lindau J. W. and Merkle C. L. computational Formulation for Multi-Component and Multi-Phase Flows, To be presented at the AIAA Aerospace Sceiences meeting and Exhibit, Reno, NV, 2005.

26. Mavriplis, D., Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes, AIAA Paper 97-1952, 1997.

27. Merkle C. L. and Yu S., Class notes, not published.

28 Merkle, C. L., Venkateswaran, S. and Buelow, P. E. O., The relationship between pressure-based and density-based algorithms," AIAA 92-0425, 30th Aerospace Sciences Meeting and Exhibit, Jan 6-9, 1992, Reno, NV.

29 Mulder, W. A., A high-resolution Euler solver based on multigrid, semi-coarsening and defect correction. Journal of Computational Physics, 100: 91-104, 1992.

30. Patankar, S. V., Numerical heat transfer and fluid flow, Series in Compuational Methods in Mechanics and Thermal Sciences, McGraw-Hill Book Company.

31. Patankar, S.V. and Spalding, D.B., ?A calculation procedure for heat, mass and momentum transfer, Intl. Journal of Heat and Mass Transfer, 15: 1787-1806, 1972.

32. Pierce, N. A. and Giles, M. B., Preconditioned compressible flow calculations in stretched meshes, AIAA Paper 96-0889, 1996.

33. Rehm, R. G. and Baum, H. R., Journal of Research of the National Bureau of Standards, 83: 297, 1978.

34. Roe P. L., Approximate Riemann solvers, Parameter Vectors, and Difference Schemes, Journal of Computational Physics, 43: 357-372, 1981.

35. Saad, Y. and Schultz, M. H., GMRES: a geralized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7(3): 856-869

36. Shuen, J. S., Chen, K. H. and Choi, Y. H., A time-accurate algorithm for chemical non-equilibrium viscous flows at all speeds, AIAA Paper 92-3639, 1992.

37. Turkel, E., Fiterman, A. and van Leer, B., Preconditioning and the limit to the incompressible flow equations, Computing the Future: Frontiers of Computational Fluid Dynamics, eds. Caughey, D. A. and Hafez, M. M., Wiley Publishing, pp. 215-234, 1994.

38. Turkel, E, Preconditioning methods for solving the incompressible and low speed compressible equations, Journal of Computation Physics, 72: 277-298, 1987.

39. Van Leer, B., Lee, W. T. and Roe, P. L., Characteristic Time-Stepping or Local Preconditioning of the Euler Equations, AIAA Paper 91-1552-CP, Computational Fluid Dynamics Conference, Honolulu, 1991.

40. Venkateswaran S., Lindau J., Kunz R. and Merkle C. L., Computation of multiphase mixture flows with compressibility effects," Journal of Computational Physics, 180: 54-77, 2002.

41. Venkateswaran S. and Merkle C. L., Dual time-stepping and preconditioning for unsteady computations, AIAA 95-0078, 33rd Aerospace Science Sciences Meeting and Exhibit, Reno, NV, 1995.

42. Venkateswaran S. and Merkle C. L., Analysis of preconditioning methods for the Euler and Navier-Stokes equations, Von Karman Institute Lecture Series, 1999.

43. Venkateswaran S. and Merkle C. L., Evaluation of artificial dissipation models and their relationship to the accuracy of the Euler and Navier-Stokes Computations, 16th International Conference on Numerical Methods in Fluid Dynamics, Arcachon, France , July 6-10, 1998.

44. Venkateswaran, S., and Merkle, C. L., "Artificial Dissipation Control for Viscous and Unsteady Computations," AIAA Paper No. 2003-3695, 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, June 23-26, 2003.

45. Venkateswaran S. Deshpande, M.and Merkle C. L., Application of preconditioning to reacting flow computations, AIAA Paper 95-1673, June, 1995.

46. Venkateswaran, S., Li, D., and Merkle, C. L., Influence of stagnation regions on preconditioned solutions at low speeds, AIAA Paper 2003-435, 41st Aerospace Sciences

Meeting and Exhibit, Reno, NV, Jan. 2003.

47. Venkateswaran, S., Zeng X., Li, D. and Charles L. Merkle, "Influence of Large-Scale Pressure Changes on Pre-Conditioned Solutions at Low Mach Number," AIAA 2002-2957, 32nd Fluid Dynamics Conference, St. Louis MO, 24-26 June, 2002

48. Venkateswaran S., Deshpande M. and Merkle C. L., computation of turbulent reacting flow fields using the preconditioned Navier-Stokes equations, CFD Journal, 8 (2): 1978-1985, 1999.

49. Venkateswaran, S., Tamamidis, P. and Merkle, C. L., Interpretation of pressure-based methods as time-marching schemes, Fifteenth International Conference on Numerical Methods in Fluid Dynamics, June 24-28, 1996, Monterey, CA.

50. Viviand, H., Pseudo-unsteady systems for steady inviscid calculations, Numerical Methods for the Euler Equations of Fluid Dynamics, SIAM Journal on Numerical Analysis, pp. 334, 1985.

51. Warming, R. F. and Beam, R. M., On the construction and application of implicit factored schemes for conservation law, SIAM-AMS Proceedings, 11: 85-129

52. Weiss, J. M., Maruszewski, J. P. and Smith, W. A., Implicit solution of the preconditioned Navier-Stokes equations using algebraic multigrid, AIAA Journal, 37: 29, 1999

53. Wigton, L. B., Yu, N. J. and Young, D. P., GMRES acceleration of computational

fluid dynamics codes, AIAA Paper 85-1494.

54. Withington, J. P., Shuen, J. S. and Yang, V., A time-accurate implicit method for chemically reacting flows at all Mach numbers, AIAA Paper 91-0581, 1991.

55. Zeng, X. Q., Venkateswaran, S., Li, D. and Merkle, C. L., AIAA Paper No. 2003-3707, 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, June 23-26, 2003.

56. Tapia A. R., Lanius C., Zeal C. M. M and Parks, T. A., Computational science: tools for a changing world, Website: http://ceee.rice.edu/Books/CS/.

Appendices

Appendix A

MHD Eigen System for Arbitrary Flow Direction

This appendix presents the eigen system of three-dimensional, viscous, resistive magnetohydrodynamic (MHD) plasma model for an arbitrary flow direction. The eigen system in the Cartesian coordinate for the flow along x, y and z direction has already been developed by Powell (references), where $(r, Y, u, v, w, B_x, B_y, B_z, P)$ is chosen to be the primary variable. The eigen system for an arbitrary flow direction is very desirable in many computations, especially the unstructured grid computations, but is not available so far. In this appendix, the eigen system for the arbitrary flow direction is developed on the base of the Powell's eigen system. Also, the eigen system developed here uses the primary variable of $(p, Y, u, v, w, T, B_x, B_y, B_z)$ instead of the primary variable of $(r, Y, u, v, w, B_x, B_y, B_z, P)$.

The three-dimensional, viscous, magnetohydrodynamic (MHD) plasma governing equations include conservation of mass, species mass fraction, momentum, energy and the magnetic induction equation. These conservation equations, complemented by Ohm's and Ampere's laws, are the following,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{A.1}$$

$$\frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \vec{v}) = \nabla \cdot \rho D \nabla Y \tag{A.2}$$

$$\frac{\partial r\vec{\mathrm{v}}}{\partial t}+\nabla\cdot(r\vec{v}\vec{v}+pI)=\nabla\cdot\vec{t}+\vec{\mathrm{J}}\times\vec{\mathrm{B}} \tag{A.3}$$

$$\frac{\partial(rh_0-p)}{\partial t}+\nabla\cdot(rh^0\vec{v})=-\nabla\cdot\vec{\mathrm{q}}+\nabla\cdot(\vec{t}\cdot\vec{\mathrm{v}})+\vec{J}\cdot\vec{\mathrm{E}} \tag{A.4}$$

$$\frac{\partial\vec{B}}{\partial t}+\nabla\times\vec{E}+\vec{v}\left(\nabla\cdot\vec{B}\right)=0 \tag{A.5}$$

$$\vec{E}=-\vec{v}\times\vec{B}+\frac{1}{s}\vec{J}+\frac{1}{ne}\vec{J}\times\vec{B} \tag{A.6}$$

$$\vec{J}=\frac{1}{m_0}\nabla\times\vec{B} \tag{A.7}$$

Where r , p , $\vec{v}(u,v,w)$, Y, h^0, e and n are density, pressure, velocity, species mass fraction, stagnation enthalpy, total energy and number of electrons respectively. The quantity, I is a three by three unit matrix. The vectors, $\vec{J}=\left(J_x,J_y,J_z\right)$ and $\vec{B}=\left(B_x,B_y,B_z\right)$ are the electric current density and the magnetic flux density respectively, while $\vec{J}\times\vec{B}$ represents the Lorentz force and $\vec{J}\cdot\vec{E}$ is the electric power dissipation where $E=\left(E_x,E_y,E_z\right)$ is the electric field. The tensor, $\vec{t}$ is the shear stress and the heat flux vector $\vec{q}$ is,

$$\vec{q}=-l\nabla T+rDh\nabla Y$$

where l , D and h are the heat conduction coefficient, the mass diffusivity coefficient and the enthalpy respectively.

The equations A.1 through A.7 are closed by adding the state equation, enthalpy relation, the state-dependent electrical conductivity, electron number density and transport properties as the following:

$$r=r(p,T,Y),\quad h=h(p,T,Y),\quad s=s(p,T,Y),$$

$$n = n(p,T,Y), \quad m = m(p,T,Y), \quad k = k(p,T,Y)$$

The conservative equations A.1 through A.7 can be expressed as a coupled vector system (references)

$$\frac{\partial Q}{\partial t} + \nabla \cdot \left(\vec{F} - \vec{F}_v\right) + A_{pnc} \nabla \cdot Q_p = 0 \tag{A.8a}$$

The conservative time derivative in the last equation can be written in term of the non-conservative time derivative of the primary variable, then we get

$$\Gamma_p \frac{\partial Q_p}{\partial t} + \nabla \cdot \left(\vec{F} - \vec{F}_v\right) + A_{pnc} \nabla \cdot Q_p = 0 \tag{A.8b}$$

In equations A.8a and A.8b, the conservative and primary variables, Q and Q_p are

$$Q = \begin{bmatrix} r \\ rY \\ r\vec{v} \\ rh_B^0 - p_B \\ \vec{B} \end{bmatrix}, \qquad Q_p = \begin{bmatrix} P \\ Y \\ \vec{v} \\ T \\ \vec{B} \end{bmatrix}$$

Where $h_B^0 = h^0 + \frac{\vec{B} \cdot \vec{B}}{r m_0}$ (m_0 is the viscosity coefficient) and $p_B = p + \frac{\vec{B} \cdot \vec{B}}{2 m_0}$

The Jacobian matrix Γ_p is

$$\Gamma_p = \frac{\partial Q}{\partial Q_p} = \begin{bmatrix} r_p & r_Y & 0 & r_T & 0 \\ r_p Y & r_Y Y + r & 0 & r_T Y & 0 \\ r_p \vec{v} & r_Y \vec{v} & rI & r_T \vec{v} & 0 \\ r_p h^0 + r h_p - 1 & r_Y h^0 + r h_Y & r\vec{v}^T & r_T h^0 + r h_T & \frac{1}{m_0}\vec{B}^T \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \tag{A.9}$$

where the superscript 'T' of the vector indicates the transpose of that vector (column vector).

The convection flux vector, $\vec{F}$ is

$$\vec{F} = \begin{bmatrix} r\vec{v} \\ r\vec{v}Y \\ r\vec{v}\vec{v} - \frac{1}{m_0}\vec{B}\vec{B} + p_B\vec{I} \\ r\vec{v}h_B^0 - \frac{1}{m_0}\left(\vec{v}\cdot\vec{B}\right)\vec{B} \\ \vec{v}\vec{B} - \vec{B}\vec{v} \end{bmatrix} \tag{A.10}$$

and the viscous flux vector, $\vec{F}_v$ is

$$\vec{F}_v = \begin{bmatrix} 0 \\ rDY \\ \vec{\vec{t}} \\ \left(\vec{\vec{t}}\cdot\vec{v}\right) - \vec{q} - \frac{1}{m_0{}^2}\left\{\frac{1}{s}\left[\left(\nabla\times\vec{B}\right) + \left(\nabla\times\vec{B}\right)\times\vec{B}\right]\times\vec{B}\right\} \\ \vec{\vec{p}} \end{bmatrix} \tag{A.11}$$

where $\vec{\vec{p}}$ is the resistivity tensor and

$$\nabla\cdot\vec{\vec{p}} = -\nabla\times\left(\frac{1}{s\,m_0}\nabla\times\vec{B}\right) - \nabla\times\left(\frac{1}{m_0 ne}\vec{J}\times\vec{B}\right)$$

and the notation, $\vec{v}\vec{B}$ is a matrix of dimension three as the following

$$\vec{v}\vec{B} = \begin{pmatrix} uB_x & uB_y & uB_z \\ vB_x & vB_y & vB_z \\ wB_x & wB_y & wB_z \end{pmatrix}$$

The other vector by vector notation can be expanded in a similar way.

The non-conservative Jacobian A_{pnc} (the subscript 'nc' indicates non-conservative) is given as

$$A_{pnc} = \frac{1}{m_0}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \vec{B} \\ 0 & 0 & 0 & 0 & \vec{v}\cdot\vec{B} \\ 0 & 0 & 0 & 0 & \vec{v}m_0 \end{pmatrix} \tag{A.12}$$

For a general direction $\vec{n} = (n_x, n_y, n_z)$, equation A.8b becomes

$$\Gamma_p \frac{\partial Q_p}{\partial t} + \nabla \cdot \left(\vec{F}_n - \vec{F}_{vn}\right) + A_{pncn} \nabla \cdot Q_p = 0 \tag{A.13}$$

where the subscript 'n' in F_n, F_{vn} and A_{pncn} indicates the direction, $\vec{n}$.

From equation A.10, the convection flux vector, $\vec{F}$ in direction $\vec{n}$, F_n is

$$F_n = \vec{F} \cdot \vec{n} = \begin{bmatrix} rU \\ rUY \\ rU\vec{v} - \frac{1}{m_0} B_n \vec{B} + p_B \vec{n} \\ rUh_B^0 - \frac{1}{m_0}\left(\vec{v} \cdot \vec{B}\right) B_n \\ U\vec{B} - B_n \vec{v} \end{bmatrix} \tag{A.14}$$

Where $B_n = \vec{B} \cdot \vec{n}$ and the quantity, U is the magnitude of the velocity $\vec{v}$.

and from equation A.11, the viscous flux vector, $\vec{F}_v$ in direction $\vec{n}$, F_{vn} is

$$\vec{F}_{vn} = \begin{bmatrix} 0 \\ (rDY) \cdot \vec{n} \\ \vec{t} \cdot \vec{n} \\ \left\{\left(\vec{t} \cdot \vec{v}\right) - \vec{q} - \frac{1}{{m_0}^2}\left\{\frac{1}{s}\left[\left(\nabla \times \vec{B}\right) + \left(\nabla \times \vec{B}\right) \times \vec{B}\right] \times \vec{B}\right\}\right\} \cdot \vec{n} \\ \vec{\vec{p}} \cdot \vec{n} \end{bmatrix}$$

and from equation A.12, the non-conservative Jacobi matrix, A_{pnc} in direction $\vec{n}$, A_{pncn} is

$$A_{pncn} = \frac{1}{m_0}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \vec{B}\vec{n} \\ 0 & 0 & 0 & 0 & \left(\vec{v}\cdot\vec{B}\right)\vec{n} \\ 0 & 0 & 0 & 0 & m_0\vec{v}\vec{n} \end{pmatrix} \tag{A.15}$$

We write the divergence term, $\nabla\cdot\vec{F}_n$ in equation A.13, as the Jacobian matrix A_{pn} multiplies the divergence of the primary variable, equation A.13 becomes

$$\Gamma_2\frac{\partial Q_p}{\partial t_2} + \left(A_{pn} + A_{pncn}\right)\nabla\cdot Q_p = \nabla\cdot\vec{F}_{vn} \tag{A.16a}$$

where the Jacobian matrix A_{pn} is (from equation A.14)

$$A_{pn} = \frac{\partial F_n}{\partial Q_p}$$

$$= \begin{bmatrix} r_p U & r_Y U & r\vec{n} & r_T U & 0 \\ r_p UY & (r_Y Y + r)U & rY\vec{n} & r_T UY & 0 \\ r_p U\vec{v} + \vec{n} & r_Y U\vec{v} & r\vec{v}\vec{n} + rU\vec{\vec{d}} & r_T U\vec{v} & \frac{1}{m_0}(\vec{n}\vec{B} - \vec{B}\vec{n} - B_n\vec{\vec{d}}) \\ (e_p + 1)U & e_Y U & r\vec{v}U + (e + p_B)\vec{n} - \frac{1}{m_0}B_n\vec{B} & e_T U & \frac{1}{m_0}\left[2\vec{B}U - \vec{v}B_n - \left(\vec{B}\cdot\vec{v}\right)\vec{n}\right] \\ 0 & 0 & \vec{B}\vec{n} - B_n\vec{\vec{d}} & 0 & U\vec{\vec{d}} - \vec{v}\vec{n} \end{bmatrix} \tag{A.17}$$

Where

$e = rh_B^0 - p_B$ (**need to check**)

Multiply equation A.16a by the inverse of matrix Γ_p to obtain,

$$\frac{\partial Q_p}{\partial t} + \Gamma_p^{-1}\left(A_{pn} + A_{pncn}\right)\nabla\cdot Q_p = \Gamma_p^{-1}\nabla\cdot\vec{F}_{vn} \tag{A.16b}$$

The characteristic of equation A.16b is controlled by the eigen system of the matrix $\Gamma_p^{-1}\left(A_{pn} + A_{pncn}\right)$.

Add equations A.15 and A.17 to obtain

$$A_{pn}+A_{pncn}=\begin{bmatrix} r_pU & r_yU & r\vec{n} & r_TU & 0 \\ r_pUY & (r_yY+r)U & rY\vec{n} & r_TUY & 0 \\ r_pUv+\vec{n} & r_yUv & r\vec{v}\vec{n}+rU\vec{\vec{d}} & r_TUv & \frac{1}{m_0}(\vec{n}\vec{B}-B^n\vec{\vec{d}}) \\ (e_p+1)U & e_yU & r\vec{v}U+(e+p_B)\vec{n}-\frac{1}{m_0}B^n\vec{B} & e_TU & \frac{1}{m_0}(2\vec{B}U-\vec{v}B^n) \\ 0 & 0 & \vec{B}\vec{n}-B^n\vec{\vec{d}} & 0 & U\vec{\vec{d}} \end{bmatrix}$$

We multiply this matrix by Γ_p^{-1} to obtain

$$\Gamma_p^{-1}(A_c+A_{nc})=\begin{pmatrix} U & 0 & \frac{r^2h_T}{\Delta}\vec{n} & 0 & \vec{0} \\ 0 & U & \vec{0} & 0 & \vec{0} \\ \frac{\vec{n}}{r} & 0 & U\vec{\vec{d}} & \vec{0} & \frac{\vec{n}\vec{B}-(\vec{B}\cdot\vec{n})\vec{\vec{d}}}{rm_0} \\ 0 & 0 & \frac{r(1-rh_p)}{\Delta}\vec{n} & U & \vec{0} \\ \vec{0} & \vec{0} & \vec{B}\vec{n}-(\vec{B}\cdot\vec{n})\vec{\vec{d}} & \vec{0} & U\vec{\vec{d}} \end{pmatrix} \qquad \text{(A.18)}$$

After a lot of algebraic operations, the matrix in equation A.18 can be diagonalized as

$$\Gamma_p^{-1}(A_c+A_{nc})=M\Lambda M^{-1}$$

Where

$$\Lambda=\begin{pmatrix} U & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & U & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & U & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & U+\frac{\vec{B}\cdot\vec{n}}{\sqrt{rm_0}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & U-\frac{\vec{B}\cdot\vec{n}}{\sqrt{rm_0}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & U+c_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & U-c_f & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & U+c_s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & U-c_s \end{pmatrix} \qquad \text{(A.19)}$$

$$M=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \frac{r^2 h_T}{\Delta} & \frac{r^2 h_T}{\Delta} & \frac{r^2 h_T}{\Delta} & \frac{r^2 h_T}{\Delta} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vec{0} & \vec{0} & \vec{0} & \vec{n}\times\vec{B} & \vec{n}\times\vec{B} & \vec{W}_f & -\vec{W}_f & \vec{W}_s & -\vec{W}_s \\ \frac{1}{r_T} & 0 & 0 & 0 & 0 & \frac{r(1-rh_p)}{\Delta} & \frac{r(1-rh_p)}{\Delta} & \frac{r(1-rh_p)}{\Delta} & \frac{r(1-rh_p)}{\Delta} \\ \vec{0} & \vec{0} & \vec{n} & -\sqrt{rm_0}\vec{n}\times\vec{B} & \sqrt{rm_0}\vec{n}\times\vec{B} & \vec{M}_f & \vec{M}_f & \vec{M}_s & \vec{M}_s \end{pmatrix} \tag{A.20}$$

$$M^{-1}=\begin{pmatrix} r_p-\frac{1}{a^2} & 0 & \vec{0} & r_T & \vec{0} \\ 0 & 1 & \vec{0} & 0 & \vec{0} \\ 0 & 0 & \vec{0} & 0 & \vec{n} \\ 0 & 0 & \frac{\vec{n}\times\vec{B}}{2(\vec{B}\cdot\vec{B}-B_n^2)} & 0 & -\frac{\vec{n}\times\vec{B}}{2(\vec{B}\cdot\vec{B}-B_n^2)\sqrt{rm_0}} \\ 0 & 0 & \frac{\vec{n}\times\vec{B}}{2(\vec{B}\cdot\vec{B}-B_n^2)} & 0 & \frac{\vec{n}\times\vec{B}}{2(\vec{B}\cdot\vec{B}-B_n^2)\sqrt{rm_0}} \\ \frac{1}{G_f} & 0 & \frac{r\vec{W}_f}{G_f} & 0 & \frac{\vec{M}_f}{G_f m_0} \\ \frac{1}{G_f} & 0 & -\frac{r\vec{W}_f}{G_f} & 0 & \frac{\vec{M}_f}{G_f m_0} \\ \frac{1}{G_s} & 0 & \frac{r\vec{W}_s}{G_s} & 0 & \frac{\vec{M}_s}{G_s m_0} \\ \frac{1}{G_s} & 0 & -\frac{r\vec{W}_s}{G_s} & 0 & \frac{\vec{M}_s}{G_s m_0} \end{pmatrix} \tag{A.21}$$

Where in these three matrices, a is the sound speed and some of the terms are defined as

$$c_{f,s}^2=\frac{1}{2}\left(a^{*2}\pm\sqrt{a^{*4}-\frac{4h_T(\vec{B}\cdot\vec{n})^2}{\Delta m_0}}\right) \tag{A.22}$$

$$\vec{W}_{f,s}=\frac{c_{f,s}(B_n^2\vec{n}-B_n\vec{B})}{rm_0c_{f,s}^2-B_n^2}+c_{f,s}\vec{n} \tag{A.23}$$

$$\vec{M}_{f,s}=\frac{rm_0c_{f,s}^2(\vec{B}-B_n\vec{n})}{rm_0c_{f,s}^2-B_n^2} \tag{A.24}$$

$$G_{f,s}=2\left[\frac{r^2h_T}{\Delta}+\frac{\vec{B}\cdot\vec{B}-B_n^2}{m_0}\left(\frac{rm_0c_{f,s}^2}{rm_0c_{f,s}^2-B_n^2}\right)^2\right]=2\left(\frac{r^2h_T}{\Delta}+\frac{\vec{M}_{f,s}\cdot\vec{M}_{f,s}}{m_0}\right) \tag{A.25}$$

In equation A.22, the two velocities, C_f and C_s, are the fast and slow magneto-acoustic speeds respectively and

$$a^{*2} = \frac{rh_T}{\Delta} + \frac{\vec{B}\cdot\vec{B}}{rm_0} \tag{A.26}$$

$$\Delta = rr_P h_T + r_T(1 - rh_P) \tag{A.27}$$

The matrices M and M^{-1} become singular in the following threecases

1. $\vec{B}$ is perpendicular to $\vec{n}$ $(\vec{B}\cdot\vec{n} = 0)$

   If $\vec{B}$ is perpendicular to the direction vector, $\vec{n}$, we obtain

   $$\vec{B}\cdot\vec{n} = 0 \tag{A.28a}$$

   Substitute the equation A.28a into equations A.22 to obtain

   $$c_f^2 = a^{*2} = \frac{rh_T}{\Delta} + \frac{\vec{B}\cdot\vec{B}}{rm_0}$$

   and

   $$c_s = 0 \tag{A.28b}$$

   Substituting equation A.28a into equation A.23, we get

   $$\vec{W}_{f,s} = \frac{c_{f,s}(B_n^2\vec{n} - B_n\vec{B})}{rm_0 c_{f,s}^2 - B_n^2} + c_{f,s}\vec{n} = c_{f,s}\vec{n} \tag{A.28c}$$

   Substitute equation A.28b into equation A.28c to obtain

   $$\vec{W}_s = c_s\vec{n} = 0 \tag{A.28d}$$

   Substituting equation A.28d into equations A.20 and A.21, we can see that the last two columns of the matrix, M, and the last two rows of the matrix, M^{-1} degenerate.

2. $\vec{B}$ and $\vec{n}$ are in the same direction $(B_n{}^2 = \vec{B}\cdot\vec{B})$ and $B_n{}^2 \neq rm_0\frac{rh_T}{\Delta} = rm_0 a^2$

   If $\vec{B}$ and $\vec{n}$ are in the same direction, we obtain

   $$B_n{}^2 = \left(\vec{B}\cdot\vec{n}\right)^2 = \vec{B}\cdot\vec{B} \tag{A.29}$$

   Substituting into equation A.21, we can see that the fourth and fifth rows of matrix

M^{-1} do not exist.

3. $\vec{B}$ and $\vec{n}$ are in the same direction ($B_n^{\,2} = \vec{B} \cdot \vec{B}$) and $B_n^{\,2} = rm_0 \dfrac{rh_T}{\Delta}$

$$B_n^{\,2} = rm_0 \frac{rh_T}{\Delta} \tag{A.30a}$$

Substituting equations A.29 and A.30a into equation A.22, we obtain

$$c_s = c_f = c_a = \frac{rh_T}{\Delta} \tag{A.30b}$$

Substituting equation A.29 into equations A.24 and A.23, we obtain

$$\vec{M}_{f,s} = 0 \tag{A.30c}$$

and

$$\vec{W}_{f,s} = c_{f,s}\vec{n} \tag{A.30d}$$

Substituting equation A.30c into equation A.25, we obtain

$$G_{f,s} = 2\left(\frac{r^2 h_T}{\Delta} + \frac{\vec{M}_{f,s} \cdot \vec{M}_{f,s}}{m_0} \right) = 2\frac{r^2 h_T}{\Delta} \tag{A.30e}$$

Substituting equation A.30a, b, c, d, e into equations A.20 and A.21, we see that, in the matrix M, the sixth column is the same as the eighth column, the seventh column is the same as the ninth column, and in the matrix M^{-1}, the fourth and fifth rows of M^{-1} do not exist, the sixth row is the same as the eighth row, and the seventh row is the same as the ninth row of M^{-1}.

Clearly, further improvement on the eigenvector matrices should be conducted to deal with the degeneration cases.

Appendix B

Four-Sweep DDLGS

This appendix derives the four-sweep DDLGS approximate factorization for solving the triple time equation 2.33 (re-labeled here as B.1).

$$\left(\frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \right) \Delta \tilde{Q}_p = -R_3 \tag{B.1}$$

For convenience, the left hand side operator is defined as L_3 in equation 2.22b (re-labeled as B.2)

$$L_3 = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} + J_{i,j+1} + J_{i,j-1} \tag{B.2}$$

We start the procedure with the two-sweep DDLGS in x-direction (equation 2.40a) and repeat this equation here as equation B.3a,

$$\left(D_x + J_{i-1,j} + J_{i+1,j} + J_{i-1,j} D_x^{-1} J_{i+1,j} \right) \Delta \tilde{Q}_p = -R_3 \tag{B.3a}$$

where D_x is defined in equation 2.35 (re-label as B.4a) as

$$D_x = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i,j+1} + J_{i,j-1} \tag{B.4a}$$

Similarly, we can write the two-sweep DDLGS in y-direction as

$$\left(D_y + J_{i,j-1} + J_{i,j+1} + J_{i,j-1} D_y^{-1} J_{i,j+1} \right) \Delta \tilde{Q}_p = -R_3 \tag{B.3b}$$

where D_y is defined as

$$D_y = \frac{\Gamma_3}{\Delta t_3} + \frac{\Gamma_2}{\Delta t_2} + J_{ij} + J_{i+1,j} + J_{i-1,j} \tag{B.4b}$$

We define a new term D as the exact operator L_3 plus the two error terms in equations

B.3a and b as

$$D = L_3 + J_{i-1,j} D_x^{-1} J_{i+1,j} + J_{i,j-1} D_y^{-1} J_{i,j+1} \tag{B.4c}$$

From equation B.4c, we obtain

$$\begin{aligned} L_3 &= D - J_{i-1,j} D_x^{-1} J_{i+1,j} - J_{i,j-1} D_y^{-1} J_{i,j+1} \\ &= D\left[I - D^{-1}\left(J_{i-1,j} D_x^{-1} J_{i+1,j}\right) - D^{-1}\left(J_{i,j-1} D_y^{-1} J_{i,j+1}\right)\right] \end{aligned} \tag{B.4d}$$

Since the last two terms at the right side of equation B.4d are small for small values Δt_2 or Δt_3, a good approximation for the operator L_3 would be

$$L_3 = D\left[I - D^{-1}\left(J_{i-1,j} D_x^{-1} J_{i+1,j}\right)\right]\left[I - D^{-1}\left(J_{i,j-1} D_y^{-1} J_{i,j+1}\right)\right] \tag{B.5}$$

Upon symmetrizing equation B.5 and substituting into equation B.1, we have the approximately factorized version of equation B.1 as

$$\left[D - J_{i-1,j} D_x^{-1} J_{i+1,j}\right] D^{-1} \left[D - J_{i,j-1} D_y^{-1} J_{i,j+1}\right] \Delta\tilde{Q}_p = -R_3 \tag{B.6}$$

Expanding the operator on the left hand side of equation B.6 and using equation B.4c gives

$$\left[L_3 + \left(J_{i,j-1} D_y^{-1} J_{i,j+1}\right) D^{-1} \left(J_{i-1,j} D_x^{-1} J_{i+1,j}\right)\right] \Delta\tilde{Q}_p = -R_3 \tag{B.7}$$

We can see from equation B.7 that the equations B.6 are an approximation for the exact equation with an error, $\left(J_{i,j-1} D_y^{-1} J_{i,j+1}\right) D^{-1} \left(J_{i-1,j} D_x^{-1} J_{i+1,j}\right)$. Again, this error is small if Δt_2 or Δt_3 is small.

Equation B.6 can be expanded as the following two equations B.8a and b in a similar way as we expand equation 2.21e to equations 2.39b and e in chapter two

$$\left[D - J_{i-1,j} D_x^{-1} J_{i+1,j}\right] \Delta\tilde{Q}_p^{**} = -R_3 \tag{B.8a}$$

$$\left[D - J_{i,j-1} D_y^{-1} J_{i,j+1}\right] \Delta\tilde{Q}_p = D\Delta\tilde{Q}_p^{**} \tag{B.8b}$$

where $\Delta\tilde{Q}_p^{**}$ is an intermediate variable.

By substituting equation B.4c into equations B.8a and b, we can see that the left side

operators of equation B.8a and b are exactly the same as the left side operator of equation B.3a and b. Accordingly, equations B.8a and B.8b can be further expanded into two equations respectively in the exactly the same way as we did for the two-sweep DDLGS in chapter two. After doing so, we obtain the following four equations

$$\left(D_x + J_{i-1,\,j}\right)\Delta\tilde{Q}_p^{\,*} = -R_3 \tag{B.9a}$$

$$\left(D_x + J_{i+1,\,j}\right)\Delta\tilde{Q}_p^{\,**} = D_x\Delta\tilde{Q}_p^{\,*} \tag{B.9b}$$

$$\left(D_y + J_{i,\,j-1}\right)\Delta\tilde{Q}_p^{\,***} = D\Delta\tilde{Q}_p^{\,**} \tag{B.9c}$$

$$\left(D_y + J_{i,\,j+1}\right)\Delta\tilde{Q}_p = D_y\Delta\tilde{Q}_p^{\,***} \tag{B.9d}$$

Equation set B.9 is the four-step form of the four-sweep DDLGS approximate factorization and exactly the same as equation set 2.49 in chapter two.

There is another form of the four-sweep DDLGS approximate factorization [9], which is very beautiful in form and very easy for coding. In the following, we show that it is equivalent to equation set B.9.

We begin with the Y-direction sweep equation B.8b. We substitute equation B.4c into equation B.8b and write equation B.8b in a factorized form as

$$\left(D_y + J_{i,\,j-1}\right)D_y^{-1}\left(D_y + J_{i,\,j+1}\right)\Delta\tilde{Q}_p = D\Delta\tilde{Q}_p^{\,**} \tag{B.10a}$$

Solving equation B.8a for $D\Delta\tilde{Q}_p^{\,**}$ and substituting into equation B.10a to obtain

$$\left(D_y + J_{i,\,j-1}\right)D_y^{-1}\left(D_y + J_{i,\,j+1}\right)\Delta\tilde{Q}_p = -R_3 + \left(J_{i,\,j-1}D_y^{-1}J_{i,\,j+1}\right)\Delta\tilde{Q}_p^{\,**} \tag{B.10b}$$

The operator of the last term on the right hand side of equation B.10b can be written as

$$J_{i,\,j-1}D_y^{-1}J_{i,\,j+1} = \left(D_y + J_{i,\,j-1}\right)D_y^{-1}J_{i,\,j+1} - J_{i,\,j+1} \tag{B.11}$$

Substituting equation B.11 into equation B.10b, after arrangement, we obtain

$$\left(D_y + J_{i,\,j-1}\right)D_y^{-1}\left[\left(D_y + J_{i,\,j+1}\right)\Delta\tilde{Q}_p - J_{i,\,j+1}\Delta\tilde{Q}_p^{\,**}\right] = -R_3 - J_{i,\,j+1}\Delta\tilde{Q}_p^{\,**} \tag{B.12}$$

Define the last two terms at the left hand side of equation B.12 as

$$D_y^{-1}\left[\left(D_y + J_{i,j+1}\right)\Delta\tilde{Q}_p - J_{i,j+1}\Delta\tilde{Q}_p^{**}\right] = \Delta\tilde{Q}_p^{***} \tag{B.13}$$

Then equation B.12 becomes

$$\left(D_y + J_{i,j-1}\right)\Delta\tilde{Q}_p^{***} = -R_3 - J_{i,j+1}\Delta\tilde{Q}_p^{**} \tag{B.14a}$$

Solving equation B.14a for $D_y\Delta\tilde{Q}_p^{***}$ and substituting into equation B.13, after arrangement we obtain

$$\left(D_y + J_{i,j+1}\right)\Delta\tilde{Q}_p = -R_3 - J_{i,j-1}\Delta\tilde{Q}_p^{***} \tag{B.14b}$$

Equations B.14a and b are the expansion of Y-direction sweep equation B.8b.

The X-direction sweep equation B.8a is already expanded to equations B.9a and b. We solve equation B.9a for $D_x\Delta\tilde{Q}_p^{*}$ and substitute the solution into equation B.9b to obtain

$$\left(D_x + J_{i+1,j}\right)\Delta\tilde{Q}_p^{**} = -R_3 - J_{i-1,j}\Delta\tilde{Q}_p^{*} \tag{B.14c}$$

Finally, using equations B.2, B.4a and B.4b and writing the operator at the left hand side of equations B.9a, B.14a, b and c in terms of the exact operator L_3, we obtain

$$\left(L_3 - J_{i+1,j}\right)\Delta\tilde{Q}_p^{*} = -R_3 \tag{B.15a}$$

$$\left(L_3 - J_{i-1,j}\right)\Delta\tilde{Q}_p^{**} = -R_3 - J_{i-1,j}\Delta\tilde{Q}_p^{*} \tag{B.15b}$$

$$\left(L_3 - J_{i,j+1}\right)\Delta\tilde{Q}_p^{***} = -R_3 - J_{i,j+1}\Delta\tilde{Q}_p^{**} \tag{B.15c}$$

$$\left(L_3 - J_{i,j-1}\right)\Delta\tilde{Q}_p = -R_3 - J_{i,j-1}\Delta\tilde{Q}_p^{***} \tag{B.15d}$$

Equation set B.15 is exactly the same as equation set 2.50 in chapter two. The left side of the equation looks symmetry to the right side except equation B.15a. The equation set B.15 seems arbitrary, but from the derivation, we see that it is a good approximation for our exact equation B.1.

Vita

Xiaoqiang Zeng was born in Pingxiang county, Jiangxi Province, China on December 12, 1975. He completed his five-year study in elementary school and six-year study in high school from 1982 to 1993 in his hometown. In 1993, he attended University of Science & Technology Beijing for his undergraduate study majoring in thermal engineering and earned his B.S. degree in July 1997. After that he attended the Institute of Engineering Thermophysics, Chinese Academy of Sciences for his master's study under Dr. Bin Zhu's guidance and obtained his M.S degree in July 2000. In September 2000, he came to the United States to pursue his Ph.D degree at the University of Tennessee Space Institute.