A Newton-Krylov Solver for the Euler Equations on Unstructured Grids

by

Edward Wehner

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

© Copyright by Edward Wehner 2001
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-62898-1
Abstract

A Newton-Krylov Solver for the Euler Equations on Unstructured Grids

Edward Wehner
Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto
2001

A Newton-Krylov solver for the Euler equations that govern the flow of a compressible inviscid flow, with application to flows over aerodynamic configurations, is presented. The Euler equations and a nonlinear artificial dissipation scheme are discretised in space using a node-based finite-volume formulation on arbitrary polygonal unstructured grids. The resulting system of nonlinear algebraic equations is solved using an inexact Newton strategy in which the linear problem arising at each Newton iteration is solved iteratively using the matrix-free GMRES algorithm preconditioned by an approximate Jacobian matrix subjected to an incomplete factorisation and reordering. Implicit Euler local time stepping is applied when solving transonic flows.

The solver is tested with single and multi-element airfoils in the subsonic and transonic flow regimes. Various GMRES tolerances, preconditioner fill-in levels, and preconditioned tolerances are investigated. Sets of optimum base parameters are determined. Results show the solver to be competitive with other solvers. Recommendations are made for aspects of the solver that are worthy of further study.
Acknowledgements

I wish to thank my supervisor, Prof. D. W. Zingg, for his knowledge, guidance and support during my research. His approachability and respect for self-initiative were always conducive to a comfortable research atmosphere.

I am grateful to my colleagues Luis Manzano, Jason Lassaline, and Peter Blaser, with whom it was always a pleasure to collaborate on the Newton-Krylov project of the UTIAS CFD group, and Todd Chisholm and Marian Nemec, who provided helpful insights into the Newton-Krylov topic.

Words cannot express my gratitude to my mother for everything a mother does for her son.

The Government of Ontario / Victoria Noakes Graduate Scholarship in Science and Technology, and the Graduate Scholarship / Loan provided by the Consejo Nacional de Ciencia y Tecnología (CONACYT) of Mexico are also appreciated.
# Contents

List of Figures ix
List of Tables xi
List of Symbols and Abbreviations xiii

1 Introduction 1
  1.1 Motivation ................................... 1
  1.2 Objective .................................. 3
  1.3 Outline .................................... 3

2 Equation Formulation 5
  2.1 The Euler Equations ........................... 5
  2.2 Spatial Discretisation ........................ 6
  2.3 Artificial Dissipation ......................... 8
  2.4 Boundary Conditions .......................... 9
    2.4.1 Body Surface Boundary ................... 10
    2.4.2 Far-field Boundary ....................... 11

3 Algorithm Formulation 13
  3.1 The Residual Function ......................... 13
  3.2 An Inexact Newton Method ..................... 14
  3.3 Matrix-free GMRES ............................. 15
  3.4 ILU Preconditioning ........................... 16
    3.4.1 The Approximate Jacobian Matrix ........... 17
  3.5 RCM Reordering ............................... 19
  3.6 Startup Strategy .............................. 19

4 Results 21
  4.1 Single-element Airfoil Cases .................. 22
    4.1.1 Subsonic Flow ............................ 23
    4.1.2 Transonic Flow ........................... 27
  4.2 Multi-element Airfoil Case .................... 31
5 Conclusions

Appendices

A Algorithm Flow Diagram

B Examples of the Jacobians for $J_A$

References
List of Figures

2.1 A control volume .................................................. 6
2.2 A representative interior cell .................................. 7
2.3 A representative boundary cell .................................. 9

3.1 Structure of the approximate Jacobian matrix (3,631-node grid) .... 17
3.2 Approximate Jacobian matrix with RCM reordering (3,631-node grid)... 19

4.1 3,631 and 14,193-node triangular grids about NACA 0012 airfoil ...... 23
4.2 Mach number contours and $C_p$ plot ($M_\infty = 0.63, \alpha = 2.0^\circ$)... 24
4.3 Convergence history using base parameters ($M_\infty = 0.63, \alpha = 2.0^\circ$)... 24
4.4 Comparison of $\sigma$ values ($M_\infty = 0.63, \alpha = 2.0^\circ$) .......... 25
4.5 Comparison of GMRES tolerances, $G$ ($M_\infty = 0.63, \alpha = 2.0^\circ$)... 26
4.6 Comparison of $ILU(p)$ fill levels ($M_\infty = 0.63, \alpha = 2.0^\circ$) ....... 26
4.7 Comparison of preconditioner freezing tolerances, $P$ ($M_\infty = 0.63, \alpha = 2.0^\circ$)... 27
4.8 Other single-element subsonic cases .................................. 27
4.9 Mach number contours and $C_p$ plot ($M_\infty = 0.8, \alpha = 1.0^\circ$) ... 28
4.10 Convergence history using base parameters ($M_\infty = 0.8, \alpha = 1.0^\circ$)... 29
4.11 Comparison of $\sigma$ values ($M_\infty = 0.8, \alpha = 1.0^\circ$) .......... 29
4.12 Comparison of GMRES tolerances, $G$ ($M_\infty = 0.8, \alpha = 1.0^\circ$) .... 30
4.13 Comparison of $ILU(p)$ fill levels ($M_\infty = 0.8, \alpha = 1.0^\circ$) ....... 30
4.14 Comparison of preconditioner freezing tolerances, $P$ ($M_\infty = 0.8, \alpha = 1.0^\circ$)... 31
4.15 Other single-element transonic cases .................................. 31
4.16 17,894-node triangular grid about multi-element airfoil ............... 32
4.17 Mach number contours and $C_p$ plot ($M_\infty = 0.2, \alpha = 5.0^\circ$) .... 32
4.18 Convergence history using base parameters ($M_\infty = 0.2, \alpha = 5.0^\circ$) ... 33
4.19 Comparison of $\sigma$ values ($M_\infty = 0.2, \alpha = 5.0^\circ$) .......... 34
4.20 Comparison of GMRES tolerances, $G$ ($M_\infty = 0.2, \alpha = 5.0^\circ$) .... 34
4.21 Comparison of $ILU(p)$ fill levels ($M_\infty = 0.2, \alpha = 5.0^\circ$) ....... 35
4.22 Comparison of preconditioner freezing tolerances, $P$ ($M_\infty = 0.2, \alpha = 5.0^\circ$)... 35
List of Tables

2.1 Far-field inflow and outflow boundary conditions. .................. 12
4.1 Suite of test cases. ............................................. 21
4.2 Base parameters for single-element airfoil cases. ..................... 23
4.3 Drag, lift and moment coefficients ($M_\infty = 0.63, \alpha = 2.0^o$). 23
4.4 Drag, lift and moment coefficients ($M_\infty = 0.8, \alpha = 1.0^o$). 28
4.5 Base parameters for multi-element high-lift airfoil case. ............. 31
4.6 Drag, lift and moment coefficients ($M_\infty = 0.2, \alpha = 5.0^o$). 33
List of Symbols and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Generic matrix in linear system of equations</td>
</tr>
<tr>
<td>$C_D$</td>
<td>Drag coefficient</td>
</tr>
<tr>
<td>$C_L$</td>
<td>Lift coefficient</td>
</tr>
<tr>
<td>$C_M$</td>
<td>Moment coefficient</td>
</tr>
<tr>
<td>$C_p$</td>
<td>Pressure coefficient</td>
</tr>
<tr>
<td>$\mathbf{F}$</td>
<td>Flux tensor</td>
</tr>
<tr>
<td>$G$</td>
<td>GMRES tolerance</td>
</tr>
<tr>
<td>$H$</td>
<td>Total enthalpy</td>
</tr>
<tr>
<td>$K_m$</td>
<td>Krylov subspace of dimension $m$</td>
</tr>
<tr>
<td>$M$</td>
<td>Mach number</td>
</tr>
<tr>
<td>$P$</td>
<td>Preconditioner tolerance</td>
</tr>
<tr>
<td>$\mathbf{Q}$</td>
<td>Vector of conserved variables</td>
</tr>
<tr>
<td>$\mathbf{Q}_G$</td>
<td>Global solution vector</td>
</tr>
<tr>
<td>$\mathbf{R}$</td>
<td>Residual function</td>
</tr>
<tr>
<td>$R$</td>
<td>Density residual</td>
</tr>
<tr>
<td>$\mathbf{R}$</td>
<td>Riemann invariants</td>
</tr>
<tr>
<td>$S$</td>
<td>Interior scheme portion of residual function</td>
</tr>
<tr>
<td>$T$</td>
<td>Boundary scheme portion of residual function</td>
</tr>
<tr>
<td>$V$</td>
<td>Flow velocity</td>
</tr>
<tr>
<td>$V_n$, $V_t$</td>
<td>Normal and tangential flow velocity components</td>
</tr>
<tr>
<td>$a$</td>
<td>Speed of sound</td>
</tr>
<tr>
<td>$b$</td>
<td>Generic left-hand side vector in linear system of equations</td>
</tr>
<tr>
<td>$e$</td>
<td>Total energy per unit mass</td>
</tr>
<tr>
<td>$h$</td>
<td>Time step</td>
</tr>
<tr>
<td>$i$, $j$</td>
<td>Unit vectors in the x and y directions</td>
</tr>
<tr>
<td>$m$</td>
<td>GMRES iteration number</td>
</tr>
<tr>
<td>$n$</td>
<td>Newton iteration number</td>
</tr>
<tr>
<td>$n$</td>
<td>Local normal vector</td>
</tr>
<tr>
<td>$n_x$, $n_y$</td>
<td>Normal vector components in the x and y directions</td>
</tr>
<tr>
<td>$n_e$</td>
<td>Summation limit for cell edges</td>
</tr>
<tr>
<td>$n_f$</td>
<td>Summation limit for boundary cell faces</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>( p )</td>
<td>Pressure</td>
</tr>
<tr>
<td>( p )</td>
<td>Level of ILU fill-in</td>
</tr>
<tr>
<td>( r_m )</td>
<td>Residual at the ( m )th GMRES iteration</td>
</tr>
<tr>
<td>( r )</td>
<td>Position vector</td>
</tr>
<tr>
<td>( \mathbf{t} )</td>
<td>Local tangent vector</td>
</tr>
<tr>
<td>( t_x, t_y )</td>
<td>Tangent vector components in the ( x ) and ( y ) directions</td>
</tr>
<tr>
<td>( u, v )</td>
<td>Velocity component in the ( x ) and ( y ) directions</td>
</tr>
<tr>
<td>( x )</td>
<td>Generic solution vector in linear system of equations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>Angle of attack</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>Perturbation constant for matrix-free GMRES</td>
</tr>
<tr>
<td>( \varepsilon^{(2)}, \varepsilon^{(4)} )</td>
<td>Second and fourth-difference artificial dissipation switches</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Specific heat ratio of gas</td>
</tr>
<tr>
<td>( \kappa^{(2)}, \kappa^{(4)} )</td>
<td>Second and fourth-difference artificial dissipation control constants</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Artificial dissipation scaling factor</td>
</tr>
<tr>
<td>( \nu )</td>
<td>Vector of matrix-free GMRES vector product</td>
</tr>
<tr>
<td>( \Omega )</td>
<td>Surface area</td>
</tr>
<tr>
<td>( \partial \Omega )</td>
<td>Surface contour</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Density</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Fourth-difference artificial dissipation control constant</td>
</tr>
<tr>
<td>( \mathbf{I} )</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>( \mathbf{J}_E, \mathbf{J}_A )</td>
<td>Exact and approximate Jacobian matrices</td>
</tr>
<tr>
<td>( b )</td>
<td>Subscript to indicate a boundary value</td>
</tr>
<tr>
<td>( e )</td>
<td>Symbol and subscript to indicate an extrapolated value</td>
</tr>
<tr>
<td>( i, k )</td>
<td>Subscripts for arbitrary cells</td>
</tr>
<tr>
<td>( m )</td>
<td>Subscript to indicate a boundary node</td>
</tr>
<tr>
<td>( \infty )</td>
<td>Symbol and subscript to indicate freestream condition</td>
</tr>
<tr>
<td>( - )</td>
<td>Superscript to indicate non-dimensionalised variable</td>
</tr>
<tr>
<td>( 0 )</td>
<td>Subscript to indicate stagnation condition</td>
</tr>
</tbody>
</table>

**Abbreviations**

- CFD: Computational Fluid Dynamics
- CFL: Courant-Friedrich-Lewy (number)
- CPU: Central Processing Unit
- GMRES: Generalized Minimum Residual (algorithm)
- ILU: Incomplete Lower-Upper (factorisation)
- PETSc: Portable, Extensible Toolkit for Scientific Computation
- RCM: Reverse Cuthill-McKee (reordering)
- RHS: Right-Hand Side
Chapter 1

Introduction

1.1 Motivation

A principal interest in the field of Computational Fluid Dynamics (CFD) is the development of efficient solution methods for the Navier-Stokes equations, which model the behaviour of fluid flows. This interest is strongly motivated by the application of such methods in the aerodynamic design cycle.

Newton-Krylov solvers are a family of implicit solvers that apply quasi-Newton methods to solve the Navier-Stokes equations. As with other solvers, they are usually first developed for the Euler equations, since, without friction and thermal conductivity terms, the Euler equations are simpler than the Navier-Stokes equations but retain the non-linear behaviour and much of the physical significance of the superset equations. Some Newton-Krylov solvers use structured grids to transform physical space into computational space, while others use unstructured grids. Structured grids are simpler, but unstructured grids conform better to complex geometries and are easier to adapt and refine as needed for increased accuracy.

Application of an exact Newton method to solve large systems of partial differential equations is impractical due to the memory and CPU resources required. In their work of 1982, Dembo, Eisenstat, and Steihaug [1] showed that, if properly implemented, inexact-Newton methods can save on these resources considerably by not performing each Newton iteration exactly, while still offering quadratic convergence. Since then, a variety of independent efforts have used inexact Newton methods in CFD. Vanka [2] developed Newton solvers for the Navier-Stokes equations. Venkatakrishnan [3], Orkwis [4], and Whitfield
Chapter 1. Introduction

and Taylor [5] applied Newton-like methods to solve transonic flows. These efforts used direct methods or stationary iterative methods to solve the linear problem that arises at each Newton iteration, and explicitly formed the Jacobian matrix.

The replacement of direct methods with Krylov iterative methods, and of explicit Jacobian matrix formation with matrix-free formulations, originates in works by Gear and Saad [6], Chan and Jackson [7], Brown and Hindmarsh [8], and Brown and Saad [9]. It may have been in this last work that the term "Newton-Krylov" appeared for the first time. The first use in CFD of the Generalized Minimum Residual (GMRES) [10] Krylov method can be traced to Wigton, Yu, and Young [11], and Johan, Hughes, and Shakib [12, 13]. Venkatakrishnan and Mavriplis [14] showed Newton-Krylov methods with Incomplete Upper-Lower (ILU) factorisation preconditioning to be competitive with multigrid methods. Keyes [15] showed the same with a matrix-free formulation. Tidriri [16] studied preconditioning techniques for matrix-free Newton-Krylov methods. Other works that have investigated particular aspects of matrix-free inexact Newton-Krylov solvers include [17, 18, 19, 20, 21, 22, 23].

The immediate predecessors of the present thesis, within the University of Toronto Institute for Aerospace Studies (UTIAS) CFD group, are two inexact Newton-Krylov solvers written by Blanco [24, 25], and by Pueyo [26, 27]. The work of Blanco solves the Euler equations and is based on unstructured triangular grids using a node-based, centred finite-volume discretisation, while the work of Pueyo solves both the Euler and Navier-Stokes equations and is based on structured "C" grids using second-order centred-difference operators. These works apply second and fourth-difference scalar artificial dissipation, and use the GMRES algorithm, preconditioned by the Jacobian matrix subjected to an ILU factorisation and Reverse Cuthill-McKee (RCM) reordering, to solve the linear system that arises at each Newton iteration. Blanco has tested both matrix-present and matrix-free GMRES, while Pueyo has concentrated on a restarted version of the latter. In their use of matrix-free GMRES, both researchers have used an approximation to the Jacobian matrix for the preconditioner. In order to overcome the instabilities that can develop during the first iterations of transonic cases, as a start-up strategy Blanco has applied matrix-present GMRES with a first-order Jacobian, while Pueyo has used the approximately factored algorithm of ARC2D in diagonal form with two levels of mesh sequencing.
1.2 Objective

The objective of this thesis project is to develop an efficient two-dimensional Newton-Krylov solver for the Euler equations, with application to subsonic and transonic flows over aerodynamic configurations. The solver uses a node-based finite-volume formulation on arbitrary polygonal unstructured grids. The solver applies second and fourth-difference scalar artificial dissipation, and resolves the linear system that arises at each Newton iteration using matrix-free GMRES with an ILU-factorised, RCM-reordered approximate Jacobian preconditioner. An implicit Euler local time step strategy is used for the start-up of transonic cases. Tests with single and multi-element airfoils are run to verify the proper functioning of the solver and determine the parameters that optimise its performance.

1.3 Outline

This document is divided into five chapters. Following on the introduction of the first chapter, the second chapter describes the Euler equations and their spatial discretisation, including artificial dissipation and boundary conditions. The third chapter sets forth the methodology applied to solve the Euler equations as cast in the second chapter. The fourth chapter is an exposition of test results. Based on these results, the fifth and final chapter presents conclusions and suggestions for future work related to the present solver.
Chapter 2

Equation Formulation

This chapter presents the Euler equations and their spatial discretisation, including artificial dissipation and boundary conditions.

2.1 The Euler Equations

The integral conservative form of the Euler equations for an unsteady, two-dimensional, compressible inviscid flow across a surface \( \Omega \), can be expressed as:

\[
\frac{d}{dt} \int_{\Omega} Q \, dA + \int_{\Omega} \nabla \cdot \overline{F} \, dA = 0
\]  

(2.1)

where \( Q \) is the vector of conserved variables and \( \overline{F}(Q) \) is the inviscid flux tensor:

\[
Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}
\]  

(2.2)

\[
\overline{F} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho w \\ u(e + p) \end{bmatrix} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{bmatrix}
\]  

(2.3)

The dimensional variables, density \( (\bar{\rho}) \), velocity \( (\bar{\bar{u}}, \bar{\bar{v}}) \) and total energy \( (\bar{e}) \), are non-dimensionalised using free-stream \( (\infty) \) values:
Chapter 2. Equation Formulation

Figure 2.1: A control volume

\[ \rho = \frac{\hat{\rho}}{\rho_\infty} ; \quad u = \frac{\hat{u}}{a_\infty} ; \quad v = \frac{\hat{v}}{a_\infty} ; \quad e = \frac{\hat{e}}{\rho_\infty a_\infty^2} \] (2.4)

where \( a \) is the speed of sound, which for ideal fluids is \( a = \sqrt{\gamma p/\rho} \). Pressure is related to the conserved variables by the equation of state for a perfect gas:

\[ p = (\gamma - 1) \left[ e - \frac{1}{2} \rho (u^2 + v^2) \right] \] (2.5)

which provides closure for Eq. 2.1, by relating the thermodynamic variables of the system.

2.2 Spatial Discretisation

The computational domain is discretised using a finite-volume formulation in which the control volumes are arbitrary polygonal cells. These cells compose a dual grid, formed by connecting the centroids of cells of a triangular primary grid, as shown in Fig. 2.1.

A representative cell \( i \) from the interior of the domain, whose \( n_e \) edges border on cells such as representative neighbour cell \( k \), is shown in Fig. 2.2. The cell has an area \( \Omega_i \) and outward normals \( \mathbf{n}_{ik} = [\Delta y_{ik}, -\Delta x_{ik}] \). The four conserved variables of the solution vector \( \mathbf{Q}_i \) are stored at the cell node.
From Gauss' theorem in two dimensions, which transforms surface integrals into line integrals, Eq. 2.1 can be rewritten as:

$$\frac{d}{dt} \int_{\Omega} Q \, dA + \oint_{\partial \Omega} \mathbf{F} \cdot \mathbf{n} \, ds = 0$$  \hspace{1cm} (2.6)

where $\partial \Omega$ is the contour enclosing surface $\Omega$, and $\mathbf{n}$ is a vector outwardly normal to the surface. For cell $i$, this equation is applied as:

$$\Omega_i \frac{d}{dt} Q_i + \sum_{k=1}^{n_{e}} F_{ik} \cdot \mathbf{n}_{ik} = 0$$  \hspace{1cm} (2.7)

on the assumption that the values of the conserved variables of $Q_i$ are constant throughout the cell and equal to their values at the cell node. The flux tensor $F_{ik}$ is taken as the average of the flux tensors of cells $i$ and $k$, thus Eq. 2.7 can be written as:

$$\Omega_i \frac{d}{dt} Q_i = -\frac{1}{2} \sum_{k=1}^{n_{e}} (F_{i} + F_{k}) \cdot \mathbf{n}_{ik}$$  \hspace{1cm} (2.8)

For interior cells, $\sum F_{i} \cdot \mathbf{n}_{ik} = 0$, since it is a sum over a closed contour.
2.3 Artificial Dissipation

A scalar artificial dissipation term is added to the right-hand side of Eq. 2.8, to provide numerical stability. The term consists of a scalar dissipation coefficient, a second-difference Laplacian operator, which is applied near shocks, and a weaker fourth-difference biharmonic operator, which is used everywhere else in the computational domain.

The second-difference Laplacian operator is:

$$\sum_{k=1}^{n_e} \varepsilon_{ik}^{(2)} (Q_k - Q_i)$$

(2.9)

where

$$\varepsilon_{ik}^{(2)} = \kappa^{(2)} \sum_{k=1}^{n_e} \left| p_k - p_i \right| \div \left( \sum_{k=1}^{n_e} (p_k + p_i) \right)$$

(2.10)

The coefficient $\varepsilon_{ik}^{(2)}$ uses the pressure gradient across cells $i$ and $k$ to detect shocks and thus trigger the second-difference dissipation. $\kappa^{(2)}$ is a user-defined constant that controls the strength of the dissipation. For each cell edge contribution, the average of $\varepsilon_{ik}^{(2)}$ for neighbour cells $i$ and $k$ is used.

The fourth-difference biharmonic operator is:

$$\sum_{k=1}^{n_e} \varepsilon_{ik}^{(4)} (L_k - L_i)$$

(2.11)

where

$$L_i = \sum_{k=1}^{n_e} (Q_k - Q_i)$$

(2.12)

and

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

(2.13)

The coefficient $\varepsilon_{ik}^{(4)}$ is a switch that turns on the fourth-difference dissipation whenever the second-difference dissipation is turned off. $\kappa^{(4)}$ is a user-defined constant that controls the strength of the dissipation.

So that the artificial dissipation is properly scaled with respect to solution values, it is multiplied by a scaling factor obtained from the largest eigenvalue of the flux Jacobian:

$$\lambda_{ik} = \sum_{k=1}^{n_e} \left| u \Delta y - v \Delta x \right| + a \sqrt{\Delta x^2 + \Delta y^2}$$

(2.14)
Section 2.4. Boundary Conditions

Figure 2.3: A representative boundary cell

where \( u, v \) and \( a \) are calculated based on the average of the solution vectors of cells \( i \) and \( k \):

\[
Q_{ik} = \frac{Q_i + Q_k}{2}
\]  

(2.15)

Combining the scaling factor and the second and fourth-difference operators, the complete scalar artificial dissipation vector is:

\[
\frac{1}{2} \sum_{k=1}^{n_f} \lambda_{ik} \left[ \varepsilon_{ik}^{(2)} (Q_k - Q_i) - \varepsilon_{ik}^{(4)} (L_k - L_i) \right]
\]  

(2.16)

This vector applies only to cell edges; no artificial dissipation is applied at the boundary cell faces described in Section 2.4.

2.4 Boundary Conditions

The computational domain presents two types of boundaries: the body surface boundary of the solid aerodynamic configuration placed in the domain, and the far-field boundary at the outer limit of the domain itself. In order to properly model the flow in the domain, certain conditions must be imposed at the boundaries.

A representative boundary cell is shown in Fig. 2.3. In addition to the centre node, there is a node at the mid-point \( m \) of each of \( n_f \) boundary faces. A vector \( Q_b \), containing four boundary values, is stored at each face node. Some of the boundary values are calculated from free-stream values, while others are extrapolated to zeroth or first order
from the conserved variables stored at the centre node. The extrapolations are based on the Taylor expansion:

$$ \text{Zeroth Order} $$

$$ Q_x = Q_i + (\nabla Q)_i \cdot \Delta r + O(\Delta r)^2 $$

(2.17)

where

$$ (\nabla Q)_i = \frac{1}{\Omega_i} \sum_{k=1}^{n_e} (Q_{ik} n_x i + Q_{ik} n_y j) $$

(2.18)

and

$$ \Delta r = r_x i + r_y j $$

(2.19)

The flux contribution from the boundary faces is:

$$ \sum_{m=1}^{n_f} \bar{F}_m \cdot n_{im} $$

(2.20)

where now \( \bar{F}(Q_b(Q_i)) \), instead of simply \( \bar{F}(Q_i) \) as at all other nodes in the computational domain.

### 2.4.1 Body Surface Boundary

Given that an aerodynamic configuration such as an airfoil presents a rigid surface to the flow, at body surface boundaries the normal component of the flow velocity is zero. That is:

$$ V_n = V \cdot n = (u i + v j) \cdot (n_x i + n_y j) = u n_x + v n_y = 0 $$

(2.21)

where \( n \) is the local unit vector normal to the body surface. The tangential component of the flow velocity, given by

$$ V_t = V \cdot t = (u i + v j) \cdot (t_x i + t_y j) = u t_x + v t_y $$

(2.22)

where \( t \) is the local unit vector tangent to the body surface, is extrapolated to first order. Pressure \( p \) is also extrapolated to first order, and stagnation enthalpy \( [H = (e + p)/\rho] \) is set to free-stream values.

In view of these considerations, for body surface boundaries, density and energy are expressed as:

$$ \rho = \frac{\gamma p_e}{(\gamma - 1)(H_\infty - \frac{1}{2}V_{in}^2)} $$

(2.23)
Section 2.4. Boundary Conditions

The velocities $u$, $v$, $V_n$, and $V_t$ are related by:

$$
\begin{bmatrix}
  u \\
  v
\end{bmatrix} =
\begin{bmatrix}
  t_x & n_x \\
  t_y & n_y
\end{bmatrix}
\begin{bmatrix}
  V_t \\
  V_n
\end{bmatrix}
$$

Thus, considering Eq. 2.21, and substituting Eqs. 2.22, 2.23, and 2.24 in Eq. 2.2, at body surface boundaries there is:

$$
Q_b = \begin{bmatrix}
  q_{b1} \\
  q_{b2} \\
  q_{b3} \\
  q_{b4}
\end{bmatrix} =
\begin{bmatrix}
  \frac{\gamma p_e}{(\gamma - 1)(H_{\infty} - \frac{1}{2}V_t^2)}V_t t_x \\
  \frac{\gamma p_e}{(\gamma - 1)(H_{\infty} - \frac{1}{2}V_t^2)}V_t t_y \\
  \frac{\gamma p_e}{\gamma - 1} + \frac{1}{2} \frac{\gamma p_e}{(\gamma - 1)(H_{\infty} - \frac{1}{2}V_t^2)}V_t^2
\end{bmatrix}
$$

2.4.2 Far-field Boundary

The flow over the aerodynamic configuration placed in the computational domain generates disturbances in the free-stream conditions. These disturbances convect and diffuse throughout the domain, and when they reach the far-field boundary, instead of exiting, they can be reflected back into the domain, producing large oscillations that hinder solver convergence. To ensure that the disturbances exit the domain, non-reflecting boundary conditions, which use the four Riemann invariants, are applied. Flow through the far-field boundary is taken to be locally normal to the boundary, so one-dimensional Riemann invariants are used. They are:

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

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

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

$$
R_4 = V_t
$$

In terms of these Riemann invariants, there is:

$$
V_n = \frac{1}{2}(R_1 + R_2)
$$

and

$$
a = \frac{1}{4}(\gamma - 1)(R_2 - R_1)
$$
such that density and energy are:

\[
\rho = \left( \frac{a^2}{\gamma R_3} \right)^{\frac{1}{\gamma-1}}
\]

\[
e = \frac{\rho a^2}{\gamma(\gamma - 1)} + \frac{1}{2}(u^2 + v^2)
\]

where \( u \) and \( v \) are obtained considering Eqs. 2.21, 2.22, 2.25, 2.30, and 2.31.

The Riemann boundary conditions are calculated from free-stream values or extrapolated to zeroth order, depending on the free-stream Mach number \( M_\infty \) and whether the boundary is of inflow or outflow. Table 2.1 summarises how the far-field boundary conditions are calculated. As an example, substituting Eqs. 2.27 to 2.34 in Eq. 2.2, and applying the rules of Table 2.1 for a far-field inflow boundary with \( M_\infty < 1 \), there is:

\[
Q_b = \begin{bmatrix}
q_b1 \\
q_b2 \\
q_b3 \\
q_b4
\end{bmatrix} = \begin{bmatrix}
A \\
AB \\
AC \\
D
\end{bmatrix}
\]

where

\[
A = \left[ \left( \frac{1}{2} \left( \frac{(\gamma - 1)}{\gamma} \frac{V_{n_e} + 2a_e}{\gamma - 1} - \left( \frac{V_{n_\infty} - 2a_\infty}{\gamma - 1} \right) \right) \right]^{\frac{1}{\gamma-1}}
\]

\[
B = \frac{1}{2} \left( \frac{V_{n_\infty} - 2a_\infty}{\gamma - 1} + \left( \frac{V_{n_e} + 2a_e}{\gamma - 1} \right) \right) n_x + V_{l_\infty} t_x
\]

\[
C = \frac{1}{2} \left( \frac{V_{n_\infty} - 2a_\infty}{\gamma - 1} + \left( \frac{V_{n_e} + 2a_e}{\gamma - 1} \right) \right) n_y + V_{l_\infty} t_y
\]

\[
D = \frac{A}{\gamma(\gamma - 1)} \left( \frac{1}{4} (\gamma - 1) \left[ \left( \frac{V_{n_e} + 2a_e}{\gamma - 1} \right) - \left( \frac{V_{n_\infty} - 2a_\infty}{\gamma - 1} \right) \right] \right)^2 + \frac{1}{2} (B^2 + C^2)
\]

<table>
<thead>
<tr>
<th>( M_\infty )</th>
<th>( R_1 )</th>
<th>( R_2 )</th>
<th>( R_3 )</th>
<th>( R_4 )</th>
<th>( R_1 )</th>
<th>( R_2 )</th>
<th>( R_3 )</th>
<th>( R_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1</td>
<td>( \infty )</td>
<td>( e )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( e )</td>
<td>( e )</td>
<td>( e )</td>
</tr>
</tbody>
</table>
| > 1 | \( \infty \) | \( \infty \) | \( \infty \) | \( \infty \) | \( e \) | \( e \) | \( e \) | \( e \)

Table 2.1: Far-field inflow and outflow boundary conditions.
Chapter 3

Algorithm Formulation

This chapter explicitly states the equation resolved by the present solver and describes the methodology used to obtain the solution. The methodology involves an inexact Newton method, the matrix-free Generalized Minimum Residual (GMRES) algorithm, Incomplete Lower-Upper (ILU) preconditioning, Reverse Cuthill-McKee (RCM) reordering, and Local Time Stepping. The solution algorithm is written in the C++ computer language and uses routines from the Portable, Extensible Toolkit for Scientific Computation (PETSc) [28]. A flow diagram of the algorithm is provided in Appendix A.

3.1 The Residual Function

The discretised flux integrals, artificial dissipation term and boundary condition equations cast in the previous chapter can be combined into a single function for the entire computational domain. The function, referred to as the residual function, is a vector of cells four-entry vectors and leads to the non-linear system of equations:

\[ R(\hat{Q}) = 0 \]  

(3.1)

It is this equation that is resolved by the present solver, finding the solution vector \( \hat{Q} \) that satisfies the system.

The ith equation of the system of Eq. 3.1 is obtained from Eq. 2.8, whose time derivative is identically zero at steady state, and expressions 2.16 and 2.20. It can be expressed as:

\[ R_i(\hat{Q}) = S_i(\hat{Q}) + T_i(\hat{Q}) = 0 \]  

(3.2)
where

\[ S_i = \frac{1}{\Omega_i} \left[ \frac{1}{2} \sum_{k=1}^{n_c} (\mathbf{F}_i + \mathbf{F}_k) \cdot \mathbf{n}_{ik} - \sum_{k=1}^{n_c} \lambda_{ik} \left[ \epsilon_{ik}^{(2)} (\mathbf{Q}_k - \mathbf{Q}_i) - \epsilon_{ik}^{(4)} (\mathbf{L}_k - \mathbf{L}_i) \right] \right] \]  (3.3)

and

\[ T_i = \frac{1}{\Omega_i} \sum_{m=1}^{n_f} \mathbf{F}_m \cdot \mathbf{n}_{im} \]  (3.4)

### 3.2 An Inexact Newton Method

The present solver applies an inexact Newton method to solve Eq. 3.1.

Newton's method is based on Newton's linearisation to first order which, applied to Eq. 3.1, is:

\[ \mathbf{R}(\tilde{\mathbf{Q}}^{(n+1)}) \approx \mathbf{R}(\tilde{\mathbf{Q}}^{(n)}) + \mathbf{J}_E^{(n)} \Delta \tilde{\mathbf{Q}}^{(n)} = 0 \]  (3.5)

where \( \Delta \tilde{\mathbf{Q}}^{(n)} \equiv \tilde{\mathbf{Q}}^{(n+1)} - \tilde{\mathbf{Q}}^{(n)} \), and \( \mathbf{J}_E \), the exact Jacobian matrix of \( \mathbf{R} \), is:

\[ \mathbf{J}_E = \frac{\partial \mathbf{R}(\tilde{\mathbf{Q}})}{\partial \tilde{\mathbf{Q}}} \]  (3.6)

Hence, to first-order accuracy, the non-linear system of Eq. 3.1 can be replaced by the linear system of equations:

\[ \mathbf{R}(\tilde{\mathbf{Q}}^{(n)}) + \mathbf{J}_E^{(n)} \Delta \tilde{\mathbf{Q}}^{(n)} = 0 \]  (3.7)

which it is customary to write as:

\[ \mathbf{J}_E^{(n)} \Delta \tilde{\mathbf{Q}}^{(n)} = -\mathbf{R}(\tilde{\mathbf{Q}}^{(n)}) \]  (3.8)

with the right-hand term being referred to as "the right-hand side" or RHS.

The solution vector \( \tilde{\mathbf{Q}} \) that satisfies Eq. 3.1 is found by an iterative process, started by proposing an initial guess vector, \( \tilde{\mathbf{Q}}^{(0)} \), based on, for example, free-stream values. At each iteration, referred to as a "Newton iteration", the solution vector is updated using the definition of \( \Delta \tilde{\mathbf{Q}}^{(n)} \) as:

\[ \tilde{\mathbf{Q}}^{(n+1)} = \tilde{\mathbf{Q}}^{(n)} + \Delta \tilde{\mathbf{Q}}^{(n)} \]  (3.9)

where the updating vector \( \Delta \tilde{\mathbf{Q}}^{(n)} \) is found by solving Eq. 3.8 using the method described in Section 3.3. At the nth Newton iteration, there is:

\[ \| \mathbf{R}(\tilde{\mathbf{Q}}^{(n)}) + \mathbf{J}_E^{(n)} \Delta \tilde{\mathbf{Q}}^{(n)} \| \leq \eta^{(n)} \| \mathbf{R}(\tilde{\mathbf{Q}}^{(n)}) \| \]  (3.10)
which represents an inexact Newton method where the solution vector $\hat{Q}^{(n)}$ does not zero the residual function of Eq. 3.8, but rather reduces it by a factor of $\eta^{(n)}$. In the limit, when $\eta^{(n)} = 0$, Eq. 3.10 reverts to Eq. 3.8 of Newton’s method.

To determine the level of convergence of the residual function at the $n$th Newton iteration, the present solver calculates a residual $R^{(n)}$ given by the $L_2$-norm of the first equation of the residual function, corresponding to conservation of mass, normalised by the same $L_2$-norm taken at the first Newton iteration:

$$R^{(n)} = \frac{||R_1^{(n)}||_2}{||R_1^{(0)}||_2}$$

(3.11)

This residual is referred to as the “density residual”. The present solver stops the Newton iterations when the density residual is equal to or smaller than $10^{-10}$.

### 3.3 Matrix-free GMRES

The linear system of Eq. 3.8 is solved for $\Delta \hat{Q}$ at each Newton iteration using the GMRES algorithm, an iterative solver belonging to the family of Krylov solvers.

For any linear system of equations of the form:

$$Ax = b$$

(3.12)

the GMRES algorithm finds the iterate $x_m \in \{x_0 + K_m\}$ that minimises the $L_2$ norm of the residual $r_m = b - Ax_m$, where $x_0$ is an initial guess, $K_m$ is a Krylov subspace of the form:

$$K_m = \text{span}\{\nu, A\nu, A^2\nu, \ldots, A^{m-1}\nu\}$$

(3.13)

and

$$\nu_1 = \frac{r_0}{||r_0||_2} = \frac{b - Ax_0}{||b - Ax_0||_2}$$

(3.14)

When solving Eq. 3.8, at the $m$th GMRES iteration of the $n$th Newton iteration the GMRES residual is:

$$r_m = -R(\hat{Q}^{(n)}) - \mathcal{J}^{(n)}(\Delta \hat{Q})_m$$

(3.15)

If $r_m \neq 0$, then $(\Delta \hat{Q})_m$ is still not the exact solution of Eq. 3.8. However, Pueyo [26, 27] has shown that solving Eq. 3.8 accurately does not necessarily equally reduce the residual function of the non-linear system of Eq. 3.1. Therefore, to save CPU time, the present
solver stops the GMRES iterations when the GMRES residual has decreased by $G$ orders of magnitude defined by the user. In addition, since the storage requirements of GMRES increase linearly and the CPU expense increases quadratically with the number of vectors in the Krylov subspace, the present solver terminates GMRES at $m = 40$, and restarts it using the latest value of $\Delta \mathbf{Q}$ to calculate the new initial residual $r_0$. This restarted form of the GMRES algorithm is denoted GMRES(40).

The matrix-vector products required by the GMRES algorithm to form the Krylov subspace of Eq. 3.13 can be approximated using finite-differences:

$$J_E^{(n)} \nu \simeq \frac{R(\hat{\mathbf{Q}}^{(n)} + \epsilon \nu) - R(\hat{\mathbf{Q}}^{(n)})}{\epsilon} \quad (3.16)$$

where $\epsilon$ is obtained from:

$$\epsilon \|\nu\|_2 \simeq \sqrt{\epsilon_c} \quad (3.17)$$

with the present solver setting $\epsilon_c$ to machine zero. This matrix-free GMRES strategy eliminates the need to explicitly form the exact Jacobian matrix $J_E$, thus producing a savings in matrix storage.

### 3.4 ILU Preconditioning

In solving systems of the form of Eq. 3.12, the GMRES algorithm can converge slowly or not at all if the system is ill-conditioned. The exact Jacobian matrix $J_E$ is off-diagonally dominant, causing the system of Eq. 3.8 to be ill-conditioned and thus susceptible to this shortcoming. To alleviate this difficulty, the present solver right preconditioning the system of Eq. 3.8.

Applied to a system of the form of Eq. 3.12, right-preconditioning can be expressed as:

$$(AM^{-1})(M\mathbf{x}) = \mathbf{b} \quad (3.18)$$

where $M$ is a matrix that makes this system better conditioned than the original one. The matrix $M$ used by this solver is obtained from an incomplete lower-upper factorisation of an approximate Jacobian matrix $J_A$, described in the following Subsection 3.4.1. With right-preconditioning, Eq. 3.8 becomes:

$$[J_E^{(n)}(J_A^{(n)})^{-1}][J_A^{(n)}\Delta \mathbf{Q}^{(n)}] = -R(\hat{\mathbf{Q}}^{(n)}) \quad (3.19)$$
The factorisation of the approximate Jacobian matrix $\mathcal{J}_A$ is denoted $ILU(p)$, where the parameter $p$ refers to the level of fill-in allowed in the factorisation. A fill-in level of $p = 0$ indicates that no fill-in beyond the original matrix pattern of nonzeros is allowed in the $L$ and $U$ factor matrices. As $p$ grows, more nonzero elements are added to the $L$ and $U$ factors, thus making the factorisation more complete. A more complete factorisation should reduce the number of GMRES iterations needed to converge the GMRES residual, but this happens at the expense of increased storage and time per iteration.

It has been shown by Pueyo [26, 27] that the number of GMRES iterations at each Newton iteration does not increase if the preconditioner matrix $\mathcal{J}_A$ is not recalculated at each Newton iteration. Thus, the present solver freezes the preconditioner when the density residual of Eq. 3.11 has decreased to a user-defined tolerance, $P$.

3.4.1 The Approximate Jacobian Matrix

The approximate Jacobian matrix $\mathcal{J}_A$ on which the preconditioning of Eq. 3.8 is based is a non-symmetric sparse block matrix with block dimensions $\text{cells} \times \text{cells}$. As the position of the off-diagonal blocks depends on the ordering of the cell nodes, the unstructured nature of the grid of the present solver produces a matrix with a large bandwidth. An example of the matrix structure is shown in Fig. 3.1.

![Figure 3.1: Structure of the approximate Jacobian matrix (3,631-node grid).](image-url)
Chapter 3. Algorithm Formulation

Each nonzero entry of \( J_A \) is a 4 \( \times \) 4 Jacobian matrix corresponding to a partial derivative of Eq. 3.2 with respect to \( Q_i \) (diagonal blocks) or \( Q_k \) (off-diagonal blocks). Including three approximations, the diagonal terms are of the form:

\[
\frac{\partial R_i}{\partial Q_i} = \frac{1}{\Omega_i} \left[ \sum_{k=1}^{n_x} \frac{\partial \bar{F}_i}{\partial Q_i} \cdot n_{ik} + \frac{1}{2} \sum_{k=1}^{n_x} \lambda_{ik} \left( \varepsilon_{RHS}^{(2)} + \sigma \varepsilon_{RHS}^{(4)} \right) + \sum_{m=1}^{n_f} \left( \frac{\partial \bar{F}_m}{\partial Q_b} \cdot \frac{\partial Q_m}{\partial Q_j} \right) \cdot n_{im} \right]
\]  

(3.20)

and off-diagonal terms are of the form:

\[
\frac{\partial R_i}{\partial Q_k} = \frac{1}{\Omega_i} \left( \frac{1}{2} \sum_{k=1}^{n_x} \left[ \frac{\partial \bar{F}_i}{\partial Q_k} \cdot n_{ik} - \lambda_{ik} \left( \varepsilon_{RHS}^{(2)} + \sigma \varepsilon_{RHS}^{(4)} \right) \right] \right)
\]  

(3.21)

where, since \( \sum \bar{F}_i \cdot n_{ik} = 0 \) for interior cells, the first sum of Eq. 3.20 is nonzero only for boundary cells. Examples of the Jacobians contained in Eqs. 3.20 and 3.21 are provided in Appendix B.

The three approximations incorporated in Eqs. 3.20 and 3.21 are:

1. The scalar dissipation coefficient \( \lambda_{ik} \) is treated as if it were independent of \( Q \).

2. The fourth-difference biharmonic operator (2.11) is neglected and the coefficient \( \varepsilon_{ik}^{(2)} \) of the second-difference Laplacian operator (2.9) is replaced by a linear combination of the second and fourth-difference dissipation coefficients of the RHS, that is:

\[
\varepsilon_{J_A}^{(2)} = \varepsilon_{RHS}^{(2)} + \sigma \varepsilon_{RHS}^{(4)}
\]  

(3.22)

where \( \sigma \) is a user-defined control constant. Since the fourth-difference operator involves neighbour-of-neighbour nodes, its exclusion reduces the off-diagonal dominance of the approximate Jacobian matrix and decreases the number of blocks per row, thus economising matrix storage.

3. Given that \( \bar{F}_m (Q_b (Q_c (Q_i))) \), applying the chain rule to Eq. 3.4 gives:

\[
\frac{\partial \Gamma_i}{\partial Q_i} = \sum_{m=1}^{n_f} \left( \frac{\partial \bar{F}_m}{\partial Q_b} \cdot \frac{\partial Q_m}{\partial Q_c} \cdot \frac{\partial Q_c}{\partial Q_i} \right) \cdot n_{im}
\]

(3.23)

However, for the approximate Jacobian matrix, it is taken that \( Q_c = Q_i \), corresponding to a zeroth-order extrapolation, as shown in Eq. 2.17, and thus:

\[
\frac{\partial Q_c}{\partial Q_i} = \mathcal{I}
\]

(3.24)

where \( \mathcal{I} \) is the identity matrix. This reduction of the boundary condition treatment from first order to zeroth order economises matrix storage.
3.5 RCM Reordering

The large bandwidth of the approximate Jacobian matrix $J_A$ can affect the quality of the preconditioning matrix. This is because off-diagonal matrix elements produce many nonzero elements in the $L$ and $U$ factor matrices of a complete $LU$ factorisation, and such elements may be lost in the incomplete $LU$ factorisation used to form the preconditioning matrix. The present solver addresses this difficulty by reordering the approximate Jacobian matrix using the Reverse Cuthill-McKee (RCM) bandwidth reduction strategy. Fig. 3.2 shows the structure of the matrix of Fig. 3.1 after RCM reordering.

3.6 Startup Strategy

The initial Newton iterations are susceptible to converge slowly or even to diverge if the initial guess vector $\mathbf{Q}^{(0)}$ is far from the final solution vector. This is particularly the case for flows with shocks.

The present solver applies an implicit Euler start up strategy to attenuate this difficulty when solving transonic flows, in which freestream Mach numbers are in the range $0.7 < M_{\infty} < 1.2$. A time step $\Delta t$ is added to the diagonal elements of the approximate Jacobian matrix, thus increasing the diagonal dominance of the matrix.
Chapter 3. Algorithm Formulation

With the time step, Eq. 3.8 becomes:

$$\left( R^{(n)} \cdot \frac{I}{h^{(n)}} + J_E^{(n)} \right) \Delta Q^{(n)} = -R(Q^{(n)})$$

(3.25)

The time step is calculated locally at each cell from:

$$h_i = CFL \cdot \frac{\Omega_i}{\sum_{k=1}^{n_t} \lambda_{ik}}$$

(3.26)

where the Courant-Friedrich-Lewy (CFL) number is a user-defined value. It is incorporated into both the exact Jacobian $J_E$ and the approximate preconditioner Jacobian $J_A$, but is not included for boundary cell faces. The $I/h$ term of Eq. 3.25 is multiplied by the density residual $R^{(n)}$ of Eq. 3.11, which acts as a scaling coefficient. As the $L_2$-norm of the density residual tends to zero, so does the scaling coefficient, and the term $I/h$ tends to zero. In the limit, as $\|R^{(n)}\|_2$ approaches zero, Eq. 3.25 approaches Eq. 3.8.
Chapter 4

Results

This chapter presents the results of tests to verify and optimise the performance of the Newton-Krylov solver.

The tests were conducted on a suite of eight cases, involving three subsonic and four transonic flows around the single-element NACA 0012 airfoil, and one subsonic flow about a multi-element high-lift airfoil.

<table>
<thead>
<tr>
<th>Airfoil</th>
<th>Flow regime</th>
<th>$M_\infty$</th>
<th>$\alpha$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-element</td>
<td>Subsonic</td>
<td>0.63</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>Transonic</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.729</td>
<td>2.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>Multi-element</td>
<td>Subsonic lifting</td>
<td>2.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 4.1: Suite of test cases.

The three cases shown in bold type in Table 4.1, one for each flow regime, were taken as primary validation cases. They were tested using a set of base parameters deter-
determined to be optimum for each one. The five remaining cases of the suite were tested in less detail with the set of base parameters used for the primary validation case of their flow regime.

The sets of base parameters include the fourth-difference dissipation control constant, $\sigma$ (Subsection 3.4.1); the artificial dissipation control constants $\kappa_2$ and $\kappa_4$ (Section 2.3); the tolerance $G$ applied to the GMRES residual, $r_m$ (Section 3.3); the level of ILU fill-in, $p$ (Section 3.4); and the tolerance $P$ applied to the density residual $R$ (Section 3.2) for freezing the preconditioner matrix (Section 3.4). For the transonic cases, a local time step, $h$ (Section 3.6) was applied using a CFL value determined to be optimum.

Results for the primary validation cases include Mach number contours, pressure coefficient ($C_p$) plots, and force and moment coefficients, to verify the performance of the solver. These results are complemented with convergence histories for tests that compare the performance of the solver when one of the base parameter values is varied. Results for the remaining cases of the suite only show the convergence history for a test run with the corresponding base parameters.

Convergence histories are reported in terms of the CPU time required for the density residual $R$ to converge to $10^{-10}$. For these results to be independent of the computer or compiler used, the CPU time is normalised by the cost of one evaluation of the RHS. This cost includes all flux Jacobian evaluations, the pressure field update, and the calculation of artificial dissipation. The convergence histories are complemented with graphs showing the number of GMRES iterations per Newton iteration.

All tests were run on a Compaq Alpha ES40 server with 667 MHz Alpha 21264A processors.

### 4.1 Single-element Airfoil Cases

The single-element airfoil cases were run on the 3,631 and 14,193-node triangular grids shown in Fig. 4.1, with the optimum base parameters indicated in Table 4.2. Unless specifically noted, all of the single-element airfoil test results reported are for the 14,193-node grid.
Section 4.1. Single-element Airfoil Cases

4.1. Subsonic Flow

The primary validation case for subsonic flow over a single-element airfoil is a case with \( M_\infty = 0.63 \) and \( \alpha = 2.0^\circ \).

The Mach number contours and \( C_p \) plot for this case are shown in Fig. 4.2. As expected for an inviscid flow without shocks, the figures show a smooth flow field. The force and moment coefficients of Table 4.3 agree well with published results. In particular, the drag coefficient \( (C_D) \), which should be zero, is sufficiently small to be attributed to numerical error. Such error may be due to the use of zeroth instead of first-order extrapolation at the airfoil boundaries.

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{Flow Regime} & M_\infty & \alpha^{(\circ)} & \sigma & \kappa_2 & \kappa_4 & CFL & G & p \\
\hline
\text{Subsonic} & 0.63 & 2.0 & 5 & 0.0 & 0.3 & - & 10^{-1} & 5 & 10^{-1} \\
\text{Transonic} & 0.8 & 1.0 & 4 & 20.0 & 0.3 & 100 & 10^{-1} & 7 & 10^{-2} \\
\hline
\end{array}
\]

Table 4.2: Base parameters for single-element airfoil cases.

\[
C_L \quad C_D \quad C_M
\]

\[
0.3215 \quad 0.0027 \quad 0.0826
\]

Table 4.3: Drag, lift and moment coefficients \( (M_\infty = 0.63, \alpha = 2.0^\circ) \).
Figure 4.2: Mach number contours and $C_p$ plot ($M_{\infty} = 0.63$, $\alpha = 2.0^\circ$).

Fig. 4.3 shows the RHS evaluations for this primary validation case run with the optimum base parameters. Each symbol represents a Newton iteration. On the 3,631-node grid, this case ran in approximately 20 s, with 0.12 s of CPU time spent per RHS evaluation, and an average of 9.6 GMRES iterations per Newton iteration. On the 14,193-node grid, the case ran in approximately 2448 s, with 0.53 s of CPU time spent per RHS evaluation, and an average of 15.1 GMRES iterations per Newton iteration.

The choice of optimum values for the base parameters of this primary validation

---

Figure 4.3: Convergence history using base parameters ($M_{\infty} = 0.63$, $\alpha = 2.0^\circ$).
Section 4.1. Single-element Airfoil Cases

Figure 4.4: Comparison of $\sigma$ values ($M_{\infty} = 0.63$, $\alpha = 2.0^\circ$).

The case is corroborated by the results of Figs. 4.4 to 4.7.

Fig. 4.4 compares solver performance when the fourth-difference dissipation control constant is varied. Optimum results are obtained when $\sigma = 5$. For $\sigma = 1$, the number of RHS evaluations more than doubles, and for each of the other values of $\sigma$, except $\sigma = 7$, the difference in the number of RHS evaluations is practically indistinguishable. The first four values of $\sigma$ required $\kappa_4$ to be raised in the set of base parameters from 0.3 to 0.4 for $\sigma = 2, 3, 4$, and to 0.6 for $\sigma = 1$.

Fig. 4.5 compares solver performance when the orders of magnitude that the GMRES residual is reduced at each Newton iteration are varied. The results confirm that it is counterproductive to reduce the GMRES residual by more than 1 order of magnitude. The cases in which the GMRES residual was reduced by 2 and 3 orders of magnitude required a local time step with $CFL = 100$, so, although not required, for consistency the 1-order case was also run with this time step. Therefore, a true performance comparison is one made between the more than 3,000 RHS evaluations taken by the 3-order reduction case of Fig. 4.5, and the 320 RHS evaluations of the 14,193-node case of Fig. 4.3.

Fig. 4.6 compares solver performance when the fill-in level of the preconditioner matrix is varied. The factorisation of $ILU(0)$ is too incomplete, leading to a very large number of RHS evaluations. An $ILU(1)$ factorisation offers a significant improvement, but the best results are obtained for $ILU(4)$ and above, for which the differences in the number of RHS evaluations are almost indiscernible. An $ILU(5)$ factorisation was chosen.
Chapter 4. Results

Figure 4.5: Comparison of GMRES tolerances, $G \left( M_\infty = 0.63, \alpha = 2.0^\circ \right)$.

Figure 4.6: Comparison of $ILU(p)$ fill levels ($M_\infty = 0.63, \alpha = 2.0^\circ$).

as optimum, because it offers a slight advantage in the number of RHS evaluations, and can be run with the $\kappa_4$ value of the base parameters, whereas $ILU(6)$ and $ILU(7)$ require $\kappa_4 = 0.4$.

Fig. 4.7 compares solver performance when the tolerance applied to the density residual for freezing the preconditioner matrix is varied. Optimum results are obtained when the preconditioner matrix is frozen as soon as the density residual is equal to or less than $10^{-1}$.

Application of the base parameters to the cases of $M_\infty = 0.3; \alpha = 5.0^\circ$, and $M_\infty = 0.5; \alpha = 3.0^\circ$ yields the results of Fig. 4.8. The convergence history for the $M_\infty = 0.5$;


Section 4.1. Single-element Airfoil Cases

4.1. Single-element Airfoil Cases

Figure 4.7: Comparison of preconditioner freezing tolerances, \( P (M\infty = 0.63, \alpha = 2.0^\circ) \).

Figure 4.8: Other single-element subsonic cases.

\( \alpha = 3.0^\circ \) case is very similar to that shown in Fig. 4.3 for the primary validation case. However, as expected, the \( M\infty = 0.3; \alpha = 5.0^\circ \) case requires more RHS iterations given the combination of lower freestream Mach number and higher angle of attack.

4.1.2 Transonic Flow

The primary validation case for transonic flow over a single-element airfoil is a case with \( M\infty = 0.8 \) and \( \alpha = 1.0^\circ \).

The Mach number contours and \( C_p \) plot for this case are shown in Fig. 4.9. As expected for this case, they show the formation of a shock on the upper surface of the airfoil. The force and moment coefficients of Table 4.4 agree well with published results. Given that the present solver considers the flow to be inviscid, the fact that the value of \( C_D \) is
Figure 4.9: Mach number contours and $C_p$ plot ($M_\infty = 0.8, \alpha = 1.0^\circ$).

<table>
<thead>
<tr>
<th>$C_L$</th>
<th>$C_D$</th>
<th>$C_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2443</td>
<td>0.0255</td>
<td>0.0843</td>
</tr>
</tbody>
</table>

Table 4.4: Drag, lift and moment coefficients ($M_\infty = 0.8, \alpha = 1.0^\circ$).

approximately ten times greater than for the subsonic validation case is attributed solely to the pressure discontinuity across the shock.

Fig. 4.10 shows the RHS evaluations for this primary validation case run with the optimum base parameters. Each symbol represents a Newton iteration. On the 3,631-node grid, this case ran in approximately 42 s, with 0.1 s of CPU time spent per RHS evaluation, and an average of 6 GMRES iterations per Newton iteration. On the 14,193-node grid, the case ran in approximately 7 m 9 s, with 0.45 s of CPU time spent per RHS evaluation, and an average of 8.3 GMRES iterations per Newton iteration. The appearance of “spikes” during convergence is attributable to a poor initial guess $\hat{Q}^{(0)}$ that tends to mislead the solver, especially during the first Newton iterations.

The choice of optimum values for the base parameters of this primary validation case is corroborated by the results of Figs. 4.11 to 4.14.

Fig. 4.11 compares solver performance when the fourth-difference dissipation con-
Section 4.1. Single-element Airfoil Cases

Figure 4.10: Convergence history using base parameters ($M_\infty = 0.8, \alpha = 1.0^\circ$).

Figure 4.11: Comparison of $\sigma$ values ($M_\infty = 0.8, \alpha = 1.0^\circ$).

trol constant is varied. Optimum results are obtained when $\sigma = 4$. The case with $\sigma = 2$
did not converge.

Fig. 4.12 compares solver performance when the orders of magnitude that the
GMRES residual is reduced at each Newton iteration are varied. Again, as with the subsonic
validation case, the results confirm that it is counterproductive to reduce the GMRES
residual by more than 1 order of magnitude at each Newton iteration. The 2 and 3-order
cases were run with CFL values of 30 and 10 respectively.

Fig. 4.13 compares solver performance when the fill-in level of the preconditioner
matrix is varied. The factorisations with fill-in levels below 4 are too incomplete, causing
Figure 4.12: Comparison of GMRES tolerances, $G (M_\infty = 0.8, \alpha = 1.0^\circ)$.

Figure 4.13: Comparison of $ILU(p)$ fill levels ($M_\infty = 0.8, \alpha = 1.0^\circ$).

the solver to not converge. The optimum convergence rate is obtained with $ILU(7)$, which is higher than for the single-element subsonic validation case.

Fig. 4.14 compares solver performance when the tolerance applied to the density residual for freezing the preconditioner matrix is varied. Optimum results are obtained if the preconditioner is frozen as soon as the density residual is equal to or smaller than $10^{-2}$. Freezing the preconditioner below this tolerance has no effect on the number of GMRES iterations per Newton iteration.

Application of the base parameters to the cases of $M_\infty = 0.729; \alpha = 2.31^\circ$, $M_\infty = 0.8; \alpha = 1.25^\circ$, and $M_\infty = 0.95; \alpha = 1.0^\circ$ yields the results of Fig. 4.15.
4.2 Multi-element Airfoil Case

The validation case for subsonic lifting flow over a multi-element airfoil is a case with $M_\infty = 0.2$ and $\alpha = 5.0^\circ$. It was run on the 17,894-node triangular grid shown in Fig. 4.16, with the optimum base parameters indicated in Table 4.5.

The Mach number contours and $C_p$ plot for this validation case are shown in

<table>
<thead>
<tr>
<th>Flow Regime</th>
<th>$M_\infty$</th>
<th>$\alpha$ [°]</th>
<th>$\sigma$</th>
<th>$\kappa_2$</th>
<th>$\kappa_4$</th>
<th>$CFL$</th>
<th>$G$</th>
<th>$p$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsonic lifting</td>
<td>0.2</td>
<td>5.0</td>
<td>5</td>
<td>0.0</td>
<td>0.4</td>
<td>-</td>
<td>$10^{-1}$</td>
<td>4</td>
<td>$10^{-2}$</td>
</tr>
</tbody>
</table>

Table 4.5: Base parameters for multi-element high-lift airfoil case.
Figure 4.16: 17,894-node triangular grid about multi-element airfoil.

Figure 4.17: Mach number contours and $C_p$ plot ($M_\infty = 0.2$, $\alpha = 5.0^\circ$).

Fig. 4.17. The force and moment coefficients are shown in Table 4.6.

Fig. 4.18 shows the RHS evaluations for this validation case run with the optimum base parameters. Each symbol represents a Newton iteration. The case ran in approximately
Section 4.2. Multi-element Airfoil Case

<table>
<thead>
<tr>
<th></th>
<th>$C_L$</th>
<th>$C_D$</th>
<th>$C_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leading-edge slat</td>
<td>0.2904</td>
<td>0.1859</td>
<td>0.3694</td>
</tr>
<tr>
<td>Wing section</td>
<td>1.138</td>
<td>0.4261</td>
<td>0.8490</td>
</tr>
<tr>
<td>Trailing-edge flap 1</td>
<td>3.755</td>
<td>-0.0854</td>
<td>-0.3930</td>
</tr>
<tr>
<td>Trailing-edge flap 2</td>
<td>0.5003</td>
<td>-0.4313</td>
<td>-0.2753</td>
</tr>
</tbody>
</table>

Table 4.6: Drag, lift and moment coefficients ($M_\infty = 0.2, \alpha = 5.0^\circ$).

Figure 4.18: Convergence history using base parameters ($M_\infty = 0.2, \alpha = 5.0^\circ$).

11 m 49 s, with 0.82 s of CPU time spent per RHS evaluation, and an average of 37.2 GMRES iterations per Newton iteration. These values are higher than for the subsonic and transonic single-element airfoil cases. This is to be expected given the greater complexity of the aerodynamic configuration.

The choice of optimum values for the base parameters of this validation case is corroborated by the results of Figs. 4.19 to 4.22.

Fig. 4.19 compares solver performance when the fourth-difference dissipation control constant is varied. As with the single-element validation cases, optimum results are obtained when $\sigma = 5$. For this case, however, the solver does not converge for any value of $\sigma$ lower than this.

Fig. 4.20 compares solver performance when the orders of magnitude that the GMRES residual is reduced at each Newton iteration are varied. The results confirm that it is counterproductive to reduce the GMRES residual by more than 1 order of magnitude. The cases in which the GMRES residual was reduced by 2 and 3 orders of magnitude
Chapter 4. Results

Figure 4.19: Comparison of $\sigma$ values ($M_\infty = 0.2, \alpha = 5.0^\circ$).

Figure 4.20: Comparison of GMRES tolerances, $G$ ($M_\infty = 0.2, \alpha = 5.0^\circ$).

required a local time step with $CFL = 100$, so, although not required, for consistency the
1-order case was also run with this time step. Therefore, a true performance comparison
is one made between the more than 5,000 RHS evaluations taken by the 3-order reduction
case of Fig. 4.20, and the 700 RHS evaluations of Fig. 4.18.

Fig. 4.21 compares solver performance when the fill-in level of the preconditioner
matrix is varied. As in the previous validation cases, the factorisation of $ILU(0)$ is too
incomplete, leading to a very large number of RHS evaluations. An $ILU(4)$ factorisation
Section 4.2. Multi-element Airfoil Case

Figure 4.21: Comparison of $ILU(p)$ fill levels ($M_\infty = 0.2, \alpha = 5.0^\circ$).

Figure 4.22: Comparison of preconditioner freezing tolerances, $P$ ($M_\infty = 0.2, \alpha = 5.0^\circ$).

offers optimum results. The solver does not converge for higher levels of fill-in.

Fig. 4.22 compares solver performance when the tolerance applied to the density residual for freezing the preconditioner matrix is varied. Optimum results are obtained when the preconditioner matrix is frozen as soon as the density residual is equal to or less than $10^{-2}$. Recalculating the preconditioner matrix for further Newton iterations produces no benefit.
Chapter 5

Conclusions

A node-based finite-volume Newton-Krylov solver has been implemented to solve the Euler equations on arbitrary polygonal unstructured grids about aerodynamic configurations. The solver applies an inexact Newton method to solve the non-linear system of equations that results from the spatial discretisation of the Euler equations. A matrix-free GMRES strategy is used to solve the linear system arising at each Newton iteration. This system is right-preconditioned using an approximate Jacobian matrix subjected to ILU factorisation and RCM reordering.

The solver has been tested with a single-element airfoil in the subsonic and transonic flow regimes, and with a multi-element airfoil in a subsonic high-lift case. The tests successfully validate the functioning of the solver and investigate the effect on convergence of varying grid size, scalar artificial dissipation parameters, preconditioner fill-in, and GMRES and preconditioner-freezing tolerances. Sets of base parameters that optimise the performance of the solver have been determined.

The solver is ready to be extended from the Euler equations to the full Navier-Stokes equations; however, some of its present aspects are worthy of further attention. A full linearisation of artificial dissipation terms and an extension from zeroth to first-order extrapolation at boundaries could be applied to determine if an improved preconditioner can be obtained without incurring excessive speed and memory penalties. Matrix dissipation, instead of the current scalar dissipation, could be applied in an attempt to lower the values of the second and fourth-difference dissipation terms, which might improve accuracy. Different levels of GMRES restart could be tested to determine their impact on solver performance. Finally, start-up strategies, such as mesh sequencing, could be attempted to provide the
solver with an improved initial guess vector, which would improve the speed of the solver, especially in subsonic lifting cases and transonic flow cases with shocks.
Appendix A

Algorithm Flow Diagram

1. Propose initial guess
2. Update solution vector: $\Delta \hat{Q}^{(m)} = \hat{Q}^{(m)} + \Delta \hat{Q}^{(0)}$
3. Advance one Newton iteration
   - Form $J^{(m)}_e$ matrix ([ILU(n) & RCM] to precondition
   - $J^{(m)}_e \Delta \hat{Q}^{(m)} = -R\left(\hat{Q}^{(m)}\right)$
   - $M = J^{(m)}_{A_{precond}}$
   - Apply GMRES to solve for $\Delta \hat{Q}^{(m)}$ in
     $(J^{(m)}_e M^{-1}) (M \Delta \hat{Q}^{(m)}) = -R\left(\hat{Q}^{(m)}\right)$
4. Advance one GMRES iteration
   - Is $f_m \leq$ Tolerance G?
5. Update solution vector: $\hat{Q}^{(m+1)} = \hat{Q}^{(m)} + \Delta \hat{Q}^{(m)}$
6. Check convergence: $R^{(m)} \leq$ Tolerance P?
7. If YES, stop. If NO, check convergence: $R^{(m)} \leq 10^{-10}$?
8. If YES, stop. If NO, return to step 2.

Stop
Appendix A. Algorithm Flow Diagram
Appendix B

Examples of the Jacobians for $J_A$

**Interior Cells**

For interior cells $i$ and $k$, the flux Jacobians of Eqs. 3.20 and 3.21 are of the form:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \frac{\partial f_1}{\partial q_3} & \frac{\partial f_1}{\partial q_4} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \frac{\partial f_2}{\partial q_3} & \frac{\partial f_2}{\partial q_4} \\ \frac{\partial f_3}{\partial q_1} & \frac{\partial f_3}{\partial q_2} & \frac{\partial f_3}{\partial q_3} & \frac{\partial f_3}{\partial q_4} \\ \frac{\partial f_4}{\partial q_1} & \frac{\partial f_4}{\partial q_2} & \frac{\partial f_4}{\partial q_3} & \frac{\partial f_4}{\partial q_4} \end{bmatrix}$$

(B.1)

From Eqs. 2.2, 2.5, and 2.3, there is:

$$p = (\gamma - 1) \left[ q_4 - \frac{1}{2q_1} (q_2^2 + q_3^2) \right]$$

(B.2)

and

$$\mathbf{F}(\mathbf{Q}) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \frac{q_2}{q_1} \\ \frac{q_2}{q_1} + (\gamma - 1) \left[ q_4 - \frac{1}{2q_1} (q_2^2 + q_3^2) \right] \\ \frac{q_3}{q_1} \left( q_4 + (\gamma - 1) \left[ q_4 - \frac{1}{2q_1} (q_2^2 + q_3^2) \right] \right) \\ \frac{q_3}{q_1} + (\gamma - 1) \left[ q_4 - \frac{1}{2q_1} (q_2^2 + q_3^2) \right] \end{bmatrix}$$

(B.3)
Therefore, the flux Jacobians are:

\[
\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix}
0 & \frac{1}{q_1} \left[ -q_2^2 + \frac{\gamma - 1}{2} (q_2^2 + q_3^2) \right] & \frac{1}{q_1} \left[ \gamma q_4 + \frac{\gamma - 1}{2q_1} (3q_2^2 - q_3^2) \right] \\
\frac{1}{q_1} \left[ -q_2 q_4 + (\gamma - 1) q_2 \left[ -q_4 + \frac{1}{q_1} (q_2^2 + q_3^2) \right] \right] & \frac{1}{q_1} \left( \gamma q_4 + \frac{\gamma - 1}{2q_1} (3q_2^2 - q_3^2) \right) & 0 \\
\frac{1}{q_1} \frac{q_1}{q_1} & \frac{q_1}{q_1} & \frac{q_1}{q_1} \\
0 & \frac{1}{q_1} \frac{q_1}{q_1} & \frac{1}{q_1} \frac{q_1}{q_1} \\
\frac{1}{q_1} \left[ -q_2 q_4 + (\gamma - 1) q_2 \left[ -q_4 + \frac{1}{q_1} (q_2^2 + q_3^2) \right] \right] & \frac{1}{q_1} \left( \gamma q_4 + \frac{\gamma - 1}{2q_1} (3q_2^2 - q_3^2) \right) & 0 \\
-\frac{\gamma - 1}{q_1} & \frac{\gamma - 1}{q_1} & 0 \\
\frac{\gamma - 1}{q_1} & \frac{\gamma - 1}{q_1} & \frac{\gamma - 1}{q_1} \\
0 & 0 & \frac{1}{q_1} \left[ q_4 - (\gamma - 1) \frac{q_1}{q_1} \right] \\
\end{bmatrix}
\]

(B.4)

Boundary Faces

For boundary faces \( m \), analogously, the flux Jacobians of Eqs. 3.20 and 3.21 are of the form:

\[
\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix}
\frac{\partial f_1}{\partial \theta_0} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \theta_3} \\
\frac{\partial f_2}{\partial \theta_0} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \theta_3} \\
\frac{\partial f_3}{\partial \theta_0} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \theta_3} \\
\frac{\partial f_4}{\partial \theta_0} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \theta_3} \\
\end{bmatrix}
\]

(B.5)

For the body surface boundary case of Subsection 2.4.1, the flux tensor of Eq. 2.3 is expressed in terms of \( \mathbf{Q}_b \) of Eq. 2.26, where the pressure \( p \) and the tangential velocity component \( V_t \) are extrapolated values expressed in terms of the variables \( q_{e1}, q_{e2}, q_{e3}, \) and \( q_{e4} \). Similarly, for each of the four far-field boundary cases of Subsection 2.4.2, the flux tensor of Eq. 2.3 is expressed in terms of the corresponding \( \mathbf{Q}_b \) vector; the example vector of Eq. 2.35 corresponds to the subsonic inflow case. Depending on the case, \( a, \rho, \rho, V_n, \) and \( V_t \) are extrapolated values, expressed in terms of the variables \( q_{e1}, q_{e2}, q_{e3}, \) and \( q_{e4} \).
The \( \mathbf{Q} \) vector Jacobians of Eq. 3.20 are of the form:

\[
\frac{\partial \mathbf{Q}_b}{\partial \mathbf{Q}_e} = \begin{bmatrix}
\frac{\partial q_{b1}}{\partial q_{e1}} & \frac{\partial q_{b1}}{\partial q_{e2}} & \frac{\partial q_{b1}}{\partial q_{e3}} & \frac{\partial q_{b1}}{\partial q_{e4}} \\
\frac{\partial q_{b2}}{\partial q_{e1}} & \frac{\partial q_{b2}}{\partial q_{e2}} & \frac{\partial q_{b2}}{\partial q_{e3}} & \frac{\partial q_{b2}}{\partial q_{e4}} \\
\frac{\partial q_{b3}}{\partial q_{e1}} & \frac{\partial q_{b3}}{\partial q_{e2}} & \frac{\partial q_{b3}}{\partial q_{e3}} & \frac{\partial q_{b3}}{\partial q_{e4}} \\
\frac{\partial q_{b4}}{\partial q_{e1}} & \frac{\partial q_{b4}}{\partial q_{e2}} & \frac{\partial q_{b4}}{\partial q_{e3}} & \frac{\partial q_{b4}}{\partial q_{e4}}
\end{bmatrix}
\]

(B.6)

There are five such Jacobians: one for the body surface boundary \( \mathbf{Q}_b \) vector of Eq. 2.26, and four for the far-field boundary \( \mathbf{Q}_b \) vectors, of which Eq. 2.35 is the first. The Jacobian for the supersonic inflow case is null. As an example, for the body surface boundary case:

\[
\frac{\partial \mathbf{Q}_b}{\partial \mathbf{Q}_e} = \begin{bmatrix}
\frac{1}{2} \frac{\gamma (q_{e2}^2 + q_{e3}^2)}{q_{e1}^2 (H_{\infty} - \frac{1}{2} V_1^2)} & -\frac{7 q_{e2}}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{\gamma (q_{e2}^2 + q_{e3}^2)}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e2} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{1}{2} \frac{q_{e2}^2 (H_{\infty} - \frac{1}{2} V_1^2)}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & -\frac{7 q_{e2} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{1}{2} \frac{q_{e2}^2 (H_{\infty} - \frac{1}{2} V_1^2)}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e2} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{1}{2} \frac{q_{e2}^2 (H_{\infty} - \frac{1}{2} V_1^2)}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e2} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{7 q_{e3}}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e3} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{7 q_{e3} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e3} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{7 q_{e3} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & \frac{7 q_{e3} V_1}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} \\
\frac{7 q_{e3} V_1^2}{q_{e1} (H_{\infty} - \frac{1}{2} V_1^2)} & 1 + \frac{1}{2} \frac{V_1^2}{(H_{\infty} - \frac{1}{2} V_1^2)}
\end{bmatrix}
\]

(B.7)

These Jacobians are calculated using the Maple [29] symbolic mathematical computation system. As an example, the code for generating the C code for the Jacobian of Eq. B.7 is:

```maple
with(linalg);
Qc:=vector(4,[qe1,qe2,qe3,qe4]);
Qb:=vector(4,[qb1,qb2,qb3,qb4]);
Vtangent:=(qe2*TangentX+qe3*TangentY)/qe1;
P:=(Gamma-1)*(qe4-0.5*((qe2*qe2+qe3*qe3)/qe1));
qb1:=Gamma*P/(Gamma-1.0)/(Hinf-0.5*Vtangent*Vtangent);
qb2:=qb1*Vtangent*TangentX;
qb3:=qb1*Vtangent*TangentY;
qb4:=P/(Gamma-1.0)+0.5*qb1*Vtangent*Vtangent;
J:=jacobian(Qb,Qc);
```
Appendix B. Examples of the Jacobians for $J_A$

```maple
readlib(C);
C(J.optimized);
```
References


### References


