A Fully Coupled Newton-Krylov Solver with a One-Equation Turbulence Model

by

Todd Thomas Chisholm

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto

Abstract

A Fully Coupled Newton-Krylov Solver with a One-Equation Turbulence Model

Todd Thomas Chisholm

<todd@oddjob.utias.utoronto.ca>

Doctor of Philosophy

Graduate Department of Institute for Aerospace Studies

University of Toronto

2006

An efficient Newton-Krylov method is presented for the computation of steady, compressible, high Reynolds number flows about single- and multi-element airfoils using structured grids. A second-order centred-difference method is used to discretize the Navier-Stokes equations, using either scalar or matrix dissipation for stability. Turbulence is modeled following Spalart and Allmaras. The equations are solved with Newton's method, using Generalized Minimal Residual to approximately solve the linear system. Incomplete lower-upper preconditioning is used to speed linear system convergence. Full coupling is used between the mean flow and turbulence model equations.

Experiments are presented to determine the most efficient, robust approach. Extensive use of a node- and iteration-varying time step stabilizes the Newton method, and allows rapid convergence. In particular, a new time step replaces the M-matrix method of Spalart and Allmaras to ensure positive updates. Equation scaling ensures that the linear solver provides an adequate solution of the linear system. Implementation details of the reverse Cuthill-McKee reordering are examined and optimized, revealing that a downwind node should be chosen as the root node in the reordering. A potential problem with errors in the matrix-free method is addressed by optimizing the perturbation parameter.

The solver has been applied to a variety of test cases, from simple inviscid single-element examples to complex, multi-element configurations at high angles-of-attack and Reynolds numbers. It performs very well in all examples, typically converging in around 1000 equivalent function evaluations when transition terms are disabled, and around 2000 when they are enabled. It is also shown to be a good choice for use with numerical airfoil optimization.

Acknowledgements

I have had a great deal of help on the long path to finishing this thesis. My parents provided moral and financial support, as well as gentle encouragement at the right times.

Professor David Zingg has been a wonderful supervisor. His patience and guidance are greatly appreciated. His insight and ability to ask the right questions has benefited everyone who had the pleasure of working with him. My thesis committee, Tom Nelson and Professor Jean Sislian, were very helpful in guiding this research.

My partner Kathryn Ross has been incredibly understanding and supportive. She shared the excitement when things went well and calmly put up with the frustrations and long hours when they did not. Don and Edie Ross welcomed me into their house, making it possible to finish when things were most difficult.

My fellow researchers at UTIAS were always available to help when the going got difficult. Max Blanco, Marian Nemec, Stan DeRango, Luis Manzano, Jason Lassaline, Mike Sullivan and Gary Zuliani kept my time from being all work and made the work a pleasure. Alberto Pueyo was tremendous in helping with understanding the details of the research.

Jon Driver kindly provided the grids used in the optimization study.

Finally, I'd like to thank Bombardier, the Governments of Ontario and Canada, and the University of Toronto for financial assistance.

CONTENTS

List of Figures

List of Tables

NOTATION

ALPHANUMERIC

α Angle of attack

δ Matrix-free perturbation size parameter

γ Ratio of specific heats

σ Left hand side dissipation parameter

ν Kinetic eddy viscosity

ν_t Turbulence kinetic eddy viscosity

$\tilde{\nu}$ Turbulence model variable

τ Stress terms

ξ, η Computational directions

μ Laminar eddy viscosity

μ_t Turbulent eddy viscosity

Δ, ∇ Difference operators

κ_2, κ_4 Dissipation constants

Δt Time step

C_l coefficient of lift

C_p coefficient of pressure

E inviscid flux in x-direction

E_v viscous flux in x-direction

F inviscid flux in y-direction

F_v viscous flux in y-direction

$\mathcal{F}$ The discretized equations

H enthalpy

J metric Jacobian

K_m Krylov subspace of dimension m

M Mach number

N Total number of equations in the system

$\hat{P}$ viscous flux in the thin-layer approximation

Q vector of conservative variables at a single node

$\hat{Q}$ vector of conservative variables at a single node after applying curvilinear coordinate transformation

$\hat{\mathcal{Q}}$ vector of conservative variables over all nodes

$R_{1,2}$ Riemann invariants

S entropy

$\mathcal{T}$ the vector with each element containing either zero, if the associated equation results from a boundary, or $\frac{d\hat{Q}}{dt}$ in the case of an interior equation.

U contravariant velocity in ξ-direction

V contravariant velocity in η-direction

V_m $N \times m$ matrix containing the basis of the Krylov subspace

V_n velocity component normal to boundary

V_t velocity component tangential to boundary

a speed of sound

b right-hand-side of linear system solved by GMRES

c chord of airfoil

d artificial dissipation

e total energy

$h_{i,j}$ elements of Hessenberg matrix

m size of Krylov subspace

Λ Matrix of eigenvalues

$\mathcal{R}e$ Reynolds number

$\mathcal{A}$ Jacobian matrix

$\mathcal{M}$ Base matrix for the preconditioner

$\mathcal{S}_r$ Diagonal row scaling matrix

$\mathcal{S}_c$ Diagonal column scaling matrix

Chapter 1

INTRODUCTION

1.1 Background

In 1976, Cray Research released the Cray 1 supercomputer, with a world record theoretical capability to perform 160 million float point operations per second (Mflops), using 8 megabytes of memory. Six years later, it was superseded by the XMP, boasting 16 megabytes of memory and capable of 235 Mflops per processor. In 1990, the C90 achieved 1 Gflop per processor, with 256 megabytes of memory. Today, a desktop computer can achieve these processing rates, and it is common to have more than a gigabyte of high-speed memory.

Despite these enormous advances in speed, numerical simulations of airflow have always been limited by computational resources. New techniques to increase accuracy are not practical until hardware is available that can perform the calculations in a reasonable time. For this reason, it is important that we make the best possible use of these resources. Algorithms and computers are now advanced enough to be used as regular design tools in industry. However, long run times are still a major problem. Optimization is an emerging technique with great promise, but often requires many flow solves. Our purpose is to reduce the solution time needed to compute steady flows over complex airfoil geometries.

1.2 Iterative methods for steady flows

The equations which predict airflow have been subjected to an astonishing number of attempts to produce efficient solutions. These efforts fall into two broad and not clearly defined categories: explicit and implicit. One of the first examples of the former was introduced by MacCormack [1], who used a predictor-corrector method. The highly stretched grids seen with turbulent viscous flows severely slow convergence of explicit methods. To combat this effect, multigrid was introduced. Brandt gives a good overview of this method [2]. This was proven to be very effective in accelerating convergence. South and Brandt [3] and Jameson [4] demonstrated solving transonic potential flows with multigrid. The Euler equations were solved by Ni [5] and Jameson [6]. Martinelli et al. [7] and Mavriplis [8] successfully used an explicit method with multigrid to solve the Navier-Stokes equations. One of the most popular and efficient arrangements uses Runge-Kutta time marching, local time-stepping, and implicit residual smoothing with multigrid, as given by Jameson [9]. The Runge-Kutta method is easily implemented and requires little in the way of memory resources, but has small stability limits. It is clearly an explicit method, but adding some measure of 'implicitness' increases the stability, allowing larger time steps and efficiency.

This can be contrasted with the approximate factorization method first used with the Euler and Navier-Stokes equations by Beam and Warming [10]. This method is an implicit method, in that the implicit Euler method is used. However, the 'implicitness' is weakened by using an approximation to factor the left hand side matrix and using explicit boundary conditions. This reduces the computational time required to perform each iteration, but limits the effective time step. Steger introduced ARC2D [11], which used the approximate factorization method. A number of improvements were added to ARC2D, including diagonalization of the blocks of the implicit operator by Pulliam and Chaussee [12]. Local time-stepping and grid sequencing further increased efficiency for steady state problems. Nelson [13] extended ARC2D to use multi-block grids, creating the code called TORNADO. He also added the ability to use the one-equation transport turbulence model by Spalart and Allmaras [14]. These allowed the solution of multi-element airfoils. Godin et al.[58] implemented and tested two-equation turbulence models in TORNADO and a new single-element code, called Cyclone. De Rango and Zingg [15] added high-order spatial discretization to both TORNADO and Cyclone. Multigrid acceleration of the approximate factorization method was added to ARC2D by Chisholm and Zingg [16] and by Manzano [17] to TORNADO. These additions produced a very efficient solver.

As a result of approximate factorization, convergence slows as the time step is increased. Hence there is an optimum Courant number, but convergence can slow on highly stretched grids, such as those seen with turbulent flows. To relieve this limitation, researchers have looked to the fully implicit Newton's method. This requires the exact solution of the linear equation at each iteration. A number of authors have done this with a direct solver, on both the inviscid and viscous equations [18], [19], [20], [21], [22], [23]. The direct solve makes this method very robust, but the cost in CPU time and memory make this approach quite uncompetitive.

It would appear that becoming 'too implicit' can also be a detriment. The problem in this case is the cost of the direct solver. If this is replaced with an iterative linear solver, we have a quasi-Newton method. Iterative solvers have the wonderful property of being able to reduce the linear residual by a

specified amount. The time required to perform each Newton iteration can be reduced substantially if we do not fully converge the linear system. In a method like this, the linear solve can require the vast majority of the total time. An efficient linear solver is therefore very important to the overall efficiency. The first linear solvers were stationary methods such as the symmetric successive over-relaxation (SSOR) method. The newer non-stationary (Krylov) methods are substantially faster than stationary methods. We have a choice of a number of methods, each with somewhat different characteristics. For example, the Conjugate Gradient method by Hestenes and Stiefel [24] is appropriate only for symmetric positive-definite matrices, while the Generalized Minimal Residual (GMRES) due to Saad and Schultz [25] (which is used in this work) is applicable with general matrices, but requires storing subspace vectors. A summary of these methods can be found in [26].

The application of the Newton-Krylov method to equations predicting fluid flow has been performed by a number of researchers. Pueyo [27] provides an excellent summary of many of these efforts. Here we focus on pivotal examples and more recent innovations. Wigton, et al. [28] were one of the first to apply a Krylov method in solving the Navier-Stokes equations. Also of particular interest was the fact that they used a matrix-free process to provide the matrix-vector product needed by GMRES. A more recent survey of approaches to the matrix-free method (called Jacobian-free by the authors to indicate the possibility of explicitly forming a preconditioner matrix) is provided by Knoll and Keyes [29].

Venkatakrishnan and Mavriplis [30] experimented with a Newton-Krylov method on unstructured grids for steady turbulent viscous flows. They used an approximate Jacobian. This simpler method reduced the storage required and made the matrix easier to calculate, but increased the number of Newton iterations. They related the time-step to the inverse of the norm of the residual, though they restricted it to a maximum value. This was likely needed by their choice of preconditioner. They experimented with block-diagonal, Incomplete Lower-Upper factorization with a level of fill of zero (ILU(0)), and Symmetric Successive Over-Relaxation preconditioners. They concluded that the incomplete factorization preconditioner is more effective.

Ajmani et al. [31] also compared different types of preconditioners, with a quasi-Newton method and a modified Jacobian. The block ILU(0) and Lower-Upper SSOR were tried, with the latter being more efficient. This solver was compared to an implicit line Gauss-Seidel solver and an approximately factored solver for hypersonic flows around a cylinder and transonic flow through a turbine cascade. They found the Newton-Krylov solver to be competitive with these standard solvers. In later work [32], they compared different Krylov methods and concluded that GMRES was a better choice than biconjugate gradients stabilized (Bi-CGSTAB) or quasi-minimal residual (QMR).

McHugh and Knoll [33] also performed a comparison study of Krylov methods. They tested conjugate gradients squared (CGS), transpose-free quasi-minimal residual (TFQMR), bi-conjugate gradient (Bi-CG), and GMRES. They also compared the matrix-free approach to the explicit calculation of a matrix. They concluded that GMRES with the matrix-free implementation is a good choice.

Forsyth and Jiang studied the solution of inviscid [34] and laminar viscous [35] flows with a Newton-CGSTAB method. Their preconditioner was more advanced, using ILU with fill. They observed convergence problems when solving flows with strong shocks and using a simplified matrix for the Jacobian. Also, when compared to the true Jacobian, the simplifed matrix approach was less efficient. They did not modify the matrix used for the preconditioner and so required a fill level of at least two to get

convergence.

Rogers [36] compared a number of implicit approaches to solving the incompressible Navier-Stokes equations with artificial compressibility. These included Point-Jacobi relaxation, Gauss-Seidel line-relaxation, and incomplete LU decomposition (with zero fill) used as a solver. In addition, he used a GMRES method with each of the above as a preconditioner, which he concluded was the best approach when used with ILU.

Barth and Linton [37] used a parallelized matrix-free Newton-GMRES method to solve the three-dimensional turbulent Navier-Stokes equations on unstructured meshes. They used a variant of the ILU(0) preconditioner with a new technique for constructing the matrix-vector products.

Pueyo [27] studied the optimization of a number of parameters for a Newton-GMRES method solving the Navier-Stokes equations with an algebraic turbulence model on single-block structured grids. He suggested the use of the matrix-free technique. He noted that the method is much more successful with careful modification of the approximation to the Jacobian matrix that the preconditioner is based on. Using only second-difference dissipation on the LHS, with an increase in the dissipation coefficient, results in roughly half the memory usage and a much more effective preconditioner, since the factorization is much more stable. He found that ILU with a fill of about two is a good choice, having compared ILU with various levels of fill and a version of ILU with thresholding (ILUT, developed by Saad [38]). The ordering used for the nodes in the preconditioner was also found to be important. Pueyo presented a number of reordering methods, and found the reverse Cuthill-McKee algorithm [39] to be most effective. The program that Pueyo produced, PROBE, is the basis for much of this work.

Geuzaine [40] wrote an upwind finite volume solver for unstructured grids, using a Newton-Krylov solver. He studied both the Spalart-Allmaras and the k-ϵ turbulence models. He used different levels of fill in an ILU preconditioner based on a matrix with only nearest-neighbours. ILU(0) was found to be sufficient, but higher fill was more efficient in CPU time. He concluded that this approach is competitive with multigrid techniques.

The preconditioner is a particularly important part of the algorithm and thus has been a focus for research. Besides the approximate LU factorization techniques in common use, multigrid as a preconditioner has also been successfully implemented by Knoll and Mousseau [41] and Pernice and Tocci [42]. Fully matrix-free preconditioners based on the physics of the problem have been developed for radiation diffusion problems by Mousseau and Knoll [43]. These have the advantage of substantial memory savings over an ILU preconditioner, but are not directly applicable to different problems. Luo et al. [44] developed a three-dimensional unstructured solver for the turbulent Navier-Stokes equations using a lower-upper symmetric Gauss-Seidel preconditioner. The preconditioner did not require the storage of a matrix, relying on approximating the Jacobian with numerical fluxes. Preconditioners designed with parallelization in mind have recently been investigated. In particular, Newton-Krylov-Schwarz methods have been explored by Cai et al. [45] and Tidriri [46] for fluid flows and McHugh et al. [47] for combustion problems.

An interesting recent trend has been the use of highly implicit methods in time accurate studies. Isonos and Zingg [48] have developed an interesting Runge-Kutta-Newton-Krylov algorithm and tested it with unsteady laminar flows. Another very promising field of research is numerical optimization. Nemec, et al. [49] showed that combining a gradient-based optimization technique with a Newton-Krylov solver

can be very efficient. This is a natural combination, since a very precise Jacobian is required for the optimization. Gatsis and Zingg [50] also studied optimization, using a fully-coupled approach, solving the system with a Newton-Krylov method.

A number of researchers are experimenting with three-dimensional equations using Newton-GMRES solvers. Nichols and Zingg [51] have solved the Euler equations on multi-block grids. Wong and Zingg [52] have solved the Navier-Stokes equations with the Spalart-Allmaras turbulence model on unstructured grids. In both cases, they concluded the approach was highly efficient with an ILU preconditioner with fill. Nielsen et al. [53] solved the three-dimensional Euler equations with a Newton-GMRES method and an ILU(0) preconditioner. They experienced some difficulties with stabilizing the initial iterations.

1.3 Objectives

Pueyo demonstrated the potential of using a Newton-Krylov method to solve the Navier-Stokes equations on structured grids. Our first objective is to create a new solver (which we call Scirocco) with more capabilities. A one-equation transport turbulence model and matrix dissipation increase the accuracy of the solution. The extension to multi-block grids permits the solution of multi-element airfoils. These changes cause problems with the Newton-Krylov solver as presented by Pueyo, which leads us to arguably the most important objective. To maximize the efficiency and reliability of the solver, we must increase our understanding of the behaviour of the components of the Newton-Krylov method. To ease development, the PETSc package [54] is used to handle many of the matrix operations.

Chapter 2

GOVERNING EQUATIONS

This chapter discusses the equations solved by Scirocco. These include the two-dimensional Euler or thin-layer Navier-Stokes equations and the Spalart-Allmaras one-equation turbulence model equations.

2.1 Navier-Stokes Equations

The Navier-Stokes equations predict aerodynamic flows. They operate on the non-dimensional conservative variables

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.1}$$

We have scaled the dimensional Cartesian coordinates $(\tilde{x}, \tilde{y})$, density $(\tilde{\rho})$, velocity $(\tilde{u}, \tilde{v})$, total energy $(\tilde{e})$, and time $(\tilde{t})$, as

$$x = \frac{\tilde{x}}{c}, \quad y = \frac{\tilde{y}}{c}, \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty}, \quad u = \frac{\tilde{u}}{\tilde{a}_\infty}, \quad v = \frac{\tilde{v}}{\tilde{a}_\infty}, \quad e = \frac{\tilde{e}}{\tilde{\rho}_\infty a_\infty^2}, \quad t = \frac{\tilde{t} a_\infty}{c} \tag{2.2}$$

where the ∞ subscript indicates free-stream quantities, c is the airfoil chord, and a is the sound speed, which for ideal fluids is $a = \sqrt{\gamma p/\rho}$. The ratio of specific heats, γ, is taken as 1.4 for air. The equations in two-dimensional Cartesian coordinates are

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = \mathcal{R}e^{-1} \left(\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} \right) \tag{2.3}$$

The inviscid fluxes are

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e+p)u \end{bmatrix}, \quad F = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ (e+p)v \end{bmatrix} \tag{2.4}$$

The viscous fluxes are

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ E_{v,4} \end{bmatrix}, \quad F_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ F_{v,4} \end{bmatrix} \tag{2.5}$$

with the stress terms

$$\begin{aligned} \tau_{xx} &= (\mu + \mu_t)(4u_x - 2v_y)/3 \\ \tau_{xy} &= (\mu + \mu_t)(u_y - v_x)/3 \\ \tau_{yy} &= (\mu + \mu_t)(-2u_x + 4v_y)/3 \\ E_{v,4} &= u\tau_{xx} + v\tau_{xy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\partial_x(a^2) \\ F_{v,4} &= u\tau_{xy} + v\tau_{yy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\partial_y(a^2) \end{aligned} \tag{2.6}$$

where μ and μ_t are the dynamic and turbulent eddy viscosities, respectively, and $\mathcal{P}r = 0.72$ and $\mathcal{P}r_t = 0.90$ are the laminar and turbulent Prandtl numbers, taken as constants. The pressure, p, is found from the equation of state for an ideal gas

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{2.7}$$

$\mathcal{R}e$ is the Reynolds number, which is defined as

$$\mathcal{R}e = \frac{\rho_\infty \; c \; a_\infty}{\mu_\infty} \tag{2.8}$$

2.1.1 Generalized Curvilinear Coordinate Transformation

Since Scirocco uses structured grids, a curvilinear coordinate transformation is used to map physical space into an evenly spaced computational space

$$\xi = \xi(x, y), \qquad \eta = \eta(x, y) \tag{2.9}$$

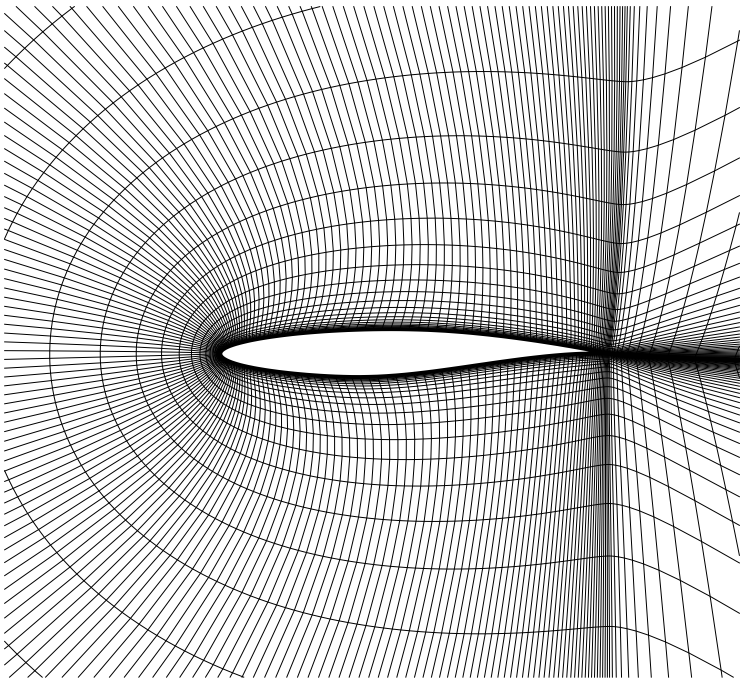
Details can be found in [55]. Figure 2.1 shows examples of a single- and a multi-block grid.

The transformed conservative equations operate on the state variable vector

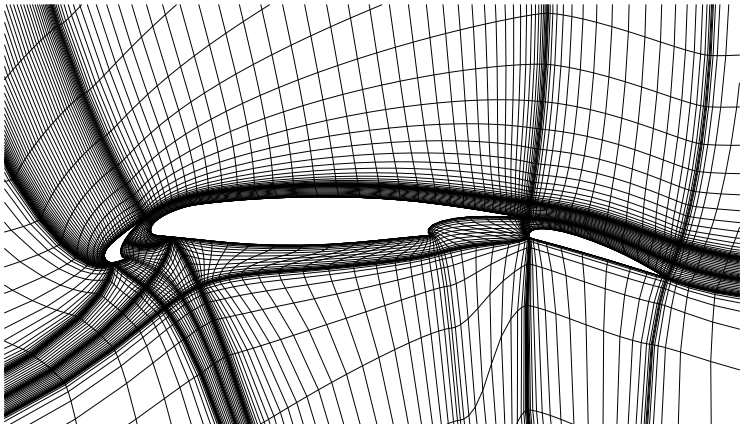
$$\hat{Q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.10}$$

with J being the metric Jacobian of the transformation

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi) \tag{2.11}$$



(a) Single-block C-grid



(b) Multi-block H-grid

Figure 2.1: Curvilinear grid examples

The two-dimensional Navier-Stokes equations in curvilinear coordinates are

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = \mathcal{R}e^{-1}\left(\frac{\partial \hat{E}_v}{\partial \xi} + \frac{\partial \hat{F}_v}{\partial \eta}\right) \tag{2.12}$$

The inviscid flux vectors become

$$\hat{E} = J^{-1}\begin{bmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U \end{bmatrix}, \qquad \hat{F} = J^{-1}\begin{bmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V \end{bmatrix} \tag{2.13}$$

U and V are the contravariant velocities

$$U = \xi_x u + \xi_y v, \qquad V = \eta_x u + \eta_y v \tag{2.14}$$

The viscous fluxes are

$$\hat{E}_v = J^{-1}(\xi_x E_v + \xi_y F_v), \qquad \hat{F}_v = J^{-1}(\eta_x E_v + \eta_y F_v) \tag{2.15}$$

with the transformed stress terms

$$\begin{aligned}
\tau_{xx} &= (\mu+\mu_t)(4(\xi_x u_\xi + \eta_x u_\eta) - 2(\xi_y v_\xi + \eta_y v_\eta))/3 \\
\tau_{xy} &= (\mu+\mu_t)(\xi_y u_\xi + \eta_y u_\eta \xi_x v_\xi + \eta_x v_\eta) \\
\tau_{yy} &= (\mu+\mu_t)(-2(\xi_x u_\xi + \eta_x u_\eta) + 4(\xi_y v_\xi + \eta_y v_\eta))/3 \\
E_{v,4} &= u\tau_{xx} + v\tau_{xy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma-1)^{-1}(\xi_x \partial_\xi(a^2) + \eta_x \partial_\eta(a^2)) \\
F_{v,4} &= u\tau_{xy} + v\tau_{yy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma-1)^{-1}(\xi_y \partial_\xi(a^2) + \eta_y \partial_\eta(a^2))
\end{aligned} \tag{2.16}$$

2.1.2 Thin-Layer Approximation

In the airfoil specific case, it is assumed that the viscous effects resulting from the derivatives in the direction of the body are small enough to be neglected. Applying this assumption results in the thin-layer Navier-Stokes equations. This assumption does not hold for low Reynolds numbers, or in a case with large regions of separation. The equations are

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = \mathcal{R}e^{-1}\frac{\partial \hat{P}}{\partial \eta} \tag{2.17}$$

The modified viscous flux vector is

$$\hat{P} = J^{-1}\begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x(u m_1 + v m_3 + m_4) + \eta_y(u m_2 + v m_3 + m_5) \end{bmatrix} \tag{2.18}$$

with

$$m_1 = (\mu+\mu_t)(4\eta_x u_\eta - 2\eta_y v_\eta)/3$$

$$\begin{aligned} m_2 &= (\mu+\mu_t)(\eta_y u_\eta + \eta_x v_\eta)/3 \\ m_3 &= (\mu+\mu_t)(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \\ m_4 &= (\mu\mathcal{P}r^{-1} + \mu_t\mathcal{P}r_t^{-1})(\gamma-1)^{-1}\eta_x\partial_\eta(a^2) \\ m_5 &= (\mu\mathcal{P}r^{-1} + \mu_t\mathcal{P}r_t^{-1})(\gamma-1)^{-1}\eta_y\partial_\eta(a^2) \end{aligned} \tag{2.19}$$

2.1.3 Boundary Conditions

On each side of each block, a boundary condition must be specified for every node. There are three classes of boundaries: free-stream, solid surface, and interior (block to block). Scirocco is not constrained to one boundary type per block side.

Free-Stream

For inviscid flows, the four required equations are derived by setting the two locally one-dimensional Riemann invariants R_1 and R_2, the tangential velocity V_t, and an entropy function to either free-stream values or values extrapolated from the interior. The Riemann invariants are

$$R_1 = V_n + \frac{2a}{\gamma-1} \tag{2.20}$$

$$R_2 = V_n - \frac{2a}{\gamma-1} \tag{2.21}$$

Where V_n is the velocity normal to the boundary. Note that $V_n < 0$ is considered an inflow situation. The entropy function used is

$$S = \frac{\rho^\gamma}{p} \tag{2.22}$$

In the inflow case, R_2 is extrapolated from the interior, and R_1, V_t, and S are set to free-stream conditions. In the outflow case, R_1 is set from the interior, while the remaining three are set to free-stream.

Effects resulting from the wake crossing the outflow boundary make this approach inappropriate for viscous flows downstream of the body. In this case, a simple zeroth-order extrapolation of ρ, ρu, ρv, and p is used. For all other free-stream boundaries the previous Riemann conditions are used.

Pulliam [55] has shown that, in the case of lifting bodies, the free-stream boundary needs to be as many as 96 chords away from the body in order to minimize the effect of this boundary. This distance can be significantly reduced by modifying the free-stream velocity at the boundary with a perturbation resulting from the solution of the compressible potential vortex solution, following Salas et al. [56]. This gives

$$u_f = u_\infty + \frac{\beta\Gamma\sin(\theta)}{2\pi r[1 - M_\infty^2\sin^2(\theta-\alpha)]} \tag{2.23}$$

where $\Gamma = \frac{1}{2}M_\infty c C_l$, c is the chord of the airfoil, C_l is the coefficient of lift, M_∞ is the free-stream Mach number, α is the angle of attack, $\beta = \sqrt{1-M_\infty^2}$, and r and θ are the polar coordinates to the boundary node, with respect to the quarter-chord point of the airfoil. The speed of sound is also corrected to ensure constant free-stream enthalpy at the boundary.

$$a_f^2 = (\gamma-1)\left(H_\infty - \frac{1}{2}(u_f^2 + v_f^2)\right) \tag{2.24}$$

Zingg [57] demonstrated that the minimum distance to the free-stream boundary for accurate drag prediction was reduced to 12 chords using this technique.

Solid Wall

When solving inviscid flows, tangency is required at solid walls, giving $V_n = 0$. The tangential velocity and pressure are extrapolated from the interior. Stagnation enthalpy, $(e+p)/\rho$, is set to the free-stream value, H_∞.

Viscous flows require a no-slip condition, giving $u = 0$, and $v = 0$. The gradient of pressure is set to zero

$$\frac{\partial p}{\partial \eta} = 0 \tag{2.25}$$

Finally, an adiabatic condition is assumed at the surface, which when coupled with the zero pressure gradient and the perfect gas law results in a zero gradient of density

$$\frac{\partial \rho}{\partial \eta} = 0 \tag{2.26}$$

Interior Boundaries

Scirocco can treat interior, or block to block, boundaries in two ways. The same equations that are applied on the interior may used. An average condition may also be used, so that the variables ρ, ρu, ρv, and p are averaged from the two neighbouring nodes in the computational direction perpendicular to the boundary.

2.2 Turbulence Model

The effect of the turbulence model is found in the Navier-Stokes equations through the turbulence viscosity term μ_t. We use the one-equation Spalart-Allmaras [14] model to determine μ_t. This model has been shown to work well for high-lift multi-element flows, see Godin et al. [58]. The dependent variable is $\tilde{\nu}$. The equation is

$$\begin{aligned}\frac{D\tilde{\nu}}{Dt} = c_{b1}[1-f_{t2}]\tilde{S}\tilde{\nu} + \frac{1+c_{b2}}{\sigma}\nabla\cdot[(\nu+\tilde{\nu})\nabla\tilde{\nu}] - \frac{c_{b2}}{\sigma}(\nu+\tilde{\nu})\nabla^2\tilde{\nu} \\ - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + f_{t1}\Delta U^2\end{aligned} \tag{2.27}$$

The first term on the right hand side is the production term, while the fourth is the destruction. The second and third are diffusion terms, and the fifth is the trip, or transition, term.

The kinematic eddy viscosity ν_t is found from $\tilde{\nu}$ by

$$\nu_t = \tilde{\nu} f_{v1} \tag{2.28}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.29}$$

and

$$\chi = \frac{\tilde{\nu}}{\nu} \tag{2.30}$$

The modified vorticity, $\tilde{S}$ in the production term is given by

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \tag{2.31}$$

where S is the magnitude of the vorticity, d is the distance to the wall, and

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.32}$$

The destruction function f_w is

$$f_w = g \left[\frac{1 + c_{w3}^3}{g^6 + c_{w3}^6} \right]^{\frac{1}{6}} \tag{2.33}$$

with

$$g = r + c_{w2}(r^6 - r) \tag{2.34}$$

and

$$r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \tag{2.35}$$

Transition is enforced using the two functions f_{t1} and f_{t2}. The former is a source term 'seeding' the turbulence in the region of the trip points, while the latter tends to bring small values of $\tilde{\nu}$ to zero. This helps to ensure that transition only occurs near the trip point. The functions are

$$f_{t1} = c_{t1} g_t \exp(-c_{t2} \frac{\omega_t^2}{\Delta U^2} [d^2 + g_t^2 d_t^2]) \tag{2.36}$$

$$f_{t2} = c_{t3} \exp(-c_{t4} \chi^2) \tag{2.37}$$

The factor g_t ensures that the trip term is non-zero over at least a few nodes. It is defined as

$$g_t = \min \left(0.1, \frac{\Delta U}{\omega_t \Delta x_t} \right) \tag{2.38}$$

d_t is the distance to the trip point, ω_t is the vorticity at the trip point, and Δx_t is the grid spacing at the trip point. ΔU is the velocity difference between the point at which the equation is being calculated and the trip point. The remaining constants are

$$\begin{aligned}
c_{b1} &= 0.1355 & c_{b2} &= 0.622 \\
c_{t1} &= 5.0 & c_{t2} &= 2.0 \\
c_{t3} &= 1.2 & c_{t4} &= 0.5 \\
c_{w1} &= c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma & & \\
c_{w2} &= 0.3 & c_{w3} &= 2.0 \\
c_{v1} &= 7.1 & & \\
\sigma &= \tfrac{2}{3} & \kappa &= 0.41
\end{aligned} \tag{2.39}$$

f_{t1} is only evaluated at nodes within a small radius of the trip node, because it drops off exponentially with decreasing wall distance. This significantly simplifies the code involved with choosing which trip node to use at each interior node.

We also have the option of applying the modifications first seen in [59]. The vorticity factor is changed to keep $\tilde{S}$ positive.

$$\tilde{S} = S f_{v3} + \frac{\tilde{\nu}_T}{\kappa^2 d^2} f_{v2} \tag{2.40}$$

$$f_{v2} = \left(1 + \frac{\chi}{c_{v2}}\right)^{-3} \tag{2.41}$$

$$f_{v3} = \frac{(1 + \chi f_{v1})(1 - f_{v2})}{\chi} \tag{2.42}$$

with $c_{v2} = 5.0$. This seems to help reduce the effect of local minima.

2.2.1 Boundary Conditions

The boundary conditions used with the Spalart-Allmaras model are quite simple. At solid walls, $\tilde{\nu}$ is set to zero. At far-field boundaries, we have two options. TORNADO simply sets $\tilde{\nu}$ to a free-stream value, typically 0.01. Scirocco also has the option of using flow direction to determine the condition used. In the inflow case, $\tilde{\nu}$ is set to a free-stream value, typically 0.01. If an outflow condition is found, a zeroth-order extrapolation from the interior is used, giving zero gradient in the turbulence quantity.

At interior block boundaries, the user may choose between a transparent interior condition, or an average condition. The former uses the same equation that is used on interior nodes. The latter averages the value of $\tilde{\nu}$ on either side of the boundary.

Chapter 3

Spatial Discretization

In order to approximate the solution of the continuous system formed by the Navier-Stokes equations (2.12) and the Spalart-Allmaras turbulence model (2.27), they can be converted to a system of ordinary difference equations. This is a two-step process. This chapter discusses the first step where the spatial derivatives are approximated by a difference operator. The Jacobian, or derivative of the spatially discretized equations with respect to the dependent variables, is required by the quasi-Newton method, as discussed in Chapter 4.

3.1 Navier-Stokes Equations

The method of spatial discretization of the mean-flow equations follows that of ARC2D, the implicit finite-difference solver developed by Steger [11] and Pulliam [55], with the option of using either the scalar dissipation scheme of Jameson et al. [60] or the matrix dissipation scheme of Swanson and Turkel [61].

3.1.1 Inviscid Fluxes

The derivatives of the inviscid fluxes in the ξ direction are approximated as

$$\left(\frac{\partial \hat{E}}{\partial \xi}\right) \approx \delta_\xi \hat{E}_{j,k} - \nabla_\xi d_{j+\frac{1}{2},k} \tag{3.1}$$

The first term is the second-order centred difference operator

$$\delta_\xi \hat{E}_{j,k} = \frac{\hat{E}_{j+1,k} - \hat{E}_{j-1,k}}{2} \tag{3.2}$$

The second term is the artificial dissipation. In the scalar dissipation case:

$$d_{j+\frac{1}{2},k} = \sigma_{j+\frac{1}{2},k} J^{-1}_{j+\frac{1}{2},k} \left(\epsilon^{(2)}_{j+\frac{1}{2},k} \Delta_\xi J_{j,k} \hat{Q}_{j,k} - \epsilon^{(4)}_{j+\frac{1}{2},k} \Delta_\xi \nabla_\xi \Delta_\xi J_{j,k} \hat{Q}_{j,k} \right) \tag{3.3}$$

Δ_ξ and ∇_ξ are the first-order forward and backward difference operators

$$\begin{aligned} \Delta_\xi q_{j,k} &= q_{j+1,k} - q_{j,k} \\ \nabla_\xi q_{j,k} &= q_{j,k} - q_{j-1,k} \end{aligned} \tag{3.4}$$

The dissipation factor is found by

$$\sigma = |U| + a\sqrt{\xi_x + \xi_y} \tag{3.5}$$

Following the calculation used in ARC2d and TORNADO, the first-order dissipation varies with the direction. In the ξ direction:

$$\begin{aligned} \Upsilon_{j,k} &= \frac{|p_{j+1,k} - 2p_{j,k} + p_{j-1,k}|}{|p_{j+1,k} + 2p_{j,k} + p_{j-1,k}|} \\ \Upsilon'_{j,k} &= \max(\Upsilon_{j+1,k}, \Upsilon_{j,k}, \Upsilon_{j-1,k}) \\ \epsilon^{(2)}_{j,k} &= \kappa_2 \left(\frac{1}{4}\Upsilon'_{j-1,k} + \frac{1}{2}\Upsilon'_{j,k} + \frac{1}{4}\Upsilon'_{j+1,k} \right) \end{aligned} \tag{3.6}$$

In the η direction, the second stage is skipped:

$$\begin{aligned} \Upsilon_{j,k} &= \frac{|p_{j,k+1} - 2p_{j,k} + p_{j,k-1}|}{|p_{j,k+1} + 2p_{j,k} + p_{j,k-1}|} \\ \epsilon^{(2)}_{j,k} &= \kappa_2 \left(\frac{1}{4}\Upsilon_{j,k-1} + \frac{1}{2}\Upsilon_{j,k} + \frac{1}{4}\Upsilon_{j,k+1} \right) \end{aligned} \tag{3.7}$$

Υ is a pressure switch to control first-order dissipation near shocks. In both directions, the third-order dissipation coefficient is found from

$$\epsilon^{(4)}_{j,k} = \max(0, \kappa_4 - \epsilon^{(2)}_{j,k}) \tag{3.8}$$

κ_2 and κ_4 are the dissipation constants. Typically, $\kappa_4 = 0.02$, and $\kappa_2 = 0$ for subsonic flows; $\kappa_2 = 1.0$ for transonic flows.

When the matrix dissipation model is used, equation 3.3 is replaced with

$$d_{j+\frac{1}{2},k} = |\hat{A}|_{j+\frac{1}{2},k} J^{-1}_{j+\frac{1}{2},k} \left(\epsilon^{(2)}_{j+\frac{1}{2},k} \Delta_\xi J_{j,k} \hat{Q}_{j,k} - \epsilon^{(4)}_{j+\frac{1}{2},k} \Delta_\xi \nabla_\xi \Delta_\xi J_{j,k} \hat{Q}_{j,k} \right) \tag{3.9}$$

which is identical except for the scalar factor σ being replaced with the matrix $|\hat{A}|$. This is formed from the eigenvalue diagonalization of the local Jacobian block, $\hat{A} = \frac{\partial \hat{E}}{\partial \xi}$:

$$|\hat{A}| = T_\xi |\Lambda_\xi| T_\xi^{-1} \tag{3.10}$$

where $|\Lambda_\xi|$ is the matrix of the absolute values of the eigenvalues of the local flux Jacobian block:

$$|\Lambda_\xi| = \begin{bmatrix} |\lambda_1| & 0 & 0 & 0 \\ 0 & |\lambda_2| & 0 & 0 \\ 0 & 0 & |\lambda_3| & 0 \\ 0 & 0 & 0 & |\lambda_4| \end{bmatrix} = \begin{bmatrix} |U| & 0 & 0 & 0 \\ 0 & |U| & 0 & 0 \\ 0 & 0 & |U + a\sqrt{\xi_x^2 + \xi_x^2}| & 0 \\ 0 & 0 & 0 & |U - a\sqrt{\xi_x^2 + \xi_x^2}| \end{bmatrix} \tag{3.11}$$

and T_ξ is the eigenvector matrix. We use a simple average when evaluating $|\hat{A}|$ at half-nodes $|\hat{A}|_{j+\frac{1}{2},k} = \frac{1}{2}(|\hat{A}|_{j,k} + |\hat{A}|_{j+1,k})$. To avoid zero eigenvalues, the elements of $|\Lambda|$ are modified as follows:

$$\begin{aligned} \tilde{\lambda}_1, \tilde{\lambda}_2 &= \max(\lambda_{1,2}, V_l\sigma) \\ \tilde{\lambda}_3 &= \max(\lambda_3, V_n\sigma) \\ \tilde{\lambda}_4 &= \max(\lambda_4, V_n\sigma) \end{aligned} \tag{3.12}$$

Typical values are $V_l = V_n = 0.025$ for subsonic cases, and $V_l = 0.025$ and $V_n = 0.25$ for transonic cases.

Jacobian of Inviscid Fluxes

Here we seek

$$\frac{\partial}{\partial \hat{Q}} \left(\delta_\xi \hat{E}_{j,k} - \Delta_\xi d_{j+\frac{1}{2},k} \right) \tag{3.13}$$

The first term is reasonably simple

$$\frac{\partial}{\partial \hat{Q}} \left(\delta_\xi \hat{E}_{j,k} \right) = \frac{1}{2} \left[\frac{\partial \hat{E}_{j+1,k}}{\partial \hat{Q}_{j+1,k}} - \frac{\partial \hat{E}_{j-1,k}}{\partial \hat{Q}_{j-1,k}} \right] \tag{3.14}$$

The flux Jacobian is

$$\frac{\partial \hat{E}}{\partial \hat{Q}} = \begin{bmatrix} 0 & \varsigma_x & \varsigma_y & 0 \\ -u\theta + \varsigma_x\phi^2 & \theta - (\gamma-2)\varsigma_x u & \varsigma_y u - (\gamma-1)\varsigma_x v & (\gamma-1)\varsigma_x \\ -v\theta + \varsigma_y\phi^2 & \varsigma_x v - (\gamma-1)\varsigma_y u & \theta - (\gamma-2)\varsigma_y v & (\gamma-1)\varsigma_y \\ \theta(\phi^2 - a_1) & \varsigma_x a_1 - (\gamma-1)u\theta & \varsigma_y a_1 - (\gamma-1)v\theta & \gamma\theta \end{bmatrix} \tag{3.15}$$

with

$$\begin{aligned} a_1 &= \gamma\frac{e}{\rho} - \theta^2 \\ \theta &= \varsigma_x u + \varsigma_y v \\ \phi^2 &= \frac{1}{2}(\gamma-1)(u^2+v^2) \end{aligned} \tag{3.16}$$

where ς is a place-holder for the computational direction, either ξ or η.

The expansion of the dissipation term gives a five-point stencil in each direction. The spectral radius or $|\hat{A}|$, and the pressure switch, Υ, are treated as constant.

3.1.2 Viscous Fluxes

The viscous contribution is in the form

$$\partial_\eta(\alpha_{j,k}\partial_\eta\beta_{j,k}) \tag{3.17}$$

$\partial_\eta\beta_{j,k}$ is approximated by a centred difference at half-nodes. The second derivative is approximated by a centred difference at the grid node using the half-node values. Equation 3.17 is then approximated by

$$\nabla_\eta(\alpha_{j,k+\frac{1}{2}}\Delta_\eta\beta_{j,k}) = \alpha_{j,k+\frac{1}{2}}(\beta_{j,k+1} - \beta_{j,k}) - \alpha_{j,k-\frac{1}{2}}(\beta_{j,k} - \beta_{j,k-1}) \tag{3.18}$$

Values of α at the half-nodes are computed by averaging over the bracketing grid nodes

$$\alpha_{j,k+\frac{1}{2}} = \frac{\alpha_{j,k+1} + \alpha_{j,k}}{2} \tag{3.19}$$

Jacobian of Viscous Fluxes

The viscous fluxes are differentiated the same way as the inviscid fluxes. The flux Jacobian is

$$\frac{\partial \hat{P}}{\partial \hat{Q}} = J^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1 \partial_\eta(\rho^{-1}) & \alpha_2 \partial_\eta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2 \partial_\eta(\rho^{-1}) & \alpha_3 \partial_\eta(\rho^{-1}) & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} J \tag{3.20}$$

with

$$\begin{aligned} m_{21} &= -\alpha_1 \partial(u/\rho) - \alpha_2 \partial(v/\rho) \\ m_{31} &= -\alpha_2 \partial(u/\rho) - \alpha_3 \partial(v/\rho) \\ m_{41} &= \alpha_4 \partial_\eta[-(e/\rho^2) + (u^2+v^2)/\rho] - \alpha_1 \partial_\eta(u^2/\rho) \\ &\quad -2\alpha_2 \partial_\eta(uv/\rho) - \alpha_3 \partial_\eta(v^2/\rho) \\ m_{42} &= -\alpha_4 \partial_\eta(u/\rho) - m_{21} \\ m_{43} &= -\alpha_4 \partial_\eta(v/\rho) - m_{31} \\ m_{44} &= \alpha_4 \partial_\eta(\rho^{-1}) \\ \alpha_1 &= (\mu + \mu_t)\left[(4/3)\eta_x^2 + \eta_y^2\right] \\ \alpha_2 &= \frac{1}{3}(\mu + \mu_t)\eta_x \eta_y \\ \alpha_3 &= (\mu + \mu_t)\left[\eta_x^2 + (4/3)\eta_y^2\right] \\ \alpha_4 &= \gamma\left(\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1}\right)\left(\eta_x^2 + \eta_y^2\right) \end{aligned} \tag{3.21}$$

3.1.3 Boundary Conditions

Before giving details regarding implementation of the boundary conditions, it is useful to define how the normal and tangential velocities are calculated.

$$V_n = \begin{cases} -(\bar{\varsigma}_x u + \bar{\varsigma}_y v) & \text{at low boundary } (\bar{\varsigma} = \bar{\varsigma}_{min}) \\ \bar{\varsigma}_x u + \bar{\varsigma}_y v & \text{at high boundary } (\bar{\varsigma} = \bar{\varsigma}_{max}) \end{cases} \tag{3.22}$$

$$V_t = \bar{\varsigma}_y u - \bar{\varsigma}_x v \tag{3.23}$$

A low boundary has a computational index of zero. $\bar{\varsigma}_x$ and $\bar{\varsigma}_y$ are the normalized metrics of the curvilinear coordinate transformation, in the computational direction perpendicular to the boundary

$$\bar{\varsigma}_x = \frac{\varsigma_x}{\sqrt{\varsigma_x^2 + \varsigma_y^2}} \tag{3.24}$$

The linearization of the boundaries is simplified by differentiating the boundary equations with respect to the primitive variables and then applying a transformation to obtain the derivative with respect to the conservative variables. The discretized equations can be written as

$$\mathcal{B}(R) = 0 \tag{3.25}$$

with the primitive variables

$$R = \begin{bmatrix} \rho \\ u \\ v \\ p \end{bmatrix} \tag{3.26}$$

The Jacobian of the boundary equations is then

$$\frac{\partial \mathcal{B}}{\partial \hat{Q}} = \left(\frac{\partial \mathcal{B}}{\partial R}\right)\left(\frac{\partial R}{\partial \hat{Q}}\right) \tag{3.27}$$

The Jacobian of the transformation is

$$\frac{\partial R}{\partial \hat{Q}} = J^{-1}\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 \\ (\gamma-1)\frac{u^2+v^2}{2} & -(\gamma-1)u & -(\gamma-1)v & \gamma-1 \end{bmatrix} \tag{3.28}$$

Free-Stream

Consider first the Riemann condition, with inflow. R_1, V_t, and S are set from free-stream conditions, while R_2 is set to a zeroth order extrapolation from the interior:

$$\left(\bar{\varsigma}_x u + \bar{\varsigma}_v - \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_b - \left(V_n - \frac{2a}{\gamma-1}\right)_\infty = 0 \tag{3.29}$$

$$\left(\bar{\varsigma}_x u + \bar{\varsigma}_v + \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_b - \left(\bar{\varsigma}_x u + \bar{\varsigma}_v + \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_i = 0 \tag{3.30}$$

$$\left(\frac{\rho^\gamma}{p}\right)_b - S_\infty = 0 \tag{3.31}$$

$$\left(\bar{\eta}_y u - \bar{\eta}_x v\right)_b - (V_t)_\infty = 0 \tag{3.32}$$

When an outflow situation exists,

$$\left(\bar{\varsigma}_x u + \bar{\varsigma}_v - \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_b - \left(\bar{\varsigma}_x u + \bar{\varsigma}_v - \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_i = 0 \tag{3.33}$$

$$\left(\bar{\varsigma}_x u + \bar{\varsigma}_v + \frac{2}{\gamma-1}\sqrt{\frac{\gamma p}{\rho}}\right)_b - \left(V_n + \frac{2a}{\gamma-1}\right)_\infty = 0 \tag{3.34}$$

$$\left(\frac{\rho^\gamma}{p}\right)_b - \left(\frac{\rho^\gamma}{p}\right)_i = 0 \tag{3.35}$$

$$\left(\bar{\eta}_y u - \bar{\eta}_x v\right)_b - \left(\bar{\eta}_y u - \bar{\eta}_x v\right)_i = 0 \tag{3.36}$$

where the subscripts b and i indicate values at the boundary and first interior node, respectively.

For viscous flows, the downstream boundary condition is zeroth-order extrapolation of ρ, ρu, ρv, and p

$$\rho_b - \rho_i = 0 \tag{3.37}$$

$$(\rho u)_b - (\rho u)_i = 0 \tag{3.38}$$

$$(\rho v)_b - (\rho v)_i = 0 \tag{3.39}$$

$$p_b - p_i = 0 \tag{3.40}$$

When an inflow situation exists, the Jacobian of the equations with respect to the primitive variables is

$$\frac{\partial \mathcal{B}}{\partial R} = \begin{bmatrix} \frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{-\gamma}{(\gamma-1)a\rho} \\ -\frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{\gamma}{(\gamma-1)a\rho} \\ \frac{\gamma\rho^{\gamma-1}}{p} & 0 & 0 & \frac{-\rho^\gamma}{p^2} \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \end{bmatrix}_b, \begin{bmatrix} \frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{-\gamma}{(\gamma-1)a\rho} \\ -\frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{\gamma}{(\gamma-1)a\rho} \\ \frac{\gamma\rho^{\gamma-1}}{p} & 0 & 0 & \frac{-\rho^\gamma}{p^2} \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \end{bmatrix}_i \tag{3.41}$$

where the subscript b indicates the Jacobian at the boundary node, and i indicates the first offwall node. In the outflow case

$$\frac{\partial \mathcal{B}}{\partial R} = \begin{bmatrix} \frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{-\gamma}{(\gamma-1)a\rho} \\ -\frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{\gamma}{(\gamma-1)a\rho} \\ \frac{\gamma\rho^{\gamma-1}}{p} & 0 & 0 & \frac{-\rho^\gamma}{p^2} \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \end{bmatrix}_b, \begin{bmatrix} \frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{-\gamma}{(\gamma-1)a\rho} \\ -\frac{a}{(\gamma-1)\rho} & \bar{\varsigma}_x & \bar{\varsigma}_y & \frac{\gamma}{(\gamma-1)a\rho} \\ \frac{\gamma\rho^{\gamma-1}}{p} & 0 & 0 & \frac{-\rho^\gamma}{p^2} \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \end{bmatrix}_i \tag{3.42}$$

For the viscous boundaries downstream of the body, the Jacobian of the zeroth-order extrapolation is

$$\frac{\partial \mathcal{B}}{\partial R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_b, - \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_i \tag{3.43}$$

Solid Wall

For inviscid flows, we use first-order extrapolation from the interior for the tangential velocity and pressure. The four slip equations are then

$$(\bar{\varsigma}_x u + \bar{\varsigma}_y v)_b = 0 \tag{3.44}$$

$$(\bar{\varsigma}_y u - \bar{\varsigma}_x v)_b - 2(\bar{\varsigma}_y u - \bar{\varsigma}_x v)_i + (\bar{\varsigma}_y u - \bar{\varsigma}_x v)_{ii} = 0 \tag{3.45}$$

$$p_b - 2p_i + p_{ii} = 0 \tag{3.46}$$

$$\left[\frac{\gamma}{\gamma-1}p + \frac{1}{2}\rho(u^2+v^2) - \rho H_\infty\right]_b = 0 \tag{3.47}$$

The subscripts b, i, and ii indicate the boundary node, first offwall, and the second offwall nodes respectively.

For viscous flows, no-slip is enforced, along with zeroth order extrapolation of density and pressure

$$\rho_b - \rho_i = 0 \tag{3.48}$$

$$\rho u_b = 0 \tag{3.49}$$

$$\rho v_b = 0 \tag{3.50}$$

$$p_b - p_i = 0 \tag{3.51}$$

The Jacobian of the slip equations at the boundary node is

$$\frac{\partial \mathcal{B}}{\partial R} = \begin{bmatrix} 0 & \bar{\varsigma}_x & \bar{\varsigma}_y & 0 \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2}(u^2+v^2) - H_\infty & \rho u & \rho v & \gamma(\gamma-1)^{-1} \end{bmatrix}_b \tag{3.52}$$

At the offwall nodes, the Jacobian is

$$\frac{\partial \mathcal{B}}{\partial R} = -2 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}_i, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \bar{\varsigma}_y & -\bar{\varsigma}_x & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{ii} \tag{3.53}$$

The Jacobian of the no-slip equations is

$$\frac{\partial \mathcal{B}}{\partial R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_b, \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}_i \tag{3.54}$$

Order of Boundary Condition Equations

The order of the boundary equation within each block is not defined. Pueyo and Zingg [27] discovered (though have not reported) that there is an advantage to carefully selecting the ordering. It is possible with the Riemann and inviscid solid wall (slip) boundary conditions to get zeros or very small values as the diagonal elements. It is a simple matter to reorder equations (only within each node) to maximize the diagonal elements. The best row mapping is determined only at the first iteration. This is akin to partial pivoting.

3.2 Spalart-Allmaras Turbulence Model

The convection terms of the turbulence model are first transformed into curvilinear coordinates

$$\vec{u} \cdot \nabla \tilde{\nu} = U \frac{\partial \tilde{\nu}}{\partial \xi} + V \frac{\partial \tilde{\nu}}{\partial \eta} \tag{3.55}$$

and are then approximately differentiated with a first-order upwind difference, based on the contravariant velocity

$$U \frac{\partial \tilde{\nu}}{\partial \xi} \approx U_i^+ (\tilde{\nu}_i - \tilde{\nu}_{i-1}) + U_i^- (\tilde{\nu}_{i+1} - \tilde{\nu}_i) \tag{3.56}$$

with

$$U_i^+ = \frac{1}{2}(U_i + |U_i|), \qquad U_i^- = \frac{1}{2}(U_i - |U_i|) \tag{3.57}$$

The cross terms are dropped after the diffusion terms are transformed to curvilinear coordinates, giving

$$\begin{aligned} \nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] &\approx \partial_\xi[(\xi_x + \xi_y)(\nu + \tilde{\nu})(\xi_x + \xi_y)\partial_\xi(\tilde{\nu})] + \\ &\quad (\eta_x + \eta_y)\partial_\eta[(\nu + \tilde{\nu})(\eta_x + \eta_y)\partial_\eta(\tilde{\nu})] \\ \nabla^2\tilde{\nu} &\approx (\xi_x + \xi_y)\partial_\xi[(\xi_x + \xi_y)\partial_\xi(\tilde{\nu})] + \\ &\quad (\eta_x + \eta_y)\partial_\eta[(\eta_x + \eta_y)\partial_\eta(\tilde{\nu})] \end{aligned} \tag{3.58}$$

The derivatives are approximated by centred differences, using half-nodes for the inner derivatives

$$\begin{aligned} &(\xi_x + \xi_y)\partial_\xi[(\nu + \tilde{\nu})(\xi_x + \xi_y)\partial_\xi(\tilde{\nu})] \approx \\ &(\xi_x + \xi_y)_{j,k}[(\nu + \tilde{\nu})_{j+\frac{1}{2}}(\xi_x + \xi_y)_{j+\frac{1}{2},k}(\tilde{\nu}_{j+1,k} - \tilde{\nu}_{j,k}) - (\nu + \tilde{\nu})_{j-\frac{1}{2}}(\xi_x + \xi_y)_{j-\frac{1}{2},k}(\tilde{\nu}_{j,k} - \tilde{\nu}_{j-1,k})] \end{aligned} \tag{3.59}$$

$$\begin{aligned} &(\xi_x + \xi_y)\partial_\xi[(\xi_x + \xi_y)\partial_\xi(\tilde{\nu})] = \\ &(\xi_x + \xi_y)_{j,k}[(\xi_x + \xi_y)_{j+\frac{1}{2},k}(\tilde{\nu}_{j+1,k} - \tilde{\nu}_{j,k}) - (\xi_x + \xi_y)_{j-\frac{1}{2},k}(\tilde{\nu}_{j,k} - \tilde{\nu}_{j-1,k})] \end{aligned} \tag{3.60}$$

The half-node values are taken as averages

$$(\nu + \tilde{\nu})_{j+\frac{1}{2}} = \frac{1}{2}[(\nu + \tilde{\nu})_{j+1,k} + (\nu + \tilde{\nu})_{j,k}] \tag{3.61}$$

The vorticity S is also transformed, then centred differences are applied

$$\begin{aligned} S &= \left|\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right| \\ &\approx \left|\frac{1}{2}[\xi_x(v_{j+1,k} - v_{j-1,k}) + \eta_x(v_{j,k+1} - v_{j,k-1}) - \xi_y(u_{j+1,k} - u_{j-1,k}) - \eta_y(u_{j,k+1} - u_{j,k-1})]\right| \end{aligned} \tag{3.62}$$

Jacobian of Turbulence Model

The differentiation of the turbulence equation with respect to $\tilde{\nu}$ is reasonably straightforward, if somewhat tedious. All terms are analytically differentiated, including those with respect to the conservative flow variables, $\hat{Q}$. However, following Spalart and Allmaras [14], we modify the Jacobian during start-up. Start-up issues are discussed in Section 5.3. The modifications made are such that the new Jacobian matrix is M-type.[1] This, along with ensuring that the residual vector is positive, gives positive updates of the turbulence variable, in the Lagrangian frame of reference. This means the local (Eulerian) turbulence quantity can be decreased through convection, but cannot drop below zero. The derivatives with respect to $\hat{Q}$ are not included during start-up, since it is difficult to enforce the M-matrix property of these terms.

Spalart and Allmaras showed that the advection and diffusion terms satisfy positivity constraints. However, the derivatives of the production and destruction terms do not. The terms are

$$\begin{aligned} P(\tilde{\nu})\tilde{\nu} &= c_{b1}[1 - f_{t2}]\tilde{S}\tilde{\nu} \\ D(\tilde{\nu})\tilde{\nu} &= \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left[\frac{\tilde{\nu}}{d}\right]^2 \end{aligned} \tag{3.63}$$

[1] A square matrix is of M-type if it is diagonally dominant, all off-diagonal entries are non-positive, and all diagonal entries are positive.

The unmodified derivatives are

$$\begin{aligned} [P(\tilde{\nu})\tilde{\nu}]' &= P(\tilde{\nu}) + P'(\tilde{\nu})\tilde{\nu} \\ [D(\tilde{\nu})\tilde{\nu}]' &= D(\tilde{\nu}) + D'(\tilde{\nu})\tilde{\nu} \end{aligned} \tag{3.64}$$

The terms are considered together, so that the modified entry into the Jacobian becomes

$$\bar{D} - \bar{P} = \text{pos}(D - P) + \text{pos}(D' - P')\tilde{\nu} \tag{3.65}$$

where the $\text{pos}(x)$ function is defined as

$$\text{pos}(x) = \begin{cases} x & \text{if} \quad x \geq 0 \\ 0 & \text{if} \quad x < 0 \end{cases} \tag{3.66}$$

Chapter 4

Iteration to Steady-State

In the preceding chapter, we discussed forming the system of nonlinear equations which we wish to solve. These were then discretized in space. This chapter gives techniques used to find the (steady state) solution of these discretized equations. Ideally, this system could be solved directly with Newton's method, and all of the time derivatives could be ignored. Unfortunately, except in the simplest cases, the nonlinearity of the equations will result in divergence. To address this problem, an implicit Euler method is used to advance the equations in time. Newton's method is approached as the time step used is increased as the equations near steady state. This requires including and discretizing the time components in the spatially discretized system.

At each step of this method, a linear matrix equation is formed. The large, sparse systems found in typical cases are ill-conditioned, and not at all diagonally-dominant. The size of the system requires the use of some sort of approximate linear solver, such as GMRES, instead of a direct solver. Preconditioning is needed to handle the poor conditioning of the Jacobian matrix. Details for implementation of these methods is given in this chapter.

4.1 The Nonlinear System

The nonlinear system is a mix of the interior and boundary equations for the Navier-Stokes and turbulence model equations. They can be cast in the form

$$\mathcal{T}(\hat{\mathcal{Q}}) = \mathcal{F}(\hat{\mathcal{Q}}) \tag{4.1}$$

where $\hat{\mathcal{Q}}$ is the block vector with each block being the values of $\hat{Q}$ at each node. $\mathcal{T}$ is a vector with each element containing either zero, if the associated equation results from a boundary, or $\frac{d\hat{Q}}{dt}$ in the case of an interior equation. $\mathcal{F}(\hat{\mathcal{Q}})$ are the discretized equations from the previous chapter.

To discretize in time, first apply the implicit Euler method

$$\frac{\hat{\mathcal{Q}}^{(n+1)} - \hat{\mathcal{Q}}^{(n)}}{\Delta t} = \mathcal{F}^{(n+1)} \tag{4.2}$$

then linearize the right hand side

$$\mathcal{F}^{(n+1)} \approx \mathcal{F}^{(n)} + \mathcal{A}^{(n)} \Delta\hat{\mathcal{Q}}^{(n)} \tag{4.3}$$

where we have defined

$$\mathcal{F}^{(n)} \equiv \mathcal{F}(\hat{\mathcal{Q}}^{(n)}) \quad \text{and} \quad \Delta\hat{\mathcal{Q}}^{(n)} \equiv \hat{\mathcal{Q}}^{(n+1)} - \hat{\mathcal{Q}}^{(n)} \tag{4.4}$$

$\mathcal{A}^{(n)}$ is the Jacobian matrix of $\mathcal{F}$:

$$\mathcal{A} = \frac{\partial \mathcal{F}(\hat{\mathcal{Q}})}{\partial \hat{\mathcal{Q}}} \tag{4.5}$$

evaluated at $\hat{\mathcal{Q}}^{(n)}$. Δt is a vector, where each element is the length of the time step taken at the corresponding node. The entry is infinite at boundary nodes. The linear system formed at each step is then

$$(\mathcal{A}^{(n)} - \mathcal{I}/\Delta t)\Delta\hat{\mathcal{Q}}^{(n)} = -\mathcal{F}^{(n)} \tag{4.6}$$

$\mathcal{I}$ is a matrix with all zero entries, except for a one on the diagonal for interior equations. Note that as $\Delta t \to \infty$, Newton's method is recovered. We also allow different values of Δt for each interior node.

4.2 The Linear System

The linear system in equation 4.6 is very sparse, and typically quite large. This makes the use of a direct solver prohibitively expensive in terms of both memory requirements and processor time. Nonstationary iterative linear solvers such as GMRES, BiCGSTAB, and CGS, are much more attractive options. Barrett, et al. [26] provide an overview of each of the methods. Following Pueyo and Zingg [27], we choose GMRES [25].

4.2.1 GMRES

Given a linear system of the form

$$\mathcal{A}x = b \tag{4.7}$$

GMRES finds the iterate $x_m \in x_0 + K_m$ that minimizes the L_2 norm of the residual $r_m = b - \mathcal{A}x_m$. The vector x_0 is the initial guess, and K_m is the Krylov subspace

$$K_m = \text{span}\{v_1, \mathcal{A}v_1, \mathcal{A}^2 v_1, \ldots, \mathcal{A}^{m-1} v_1\} \tag{4.8}$$

v_1 is the normalized initial residual

$$v_1 = \frac{r_0}{\|r_0\|_2} = \frac{b - \mathcal{A}x_0}{\|b - \mathcal{A}x_0\|_2} \tag{4.9}$$

In our application, the initial guess, x_0 is taken to be zero. This is done because there is no reason to expect subsequent outer iterations to have a similar update.

GMRES begins by calculating the vector v_1. Then, an orthonormal basis is formed from the Krylov space using Arnoldi's method

$$\begin{array}{lll} 1. & \text{For} & j = 1, 2, \ldots, j \\ 2. & & h_{i,j} = (\mathcal{A}v_j, v_i),\ i = 1, 2, \ldots, j \\ 3. & & \hat{v}_{j+1} = \mathcal{A}v_j - \sum_{i=1}^{j} h_{i,j} v_i \\ 4. & & h_{j+1,j} = \|\hat{v}_{j+1}\| \\ 5. & & v_{j+1} = \hat{v}_{j+1}/h_{j+1,j} \\ 6. & \text{end} & \end{array} \tag{4.10}$$

From lines 3, 4, and 5 we can see that

$$\mathcal{A}V_m = V_{m+1}\bar{H}_m \tag{4.11}$$

where V_m is an $N \times m$ matrix with the orthonormal basis column vectors $v_1, \ldots, v_m$ and $\bar{H}_m$ is an $(m+1) \times m$ Hessenberg matrix with the entries being the $h_{i,j}$ coefficients calculated in line 2. N is the size of the of square matrix $\mathcal{A}$.

The next step is to minimize the residual vector. Any vector x in the space $\{x_0 + K_m\}$ can be written as

$$x = x_0 + V_m y \tag{4.12}$$

Considering the L_2 norm of the residual vector and applying equations 4.11 and 4.12 gives

$$\begin{aligned} \|r(y)\|_2 &= \|b - \mathcal{A}x\|_2 \\ &= \|b - \mathcal{A}(x_0 + V_m y)\|_2 \\ &= \|r_0 - \mathcal{A}V_m y\|_2 \\ &= \|\beta v_1 - V_{m+1}\bar{H}_m y\|_2 \\ &= \|V_{m+1}(\beta e_1 - \bar{H}_m y)\|_2 \end{aligned} \tag{4.13}$$

where $\beta = \|r_0\|$ and e_1 is the first column of the $m \times m$ identity matrix. The column-vectors of V_{m+1} are orthonormal, so that

$$\|r(y)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2 \tag{4.14}$$

We now have a form which can be easily minimized. This is an $(m+1) \times m$ least-squares problem with $m \ll N$. The structure of the Hessenberg matrix is such that simple rotations transform it into an upper triangular matrix. Once we have y_m, which minimizes $||r(y)||_2$, we find x_m from equation 4.12.

GMRES has the property of having the L_2 norm of the residual available at each step, without explicitly calculating either the approximate solution vector x_j or the corresponding residual $r_j = b - \mathcal{A}x_j$. This is done by applying the rotations to the vector βe_1 after each search direction is added. If the vector resulting from rotating βe_1 is $(\gamma_1, \ldots, \gamma_{j+1})^T$, then it follows that $\|r_j\|_2 = |\gamma_{j+1}|$. This is a direct result of the least squares minimization.

1. Given x_0, compute $r_0 = b - \mathcal{A}x_0$ and $v_1 = r_0/\|r_0\|_2$
2. For $j = 1, \ldots, m$

   $w_j = \mathcal{A}v_j$
   $h_{i,j} = (w_j, v_i)\ i = 1, 2, \ldots, j$
   $\hat{v}_{j+1} = w_j - \sum_{i=1}^{j} h_{i,j} v_j$
   $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$
   Perform rotation on $\bar{H}_j$ and rhs vector.
   If $\|r_j\|_2$ is small enough exit

3. Form approximate solution

   Solve for y_m
   Form $x_m = x_0 + V_m y_m$

4. Restart

   Compute $r_m = b - \mathcal{A}x_m$
   If $\|r_j\|_2$ is small enough stop
   Set $x_0 \leftarrow x_m$ and go to 1

Figure 4.1: Algorithm for restarted GMRES

Restarting

As the dimension of the Krylov subspace, represented by m, increases, the storage increases linearly and computational time quadratically. This becomes prohibitive long before m approaches N. The algorithm must be stopped for some value of $m \ll N$. The solution at that point may not be acceptably converged. The algorithm may be restarted, using the previous solution as the initial guess. This restarted version is denoted GMRES(m). If $\mathcal{A}$ is nearly positive definite, GMRES(m) will converge for reasonably small m. However, the systems being considered here are typically far from positive definite. Use of a preconditioner, discussed in Section 4.2.2, helps to alleviate this problem. The size of m is discussed in Chapter 5. Figure 4.1 gives pseudocode for restarted GMRES. An initial guess of $x_0 = 0$ is always used.

Matrix-free GMRES

GMRES requires only matrix-vector products. In our case, the matrix in question also is a Jacobian matrix. This means that $\mathcal{A}$ does not need to be explicitly formed, but instead finite-differencing can be used to obtain an approximation to the matrix-vector product. This has the advantage that the matrix does not need to be stored (although a reduced form is required for the preconditioners which we use), as well as possibly saving computational time. Terms which are difficult to linearize, such as the dissipation coefficients, the switch between second and fourth-difference dissipation, and the use of far-field circulation correction, are also inherently included in the approximation.

We normally use a forward-difference approximation which requires only one extra right-hand side

evaluation, instead of a centred-difference, which requires two. The approximation is then

$$\mathcal{A}v \approx \frac{\mathcal{F}(\hat{\mathcal{Q}} + \varepsilon v) - \mathcal{F}(\hat{\mathcal{Q}})}{\varepsilon} \tag{4.15}$$

where ε is a small scalar used to perturb $\hat{\mathcal{Q}}$ in the direction of v. Nielsen et al. [53] chose ε as

$$\varepsilon \|v\|_2 \approx \sqrt{\varepsilon_m} \tag{4.16}$$

where ε_m is the value of machine zero. In Section 5.7 we discuss a small modification to this method.

Occasionally, a higher-order approximation may be needed, as discussed in Section 5.7. In these instances, a central difference is used to get a second order approximation

$$\mathcal{A}v \approx \frac{\mathcal{F}(\hat{\mathcal{Q}} + \varepsilon v) - \mathcal{F}(\hat{\mathcal{Q}} - \varepsilon)}{2\varepsilon} \tag{4.17}$$

This approximation is only valid for a Jacobian matrix. Equation 4.6 requires $\left(\mathcal{A} - \frac{\mathcal{I}}{\Delta t}\right)\Delta\hat{\mathcal{Q}}$ which includes an extra term. To include this with the matrix-free method, expand the expression

$$\mathcal{A}\Delta\hat{\mathcal{Q}} - \frac{\mathcal{I}}{\Delta t}\Delta\hat{\mathcal{Q}} \tag{4.18}$$

so that after the first term is found with the matrix-free method, the second term can be added to the resulting vector.

4.2.2 Preconditioning

The rate of convergence of a Krylov method is strongly dependent on how well the matrix is conditioned. The matrices resulting from the discretized Navier-Stokes equations are very ill-conditioned for typical grids. The conditioning is usually so bad that the Krylov method will not converge in a reasonable number of iterations. The efficiency for these systems is thus enormously improved by the use of a good preconditioner. Figure 4.2 shows the eigenvalues from the Jacobian matrix of a simple channel flow without preconditioning and with an ILU(4) preconditioner. The clustering of the eigenvalues is much tighter in the latter case, resulting in the total time required to converge being roughly 1000 times faster with preconditioning.

Preconditioning involves transforming the system by either pre- or post-multiplying by a matrix which approximates $\mathcal{A}$

$$\mathcal{M}_l^{-1}\mathcal{A}x = \mathcal{M}_l^{-1}b \tag{4.19}$$

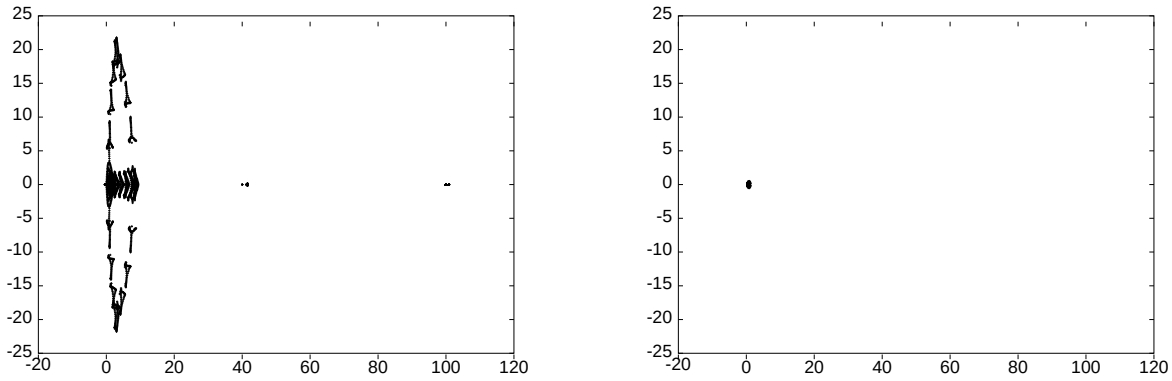
or

$$\mathcal{A}\mathcal{M}_r^{-1}\mathcal{M}_r x = b \tag{4.20}$$

In either case, we desire $\mathcal{M}^{-1}$ to be as close to $\mathcal{A}^{-1}$ as possible. Of course, the final solution is unaffected by the preconditioner. In the former (left preconditioning) case, the residual of the system is changed, which affects the stopping criterion of the method. Right preconditioning does not affect the residual. We use right preconditioning for this reason.

The preconditioned GMRES algorithm is given in Figure 4.3. Note that we do not explicitly form $\mathcal{A}\mathcal{M}^{-1}$, but instead form an intermediate vector z_j, which is significantly more efficient. This vector is found by solving

$$\mathcal{M}z_j = v_j \tag{4.21}$$



(a) No preconditioner

(b) ILU(4) preconditioner

Figure 4.2: Eigenvalues of Jacobian matrix

1. Given x_0, compute $r_0 = b - \mathcal{A}x_0$ and $v_1 = r_0/\|r_0\|_2$
2. For $j = 1, \ldots, m$

   Precondition: $z_j = \mathcal{M}^{-1} v_j$
   $w_j = \mathcal{A} z_j$
   $h_{i,j} = (w_j, v_i)\ i = 1, 2, \ldots, j$
   $\hat{v}_{j+1} = w_j - \sum_{i=1}^{j} h_{i,j} v_j$
   $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$
   Perform rotation on $\bar{H}_j$ and rhs vector.
   If $\|r_j\|_2$ is small enough exit

3. Form approximate solution

   Solve for y_m
   $u_m = \mathcal{M}^{-1}(V_m y_m)$
   Form $x_m = x_0 + u_m$

4. Restart

   Compute $r_m = b - \mathcal{A}x_m$
   If $\|r_j\|_2$ is small enough stop
   Set $x_0 \leftarrow x_m$ and go to 1

Figure 4.3: Algorithm for preconditioned GMRES

There are two classes of preconditioners to do this. The first actually forms some form of an approximation to the inverse. The second uses an iterative method to approximately solve Equation 4.21. Popular iterative methods for preconditioning are Jacobi (diagonal), Gauss-Seidel, and SSOR. GMRES has also been chosen as a preconditioner, leading to the nested GMRES method [62]. Note that with these methods, $\mathcal{M}^{-1}$ changes at each step, so that

$$x_m = x_0 + \mathcal{M}^{-1} V_m y_m \tag{4.22}$$

is no longer valid, but instead

$$x_m = x_0 + Z_m y_m \tag{4.23}$$

must be used, where Z_m is the matrix formed from the m vectors $z_j = \mathcal{M}_j^{-1} v_j$. This doubles the memory requirement of GMRES, since both v_j and z_j are required. This variant of GMRES, developed by Saad [63] is called flexible GMRES (FGMRES).

Following Pueyo [64], we have focused on the class of preconditioner where an approximation to $\mathcal{A}^{-1}$ is stored, specifically the incomplete lower-upper factorization preconditioner (ILU). In this method, an LU decomposition is approximated

$$\mathcal{A} = \mathcal{L}\mathcal{U} - \mathcal{E} \tag{4.24}$$

where $\mathcal{L}$ and $\mathcal{U}$ are lower and upper triangular matrices, respectively. As the factorization proceeds, elements are dropped according to some rule. In the simplest case, only elements which have a corresponding entry in the matrix $\mathcal{A}$ are kept. This is called ILU(0), due to the *level of fill* being zero. Higher levels of fill are possible, giving ILU(n). In this case, each element to be added to the preconditioner during the factorization has a level calculated for it. If it has a corresponding entry in $\mathcal{A}$, it is assigned zero. Otherwise, the level of fill is the sum of the fill of the two elements which contributed to its formation, plus one. If the new element's level of fill is greater than n, it is dropped from the factorization. The ILU(n) algorithm is summarized in Figure 4.4.

The equations being solved are naturally organized into sets of four or five if the turbulence model is being used. The Jacobian matrix is therefore comprised of blocks. Each of these blocks may have a number of zero entries within it. We have three options when applying the preconditioner. The first is simply to use the standard scalar version, where the blocks are ignored. The second, called BFILU, used by Orkwis [65] and Pueyo [64] forces all of the entries within the block to have a level of fill of zero, even those which may have a value of zero. They have found BFILU more robust than scalar ILU. The third method called BILU, is to treat each block as a single entry. In this case, division operators become block matrix inversions.

ILU Breakdown

Elman [66] has shown that an ILU preconditioner can prove to be ineffective, or even detrimental in a number of situations.

- Poor representation of the original matrix.
- Inaccuracy due to small pivots.

For all nonzero elements $a_{i,j}$

$u_{i,j} = a_{i,j}$

$\text{lev}(u_{i,j}) = 0$

For $i = 2, \ldots, N$

For $k = 1, \ldots, i-1$

if $u_{i,k} \neq 0$

$l_{i,k} = u_{i,k}/u_{k,k}$

$\text{lev}(l_{i,k}) = \text{lev}(u_{i,k})$

For $j = 1, \ldots, N$

$u_{i,j} = u_{i,j} - l_{i,k}u_{k,j}$

$\text{lev}(u_{i,j}) = \min\{\text{lev}(u_{i,j}), \text{lev}(l_{i,k}) + \text{lev}(u_{k,j}) + 1\}$

Replace any element in row i with $\text{lev}(u_{i,j}) > n$ by zero

Figure 4.4: ILU(n) algorithm

- Unstable triangular solves.

Chow and Saad [67] give details on these problems. They suggest three statistics to help determine which is causing poor convergence of the linear solver:

condest	$\|\|(LU)^{-1}e\|\|_\infty$, $e = (1, \ldots, 1)^T$
1/pivot	size of reciprocal of the smallest pivot
$\max(L+U)$	size of largest element in L and U factors

The condest is an estimate of the condition number of the preconditioner, and is a lower bound for $\|\|(LU)^{-1}\|\|_\infty$. It gives an indication of stability of the triangular solves. Instability can be caused by small pivots. To identify this, the second statistic is used. When a problem occurs, the values become quite high, on the order of 10^{15}.

4.2.3 Reordering

Direct solvers commonly use reorderings to improve performance and to reduce memory requirements. There are a number of popular reordering methods, including nested dissection [68], one-way dissection [69], reverse Cuthill-McKee (RCM) [39], and minimum degree [70]. These use a permutation matrix P to reduce either the bandwidth or the profile of the matrix.

$$P \cdot \mathcal{A} \cdot P^T (P \cdot x) = P \cdot b \tag{4.25}$$

The matrix P is the unity matrix with a series of row and column exchanges performed. The bandwidth is

$$\max(\max(k - i, \text{ for all } a_{i,k} \text{ nonzero entries in } \mathcal{A}), \text{for } i = 1, \ldots, N) \tag{4.26}$$

while the profile is

$$2\sum_{i=1}^{N} \max(k-i, \text{ for all } a_{i,k} \text{ nonzero entries in } \mathcal{A}) \tag{4.27}$$

Note that these definitions assume a symmetric matrix graph, valid since most reorderings require this. The graph resulting from the Jacobian found in our systems is not symmetric, due to some boundary conditions, and so we need to take measures to make it so. Some packages make these graphs symmetric by dropping entries. A better approach is to add entries, which avoids disassociating parts of the graph.

A number of authors have determined that applying some sort of permutation to the system can have a significant effect on the convergence rate of the linear system [30], [71]. The number of elements dropped from the preconditioner is significantly lowered by decreasing the profile of the matrix.

Pueyo and Zingg [64] found the reverse Cuthill-McKee most effective for the Navier-Stokes equations. They also found that the ordering of the equations before the algorithm is applied is quite important. To see how the RCM algorithm is affected by the initial ordering, we review how the Cuthill-McKee algorithm operates. All of the nodes are divided into a number of level sets $ls(m)$.

1. Choose root node of minimum degree, which becomes $ls(0)$. $s = 0$.
2. For each node in $ls(s)$, add all unordered neighbours to $ls(s+1)$. When adding, add in order of increasing degree.
3. $s = s + 1$
4. If unordered nodes remain, goto 2.
5. The final order is all entries of $ls(0)$, followed by $ls(1)$, etc.
6. For reverse Cuthill-McKee, reverse the order.

The degree of a node is the number of neighbours it has. The algorithm operates by locally minimizing the average bandwidth of a wavefront. A wavefront is the set of nodes in a level set. Consider that the bandwidth of any node is the number of elements between its position in the order and the farthest neighbour's position. By adding neighbours of the nodes with smaller degrees, the algorithm minimizes the number of elements that have to be crossed by the first entries in the level set. To give example, imagine we have four nodes, each of which share a common node (R, added in the previous level set), and their neighbours:

node	neighbours	degree
a	aa	1
b	ba, bb	2
c	ca, cb	2
d	da, db, dc, dd	4

The ordering would then be R,a,b,c,d,aa,ba,bb,ca,cb,da,db,dc,dd. The local bandwidths are then 4,5,6, and 9, for nodes a, b, c, and d, respectively. If we had ordered the opposite way, adding the nodes with highest degree first, then we would have $R, d, c, b, a, dd, dc, db, da, cb, ca, bb, ba, aa$. The local bandwidths would be 7,8,9, and 9 for nodes d, c, b, and a. While the last bandwidth is the same, the previous three are smaller.

The algorithm as we have presented it is not entirely well defined. First, the root node is simply a node of minimum degree. Before starting the graph is searched for such a node. There are often a

number of nodes with the same minimum degree. We have some choice as to which to choose. Second, when neighbours of a node are being added to the next level set, some may be of equal degree. Again, we have a choice of which node to place first, since the average bandwidth will not be affected. Most often, the initial ordering of the nodes is used to fully define the algorithm. In the above example, b and c have equal degrees. Typically b is then choosen before c. However, Pueyo [27] has shown that careful selection of the initial ordering is crucial to having an efficient preconditioner. He used C-grids over single-element airfoils with an initial ordering starting at the outflow, proceeding across the wakecut, then moving forward. This offers some clue that flow direction is important. The importance of the root node was also discussed by Benzi et al. [72], who studied the solution of a number of linear systems, using various reordering strategies.

It is difficult to pre-order, as Pueyo does, with multi-block grids, so an alternate method is used. When a choice needs to made between two or more nodes with equal degree, the node which is downwind is choosen first. In all cases tried, this yielded a significantly superior preconditioner than the opposite ordering. In many cases, ordering upwind first would not converge. A similar effect is seen with the root node. Downwind roots are often significantly better than upwind. This is discussed in Section 5.6.6.

Chapter 5

Algorithm Optimization

There are a number of parts of the algorithm presented that need investigation to achieve the most efficient and robust solver possible. Properly setting many of these is crucial to even achieving convergence. These adjustments are in either the system solver, including both time stepping and linear system solution, or in modifications to the system of equations. They will be discussed together, since they are so closely related.

We have three goals: to achieve the fastest possible solver while still maintaining robustness, to maintain the same final solution as Cyclone and TORNADO, and to find a set of parameters which is appropriate for the largest number of cases possible. We will see that it is prudent to develop different sets of parameters for laminar and turbulent cases. This is due to the strong effect of the turbulence model, which is significantly less stable than the Navier-Stokes equations. In fact, the turbulence model has a number of components that are highly nonlinear, especially in the trip terms, which are a challenge for a Newton solver. There is often a large difference in the magnitude of the residuals and elements of the Jacobian matrix between the turbulence model and mean-flow equations, which is a challenge for the linear solver. These and other issues need to be addressed.

The optimization is complex, and we will examine a large number of methods. An iterative approach has been taken, so that the best combination has hopefully been found. The reader has been spared all but the end of this long process. We have presented each individual optimization with the best characteristics from all others. The presentation order is somewhat arbitrary. To summarize:

- Equation scaling

Case	Airfoil	Mach	Angle of Attack	Reynolds Num.
1	NACA0012	0.15	6.0°	—
2	NACA0012	0.7	1.49°	—
3	NLR	0.2	5.0°	—
4	NACA0012	0.15	6.0°	$9 \cdot 10^6$
5	RAE	0.729	2.31°	$6.5 \cdot 10^6$
6	NLR	0.3	6.0°	$2.51 \cdot 10^6$

Table 5.1: Optimization test cases

- The start-up phase, including choice of the local time step for the Euler method
- Grid sequencing to provide a good initial guess
- Turbulence model stabilization including a new local time step to alleviate the need to use the matrix-explicit method
- Preconditioning, including using an approximate Jacobian, adjustments for matrix dissipation, level of fill, and reordering optimization
- Approximate solution of linear system
- Matrix-free vs. matrix-explicit GMRES

5.1 Test Cases

The test cases which will be used to optimize Scirocco are given in Table 5.1, which lists the airfoils and flow conditions. Table 5.2 shows the major grid information. The grids were generated using AMBER2D [73], an elliptical grid generator allowing significant user control. Some care was taken to generate a decent grid, including decreasing clustering at the far-field boundary on wake cuts. The blocks have dimensions such that they can be coarsened twice in each direction. Once the grids were generated, they were not adjusted for the solvers.

Three inviscid and three turbulent cases are used. Cases 2 and 5 are transonic. Cases 3 and 6 (based on experimental work by van den Berg [74]) demonstrate multi-element capabilities. Second-difference dissipation is set to zero for all the subsonic cases. All cases were run on an Intel 2800 Pentium 4 desktop computer with 1.5 gigabytes of memory. Further cases to demonstrate the effectiveness of the algorithm are presented in Chapter 6.

5.2 Equation Scaling

We have the option of applying row and column scaling to our equations:

$$\mathcal{S}_r \mathcal{A} \mathcal{S}_c \mathcal{S}_c^{-1} x = \mathcal{S}_r b \tag{5.1}$$

Case	Dimensions	Number of nodes	Offwall spacing	Maximum aspect ratio
1	245×49	12005	$5 \cdot 10^{-4}$	9211
2	245×49	12005	$5 \cdot 10^{-4}$	9211
3	—	27019	$5 \cdot 10^{-4}$	1913
4	305×57	17385	10^{-6}	13672
5	305×57	17385	$2 \cdot 10^{-6}$	8483
6	—	44059	10^{-6}	25514

Table 5.2: Optimization test case grid characteristics

$$\begin{bmatrix} 3 \cdot 10^{2} & 0 & 0 & 0 & 0 \\ -1 \cdot 10^{1} & 4 \cdot 10^{2} & -9 \cdot 10^{-1} & 0 & -8 \cdot 10^{-10} \\ -9 \cdot 10^{-1} & -9 \cdot 10^{-1} & 4 \cdot 10^{2} & 0 & -2 \cdot 10^{-11} \\ -2 \cdot 10^{2} & -7 \cdot 10^{0} & -3 \cdot 10^{-1} & 4 \cdot 10^{2} & -1 \cdot 10^{-11} \\ -6 \cdot 10^{7} & 1 \cdot 10^{8} & 4 \cdot 10^{9} & -5 \cdot 10^{4} & 1 \cdot 10^{2} \end{bmatrix}$$

Figure 5.1: Typical unscaled diagonal block

where $\mathcal{S}_r$ and $\mathcal{S}_c$ are diagonal matrices. Using $\mathcal{S}_r$ effectively scales the residual vector, while $\mathcal{S}_c$ scales the state vector. There are good reasons to use these scalings, most of which are related to the fact that the linear system is solved approximately. For example, if the equations are in the form presented in 2.12 and 2.27, the residual norm of the turbulence model equations is orders of magnitude higher than the mean-flow equations. To satisfy the reduction of the linear residual by one order, only the turbulence model equation residual needs to be reduced. The linear residual of the mean-flow equations can increase, potentially causing negative pressures or densities during the next nonlinear evaluation. A similar effect can occur with just the Navier-Stokes equations. The residual near the body is not necessarily reduced by the linear solver because the inverse Jacobian scaling there results in a residual that is orders of magnitude smaller than that where the cells are larger. Without addressing scaling problems, this situation requires the use of lower time steps or much tighter inner tolerances, both of which slow convergence.

Entries in each Jacobian block are also strongly affected by lack of proper scaling. Figure 5.1 shows a typical unscaled diagonal block in the wake, near the body. The off-diagonal entries for the turbulence model are many orders of magnitude higher than the off-diagonal entries of the mean-flow equations, especially those from the derivative with respect to the turbulence variable. The diagonal elements are quite close in magnitude. This disparity results in part from the mean-flow variables and equations being scaled by J^{-1}. In the example given, the value of J is roughly 10^5.

In the unmodified equations, the diagonal elements (just the scalar diagonal, not the block diagonal) of the Jacobian are already well scaled. This is due to the fact that the mean-flow dependent variables are scaled by the same factor as the residual. Keeping this desirable property requires only that the column scaling matches the row scaling, so the $\mathcal{S}_r = \mathcal{S}_c^{-1}$. This also ensures the eigenvalues of the scaled system are the same as the original.

We will consider two categories of scaling: inherent scaling of equations, and auto-scaling. The former is used to eliminate permanent scaling problems, such as the inherent scaling differences between the turbulence model and mean-flow equations. This is equivalent to simply rewriting the equations in a different form. The latter is calculated on an iteration-by-iteration basis, and is used when discrepancies arise between residual norms of each of the four or five equations. It is important to understand the causes of the difference in scaling between equations. These are:

- The mean-flow equations and state vector have a node-by-node scaling of the inverse of the metric Jacobian, as presented in equation 2.12
- The turbulence model equation has no inherent geometric scaling.
- The turbulence quantity ranges up to about 1000, while the mean-flow quantities typically do not exceed 2.0
- Local differences may arise, for example, near trip points upon transition.

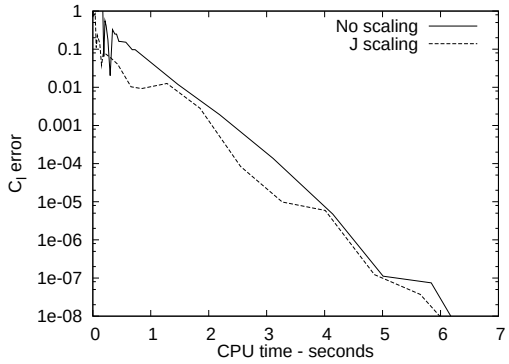
During convergence, the residual norms do not necessarily change equally. For example, the residual of the turbulence model may increase by an order or more with changes in the trip terms, though this does not immediately affect the mean-flow equations. This can lead to negative pressures or densities if the linear residual of the mean-flow equations is allowed to increase. Auto-scaling addresses this problem by calculating a scaling for each set of equations at each iteration. Bringing each equation residual norm within an order of magnitude of each other will not allow an increase in residual (assuming the linear residual tolerance is an order of magnitude reduction). We call this auto-scaling. The largest difference between the norms may also be chosen to be less than the set reduction of the linear system. This potentially allows the nonlinear iterations to converge more rapidly, but requires more inner iterations. In practice, using a factor between one and ten gives little difference in convergence time.

The reference time step evaluation, as presented in Section 5.3.1, is strongly affected by the row scaling ($\mathcal{S}_r$ in equation 5.1). The choice of α is clearly changed with scaling, but there is a more subtle effect. The relative magnitude of the norms of mean-flow and turbulence model equations will dictate which sets the time step. For example, when the unscaled equations are used, the turbulence model effectively controls the time step because the residual is higher. Application of different scaling at each iteration can have an unpredictable effect on the time step. For this reason, the reference time step is calculated before the iteration-wise scaling is applied, but after the inherent scaling.

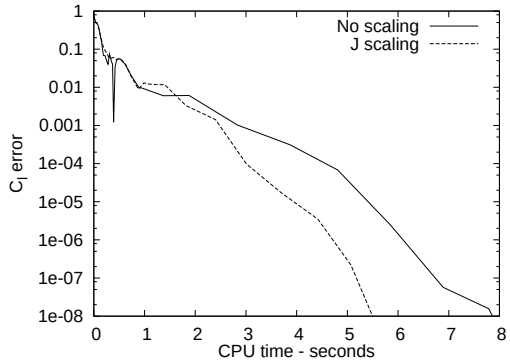
We begin the investigation into scaling with a simple inviscid flow. Figure 5.2 compares the unscaled equations with equations scaled by the metric Jacobian, so that $\mathcal{S}_r = \mathcal{S}_c^{-1} = J$. The residual cannot be directly compared, due to the rescaling, so the error in the coefficient of lift is shown. The error is found by taking the absolute value of the difference between the current and final value. There is no significant difference in the subsonic cases. A more aggressive time step can be used with the transonic case when the equations are scaled, decreasing the time required.

Figure 5.3 shows four options for the turbulent cases:

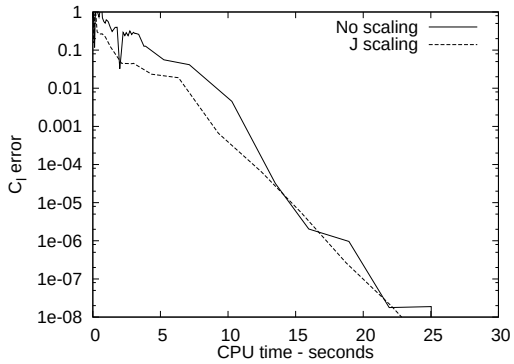
- No scaling, equations as given in equations 2.12 and 2.27.



(a) Inviscid single-element subsonic

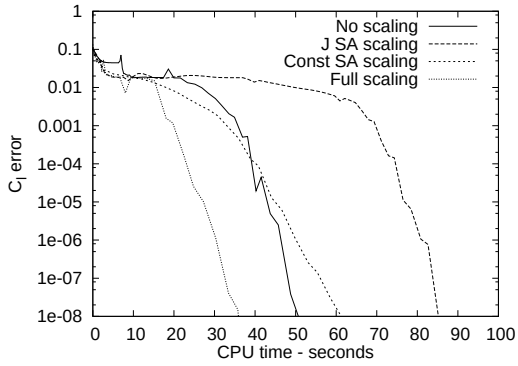


(b) Inviscid single-element transonic

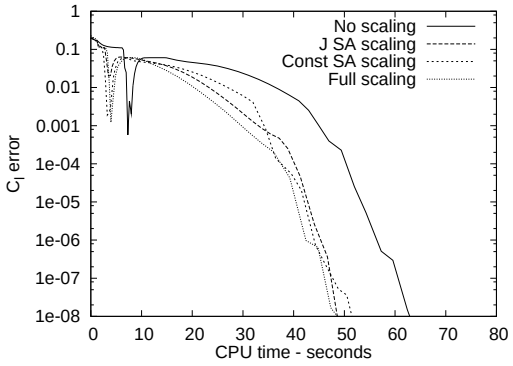


(c) Inviscid multi-element

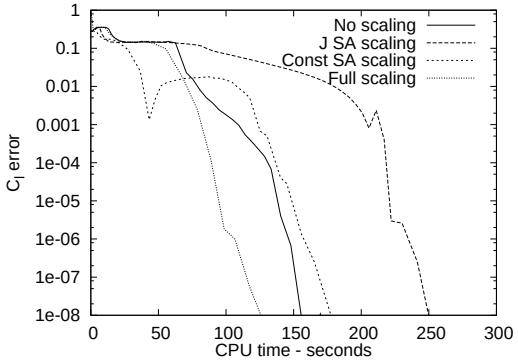
Figure 5.2: Effect of scaling on inviscid cases



(a) Turbulent single-element subsonic



(b) Turbulent single-element transonic



(c) Turbulent multi-element

Figure 5.3: Effect of scaling on turbulent cases

- The turbulence model equation is scaled by the inverse of the metric Jacobian to match the mean-flow equations. The turbulence variable is scaled by the same factor.

- The turbulence model equation is scaled by a constant value, 10^6, that makes the norm of the turbulence model equations roughly the same as the mean-flow equations.

- The mean-flow equations and variables are scaled by the metric Jacobian. The turbulence model equation and variable is scaled by 10^{-3} to normalize the maximum value of the turbulence quantity. Auto-scaling of the residuals is also performed, bringing the maximum difference between the residual norms to within a factor of ten. The initial scaling affects the scaling on a node-by-node basis, as well as making the time step affected by both sets of equations.

The results clearly show the benefits of the last approach to scaling. There is also a significant increase in robustness.

5.3 Start-up

The equations being solved can be highly nonlinear, especially the turbulence model. For this reason, steps must be taken to stabilize the Newton method until the linear approximation is adequate to converge. This must be done carefully to ensure that convergence is slowed as little as possible. The primary tool to accomplish this is the addition of a time step, as described in Section 4.1. Each node and each equation can have an individual time step that changes at each iteration. To simplify this situation, we use the following form for the time step:

$$\Delta t = \Delta t_{ref} \cdot \Delta t_{loc} \tag{5.2}$$

The reference time step, Δt_{ref}, is a scalar, and the same value is used for all nodes. This value changes at each iteration, and is set depending on the residual norms, and possibly other conditions. Δt_{loc} varies at each node, and potentially for each equation, but is constant with the iteration. The best choices for Δt_{ref} and Δt_{loc} are discussed below.

In some instances, this method is overridden. The turbulence model requires an extra degree of stabilization to prevent negative values of the turbulence quantity from occurring. A special local time step is used when necessary to accomplish this. This is discussed in Section 5.5.2.

5.3.1 Reference Time Step

Some authors have suggested using a function to define Δt_{ref}, dependent on the norm of the residual. For example, we have had success with

$$\Delta t_{ref} = \max(\alpha \cdot ||R||_2^{-\beta}, \Delta t_{min}) \tag{5.3}$$

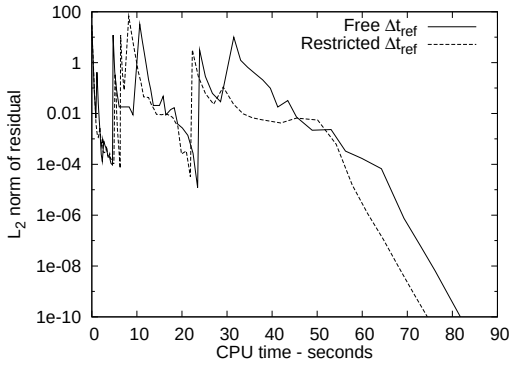
There are a few interesting details to recognize when using this approach. The residual, R, has been scaled such that inherent scaling differences are removed, as discussed in Section 5.2. The interior of the domain has zero residual at the first iteration. The only contribution to the residual is from the equations at the body. The residual from these equations, which do not have a time component, should not be used to set the coefficients in the above equation. Until the residual of the interior is high enough to drive 5.3, Δt_{ref} should be held to the minimum value. Usually this only takes two or three iterations. This is the purpose of Δt_{min}. The maximum function in Equation 5.3 is only useful if the boundary condition equations have a magnitude that is large enough that the norm of the initial residual does not cause the time step to increase past Δt_{min}. In other words, if the boundary conditions are scaled such that the initial residual is very small, Δt_{ref} will be determined by the magnitude of the boundary equations. This is not ideal since the time step is only applicable in the interior, and so the residual at the boundaries should not directly influence the evaluation of Δt_{ref}.

The proper choices of Δt_{min}, α, and β are quite important. The time step will increase very slowly if α or Δt_{min} are chosen, or are required to be, too small. Fortunately, a set of coefficients that gives fast results is available. They depend strongly on the scaling of the equations. This is discussed in Section 5.2. The set of coefficients also needs to be adjusted depending on the type of case being solved. For example, using a turbulence model requires a more conservative time step schedule. The existence of a shock also requires a smaller minimum Δt_{ref}, due to increased nonlinearities in the mean-flow equations.

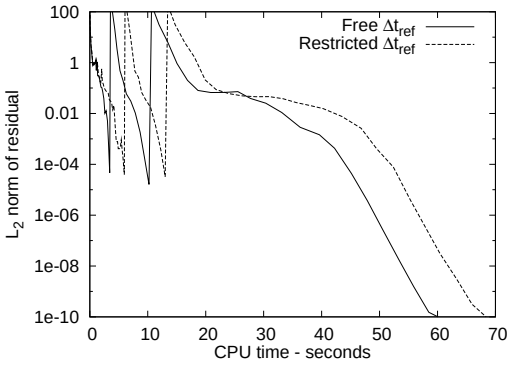
Equation 5.3 is really only useful while Δt_{ref} is below about 10^4. Above this, we are effectively using Newton's method. Hopefully, the time spent going from the minimum time step value to Newton's method is reasonably short, relative to total time to converge. Choosing β strongly affects this. We find that $\beta = 1$ provides a rapid increase of time step, while maintaining stability. For inviscid flows, $\alpha = 1000$ is appropriate, while $\alpha = 100$ is better for turbulent flow. This reflects the increased nonlinearity of the turbulence model equations. Subsonic cases can use an initial time step of $\Delta t_{min} = 100$, while transonic cases should use a lower value of 10 due to the presence of the shock. These values are conservative. Quite often, Δt_{ref} and α can be an order of magnitude higher, and convergence is still obtained.

Finally, this method is only efficient when there is a clear connection between the maximum potential time step and the residual. Unfortunately, this is not always true, especially with turbulent flows. In many of these cases, if too high a multiplier, α, is used divergence will occur after the residual drops an order of magnitude. On the other hand, if α is decreased, a long plateau occurs in convergence. During this time, when the residual is not changing significantly, the time step could be ramped up, significantly shortening this phase. To overcome this, we can use indicators other than the residual. One which is successful is limiting the reference time step to the minimum value when there are nodes with a restricted time step due to the stabilization of the turbulence model (see Section 5.5.2). In this case, there are nodes that are using a lower value than that found from equation 5.2. There are two potential advantages here. First, decreasing the time-step outside of the problem nodes reduces the number of inner iterations. Hopefully then, the total inner iterations needed until the problem nodes are stabilized will be decreased, since the number of outer iterations are primarily set by the problem nodes with the small time-step. Second, there is also a potential increase in stability. Reduced time steps indicate problem areas in the solution. Holding the rest of the nodes to a lower time step helps to ensure these nodes remain stable. Figure 5.4 shows the result of holding the reference time step to the minimum value when there are nodes with a restricted time step. These cases have the trip terms activated. There is little difference seen in the single-element cases, and in fact a small slowdown is seen with the transonic case. However, the multi-element case is strongly improved. A third run was performed with the multi-element case showing the effect of decreasing α (the reference time step multiplier from equation 5.3) by one order of magnitude, and not holding the reference time step to the low value. This case has particularly pugnacious trip terms, so it benefits the most. Cases with the trip terms deactivated generally do not see much benefit from this technique. This is likely due to the increased nonlinearity with the trip terms, which are not always manifested through a higher residual.

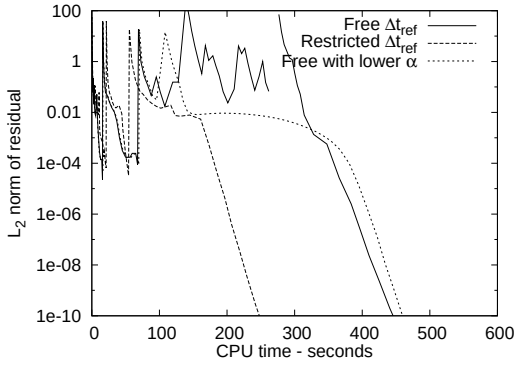
Another restriction on the reference time step is useful for improving stability. A common problem sometimes occurs after a large jump in the reference time step, when the linear solver breaks down (meaning the linear solver returns a vector with significant error, see Section 5.7). The reason for this is not clear. An easy method to avoid this situation is to restrict the increase in the reference time step between iterations to one order of magnitude. Normally, this restriction will not be in effect. This is because the linear system is only reduced by one order of magnitude, so that the nonlinear residual should also only be reduced by an order, leading to an order of magnitude increase in the reference time step. Drops in the nonlinear residual greater than one order may indicate a strong nonlinearity. Figure 5.5 shows the residual of the turbulent cases with and without this restriction. As expected, there is usually little difference in the results. However, using this provides extra stability in some cases,



(a) Turbulent single-element subsonic



(b) Turbulent single-element transonic



(c) Turbulent multi-element

Figure 5.4: Effect of holding reference time step to minimum (with trip)

especially when the trip terms are active.

5.3.2 Local Time Step Choice

To complete equation 5.3, an expression for Δt_{loc} is required. We follow Pulliam [55], who suggests using a local time step for the mean-flow equations dependent on grid metrics. This approximates a constant CFL, and was shown to provide a large improvement in convergence rates when compared with a constant time step. This geometric time step is

$$\Delta t_{loc} = \frac{1}{1 + \sqrt{J}} \tag{5.4}$$

We also need to choose a time step for the turbulence model. The geometric time step is compelling for the same reason as for the mean-flow equations: an approximately constant CFL improves stability for the convection part of the equations. However, the turbulence model is much less convection dominated than the mean-flow equations, and TORNADO uses a spatially constant time step for the turbulence model, so we will compare the spatially constant and varying time steps.

In both methods, the experiments were carried out using the varying reference time step as described in Section 5.3.1. We have to choose a set of parameters (Δt_{min}, α, and β) for each method. Also, we should allow a different Δt_{ref} for the mean-flow and turbulence equations. This is especially important when a spatially different time step is used for the mean-flow and turbulence equations. We could use an entirely separate parameter set for each, based on the corresponding residual norm. However, to simplify, one set of parameters is used for both sets of equations, but the turbulence reference time step is scaled by a factor τ to allow some flexibility. We do this to reduce the number of parameters to optimize. This also limits the time step of the mean-flow equations when the residual of the turbulence model is high, which is usually desirable.

To summarize, a spatially varying time step for the turbulence model is

$$\Delta t = \frac{\tau \cdot \Delta t_{ref}}{1 + \sqrt{J}} \tag{5.5}$$

while the constant time step is simply

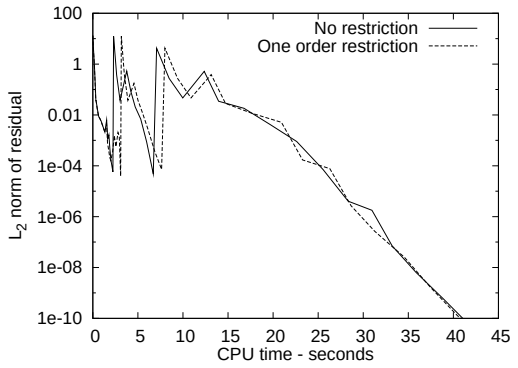
$$\Delta t = \tau \cdot \Delta t_{ref} \tag{5.6}$$

Using $\tau = 1$ is appropriate for a spatially varying system, which is perhaps not surprising. Choosing $\tau = 4$ is optimal for a constant time step. Both methods use $\alpha = 10$, $\beta = 1$, and $\Delta t_{min} = 10$. Full scaling was used, as discussed in Section 5.2. If a different method of scaling the residual were used, the optimization would need to be repeated.

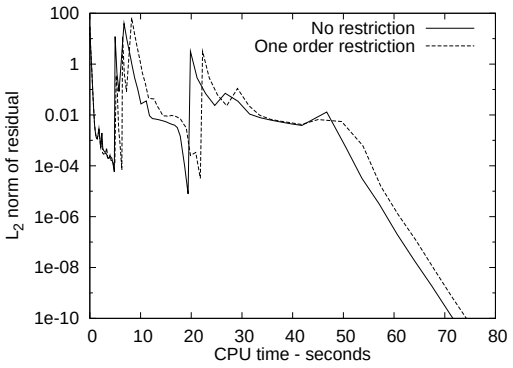
Figure 5.6 compares the geometric time step and a constant time step in the turbulence model for the subsonic single-element and the NLR cases. Using a local time step based on the approximate CFL is the better choice.

5.4 Grid Sequencing

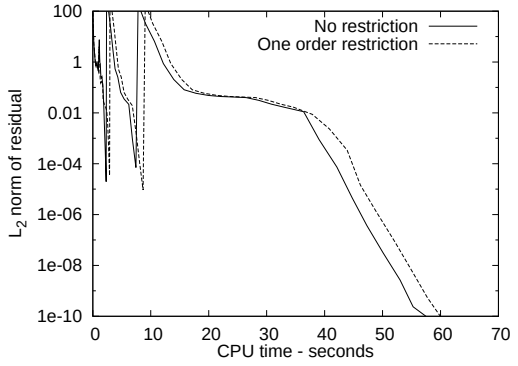
Grid sequencing uses a series of coarse grids to provide a good initial guess on the final grid. This has the advantage of initiating the fine grid much closer to the region of convergence of Newton's method, and reducing the start-up time. Potentially, a larger Δt_{min} can be used on the fine grid.



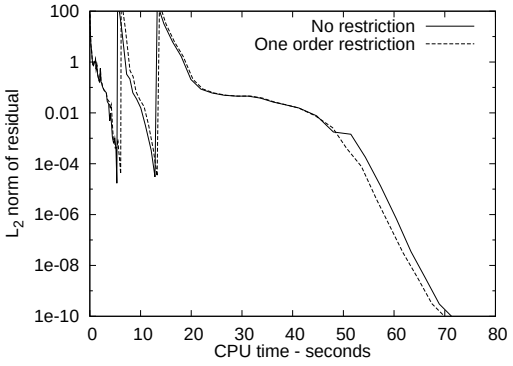
(a) Turbulent single-element subsonic without trip



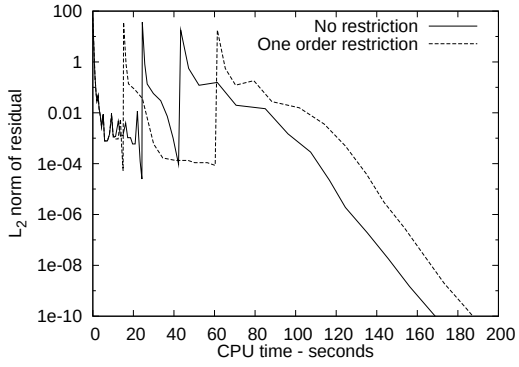
(b) Turbulent single-element subsonic with trip



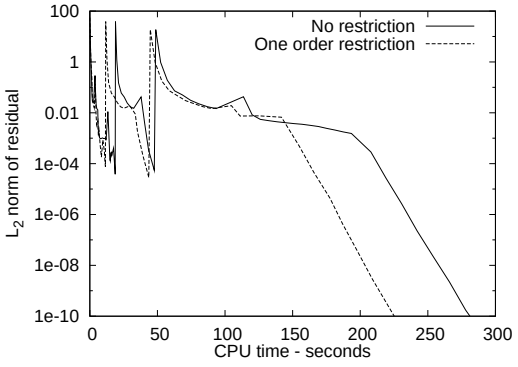
(c) Turbulent single-element transonic without trip



(d) Turbulent single-element transonic with trip

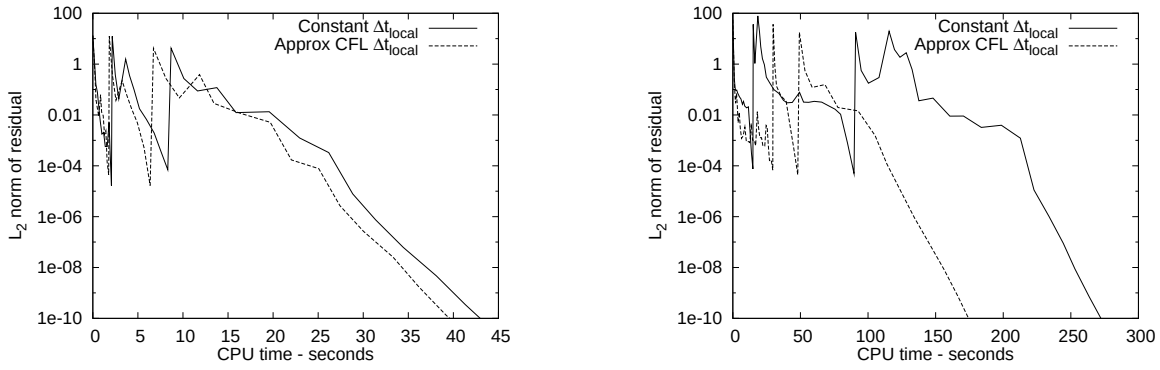


(e) Turbulent multi-element without trip



(f) Turbulent multi-element with trip

Figure 5.5: Effect of restricting reference time step increase



(a) Turbulent single-element subsonic

(b) Turbulent multi-element

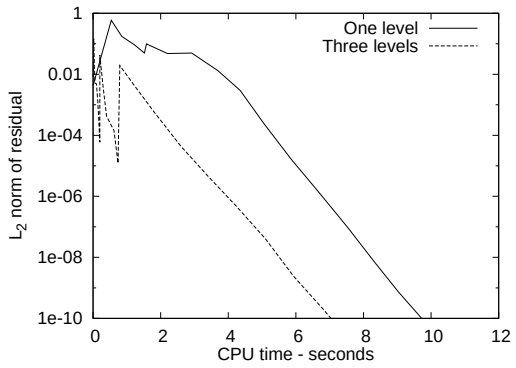
Figure 5.6: Spatially varying vs. constant time step in the turbulence model equation

Each coarse grid is formed by removing every other grid line in both directions from the parent grid. Figure 5.7 compares the results of using a three-grid sequence and a single grid. The three-grid cases use scalar dissipation on the coarser grids, and matrix dissipation on the fine grid. The single grid cases begin with scalar dissipation, and switch to matrix dissipation after the residual drops to 10^{-3}. The change in the dissipation explains the spike in the residual. The turbulent cases do not use trip terms. We see significant benefit to using grid sequencing. It also makes it easier to choose the coefficients for the reference time step equation. A more aggressive sequence can be used on the fine grid, while more conservative values can be used on coarse grids, where the iterations are quicker. Using $\alpha = 10$ in the calculation of the reference time step is necessary when starting up. This slows the single level cases significantly. It is potentially possible to increase the time step earlier, especially in case four, but it is difficult to do this reliably.

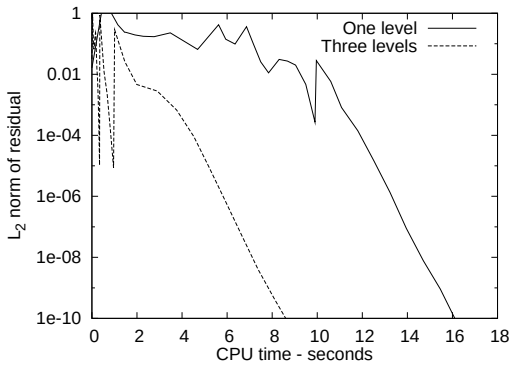
5.5 Turbulence Model Stabilization

Examination of the turbulence model with a value of $\tilde{\nu}$ less than zero shows that production, which is roughly proportional to $\tilde{\nu}$, will become negative, while the destruction term, which is proportional to $\tilde{\nu}^2$, remains negative. This shows that the model becomes unstable as $\tilde{\nu}$ decreases from zero. Therefore, it is crucial to take steps to ensure that $\tilde{\nu}$ remain positive. An obvious fix is to clip the turbulence variable after each update. This is useful, but not sufficient to stabilize start-up without the use of an extremely small time step. This is a result of the strong coupling between the turbulence model and mean-flow equations. A large update of the turbulence quantity can drive the pressure or density negative. Further measures are required. We present two methods of stabilizing the turbulence model during the critical start-up. Both are aimed at preventing updates to $\tilde{\nu}$ that will make it negative. The first follows Spalart and Allmaras in modifying the Jacobian of the turbulence equations. The second presents a new time step designed to limit updates. Clipping as well as these methods are discussed below.

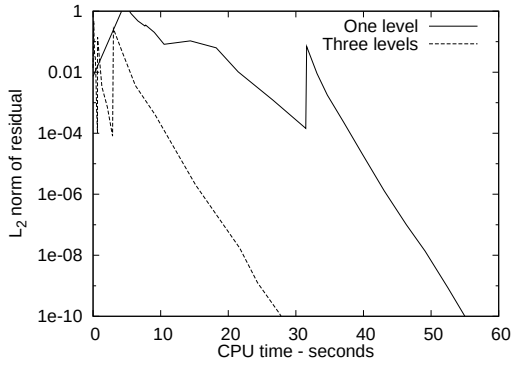
When the mean-flow equations are started from free-stream, there are usually large transients present at the body during the first few iterations. The production and trip terms can produce very large residuals



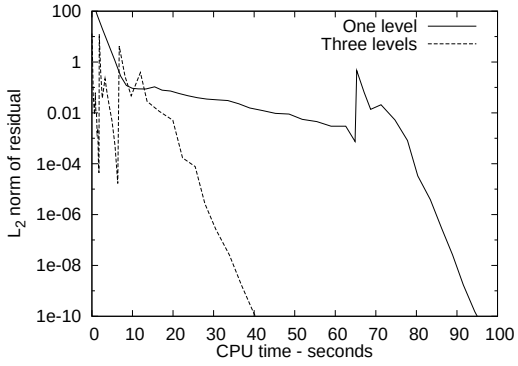
(a) Inviscid single-element subsonic



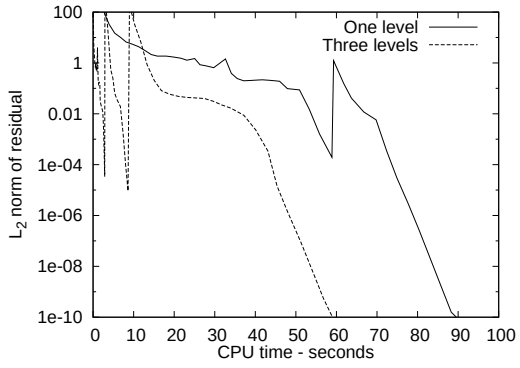
(b) Inviscid single-element transonic



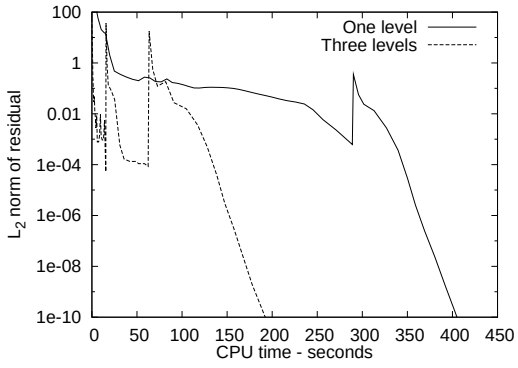
(c) Inviscid multi-element



(d) Turbulent single-element subsonic



(e) Turbulent single-element transonic



(f) Turbulent multi-block

Figure 5.7: Grid sequencing convergence

because of this. It is generally a good idea to perform a few iterations of the mean-flow solver without the turbulence model to establish a reasonable flow at the body.

A common problem occurs when the turbulence model fails to trip in a portion of the flow during the early iterations. The residual can drop a few orders of magnitude with this laminar region. Unfortunately, this means that the time step has increased, and can be approaching Newton's method, when this region finally becomes turbulent. The result can be a spike in residual with a corresponding slow down in convergence, or can often be divergence because of the large time step. This is a more common problem when fully turbulent flow is used, with no trip terms to help transition. Fortunately, this is easily treated. By starting with a low level of turbulence where the final solution is expected to have transitioned, and by starting the cases with just the mean-flow equations as described above, we can obtain proper tripping. This turbulence viscosity is rapidly convected from regions that lack enough shear to sustain it. By starting with a few mean-flow iterations, we establish enough shear near the body to hold the turbulence viscosity where we want it. Using $\tilde{\nu} = 10$ is high enough to seed tripped regions, but still convect out quickly.

When no trip terms are used, we are assuming fully turbulent flow, so all nodes start with this small turbulence quantity. When using trip terms, just those regions behind the trip points are set to nonzero. It is important to leave the turbulence viscosity at zero in regions upwind of trip points. Otherwise, it is common that the flow remains fully transitioned, even when fully converged.

5.5.1 Modified Jacobian

Spalart and Allmaras [14] suggest modifying the Jacobian of the turbulence model so that it becomes an M-matrix. These modifications are discussed in Section 3.2. Note that these changes preclude the use of matrix-free GMRES, and prevent Newton-like (quadratic) convergence. The rationale behind the change is that the updates to the turbulence quantity will always be positive.

The original modifications were derived for a loosely coupled or uncoupled solver, so that there are no derivatives of the turbulence model with respect to the flow state vector. The presence of these coupling derivatives could complicate the modifications significantly. However, experimentation has shown that changes to these entries are not necessary to obtain positive updates.

Of course, these changes are not desirable when using a Newton solver and slow the outer iterations significantly. However it is during start-up when the highest transients are occurring that we are at the most risk of seeing very large negative values. After the residual has dropped, it is possible to return to the unmodified Jacobian without much risk of encountering negative values of $\tilde{\nu}$.

The time step method presented in Section 5.3 does not work well for this method. It requires much higher time steps for efficiency. Also, there is no clear relationship between the maximum stable time step and residual. In fact, a time step of infinity is often possible, after a few iterations to smooth the solution. Convergence is instead limited by the modified Jacobian. For these reasons, we use an alternative method to set the time step when the modified Jacobian is used. A typical time step series using grid sequencing proceeds as follows:

- On the twice-coarsened grid
  - $\Delta t_{ref} = 50$ until turbulence model trips

  - $\Delta t_{ref} = 500$ until $||R|| < 10^{-4}$

- Interpolate to once-coarsened grid
  - $\Delta t_{ref} = 50$ for 3 iterations
  - $\Delta t_{ref} = 500$ until $||R|| < 10^{-4}$

- Interpolate to final grid
  - $\Delta t_{ref} = 50$ for 3 iterations
  - $\Delta t_{ref} = 500$ until $||R|| < 10^{-4}$
  - Newton stage, with true SA Jacobian

The first stage on the twice-coarsened grid ends after the turbulence residual has peaked. The value it reaches varies strongly by case, but it is generally safe to choose a minimum of 15 iterations, and switch to the next stage when the residual drops below one. This stage is quite fast because a coarse grid is used, so a conservative number of iterations should be used.

After interpolating, a few iterations are performed at a lower time step in order to smooth out interpolation errors. Negative values of $\tilde{\nu}$ often show up here. Clipping is useful to maintain stability. On the final grid, the transition to the true Jacobian should happen as soon as possible, since the modifications to the Jacobian slow convergence significantly.

The above assumes that full scaling is used, as described in Section 5.2. If a different scaling is used, the linear residual may need to be reduced much more than one order of magnitude in order to maintain stability. There seem to be two reasons for this. The modifications to the Jacobian require tighter linear tolerances to enforce a positive update. Second, since large time steps are being used, the linear residual of the mean-flow equations must be reduced, or at least not allowed to grow in order to preserve nonlinear stability. They are much smaller than the turbulence model residuals, so a correspondingly smaller linear tolerance is required. This subject is discussed more fully in Section 5.7.

Inner tolerances as low as 10^{-4} are often necessary for stability when using Jacobian scaling with the turbulence model. This is not nearly as detrimental as would first seem. The first few orders of magnitude reduction of the linear residual happen much faster than in the laminar case, often in only two or three iterations. In fact, a good rule of thumb is to perform an extra order of magnitude of reduction after the elbow in linear convergence is seen. Figure 5.8 shows an obvious example. This can be explained by realizing the first few orders of reduction are in the turbulence model residual, which has an M-matrix Jacobian. This is quite beneficial for the linear solver. The linear residual of the mean-flow equations begins to be reduced when the elbow occurs. In the example shown, it would be appropriate to stop at the seventh or eighth iteration.

5.5.2 Local Time Step

The production and destruction terms are highly nonlinear and exhibit some very undesirable characteristics when used with a Newton solver. Figure 5.9 shows the local residual of the turbulence model at a problem node. In this particular case, the residual and derivative were evaluated at $\tilde{\nu}$ roughly equal

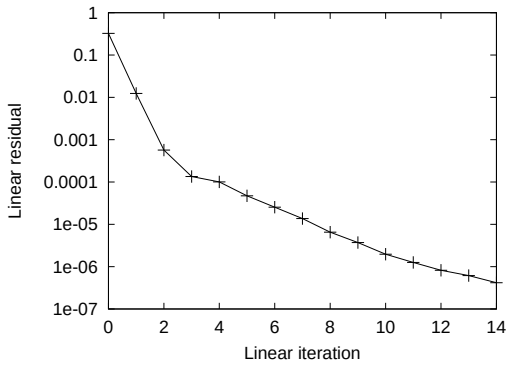


Figure 5.8: Linear convergence during start-up of turbulent case

to 17. The slope is positive, but very close to zero. This situation leads to a large negative update. Fortunately, it is also reasonably easy to avoid, if we use an appropriate time step. The following applies only locally and does not address coupling effects. However, most coupling results from convection and diffusion, both of which are unlikely to result in large negative updates, and trip terms, which are discussed in Section 5.9

If Newton's method is used on this uncoupled equation, we get the following update:

$$\Delta\tilde{\nu} = \frac{R}{-J_D} \tag{5.7}$$

where R and J_D are the residual and diagonal element of the equation, respectively. We want to limit this update so that

$$|\Delta\tilde{\nu}| < |r| \cdot \max(\tilde{\nu}, 1.) \tag{5.8}$$

where $|r|$ is a specified ratio. Choosing $|r| = 1$ would keep the updated $\tilde{\nu}$ positive in the uncoupled case, but in practice a smaller value is more robust. The maximum function ensures that values of $\tilde{\nu}$ very close to zero can be increased in a reasonable time. While it also allows negative updates, these are very small and are clipped to zero (see the following section). To enforce the limit on $\Delta\tilde{\nu}$, we use a local time step, $\Delta t_{\tilde{\nu}}$, determined from

$$\Delta\tilde{\nu} = \frac{R}{\Delta t_{\tilde{\nu}}^{-1} - J_D} \tag{5.9}$$

Applying the target update gives

$$\Delta t_{\tilde{\nu}} = \left[\frac{R}{r \cdot \max(\tilde{\nu}, 1)} + J_D\right]^{-1} \tag{5.10}$$

$\Delta\tilde{\nu}$ and therefore r should have the same sign as R. This moves the update in the direction of the residual. Figure 5.10 shows different choices of $|r|$. The method is reasonably insensitive to the value, within a range of about 0.4 through 0.8. Values higher than 0.8 can occasionally allow large negative updates.

5.5.3 Clipping

The use of clipping allows a much more aggressive schedule to be used during start-up. When a modified Jacobian is used, transition to the unmodified matrix can be performed earlier. Higher time steps are

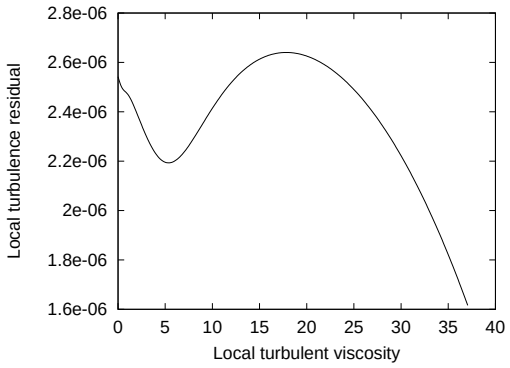
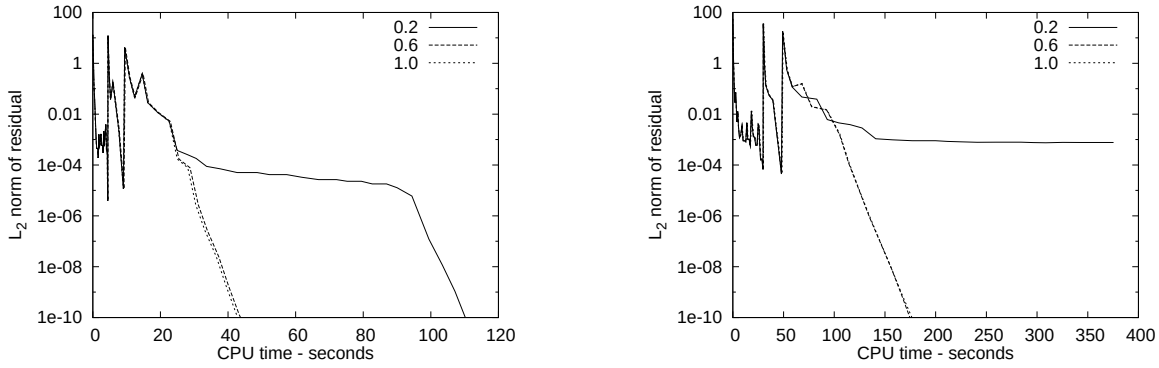


Figure 5.9: Local turbulence residual at problem node



(a) Turbulent single-element subsonic

(b) Turbulent multi-element

Figure 5.10: Local turbulence time step limiting ratio

possible, since mild negative values can be tolerated. The techniques suggested will be used throughout the convergence history. The simplest correction is to 'clip' any negative values, forcing them back to zero. There are two ways to do this. The most obvious is simply to set any negative values of $\tilde{\nu}$ to zero after each update. The second way is to replace the usual turbulence equation for the node in question with $\tilde{\nu} = 0$ at the next iteration. The advantage to the latter method is that it is done implicitly, so that the recovery is included in the surrounding updates. The disadvantage is a loss of an iteration for the negative nodes. In practice, there is little difference between these methods, so the former is used.

A third method proposed is somewhat more sophisticated. The production and destruction terms are the culprits in causing the instability, so they are modified at negative values. In fact, these terms lose their meaning with negative values, so the only consideration is numerical stability. We could zero the production and destruction and hope the convection and viscous terms raise the negative value. Studies have shown that this is often ineffective, and the negative value persists. A better method is to reverse the sign of the production and destruction terms to increase the value.

With a good system to prevent the occurrence of large negative values of $\tilde{\nu}$ in the first place, the method chosen is not terribly important. We use simple clipping, but systems with negative values when fully converged, such as some unstructured solvers, should use a sign change of the production and destruction terms.

5.5.4 Comparison of Stabilization Method

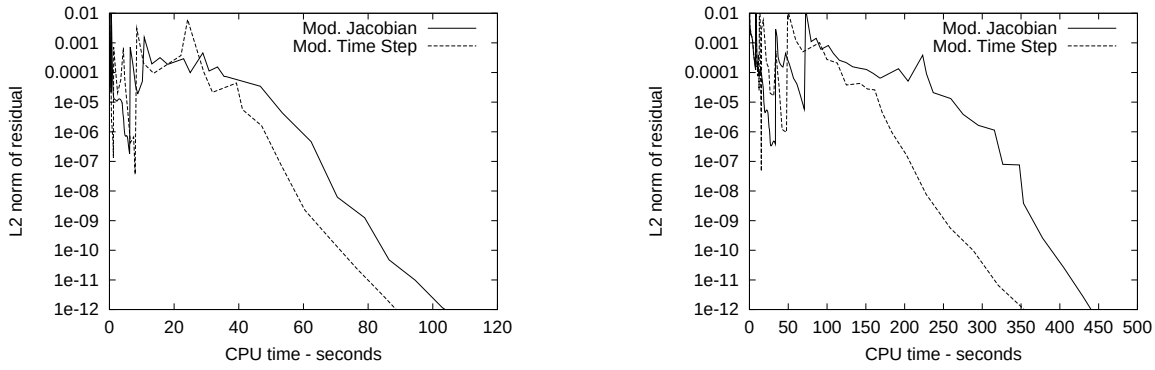
Figure 5.11 compares the modified Jacobian method to the modified time step. The new time step shows a modest improvement over the modified Jacobian. There are other advantages to the modified time step that are not evident in the convergence plots. It does not require an explicit Jacobian matrix to be used, allowing the use of the matrix-free technique and resulting in significant memory savings, as well as opening the possibility to use entirely matrix-free methods. We do not have to decide where to transition from the modified to the exact (or matrix-free) Jacobian. The one or two iterations after this transition can be problematic. Large updates may occur, which can cause instability in both the nonlinear and linear systems. For these reasons, we prefer the modified time step method.

5.6 Preconditioning

The preconditioner is arguably the most important part of the method being developed. Close to half the time spent in the algorithm is spent forming, factoring, and applying the preconditioner. A poorly conditioned factorization can easily breakdown and stop convergence entirely. An over-effective preconditioner wastes both time and memory. We have used the incomplete LU preconditioner discussed in Section 4.2.2 as it has been shown to be effective with the type of equations being solved. However, a number of implementation details remain to be optimized.

5.6.1 Preconditioners Based on Approximate Jacobians

Pueyo [27] found that using an approximate Jacobian for the preconditioner factorization provides a more efficient solver. Blanco and Zingg [75] came to the same conclusion with an unstructured system.



(a) Turbulent single-element subsonic

(b) Turbulent multi-element

Figure 5.11: Modified Jacobian vs time step methods of stabilizing turbulence model

We have also found that the true Jacobian is not a good choice for the preconditioner. In fact, most cases of reasonable difficulty will not converge without high levels of fill. This can be attributed to the low diagonal dominance of the full Jacobian. This leads to a very high condition number of the factorization, and preconditioner breakdown. While it may be possible to artificially increase the diagonal dominance of the matrix the preconditioner is based on, Pueyo instead used a first-order Jacobian for the factorization. This approximation uses the proper linearization of the inviscid and viscous fluxes and second-difference dissipation, but approximates the fourth-difference dissipation by including extra second-difference. This has two major advantages. The only entries on the diagonal are from the viscous fluxes and from dissipation. Introducing extra dissipation increases the diagonal dominance significantly. Removing fourth-difference dissipation also reduces the stencil size from nine blocks per equation to five, giving a large savings in memory, especially after the extra entries from fill are included.

To determine how to increase the dissipation in the approximate Jacobian, Pueyo used the following:

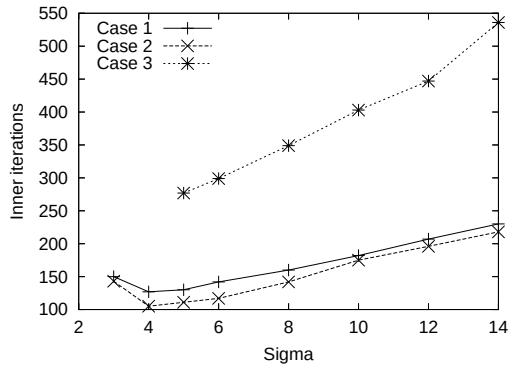
$$\epsilon_2^l = \epsilon_2^r + \sigma\epsilon_4^r \tag{5.11}$$

ϵ is the coefficient of dissipation. The subscript indicates second- or fourth-difference dissipation. The superscript indicates dissipation applied to the matrix (l) or residual (r).

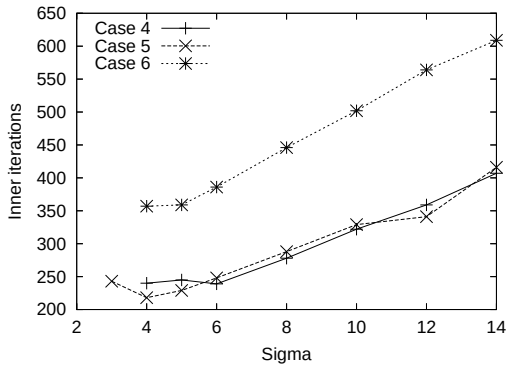
The parameter σ must be optimized. Figure 5.12 shows the effect of σ on the number of inner iterations on the finest grid. Scalar dissipation is used. $\sigma \leq 3$ would not usually converge. $\sigma = 5$ is a good choice.

5.6.2 Matrix Dissipation

Matrix dissipation increases accuracy by reducing the dissipation necessary to maintain stability. This exacerbates the problem with diagonal dominance. The values of the switches are important settings. As the switches approach zero, the condition number rapidly worsens. Subsonic cases use a minimum of $V_l = 0.025$ and $V_n = 0.025$, while transonic cases use $V_l = 0.025$ and $V_n = 0.25$ to provide enough dissipation to control the shock. The values of sigma presented for use with scalar dissipation are not sufficient to stabilize the preconditioner when using such low values for the switches.

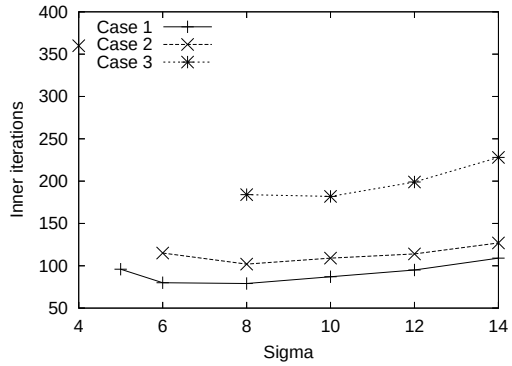


(a) Inviscid cases

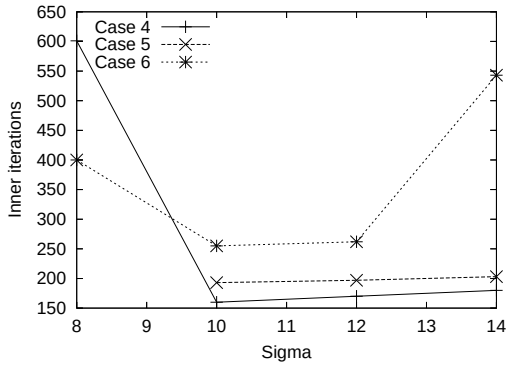


(b) Turbulent cases

Figure 5.12: Sigma optimization for scalar dissipation



(a) Inviscid cases

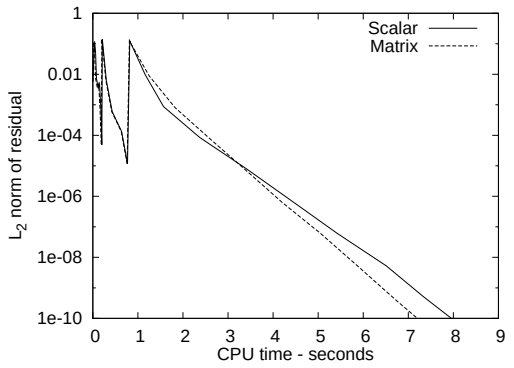


(b) Turbulent cases

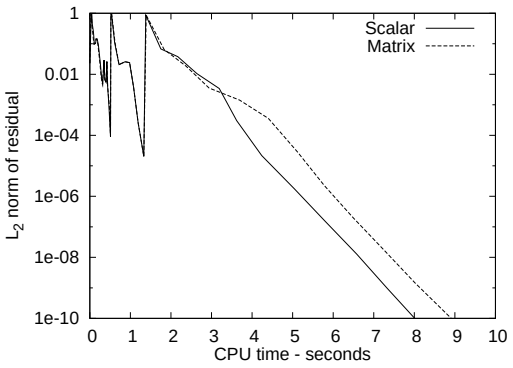
Figure 5.13: Sigma optimization for matrix dissipation

Figure 5.13 shows the inner iterations vs σ for the test cases with matrix dissipation. Using $\sigma = 10$ is a safe choice. An interesting feature of this plot is the sharp increase in inner iterations around σ of 8 or 9, and the very slow increase past 10. The former is characteristic of the preconditioner starting to breakdown, which happens quite abruptly. On the other hand, 'over-stabilization' is not terribly detrimental, so we should err on this side. Figure 5.14 compares convergence rates between matrix and scalar dissipation, using $\sigma = 10$ for the former and $\sigma = 5$ for the latter. It is somewhat surprising that matrix dissipation is slightly faster than scalar. The reason for this is not clear. One theory is that the matrix the preconditioner is based on is closer to the true matrix. Lower levels of dissipation introduce less error in the approximate matrix, since a number of components in the dissipation are not fully differentiated. This results in a more effective preconditioner, as long as it can be properly stabilized.

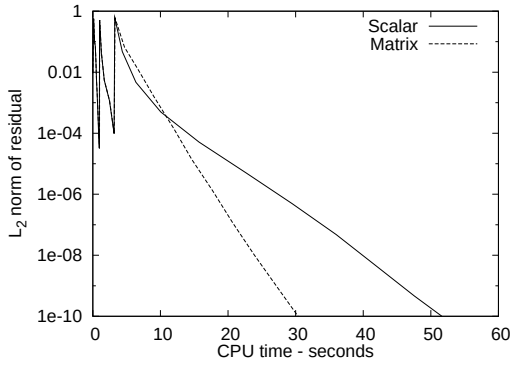
Other methods of stabilization have also been tested. These included increasing V_l and V_n in the matrix the preconditioner is based on. A maximum time step in the matrix was used, as well as combinations of these two techniques. It was possible in most cases to stabilize the factorization, but in practice it was difficult to find a standard set of values that would work consistently.



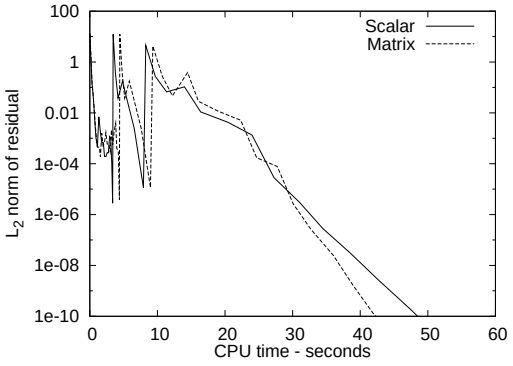
(a) Inviscid single-element subsonic



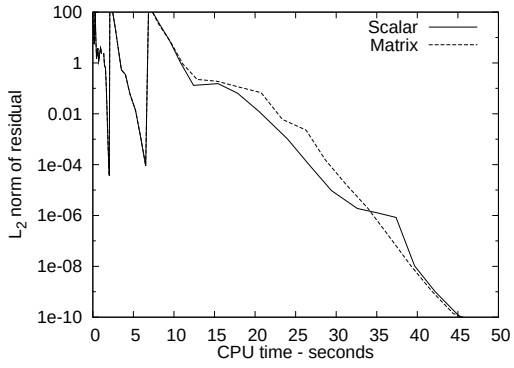
(b) Inviscid single-element transonic



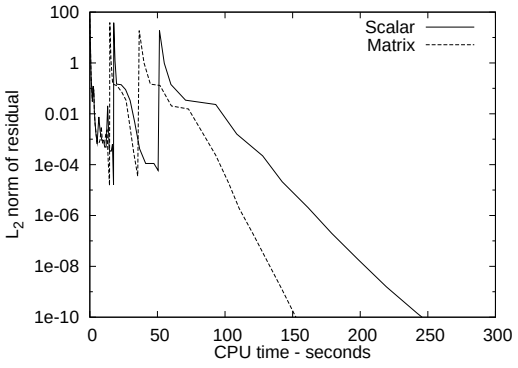
(c) Inviscid multi-element



(d) Turbulent single-element subsonic



(e) Turbulent single-element transonic



(f) Turbulent multi-element

Figure 5.14: Convergence with scalar and matrix dissipation

	Scalar				Matrix			
	Mat-exp		Mat-free		Mat-exp		Mat-free	
Case	in	out	in	out	in	out	in	out
1	126	8	126	8	83	8	83	8
2	123	25	123	19	137	29	95	11
3	400	9	403	9	207	12	183	9

Table 5.3: Importance of dissipation derivatives

5.6.3 Differentiation of Numerical Dissipation Terms

The differentiation of the numerical dissipation terms deserves consideration. The full differentiation of the pressure switch requires a seven-point stencil in each direction (see Equation 3.6). Adding extra storage to the Jacobian matrix is not desirable and is not done. The derivative of the coefficient of dissipation can be prohibitively expensive to calculate, especially in the matrix case. The sheer number of calculations to find the derivatives at each node easily outstrips any benefits. To determine how important these contributions to the Jacobian are, the inviscid cases were run with matrix-free and matrix-explicit. The number of inner and outer iterations were compared to determine whether leaving these terms out will degrade convergence. Note that the derivative of dissipation was included in the matrix-explicit case, but with the assumption that the pressure switch and coefficients are constant. Only inviscid cases were used because slight changes in the method can change turbulence convergence enough to make this comparison meaningless. The circulation correction was not included to ensure that the only difference in differentiation was in the dissipation terms. Table 5.3 shows the results. The subsonic cases, which use only fourth-difference dissipation, show very little change. The transonic case shows only a small difference when scalar dissipation is used. When matrix dissipation is used the number of outer iterations jumps up by almost a factor of three. If the derivative of the matrix dissipation coefficient, and the part of the pressure switch within the existing stencil is included, the number of outer iterations drops to 15. The remaining difference can be attributed to the missing pressure switch components.

5.6.4 Preconditioner Level of Fill

The level of fill used by the preconditioner is an important parameter because it affects robustness, efficiency, and memory use. The preconditioner needs to be powerful enough to remain stable and adequately reduce the eigenvalue spectrum, but too high a level of fill will waste time as the factorization and solves become more expensive. Figure 5.15 shows the effect of fill on convergence. All cases use matrix dissipation. In general, ILU(4) is the fastest choice. ILU(2) shows nearly the same efficiency as ILU(4), with some memory savings. Table 5.4 compares the memory requirements for each level of fill. ILU(4) needs slightly more than twice the memory of ILU(0). Fills of zero would not usually converge at all, unless measures were taken to reduce the time step significantly. Table 5.5 shows the condition number estimates of the cases with different fills. The estimate is found with $||(LU)^{-1}e||_\infty$,

	Preconditioner memory (Mb)					
Case\Fill	1	2	3	4	5	6
0	9.6	9.6	21.6	21.3	21.3	54.3
1	12.6	12.6	28.5	28.0	28.0	71.6
2	15.5	15.5	35.1	34.7	34.7	88.7
3	18.4	18.4	41.8	41.3	41.3	105
4	21.3	21.4	48.4	47.9	47.9	122
5	24.3	24.3	54.1	54.3	54.3	140

Table 5.4: Memory requirements by level of fill

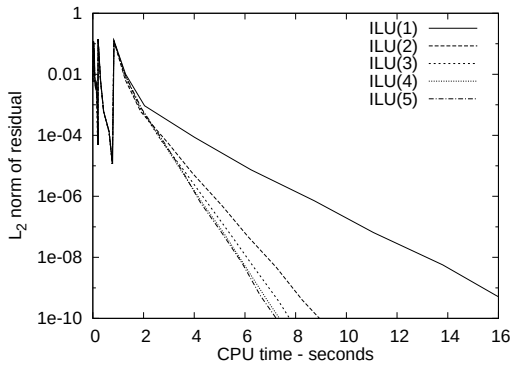
	Condition number estimate					
Case\Fill	1	2	3	4	5	6
0	$1.2 \cdot 10^6$	$3.1 \cdot 10^6$	$2.1 \cdot 10^6$	$1.5 \cdot 10^5$	$1.1 \cdot 10^7$	$3.4 \cdot 10^7$
1	4419	967	13641	4246	447	1338
2	137	159	216	4442	805	8312
3	217	67	430	12314	519	6959
4	230	64	529	9466	2701	19635
5	197	54	329	20293	669	20497

Table 5.5: Condition estimates by level of fill

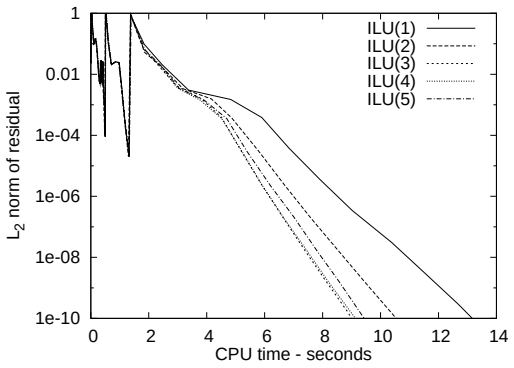
$e = (1, \ldots, 1)^T$, following Chow and Saad [67]. Using a fill of zero gives a very high condition estimate, which explains the need for a low time step.

5.6.5 Preconditioner Freezing

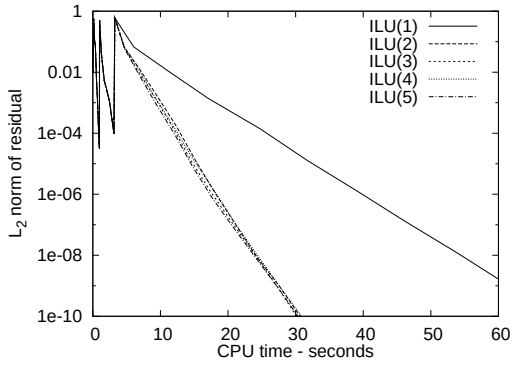
Pueyo suggested freezing the preconditioner to save time. He used approximate-factorization to bring the solution within the Newton region of convergence, so that freezing the preconditioner from the first iteration was useful. Under the cases studied here, there is often a very early switch to Newton's method, so that the Jacobian is changing significantly in the first few iterations. It is not a good idea to freeze the preconditioner during this time. A simple rule is used to decide when it is safe to freeze the preconditioner. After each iteration, the L_2-norm of the update vector is added to a total. If this total exceeds a set value, the preconditioner is updated, and the total is zeroed. An appropriate value at which the preconditioner is recalculated is 0.1. Table 5.6 shows the inner and outer iterations and the total time required with and without preconditioner freezing. The number of inner iterations is not significantly increased with preconditioner freezing. However, the time savings are modest, and in some cases the method will fail to converge (indicated by DNC). For these reasons, we do not recommend using this technique.



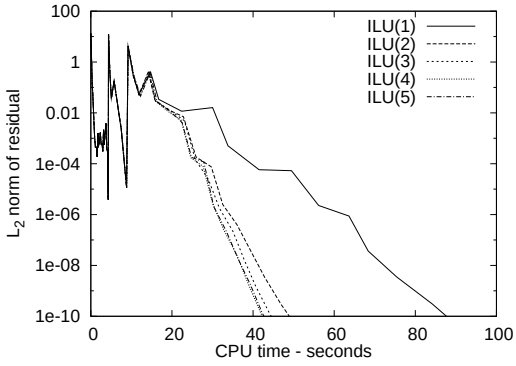
(a) Inviscid single-element subsonic



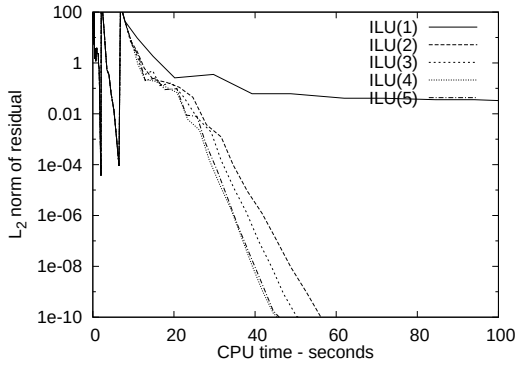
(b) Inviscid single-element transonic



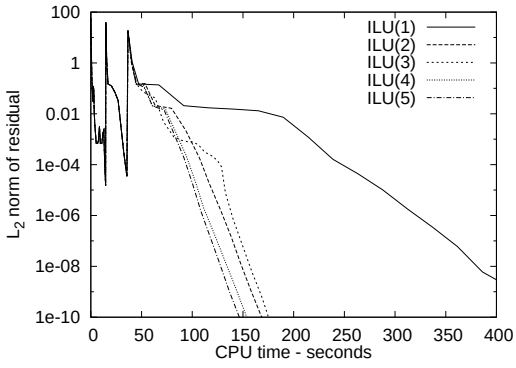
(c) Inviscid multi-element



(d) Turbulent single-element subsonic



(e) Turbulent single-element transonic



(f) Turbulent multi-element

Figure 5.15: Effect of level of fill on convergence

Case	Freezing	Inner	Outer	Total time (sec)
1	Off	87	8	7.5
	On	90	8	6.2
2	Off	100	11	9.6
	On	100	11	8.0
3	Off	182	9	30
	On	180	9	27
4	Off	160	13	45
	On	158	13	40
5	Off	168	16	46
	On	194	16	46
6	Off	255	13	161
	On	—	—	DNC

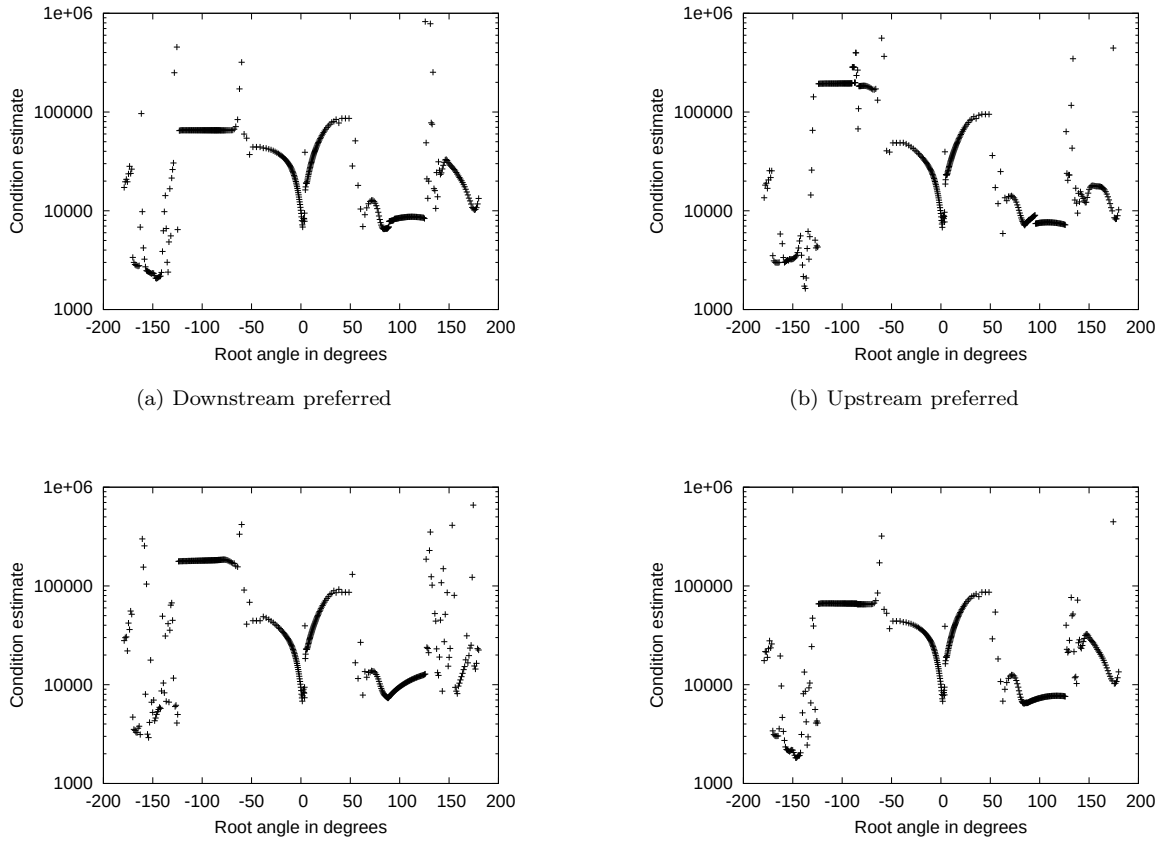
Table 5.6: Effect of preconditioner freezing

5.6.6 RCM Optimization

Section 4.2.3 discussed the operation of the the RCM reordering method. Note that the graph of the preconditioner matrix is used to find the reordering. This is because the reordering is used to improve the effectiveness of the factorization of the preconditioner.

As given, there is some flexibility within the RCM method. There are two items which need to be defined: the method of choosing between two nodes which are equal in degree, and how to choose the root node. The root node is normally a node with minimum degree. In the case of the equations being solved here this is a degree of two, and occurs on the far-field or solid wall boundary only.

It is difficult to know how to decide between equal degree nodes. We have a couple of clues, however. Pueyo [27] used a pre-ordering beginning at the outflow and proceeding to the inflow. For this reason, we try choosing downwind nodes before upwind. Experiments have also shown that choosing a root node at the far-field is often a better choice than at the body. Extrapolating on this idea, we will use the distance from the body to decide. Note that we are using the *reverse* Cuthill-McKee method, so the root node used will be the last node in the permutation. Selecting a root node is particularly difficult, as what makes a good root is not clear. A bad root, however, is easy to spot, since the condest of the preconditioner formed will be quite high. Using this idea, Figures 5.16 and 5.17 show the condest of the preconditioner formed vs. the angle which the position of the root node makes with x-axis. The origin is taken to be the leading edge. Thus, a node directly downstream of the body has an angle of $0°$, while a far-field node upstream of the body is either $\pm 180°$. Choosing the leading edge as the point of reference results in the nodes on the body having a range of only $-90°$ to $90°$. This example is the turbulent single-element subsonic case, part way through the convergence history. Figure 5.16 shows the root nodes at the far-field, while Figure 5.17 shows root nodes at the body. Each of the subfigures uses a different criterion for choosing between equal degree nodes. Note that there are a number of nodes



(a) Downstream preferred

(b) Upstream preferred

(c) Closer to body preferred

(d) Further from body preferred

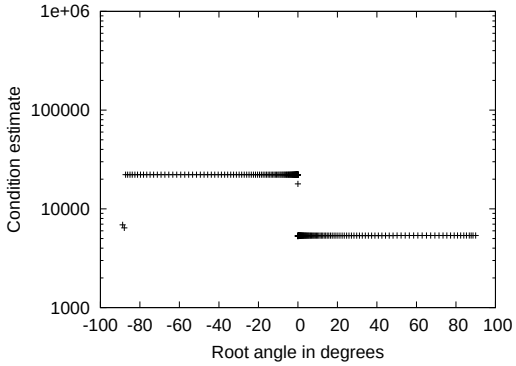
Figure 5.16: Condest with different far-field root nodes and selection schemes

with very high condition estimates that are off the scale of graphs.

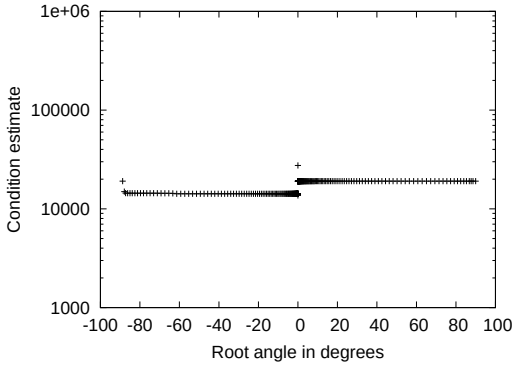
It is interesting to note that the method of deciding between equal degree nodes is not terribly important, especially with a downstream root node. This is possibly due to the limited number of times this choice actually needs to be made. In the process of reordering over 40,000 nodes, on the order of tens to hundreds of equal degree nodes are seen.

Although a high condest will certainly guarantee a poor choice, picking the lowest value does not ensure the best choice. Figures 5.18 through 5.23 show the required CPU time against the root choice. Any instances which lie on the x-axis indicate cases that either did not converge, or were taking too long to be considered. The results demonstrate that a node downstream of the body (i. e. at a root angle near zero) is a safe choice. The turbulent cases are much more sensitive to the root choice.

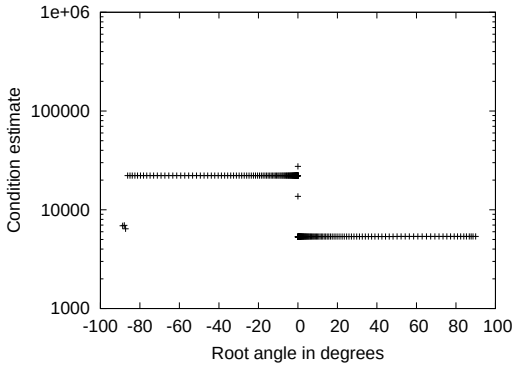
The reason for one root node being better than another is not clear. Examining the total profile size and maximum bandwidth of the reordering shows no correlation with root node quality. One plausible explanation is that equations with poor diagonal dominance are placed later in the ordering, so that numerical errors from small pivots affect fewer equations.



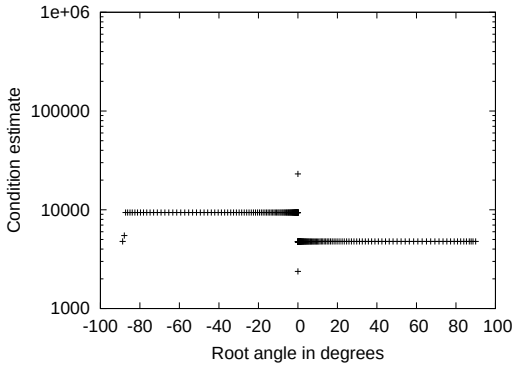
(a) Downstream preferred



(b) Upstream preferred

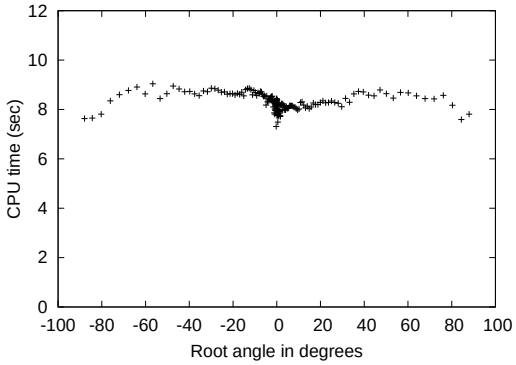


(c) Closer to body preferred

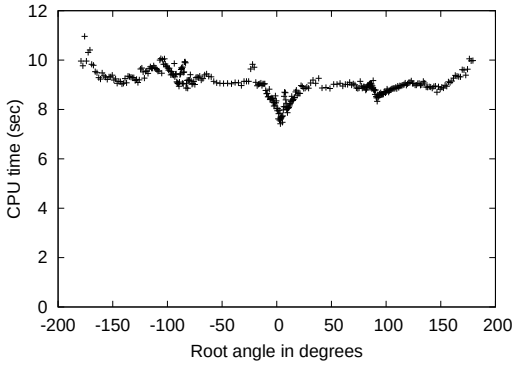


(d) Further from body preferred

Figure 5.17: Condest with different body root nodes and selection schemes

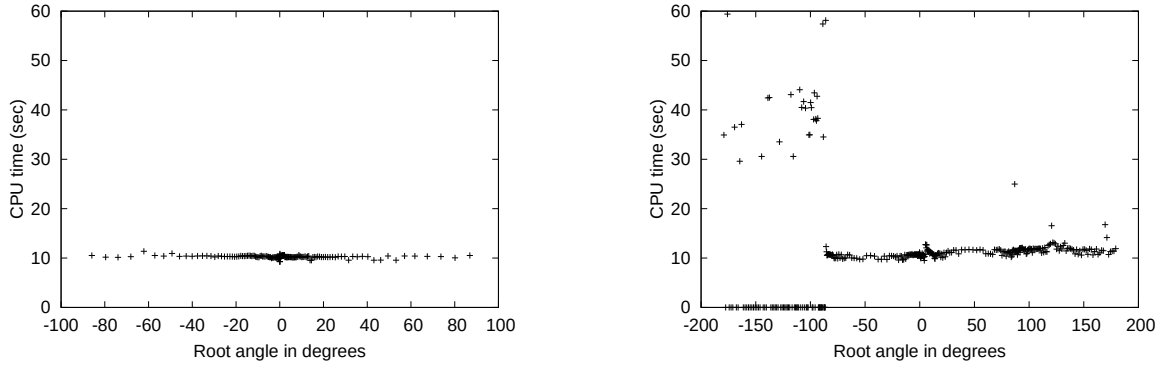


(a) Body root node



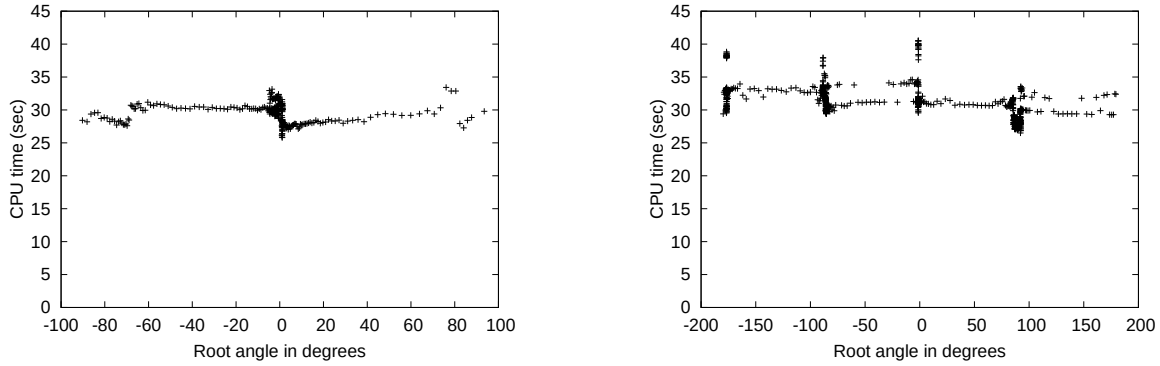
(b) Far-Field root node

Figure 5.18: Inviscid single-element subsonic CPU time against root node



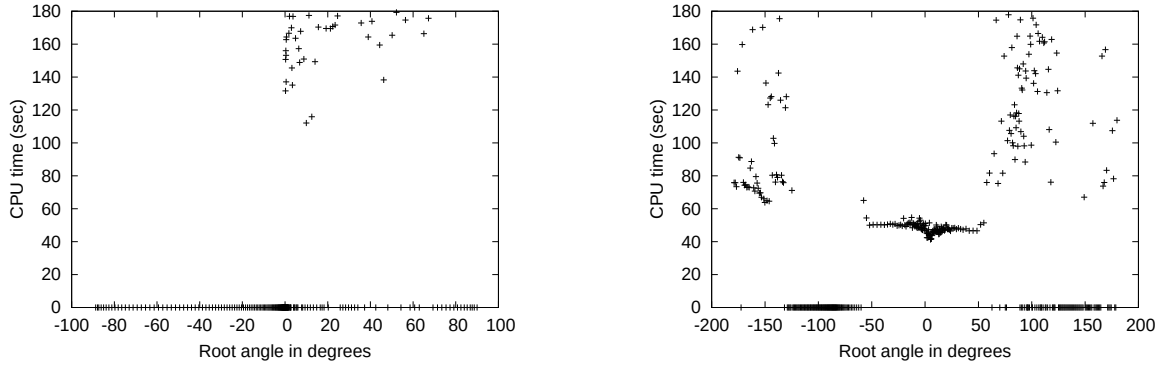
(a) Body root node (b) Far-Field root node

Figure 5.19: Inviscid single-element transonic CPU time against root node



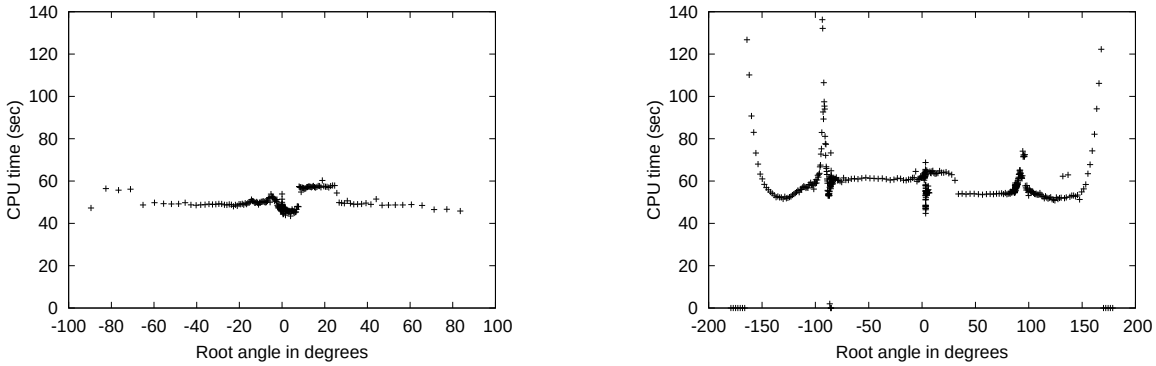
(a) Body root node (b) Far-Field root node

Figure 5.20: Inviscid multi-element CPU time against root node



(a) Body root node (b) Far-Field root node

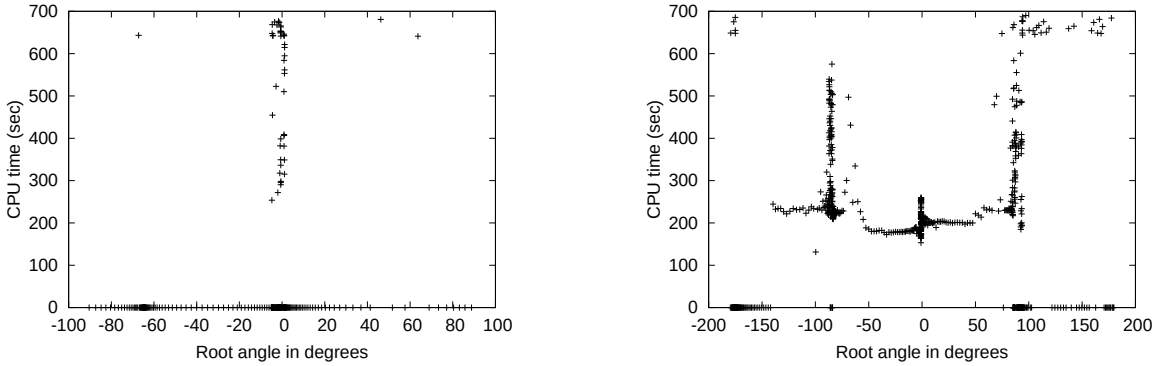
Figure 5.21: Turbulent single-element subsonic CPU time against root node



(a) Body root node

(b) Far-Field root node

Figure 5.22: Turbulent single-element transonic CPU time against root node



(a) Body root node

(b) Far-Field root node

Figure 5.23: Turbulent multi-element CPU time against root node

5.7 Approximate Solution of the Linear System

Solving the linear system approximately has a couple of pitfalls which must be avoided for robustness. They happen almost exclusively with turbulent flows. The first problem occurs when there is a disparity between the norms of the residuals of the mean-flow and the turbulence model equations. If the turbulence residual is significantly higher, as is common in the first few iterations after interpolating onto a grid, the linear solver need only reduce the turbulence residual to achieve the required reduction in the global residual norm. In fact, it is not uncommon to see an increase in the residual of the mean-flow equations. This can destabilize the solution, especially with complex flows, the presence of a shock, or early in convergence. There are two ways of handling this. The first, discussed in Section 5.2 uses residual scaling to bring the magnitude of the turbulence model residual to within an order of magnitude of that of the mean-flow equations. The second method uses a minimum number of linear iterations. This is obviously a blunt tool since it depends on having a reasonable scaling. However, it has the advantage that it is active only early in the convergence, when the time step is low. It is effective when problems are had with mean-flow stability in the first few iterations. If the residual vector is examined in situations where this approach is useful, locally large values of the turbulence model residual will often be found. These local peaks are not addressed by scaling, which only effects the entire turbulence model residual, but does not adjust scaling on a node-by-node basis. The residual norm can thus be decreased by reducing the peak value, often in only one or two linear iterations. We suggest a standard minimum of five linear iterations. The number of linear iterations performed usually does not drop below five, even with small time steps, so this is rarely enforced. When it is, however, it is usually needed.

The second problem occurs with the matrix-free method, and is related to the first. Recall that the Jacobian-vector product is approximated by a forward-difference

$$\mathcal{A}v \approx \frac{\mathcal{F}(\hat{\mathcal{Q}} + \varepsilon v) - \mathcal{F}(\hat{\mathcal{Q}})}{\varepsilon} \tag{5.12}$$

with

$$\varepsilon \|v\|_2 \approx \sqrt{\delta} \tag{5.13}$$

Some authors have suggested using $\delta = \varepsilon_m$ where ε is the value of machine zero.

There are two competing factors in choosing δ to maximize accuracy. The value must be small enough that truncation error is minimized. Too small a choice will induce round-off errors. The difference in the numerator must be larger than a certain portion of the residual, and εv must be a reasonable portion of $\tilde{\mathcal{Q}}$. The difference in the numerator depends on the matrix, as well as the vector v.

The problem, which we call matrix-free breakdown, arises when there are large scaling differences in the components of v. Equation 5.13 will choose ε appropriate for the largest components of vector v. Large round-off errors can then result from the very small components. This is regularly witnessed when solving a system with both the mean-flow and turbulence model equations, especially with trip terms. The manifestation can be subtle and is easily attributed to the nonlinear portion of the algorithm. This is because the linear solver will converge when this problem occurs. Effectively, a different matrix is being solved. The GMRES solver may show little sign of difficulty because the linear residual calculated during each iteration incorporates this error. However, if the true linear residual is calculated after the Krylov method finishes, the residual of the mean-flow equations is seen to have increased by one or two

orders of magnitude. Another symptom is an increase in number of inner iterations, resulting from the preconditioner no longer representing the Jacobian that is effectively being solved. This increase may be small if it occurs with a small time step, or it may be quite large later in convergence.

Note that in the case of GMRES, v is difficult to predict, except for the first iteration. Also, the preconditioner has been applied, so that in the first iteration $v = M^{-1} \cdot \mathcal{S}_r \cdot R$. This makes it difficult to use scaling to correct, since the preconditioner cancels out the row scaling.

In order to find an appropriate value for δ, it would be ideal to compare the result of equation 5.12 with the exact matrix-vector product. However, an exact matrix is not available. Instead, we compare against a higher-order approximation. sixth-order is sufficient to compare both first-order and second-order approximations. The sixth-order evaluation also requires a value for δ, which we call δ_{ref}, so we examine a three-dimensional plot to determine the optimal value for this reference step size. Figure 5.24 shows an example. The z-axis shows the average of norms of the difference between the first order and sixth-order evaluations for one outer iteration. The error vectors have been normalized by $||\mathcal{A}v||$, and the average is taken over the inner iterations. The x-axis is the step size of the first-order approximation, while the y-axis is the step size of the sixth-order approximation. Figure 5.24 is used to choose the value of δ_{ref} from the y-axis. The choice of δ_{ref} should minimize the error in the sixth-order approximation. The error in the sixth-order approximation should then be significantly lower than the error in the first-order method. When the plot is constant in a region along the y-axis, we have a good choice for δ_{ref} in the centre of the region. In Figure 5.24, this occurs from $\delta_{ref} = 10^{-10}$ to $\delta_{ref} = 10^{-8}$.

Figure 5.25 shows the error against δ for the first-order method, after an appropriate value of δ for the sixth-order method was determined. Note that this method is far from perfect, and should be used only as a guideline. For example, strong nonlinearities, with large higher derivatives, will affect the accuracy of the sixth-order method. However, we can certainly see that we should at least try values of δ higher than the square root of machine zero, which is the value recommended by Nielsen et al. [53].

Figure 5.26 shows similar results for the turbulent cases, using a second-order method instead of the first order. As expected, the optimal step-size is somewhat larger than the first-order case. We have experimented with switching to a second-order matrix-free method if the first-order matrix-free breaks down. We have found it is much better to try to prevent the breakdown, instead of dealing with it if it does happen. However, with more complex systems than the one discussed here, one may consider the second-order method.

The plots show three lines, where the results are found at different times during convergence. The results show that good choices of δ are $\delta = 10^{-12}$ for the inviscid cases, and $\delta = 10^{-10}$ for the turbulent. Using these values, far fewer matrix-free breakdowns are seen, especially with the turbulent cases.

5.7.1 Linear Solution Tolerance

The linear solution tolerance is a parameter that needs to be determined empirically. There are two factors that translate the linear tolerance into efficiency. First, the manner in which the linear system converges determines whether if is more efficient to perform more linear iterations. Figure 5.27 shows the linear convergence history for the multi-element cases, at a few different outer iterations. At the beginning of convergence, there is a small plateau, most noticable in the inviscid case, before settling

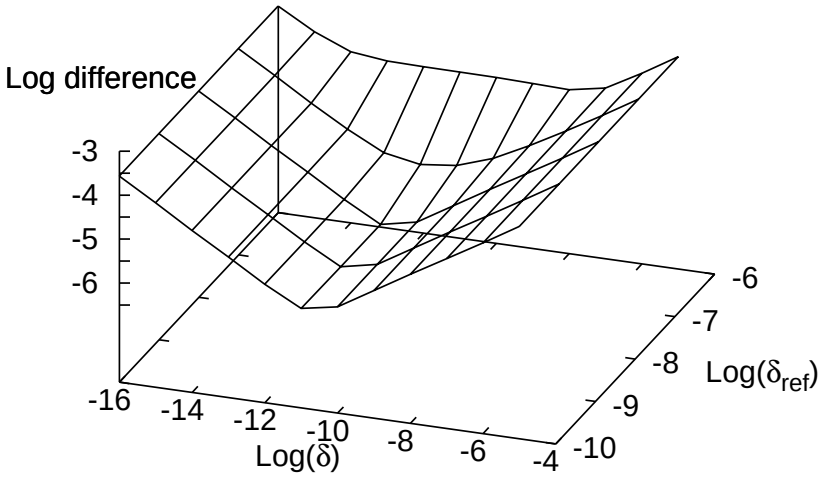


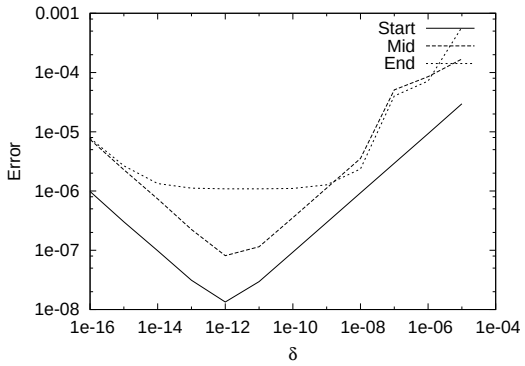
Figure 5.24: Using higher order method to determine optimum δ

into a linear decrease. Thus there is a penalty for looser tolerances.

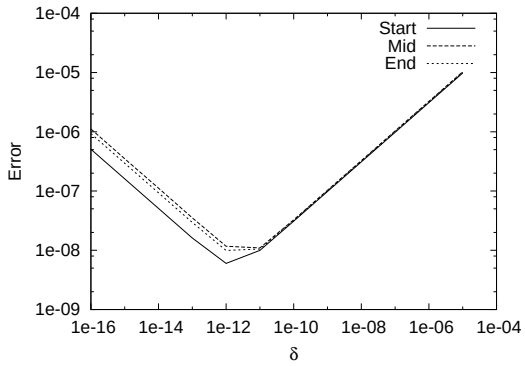
A perhaps more significant effect is the linearity of the nonlinear system. It does no good to oversolve the linear system if it is not a good representation. This will change as nonlinear convergence proceeds. Figure 5.28 shows convergence histories with three different choices in linear tolerance. The tighter tolerances show an acceleration as the residual decreases, versus the steady decline of the tolerance of 0.5. The optimal value is 0.1. It appears that tightening the tolerance of the linear solver as the system converges would be beneficial. To test this, the same examples were repeated, beginning with an inner tolerance of 0.1, then switching to 0.01 after the L_2 norm is less than 10^{-4}. Figure 5.29 shows the results. The lack of benefit to switching is interesting. Likely, the tighter tolerances are reducing error earlier in convergence, which are not reflected by an immediate decrease in residual.

5.8 Matrix-free vs Matrix-explicit GMRES

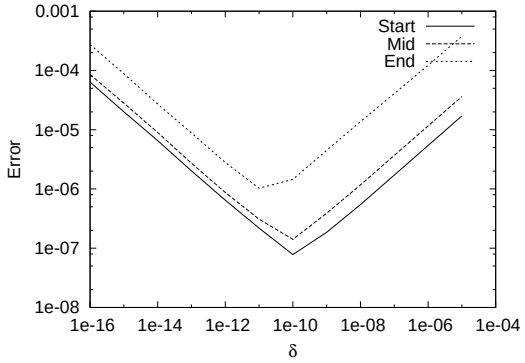
As discussed in Section 4.2.1, GMRES does not explicitly need the Jacobian matrix, but instead may use a Frechet derivative to supply the Jacobian-vector multiply required. There is a trade-off in speed between matrix-free and matrix-explicit GMRES, which depends on the number of linear iterations. The former requires one residual evaluation per iteration. The latter requires a matrix construction when beginning the linear solve, plus a matrix-vector multiply per iteration. Since the matrix-vector multiply is cheaper than a residual evaluation, matrix-free GMRES becomes less efficient with more difficult systems. Figure 5.30 compares these two methods. The inviscid single-element cases favour the matrix-free method. This is due in large part to the low number of iterations, the use of the circulation correction, which is not represented in the Jacobian, and the presence of the shock in the second case. The shock activates the pressure switch, which is not linearized in the explicitly calculated Jacobian. A



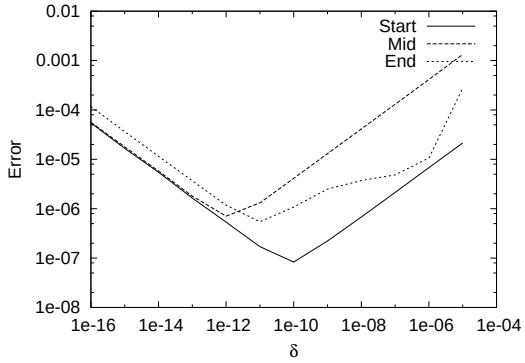
(a) Inviscid single-element transonic



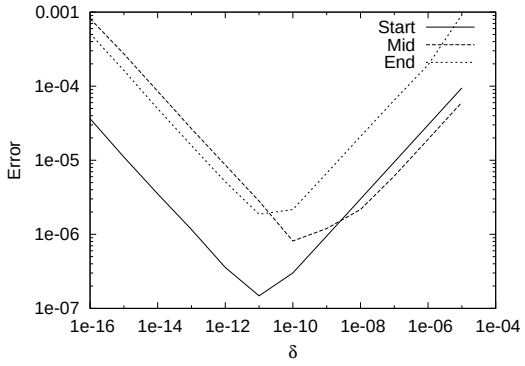
(b) Inviscid multi-element



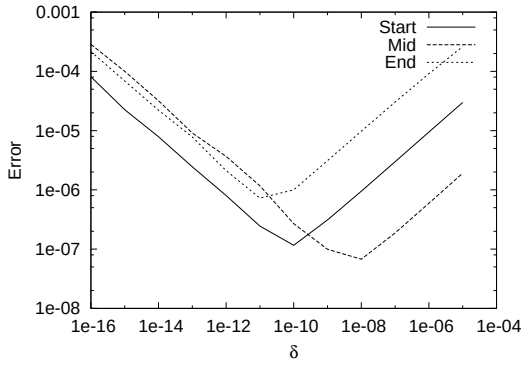
(c) Turbulent single-element subsonic without trip



(d) Turbulent multi-element without trip

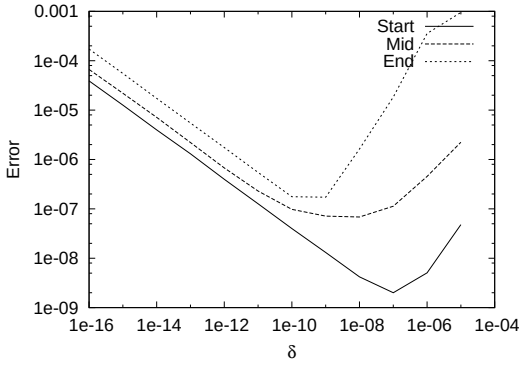


(e) Turbulent single-element subsonic with trip

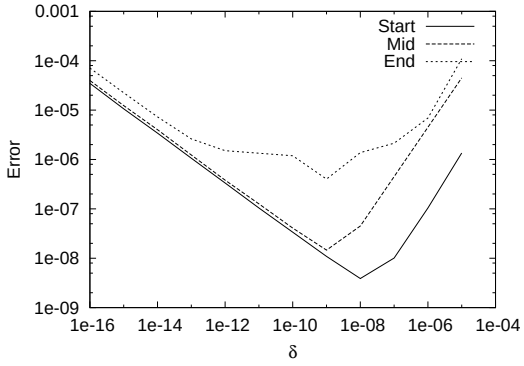


(f) Turbulent multi-element with trip

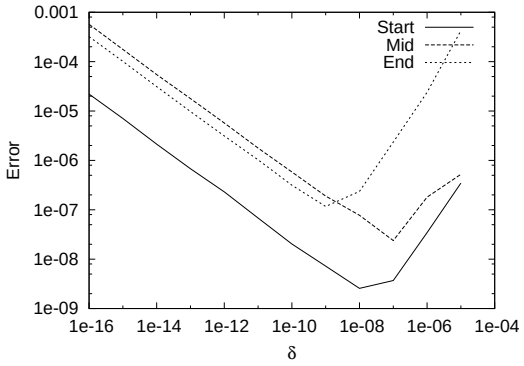
Figure 5.25: Approximate first-order matrix-vector error against δ



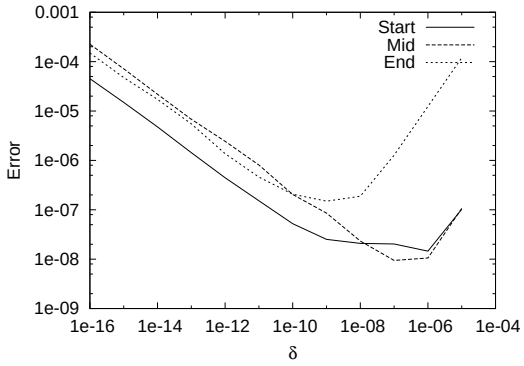
(a) Turbulent single-element subsonic without trip



(b) Turbulent multi-element without trip

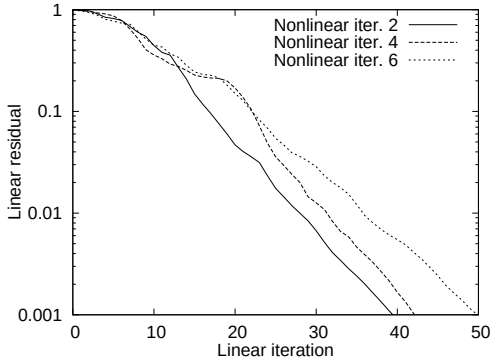


(c) Turbulent single-element subsonic with trip

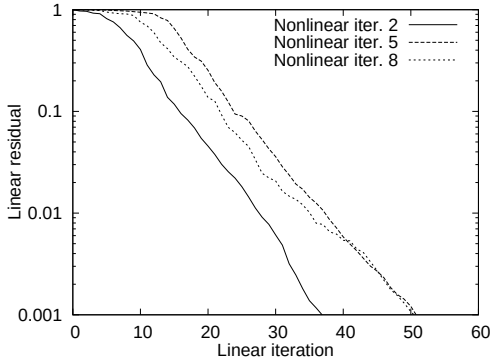


(d) Turbulent multi-element with trip

Figure 5.26: Approximate second-order matrix-vector error against δ

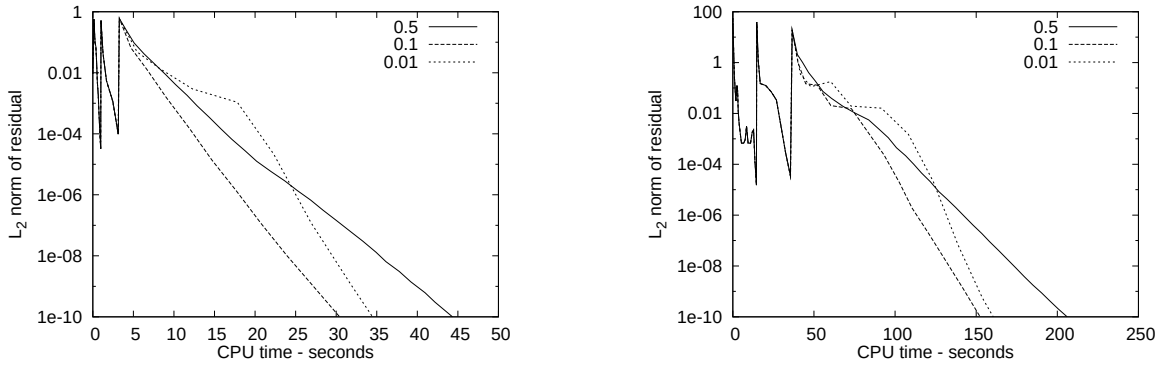


(a) Inviscid multi-element



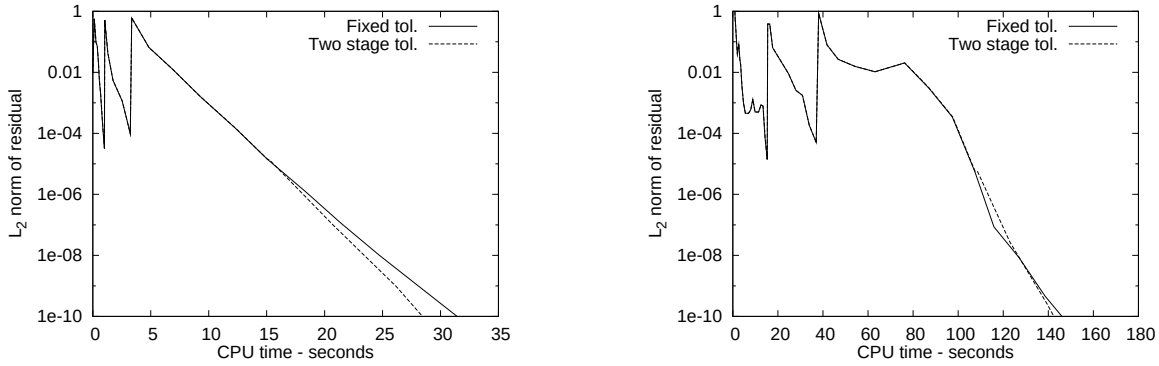
(b) Turbulent multi-element

Figure 5.27: Linear convergence at different outer iterations



(a) Inviscid multi-element (b) Turbulent multi-element

Figure 5.28: Effect of linear tolerance on convergence



(a) Inviscid multi-element (b) Turbulent multi-element

Figure 5.29: Effect of varying linear tolerance on convergence

similar trend is seen with the single-element turbulent cases, especially the transonic case. The multi-element cases show the matrix-explicit method being much closer to the matrix-free method. This is explained by the higher number of inner iterations required.

Matrix-free is clearly the best choice, both for the the memory savings, and for the improvement in convergence rate, due to the implicit linearization of terms which are difficult to differentiate. The Jacobian is nearly twice the size of the preconditioner matrix, because it stores a nine-point stencil.

5.8.1 GMRES Restart

A common technique to reduce the memory requirements of GMRES is to restart the algorithm after a set number of iterations. This worked well for Pueyo [27], who used a restart of 20. His system was somewhat simpler, lacking a turbulence model and matrix dissipation. We have found that using restarted GMRES can cause a large increase in the number of inner iterations and so is not recommended.

5.9 Trip Terms

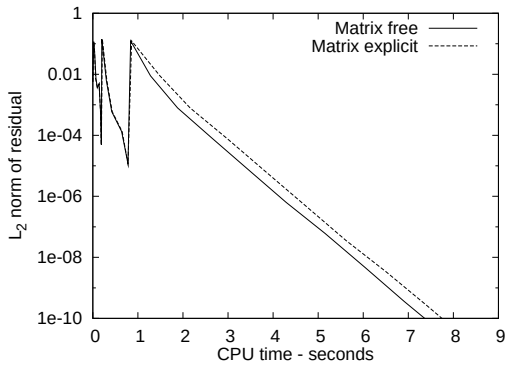
The use of trip terms is strongly recommended by Spalart and Allmaras in order to ensure that laminar-turbulent transition occurs where desired. We follow their suggestion in using a limited number of nodes 'within a reasonable distance of the trip' for the calculation of the first trip function f_{t1}. This simplifies the coding, and speeds calculation of the residual slightly. If the region is chosen properly, the residual should not be affected because of the exponential decrease in f_{t1} with increasing distance from the trip point.

Using the trip terms provides a challenge for the Newton solver. f_{t1} is quite nonlinear and can show extreme sensitivity to the velocities at the trip point. This is due to the exponential nature of f_{t1} and the fact that the vorticity is a derivative of the velocities. To further complicate the issue, the residuals of the mean-flow equations are scaled by J^{-1}, which means that the linear residual near the body may not be reduced, leading to fluctuations in the velocities at the trip, and potentially a large increase in the trip term.

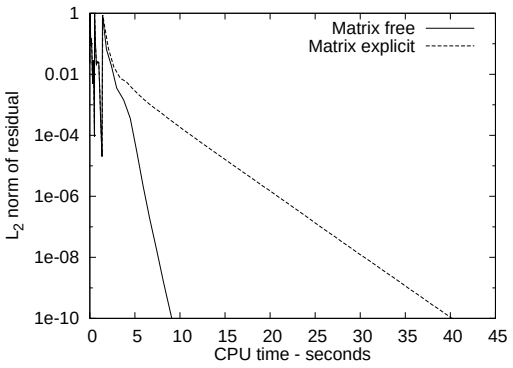
The coupling adds extra entries to the Jacobian matrix. These are important to include if an explicit matrix is being used. They are not desirable in the preconditioner, however, since they affect the reordering (by changing the degree of the nodes) and increase memory requirements, especially with higher levels of fill.

If trip terms are used with no modifications made to the algorithm presented, convergence can be slowed unpredictably as the trip term fluctuates. Measures need to be taken to damp the changes in the turbulence variable near the trip. This is important for robustness as well as efficiency. The changes in the residual can easily be large enough to destabilize the solution. The approach taken is to reduce the time step at the nodes where the trip terms show sensitivity to velocity. We want to keep the rule as simple as possible and to have it work with the reference time step presented in Section 5.3.1. Nodes where

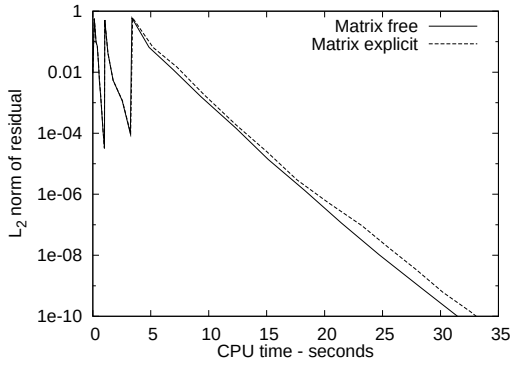
$$\left|\frac{\partial(trip)}{\partial u}\right| + \left|\frac{\partial(trip)}{\partial v}\right| > 10^5 \tag{5.14}$$



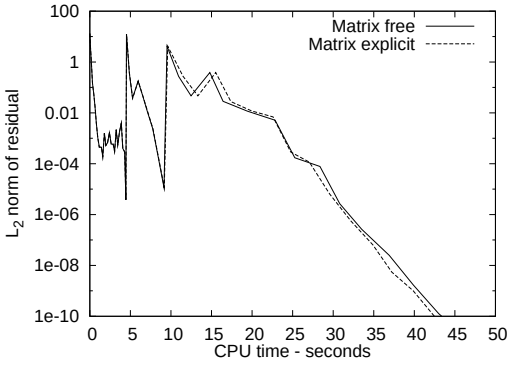
(a) Inviscid single-element subsonic



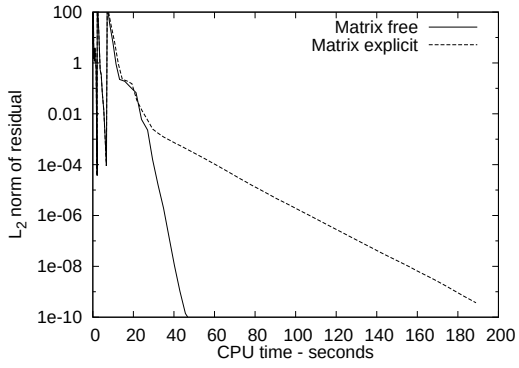
(b) Inviscid single-element transonic



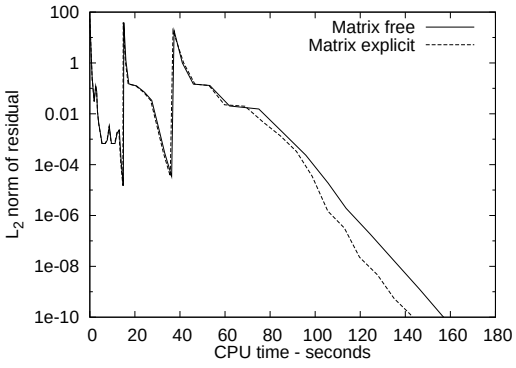
(c) Inviscid multi-element



(d) Turbulent single-element subsonic

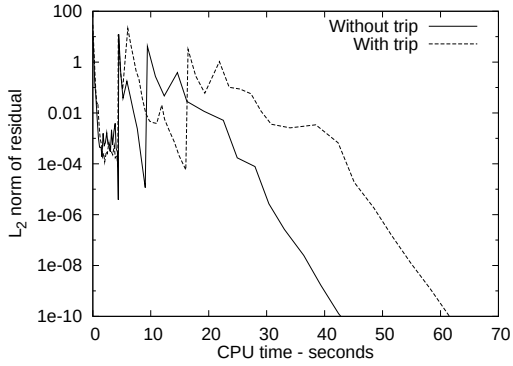


(e) Turbulent single-element transonic

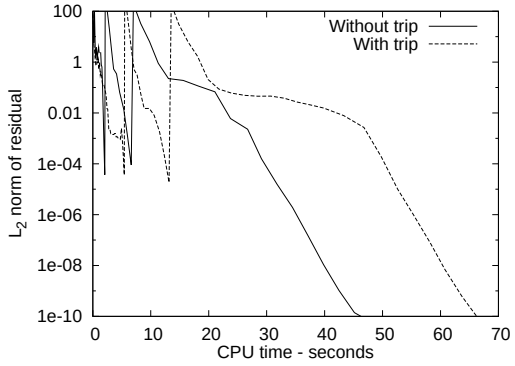


(f) Turbulent multi-element

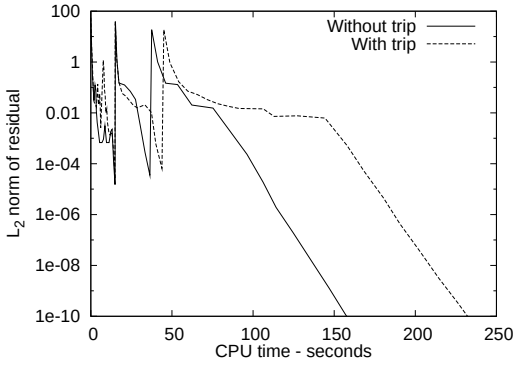
Figure 5.30: Comparison of matrix-free vs. matrix-explicit GMRES



(a) Turbulent single-element subsonic



(b) Turbulent single-element transonic



(c) Turbulent multi-element

Figure 5.31: Effect of trip terms on convergence

have the time step reduced according to the following

$$\Delta t^t_{ref} = \begin{cases} \Delta t_{ref} & : \quad \Delta t_{ref} < 10 \\ 10 & : \quad 10 < \Delta t_{ref} < 1000 \\ \frac{\Delta t_{ref}}{100} & : \quad \Delta t_{ref} > 1000 \end{cases} \tag{5.15}$$

The method used is reasonably insensitive to the limit of the sum of the derivatives, due to the exponential nature of the trip source. The overhead associated with this calculation is reasonably small, since it only needs to be performed on the nodes where the trip terms are considered.

Figure 5.31 compares the turbulent cases with and without the trip terms. These cases show roughly a 50% increase in the time required to converge when the trip terms are present.

Chapter 6

Results

In this chapter we compare the performance of Scirocco with two approximately-factored solvers, Cyclone and TORNADO, which provide a useful baseline for comparison. Cyclone is used for single-element cases, TORNADO for multi-element cases. We present the comparison for the six cases used for the optimization, as well as a number of additional cases, to demonstrate that the method is appropriate for a variety of cases. We also highlight one of the strengths: the ability to converge quickly from a 'warm' solution. Table 6.1 lists the airfoils and flow conditions for all the cases presented in this chapter, including those presented in the optimization chapter. Table 6.2 gives some information about the grids used. Further details are in the subsequent sections describing each case.

For convenience, we summarize the parameters used by Scirocco in the comparisons. These are the same as described in Chapter 5. Note that although the Newton-Krylov algorithm presented has many parameters that can be adjusted, the results in this chapter were obtained with this single set of parameters.

- The equations are scaled as described in Section 5.2. The mean-flow variables and residuals are scaled by the metric Jacobian. The turbulence model is scaled by 10^{-3}. On the fly rescaling is applied such that the maximum difference is a factor of ten.

- An ILU preconditioner with fill of four is used. It is based on the approximate Jacobian, with $\sigma = 10.0$. The preconditioner is calculated before every outer iteration.

- The preconditioner is reordered using the reverse Cuthill-McKee method. The root node is chosen

Case	Airfoil	Mach	Angle of Attack	Reynolds No.	Note
1	NACA0012	0.15	6.0°	—	Optimization case
2	NACA0012	0.7	1.49°	—	Optimization case
3	NLR	0.2	5.0°	—	Optimization case
4	NACA0012	0.15	6.0°	$9 \cdot 10^6$	Optimization case
5	RAE	0.729	2.31°	$6.5 \cdot 10^6$	Optimization case
6	NLR	0.3	6.0°	$2.51 \cdot 10^6$	Optimization case
7	NACA0012	0.15	6.0°	$9 \cdot 10^6$	Hyperbolic grid
8	NLR	0.3	6.0°	$2.51 \cdot 10^6$	Blunt trailing edges
9	AGARD A2	0.197	20.18°	$3.52 \cdot 10^6$	Three-element airfoil
10	NLR	0.185	0° – 19°	$2.51 \cdot 10^6$	Determination of max. C_l
11	NLR	0.25	4.0°	$2.51 \cdot 10^6$	Numerical optimization simulation

Table 6.1: Test cases

Case	Airfoil	Dimensions	Number of nodes	Offwall	Max aspect ratio
1	NACA0012	245×49	12005	$5 \cdot 10^{-4}$	9211
2	NACA0012	245×49	12005	$5 \cdot 10^{-4}$	9211
3	NLR	—	27019	$5 \cdot 10^{-4}$	1913
4	NACA0012	305×57	17385	10^{-6}	13672
5	RAE	305×57	17385	$2 \cdot 10^{-6}$	8483
6	NLR	—	44059	10^{-6}	25514
7	NACA0012	305×57	17385	10^{-6}	2462858
8	NLR	—	40257	10^{-6}	510577
9	AGARD A2	—	71868	10^{-6}	18434
10	NLR	—	44059	10^{-6}	25514
11	NLR	—	30190	10^{-5}	12755

Table 6.2: Optimization test case grid characteristics

to be downstream of the main airfoil.

- The matrix-vector product is calculated with the matrix-free method. The perturbation used is $\delta_{ref} = 10^{-12}$ for inviscid cases, and $\delta_{ref} = 10^{-10}$ for turbulent cases.
- The linear system residual is reduced by a factor of ten.
- The parameters used in the reference time step (equation 5.3) are:
  - $\alpha = 1000$ for inviscid flows, $\alpha = 100$ for turbulent.
  - $\Delta t_{min} = 1000$ for inviscid flows, $\Delta t_{min} = 100$ for turbulent.
  - $\beta = 1$
- The new local time step for the turbulence model is used. The limiting ratio was set to 0.5

The parameters used by the approximately-factored solvers are:

- A reference time step of 5.0 is used on all levels.
- Implicit handling of the wake cut is not used.

This optimizes efficiency for these solvers.

The results are presented in two ways. The first is the standard norm of the residual. For the turbulent cases, the residuals of the mean-flow and turbulence model equations are shown separately. Normally the mean-flow residual calculation done in the approximately-factored algorithms do not include any scaling. To provide a fair comparison, we scale the residual vector before the norm is evaluated. The second graph shows the error in the coefficient of lift. The error is the difference from the converged lift coefficient. The convergence of this may not be smooth looking, if the lift coefficient happens to pass through the final value while converging. When this happens, there is a narrow valley, but the overall trend is still easy to discern.

Two x-axes are used. The top shows the CPU time used. The bottom is the equivalent right hand side evaluations. This is found by dividing the CPU time by the time it takes to perform one residual evaluation. This evaluation includes the mean-flow equations, dissipation, the turbulence model and all boundary conditions. The residual evaluation time is quite close for both the approximate-factorization solvers and Scirocco. Table 6.3 shows the times for each of the test cases used in the optimization.

In many of the cases presented, particularly the multi-element cases, the residual of the approximate-factorization solver will fail to converge to 10^{-10}. However, in most of these cases, the coefficient of lift converges, although this does not inspire confidence in the final result. Typically, only one or two nodes have not converged though. These nodes are often close to the body. When the residual is unscaled, the contribution from these nodes is small, so that it appears to converge farther. However, in these cases, scaling is applied, so that any unconverged nodes are obvious.

6.1 Memory requirements

Scirocco takes a significant amount of memory, mostly due to the preconditioner requirement. Table 6.4 gives a breakdown of the memory usage for each case. The amount of memory needed for the Jacobian

	Without trip		With trip	
Case	Scalar diss.	Matrix diss.	Scalar diss.	Matrix diss.
1	0.008	0.016	—	—
2	0.008	0.017	—	—
3	0.016	0.039	—	—
4	0.052	0.065	0.058	0.071
5	0.052	0.064	0.057	0.072
6	0.099	0.13	0.12	0.15

Table 6.3: Right hand side evaluation time (seconds)

Case	Nodes	Jacobian	Preconditioner	Krylov subspace	Matrix-free total
1	12005	14Mb	21Mb	23Mb	44Mb
2	12005	14Mb	21Mb	23Mb	44Mb
3	27019	31Mb	48Mb	51Mb	99Mb
4	17385	31Mb	48Mb	42Mb	90Mb
5	17385	31Mb	48Mb	42Mb	90Mb
6	44059	78Mb	123Mb	105Mb	228Mb

Table 6.4: Memory requirements

is also included, although when the matrix-free approach is used, this extra memory is not required. The requirements for the approximate Jacobian are not shown, since it is overwritten when the preconditioner is formed. The last column shows the total memory needed, not including the Jacobian. While these requirements are quite a lot higher than the needs of the approximate-factorization solver, they certainly do not exceed the capabilities of a desktop computer.

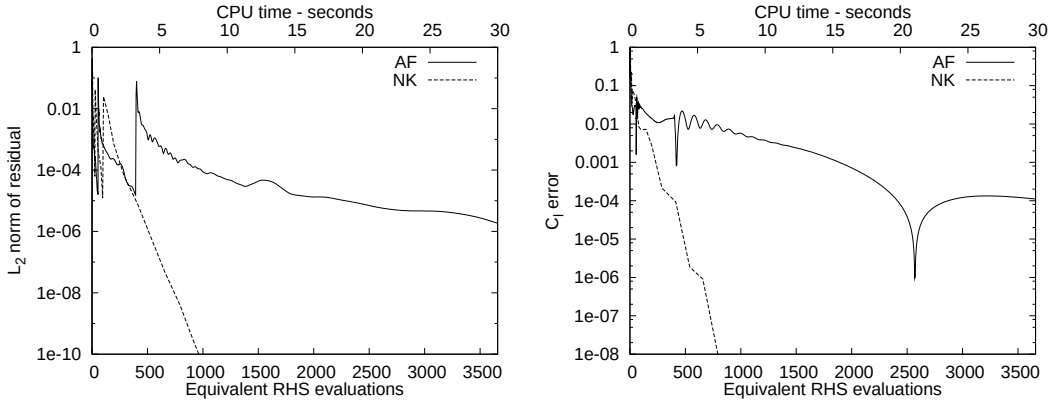
6.2 Comparisons

6.2.1 Previous Examples

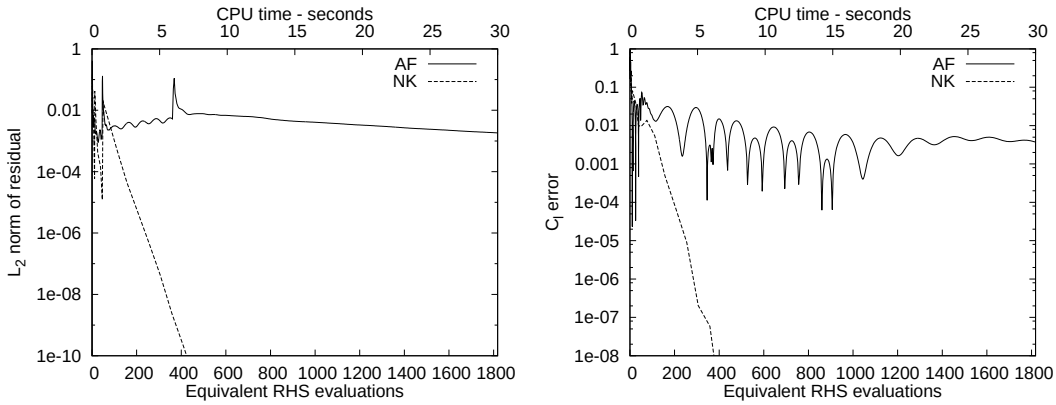
Figures 6.1 through 6.6 show the results from the cases used in the previous chapter. Scirocco is significantly more efficient than the approximate-factorization solver, especially when matrix dissipation is used. When matrix dissipation is used, around 1000 right hand side evaluations are required without trip terms, and about 2000 with.

6.2.2 Hyperbolic Grid

The next example uses the same flow conditions and airfoil as case 4, the turbulent, subsonic flow on a NACA0012 airfoil. However, instead of the grid being generated with an elliptic method, a hyperbolic algorithm is used [76]. This method is much simpler, requiring less user expertise. The user has significantly less control over grid spacing, however. Also, the technique is not suitable for multi-block grids. The grid has the same dimensions as case 4, but has significantly higher cell aspect ratios. The

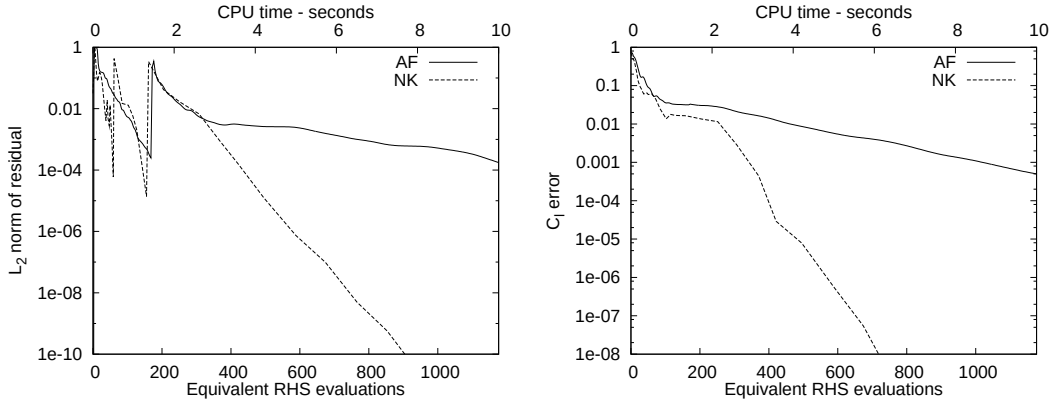


(a) Scalar dissipation

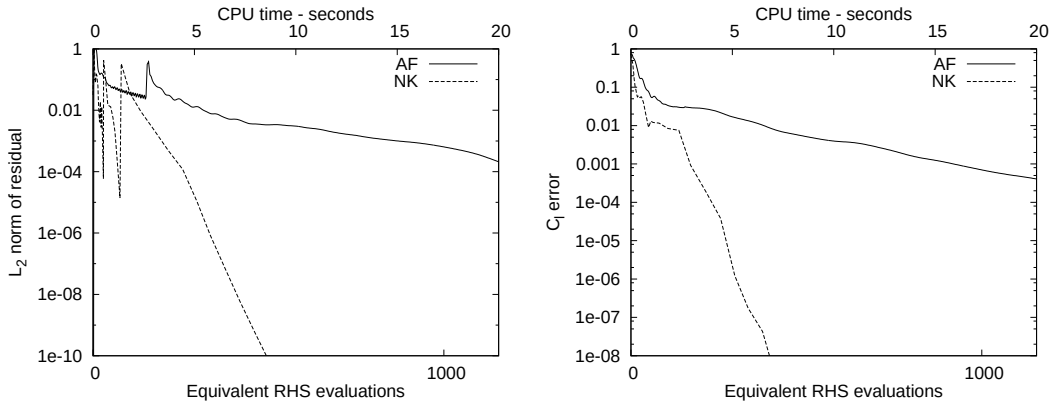


(b) Matrix dissipation

Figure 6.1: Inviscid single-element subsonic results

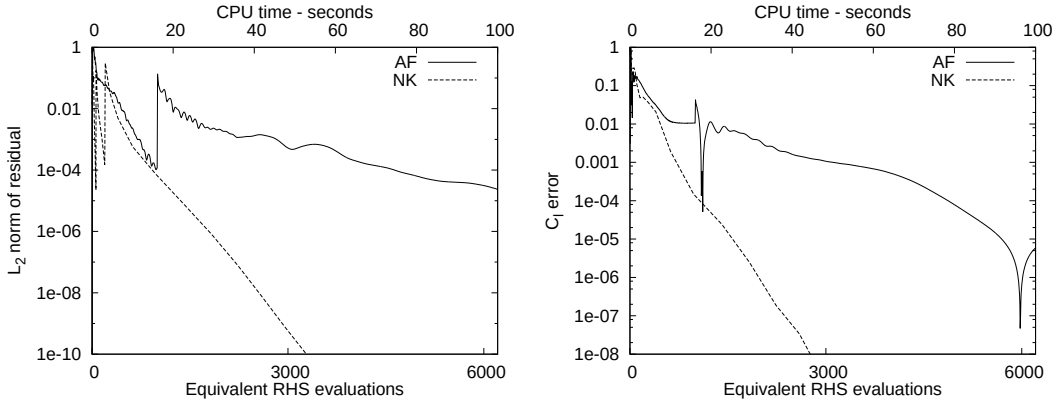


(a) Scalar dissipation

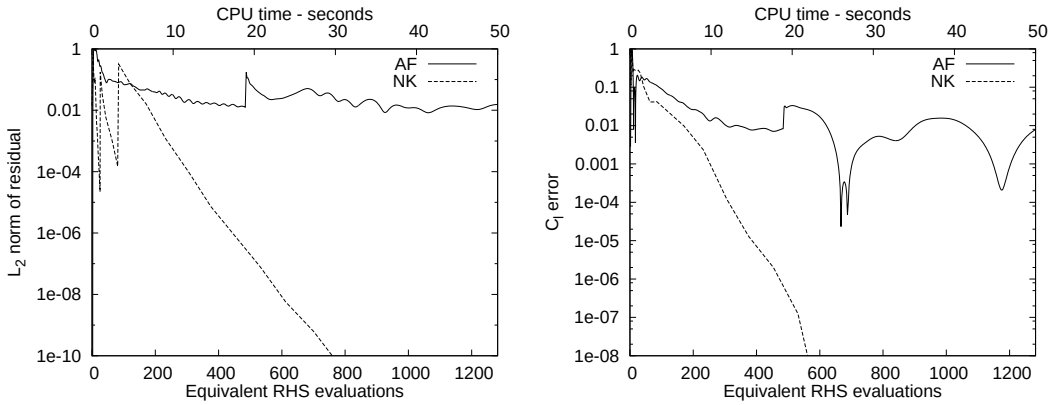


(b) Matrix dissipation

Figure 6.2: Inviscid single-element transonic results

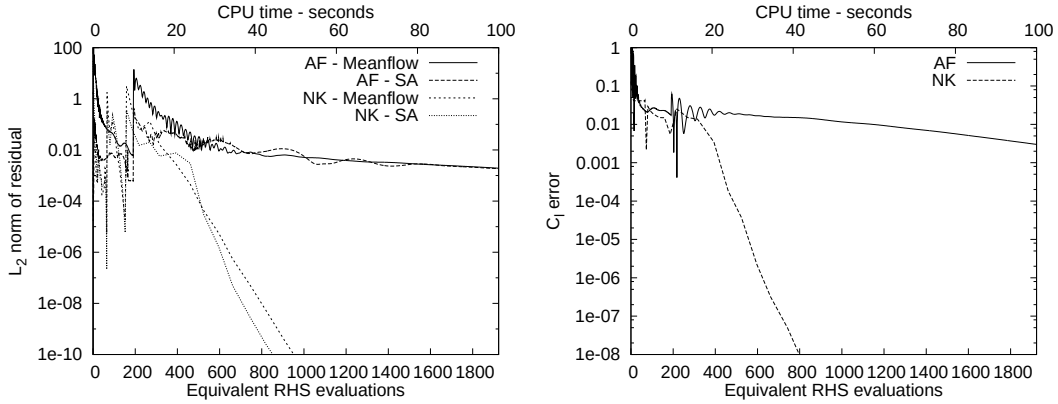


(a) Scalar dissipation

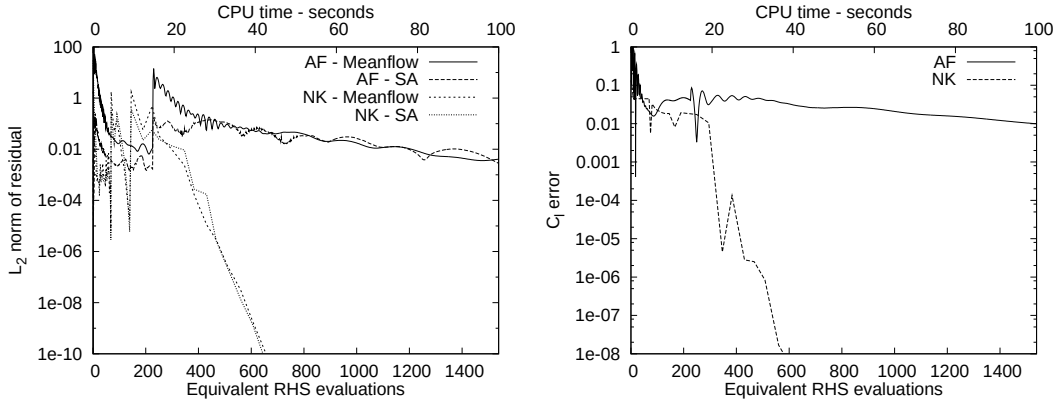


(b) Matrix dissipation

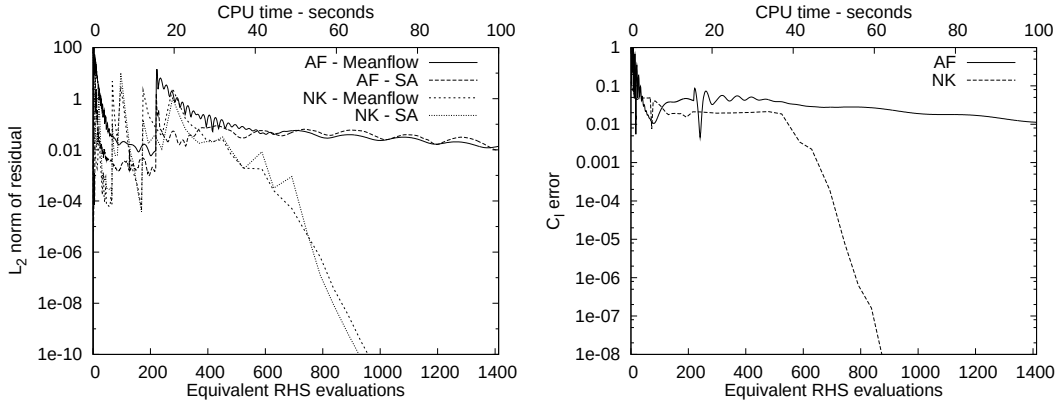
Figure 6.3: Inviscid multi-element results



(a) Scalar dissipation - no trip terms

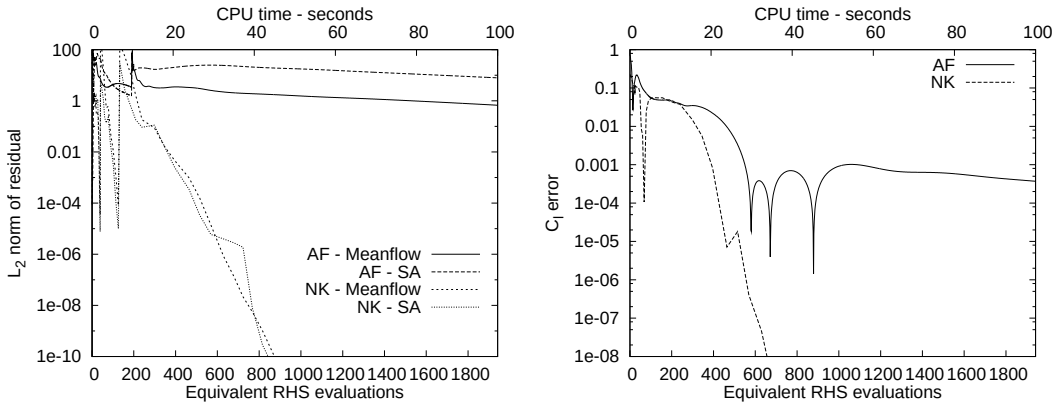


(b) Matrix dissipation - no trip terms

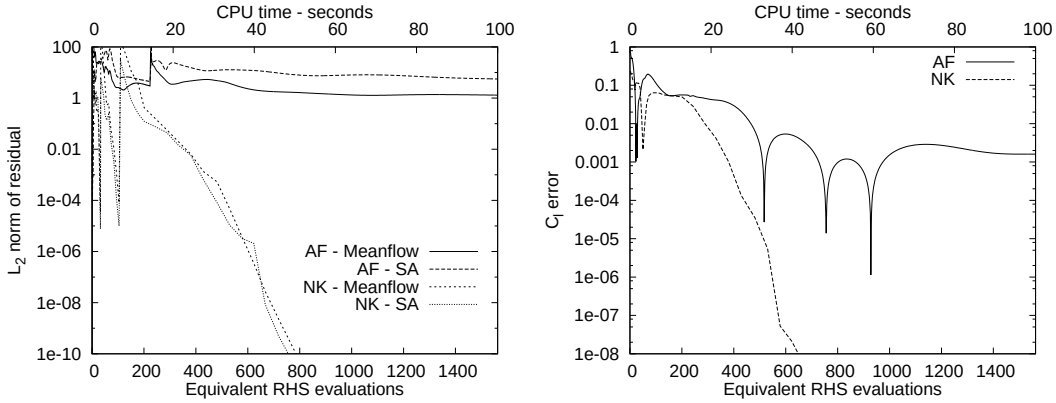


(c) Matrix dissipation - trip terms

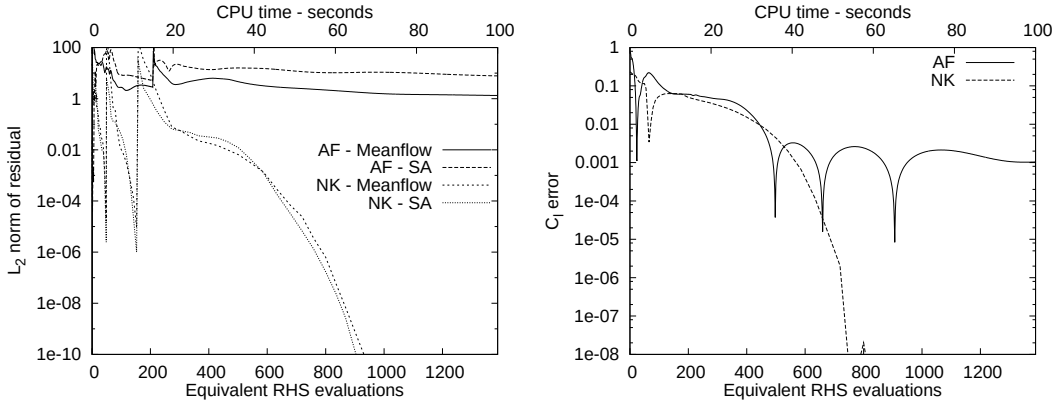
Figure 6.4: Turbulent single-element subsonic



(a) Scalar dissipation - no trip terms

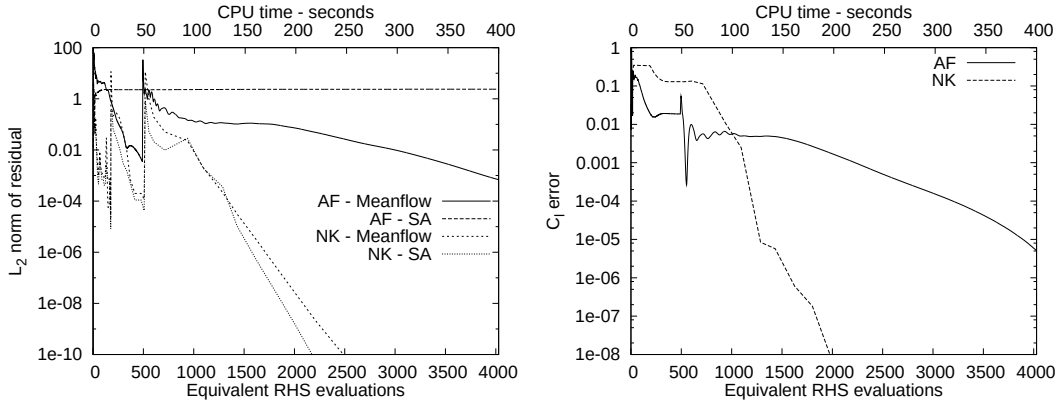


(b) Matrix dissipation - no trip terms

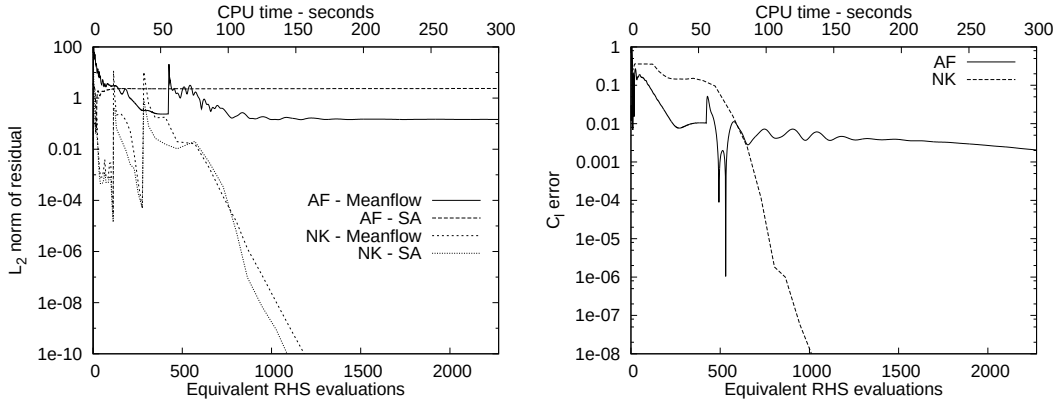


(c) Matrix dissipation - trip terms

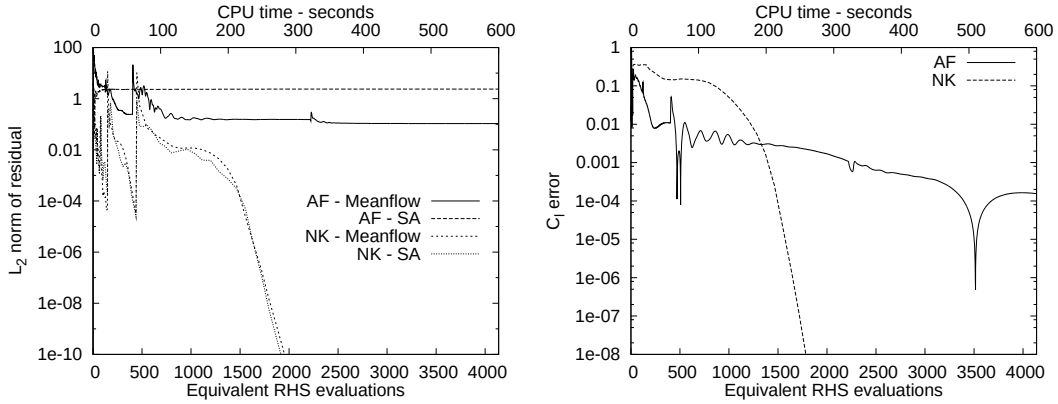
Figure 6.5: Turbulent single-element transonic



(a) Scalar dissipation - no trip terms

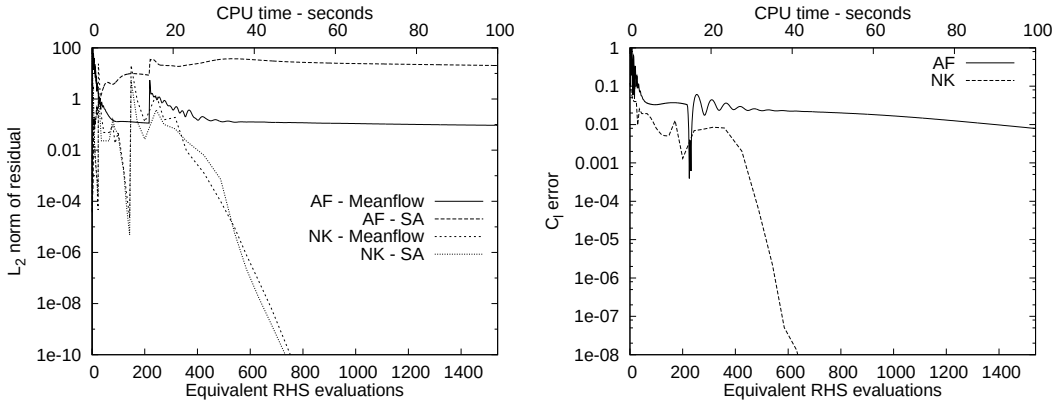


(b) Matrix dissipation - no trip terms

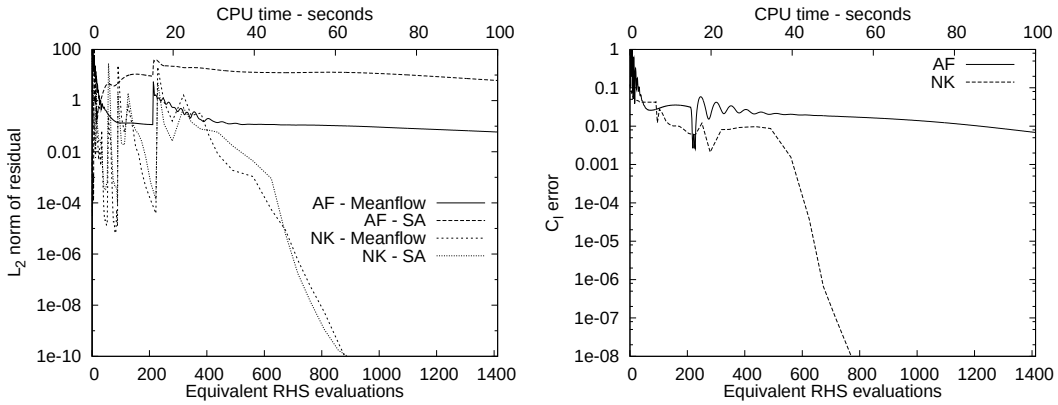


(c) Matrix dissipation - trip terms

Figure 6.6: Turbulent multi-element



(a) Without trip terms



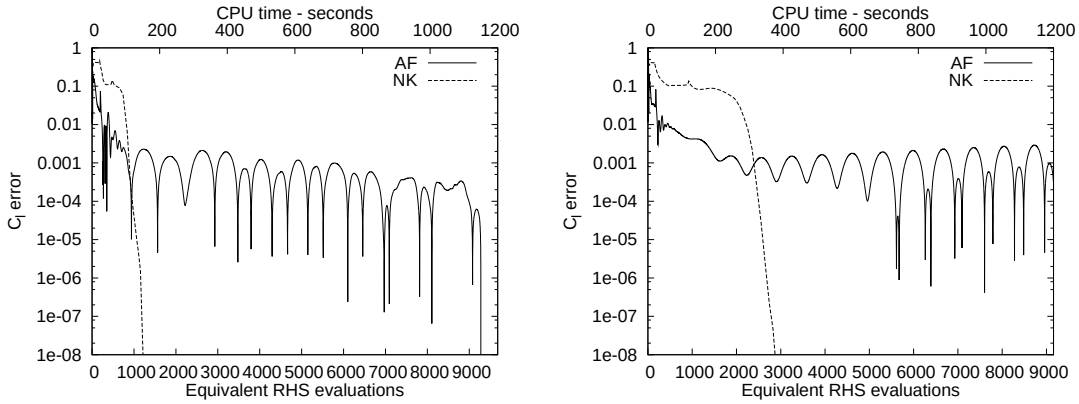
(b) With trip terms

Figure 6.7: Hyperbolic grid results

results are shown in Figure 6.7. The Newton-Krylov method was unaffected by the different grid, while the approximate-factorization convergence suffered, taking twice as long to converge if the residuals are compared. Interestingly, the lift convergence is similar until about the error is about 10^{-4} (not shown on the plot), after which the convergence on the hyperbolic grid slows significantly.

6.2.3 NLR With Blunt Trailing Edges

In industry, the airfoil shapes are commonly supplied as they would be built, that is, with blunt trailing edges. There are two ways to deal with this situation. First, the trailing edge can be closed, giving a sharp edge, as is seen in the other cases studied. The second option is to create a grid with a long thin block behind each such trailing edge. In many cases, this is an easier process than closing the end, and is more realistic in industry. We now show how well the Newton-Krylov method can handle such a grid, which has many high aspect-ratio cells. Figure 6.8 shows the results. The Newton-Krylov method handles the modified grid well, although when the trip terms are used, there is a rather slow start. The approximate-factorization would not fully converge.



(a) Without trip

(b) With trip

Figure 6.8: NLR airfoil with blunt trailing edges results

6.2.4 A2

The A2 case by Moir [77], shown in Figure 2.1, is computationally a particularly difficult case. The three elements and the high angle of attack create very complex flow features, requiring a high resolution grid. We have used a grid with 72,000 nodes. The Mach number is 0.197, the angle of attack is 20°, and the Reynolds number is $3.52 \cdot 10^6$. The results are shown in Figure 6.9. The difficulty of this case is indicated by the increased number of equivalent right hand side evaluations required.

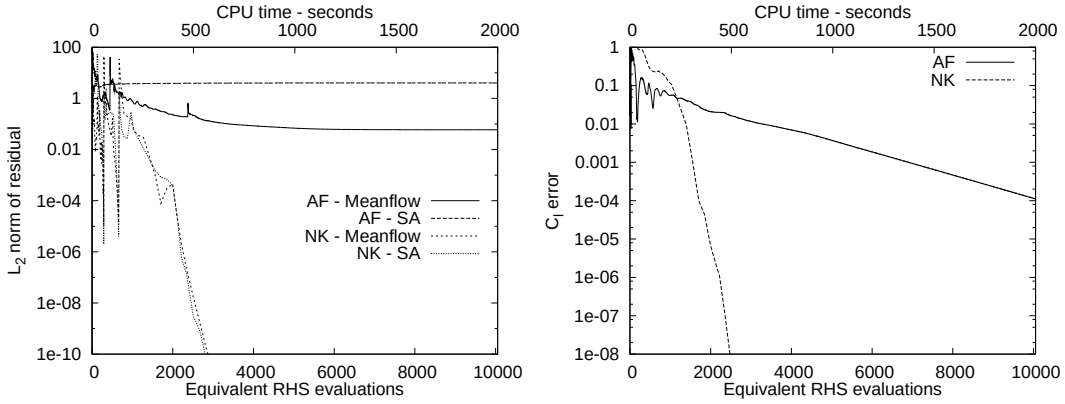
6.2.5 Determination of Maximum Coefficient of Lift

A common practice is to determine the maximum coefficient of lift of an airfoil, and the angle attack at which it occurs. To do this, a profile of C_l vs the angle of attack is produced. Figure 6.10 shows this plot for the NLR airfoil.

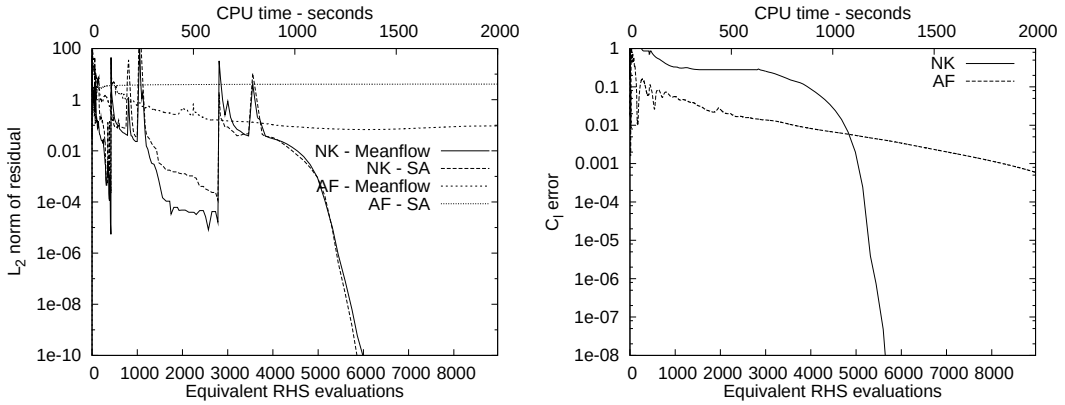
To produce this plot, restarts from the preceding stage were used. The angle was increased by one degree after each stage. This is quite beneficial for the Newton-Krylov method, since the initial guess for all cases but the first is quite good. In fact, for all but the 16° case, the process was just to perform a few iterations with $\Delta t_{ref} = 100$, then switch to a full Newton method. The case at 16° required the regular approach, since at this point, massive separation occurred, so that the preceding case did not provide a good initial guess. Figure 6.11 shows a sampling of the convergence histories.

6.2.6 Optimization Simulation

Similar to the example above, numerical optimization often requires a sequence of flow solves that are similar, allowing the effective use of restarting. The example here is the result of an optimization of flap position on the NLR airfoil. Nine iterations of the optimizer were performed, so that we have nine different grids, each with different flap positions. The sequence of grids provided unfortunately does not allow for grid coarsening, so the first optimization iteration is somewhat slower than usual. After the solution is found on the first grid, the subsequent solutions can be found using only Newton's method.



(a) Without trip terms



(b) With trip terms

Figure 6.9: A2 case results

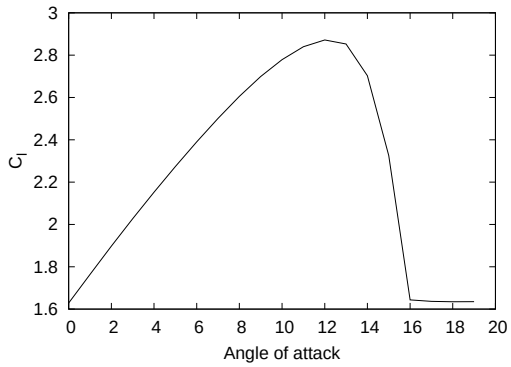
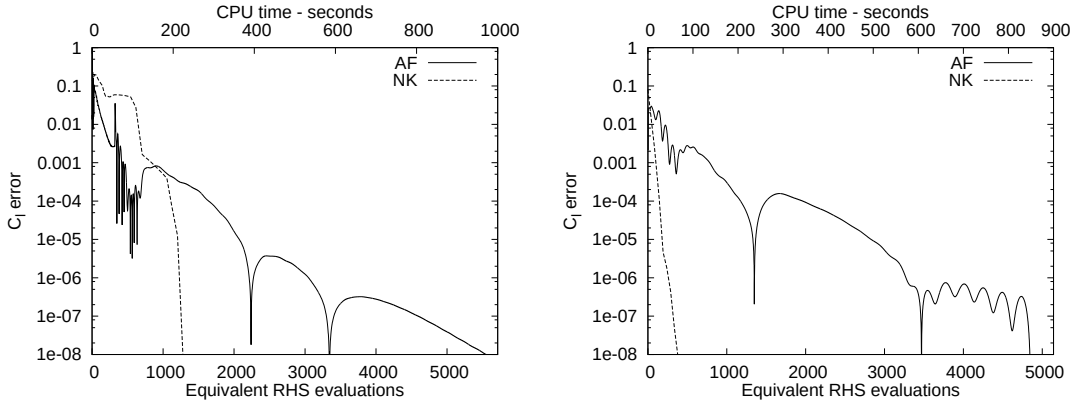
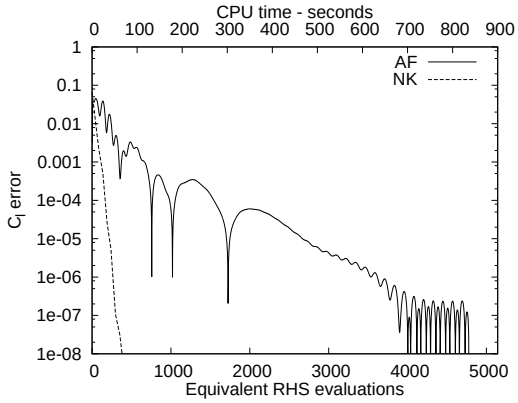


Figure 6.10: Angle sweep lift coefficients

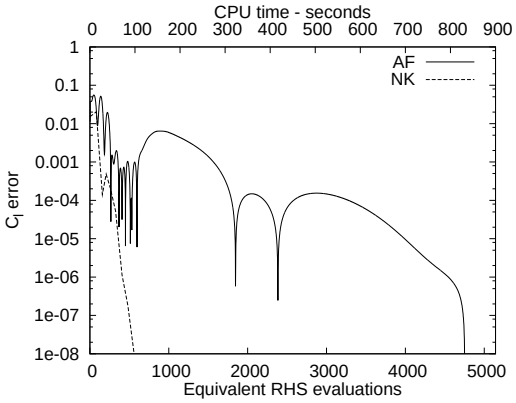


(a) 0 degrees

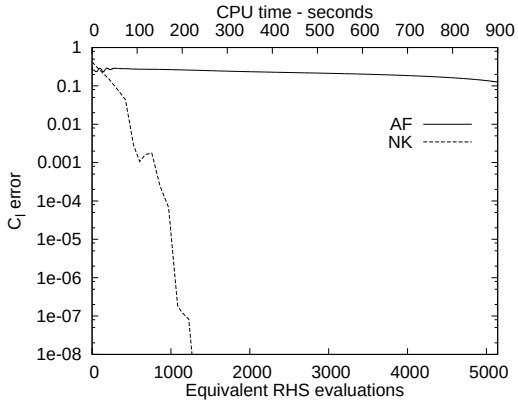
(b) 1 degree



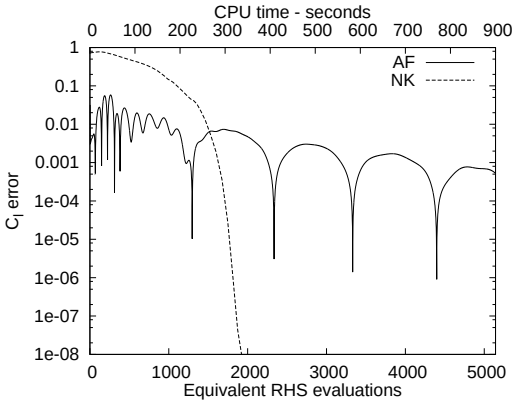
(c) 6 degrees



(d) 12 degrees



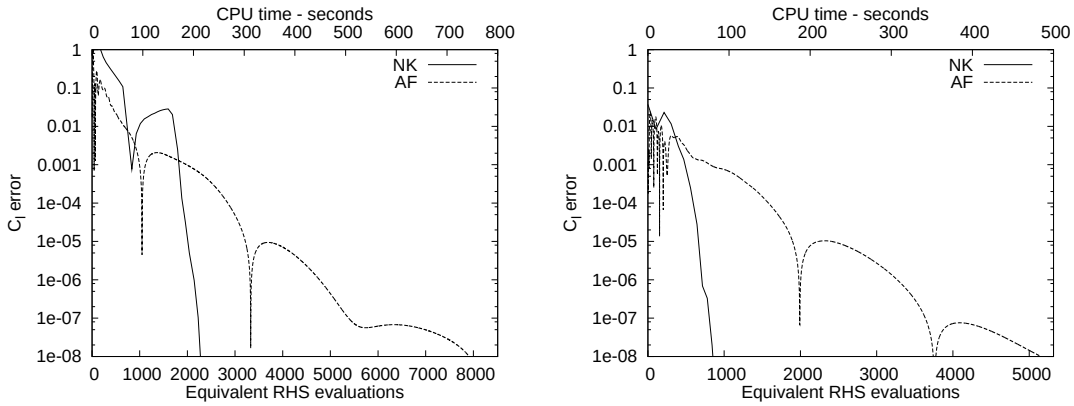
(e) 15 degrees



(f) 16 degrees

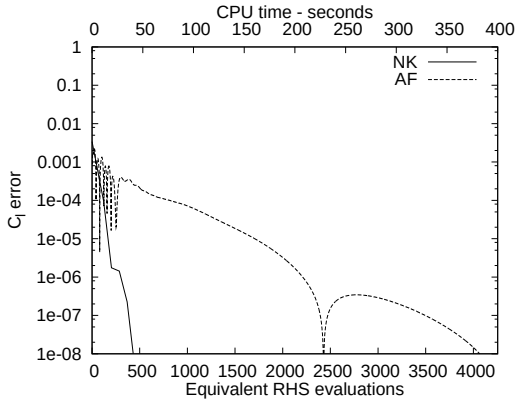
Figure 6.11: Angle sweep timing results

No time step is necessary, giving very short convergence times, less than 500 equivalent RHS evaluations after the second solve. Figure 6.12 shows a sampling of the convergence histories. The results show this approach is an excellent choice for use with optimization.

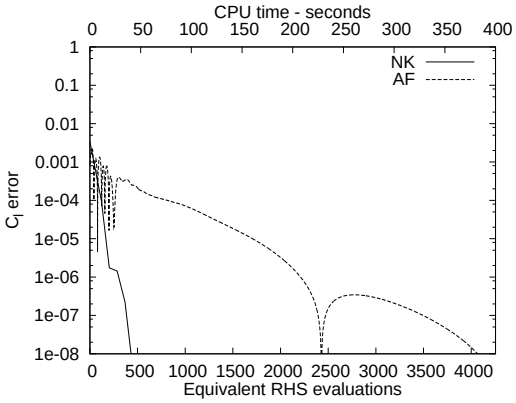


(a) First iteration

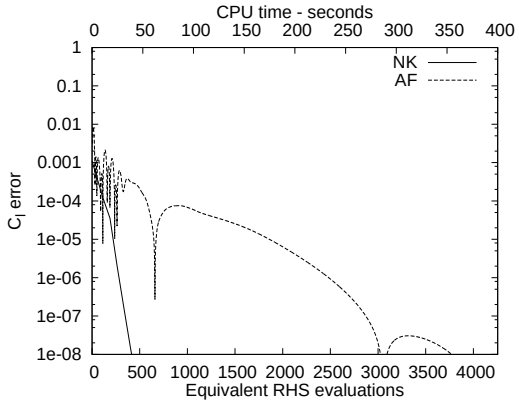
(b) Second iteration



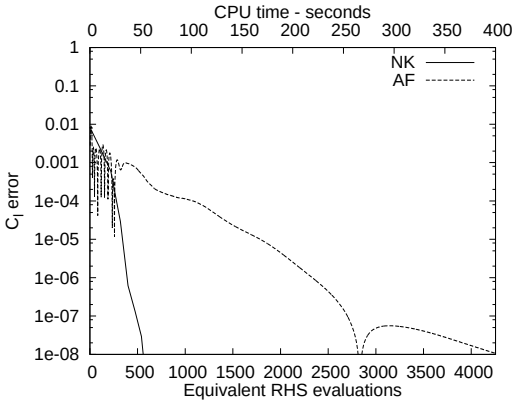
(c) Third iteration



(d) Fifth iteration



(e) Seventh iteration



(f) Ninth iteration

Figure 6.12: Flap optimization example timing results

Chapter 7

Conclusions, Contributions, and Recommendations

7.1 Conclusions

For the vast majority of cases, the method presented has been shown to be very efficient in terms of CPU time. It also tends to converge completely, giving the user more faith in the results. It is in general quite robust, with the possible exception of particularly nonlinear trip terms. In this case, the time step can be adjusted to converge successfully. The method is particularly well suited to optimization, where an accurate Jacobian and sufficient memory is already required. The 'warm starting' that occurs between optimization iterations also means that the slower initial stage can often be skipped, giving very fast flow evaluations. The method is also comparatively insensitive to grids and flow conditions.

Given these advantages, one may wonder why this method is not more popular. The simple answer is that it is more difficult to write the code for this method than most others. Careful differentiation needs to be performed, and complex matrix methods need to be implemented. While differentiation of these equations is tedious, it is not difficult (methods to aid in this error-prone process are presented in the appendix), and sophisticated and well-supported matrix packages are available, such as PETSc [54]. There is also a perception that optimizing the method for general cases can be very difficult or impossible, and that a lot of parameter tuning is needed for individual cases. This thesis has clearly

demonstrated that reliable convergence can be achieved for a wide range of cases and grids with a single set of parameters. An important part of this is having a thorough understanding of the system. Specifically:

- Preconditioning
  - Incomplete Lower-Upper preconditioning with a fill of four is a good choice to group the eigenvalues of the system.
  - The preconditioner should be based on an approximate matrix (following Pueyo [27]). However, the use of matrix dissipation requires the use of a higher value of σ to stabilize the decomposition with the lower diagonal dominance.
  - Reordering is important to producing an effective preconditioner. The reverse Cuthill-McKee method works well, although the root node of the ordering must be chosen carefully. Downwind nodes have been shown to be a good choice, though the reason for this is unclear.
- The matrix-free method
  - Using the matrix-free method to provide the matrix-vector products can be as, or more, efficient in CPU time than an explicit calculation. It is much more efficient in terms of memory use, reducing the requirements by at least half.
  - Proper scaling of the equations is very important to avoid large errors when using the matrix-free method. The large differences in residual that can arise without scaling result in either large round-off or truncation error.
  - The best choice of the perturbation parameter is higher than that recommended by Nielsen et al. [53]. We recommend 10^{-12} for inviscid cases, and 10^{-10} for turbulent.
- Stabilization of the nonlinear solution
  - Using a time step inversely proportional to the scaled system residual works well, though perhaps is not ideal. Some modifications such as introducing a minimum time step increase stability and efficiency.
  - A modified local time step for the turbulence model equations allows the use of the matrix-free method.
  - Grid sequencing provides a good initial guess, allowing larger initial time steps on the fine grid, and helps increase stability.

7.2 Contributions

Optimizing the Newton-Krylov method for the equations presented has resulted in an efficient and robust method. The major contributions from this work are shedding light on previously poorly understood processes and developing tools to identify and correct problems with the algorithm. The methodology

presented should be immediately useful to anyone who wishes to create a Newton-Krylov solver. Hopefully, the presentation is general enough to give ideas for systems of partial differential equations other than that presented here.

Particular contributions are:

- The preconditioner is prone to breakdown. The condition estimate (as presented by Chow and Saad [67]) is a critical measurement of the conditioning of the system. The use of matrix dissipation results in a less stable decomposition than the scalar dissipation used by Pueyo [27]. Experiments have shown that the best approach to stabilizing the preconditioner is to increase σ until the condition estimate is satisfactory.

- As Pueyo found, reordering before applying the decomposition is very important. We have examined the undefined parts of the reordering algorithm, and offered recommendations. This is particularly important when using multi-block grids, since the methods used by Pueyo are only applicable to C-grids.

- The standard approach to stabilizing the Spalart-Allmaras equations relies on modifying the Jacobian matrix. This precludes the use of the matrix-free calculation of the matrix-vector product. We have developed a new local time step that provides the necessary stability, but does not require any manipulations of the Jacobian.

- If large scaling differences exist in the equations, the linear solver is not forced to reduce the smaller values to meet the convergence tolerance. If the scaling differences are large enough, the smaller values can increase, leading to problems with the nonlinear calculations, such as negative densities and pressures. Therefore, we have developed a technique to scale the equations such that the Navier-Stokes and the turbulence model equations are both reduced by the linear solver.

- The GMRES Krylov solver used in this work does not explicitly calculate the L_2-norm of the residual. Instead it is found indirectly, in a process which we discovered is prone to error. This error is the sum of round-off and truncation error from the matrix-free method. If it is large enough, the solver can appear to converge, when in fact the actual linear residual has increased. In order to minimize this error, it is important to properly select the step size for the matrix-free method. We presented an empirical technique to determine the best step-size.

- A method of determining the time step has been developed that addresses the particular difficulties of the turbulence model with the very nonlinear trip terms.

7.3 Recommendations

There is a tremendous amount of potential in these methods, especially when used as the solver for numerical optimization. Future work should continue to explore the unknowns in the method that could potentially be exploited to improve the algorithm. In particular, the reason for one root node being better than another is a mystery. The inner workings of the ILU process should be examined to accomplish this. This also leads to potentially stabilizing the process by the judicious and more precise

manipulation of the matrix the preconditioner is based on. More advanced preconditioners are also very promising. Matrix-free preconditioners offer the advantage of much lower memory use. Multi-grid preconditioning may increase efficiency.

The method used to determine the time step seems less than optimal. It is based on the assumption that the residual is a good foundation for selecting the time step at each iteration.

Extending the equations solved would further demonstrate the usefulness of the Newton-Krylov method. For example, extension to three dimensions is particularly exciting. The addition of new turbulence models should be eased by the techniques presented.

Extending the solver to use higher-order spatial discretizations may prove to be useful and interesting. DeRango and Zingg [15] presented an algorithm that significantly reduced the number of nodes required to solve the turbulent Navier-Stokes equations on structured grids. Nejat and Ollivier-Gooch [78] presented a higher-order method for the Euler equations on unstructured grids.

Adding local preconditioning may also prove interesting. Unrau and Zingg have demonstrated this technique with ARC2D [79].

Appendix A

A DEBUGGING GUIDE TO NEWTON-KRYLOV METHODS

Debugging a Newton-Krylov method is particularly challenging because it is often difficult to correctly identify the cause of a problem. Here we present a simple guide and a suite of tools that should help in properly diagnosing bugs, of both the programming and conceptual type. There is one, rather large, assumption made: that the right-hand-side, or residual, evaluation is correct. However, with a properly calculated residual, we should be able to pinpoint most problems, by breaking the problem down into managable parts.

The problems can be broken down into two obvious areas: the nonlinear and the linear solution. Into the former fall problems like evaluation of an appropriate time step, and addressing highly nonlinear regions. The latter includes preconditioner failure, matrix-free breakdown, and scaling problems. In this appendix, we discuss how to determine whether the problem lies with the linear or nonlinear component of the solver, then discuss characteristics of all the problems we have experienced and what to do about them. Most of this has been discussed earlier in this thesis, but hopefully this format will be an easier reference. Finally, we present a logical progression that will minimize the number of problems experienced concurrently.

A.1 Differentiating linear and nonlinear problems

To be able to properly categorize a problem is crucial to debugging efficiently. To do this, we look at whether the linear solve has failed, since these problems are far easier to identify. As an example, we consider a situation that occurred early in research with Scirocco. While solving a turbulent, transonic case, negative pressures and densities were causing divergence. The acid test for linear problems is to examine the linear residual reduction at problem nodes. In this example, watching the linear residual reduction in the region where the negative densities occurred showed that instead of being reduced, the linear residual of the mean-flow equations was increasing. Thus there was a problem with the linear solution. If instead, the linear residual was adequately reduced, the problem would lie with the nonlinear solver. In this case, the residual vector was being dominated by the turbulence model equations, so that rescaling solved the problem.

A.2 Linear problems

Effectively, the only problem that can occur with the linear solver is failing to converge. However, there are manifestations of this problem beyond the obvious failure. First, as described in Section 5.7, the linear solver can appear to converge, although it hasn't. Second, even if it does converge globally, locally the linear residual can increase. There are fairly easy indicators that help us with diagnosis. After the linear solver finishes, the actual reduction of residual should be calculated by:

$$Reduction = \frac{||\mathcal{R}_{lin}||}{||b||} \tag{A.1}$$

where

$$\mathcal{R}_{lin} = \mathcal{A}x - b \tag{A.2}$$

is the linear residual. If the matrix-vector products are accurate, the norm of $\mathcal{R}$ is equal to that reported by the GMRES method. If, on the other hand $||\mathcal{R}_{lin}||/||b||$ is substantially greater than the set linear tolerance (usually 0.1), then matrix-free breakdown has occurred. In egregious cases, this ratio can exceed ten, usually followed by divergence.

It can also be insightful to check on the reduction of the residual on an equation by equation and node by node basis. To check the residual reduction for each equation, we check the residual ratio for each of the five equations individually. This is often a good indicator that there are overall scaling problems. If, for example, only the turbulence model equations are being reduced, the magnitude of the mean-flow equations will likely need to be increased[1].

Examining the residual reduction on a node-by-node basis is somewhat trickier, because the entire node field needs to be examined. There is also a subtle problem with initial magnitude of the residual: It is possible in a plotting package to examine the entry by entry division of two vectors, in this case $|\mathcal{R}_{lin}|/|b|$. However, many of the nodes will show huge increases. Further examination will show that these nodes started with a very small residual, well below what we consider converged. A node with

[1] Relative to the turbulence model equations. We took the approach of scaling the turbulence variable and residual by 10^{-3}, based on the largest values seen in the converged results. See Section 5.2

a linear residual that moves from 10^{-17} to 10^{-15} is likely not a concern, whereas one that jumps from 10^{-7} to 10^{-6} may be. To remedy this, we tend to plot

$$\frac{|\mathcal{R}_{lin}|}{|b| + threshold} \tag{A.3}$$

Of course, coming up with the threshold is nontrivial, as it depends on how far the equations are converged as well as scaling of the equations. 10^{-12} is a good starting point, however.

Failure to converge can also be related to problems with the preconditioner. The condition estimate of the decomposition is a good indicator of this problem.

The causes of problems that we have experienced, along with the indications and solutions, are:

Preconditioner Breakdown: This is indicated by a high (above 10^6) condition estimate. The matrix the preconditioner is based on is too poorly conditioned. This can be difficult to remedy, since it requires careful modification of the matrix. We base the preconditioner on a matrix with modified dissipation - an important source of diagonal entries. Decreasing the time step of the system also helps, though this may dramatically affect the number of nonlinear iterations. Increasing σ can be very helpful, as discussed in Section 5.6.1, though there is an optimum value, after which the error introduced overwhelms the benefit of stabilization.

If we do not want to modify the matrix, the level of fill can be increased, which will often stabilize the factorization. Reordering the preconditioner can also be very beneficial.

Insufficient Fill: In some cases, even if the condition estimate is reasonable, the preconditioner will not sufficiently cluster the eigenvalues. The only indication of this is an unacceptable number of linear iterations. Increasing the fill will decrease the number of iterations.

Preconditioner matrix error: Differences between the actual matrix used and the exact Jacobian matrix can be introduced by error, omission, or design. The first is discussed in Section A.4, where we discuss techniques to check the accuracy of the matrix. Terms can be omitted from the matrix if they are too difficult or time consuming to calculate. For example, the circulation correction entries are not added to the matrix. Finally, some steps are often taken to make the matrix more diagonally dominant or require less memory, such as only using second difference dissipation. These omissions and changes must be carefully considered. If the matrix the preconditioner is based on is too different from the true Jacobian matrix, preconditioning can become ineffective. This is indicated by a high number of linear iterations. Increasing the fill may help a little, but even in the limiting case of a full decomposition, there will still be a significant number of iterations required.

Poor scaling: Poor scaling of the equations can cause two problems. The first happens when using the matrix-free method, and will be discussed below, with nonlinearity considerations. The second problem results from not fully solving the linear system. To meet the residual reduction constraint, the linear solver may not need to reduce all the residuals, and can allow increases in some if the initial residual is small enough. This effect can be observed by examining the ratio of final to initial residuals at each node for each iteration.

Nonlinearity: Nonlinearity in the equations is only a problem for the linear solver when the matrix-vector product is provided by the matrix-free method. The error in the difference method has two components: round-off error and truncation error. When using the first-order matrix-free formula, equation 4.15, the greater the second- and higher-order derivatives, the greater the truncation error.

Ideally, we could reduce the step size to address this, but round-off error increases with decreasing perturbation. If a significant difference is seen between the approximated final residual returned by the GMRES method and the actual final residual, we can assume a large error in the matrix-free evaluation. To correct this, we may need to adjust the perturbation quantity. Section 5.7 gives an empirical approach to determining a good step size. In more difficult cases, a higher order method, such as a second-order centred approximation, may be necessary. This reduces the error resulting from nonlinearity, allowing larger step sizes and less round-off error.

A.3 Nonlinear problems

The Newton method will fail to converge if the initial guess is too far from the root. This problem is the result of nonlinearities in the function. To combat this problem, we use the implicit Euler method, which adds a time step component. The choice of the time step is absolutely critical to achieving a robust and efficient algorithm. It is important, then, to understand the causes of the nonlinearity so that they can be addressed. For example, the Spalart-Allmaras turbulence model is unstable with negative turbulence quantities. A modified time step is used to prevent negative values.

A.4 Forming an explicit matrix with finite-differences

One other problem can occur if the matrix-vector product is explicitly calculated. Any error in this matrix can slow or even prevent convergence. Even if the matrix-free method is used, errors can affect the preconditioner, so that it is important to check for accurate differentiation. Considering the complexity of coding the Jacobian evaluation, it is easy to introduce errors. For this reason, we suggest carefully checking the matrix using a finite-difference approach. The matrix can be entirely evaluated on a column-wise basis by perturbing each element of the state vector, and using a difference method to approximate the entries of the column. As usual, this method has a few caveats:

- Care must be taken to choose a good step size so that the evaluation is accurate. Using a higher-order method eases the range of effective step sizes, making the choice easier.
- The approximations used in the calculation of the Jacobian can confuse comparisons with the finite-difference matrix. In this case, the terms with the approximations should be removed for comparing the exact portions. For example, we set the dissipation terms to zero to check the flux components.
- The evaluation of the matrix can be very slow.

Another advantage of having the finite-difference matrix allows us to use this in place of the directly calculated matrix for testing. In order to do this, the speed of evaluation needs to be increased. We use an approach similar to a colouring scheme. The matrix graph is divided into sets, such that each set is comprised of nodes which are strictly independent of each other. Then, only one right hand side evaluation is needed for each set, as the independent nodes can be perturbed at the same time. This decreases the time requirements by orders of magnitude, making it feasible for experimental use.

A.5 Debugging process

As a debugging process, the following pattern is suggested. Some of this is obvious, but it should allow the development in such a way that the number of concurrent problems is minimized.

- Begin with the most simple configuration possible, then increase complexity slowly. For example, we begin with free-stream flow, then laminar and turbulent flow over a flat plate. These small problems allow us to use very high levels of fill and tight linear tolerances.
- Establish bounds for the time step. Use as high a level of fill as possible, with a tight linear tolerance.
- Use a small reference time step throughout convergence. If this does not converge, check for problems with the linear solve (see above). If the linear solve is working, likely the time step is too high, or there is an error in the time step calculations.
- Establish the largest initial reference time step that is stable.
- Use the initial reference time step for the first portion of the convergence, then switch to a full Newton solve. Identify where the full Newton solve can be used.
- Establish how the time step can be effectively ramped between the low initial time step and the Newton stage.
- Repeat while loosening the linear tolerance.
- Repeat while decreasing the level of fill. This will reveal problems with the preconditioner.

Of course, at each stage, the linear solver must be watched for problems. Also, we have not addressed the use of a spatially-varying time step. This choice should be based on the physics of the problem. For example, the spatially varying time step used (from Pulliam [55]) is roughly based on a constant-CFL number.

REFERENCES

[1] MacCormack, R. W., "The Effect of Viscosity on Hypervelocity Impact Cratering," 1969, AIAA Paper 69-0354.

[2] Brandt, A., "A Guide to Multigrid Development," *Multigrid Methods*, Springer-Verlag, 1982, pp. 220–312.

[3] South, J. C. and Brandt, A., "Application of a Multi-Level Grid Method to Transonic Flow Calculations," *Proc. of workshop on transonic flow problems in turbomachinery*, edited by T. C. Adamson and M. F. Platzer, Hemisphere, 1977, pp. 180–206.

[4] Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multigrid Method," *Proc. AIAA 4th computational fluid dynamics conference*, 1979, pp. 122–146, Williamsburg.

[5] Ni, R. H., "A Multiple Grid Scheme for Solving the Euler Equations," *Proc. AIAA 5th computational fluid dynamics conference*, 1981, pp. 257–264, Palo Alto.

[6] Jameson, A., "Solution of Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327–356.

[7] Martinelli, L., Jameson, A., and Grasso, F., "A Multigrid Method for the Navier-Stokes Equations," 1986, AIAA paper 86-0208.

[8] Mavriplis, D. J., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," *AIAA Journal*, Vol. 26, July 1988.

[9] Jameson, A. and Yoon, S., "Multigrid Solution of the Euler Equations Using Implicit Schemes," *AIAA Journal*, Vol. 24, No. 11, November 1986, pp. 1737–1743.

[10] Beam, R. and Warming, R. F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *J. Comp. Phys.*, Vol. 22, 1977, pp. 87–110.

[11] Steger, J. L., "Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils," 1977, AIAA paper 77-665.

[12] Pulliam, T. H. and Chaussee, D. S., "A Diagonal Form of an Implicit Approximate Factorization Algorithm," *J. Comp. Phys.*, Vol. 39, 1981, pp. 347–363.

[13] Nelson, T. E., *Numerical Solution of the Navier-Stokes Equations for High-Lift Airfoil Configurations*, Ph.D. thesis, University of Toronto, 1994.

[14] Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," 1992, AIAA paper 92-0439.

[15] De Rango, S. and Zingg, D. W., "A High-Order Spatial Discretization for Turbulent Aerodynamic Computations," *AIAA Journal*, Vol. 39, No. 7, July 2001, pp. 1296–1304.

[16] Chisholm, T., *Multigrid Acceleration of an Approximately-Factored Algorithm for Steady Aerodynamic Flows*, Master's thesis, University of Toronto, 1997.

[17] Manzano, L., *Implementation of Multigrid for Aerodynamic Computations on Multi-Block Grids*, Master's thesis, University of Toronto, 1999.

[18] Wigton, L. B., "Application of MACSYMA and Sparse Matrix Technology to Multielement Airfoil Calculations," 1987, AIAA paper 87-1142.

[19] Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems," 1988, AIAA paper 88-0413.

[20] Hafez, M., Palaniswamy, S., and Mariani, P., "Calculations of Transonic Flows with Shocks using Newton's Method and a Direct Solver, Part II," 1988, AIAA paper 88-0226.

[21] Venkatakrishnan, V. and Barth, T. J., "Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes," 1989, AIAA paper 89-0364.

[22] Orwis, P. D., *A Newton's Method Solver for the Two-Dimensional and Axisymmetric Navier-Stokes Equations*, Ph.D. thesis, North Carolina State University, 1990.

[23] Bailey, H. E. and Beam, R. M., "Newton's Method Applied to Finite-Difference Approximations for the Steady-State Compressible Navier-Stokes Equations," *J. Comp. Phys.*, Vol. 93, 1991, pp. 108–127.

[24] Hestenes, M. R. and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems of Equations," *J. Res. Nat. Bur. Stand.*, Vol. 49, 1952, pp. 409–436.

[25] Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, July 1986.

[26] Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and van der Vorst, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1993.

[27] Pueyo, A., *An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Toronto, 1998.

[28] Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes," 1985, AIAA paper 85-1494.

[29] Knoll, D. A. and Keyes, D. E., "Jacobian-Free Newton-Krylov methods: a Survey of Approaches and Applications," *J. Comp. Phys.*, Vol. 193, 2004, pp. 357–397.

[30] Venkatakrishnan, V. and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes," 1991, AIAA paper 91-1537-CP.

[31] Ajmani, K., Ng, W., and Liou, M., "Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations," *J. Comp. Phys.*, Vol. 110, 1994, pp. 68–81.

[32] Ajmani, K. and Liou, M., "Implicit Conjugate-Gradient Solvers," 1995, AIAA paper 95-1695.

[33] McHugh, P. R. and Knoll, D. A., "Comparison of Standard and Matrix-Free Implementations of Several Newton-Krylov Solvers," *AIAA Journal*, Vol. 12, December 1994, pp. 2394–2400.

[34] Forsyth, P. A. and Jiang, H., "Iterative Methods for Full Newton Solution of the Euler Equations," September 1995, pp. 318–323, Sixth International Symposium on Computational Fluid Dynamics.

[35] Forsyth, P. A. and Jiang, H., "Nonlinear Iteration Method for High Speed Laminar Compressible Navier-Stokes Equations," *Computers & Fluids*, Vol. 26, No. 3, 1997, pp. 249–268.

[36] Rogers, S. E., "Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations," *AIAA Journal*, Vol. 33, No. 11, Oct 1995, pp. 2066–2072.

[37] Barth, T. J. and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," 1995, AIAA paper 95-0221.

[38] Saad, Y., "ILUT: A Dual-Threshold Incomplete LU Factorization," Tech. Rep. UMSI 92/38, University of Minnesota Supercomputing Institute, 1992.

[39] Liu, W. H. and Sherman, A., "Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices," *SIAM J. Numer. Anal.*, Vol. 13, No. 2, April 1976.

[40] Geuzaine, P., *An Implicit Upwind Finite Volume Method for Compressible Turbulent Flows on Unstructured Meshes*, Ph.D. thesis, Université de Liegè, 1994.

[41] Knoll, D. A. and Mousseau, V. A., "On Newton-Krylov Multigrid Methods for the Incompressible Navier-Stokes Equations," *J. Comp. Phys.*, Vol. 163, 2000, pp. 262–267.

[42] Pernice, M. and Tocci, M., "A Multigrid-Preconditioned Newton-Krylov Method for the Incompressible Navier-Stokes Equations," *SIAM J. Sci. Comput.*, Vol. 23, 2001, pp. 398–418.

[43] Mousseau, V. A., Knoll, D. A., and Rider, W. J., "Physics-Based Preconditioning and the Newton-Krylov Method for Non-Equilibrium Radiation Diffusion," *J. Comp. Phys.*, Vol. 160, 2000, pp. 743–765.

[44] Luo, H., Baum, J. D., and Lohner, R., "High-Reynolds Number Viscous Flow Computations Using an Unstructured-Grid Method," *J. of Aircraft*, Vol. 42, No. 2, 2005, pp. 483–492.

[45] Cai, X. C., Keyes, D. E., and Venkatakrishnan, V., "Newton-Krylov-Schwarz: An Implicit Solver for CFD," *in Proceedings of the Eighth International Converence on Domain Decomposition Methods*, Wiley, 1997, pp. 387–400.

[46] Tidriri, M., "Schwarz-Based Algorithms for Compressible Flows," Tech. Rep. ICASE 96-4 CR-198273, NASA, 1996.

[47] McHugh, P. R., Knoll, D. A., and Keyes, D. E., "Application of a Schwartz-Preconditioned Newton-Krylov Algorithm to a Low-Speed Reacting Flow Problem," *AIAA Journal*, Vol. 36, 1998, pp. 290–292.

[48] Isono, S. and Zingg, D. W., "A Runge-Kutta-Newton-Krylov Algorithm for Fourth-Order Implicit Time Marching Applied to Unsteady Flows," 2003, AIAA paper 2003-3956.

[49] Nemec, M., Zingg, D. W., and Pulliam, T. H., "Multipoint and Multi-Objective Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 42, No. 6, July 1988, pp. 1057–1065.

[50] Gatsis, J. and Zingg, D. W., "A Fully-Coupled Newton-Krylov Algorithm for Aerodynamic Optimization," 2003, AIAA paper 2003-3956.

[51] Nichols, J. and Zingg, D. W., "A Three-Dimensional Multi-Block Newton-Krylov Flow Solver for the Euler Equations," 2005, AIAA paper 2005-5230.

[52] Wong, P. and Zingg, D. W., "Aerodynamic Computations on Unstructured Grids Using a Newton-Krylov Approach," 2005, AIAA paper 2005-5231.

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

[54] Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Web Page," 2001, http://www.mcs.anl.gov/petsc.

[55] Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations." Lecture Notes for The Von Karman Institute, for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Von Karman Institute, Brussels, Belgium, January 1986

[56] Salas, M., Jameson, A., and Melnik, R., "A Comparative Study of the Nonuniqueness Problem of the Potential Equation," June 1983, AIAA paper 83-1888.

[57] Zingg, D. W., "Grid Studies for Thin-Layer Navier-Stokes Computations of Airfoil Flowfields," January 1992, AIAA paper 92-0184.

[58] Godin, P., Zingg, D. W., and Nelson, T. E., "High-lift Aerodynamic Computations with One- and Two-Equation Turbulence Models," *AIAA Journal*, Vol. 35, No. 2, July 1997, pp. 237–243.

[59] Ashford, G. A., *An Unstructured Grid Generation and Adaptive Solution Technique for High Reynolds Number Compressible Flows*, Ph.D. thesis, University of Michigan, 1996.

[60] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," June 1981, AIAA paper 81-1259.

[61] Swanson, R. C. and Turkel, E., "On Central-Difference and Upwind Schemes," *J. Comp. Phys.*, Vol. 101, 1992, pp. 292–306.

[62] Van der Vorst, H. A. and Vuik, C., "GMRESR: a Family of Nested GMRES Methods," *Numerical Linear Algebra with Applications*, Vol. 1, 1993, pp. 1–7.

[63] Saad, Y., "A Flexible Inner-Outer Preconditioned GMRES Algorithm," *J. Sci. Stat. Comp.*, Vol. 14, 1993, pp. 461–469.

[64] Pueyo, A. and Zingg, D. W., "Improvements to a Newton-Krylov Solver for Aerodynamic Flows," January 1998, AIAA paper 98-0619.

[65] Orkwis, P. D., "A Comparison of Newton's and Quasi-Newton's Method Solvers for the Navier-Stokes Equations," *AIAA J.*, Vol. 31, No. 5, May 1993.

[66] Elman, H. C., "A Stability Analysis of Incomplete LU Factorizations," *Mathematics of Computation*, Vol. 47, No. 175, July 1986.

[67] Chow, E. and Saad, Y., "Experimental Study of ILU Preconditioners for Indefinite Matrices," Tech. Rep. UMSI 97/95, University of Minnestoa Supercomputing Institute, June 1997.

[68] George, A., "Nested Dissection of a Regular Finite Element Mesh," *SIAM J. Numer. Anal.*, Vol. 10, No. 2, April 1973, pp. 345–363.

[69] George, A., "An Automatic One-Way Dissection Algorithm for Irregular Finite Element Problems," *SIAM J. Numer. Anal.*, Vol. 17, No. 6, Dec. 1980, pp. 740–751.

[70] George, A. and Liu, J. W. H., "The Evolution of the Minimum Degree Ordering Algorithm," *SIAM rev.*, Vol. 31, No. 1, 1989, pp. 1–19.

[71] Dutto, L. C., "The Effect of Ordering on Preconditioned GMRES Algorithm, for Solving the Compressible Navier-Stokes Equations," *International Journal for Numerical Methods in Engineering*, Vol. 38, 1993, pp. 457–497.

[72] Benzi, M., Szyld, D. B., and Duin, A. V., "Orderings for Incomplete Factorization Preconditioning of Nonsymmetric Problems," *SIAM J. Sci. Comput.*, Vol. 20, 1999, pp. 1652–1670.

[73] "AMBER2D User's Manual," Tech. rep., University of Toronto, June 1997.

[74] van den Berg, B., "Boundary Layer Measurements on a Two-Dimensional Wing with Flap," Tech. Rep. TR 79009 U, NLR, Jan. 1979.

[75] Blanco, M. and Zingg, D. W., "A Fast Solver for the Euler Equations on Unstructured Grids Using a Newton-GMRES Method," 1997, AIAA paper 97-0331.

[76] "HYGRID User's Manual," Tech. rep., University of Toronto, June 1997.

[77] Moir, J. R. M., "Measurements on a Two-Dimensional Aerofoil with High-Lift Devices," *A Selection of Experimental Test Cases for the Validation of CFD Codes*, No. 303, AGARD, 1994.

[78] Nejat, A. and Ollivier-Gooch, C., "A High-Order Accurate Unstructured GMRES Solver for the Compressible Euler Equations," *Proc. of the Third International Conference on Computational Fluid Dynamics*, 2004.

[79] Unrau, D. and Zingg, D. W., "Viscous Airfoil Computations Using Local Preconditioning," 1997, AIAA paper 97-2027.