A Newton-Krylov Solver for the Navier-Stokes Equations on Unstructured Grids

by

Peter J. Blaser

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

A Newton-Krylov Solver for the Navier-Stokes Equations on Unstructured Grids

Peter J. Blaser

Master of Applied Science

Graduate Department of Aerospace Science and Engineering

University of Toronto

2001

A two-dimensional Newton-Krylov solver for compressible viscous flows has been developed. The Navier-Stokes equations are discretised in space using a finite-volume formulation on arbitrary polygonal meshes. Nonlinear scalar artificial dissipation is added for numerical stability. Newton's method is used to solve the discrete nonlinear algebraic equations, while an ILU-preconditioned, matrix-free GMRES method solves the resulting linear systems. RCM reordering is used to reduce bandwidth, and local implicit-Euler time stepping promotes robustness during start-up.

The solver has been verified for a variety of laminar test cases. Optimal parameters have been obtained considering both speed and memory requirements. Extension to turbulent flows is discussed.

Acknowledgements

The past two years I spent at UTIAS have truly flown by. Many individuals have contributed positively to my experiences as well as to this work. I thank all with whom I have had the pleasure of working.

I wish to acknowledge and thank my supervisor, Professor D. W. Zingg for his inspiration, guidance, patience and understanding. Dr. Zingg manages to maintain a large, talented research group yet is always available for personal assistance. It was a pleasure to work with him.

I am further grateful to my colleagues on the Newton-Krylov project, namely Jason Lassaline, Luis Manzano and Edward Wehner. Jason, it looks like someone else gets a Master's based on your work. Luis, check the code; go to the code; did you check the code? If so, blame cvs. Edward, thanks for being the guinea pig with the inviscid solver. I wish all three my best wishes for whatever the future holds.

I must acknowledge the entire CFD group at UTIAS as being a catalyst for quality research. There always seemed to be a resident expert on any topic for which I had endless questions. I particularly wish to thank Todd Chisholm for his insights into Newton-Krylov techniques, PETSc and the Spalart-Allmaras turbulence model. Additionally I wish to thank Marian Nemec for Newton-Krylov insight, meshes, solutions and comparative results.

The two years which flew by for myself must have been an eternity to my wife, Amanda. I wish to express my gratitude to her for her patience, encouragement and persistent motivation.

Finally, the financial support of the Ontario Graduate Scholarship (OGS), the Ontario Graduate Scholarship in Science and Technology (OGSST) and the University of Toronto is gratefully acknowledged.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Alphanumeric

$\mathbf{A}$	generic linear system matrix
AD	artificial dissipation
C_d, C_l, C_p	drag, lift and pressure coefficients
$\overline{\mathbf{D}}$	approximate Spalart-Allmaras destruction linearization
$\mathbf{D}(\tilde{\nu})$	Spalart-Allmaras destruction matrix
DSx, **DSy**	vectors of x- and y- components of spatial derivatives
$\overline{\mathbf{F}}$	inviscid flux tensor
$\overline{\mathbf{G}}$	viscous flux tensor
$\mathbf{I}$	identity matrix
$\mathbf{K}_m$	Krylov subspace for m^{th} iteration
$\mathbf{L}$	lower-triangular matrix
$\mathbf{L}_i$, $\mathbf{L}_k$	Laplacian operators
N	linear system size
N_{bdy}	summation limit for boundary faces of cell i
N_i	summation limit for neighbours of cell i
M_∞	free-stream Mach number
$\mathbf{P}$	preconditioning matrix
$\overline{\mathbf{P}}$	approximate Spalart-Allmaras production linearization
$\mathbf{P}(\tilde{\nu})$	Spalart-Allmaras production matrix
Pr, Pr_t	Prandtl number, turbulent Prandtl number
$\mathbf{Q}$	conservative variables vector
$\Delta\mathbf{Q}$	conservative variables update vector
$\mathbf{R}$	nonlinear residual function
R_1, R_2, R_3, R_4	Riemann invariants
Re	Reynolds number
Re_a	Reynolds number based on sound-speed
S	magnitude of vorticity
$\mathbf{S}$	source term vector
$\tilde{S}$	Spalart-Allmaras model function
T	non-dimensional temperature
$\tilde{T}$	dimensional temperature
$\tilde{T}_\infty$	dimensional, free-stream temperature
$\mathbf{U}$	upper-triangular matrix

a	non-dimensional speed of sound
$\tilde{a}$	dimensional speed of sound
$\tilde{a}_\infty$	dimensional, free-stream speed of sound
$\mathbf{b}$	generic linear system right-hand side
bdy	summation variable for boundary faces of cell i
$\tilde{c}$	dimensional chord length
c_{b1}, c_{b2}	Spalart-Allmaras model constants
c_p	specific heat at constant pressure
c_{t1}, c_{t2}, c_{t3}, c_{t4}	Spalart-Allmaras model constants
c_{v1}	Spalart-Allmaras model constant
c_{w1}, c_{w2}, c_{w3}	Spalart-Allmaras model constants
d	distance to closest wall
d_t	distance to trip point
dA	elemental area for integration
dS	elemental surface contour for integration
$d\Omega$	perimeter of control volume
e	non-dimensional energy
$\tilde{e}$	dimensional energy
f, g	viscous energy fluxes
f_{t1}, f_{t2}	Spalart-Allmaras trip functions
f_{v1}, f_{v2}	Spalart-Allmaras model functions
f_w	Spalart-Allmaras distance function
g	Spalart-Allmaras function
g_t	Spalart-Allmaras function
h	time-step
h_c, h_v	convective and diffusive time-steps
$\mathbf{i}$,$\mathbf{j}$	unit normals in x- and y- directions
k	summation variable for neighbour cells of cell i
m	linear iteration counter
min	minimum function
$\mathbf{n}$	outward pointing normal
$\mathbf{n}_x$	normal in x- direction
$\mathbf{n}_y$	normal in y- direction
p	non-dimensional pressure, preconditioner level of fill

pos	positivity function
r	Spalart-Allmaras model function
$\mathbf{r}$	linear residual vector
r_{tol}	linear residual reduction ratio
t	non-dimensional time
u	non-dimensional x-component of velocity
$\tilde{u}$	dimensional x-component of velocity
v	non-dimensional y-component of velocity
$\tilde{v}$	dimensional y-component of velocity
$\lvert\tilde{\mathbf{v}}_\infty\rvert$	dimensional, free-stream speed
$\mathbf{v}$	velocity
$\lvert\Delta\mathbf{v}\rvert$	difference between speed at the field point and the trip point
$\mathbf{v}_1$	vector used for forming Krylov subspace
x, y	Cartesian coordinates
$\mathbf{x}$	generic linear system vector
Δx	grid spacing along the wall at the trip point

Subscripts

avg	average
bdy	at boundary face bdy
c	convective
dest	destruction (source term)
diff	diffusion (source term)
ext	extrapolated
i	at cell i
ik	from cell i to cell k
k	at cell k
l	left
m	number of GMRES iterations
n	normal component
n	iteration index
nn	a next-to-nearest neighbour
prod	production (source term)
r	right
t	tangential component
trip	trip (source term)
v	diffusive
∞	free-stream
0	initial

Superscripts

n	iteration index
$*$	a particular cell

Greek

Ω	volume (area in two dimensions)
χ	ratio of turbulent transport variable to non-dimensional kinematic viscosity
ρ	non-dimensional density
$\tilde{\rho}$	dimensional density
$\tilde{\rho}_\infty$	dimensional, free-stream density
γ	ratio of specific heats
μ	nondimensional, dynamic viscosity
μ_t	nondimensional, dynamic eddy viscosity
$\tilde{\mu}_\infty$	dimensional, free-stream, dynamic viscosity
$\tau_{xx}, \tau_{xy}, \tau_{yy}$	viscous stresses
κ	Spalart-Allmaras model constant
$\kappa^{(2)}$, $\kappa^{(4)}$	second- and fourth-difference artificial dissipation constants
κ_t	thermal conductivity
ν	non-dimensional kinematic viscosity
$\tilde{\nu}$	turbulent transport variable
σ	Spalart-Allmaras model constant
σ_t	forth-difference artificial dissipation control constant
w_t	vorticity at trip point
ϕ	arbitrary variable
λ_{ik}	artificial dissipation scaling factor
$\varepsilon^{(2)}$, $\varepsilon^{(4)}$	second- and fourth-difference artificial dissipation switches
ε	perturbation constant for use with matrix-free GMRES
ε_m	machine zero
λ_c, λ_v	approximations to maximum convective and diffusive eigenvalues

Abbreviations

BCG	Bi-Conjugate Gradient
Bi-CGSTAB	Bi-Conjugate Gradient Stabilized
CFD	Computational Fluid Dynamics
CFL	Courant-Fredrichs-Lewy (number)
CGS	Conjugate Gradient Squared
CPU	Central Processing Unit
DNS	Direct Numerical Simulation
GMRES	Generalized Minimal Residual (method)
GMRES(m)	Restarted Generalized Minimal Residual
ILU	Incomplete Lower-Upper (factorization)
ILU(p)	Incomplete Lower-Upper with fill, p
LES	Large Eddy Simulation
LHS	Left-Hand Side
LU	Lower-Upper (factorization)
ND	Nested Dissection
NN	Nearest-Neighbour (stencil)
NTNN	Next-to-Nearest Neighbour (stencil)
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PETSc	Portable, Extensible, Toolkit for Scientific Computation
QMD	Quotient Minimum Degree
RANS	Reynolds-Averaged Navier-Stokes (equations)
RCM	Reverse Cuthill-McKee (reordering)
RHS	Right-Hand Side
SOR	Successive Over-Relaxation
TFQMR	Transpose Free Quasi-Minimal Residual
UTIAS	University of Toronto, Institute for Aerospace Studies

Chapter 1

Introduction

Computational Fluid Dynamics (CFD) is the art and science of numerically modeling the physics of fluid flows. The starting point is the Navier-Stokes equations, which are integral or partial differential equations in space and time, for which no analytical solution is known. These equations are spatially discretised onto finite grids and integrated through finite time intervals until the desired numerical solution is obtained.

CFD complements the two older branches of fluid dynamics, namely theory and experiment. Theory typically examines problems with significant simplifying assumptions. Even so, useful results such as classical thin airfoil theory, Prandtl's lifting line theory, boundary layer theories and linearized supersonic flows have been deduced. Yet theory alone cannot be used for quantitative analysis of complex flows over complex configurations. This sort of analysis has traditionally been performed experimentally.

"CFD is a new third dimension in aerodynamics, complementing the previous dimensions of both pure experiment and pure theory. It allows us to obtain answers to fluid dynamic problems which heretofore were intractable by classical analytical methods. Consequently, CFD is revolutionizing the airplane design process, and in many ways is modifying the way we conduct modern aeronautical research and development" [1].

1.1 Motivation

There are many ways to subdivide the wide variety of CFD problems; the first such classification is grid structure. The spatial discretisation is typically categorized as either structured or unstructured. Structured grids maintain a simple connectivity between grid

nodes and may be thought of as two- or three-dimensional arrays. Dealing with structured grids is fairly straightforward and the connectivity of nodes is directly related to array indices within a computational domain. An excellent resource on understanding grid structure is the work of Thompson et al [2].

Unstructured grids, on the other hand, maintain no specific pattern of connectivity. These grids conform well to arbitrary or complex geometries, are simpler to generate than their structured counterparts and are easily adapted to more accurately resolve areas of steep gradients such as shocks. Unfortunately, there is usually an increased cost associated with finding and storing the neighbouring cell mappings.

Another classification of CFD codes involves the temporal advancement scheme. Time-marching methods are broadly categorized as either explicit or implicit, although modern codes generally utilize features of both. Explicit methods are more easily programmed at a reduced computational cost, but are constrained by strict stability limits. The first CFD codes utilized explicit time integration techniques such as MacCormack's predictor-corrector algorithm dating to 1969 [3]. Modern explicit methods typically utilize multigrid, local time-stepping and implicit residual smoothing to accelerate convergence.

Implicit methods, on the other hand, permit the use of much larger time steps, and thus converge in far fewer iterations than an explicit method. This rapid per-iteration convergence comes at a much larger per-iteration cost, since a linear system of equations must be solved at each time step. Examples of early implicit methods may be traced to Peaceman and Rachford [4], or Douglas and Gunn [5].

Newton-Krylov methods are implicit methods and may be used on either structured or unstructured grids. They are thus named since Newton's method is used to solve the nonlinear systems. The linear system arising at every Newton iteration is solved using a Krylov subspace method. These may be categorized as inexact Newton methods, since the linear system may be solved approximately. The early work of Dembo, Eisenstat and Steihaug [6] demonstrated the advantage of inexact Newton methods compared to an exact one. These resulted in enormous savings of computational effort, while retaining the desired quadratic convergence of Newton's method.

Several works followed which were based on explicit matrix representations of the Jacobian operator as recounted by Gropp et al [7]. Vanka [8] implemented Newton solvers in primitive variable Navier-Stokes problems. Venkatakrishnan [9], Orkwis [10], and Whitfield and Taylor [11] used Newton-like methods for the linear Newton correction equation.

The use of Krylov iterative methods for solving the linear systems produced by inexact Newton iterations in a matrix-free context can be traced to the ODE-oriented papers of Gear and Saad [12], Chan and Jackson [13] and Brown and Hindmarsh [14], and the PDE-oriented work of Brown and Saad [15]. According to Gropp et al [7], it may have been in this last work [15] that the term "Newton-Krylov" was first used.

The Generalized Minimal RESidual (GMRES) method developed by Saad and Schultz [16] was popularized following the work of Wigton, Yu and Young [17] and Johann, Hughes, and Shakib [18, 19]. In 1993 Venkatakrishnan and Mavriplis showed that preconditioned Newton-Krylov methods were competitive with explicit multigrid methods for large-scale CFD problems [20]. In 1995 Keyes performed a similar comparison for the matrix-free form of such methods [21]. Other aspects of Newton-Krylov solvers may be found in [22, 23, 24, 25, 26, 27, 28, 29, 30].

The application of Newton-Krylov methods to fluid flows has been studied within the University of Toronto Institute for Aerospace Studies (UTIAS) CFD group as well, both with structured and unstructured meshes. In 1995, Blanco solved the Euler equations on unstructured triangular grids [31]. His work featured a node-based, centered finite-volume discretisation combined with a nonlinear artificial dissipation scheme. A GMRES Krylov solver was employed within the nonlinear, Newton framework while an incomplete LU preconditioner based on a lower-order Jacobian accelerated convergence. Reverse Cuthill-McKee (RCM) reordering reduced bandwidth, and start-up instabilities were avoided using implicit-Euler time integration. Comparisons with an explicit multigrid solver have shown the Newton-Krylov solver to converge 2-3 times quicker.

The aforementioned study has also been the topic of subsequent publications [32, 33] which compare a quasi-Newton algorithm with standard and matrix-free variants of an inexact full-Newton solver. Again, the results favour the matrix-free variant of the Newton-GMRES technique.

Additionally Pueyo solved the Navier-Stokes equations on structured grids with an algebraic turbulence model [34, 35, 36, 37]. He used a second-order centered-difference operator with second- and fourth-difference dissipation. A preconditioned, restarted, matrix-free GMRES solver was used for the linear systems and RCM reordering was applied. The algorithm was started with an approximately-factored solver, combined with mesh sequencing. Comparisons with other efficient implicit solvers found matrix-free, inexact-Newton-GMRES to be the fastest and most robust.

The immediate precursor to this current work is a thesis by Wehner [38], which studied a Newton-Krylov solver for the Euler equations on general unstructured grids. The spatial discretisation is taken from that of Lassaline [39] and involves a finite-volume formulation with a nonlinear artificial dissipation scheme added for numerical stability. The nonlinear, discrete system is solved using an inexact Newton strategy while the resulting linear systems are solved with a matrix-free GMRES algorithm. Preconditioning is based on an incomplete LU decomposition of a lower-order Jacobian matrix, and RCM reordering is applied. Wehner found that an implicit Euler start-up strategy was required for transonic flows only [38].

1.2 Objective

This work is a viscous extension of the inviscid study by Wehner [38]. The objective of this project is to develop and optimize a two-dimensional, Newton-Krylov solver for the Navier-Stokes equations on general unstructured grids. This solver uses a cell-based finite-volume formulation on arbitrary polygonal meshes. As with the inviscid solver, a second- and fourth-difference scalar artificial dissipation is added for numerical stability. Newton's method is used for the discrete, nonlinear, algebraic equations, while an ILU-preconditioned, matrix-free GMRES method solves the resulting linear systems. RCM reordering is used to reduce bandwidth and local, implicit-Euler time stepping promotes robustness during start-up.

The combined inviscid solver [38] and viscous solver presented here is based on the discrete, nonlinear residual as formulated by Lassaline and used for his agglomeration multigrid solver [39]. The combined solver will be referred to as Hurricane throughout this work. The specific objectives of this thesis are:

- Implement a matrix-free, Newton-Krylov solver (in conjunction with Wehner).
- Linearize the viscous terms.
- Linearize the turbulence model.
- Implement the local, implicit-Euler time stepping.
- Optimize the solver parameters for various test cases.

As will be discussed in section 2.2, the Spalart-Allmaras turbulence model is coupled with the Reynolds-Averaged Navier-Stokes equations. As a result, this work actually implements two related, yet distinct, solvers, namely those for laminar and turbulent flows. As such, implementation and optimization will examine both subsets of viscous flows. The primary objective of this work is to fully implement, optimize and verify the laminar solver, and to develop the framework for coupling the turbulence model with the flow equations.

1.3 Thesis Outline

This document is subdivided into six chapters. Following the introduction, the second chapter presents the Navier-Stokes equations and the Spalart-Allmaras turbulence model. Chapter Three discusses the spatial discretisation of the equations onto an unstructured mesh and presents the nonlinear, algebraic residual system, $\mathbf{R}(\mathbf{Q}) = \mathbf{0}$. Solution of such a system is the topic of Chapter Four, and the matrix-free Newton-Krylov algorithm is put forth. Chapter Five optimises the various parameters and strategies used in the Newton-Krylov scheme and presents some final results. Based on these results, the sixth and final chapter draws some conclusions and outlines suggestions for further study regarding this topic.

Chapter 2

Equations

In this chapter the Navier-Stokes equations and the Spalart-Allmaras turbulence model are presented and discussed.

2.1 Navier-Stokes Equations

The physics of compressible fluid flows may be described by the conservation of mass, momentum and energy. These conservation laws, the Navier-Stokes equations, are presented in integral, conservative, non-dimensional form as

$$\frac{d}{dt}\int_{\Omega} \mathbf{Q}\, dA + \int_{\Omega} \nabla \cdot \overline{\mathbf{F}}\, dA = \int_{\Omega} \nabla \cdot \overline{\mathbf{G}}\, dA \tag{2.1}$$

and are expressed as a function of the vector of conserved variables:

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.2}$$

The dimensional variables, density ($\tilde{\rho}$), velocity components ($\tilde{u}$, $\tilde{v}$) and total energy ($\tilde{e}$), are non-dimensionalised using free-stream (∞) values:

$$\rho = \frac{\tilde{\rho}}{\tilde{\rho}_{\infty}}; \quad u = \frac{\tilde{u}}{\tilde{a}_{\infty}}; \quad v = \frac{\tilde{v}}{\tilde{a}_{\infty}}; \quad e = \frac{\tilde{e}}{\tilde{\rho}_{\infty}\tilde{a}_{\infty}^2} \tag{2.3}$$

where a is the speed of sound. For ideal fluids this may be written as

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{2.4}$$

where the ratio of specific heats, γ, is taken as 1.4 for air. The Reynolds number based on the free-stream sound speed

$$Re_a = \frac{\tilde{\rho}_\infty \tilde{c} \tilde{a}_\infty}{\tilde{\mu}_\infty} \tag{2.5}$$

appears as a result of the choice of sound speed as the reference speed. This value differs from the traditional Reynolds number by

$$Re_a = \frac{Re}{M_\infty} \tag{2.6}$$

where

$$M_\infty = \frac{|\tilde{\mathbf{v}}_\infty|}{\tilde{a}_\infty} \tag{2.7}$$

is the free-stream Mach number.

Both the inviscid and viscous fluxes, respectively

$$\overline{\mathbf{F}} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix} \mathbf{i} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix} \mathbf{j} \tag{2.8}$$

and

$$\overline{\mathbf{G}} = \frac{1}{Re_a} \left(\begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ f \end{bmatrix} \mathbf{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ g \end{bmatrix} \mathbf{j} \right) \tag{2.9}$$

are also functions of $\mathbf{Q}$ as presented above and via the following relations.

Pressure, p, is related to $\mathbf{Q}$ through

$$p = (\gamma - 1) \left[e - \frac{1}{2} \rho \left(u^2 + v^2 \right) \right] \tag{2.10}$$

which is an equation of state for an ideal gas.

The viscous stress terms are given by

$$\tau_{xx} = (\mu + \mu_t) \left(\frac{4}{3} u_x - \frac{2}{3} v_y \right) \tag{2.11}$$

$$\tau_{xy} = (\mu + \mu_t) \left(u_y + v_x \right) \tag{2.12}$$

$$\tau_{yy} = (\mu + \mu_t) \left(\frac{4}{3} v_y - \frac{2}{3} u_x \right) \tag{2.13}$$

and the viscous energy flux is defined by

$$f = u\tau_{xx} + v\tau_{xy} + \left(\frac{\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}}{\gamma - 1} \right) T_x \tag{2.14}$$

$$g = u\tau_{xy} + v\tau_{yy} + \left(\frac{\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}}{\gamma - 1} \right) T_y \tag{2.15}$$

where Pr is the Prandtl number and Pr_t is the turbulent Prandtl number. Although the Prandtl number is defined by

$$Pr = \frac{c_p \mu}{\kappa_t} \tag{2.16}$$

for specific heat at constant pressure, c_p, and thermal conductivity, κ_t, it is assumed constant with values of $Pr = 0.72$ and $Pr_t = 0.90$ for the range of flows studied. The non-dimensional dynamic viscosity, μ, is related to temperature, $\tilde{T}$, using Sutherland's law

$$\mu = \frac{\tilde{\mu}}{\tilde{\mu}_\infty} = \left[\frac{\tilde{T}_\infty + 110}{\tilde{T} + 110} \right] \left(\frac{\tilde{T}}{\tilde{T}_\infty} \right)^{\frac{3}{2}} \tag{2.17}$$

where $\tilde{T}$ is expressed in Kelvin. Finally, the dynamic eddy viscosity μ_t is modeled as discussed in the following section.

Laminar solutions are obtained by keeping $\mu_t = 0$ and the inviscid or Euler equations correspond to μ being set to zero as well.

2.2 Spalart-Allmaras Turbulence Model

The Navier-Stokes equations when coupled with an equation of state and appropriate boundary conditions fully describe the physics of compressible, turbulent fluid flows. However, when these equations are descretised on finite grids and integrated though finite time intervals much of the high frequency information is lost. Due to current limits on computer memory and processor speed, Direct Numerical Simulation (DNS) of turbulence is not feasible for any practical study. Instead, the flow variables are separated into mean and fluctuating components, forming the Reynolds-Averaged Navier-Stokes (RANS) equations. The effects of these turbulent fluctuations on the mean flow are approximated by adding a dynamic eddy viscosity term μ_t to the dynamic viscosity μ as shown in equations 2.11, 2.12, 2.13, 2.14, and 2.15. A third approach, namely Large Eddy Simulation (LES),

involves a compromise between DNS and RANS in which large scale fluctuations are computed directly while the sub-grid scale turbulence is modeled. This approach is beyond the scope of this thesis.

There are various methods of determining the dynamic eddy viscosity to be used with the RANS equations ranging from algebraic, to various one- and two-equation models. As reported by Walsh [40], Godin, Nelson and Zingg [41, 42] studied the performance of several models commonly used, including the Baldwin-Barth [43] and Spalart-Allmaras [44] one-equation models as well as a two-equation model, on structured grids. Although all these models performed well based on experimentally determined surface pressures and velocity profiles, the Spalart-Allmaras model performed slightly better in complex flows.

In this work the one-equation model of Spalart and Allmaras is used to simulate the effects of turbulence in high Reynolds number flows. This transport equation for the working variable $\tilde{\nu}$ may be written in conservative, non-dimensional form as

$$\begin{aligned}\frac{\partial \tilde{\nu}}{\partial t} + (\mathbf{v}\cdot\nabla)\,\tilde{\nu} \;=\; & \frac{1}{\sigma Re_a}\left[\nabla\cdot\left[(\nu+\tilde{\nu})\,\nabla\tilde{\nu}\right] + c_{b2}\,(\nabla\tilde{\nu})^2\right] \\ & + \frac{c_{b1}}{Re_a}\left[1-f_{t2}\right]\tilde{S}\tilde{\nu} \\ & - \frac{1}{Re_a}\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left[\frac{\tilde{\nu}}{d}\right]^2 \\ & + f_{t1}Re_a\,|\Delta\mathbf{v}|^2\end{aligned} \tag{2.18}$$

The dynamic eddy viscosity, μ_t, which couples the turbulence model to the flow equations, is obtained from the kinematic eddy viscosity ν_t by

$$\mu_t = \frac{\nu_t}{\rho} \tag{2.19}$$

where

$$\nu_t = \tilde{\nu} f_{v1} \tag{2.20}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.21}$$

and

$$\chi = \frac{\tilde{\nu}}{\nu} \tag{2.22}$$

Additionally, $\tilde{S}$ is

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \tag{2.23}$$

where S is the magnitude of the vorticity,

$$S = \left| \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right| \tag{2.24}$$

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.25}$$

and d is the distance to the closest wall.

The function f_w is

$$f_w = g \left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right]^{\frac{1}{6}} \tag{2.26}$$

$$g = r + c_{w2} \left(r^6 - r \right) \tag{2.27}$$

$$r = \frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2} \tag{2.28}$$

Following the recommendations of the paper [44], f_w is limited for values of r above 10.

The f_{t2} function is given by

$$f_{t2} = c_{t3} \exp\left(-c_{t4} \chi^2 \right) \tag{2.29}$$

and the trip function, f_{t1} is

$$f_{t1} = c_{t1} g_t \exp\left(-c_{t2} \frac{w_t^2}{|\Delta \mathbf{v}|^2} \left[d^2 + g_t^2 d_t^2 \right] \right) \tag{2.30}$$

where

$$g_t = \min\left(0.1, \frac{|\Delta \mathbf{v}|}{w_t} \Delta x \right) \tag{2.31}$$

and $|\Delta \mathbf{v}|$ is the difference between the speed at the field point and that at the trip, Δx is the grid spacing along the wall at the trip, and w_t is the vorticity at the trip location.

The various constants used are:

$$\begin{array}{cccc} c_{b1} = 0.1355 & c_{b2} = 0.622 & \sigma = 2/3 & \kappa = 0.41 \\ c_{t1} = 5 & c_{t2} = 2 & c_{t3} = 1.2 & c_{t4} = 0.5 \\ c_{w1} = \dfrac{c_{b1}}{\kappa^2} + (1 + c_{b2}) & c_{w2} = 0.3 & c_{w3} = 2 & c_{v1} = 7.1 \end{array} \tag{2.32}$$

Chapter 3

Spatial Discretisation

The Navier-Stokes equations, Eq. 2.1, are continuous in space and time. As with most nonlinear integral or differential equations, closed-form, analytic solutions are rare, and a discrete approach is used to obtain a numerical representation of the solution. In this work the discretisation will be treated as a two-step process; first the spatial domain is discretised into a finite grid of cells. Next the solution will be allowed to progress through time at each of these cells until a stationary solution is obtained. Since we are only interested in steady-state solutions, no effort is put forth to maintain time-accuracy.

This chapter will deal with the spatial discretisation, while the temporal relaxation is the topic of the following one.

3.1 Finite-Volume Formulation

No specific structure is assumed about the grids used in this study. The discrete domain consists of a collection of arbitrary polygons used as control volumes. These polygons are constructed as the centroid-median dual of a primary grid as shown in figure 3.1, although this is not required. The values of the conservative variables are stored at the centroid of each cell (likely a primary grid node) and are assumed to be constant throughout the cell. The more the mesh is refined, the better this assumption becomes.

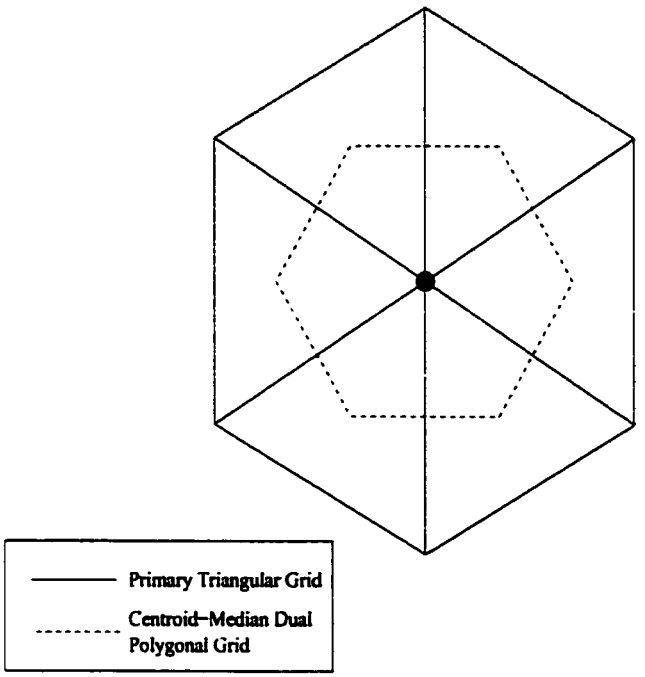

Figure 3.1: Typical Control Volume

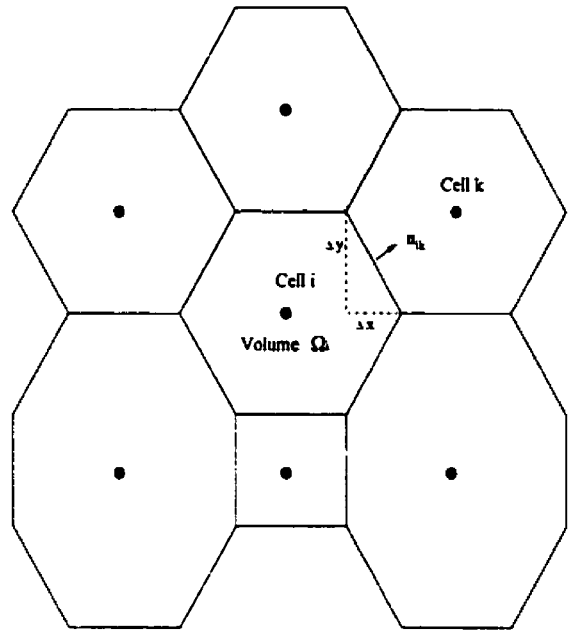

Figure 3.2: Typical Interior Cell

A typical cell is shown in Fig. 3.2. This i^{th} cell has area Ω_i and each side has outward normal $\mathbf{n}_{ik} = \mathbf{n}_{x_{ik}} + \mathbf{n}_{y_{ik}} = \Delta y_{ik}\mathbf{i} - \Delta x_{ik}\mathbf{j}$. Cell areas and face normals are precomputed and stored to reduce computational effort.

3.2 Spatial Derivatives

Before formulating the Navier-Stokes equations in the context of a finite-volume discretisation, one must consider the treatment of the spatial derivatives required by the viscous flux calculations. In this work the spatial derivatives are precalculated for each cell by summing around neighbouring control volumes, as described by Greiner [45].

These spatial derivatives, which are also assumed to be constant, are stored along with the conserved variables at the cell centroid. With the assumption of constant spatial derivatives within a cell, one may write

$$\Omega \nabla \phi = \int_{\Omega} \nabla \phi \, dA \tag{3.1}$$

for an arbitrary variable ϕ. Next Gauss's theorem in two dimensions is used to convert the

area integral into a line integral around the perimeter of the closed control volume

$$\Omega \nabla \phi = \oint_{\partial\Omega} \mathbf{n}\phi \, dS \tag{3.2}$$

where $\mathbf{n}$ is again directed out of the volume. Note that $\mathbf{n}$ is a unit vector distinct from $\Delta y_{ik}\mathbf{i} - \Delta x_{ik}\mathbf{j}$. Finally, this line integral is the sum over the edges of the cell, providing the desired discrete form

$$(\nabla \phi)_i = \frac{1}{\Omega} \sum_{k=1}^{N_i} \left[\frac{\phi_i + \phi_k}{2} \mathbf{n}_{x_{ik}} + \frac{\phi_i + \phi_k}{2} \mathbf{n}_{y_{ik}} \right] \tag{3.3}$$

If the cell contains any boundary faces, the contribution from those must be considered as well. That is

$$\nabla \phi = \frac{1}{\Omega} \left(\sum_{k=1}^{N_i} \left[\frac{\phi_i + \phi_k}{2} \mathbf{n}_{x_{ik}} + \frac{\phi_i + \phi_k}{2} \mathbf{n}_{y_{ik}} \right] + \sum_{bdy=1}^{N_{bdy}} \left[\phi_{bdy} \mathbf{n}_{x_{bdy}} + \phi_{bdy} \mathbf{n}_{y_{bdy}} \right] \right) \tag{3.4}$$

where ϕ_{bdy} is the value of the variable at the boundary face and N_{bdy} is the number of boundary faces bordering the cell in question.

Lastly, to reduce computational effort, one may note that

$$\sum_{k=1}^{N_i} \mathbf{n}_{ik} = 0 \tag{3.5}$$

for a closed cell bounded by N_i field faces. Thus the ϕ_i term is dropped in the sum in Eq. 3.3 for interior cells.

3.3 Fluxes

The same approach as is used for the spatial derivatives is used for the flux integration. At steady-state the Navier-Stokes equations, Eq. 2.1, may be written as

$$\int_\Omega \nabla \cdot \overline{\mathbf{F}} \, dA - \int_\Omega \nabla \cdot \overline{\mathbf{G}} \, dA = 0 \tag{3.6}$$

Again, Gauss's theorem is used to transform the area integrals into line integrals giving

$$\oint_{\partial\Omega} \overline{\mathbf{F}} \cdot \mathbf{n} \, dS - \oint_{\partial\Omega} \overline{\mathbf{G}} \cdot \mathbf{n} \, dS = 0 \tag{3.7}$$

Finally the line integrals may be converted into sums over all edges of each control volume converting the system of integral equations, Eq. 2.1, to a system of coupled nonlinear algebraic equations.

The remainder of this thesis will consider the Spalart-Allmaras turbulence model, Eq. 2.18, to be coupled with the flow equations, yielding a block size of five for the equations at each cell. All references to laminar cases will simply omit the turbulence model equation and set μ_t to zero in the flow equations, producing a block size of four.

3.3.1 Inviscid Fluxes

Since we assume constant variables within each cell, we may assume the flux at a control volume edge is similarly constant and equal to the average from the two cells on either side of that edge. Thus the inviscid flux term may be written in a discrete form as

$$\oint_{\partial\Omega} \overline{\mathbf{F}} \cdot \mathbf{n}\, dS = \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i + \overline{\mathbf{F}}_k}{2} \right) \cdot \mathbf{n}_{ik} \tag{3.8}$$

where

$$\overline{\mathbf{F}} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \\ 0 \end{bmatrix} \mathbf{i} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \\ 0 \end{bmatrix} \mathbf{j} \tag{3.9}$$

and N_i is the number of field faces of cell i.

3.3.2 Viscous Fluxes

Precisely the same approach is used for the viscous fluxes. This is given in discrete form as

$$\oint_{\partial\Omega} \overline{\mathbf{G}} \cdot \mathbf{n}\, dS = \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{G}}_i + \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} \tag{3.10}$$

where

$$\overline{\mathbf{G}} = \frac{1}{Re_a} \left(\begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ f \\ -Re_a u\tilde{\nu} + \frac{(\nu+\tilde{\nu})}{\sigma}\tilde{\nu}_x \end{bmatrix} \mathbf{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ g \\ -Re_a v\tilde{\nu} + \frac{(\nu+\tilde{\nu})}{\sigma}\tilde{\nu}_y \end{bmatrix} \mathbf{j} \right) \tag{3.11}$$

The two terms added to the fifth equation are referred to as the advective flux and the diffusive flux respectively. Some authors choose to include the advective term in the inviscid

flux tensor; however, this presentation is used as a reminder that turbulence is a viscous phenomenon.

3.4 Source Terms

The remainder of the turbulence model consists of source terms. For completeness this is included as

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ S_{\text{diff}} + S_{\text{prod}} + S_{\text{dest}} + S_{\text{trip}} \end{bmatrix} \tag{3.12}$$

where the subscripts refer to the diffusion, production, destruction and trip source terms as defined below:

$$S_{\text{diff}} = c_{b2} \left(\nabla \tilde{\nu}\right)^2 \tag{3.13}$$

$$S_{\text{prod}} = \frac{c_{b1}}{Re_a} \left[1 - f_{t2}\right] \tilde{S} \tilde{\nu} \tag{3.14}$$

$$S_{\text{dest}} = -\frac{1}{Re_a} \left[c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right] \left[\frac{\tilde{\nu}}{d}\right]^2 \tag{3.15}$$

$$S_{\text{trip}} = f_{t1} Re_a \left|\Delta \mathbf{v}\right|^2 \tag{3.16}$$

Care must be taken not to confuse the diffusive source term with the diffusive flux term.

3.5 Artificial Dissipation

An artificial dissipation term is used to promote stability of the numerical flow equations. A fourth-difference biharmonic operator is used except near shocks where a stronger second-difference Laplacian operator is utilized when triggered by a pressure switch. This is similar to a method developed by Jameson and Mavriplis [46]. The rest of section 3.5 is taken largely from the development by Wehner [38].

The application of artificial dissipation results in a term

$$\mathbf{AD}_i = \sum_{k=1}^{N_i} \lambda_{ik} \left[\varepsilon_{ik}^{(2)} \left(\mathbf{Q}_k - \mathbf{Q}_i\right) - \varepsilon_{ik}^{(4)} \left(\mathbf{L}_k - \mathbf{L}_i\right)\right] \tag{3.17}$$

The second-difference Laplacian operator is:

$$\sum_{k=1}^{N_i} \varepsilon_{ik}^{(2)} \left(\mathbf{Q}_k - \mathbf{Q}_i\right) \tag{3.18}$$

where

$$\varepsilon_{ik}^{(2)} = \kappa^{(2)} \sum_{k=1}^{N_i} \frac{|p_k - p_i|}{(p_k + p_i)} \tag{3.19}$$

The fourth-difference biharmonic operator is:

$$\sum_{k=1}^{N_i} \varepsilon_{ik}^{(4)} \left(\mathbf{L}_k - \mathbf{L}_i\right) \tag{3.20}$$

where

$$\mathbf{L}_i = \sum_{k=1}^{N_i} \left(\mathbf{Q}_k - \mathbf{Q}_i\right) \tag{3.21}$$

and

$$\varepsilon_{ik}^{(4)} = \max\left[0\,,\ \left(\kappa^{(4)} - \varepsilon_{ik}^{(2)}\right)\right] \tag{3.22}$$

The coefficient $\varepsilon_{ik}^{(4)}$ is a switch that turns on the fourth-difference dissipation whenever the second-difference dissipation is turned off. Both $\kappa^{(2)}$ and $\kappa^{(4)}$ are user-defined constants.

To ensure the dissipation has an appropriate magnitude, it is scaled by

$$\lambda_{ik} = |\mathbf{v}| + a \tag{3.23}$$

where $|\mathbf{v}|$ and a are calculated based on the average of the conserved variables at the cells in question:

$$\mathbf{Q}_{avg} = \frac{\mathbf{Q}_i + \mathbf{Q}_k}{2} \tag{3.24}$$

Dissipation is added for all interior or field faces, but there is no contribution from boundary faces.

A first-order upwind formulation is used with the turbulence model, rather than adding artificial dissipation to it. Greiner [45] found this to be a more stable approach and claimed that it ensured the convergence of the turbulence model at about the same rate as the flow equations for an explicit solver.

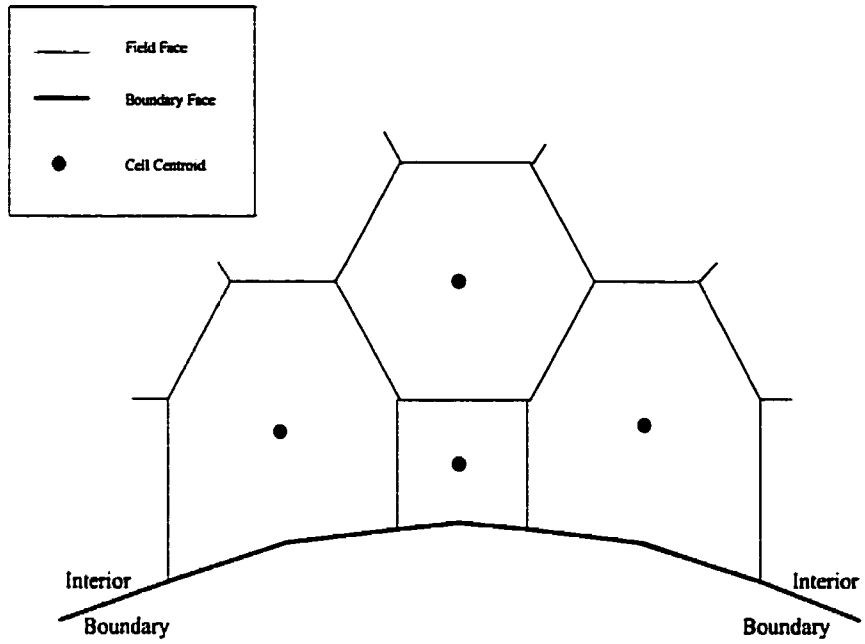

Figure 3.3: Typical Boundary Cell

3.6 Boundary Conditions

In order to solve any system of Partial Differential Equations (PDEs), appropriate initial and boundary conditions must be specified. Solution of the discrete Navier-Stokes equations will be initiated from free-stream (∞) values and iterative updates will occur until a steady solution has been obtained.

A typical boundary cell configuration is shown in figure 3.3. A boundary cell contains field faces and one or more boundary faces. When summing over all faces of a boundary cell both types must be considered separately.

The value of the conserved variable vector at the boundary is calculated in two steps; first the value at the boundary is extrapolated from the interior and then the appropriate boundary condition is applied to the extrapolated value. The same approach is used for the spatial derivatives, which are required for viscous flux calculations at boundary faces.

For this work a simple zeroth-order extrapolation is used whereby the value from the interior is simply copied to the boundary face. Higher order extrapolations are possible.

The various constraints imposed at boundary faces are discussed below.

3.6.1 Inner Boundaries

An inner boundary face is a solid wall, typically the airfoil surface. Fluid may not flow into or out of an inner boundary face; thus it is required that the velocity be tangential to the body surface at all times. Details of how this solver treats inner boundaries when run in inviscid mode have been stated by Wehner [38].

With viscous flows, however, the effects of viscosity cause the fluid velocity at the surface to be zero for stationary boundaries. This is commonly referred to as the no-slip boundary condition. Thus two of the four necessary constraints are easily satisfied.

$$u = 0 \quad ; \quad v = 0 \tag{3.25}$$

The remaining two act on temperature and pressure at the boundary face.

At steady state, the temperature of the wall and the temperature of the fluid next to it should be the same. This is expressed as an adiabatic boundary condition

$$\frac{\partial T}{\partial n} = 0 \tag{3.26}$$

where $\frac{\partial}{\partial n}$ is the component of the derivative normal to the boundary surface. Similarly the normal component of pressure is also zero:

$$\frac{\partial p}{\partial n} = 0 \tag{3.27}$$

These last two boundary conditions are easily implemented by extrapolating density, ρ, and pressure, p, from the interior, and adjusting energy, e, accordingly. To ensure proper spatial derivatives at boundary faces for flux calculations, the normal component of the temperature derivative is forced to be zero as well.

3.6.2 Far Field Boundaries

Far field boundaries are located sufficiently far from the airfoil surface, such that free-stream flow may be assumed there. Any disturbances from the body within must be allowed to exit the domain, thus non-reflecting boundary conditions are used to prevent the formation of nonphysical oscillations.

To implement this, one-dimensional Riemann invariants are applied to the component of the flow normal to the boundary, as described by Hirsch [47]. These are given by:

$$R_1 = v_n - \frac{2a}{\gamma - 1} \tag{3.28}$$

$$R_2 = v_n + \frac{2a}{\gamma - 1} \tag{3.29}$$

$$R_3 = \frac{p}{\rho^\gamma} \tag{3.30}$$

$$R_4 = v_t \tag{3.31}$$

where v_n and v_t are the components of velocity, $\mathbf{v}$, normal and tangential to the boundary respectively. Additionally, normal components of the spatial derivatives are zeroed at an outer boundary.

These four invariants are taken from free-stream and extrapolated values according to Table 3.1. From these, the conserved flow vector at the boundary face may be reconstructed.

Boundary Type	Freestream	Extrapolated
Subsonic Inflow ($-a < v_n < 0$)	R_1,R_3,R_4	R_2
Subsonic Outflow ($0 < v_n < a$)	R_1	R_2,R_3,R_4
Supersonic Inflow ($v_n < -a$)	R_1,R_3,R_4,R_2	
Supersonic Outflow ($v_n > a$)		R_1,R_3,R_4,R_2

Table 3.1: Riemann Invariant Calculation and Extrapolation

3.6.3 Symmetry Boundaries

A third possible boundary type enforces symmetry in the flow field. For example, a steady flow around a symmetric body with no angle of attack would be symmetric. By the use of symmetry boundaries, one may solve on a grid over half the object, thus reducing the time and memory required for the computation.

In this solver, symmetry boundary conditions are implemented for laminar flow, but are not part of this study. For completeness these constraints are outlined below.

As with inviscid inner boundaries, the flow is forced to be tangential to all symmetry boundary faces. Again, details may be taken from Wehner [38]. For viscous flows, the normal components of the spatial derivatives are zeroed as well.

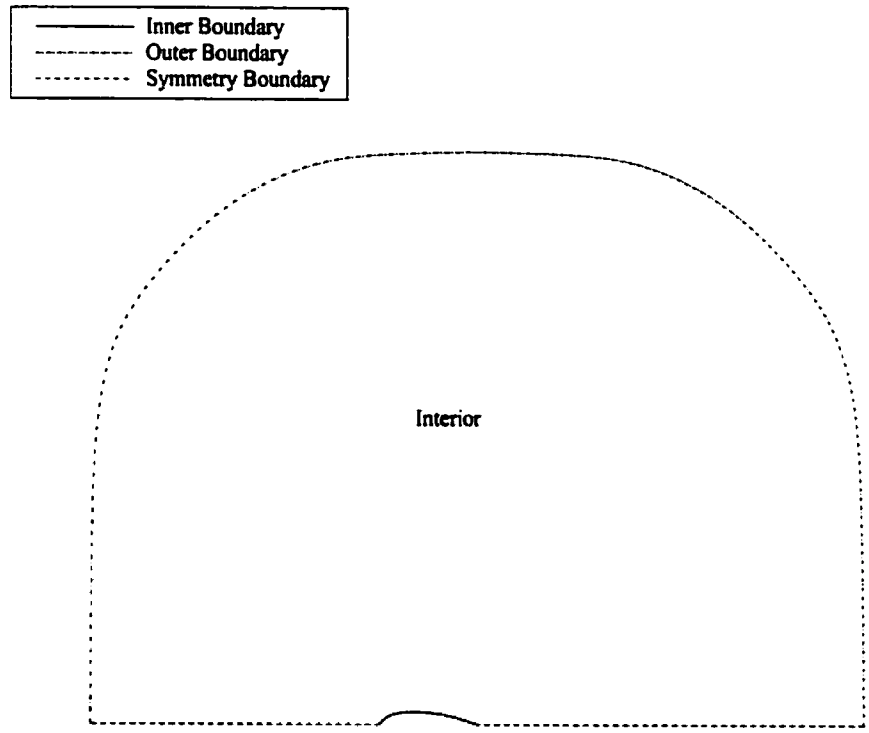

Figure 3.4: Various Boundary Types

Figure 3.4 shows the three different boundary types.

3.6.4 Turbulence Model

The choice of $\tilde{\nu}$ as the transport variable for the Spalart-Allmaras turbulence model greatly simplifies the boundary treatment for that model. The conserved turbulence variable has the property that it approaches zero both on the airfoil surface and in the far field. For inner boundary faces, components of the spatial derivatives normal to the boundary faces are zeroed as well. For outer boundaries, $\tilde{\nu}$ is extrapolated at outflow and set at inflow. To promote positivity of the model, a small positive value (10^{-3}) is used instead of zero for inflow.

3.7 Residual Formulation

The complete system of discrete equations to be solved may now be fully defined. Our starting point is the steady Navier-Stokes equations in line integral form, Eq. 3.7. By summing over all edges for the i^{th} cell we obtain

$$\sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i + \overline{\mathbf{F}}_k}{2} \right) \cdot \mathbf{n}_{ik} - \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{G}}_i + \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} = 0 \tag{3.32}$$

for the interior. Should the cell contain any boundary faces, their flux contribution must be considered as well:

$$\begin{aligned} &\textstyle\sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i + \overline{\mathbf{F}}_k}{2} \right) \cdot \mathbf{n}_{ik} - \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{G}}_i + \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} \\ &+ \textstyle\sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{F}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} - \sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{G}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} = 0 \end{aligned} \tag{3.33}$$

When coupled with the turbulence model, a source term appears as

$$\begin{aligned} &\textstyle\sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i + \overline{\mathbf{F}}_k}{2} \right) \cdot \mathbf{n}_{ik} - \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{G}}_i + \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} \\ &+ \textstyle\sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{F}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} - \sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{G}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} - \Omega_i \mathbf{S}_i = 0 \end{aligned} \tag{3.34}$$

Finally, artificial dissipation is added to all field faces to promote numerical stability

$$\begin{aligned} &\textstyle\sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i + \overline{\mathbf{F}}_k}{2} \right) \cdot \mathbf{n}_{ik} - \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{G}}_i + \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} \\ &+ \textstyle\sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{F}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} - \sum_{bdy=1}^{N_{bdy}} \overline{\mathbf{G}}_{\mathbf{bdy}} \cdot \mathbf{n}_{bdy} - \Omega_i \mathbf{S}_i + \frac{\mathbf{AD}_i}{2} = 0 \end{aligned} \tag{3.35}$$

Equation 3.35 is known as the residual for the i^{th} cell, $\mathbf{R_i}$ and is compactly written as

$$\mathbf{R}_i = \sum_{k=1}^{N_i} \left(\frac{\overline{\mathbf{F}}_i - \overline{\mathbf{G}}_i + \overline{\mathbf{F}}_k - \overline{\mathbf{G}}_k}{2} \right) \cdot \mathbf{n}_{ik} + \sum_{bdy=1}^{N_{bdy}} \left(\overline{\mathbf{F}}_{bdy} - \overline{\mathbf{G}}_{bdy} \right) \cdot \mathbf{n}_{bdy} - \Omega_i \mathbf{S}_i + \frac{\mathbf{AD}_i}{2} \quad (3.36)$$

Additionally, as with the spatial derivatives, Eq. 3.5 applies for closed, interior cells. As a result, the $\overline{\mathbf{F}}_i$ and $\overline{\mathbf{G}}_i$ terms are dropped in the sum for such cells.

In Eq. 3.36, the continuous set of integral equations, Eq .2.1 has been reduced to a discrete nonlinear system of algebraic equations. The following chapter discusses the solution to $\mathbf{R(Q)} = 0$.

Chapter 4

Solution Algorithm

In the previous chapter the discrete, nonlinear system of algebraic equations, $\mathbf{R}(\mathbf{Q}) = \mathbf{0}$, has been developed. This chapter discusses methods for solving such systems.

4.1 Newton's Method

Our starting point for solving a nonlinear system of the form $\mathbf{R}(\mathbf{Q}) = \mathbf{0}$ is Newton's method which may be written as

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^n \Delta \mathbf{Q}^n = -\mathbf{R}(\mathbf{Q}^n) \tag{4.1}$$

where $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$ is the Jacobian of the residual, $\mathbf{R}$, with respect to the conserved variables vector, $\mathbf{Q}$. Newton's method requires iterative updates of the form

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta \mathbf{Q}^n \tag{4.2}$$

starting at an initial guess, $\mathbf{Q}^0$ and proceeding until the system is satisfied. In Hurricane iteration begins from free-stream, (∞), values and continues until the L_2 norm of the nonlinear residual, $\mathbf{R}(\mathbf{Q})$ is less than a user prescribed tolerance.

4.2 Matrix-Free GMRES

Using Newton's method, Eqs. 4.1 and 4.2, the nonlinear system of algebraic equations has been reduced to a linear system of algebraic equations of the generic form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.3}$$

and a linear update. For each Newton iteration, the corresponding linear system, Eq. 4.1, must be solved. These multiple solves of potentially large linear systems call for an efficient solution method. As such, direct solvers, with their related large processor and storage requirements, are dismissed as an option.

The alternative is to use an iterative solver. Krylov subspace techniques are iterative methods which tend to display better convergence properties than the classical relaxation methods such as Jacobi, Gauss-Seidel or Successive Over-Relaxation (SOR) [20, 23, 48]. Additionally, Krylov methods do not require diagonal dominance. The technique known as the Generalized Minimal RESidual (GMRES) method developed by Saad and Schultz [16] is popular among Krylov subspace techniques and is used here. Venkatakrishnan [20] compared the Chebyshev semi-iteration technique to GMRES for structured CFD problems and found GMRES to be marginally better. Additionally, Knoll and McHugh [26, 49] compared Conjugate Gradient Squared (CGS), Transposed Free Quasi-Minimal Residual (TFQMR) and Bi-Conjugate Gradient(BCG) solvers to GMRES, with the latter being found superior for an incompressible steady flow. Lastly, GMRES has been found to be faster than BCG and Bi-Conjugate Gradient Stabilized (Bi-CGSTAB) solvers for the few aerodynamic scenarios tested by Pueyo [34, 35].

Before proceeding, an important distinction must be made. Newton's method is an iterative method for solving nonlinear systems. At each step of Newton's method a linear system is solved using GMRES, itself being an iterative method for solving linear systems. The distinction must be clear between the outer, nonlinear, Newton iterations and the inner, linear, GMRES iterations. As such, any talk of optimisation must not focus solely on reducing the number of inner iterations or outer iterations per say, but rather reduce a global measure of computational effort. This is further discussed in section 5.2.

For a linear system of the form $\mathbf{Ax} = \mathbf{b}$, GMRES searches the Krylov subspace, $\mathbf{K}_m$, for the iterate $\mathbf{x}_m \in \{\mathbf{x}_0 + \mathbf{K}_m\}$ which minimizes the L_2 norm of the linear residual given by

$$\mathbf{r}_m = \mathbf{b} - \mathbf{Ax} \tag{4.4}$$

The subspace is of the form

$$\mathbf{K}_m = \text{span}\{\mathbf{v}_1, \mathbf{Av}_1, \mathbf{A}^2\mathbf{v}_1, ..., \mathbf{A}^{m-1}\mathbf{v}_1\} \tag{4.5}$$

where the vector, $\mathbf{v}_1$ is defined as

$$\mathbf{v}_1 = \frac{\mathbf{r}_0}{||\mathbf{r}_0||_2} = \frac{\mathbf{b} - \mathbf{A}\mathbf{x}_0}{||\mathbf{b} - \mathbf{A}\mathbf{x}_0||_2} \tag{4.6}$$

GMRES is guaranteed to converge in, at most, N steps where N is the size of the system. Preconditioning is used to ensure many fewer GMRES iterations are required, as described in Section 4.3. The storage required by GMRES increases linearly with the number of search directions in the Krylov subspace, and CPU time increases quadratically. To avoid the excessive cost of a large number of inner iterations, GMRES itself may be applied iteratively by restarting it after $m << N$ iterations and using x_m as the initial guess upon restart. This restarted version of GMRES will be referred to as GMRES(m).

One extremely attractive feature of GMRES for solving systems of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, is that the matrix, $\mathbf{A}$, is never required explicitly. This greatly reduces the storage requirements of the solver. GMRES requires only matrix-vector products to form the subspace search directions. These matrix-vector products may be approximated with a forward-difference approximation or Frechet derivative given by

$$\mathbf{A}\mathbf{v}_1 \simeq \frac{\mathbf{R}(\mathbf{Q} + \epsilon\mathbf{v}_1) - \mathbf{R}(\mathbf{Q})}{\epsilon} \tag{4.7}$$

where ϵ is a small scalar used for the perturbation. When using the Frechet derivative to approximate the matrix-vector products rather than the matrix, $\mathbf{A}$, itself, we refer to the linear solver as matrix-free GMRES.

Hurricane uses a strategy for choosing ϵ as proposed by Nielsen et al. [50] which is

$$\epsilon \simeq \frac{\sqrt{\epsilon_m}}{||\mathbf{v}_1||_2} \tag{4.8}$$

where ϵ_m is "machine zero" for the particular processor used. Care in choosing ϵ is exercised since it has been shown that Eq. 4.7 is very sensitive to ϵ, especially when using forward differencing [51]. It has been suggested that dividing $||\mathbf{v}_1||_2$ by the number of entries in the vector would improve scalability, although this has not been investigated.

A second, equally attractive feature of using matrix-free GMRES to solve the linear problem is that the effects of the Jacobian matrix, $\mathbf{A} = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, are fully included. An analytical linearization of the residual function, Eq. 3.36, will certainly include several simplifying assumptions regarding quantities which are difficult to linearize such as the spectral-radius and pressure switch in the artificial dissipation (Eq. 3.17), Sutherland's law (Eq. 2.17) and the turbulence model (Eq. 2.18). Additionally, the Jacobian upon which a

preconditioner is based need not maintain the strict accuracy of the system itself. This will be further discussed in the following section.

Finally, with an iterative solver such as GMRES, one may control how accurately the linear systems are being solved. It has been shown by Pueyo and Zingg that while the nonlinear solution is relatively far from the converged solution, the correction made by Newton's method can be very inaccurate. Solving the linear system of equations very accurately does not, in general, equally reduce the residual of the nonlinear system of equations, thus wasting CPU time [35]. This is known as oversolving and is to be avoided. The effect of how accurately the linear system is solved on the overall nonlinear convergence is discussed in Section 5.4.

4.3 Preconditioning

As was mentioned in section 4.2, GMRES is guaranteed to converge in, at most, N steps. Unfortunately, performance degrades rapidly as the number of inner iterations increases. To avoid large GMRES iteration counts, the linear system is preconditioned. Preconditioning converts the linear system into one which is easier to solve by an iterative solver. An ideal preconditioner transforms the diverse eigenvalue spectrum of the inversion matrix, $\mathbf{A}$, to one which is clustered around unity.

In this work a right-preconditioned system is formed, whereby the original system, $\mathbf{Ax} = \mathbf{b}$, is converted to

$$\mathbf{AP}^{-1}\mathbf{Px} = \mathbf{b} \tag{4.9}$$

When a linear system is right-preconditioned, the norm of the modified system is the same as that of the unmodified one. As such, the norm is an accurate indicator of convergence for the inner iterations, which is not the case for the alternative, left-preconditioning.

In the ideal scenario, $\mathbf{P}^{-1} = \mathbf{A}^{-1}$ such that the product $\mathbf{AP}^{-1} = \mathbf{I}$. Of course this is a pedagogical case since if $\mathbf{A}^{-1}$ were known then the linear system would be solved. Instead, two approximations are used as described in sections 4.3.1 and 4.3.2.

4.3.1 ILU Factorization

Incomplete Lower-Upper (ILU) factorizations are efficient preconditioners when used with GMRES [23, 52, 20, 34, 31]. The theory behind an ILU preconditioner is straight-

forward. We approximate the matrix **A** with the matrix **P** where

$$\mathbf{A} \simeq \mathbf{P} = \mathbf{LU} \tag{4.10}$$

where **L** is lower-triangular and **U** upper-triangular. The incomplete aspect arises from the accuracy of the factorization. The more new non-zero entries permitted to appear in the factorization when compared to the original matrix, the more accurate the approximation. Of course the time and storage required to form and apply the preconditioner increases with the level of fill allowed.

An ILU preconditioner with a level of fill, p, will be denoted ILU(p). Zero fill or ILU(0) corresponds to a Gaussian-elimination process whereby the element is dropped unless a nonzero element exists in the original matrix. Increased levels of fill correspond to new nonzero entries being retained. Due to the block nature of the problems being solved, the fill, p, acts on nonzero blocks, as opposed to individual entries. The effects of the fill parameter on convergence are shown in section 5.5.

4.3.2 Approximate Jacobian Matrix

The second approximation used in forming the preconditioner is a result of the matrix, **P**, upon which the factorization is based. Initially one would think that the most effective matrix would be the true Jacobian, **A**. This is not the case. It has been shown by Pueyo and Zingg [36] that the use of a well chosen approximate Jacobian can produce better efficiency than the full one. For nonsymmetric, nondiagonally dominant matrices such as **A**, it has further been shown that the incomplete factors can be more ill conditioned than the original matrix [53]. As such, the recursion associated with the forward and backward solves involved may become unstable [54, 55].

The off-diagonal dominance of the preconditioning Jacobian is reduced by using a modified second-difference dissipation stencil as reported by various authors [31, 34, 35, 36, 37]. This is implemented as a linear combination of the Laplacian and biharmonic operators found in equation 3.17. The preconditioner coefficients are given by

$$\varepsilon_l^{(2)} = \varepsilon_r^{(2)} + \sigma_l \varepsilon_r^{(4)} \tag{4.11}$$

where σ_l is a free parameter. The effects of σ_l on convergence are shown in section 5.3.

Another feature of an approximate Jacobian is that it traditionally requires less storage then the full Jacobian [34, 31]. The use of Eq. 4.11 usually collapses the size of

the stencil required, both for inviscid solvers and for structured solvers. With the former, next-to-nearest neighbour (NTNN) stencils are required for dissipation only, since no spatial derivatives are used in the flux calculation. Eq. 4.11 collapses the entire stencil to nearest-neighbours (NN) only. With structured solvers, a compact stencil is traditionally used whereby the spatial derivatives are calculated at half-nodes and the fluxes are computed at the nodes. This also avoids the next-to-nearest neighbour requirement for structured solvers.

With Hurricane, however, the use of Eq. 3.3 for the spatial derivative calculation combined with Eq. 3.36, produces a next-to-nearest neighbour stencil for the viscous flux contribution to the residual. Hence, the size of the matrix is not necessarily reduced for this case. Using nearest-neighbour only contributions in the preconditioner has been attempted as is discussed in section 5.6, but is dismissed as a feasible option at that point.

Certainly this could be a topic of further research. A recent study uses an ILU(0) factorization of the distance-1 Jacobian matrix [30]. This matrix is based on a lower-order discretisation of the equations because it involves only the edge and vertex neighbours of a cell. This is not attempted here but is worthy of further investigation.

Even though the size of the matrix may not be reduced, one still benefits from the increased diagonal dominance. As mentioned in section 4.2, assumptions may be made with the lower-order Jacobian yet the system being solved using the matrix-free algorithm is the full system. The preconditioning matrix only affects the convergence and robustness of the inner iterations; it does not affect the convergence of the outer iterations or the accuracy of the solution.

4.4 Matrix Reordering

The ordering of the unknowns of the linear system plays a significant role in the convergence of the preconditioned iterative solver [56, 20]. Various strategies exist for re-ordering the unknowns, most of which focus on reducing the bandwidth of the matrix. Blanco and Zingg compared Reverse Cuthill-McKee (RCM), Nested Dissection (ND) and Quotient Minimum Degree (QMD) reorderings to the natural ordering of the unknowns for an unstructured Euler solver and found RCM to be superior [32]. Also Pueyo compared various domain decomposition, double bandwidth, minimum neighbouring and RCM reorderings to the natural one and again the results favoured RCM for a structured Navier-

Stokes solver [34].

Therefore, no further study is undertaken here. Reverse Cuthill-McKee reordering is used to reduce the bandwidth of the linear systems. Section 5.9 shows the effects of the reordering on convergence.

4.5 Linearization

The Jacobian upon which the preconditioner is based, comes from an approximate linearization of the nonlinear residual, Eq. 3.36. The linearization is referred to as approximate due to several simplifying assumptions outlined below:

- μ is assumed independent of $\mathbf{Q}$
- μ_t is assumed independent of $\mathbf{Q}$
- λ_{ik} is assumed independent of $\mathbf{Q}$
- Eq. 4.11 is used for dissipation

The linearization may be subdivided into diagonal and off-diagonal blocks.

4.5.1 Diagonal Blocks

Keeping the above assumptions in mind, one may begin linearizing Eq. 3.36 to obtain

$$\begin{aligned}\frac{\partial \mathbf{R}_i}{\partial \mathbf{Q}_i} = & \sum_{k=1}^{N_i} \left(\frac{1}{2}\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i} + \frac{1}{2}\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_i} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_i} \right) \cdot \mathbf{n}_{ik} + \frac{1}{2}\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_i} \\ & + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_i} - \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_i} \right) \cdot \mathbf{n}_{bdy} - \Omega_i \frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_i} \end{aligned} \tag{4.12}$$

which is the diagonal (i,i) block. The eight terms in Eq. 4.12 will be discussed below in order.

$\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i}$ is an inviscid term which appears in boundary cells only. For non-boundary cells Eq. 3.5 applies, thus there is no $\overline{\mathbf{F}}_i$ contribution to $\mathbf{R}_i$. More details on $\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i}$ can be found in Wehner's work [38].

$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i}$ is a viscous term which appears in boundary cells only. Again, Eq. 3.5 applies so there is no $\overline{\mathbf{G}}_i$ contribution to $\mathbf{R}_i$ for interior cells. This term has contributions from

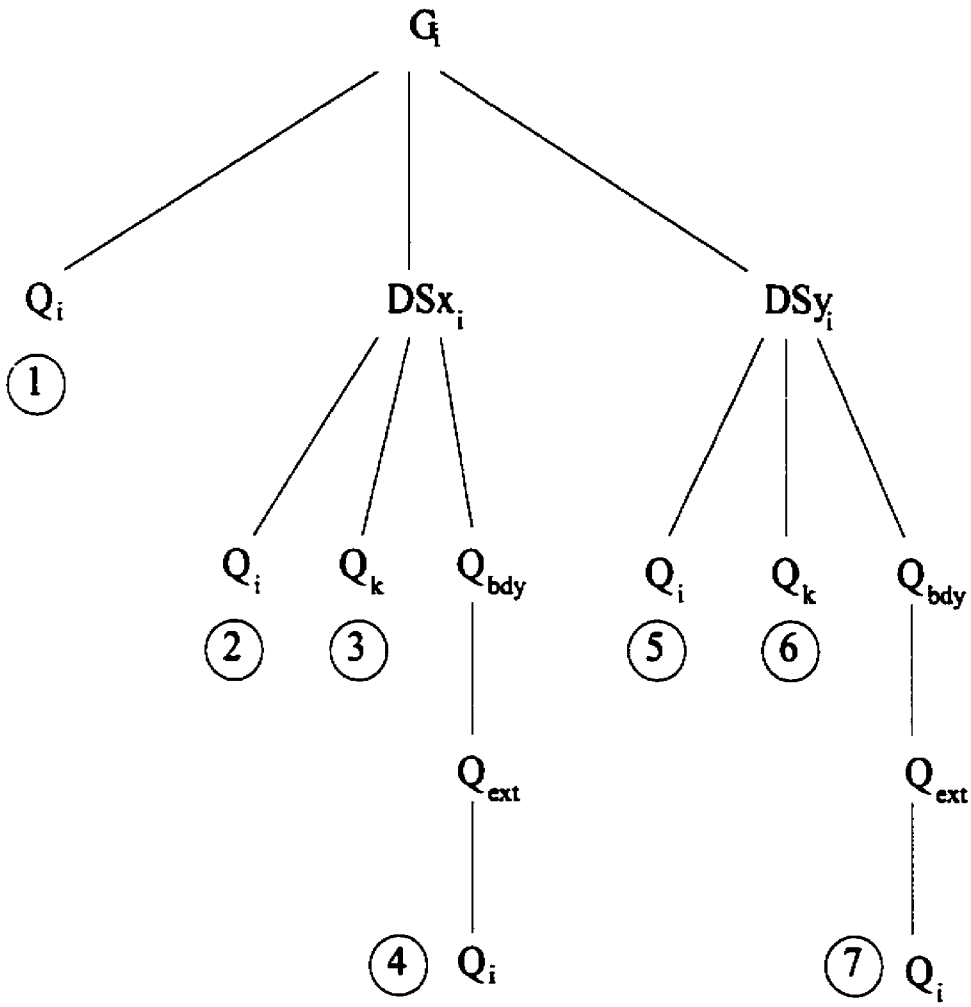

Figure 4.1: Contributions to G_i

several sources. The viscous flux, $\overline{\mathbf{G}}_i$, is a function of both the conserved variables and the spatial derivatives at cell i as illustrated in figure 4.1. The chain rule may be used to obtain

$$\begin{aligned}\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i} &= \left.\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i}\right|_{\mathbf{DSx},\mathbf{DSy}\text{ const.}} \\ &+ \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} + \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{bdy}}\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \\ &+ \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} + \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{bdy}}\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i}\end{aligned} \tag{4.13}$$

where $\mathbf{DSx}_i$ and $\mathbf{DSy}_i$ are the respective x- and y- components of the spatial derivatives of ρ, u, v and T as given by

$$\mathbf{DSx}_i = \begin{bmatrix} \frac{\partial \rho}{\partial x} \\ \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial x} \\ \frac{\partial T}{\partial x} \end{bmatrix} \tag{4.14}$$

and

$$\mathbf{DSy}_i = \begin{bmatrix} \frac{\partial \rho}{\partial y} \\ \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial y} \\ \frac{\partial T}{\partial y} \end{bmatrix} \tag{4.15}$$

The value of the conserved variables vector at the boundary face is $\mathbf{Q}_{bdy}$, while $\mathbf{Q}_{ext}$ is extrapolated from the interior. These correspond to branches 1,2,4,5 and 7 of figure 4.1.

The term is further complicated by the sums involved in the spatial derivative calculation, Eq. 3.4. Upon further examination, the above equation (4.13) contains sums buried in the spatial derivative expansion. Branches 2 and 4 of figure 4.1 can be combined to obtain

$$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} + \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{bdy}}\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} = \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i} \tag{4.16}$$

where $\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i}$ is a complete linearization of Eq. 3.4. Strictly speaking

$$\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i} = \sum_{k=1}^{N_i} \left.\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i}\right|_{\mathbf{Q}_{bdy}\text{ const.}} \mathbf{n}_x + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{bdy}}\mathbf{n}_x\right)\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \tag{4.17}$$

with the sums explicitly recorded.

Similarly branches 5 and 7 of figure 4.1 can be combined with respect to the y component of the spatial derivatives giving

$$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} + \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{bdy}}\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} = \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i} \tag{4.18}$$

where

$$\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i} = \sum_{k=1}^{N_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}}\mathbf{n}_y + \sum_{bdy=1}^{N_{bdy}}\left(\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{bdy}}\mathbf{n}_y\right)\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \tag{4.19}$$

with summation explicit.

Using these clarifications (Eqs:4.16-4.19) one obtains

$$\begin{aligned}\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i} &= \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{DSx},\mathbf{DSy}\text{ const.}} \\ &+ \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i} \\ &+ \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}\end{aligned} \tag{4.20}$$

$\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_i}$ is identically zero. The inviscid flux at one cell is not related to the state at another.

$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_i}$ is fairly straightforward and requires no modification for boundary cells when using a zeroth-order boundary extrapolation. The only contribution from $\mathbf{Q}_i$ on $\mathbf{G}_k$ is through the spatial derivatives. This may be expanded using the chain rule to give

$$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_i} = \frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{DSx}_k}\frac{\partial \mathbf{DSx}_k}{\partial \mathbf{Q}_i} + \frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{DSy}_k}\frac{\partial \mathbf{DSy}_k}{\partial \mathbf{Q}_i} \tag{4.21}$$

This corresponds to branches 3 and 6 of figure 4.1 with indicies i and k switched. The above expression, however, is simple enough that it may be coded explicitly. This avoids the overhead of excessive matrix multiplication.

$\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_i}$ is the linearization of the artificial dissipation which occurs for all field faces. This is discussed by Wehner [38].

If cell i is a boundary cell, the flux at boundary faces must be linearized as well. Of the inviscid and viscous contributions, the former, $\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_i}$, is less complex. Again a chain rule is involved giving

$$\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_i} = \frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{bdy}}\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \tag{4.22}$$

The three Jacobians are explained by Wehner [38], but a brief description is warranted here.

The first, $\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{bdy}}$ is the same function as $\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i}$ and is simply applied using the values at the boundary face, $\mathbf{Q}_{bdy}$. This is the linearization of the flux at the boundary with respect to the conserved variables vector at the boundary. We are linearizing with respect to cell i, however, thus further expansion is undertaken.

The flow variables at the boundary face are based on those extrapolated from the interior, with the appropriate boundary condition applied. $\frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}}$ is the linearization of this boundary condition.

Finally, the extrapolated variables come from those at cell i, as expressed by $\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i}$. For the zeroth-order extrapolation procedure used here, $\frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} = \mathbf{I}$. This factor is included to allow for a more elaborate boundary treatment in the future.

For the viscous flux at a boundary face, the linearization is expanded as shown in figure 4.2. As with $\overline{\mathbf{G}}_i$, $\overline{\mathbf{G}}_{bdy}$ also has contributions both from the conserved variables as well as the spatial derivatives at the boundary face. The chain rule gives

$$\begin{aligned}
\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_i} &= \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{bdy}} \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSx}_{bdy}} \frac{\partial \mathbf{DSx}_{bdy}}{\partial \mathbf{DSx}_{ext}} \frac{\partial \mathbf{DSx}_{ext}}{\partial \mathbf{DSx}_i} \frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSx}_{bdy}} \frac{\partial \mathbf{DSx}_{bdy}}{\partial \mathbf{DSx}_{ext}} \frac{\partial \mathbf{DSx}_{ext}}{\partial \mathbf{DSx}_i} \frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{bdy}} \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSy}_{bdy}} \frac{\partial \mathbf{DSy}_{bdy}}{\partial \mathbf{DSy}_{ext}} \frac{\partial \mathbf{DSy}_{ext}}{\partial \mathbf{DSy}_i} \frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}\bigg|_{\mathbf{Q}_{bdy}\text{ const.}} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSy}_{bdy}} \frac{\partial \mathbf{DSy}_{bdy}}{\partial \mathbf{DSy}_{ext}} \frac{\partial \mathbf{DSy}_{ext}}{\partial \mathbf{DSy}_i} \frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{bdy}} \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i}
\end{aligned} \tag{4.23}$$

corresponding to branches 1, 2, 4, 5, and 7 of figure 4.2. Again, this equation is further complicated by the sums involved with the spatial derivative calculations. Equations 4.17 and 4.19 are used giving

$$\begin{aligned}
\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_i} &= \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{bdy}} \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_i} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSx}_{bdy}} \frac{\partial \mathbf{DSx}_{bdy}}{\partial \mathbf{DSx}_{ext}} \frac{\partial \mathbf{DSx}_{ext}}{\partial \mathbf{DSx}_i} \frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_i} \\
&\quad + \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSy}_{bdy}} \frac{\partial \mathbf{DSy}_{bdy}}{\partial \mathbf{DSy}_{ext}} \frac{\partial \mathbf{DSy}_{ext}}{\partial \mathbf{DSy}_i} \frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_i}
\end{aligned} \tag{4.24}$$

As with the conserved variables, the spatial derivatives are also stored at boundary

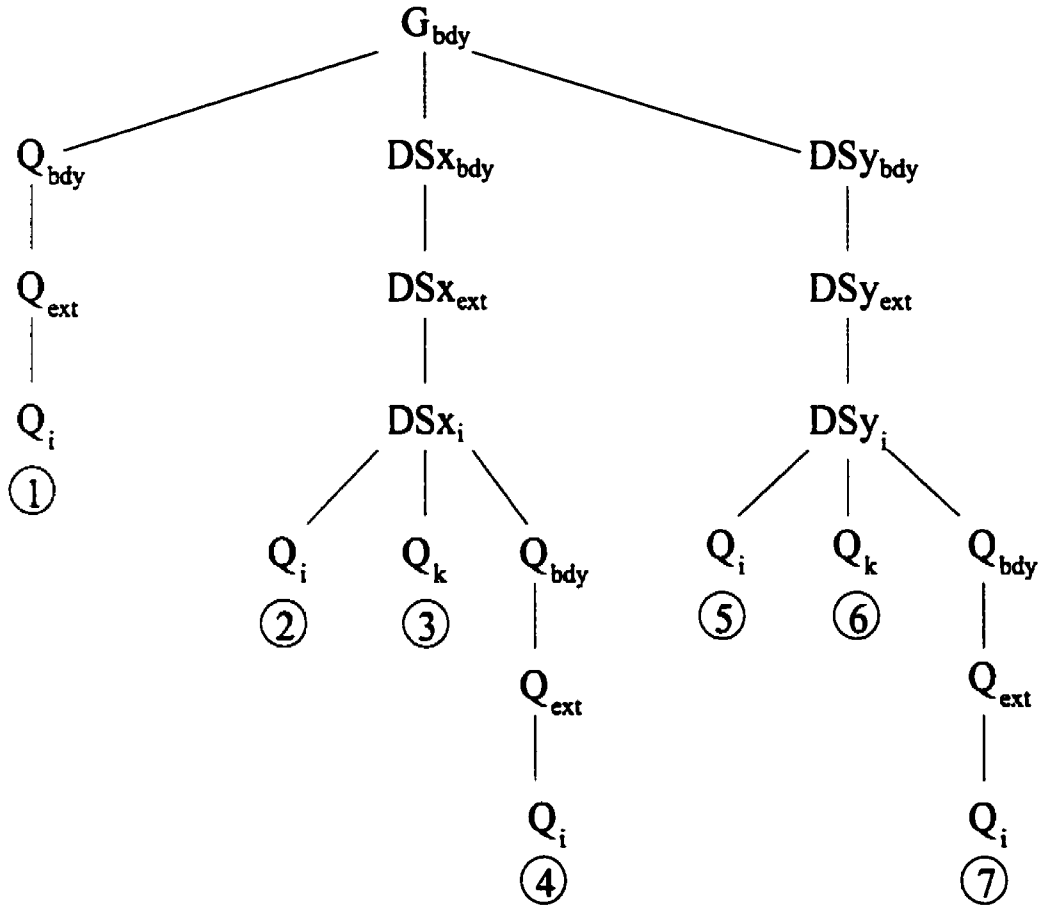

Figure 4.2: Contributions to G_{bdy}

	$\frac{\partial}{\partial \mathbf{Q}_i}$ (i-interior)	$\frac{\partial}{\partial \mathbf{Q}_i}$ (i-boundary)
$\overline{\mathbf{F}}_i$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i}$
$\overline{\mathbf{G}}_i$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i}$ - Eq: 4.20
$\overline{\mathbf{F}}_k$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}}_k$	$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_i}$ - Eq: 4.21	$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_i}$ - Eq: 4.21
$\overline{\mathbf{A}}_{ik}$	$\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_i}$	$\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_i}$
$\overline{\mathbf{F}}_{bdy}$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_i}$ - Eq: 4.22
$\overline{\mathbf{G}}_{bdy}$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_i}$ - Eq: 4.24
$\mathbf{S}_i$	$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_i}$ - Eq: 4.26	$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_i}$ - Eq: 4.26

Table 4.1: Contribution to Diagonal Blocks

faces. $\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSx}_{bdy}}$ and $\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSy}_{bdy}}$ are linearizations of the viscous flux at a boundary face with respect to the spatial derivatives there. These spatial derivatives are based on extrapolated ones where $\frac{\partial \mathbf{DSx}_{bdy}}{\partial \mathbf{DSx}_{ext}}$ and $\frac{\partial \mathbf{DSy}_{bdy}}{\partial \mathbf{DSy}_{ext}}$ are linearizations of any boundary conditions applied. Finally, $\frac{\partial \mathbf{DSx}_{ext}}{\partial \mathbf{DSx}_i}$ and $\frac{\partial \mathbf{DSy}_{ext}}{\partial \mathbf{DSy}_i}$ are both identity due to the zeroth-order extrapolation used.

The final term included in a diagonal block of the approximate Jacobian is $\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_i}$. This is a linearization of the source terms at cell i. These are simpler to compute than the flux Jacobians since there are no sums involved. If

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ S_{\text{diff}} + S_{\text{prod}} + S_{\text{dest}} + S_{\text{trip}} \end{bmatrix} = \mathbf{S}_{\text{diff}} + \mathbf{S}_{\text{prod}} + \mathbf{S}_{\text{dest}} + \mathbf{S}_{\text{trip}} \tag{4.25}$$

then

$$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_i} = \frac{\partial \mathbf{S}_{\text{diff}_i}}{\partial \mathbf{Q}_i} + \frac{\partial \mathbf{S}_{\text{prod}_i}}{\partial \mathbf{Q}_i} + \frac{\partial \mathbf{S}_{\text{dest}_i}}{\partial \mathbf{Q}_i} + \frac{\partial \mathbf{S}_{\text{trip}_i}}{\partial \mathbf{Q}_i} \tag{4.26}$$

The various terms contributing to the diagonal (i,i) block are summarized in Table 4.1.

4.5.2 Off-Diagonal Blocks

The same procedure is followed when linearizing Eq. 3.36 with respect to a particular neighbour cell, k^*, to obtain

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{Q}_{k^*}} = \sum_{k=1}^{N_i} \left(\frac{1}{2}\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_{k^*}} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_{k^*}} + \frac{1}{2}\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_{k^*}} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}} \right) \cdot \mathbf{n}_{ik} + \frac{1}{2}\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k^*}} + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{k^*}} - \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{k^*}} \right) \cdot \mathbf{n}_{bdy} - \Omega_i \frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k^*}} \tag{4.27}$$

which is the off-diagonal (i,k^*) block. The eight terms in Eq. 4.27 will also be discussed below.

The first, $\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_{k^*}}$, is identically zero. As with $\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_i}$, the inviscid flux at one cell is not related to the state at another.

$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_{k^*}}$ is added only if i is a boundary cell. For interior cells, there is no $\mathbf{G}_i$ contribution to $\mathbf{R}_i$. This term may be found by tracing out branches 3 and 6 of figure 4.1 and is written as

$$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_{k^*}} = \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSx}_i}\frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{k^*}} + \frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{DSy}_i}\frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{k^*}} \tag{4.28}$$

Here there are no sums associated with the spatial derivative calculations since only the i-k^* face yields a nonzero term.

$\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_{k^*}}$ contributes to $\frac{\partial \mathbf{R}_i}{\partial \mathbf{Q}_{k^*}}$ only if $k = k^*$. In this case the Jacobian block has the same functional form as $\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_i}$, an is simply calculated using values at cell k^*.

The $\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}}$ contribution is added when $k = k^*$ as well. One must be mindful of the case when cells i and k^* share a common neighbour. In that case there is a $\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}}$ term when k points to that common neighbour. This will be considered a next-to-nearest neighbour contribution and will be discussed in subsection 4.5.3.

When $k = k^*$, $\overline{\mathbf{G}}_{k^*}$ depends on the conserved variables, $\mathbf{Q}_{k^*}$, for all cells. This corresponds to branch 1 of figure 4.1 with indices i and $k = k^*$ switched. That is

$$\frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{Q}_{k^*}} = \frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{Q}_{k^*}} \tag{4.29}$$

This is valid for the interior.

Moreover, if k^* is a boundary cell, there is a contribution through the spatial derivatives at k^* as well. Here the functional form of $\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}}$ is identical to that of $\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_i}$. This

is given as

$$\begin{aligned}\frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{Q}_{k^*}} &= \left.\frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{Q}_{k^*}}\right|_{\mathbf{DSx},\mathbf{DSy}\ \text{const.}} \\ &+ \frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{DSx}_{k^*}} \frac{\partial \mathbf{DSx}_{k^*}}{\partial \mathbf{Q}_{k^*}} \\ &+ \frac{\partial \overline{\mathbf{G}}_{k^*}}{\partial \mathbf{DSy}_{k^*}} \frac{\partial \mathbf{DSy}_{k^*}}{\partial \mathbf{Q}_{k^*}}\end{aligned} \tag{4.30}$$

using the full derivatives given by

$$\frac{\partial \mathbf{DSx}_{k^*}}{\partial \mathbf{Q}_{k^*}} = \sum_{k=1}^{N_i} \left.\frac{\partial \mathbf{DSx}_{k^*}}{\partial \mathbf{Q}_{k^*}}\right|_{\mathbf{Q}_{bdy}\ \text{const.}} \mathbf{n}_x + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \mathbf{DSx}_{k^*}}{\partial \mathbf{Q}_{bdy}} \mathbf{n}_x\right) \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_{k^*}} \tag{4.31}$$

and

$$\frac{\partial \mathbf{DSy}_{k^*}}{\partial \mathbf{Q}_{k^*}} = \sum_{k=1}^{N_i} \left.\frac{\partial \mathbf{DSy}_{k^*}}{\partial \mathbf{Q}_{k^*}}\right|_{\mathbf{Q}_{bdy}\ \text{const.}} \mathbf{n}_y + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \mathbf{DSy}_{k^*}}{\partial \mathbf{Q}_{bdy}} \mathbf{n}_y\right) \frac{\partial \mathbf{Q}_{bdy}}{\partial \mathbf{Q}_{ext}} \frac{\partial \mathbf{Q}_{ext}}{\partial \mathbf{Q}_{k^*}} \tag{4.32}$$

The artificial dissipation contribution, $\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k^*}}$ is discussed by Wehner [38].

For the inviscid flux at the boundary face, $\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{k^*}} = 0$, since $\overline{\mathbf{F}}_{bdy}$ is only related to $\mathbf{Q}_i$ for zeroth-order extrapolation.

The viscous flux term at a boundary face, however, contributes to the off-diagonal blocks. This may be seen in branches 3 and 6 of figure 4.2, with $k = k^*$ and is written as:

$$\begin{aligned}\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{k^*}} &= \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSx}_{bdy}} \frac{\partial \mathbf{DSx}_{bdy}}{\partial \mathbf{DSx}_{ext}} \frac{\partial \mathbf{DSx}_{ext}}{\partial \mathbf{DSx}_i} \frac{\partial \mathbf{DSx}_i}{\partial \mathbf{Q}_{k^*}} \\ &+ \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{DSy}_{bdy}} \frac{\partial \mathbf{DSy}_{bdy}}{\partial \mathbf{DSy}_{ext}} \frac{\partial \mathbf{DSy}_{ext}}{\partial \mathbf{DSy}_i} \frac{\partial \mathbf{DSy}_i}{\partial \mathbf{Q}_{k^*}}\end{aligned} \tag{4.33}$$

Finally, for source terms which contain spatial derivatives, there is an off-diagonal contribution as well. This is written as

$$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k^*}} = \frac{\partial \mathbf{S}_{\mathrm{diff}_i}}{\partial \mathbf{Q}_{k^*}} + \frac{\partial \mathbf{S}_{\mathrm{prod}_i}}{\partial \mathbf{Q}_{k^*}} + \frac{\partial \mathbf{S}_{\mathrm{dest}_i}}{\partial \mathbf{Q}_{k^*}} + \frac{\partial \mathbf{S}_{\mathrm{trip}_i}}{\partial \mathbf{Q}_{k^*}} \tag{4.34}$$

The various terms contributing to the off-diagonal (i,k^*) block are summarized in table 4.2.

4.5.3 Next-To-Nearest Neighbour Blocks

The nonlinear residual at cell i, $\mathbf{R}_i$, depends on the viscous fluxes at cell k, $\overline{\mathbf{G}}_k$, where k is a neighbour of i. These viscous flux terms contain the spatial derivatives at cell

	$\frac{\partial}{\partial \mathbf{Q}_{k^*}}$ (i-interior)	$\frac{\partial}{\partial \mathbf{Q}_{k^*}}$ (i-boundary)
$\overline{\mathbf{F}}_i$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}}_i$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.28
$\overline{\mathbf{F}}_k$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{F}}_{k^*}}{\partial \mathbf{Q}_{k^*}}$
$\overline{\mathbf{G}}_k$	$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.29 (k^*-int), -Eq: 4.30 (k^*-bdy)	$\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.29 (k^*-int), -Eq: 4.30 (k^*-bdy)
$\overline{\mathbf{A}}_{ik}$	$\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k^*}}$	$\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k^*}}$
$\overline{\mathbf{F}}_{bdy}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}}_{bdy}$	$\mathbf{0}$	$\frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.33
$\mathbf{S}_i$	$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.34	$\frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k^*}}$ - Eq: 4.34

Table 4.2: Contribution to Off-Diagonal Blocks

k which depend on cell $k_{\rm nn}$, where $k_{\rm nn}$ is a neighbour of cell k. Thus one can see that R_i is not completely defined by Q_i and Q_k, but also depends on $Q_{k_{\rm nn}}$.

As mentioned in section 4.2, when using matrix-free GMRES, the accuracy of the Jacobian upon which the preconditioner is based only affects the convergence of the solution, not the solution itself. The importance of these next-to-nearest neighbour effects on convergence is the topic of section 5.6. Unfortunately, it will be shown that these effects have a great influence on the robustness of the solver.

Again, linearization of $\mathbf{R}_i$ with respect to $\mathbf{Q}_{k_{\rm nn}}$ starts with

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} = \sum_{k=1}^{N_i} \left(\frac{1}{2}\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} + \frac{1}{2}\frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_{k_{\rm nn}}} - \frac{1}{2}\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k_{\rm nn}}} \right) \cdot \mathbf{n}_{ik} + \frac{1}{2}\frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} + \sum_{bdy=1}^{N_{bdy}} \left(\frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{k_{\rm nn}}} - \frac{\partial \overline{\mathbf{G}}_{bdy}}{\partial \mathbf{Q}_{k_{\rm nn}}} \right) \cdot \mathbf{n}_{bdy} - \Omega_i \frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} \tag{4.35}$$

Fortunately, most terms in Eq. 4.35 are identically zero. The only possible contribution is through the viscous fluxes, since the artificial dissipation is treated using Eq. 4.11. Thus

$$\frac{\partial \overline{\mathbf{F}}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} = \frac{\partial \overline{\mathbf{F}}_k}{\partial \mathbf{Q}_{k_{\rm nn}}} = \frac{\partial \mathbf{AD}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} = \frac{\partial \overline{\mathbf{F}}_{bdy}}{\partial \mathbf{Q}_{k_{\rm nn}}} = \frac{\partial \mathbf{S}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} = \mathbf{0} \tag{4.36}$$

Of the viscous fluxes, only $\overline{\mathbf{G}}_k$ displays a next-to-nearest neighbour dependence. Both $\overline{\mathbf{G}}_i$ and $\overline{\mathbf{G}}_{bdy}$ depend only on cell i, neighbour k and boundary face bdy. Thus,

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{Q}_{k_{\rm nn}}} = \sum_{k=1}^{N_i} \left(-\frac{1}{2}\frac{\partial \overline{\mathbf{G}}_k}{\partial \mathbf{Q}_{k_{\rm nn}}} \right) \cdot \mathbf{n}_{ik} \tag{4.37}$$

	$\frac{\partial}{\partial \mathbf{Q}_{k_{nn}}}$ (i-interior)	$\frac{\partial}{\partial \mathbf{Q}_{k_{nn}}}$ (i-boundary)
$\overline{\mathbf{F}_i}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}_i}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{F}_k}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}_k}$	$\frac{\partial \overline{\mathbf{G}_k}}{\partial \mathbf{Q}_{k_{nn}}}$	$\frac{\partial \overline{\mathbf{G}_k}}{\partial \mathbf{Q}_{k_{nn}}}$
$\overline{\mathbf{A}_{ik}}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{F}_{bdy}}$	$\mathbf{0}$	$\mathbf{0}$
$\overline{\mathbf{G}_{bdy}}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{S}_i$	$\mathbf{0}$	$\mathbf{0}$

Table 4.3: Contribution to Off-Diagonal, Next-to-Nearest Neighbour Blocks

which is the (i,k_{nn}) block.

The contribution to the off-diagonal, next-to-nearest neighbour blocks is summarized in Table 4.3.

4.6 Start-Up

One problem with Newton's method for solving nonlinear equations such as $\mathbf{R}(\mathbf{Q}) = \mathbf{0}$, is that the solution may diverge if the initial guess, $\mathbf{Q}^0$ is far from the solution. For inviscid cases, Wehner noted that Newton's method will converge starting from free-stream values for subsonic flows, but diverges for transonic cases [38]. Even for subsonic flows, a poor choice of parameters was found to cause divergence of the method.

As such the Newton updates are damped in order to widen the domain of convergence. Implicit Euler time stepping is used with local time linearization giving

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} + \frac{\mathbf{I}}{h}\right) \Delta \mathbf{Q} = -\mathbf{R}(\mathbf{Q}) \tag{4.38}$$

where h is the temporal increment. Generally, as the iterate, $\mathbf{Q}^n$, approaches the converged solution, the time step may be increased. A variable time step is used whereby

$$h = h_0 \frac{||\mathbf{R}||_0}{||\mathbf{R}||_n} \tag{4.39}$$

where $||\mathbf{R}||_0$ is the initial L_2 norm of the nonlinear residual and $||\mathbf{R}||_n$ is the L_2 norm at iteration n. This ramping function has the property that as the residual approaches zero, the time step approaches infinity. Only a quick examination of the scheme, Eq. 4.38,

is required for one to realize that in the limit as $h \to \infty$ Newton's method, Eq. 4.1, is recovered.

In this solver the L_2 norm of density is used to measure convergence, although other measures can be used. For turbulent cases, the scalings based on ρ and $\tilde{\nu}$ are compared and the less aggressive is used. This accounts for any differences in convergence rates of the flow equations and the turbulence model.

Stricter start-up requirements are needed for turbulent cases. The exact solution of the turbulence transport equation cannot become negative. It can be shown that if $\tilde{\nu} = 0$ at some location and the surrounding values are non-negative, then $\frac{\partial \tilde{\nu}}{\partial t} \geq 0$ [44]. To mimic this analytical behavior, the recommendations of Spalart and Allmaras [44] are followed as outlined below.

The production and destruction source terms are modified for start-up to encourage the positivity of $\tilde{\nu}$. These may be written as

$$S_{\text{prod}} = P(\tilde{\nu})\tilde{\nu} \tag{4.40}$$

and

$$S_{\text{dest}} = D(\tilde{\nu})\tilde{\nu} \tag{4.41}$$

Thus the linearization is given by

$$[P(\tilde{\nu})\tilde{\nu}]' = P(\tilde{\nu}) + P'(\tilde{\nu})\tilde{\nu} \tag{4.42}$$

and

$$[D(\tilde{\nu})\tilde{\nu}]' = D(\tilde{\nu}) + D'(\tilde{\nu})\tilde{\nu} \tag{4.43}$$

Instead of including the true linearizations in the left-hand side, $\overline{P}$ and $\overline{D}$ are chosen. These are close to the true Jacobians given by equations 4.42 and 4.43, yet retain strict adherence to the positivity constraint. The sum is given by

$$\overline{P} + \overline{D} = \text{pos}(P + D) + \text{pos}(P' + D')\tilde{\nu} \tag{4.44}$$

where the $\text{pos}(\phi)$ function clips values of $\phi < 0$ to zero.

In order to enforce this constraint, start-up does not allow for a matrix-free solver. GMRES is used with the approximate Jacobian as the matrix. The flow equations are treated as before, but a small, different, constant time step is used for the turbulence

model. Additionally, the off-diagonal terms are dropped for this equation, thus decoupling the turbulent transport.

This was found to be very slow and convergence tended to stall after approximately a two order of magnitude drop in the Spalart-Allmaras residual. The remainder of this work will start with a partly converged solution, allowing matrix-free GMRES to be utilized. The possibility exists of using explicit multigrid for start-up and switching to matrix-free GMRES after a user-prescribed tolerance has been obtained. Start-up remains a topic for further research.

4.6.1 Local Time Stepping

Since steady-state solutions are desired, the time accuracy of the iterates, $\mathbf{Q}^n$, may be sacrificed. This allows the solution at each grid point to be advanced at the maximum possible time step.

The local. viscous time step is given by taken from Mavriplis et al [57] as recounted by Greiner [45]

$$h = CFL\left(\frac{h_v h_c}{h_v + h_c}\right) \tag{4.45}$$

CFL is the Courant-Friedrichs-Lewy number while h_v and h_c are given as:

$$h_v = \frac{2}{\lambda_v} \tag{4.46}$$

and

$$h_c = \frac{2}{\lambda_c} \tag{4.47}$$

where λ_v and λ_c are approximations to the maximum diffusive and convective eigenvalues, given by

$$\lambda_v = \sum_{k=1}^{N_i} \frac{\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)|\mathbf{n}_{ik}|^2 \gamma}{2\rho Re_a \Omega_i} \tag{4.48}$$

and

$$\lambda_c = \sum_{k=1}^{N_i} |\mathbf{v}| \cdot \mathbf{n}_{ik} + a\,|\mathbf{n}_{ik}| \tag{4.49}$$

respectively. The values used in the calculations are averages from cells i and k. Modification for boundary cells is straight-forward.

4.7 Implementation

Hurricane is coded in C++ with much of the Newton-Krylov implementation coming from the Portable, Extensible, Toolkit for Scientific Computation (PETSc) [58]. PETSc provides efficient data structures for linear solvers, nonlinear solvers and optimization routines. In particular Hurricane uses the vector and block, sparse matrix data structures as well as GMRES linear solver, ILU preconditioning and RCM reordering routines.

Chapter 5

Results

In the preceding chapter an algorithm has been presented to solve the nonlinear system $\mathbf{R}(\mathbf{Q}) = \mathbf{0}$. One wishing to solve such a system with the Newton-Krylov method is left with many choices of parameters. An arbitrary selection of these parameters may result in poor convergence properties, or often divergence. A well-tuned set, however, produces an efficient and robust solver.

5.1 Test Cases

Several laminar test cases have been used for parametric optimisation and verification of results as outlined in table 5.1 below.

Case Number	Airfoil	Grid Type	Number of Cells	M_∞	α [°]	Re
1	NACA 0012	Structured	12164	0.8	5.0	500
2	NACA 0012	Unstructured	11441	0.8	5.0	500
3	RAE 2822	Unstructured	9275	0.75	4.0	500
4	NACA 0012	Structured	12164	0.3	6.0	1000
5	NACA 0012	Structured	12164	0.5	1.0	250

Table 5.1: Test Cases

Cases 1 and 2 are primary test cases for which full parametric studies are undertaken. These show that the choice of parameters is reasonably independent of grid structure. The remaining test cases show that solutions may be efficiently obtained for a variety of shapes and flow conditions using the same parameter set.

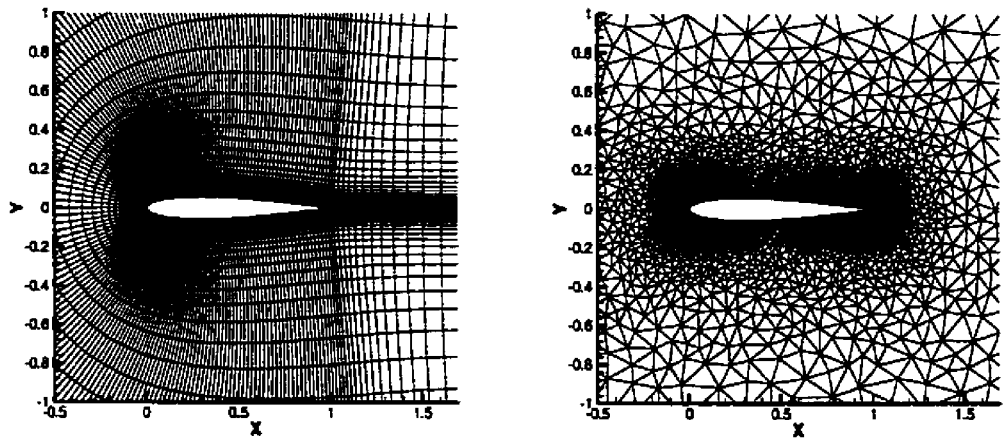

Figure 5.1: Structured and Unstructured Grids about a NACA 0012

The first two grids are shown in figure 5.1. The airfoil for each is a symmetric NACA 0012 with off-the-wall spacings of 5.0 x 10^{-4} and 1.0 x 10^{-4} respectively. Figure 5.2 is an unstructured grid about an RAE 2822 with an off-the-wall spacing of 1.0 x 10^{4}. For the unstructured grids a small structured region is used directly around the airfoils while the remainder of the domain consists of an unstructured triangular grid.

5.2 Convergence Measures

A base set of parameters for the solver may be found in section 5.10. With the exception of the dissipation constants ($\kappa^{(2)}$ and $\kappa^{(4)}$), the remaining parameters affect convergence and robustness of the solver, not the accuracy of the converged solution. Sections 5.3 through 5.9 show the effects of these parameters on convergence.

All convergence histories plot the L_2 norm of the continuity residual versus CPU time. In order to allow for comparisons between various processors, grid sizes, compilers and coding styles, the time is normalised by that required for one right-hand side residual evaluation. Although measuring convergence in RHS evaluations is by no means perfect, it nonetheless produces more repeatable results.

All tests were run on a Compaq Alpha ES40 server with 667 MHz Alpha 21264A processors. As a precaution, all results in the following sections were obtained without other processes running on the system.

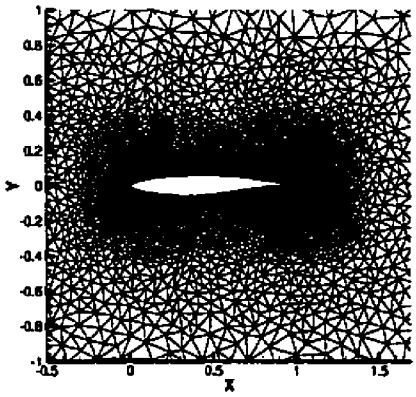

Figure 5.2: Unstructured Grid about an RAE 2822

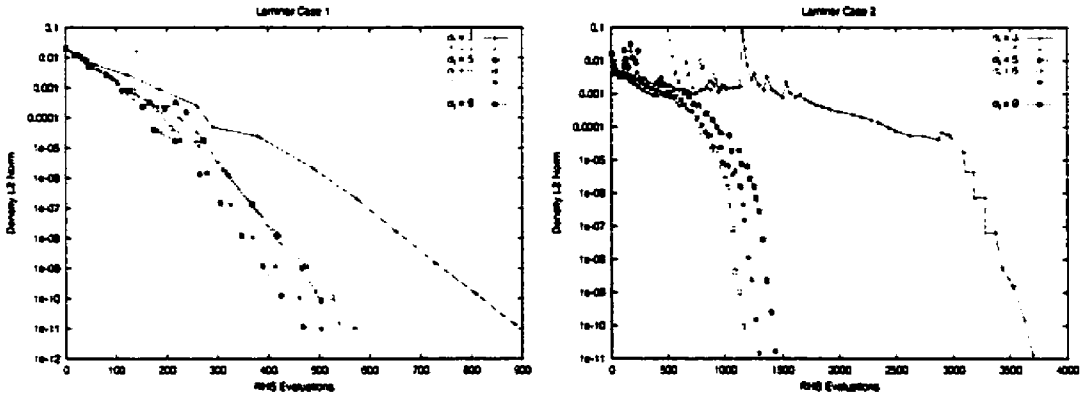

Figure 5.3: Effects of σ_l on Convergence for Cases 1 and 2

5.3 σ_l Effects

As mentioned in section 4.3.2, the off-diagonal dominance of the preconditioning Jacobian is reduced by using a modified second-difference dissipation stencil given by

$$\varepsilon_l^{(2)} = \varepsilon_r^{(2)} + \sigma_l \varepsilon_r^{(4)}$$

where σ_l is a free parameter. The effects of σ_l on convergence are shown in figure 5.3.

For case 1, $\sigma_l = 9$ produces the best convergence while $\sigma_l = 7$ also performs well. Poor results, or possibly divergence is obtained for $\sigma_l \leq 3$ or $\sigma_l \geq 10$. For case 2, however, $\sigma_l = 9$ results in a negative pressure calculation. Here, the method appears to be

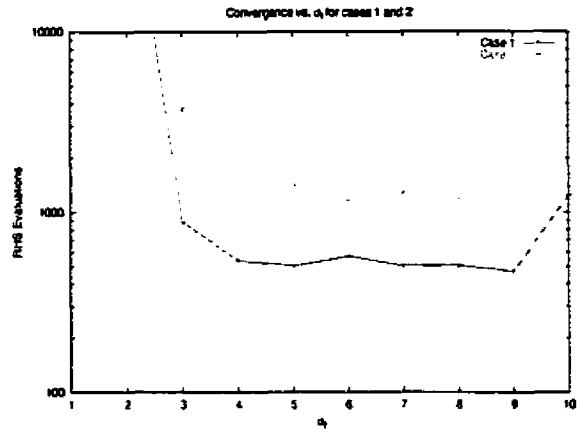

Figure 5.4: RHS Evaluations vs σ_l for Cases 1 and 2

more sensitive to the value of σ_l chosen. While $\sigma_l = 6$ or $\sigma_l = 8$ perform best, $\sigma_l = 7$ is acceptable as well.

Figure 5.4 plots the normalised CPU time for convergence to 10^{-10} vs σ_l for cases 1 and 2. From this graph, one may see that a σ_l value between 5 and 8 is best, where $\sigma_l = 7$ has been chosen as optimal.

These results compare well with previous studies. Pueyo [34] used $\sigma_l = 5$ while Wehner [38] recommends a value of 4 or 5 for σ_l. The higher value of σ_l required for these laminar cases can be attributed to the much lower values of $\kappa^{(2)}$ and $\kappa^{(4)}$.

5.4 Linear System Solution Precision

One attractive feature of using an iterative solver such as GMRES for the linear systems is that one may control how accurately those systems are solved. This ratio of inner residual reduction will be referred to as r_{tol}. As mentioned in section 4.2, oversolving of the linear system is costly and is to be avoided. The effects of r_{tol} on convergence are shown in figure 5.5.

From these plots it may be noted that obtaining accurate solutions to the linear problem hinders overall convergence. A value of $r_{tol} = 0.1$ is optimal for both cases, agreeing with results obtained by Pueyo [34] and by Wehner [38]. Figure 5.6 plots the normalised CPU time for convergence to 10^{-10} vs r_{tol} for case 1.

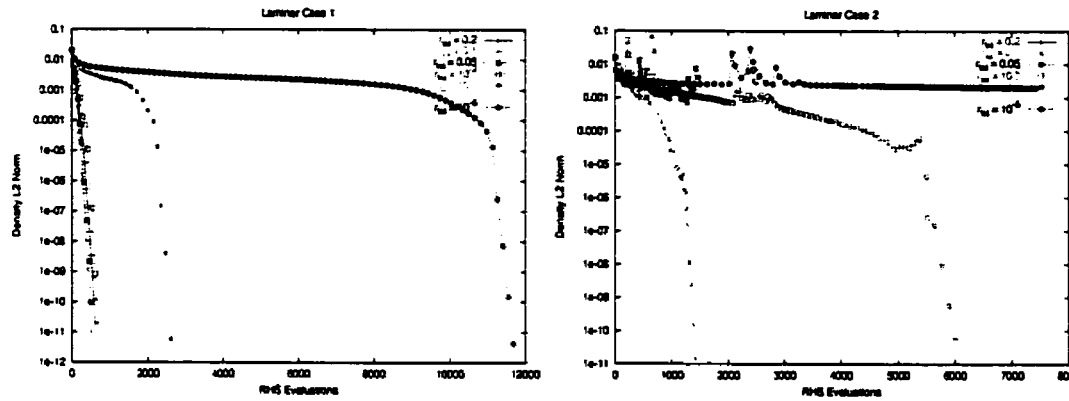

Figure 5.5: Effects of τ_{tol} on Convergence for Cases 1 and 2

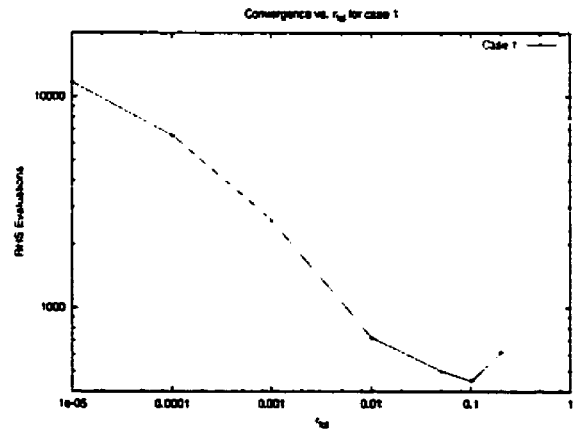

Figure 5.6: RHS Evaluations vs τ_{tol} for Case 1

It can also be seen that case 2 is very sensitive to the parameters used, as will be further discussed in section 5.10. Here, only values of $\tau_{tol} = 0.1$ and $\tau_{tol} = 0.01$ result in converged solutions with the former superior.

5.5 Preconditioner Fill

When forming the preconditioner based on the approximate Jacobian, one is left with the choice of how much fill to allow. While the effectiveness of the preconditioner should increase with increased fill, the time required to form the preconditioner and the storage needed increases as well.

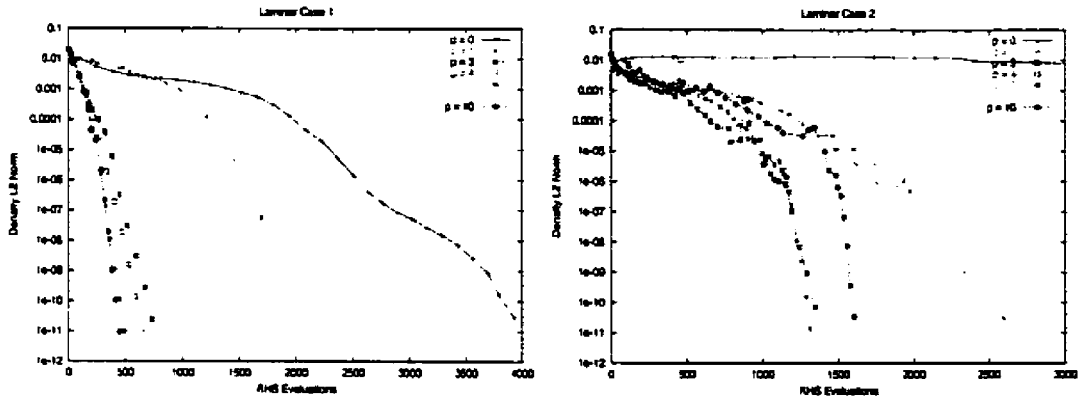

Figure 5.7: Effects of fill, p, on Convergence for Cases 1 and 2

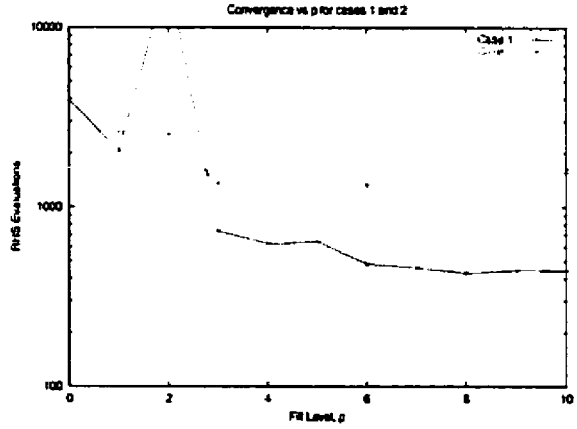

Figure 5.8: RHS Evaluations vs p for Cases 1 and 2

Figure 5.7 shows the effects of p on convergence for cases 1 and 2. Clearly $p = 0$ or $p = 1$ are poor choices for both cases. For case 1 increased fill results in better convergence up to $p = 8$ although little is gained when $p > 6$. For $p > 8$ the time required to form the preconditioner begins to become prohibitive.

Again, case 2 is more sensitive to the parameters chosen the first case. Here we can clearly see the cost associated with a fill level which is too low or too high. Fill values of 0, 1 or 10 are poor while 6 or 8 are best.

Figure 5.8 plots the normalised convergence time vs fill level for cases 1 and 2. Rapid convergence occurs for 6 or 8 levels of fill. Also one may note that some values of p result in divergence, mostly for case 2. From the plots in Section 5.10, one can see that poor

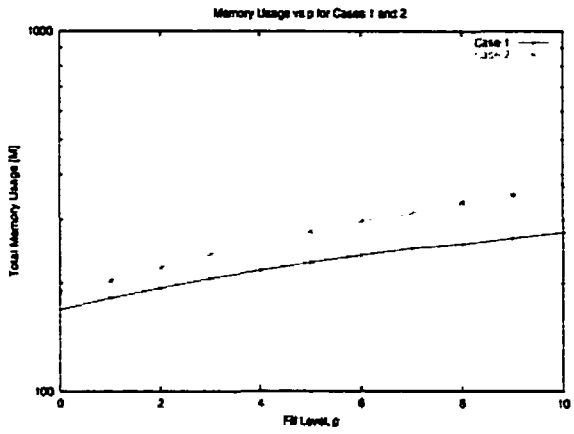

Figure 5.9: Total Memory Usage vs p for Cases 1 and 2

mesh quality is causing this sensitivity. A value of $p = 6$ was chosen as optimal, although any value about 3 is reasonable.

Finally, figure 5.9 plots the total memory usage vs fill level for cases 1 and 2. Generally as one increases the fill permitted, the memory usage increases as well. In choosing $p = 6$ as optimal, however, convergence time was the primary consideration. This is simply due to the fact that time is continuous while memory usage is quantized. The program will either fit in the system space available, or it will not. It is only in these latter cases that program size becomes a primary concern. Conversely, one can always benefit from improved convergence.

This choice of ILU(6) compares to ILU(2) used by both Pueyo [34] and Blanco [31]. Wehner, however, recommended values of p between 4 and 7 for various cases [38].

5.6 Preconditioner Accuracy

As mentioned in section 4.3.2, a complete linearization of the viscous fluxes results in a next-to-nearest neighbour stencil. Consideration has been given to using a nearest-neighbour only stencil for the viscous fluxes, thus significantly reducing the overall size of the approximate Jacobian matrix. This nearest-neighbour only stencil simply drops any entries formed by the presence of next-to-nearest neighbour cells. Figure 5.10 shows the difference in convergence rates when using a full stencil vs using nearest-neighbours only

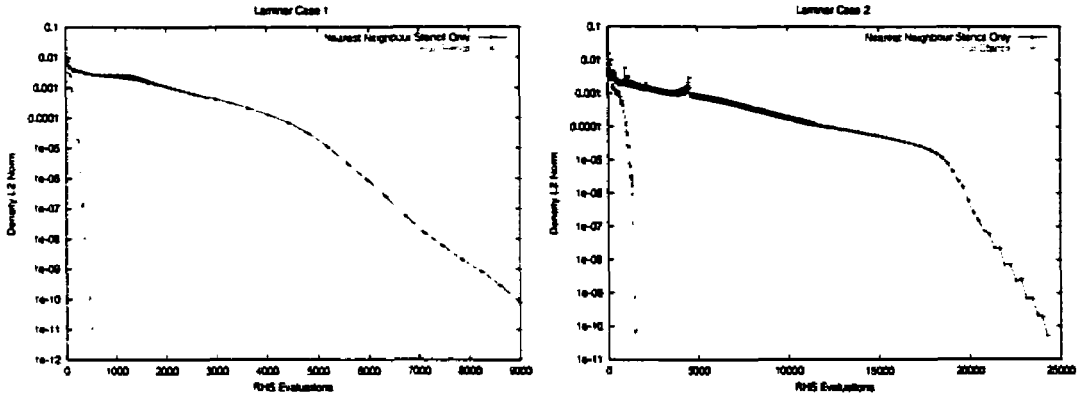

Figure 5.10: Effects of Preconditioner Accuracy on Convergence for Cases 1 and 2

for cases 1 and 2.

The full stencil results were obtained using the optimal parameters listed in section 5.10. Unfortunately these parameters resulted in divergence for the nearest-neighbour only runs. Results for these cases were obtained by allowing 100 GMRES iterations with no restart. Additionally, the initial CFL was reduced from 5000 to 1000 and 2000 for cases 1 and 2 respectively.

Case Number	Memory (NN Stencil) [M]	Memory (NTNN Stencil) [M]	Percentage Increase [%]
1	191	239	25
2	205	297	45

Table 5.2: Memory Requirements for Nearest and Next-to-Nearest Neighbour Stencils

Table 5.2 shows the storage costs associated with the nearest-neighbour and next-to-nearest neighbour stencils. The larger stencil only results in a 25% additional penalty for case 1 and a 45% increase in the second case. The extra nonzero entries produced by the next-to-nearest neighbour stencil are not the only cause of the different storage requirements; the memory required for the nearest-neighbour only cases is inflated due to the storage of the additional Krylov subspace search directions for the 100 GMRES iterations.

5.7 Startup

As mentioned in section 4.6, the iterates produced by Newton's method may diverge if the initial guess is far from the steady-state solution. To avoid this, local implicit

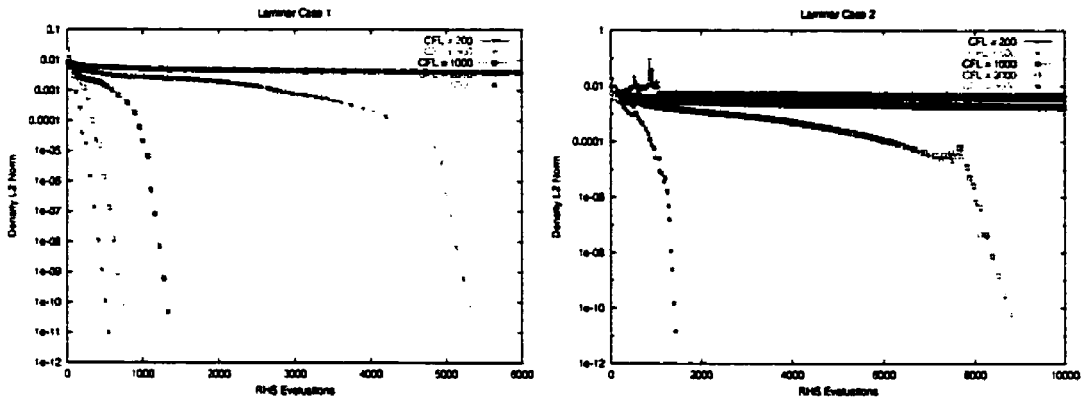

Figure 5.11: Effects of Initial CFL on Convergence for Cases 1 and 2

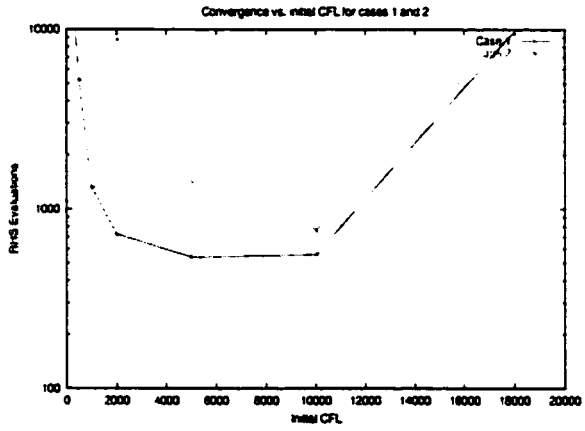

Figure 5.12: RHS Evaluations vs Initial CFL for Cases 1 and 2

Euler time stepping is used as shown in equation 4.38. As the norm of the residual approaches zero, the CFL is increased according to equation 4.39.

The remaining free parameter for startup is the initial CFL. Figure 5.11 shows the effects of initial CFL upon the overall convergence. For cases where the initial CFL is too low, convergence degrades rapidly. Conversely divergence may occur if the initial CFL is too high.

Figure 5.12 plots the convergence time vs. initial CFL for cases 1 and 2. For these two cases the optimal starting CFL is between 5000 and 10000. For case 2 an initial CFL of 10000 significantly outperforms an initial value of 5000. The aim here, however, is to obtain a set of parameters which result in convergence for a variety of test cases. Conservatively,

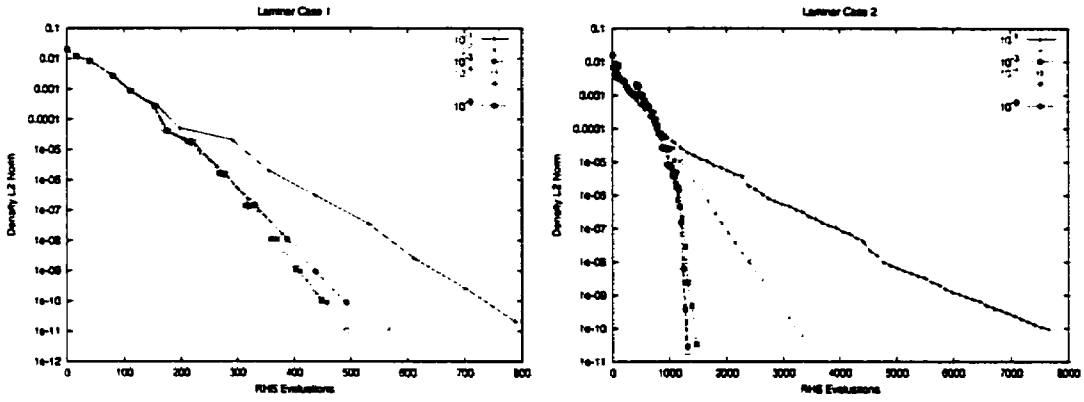

Figure 5.13: Effects of Freezing the Preconditioner on Convergence for Cases 1 and 2

5000 was chosen as optimal.

5.8 Freezing Preconditioner

Since the preconditioning matrix is based upon an approximation to the Jacobian, one may consider no longer updating the preconditioner once the solution has sufficiently converged. This will be termed "freezing" the preconditioner. Pueyo found that the preconditioner need only be formed once, following an approximately factored startup period [34]. Blanco found convergence was accelerated when freezing the preconditioner after a 5 order reduction in the nonlinear residual [31]. Wehner found optimal results when the preconditioner was frozen after a 2 order reduction of the nonlinear residual for cases where a time-step is required [38].

Figure 5.13 shows the effects of freezing the preconditioner on the overall convergence time for cases 1 and 2. It is clear the the preconditioner must not be frozen too soon. Performance degrades if the preconditioner is frozen after only one or two orders of magnitude reduction in the nonlinear residual.

Figure 5.14 plots the normalised CPU time for convergence to 10^{-10} vs orders reduction before freezing for cases 1 and 2. Again one sees that the preconditioner must not be frozen too soon. Freezing the preconditioner after a four order of magnitude reduction in the nonlinear residual is chosen as optimal although other values perform well also.

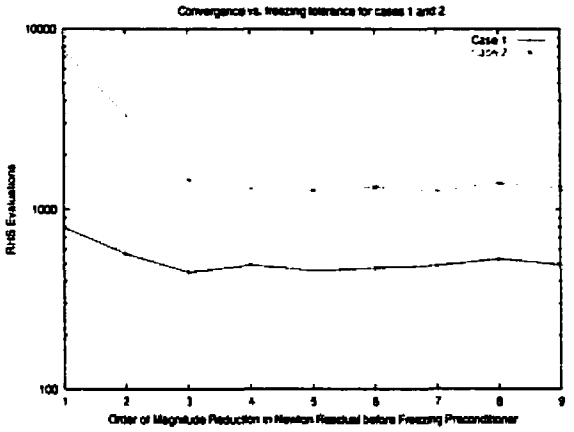

Figure 5.14: RHS Evaluations vs Freezing Tolerance for Cases 1 and 2

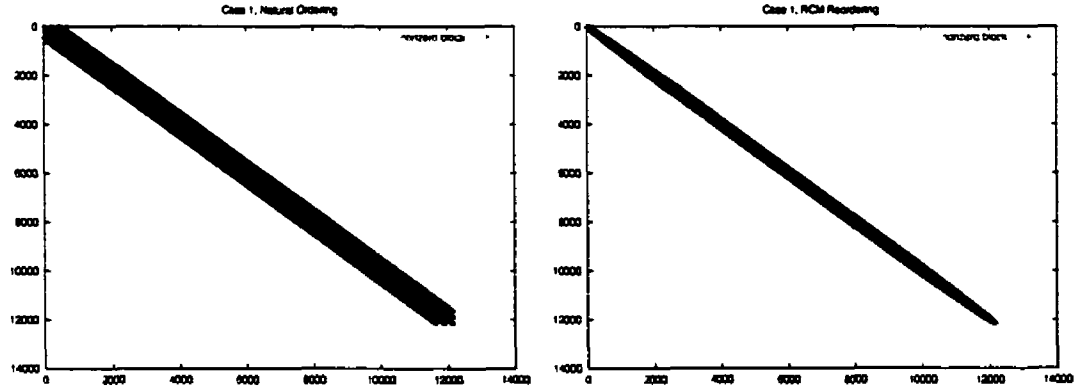

Figure 5.15: Approximate Jacobian Matrices with and without RCM Reordering for Case 1

5.9 RCM Reordering

As was stated in section 4.4, reverse Cuthill-McKee reordering will be applied to the linear systems. Figure 5.15 shows the approximate Jacobian matricies obtained before and after reordering is applied for case 1.

The effect of reordering on convergence is shown in figure 5.16. Reverse Cuthill-McKee reordering significantly improves convergence for these two cases. These results agree with the literature.

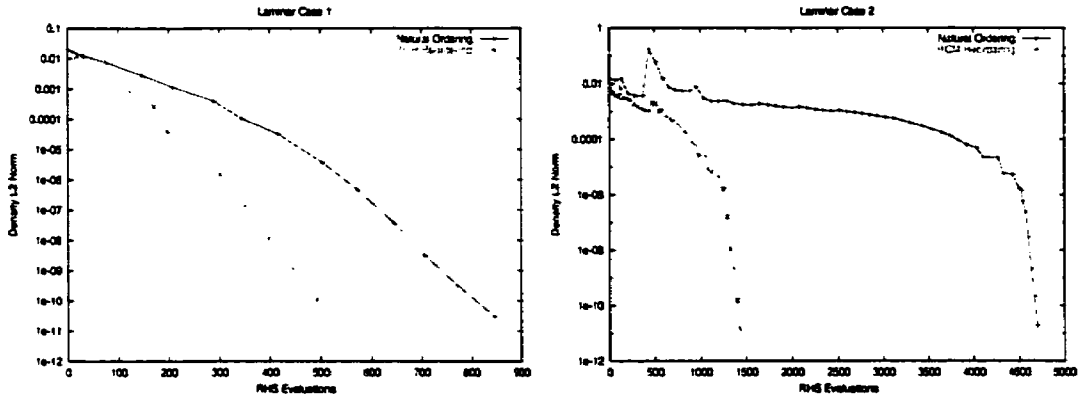

Figure 5.16: Effects of Reordering on Convergence for Cases 1 and 2

5.10 Optimal Parameters and Final Results

The base parameters used for the solver were determined from the results found in sections 5.3 through 5.9 and are listed below:

- $\sigma_l = 7$
- $r_{tol} = 0.1$
- ILU(6) preconditioner based on NTNN stencil
- Variable, local, implicit Euler time step with initial CFL of 5000
- Preconditioner frozen after a 4 order of magnitude drop in the nonlinear residual
- RCM reordering is applied
- Maximum of 40 GMRES iterations restarted after 20
- $\kappa_2 = 1.0$, $\kappa_4 = 0.01$

These parameters are to be used as a general guide for arbitrary cases and may not be optimal in all circumstances. For example, increasing the initial CFL to 10000 for case 2 will significantly reduce the time required for convergence. Nonetheless, the convergence behavior in general is desirable.

Convergence histories for the test cases may be found in figure 5.17. All converge in less than 1000 function evaluations with the exception of case 2. These results agree with those obtained by Pueyo [34] and Wehner [38].

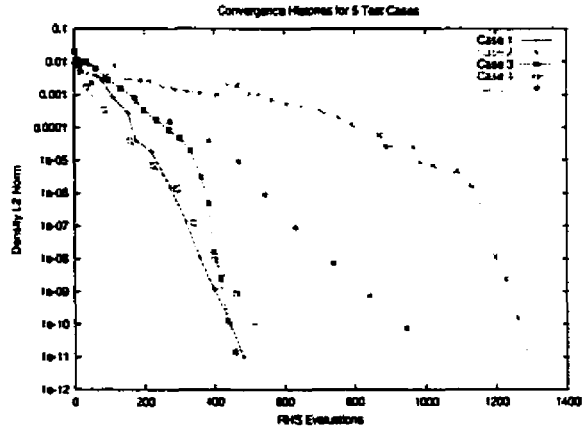

Figure 5.17: Convergence Histories for the 5 Test Cases

Case	Newton Its.	GMRES Its.	GMRES/Newton	CPU Time [RHS Evals.]
1	13	98	7.54	483
2	39	295	7.56	1287
3	17	116	6.82	458
4	9	125	13.89	514
5	11	212	19.27	943

Table 5.3: Inner and Outer Iterations for Test Cases

Table 5.3 shows the inner and outer iterations required for solving the various test cases. The results are in line with the structured results of Pueyo [34]. On average it takes less than 20 GMRES iterations per Newton step for all 5 test cases; cases 1 through 3 require less than 10. One may note that the systems stiffen as the Mach number is reduced. No low-Mach number preconditioning is employed.

The Mach number distribution throughout the domain, and the coefficient of pressure on the airfoil surface are plotted in figure 5.18 for case 1. The grid produces nice results as are evident in the smooth graphs.

Case 2 uses the same airfoil and flow parameters as case 1; the difference is the mesh used. Case 2 uses the hybrid, unstructured mesh as shown on the right of figure 5.1. The Mach number distribution and C_p plots may be found in figure 5.19. Finally it is clear why Case 2 is more sensitive to the parameters listed above. The difference in mesh structure is evident in the final solution for this case. There are two possible, related

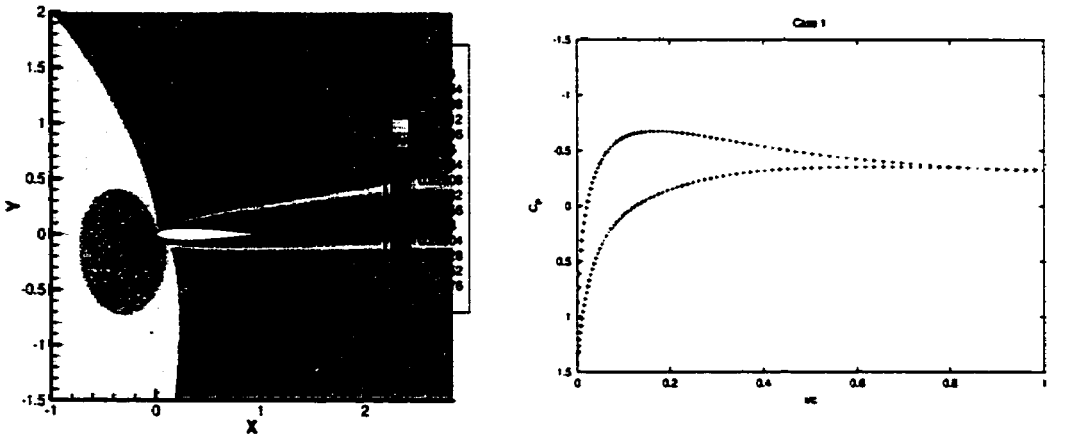

Figure 5.18: Mach Contour and C_p Plots for Case 1

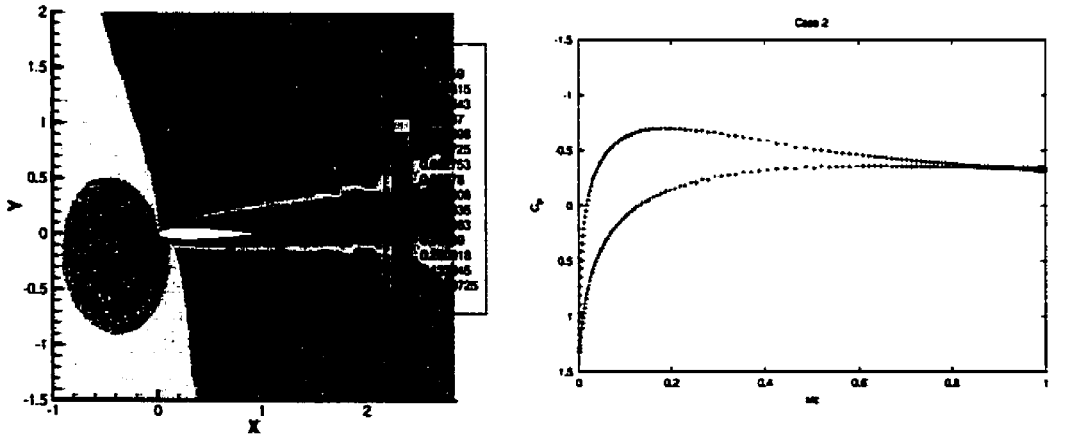

Figure 5.19: Mach Contour and C_p Plots for Case 2

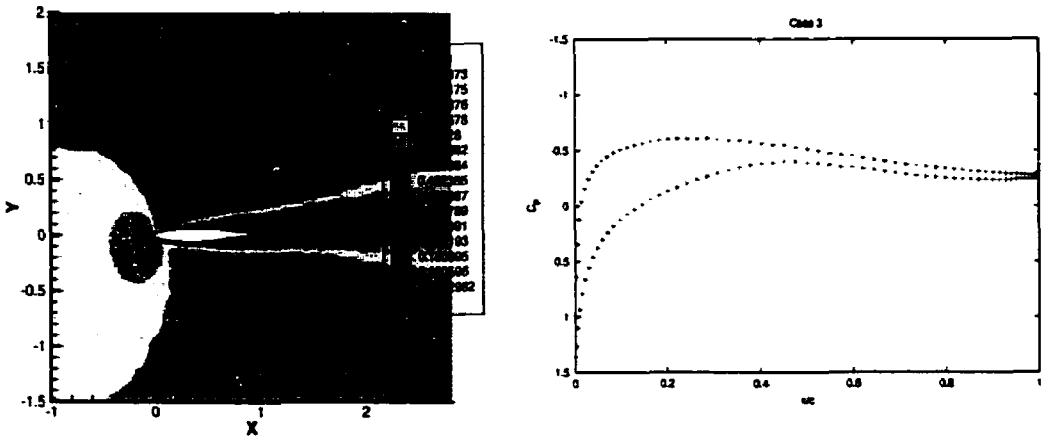

Figure 5.20: Mach Contour and C_p Plots for Case 3

causes for this phenomenon, namely the off-the-wall spacing and the mesh resolution. The problem may be solved with either a finer grid having the appropriate off-the-wall spacing, or by using an adaptive gridding strategy as discussed by Walsh [40].

Run	C_l	C_d
Case 1	0.229	0.230
Case 2	0.257	0.237
Comparison	0.239	0.215

Table 5.4: Comparison of Results for NACA0012, $M_\infty = 0.8$, $\alpha = 5^o$, $Re = 500$

Table 5.4 compares the lift and drag coefficients from cases 1 and 2 with to one obtained from the structured code Optima2D. One can see that the oscillations present behind the airfoil in figure 5.19 do not significantly affect the bulk coefficients.

Figure 5.20 shows that the same oscillations present in case 2 are present in case 3 as well. For this case κ_4 was raised to 0.1 in an attempt to damp out the oscillations in the solution.

Due to the oscillations present in the solutions on the hybrid grids, the remaining two test cases used the structured mesh. Figures 5.21 and 5.22 show the solution for cases 4 and 5 respectively. The high quality of the mesh used accounts for the smooth solutions.

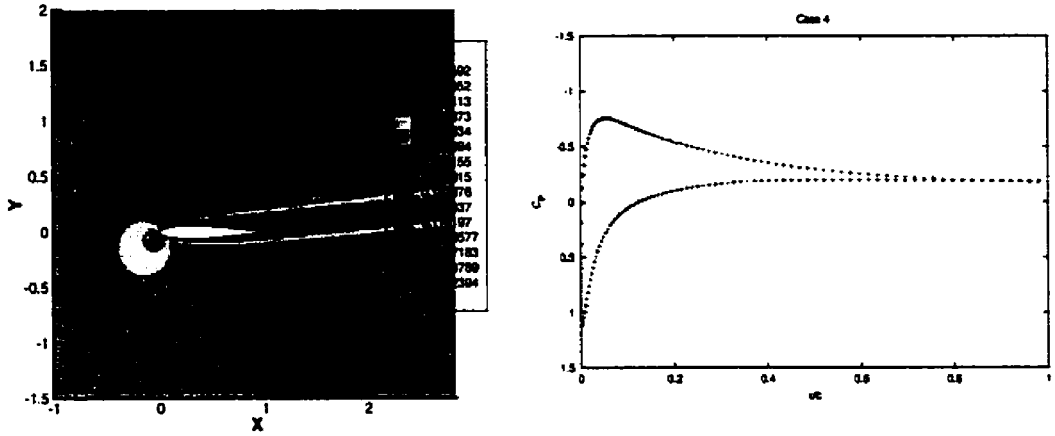

Figure 5.21: Mach Contour and C_p Plots for Case 4

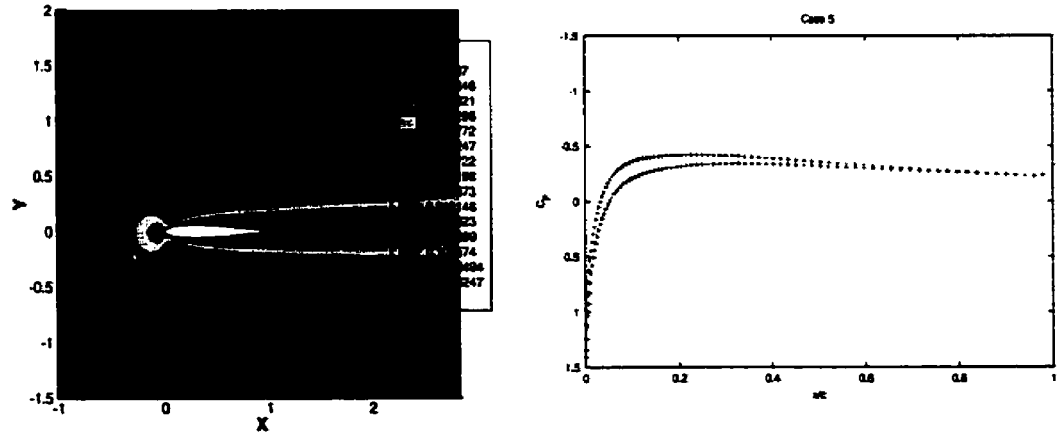

Figure 5.22: Mach Contour and C_p Plots for Case 5

Chapter 6

Concluding Remarks

6.1 Conclusions

The inviscid work of Wehner [38] has been extended to viscous flows. A Newton-Krylov solver for the Navier-Stokes equations on general unstructured grids has been developed and an optimal set of parameters determined considering both speed and storage requirements.

The solver uses a finite-volume formulation on arbitrary polygonal meshes. Second- and fourth-difference scalar artificial dissipation has been added for numerical stability. Newton's method has been used to solve the discrete, nonlinear, algebraic equations, while an ILU-preconditioned, matrix-free GMRES method solves the resulting linear systems. RCM reordering was used to reduce bandwidth and local, implicit-Euler time stepping promotes robustness during start-up.

It has been shown that an optimal choice of parameters results in a very efficient solver for a variety of laminar flow conditions; convergence is generally obtained in fewer than 1000 function evaluations. These tests successfully validate the functionality of the solver.

6.2 Recommendations

The laminar solver is ready for final extension to turbulent flows. The turbulent flux Jacobians have been verified using finite-difference comparisons and the turbulent start-up algorithm is functional as well. The solver may now be restarted from a multigrid run,

and appropriate parameters must be chosen. Once a solution is obtained on an adequate grid using an explicit solver, this Newton-Krylov algorithm may be verified. Start-up may then be accomplished via either the algorithm stated herein, or using multigrid.

Further tests ought to be run on unstructured meshes. Although a successful grid has been found for various test cases, the effects of off-the-wall spacing and grid resolution on convergence should be investigated.

Finally, storage required for the solver may be reduced as well. Although it has been shown that a nearest-neighbour only stencil results in poor convergence, the use of a distance-1 Jacobian should be studied. Additionally, other researchers have successfully reduced memory overhead by storing the preconditioner in single precision. The feasibility of implementing this using the PETSc libraries could be investigated. Any reduction in memory requirements obtained by further research would significantly impact a three-dimensional extension of this work.

References

[1] Anderson, John D., "Fundamentals of Aerodynamics, Second Edition," McGraw-Hill, Toronto, 1991.

[2] Thompson, J.F., Warsi, Z.U.A., Mastin, W.C., "Numerical Grid Generation," Elsevier Science Publishing, New York, 1985.

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

[4] Peaceman, D.W., and Rachford, H.H., "The Numerical Solution of Parabolic and Elliptic Differential Equations," SIAM J. vol. 3, no. 1, pp. 28-41, 1955.

[5] Douglas, J., and Gunn, J.E., "A General Formulation of Alternating Direction Methods," Numerische Mathematik, vol. 6, 1964.

[6] Dembo, R.S., Eisenstat, S.C., and Steihaug, T., "Inexact Newton methods," SIAM J. Numerical Analysis, 19, pp. 400-408, 1982.

[7] Gropp, W.D., Keyes, D.E., McInnes, L.C., and Tidriri, M.D., "Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel Implicit CFD," Int. J. High Performance Computing Applications 14, pp. 102-136, 2000.

[8] Vanka, S.P., "Block-Implicit Calculation of Steady Turbulent Recirculating Flows," International J. of Heat and Mass Transfer, 28, pp. 2093-2103, 1985.

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

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

[11] Whitfield, D.L., and Taylor, L.K., "Discretized Newton-Relaxation Solution of High Resolution Flux-Difference Split Schemes," In Proceedings of the AIAA Tenth Annual Computational Fluid Dynamics Conference, pp. 134-145, 1991.

[12] Gear, C.W., and Saad, Y., "Iterative Solution of Linear Equations in ODE Codes," SIAM J. Scientific and Statistical Computing, 4, pp. 583-601, 1983.

[13] Chan, T.F., and Jackson, K.R., "Nonlinearly Preconditioned Krylov Subspace Methods for Discrete Newton Algorithms," SIAM J. Scientific and Statistical Computing, 5, pp. 535-542, 1984.

[14] Brown, P.N., and Hindmarsh, A.C., "Matrix-free Methods for Stiff Systems of ODS's," SIAM J. Numerical Analysis, 23, pp. 610-638, 1986.

[15] Brown, P.N., and Saad, Y., "Hybrid Krylov Methods for Nonlinear Systems of Equations," SIAM J. Scientific and Statistical Computing, 11, pp. 450-481, 1990.

[16] 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.

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

[18] Johann, Z., Hughes, T.J.R., and Shakib, F., "A Globally Convergent Matrix-Free Algorithm for Implicit Time-Marching Schemes arising in Finite Element Analysis in Fluids," Computational methods in Applied Mechanics and Engineering, 87, pp. 281-304, 1991.

[19] Shakib, F., Hughes, T.J.R., and Johann, Z, "Element-by-Element Algorithms for Nonsymmetric Matrix Problems arising in Fluids," In Superlarge Problems in Computational Mechanics, pp. 1-34. Plenum, 1987.

[20] Venkatakrishnan V., and Mavriplis, D.J., "Implicit Solvers for Unstructured Meshes," Journal of Computational Physics 105 pp. 83-91, 1993.

[21] Keyes, D.E., "Aerodynamic Applications of Newton-Krylov-Schwarz Solvers," In Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics, pp. 1-20. Springer, 1995.

[22] Ajmani, K., Ng, W.F., and Lion, M.S., "Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations," J. Computational Physics, 110, pp. 68-81, 1994.

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

[24] Jiang, H., and Forsyth, P.A., "Robust Linear and Nonlinear strategies for Solution of the Transonic Euler Equations," Computers and Fluids, 24, pp. 753-770, 1995.

[25] Mavriplis, D.J., "On Convergence Acceleration Techniques for Unstructured Meshes," AIAA Technical Report, 98-2966, 1998.

[26] Knoll, D.A., and McHugh, P.R., "Inexact Newton's Method Solutions to the Incompressible Navier-Stokes and Energy Equations Using Standard and Matrix-Free Implementations," AIAA Paper, 93-3316, June 1993.

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

[28] Tidriri, M.D., "Krylov Methods for Compressible Flows," ICASE Technical Report 95-48, June 1995.

[29] Tidriri, M.D., "Efficient Preconditioning of Newton-Krylov Matrix-Free Algorithms for Compressible Flows," J. Computational Physics, 132, pp. 51-61, 1997.

[30] Geuzaine, Philippe, "Newton-Krylov Strategy for Compressible Turbulent Flows on Unstructured Meshes," AIAA Journal, vol. 39, no. 3, pp. 528-531, October 2000.

[31] Blanco, Max, "An Implicit Solution Method for the Euler Equations on Unstructured Triangular Grids," University of Toronto, M.A.Sc. thesis, September 1995.

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

[33] Blanco, M., and Zingg, D.W., "Fast Newton-Krylov Method for Unstructured Grids," AIAA Journal, vol. 36, no. 4, pp. 607-612, April 1998.

[34] Pueyo, Alberto, "An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations," University of Toronto, Ph.D. thesis, February 1998.

[35] Pueyo, A., and Zingg, D.W., "Efficient Newton-Krylov Solver for Aerodynamic Computations," AIAA Journal, Vol. 36, No. 11, pp. 1991-1997, November 1998.

[36] Pueyo, A., and Zingg, D.W., "Progress in Newton-Krylov Methods for Aerodynamic Calculations," AIAA Paper, 97-0877, January 1997.

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

[38] Wehner, Edward, "A Newton-Krylov Solver for the Euler Equations on Unstructured Grids," University of Toronto, M.A.Sc thesis, 2001.

[39] Lassaline, J.V., and Zingg, D.W., "Development of an Agglomeration Multigrid Algorithm with Directional Coarsening," AIAA 99-3338, June 1999.

[40] Walsh, Paul Charles, "Adaptive Solution of Viscous Aerodynamic Flows Using Unstructured Grids," University of Toronto, Ph.D. thesis, 1998.

[41] 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, pp. 237-243.

[42] Nelson, T.E., Godin, P., and Zingg, D.W., "Multi-Element Airfoil Computations with One-Equation Turbulence Models," AIAA 33rd Aerospace Sciences Meeting and Exhibit, AIAA 95-0357, Reno, Nevada, January 9-12 1995.

[43] Baldwin, B.S., and Barth, T.J., "A One-Equation Turbulence Transport Model for High Reynolds Wall-Bounded Flows," AIAA Paper, 91-0610, January 1991.

[44] Spalart, P.R., and Allmaras, S.R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, January 1992.

[45] Greiner, Ken, "A Finite Volume Solver for Viscous Turbulent Flows on Mixed Element Unstructured Meshes," University of Toronto, M.A.Sc thesis, 1998.

[46] Jameson, A., and Mavriplis, D., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh," AIAA 23rd Aerospace Sciences Meeting, AIAA-85-0435, January 1985.

[47] Hirsch, C., *Numerical Computation of Internal and External Flows, vol. 2*, John Wiley and Sons, 1990.

[48] Anderson, W.K., Rausch, R., and Bonhaus, D., "Implicit Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids," Proc. AIAA CFD Conf., 12th, San Diego, AIAA Paper, 95-1740-CP, 1995.

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

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

[51] Saleem, M., Pulliam, T.H., and Cheer, A.Y., "Acceleration of Convergence and Spectrum Transformation of Implicit Finite Difference Operators Associated with Navier-Stokes Equations," J. Comp. Phys., no. 104, pp. 1-13, 1993.

[52] Luo, H., Baum, J., Lohner, R., and Cabello, J., "Implicit Schemes and Boundary Conditions for Compressible flows on Unstructured Meshes," Tech. Rep. AIAA-94-0816, 1994.

[53] Chow, E., and Saad, Y., "Experimental Study of ILU Preconditioners for Indefinite Matrices," Supercomputing Inst., TR UMSI 97/95, Univ. of Minnesota, June 1997.

[54] Elman, H.C., "A Stability Analysis of Incomplete LU Factorizations," Math. Comp., no. 47, pp. 191-217, 1986.

[55] Bruaset, A.M., Tveito, A., and Winther, R., "On the Stability of Relaxed Incomplete LU Factorizations," Math. Comp., no. 54, pp. 701-719, 1990.

[56] 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. 36, pp. 457-497, 1993.

[57] Mavriplis, D.J., Jameson, A., and Martinelli, L., "Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes," ICASE Report 89-11, February 1989.

[58] Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., McInnes, L.C., and Smith, B.F., "PETSc home page," http://www.mcs.anl.gov/petsc, 2001.