Adaptation for Vortex Flows using a 3-D Finite Element Solver

by

Alexandra Maria Landsberg

S.B. Massachusetts Institute of Technology (1988)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

November 1990

©1990, Massachusetts Institute of Technology

Signature of Author ____________________________

Department of Aeronautics and Astronautics

November 30, 1990

Certified by ________________________________

Professor Earll M. Murman

Thesis Supervisor, Department of Aeronautics and Astronautics

Accepted by ________________________________

Professor Harold Y. Wachman

Chairman, Department Graduate Committee
Adaptation for Vortex Flows using a 3-D Finite Element Solver

by
Alexandra Maria Landsberg

Submitted to the Department of Aeronautics and Astronautics
on November 30, 1990
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics

In order to save on computational costs and increase solution accuracy, one would like to refine a computational grid only in the regions that have "interesting" flow features. In our case, vortical flows are of interest. This thesis determines what flow parameters best characterize vortical flows and then uses criteria based on the flow parameters to mark the vortical flow region for adaptation. A method to automatically set the high and low adaptation limits is specified and the effectiveness of the adaptation is then evaluated by total pressure loss errors in the vortex core. This work is built upon Dr. Richard Shapiro's PhD thesis which introduced a three dimensional finite element algorithm.

Total pressure loss and normalized helicity are used as adaptation indicators and their effectiveness is evaluated. All test cases run are variations on vortex flows propagating at $M_\infty = 1.5$ down a rectangular domain with sides modelled as far-field boundaries. Two vortex core structures are used: a Rankine vortex and a Lamb vortex. The results from the two test cases are discussed: a single vortex and two vortices of unequal strengths with opposite rotation directions.

Thesis Supervisor: Earll M. Murman,
Professor of Aeronautics and Astronautics

2
Acknowledgments

MIT can be a very stressful place at times, but I’ve been fortunate enough to have two very special friends (delinquents), Gerd and Guppy, to always come to my rescue. I cannot thank them enough for their friendship, support and never-ending playfulness, and most especially for taking me climbing even when I sometimes should have been working. They’ve made this past year amazingly fun. For over 10 years now my closest friend, Jung, has always been there for both the good times and the bad. Her friendship is invaluable, what more can I say.

I would like to especially thank my advisor, Earl Murman, for always being supportive and friendly and always giving valuable advice at the most crucial times. I truly enjoyed working with him.

This past year the CFD lab has been a great place to work (and live at times) due to the people. Dana, Dave #1, Dave #2, and Phil #1 have given me technical help more times than I can recall and are just terrific friends. I tend to be a night person which might bring to mind working alone late at night. This wasn’t exactly the case, Eric, Kousuke, Phil #2, and Mark seemed to be around as much as me. They’ve made working nights not only productive but fun. I would like to thank the rest of the CFD lab, Mike, Andre, Helene, Tom, Ling, and Missy, for their friendship. Lastly, I can not forget to thank Bob for being a unique individual. I always got a laugh out of being around Bob, even when I was the victim of his jokes.

Outside of lab, Harold, Dan, Peter, Jean, and Jerry are good friends and always made things a little more tolerable.

Finally, I’d like to thank my mom, a person whom I respect and care about very much, and my brother, Paul, and my sister-in-law, Kim, whose e-mail messages kept me laughing through the day. Thank you all.

This research was supported by the Air Force Office of Scientific Research under grant AFOSR89-0395A monitored by Dr. L. Sakell.
Contents

Abstract 2

Acknowledgments 3

1 Introduction 14

1.1 Background ...................................... 14

1.2 Overview of thesis ................................ 17

2 Governing Equations 21

2.1 Euler Equations .................................. 21

2.2 Non-Dimensionalization .......................... 22

2.3 Auxiliary Quantities .............................. 23

2.4 Physical Boundary Conditions .................. 24

3 3-D Finite Element Concepts 27

3.1 Terminology ...................................... 27

3.2 Finite Element Discretization .................. 29

3.2.1 Interpolation Functions ....................... 30
6.1 Vortex Models .................................. 59

6.2 Vortex Test Cases .............................. 62

6.2.1 Single Vortex .................................. 63

6.2.2 Two Unequal Vortices with Opposite Rotation Directions ... 92

6.3 Summary ......................................... 98

7 Conclusions ....................................... 115

Bibliography ........................................ 119

A Solid Wall Boundary Conditions ............................. 122

A.1 Physical Boundary Condition .......................... 122

A.2 Implementation .................................... 122
List of Figures

1.1 Total Pressure Loss Contour at $z = 0.75$ for the NTF Delta Wing .... 18
1.2 Total Pressure Loss Contour at $z = 1.00$ for the NTF Delta Wing .... 18
1.3 Total Pressure Loss Contour at $z = 1.20$ for the NTF Delta Wing .... 19
1.4 Total Pressure Loss Contour at $z = 1.50$ for the NTF Delta Wing .... 19
1.5 Total Pressure Loss Contour at $z = 2.00$ for the NTF Delta Wing .... 20
1.6 Total Pressure Loss along the Primary Vortex Core for the NTF Delta Wing ........................................... 20

3.1 3-D Finite Element ........................................... 28
3.2 3-D Degenerate Finite Element ................................ 28
3.3 Natural Coordinates .......................................... 29
3.4 Geometry of Three-Dimensional Element ........................ 32

4.1 Three-Dimensional Weights for Second Difference at Node 1 ....... 42

5.1 Cutaway View of Three-Dimensional Interface ........................ 48
5.2 Example of Mesh Adaptation ..................................... 49
5.3 Total Pressure Loss through a Lamb Vortex Core .................... 55
5.4 Normalized Helicity through a Lamb Vortex Core .................................. 56

6.1 Tangential Velocity Distribution for a Rankine Vortex .......................... 70
6.2 Tangential Velocity Distribution for a Lamb Vortex ............................. 70
6.3 Entropy Distribution for a Rankine Vortex ........................................ 71
6.4 Entropy Distribution for a Lamb Vortex ........................................... 71
6.5 Single Vortex - Initial Grid, $y - z$ Slice ........................................ 72
6.6 Single Vortex - Initial Grid, $z - z$ Slice ........................................ 72
6.7 Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Initial Grid ............ 73
6.8 Single Vortex - $\Delta P_0$ Contours at $z = 0.5$, Initial Grid ............. 73
6.9 Single Vortex - $\Delta P_0$ Contours at $z = 1.0$, Initial Grid ............. 74
6.10 Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Initial Grid .......... 74
6.11 Single Vortex - $H_n$ Contours at $z = 0.0$, Initial Grid ................ 75
6.12 Single Vortex - $H_n$ Contours at $z = 0.5$, Initial Grid ................ 75
6.13 Single Vortex - $H_n$ Contours at $z = 1.0$, Initial Grid ................ 76
6.14 Single Vortex - $H_n$ Contours at $z = 2.0$, Initial Grid ................ 76
6.15 Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at $z = 0.0$ 77
6.16 Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at $z = 1.0$ 77
6.17 Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at $z = 2.0$ 78
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.18</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (1 level) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.19</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 0.5$, Adaptive Grid (1 level) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.20</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 1.0$, Adaptive Grid (1 level) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.21</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (1 level) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.22</td>
<td>Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at $z = 0.0$</td>
</tr>
<tr>
<td>6.23</td>
<td>Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at $z = 1.0$</td>
</tr>
<tr>
<td>6.24</td>
<td>Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at $z = 2.0$</td>
</tr>
<tr>
<td>6.25</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (2 levels) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.26</td>
<td>Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (2 levels) Based on $\Delta P_0$</td>
</tr>
<tr>
<td>6.27</td>
<td>Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y - z$ Slice at $z = 0.0$</td>
</tr>
<tr>
<td>6.28</td>
<td>Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y - z$ Slice at $z = 1.0$</td>
</tr>
<tr>
<td>6.29</td>
<td>Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y - z$ Slice at $z = 2.0$</td>
</tr>
<tr>
<td>6.30</td>
<td>Single Vortex - $H_n$ Contours at $z = 0.0$, Adaptive Grid (1 level) Based on $H_n$</td>
</tr>
</tbody>
</table>
6.31 Single Vortex - $H_n$ Contours at $z = 1.0$, Adaptive Grid (1 level) Based on $H_n$ ......................................... 86

6.32 Single Vortex - $H_n$ Contours at $z = 2.0$, Adaptive Grid (1 level) Based on $H_n$ ......................................... 87

6.33 Single Vortex - Adaptive Grid (2 levels) based on $H_n$, $y-z$ Slice at $z = 0.0$ 88

6.34 Single Vortex - Adaptive Grid (2 levels) based on $H_n$, $y-z$ Slice at $z = 2.0$ 88

6.35 Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (2 levels) Based on $H_n$ ......................................... 89

6.36 Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (2 levels) Based on $H_n$ ......................................... 89

6.37 Single Vortex - Convergence Histories for Adaptive Cases ............... 90

6.38 Single Vortex - $\Delta P_0$ along Vortex Core for Adaptive Cases ............... 90

6.39 Single Vortex - Convergence Histories for Varying $\nu_4$ Cases ............... 91

6.40 Single Vortex - $\Delta P_0$ along Vortex Core for Varying $\nu_4$ Cases ............... 91

6.41 Total Pressure Loss Contour at $z = 1.20$ for the NTF Delta Wing ....... 92

6.42 Unequal Vortices - Initial Grid, $y-z$ Slice ............... 101

6.43 Unequal Vortices - Initial Grid, $z-z$ Slice ............... 101

6.44 Unequal Vortices - $\Delta P_0$ Contours at Inlet, Initial Grid ............... 102

6.45 Unequal Vortices - $\Delta P_0$ Contours at Outlet, Initial Grid ............... 102

6.46 Unequal Vortices - $H_n$ Contours at Inlet, Initial Grid ............... 103

6.47 Unequal Vortices - $H_n$ Contours at Outlet, Initial Grid ............... 103
6.48 Unequal Vortices - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at 
  $z = 0.0$ .................................................... 104

6.49 Unequal Vortices - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at 
  Outlet .......................................................... 104

6.50 Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (1 level) Based 
  on $\Delta P_0$ .................................................... 105

6.51 Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (1 level) 
  Based on $\Delta P_0$ .................................................... 105

6.52 Unequal Vortices - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at 
  Inlet .......................................................... 106

6.53 Unequal Vortices - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at 
  Outlet .......................................................... 106

6.54 Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (2 levels) Based 
  on $\Delta P_0$ .................................................... 107

6.55 Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (2 levels) 
  Based on $\Delta P_0$ .................................................... 107

6.56 Unequal Vortices - Adaptive Grid (1 level) based on $H_n$, $y - z$ Slice at Inlet 108

6.57 Unequal Vortices - Adaptive Grid (1 level) based on $H_n$, $y - z$ Slice at 
  Outlet .......................................................... 108

6.58 Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (1 level) Based 
  on $H_n$ .......................................................... 109

6.59 Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (1 level) 
  Based on $H_n$ .................................................... 109
6.60 Unequal Vortices - $H_n$ Contours at Inlet, Adaptive Grid (1 level) Based
\[ H_n \]

6.61 Unequal Vortices - $H_n$ Contours at Outlet, Adaptive Grid (1 level) Based
\[ H_n \]

6.62 Unequal Vortices - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at Inlet
\[ H_n \]

6.63 Unequal Vortices - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at Outlet

6.64 Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (2 levels) Based
\[ H_n \]

6.65 Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (2 levels)
Based on $H_n$

6.66 Unequal Vortices - Convergence Histories

6.67 Unequal Vortices - $\Delta P_0$ along Stronger Vortex Core

6.68 Unequal Vortices - $\Delta P_0$ along Weaker Vortex Core
List of Tables

2.1 Scaling Factors for Non-Dimensionalization ................................................. 23

3.1 Nodes for Each Face, Trilinear Element ..................................................... 32

6.1 Single Vortex - Comparison of $\Delta P_0$ Errors at Stations along Domain .... 68

6.2 Single Vortex - Comparison of $\Delta P_0$ Errors at Stations along Domain for Varying $\nu_4$ .......................................................... 68

6.3 Unequal Vortices - Comparison of $\Delta P_0$ for Stronger Vortex at Stations along Domain .................................................. 98

6.4 Unequal Vortices - Comparison of $\Delta P_0$ for Weaker Vortex at Stations along Domain .................................................. 98

6.5 Unequal Vortices - Comparison of $\Delta P_0$ Errors for Stronger Vortex at Stations along Domain ........................................... 99

6.6 Unequal Vortices - Comparison of $\Delta P_0$ Errors for Weaker Vortex at Stations along Domain ........................................... 99
The advent of parallel and vector supercomputers has placed computational fluid dynamics in the forefront in the design and analysis of aerodynamic bodies. The ability to compute complex three-dimensional flowfields around all types of aircraft is now possible. With these three-dimensional calculations the grids are usually fine enough to detect the relevant flow features such as shocks and vortices. However, with vortical flows, the truncation errors due to the grid resolution and the effects of artificial viscosity in the flow solver often result in large numerical errors which manifest themselves in the spurious diffusion of vorticity. One method that will reduce the numerical errors is to refine the computational grid around the vortical structures. The process of detecting a flow feature, marking the region to be refined, and performing the refinement or unrefinement of the grid is called grid adaptation. This thesis addresses the use of grid adaptation for vortex flows.

1.1 Background

In order to look at the effects of grid adaptation for vortex flows, an adaptive finite element algorithm for the Euler equations, developed by Dr. Richard Shapiro, was used [27]. The finite element flow solver was implemented in three dimensions using hexahedral elements. The advantage of using a finite element algorithm is that the method is “unstructured” by nature, thus lending itself to adaptive gridding where an unstructured grid is desirable, if not necessary. The advantage of using hexahedral elements is that they offer large CPU and memory reductions over tetrahedral elements. Shapiro [27] discusses three variations of the finite element method: the Galerkin method, the cell-vertex method (this method is the only method implemented in three dimensions and
thus is the method used in this thesis) and the central difference method. For a diverse range of papers on finite elements in fluids, the *Proceedings of the Seventh International Conference of Finite Element Methods in Flow Problems* [5] is recommended.

Grid adaptation can be performed using three primary methods: grid redistribution, grid regeneration and grid enrichment. In the grid redistribution method, the mesh points in an initial computational grid are allowed to move as the solution proceeds. The survey article by Eiseman [10] is a good overview of various grid redistribution schemes. With grid regeneration either the entire computational grid or some portion of it is regenerated when adaptation occurs. This method of adaptation is described in articles by Peraire, et al. [20, 21]. Grid enrichment is a method in which additional nodes and elements are inserted into the grid. The survey article by Berger [4] presents a good overview of grid enrichment methods. The grid adaptation method used in this thesis is grid enrichment or grid embedding. Essentially, the adaptation algorithm divides a hexahedral element, more easily visualized as a cube, into eight smaller cubes.

The focus of grid adaptation has predominantly been upon locating and refining shocks [27, 6]. As mentioned previously, the focus of this thesis is upon adaptation for vortex flows. The need for adaptation for vortex flows is demonstrated with an example from the calculations for the NTF delta wing performed by Becker [3]. The total pressure loss contours over the wing at \( z = 0.75 \) (Fig. 1.1), at the trailing edge \( z = 1.0 \) (Fig. 1.2), and in the wake \( z = 1.2 \) (Fig. 1.3), \( z = 1.5 \) (Fig. 1.4) and \( z = 2.0 \) (Fig. 1.5) show the diffusion of the primary vortex. The secondary vortex experiences an adverse pressure gradient and "bursts" close to the trailing edge. Although the secondary vortex bursts a region of high total pressure loss remains and drifts downstream [8]. Fig. 1.6 shows the total pressure loss along the streamline in the primary vortex core. The primary vortex forms at the leading edge and reaches roughly a constant value of total pressure loss of 0.6 about halfway along the wing. From this point to approximately 120% chord the total pressure loss is about constant. One would expect this behavior to continue for some distance downstream since the physical diffusion mechanisms are small. However, the total pressure loss at the outflow boundary is 0.417. This error is most likely due to the truncation error of the algorithm and the numerical dissipation.
added to the algorithm for stability. It is the non-physical mechanisms of diffusion that can be reduced through grid adaptation.

Grid adaptation entails determining what flow parameter, called the adaptation parameter, best indicates a feature and then using this adaptation parameter to automatically mark the region of interest. Surprisingly, there has been little work in this area for vortical flows. Krist [15] uses a structured embedded grid formulation to resolve the vortical region for a 3-D delta wing. In Powell's ScD thesis [22], a non-adaptive grid embedding scheme was used to resolve the vortical regions for solutions of the conical Euler equations. These are not adaptive schemes since the embedded regions are predetermined by inspection. Although they are approaches to refining the vortical region, they lack the flexibility that is gained by automatic adaptation where the adapted region can expand or decrease as the solution evolves. Powell [23] also developed adaptive methods for leading-edge vortex flows. One of his methods of adaptation used a refinement parameter based on numerics. The refinement parameter is a measure of mesh-convergence, constructed by comparison of locally coarse and fine grid solutions. When a cell has been refined, the solution at the center of one of the sub-cells is compared to the solution at the same point based on the coarse cell. When the refined value on the fine grid is the same as the coarse grid value, the solution is locally mesh-converged. The advantage of using a refinement parameter based numerics is that any region requiring refinement is detected. The other adaptation indicator Powell used was total pressure loss [23]. Batina [2] also chose to use total pressure loss as an adaptation indicator for vortex flows.

This thesis examines two flow parameters as adaptation indicators for vortex flows: total pressure loss and normalized helicity. In addition, the gradients of these parameters are examined to determine if they are better indicators than the quantities alone. Finally, a method by which cells are automatically marked for refinement and unrefinement is desired, this is called automatic thresholding. The automatic thresholding algorithms of Dannenhoffer [6] and Powell [23] are evaluated for the test cases in this thesis and an automatic thresholding algorithm is determined.
1.2 Overview of thesis

First the governing equations for a compressible, inviscid flow (the Euler equations) and the physical boundary conditions are described in Chapter 2. Chapter 3 defines some fundamentals of the three-dimensional finite element method used in this thesis. Chapter 4 presents a complete description of the solution algorithm and discusses many implementation issues. The spatial discretization, implementation of the boundary conditions, smoothing, and time integration scheme are the essential components of the solution algorithm described in this chapter. The requirements for consistency and conservation are also mentioned at the end of this chapter. The focus of this thesis is upon adaptation for vortex flows and in Chapter 5 the adaptation procedure is described. The adaptation criteria for vortex flows is discussed along with a description of the quantities to be adapted upon, a brief discussion of the possible use of gradients of these quantities, and a method to automatically choose the upper and lower limits for cell refinement and unrefinement. Chapter 6 then describes the vortex flow results. First the vortex core models are described. Next, the results from the two test cases are detailed. The test cases are a single vortex and two vortices of unequal strength with opposite rotation directions. Some conclusions of this research are then given in Chapter 7.
Figure 1.1: Total Pressure Loss Contour at $z = 0.75$ for the NTF Delta Wing

Figure 1.2: Total Pressure Loss Contour at $z = 1.00$ for the NTF Delta Wing
Figure 1.3: Total Pressure Loss Contour at $z = 1.20$ for the NTF Delta Wing

Figure 1.4: Total Pressure Loss Contour at $z = 1.50$ for the NTF Delta Wing
Figure 1.5: Total Pressure Loss Contour at $z = 2.00$ for the NTF Delta Wing

Figure 1.6: Total Pressure Loss along the Primary Vortex Core for the NTF Delta Wing (Length = 0 is the leading edge, Length = 1.0 is the trailing edge and Length = 2.0 is the downstream boundary)
Chapter 2
Governing Equations

This chapter describes the governing equations for the flow of a compressible, inviscid, ideal gas including the assumptions on which they are based. The set of equations representing conservation of mass, momentum, and energy describing this flow are termed the Euler equations. Additionally, in this chapter the non-dimensionalization of the equations will be discussed, and the physical boundary conditions will be described.

2.1 Euler Equations

The Euler equations for a three-dimensional flow consist of five partial differential equations representing conservation of mass, conservation of momentum (one for each component of momentum), and conservation of energy. In order for the Euler equations to describe the flow of a compressible, inviscid, ideal gas, the following assumptions are necessary:

- The fluid is a homogeneous continuum;
- There is no viscosity;
- There is no heat transfer (non-conducting);
- The fluid obeys the ideal gas law.

Another assumption made in this thesis is that the body forces are zero. Although the Euler equations are not exact for any real flow, they are a good approximate model for many problems of interest. In this thesis the steady-state solution is sought. In
order to achieve the steady-state solution, a time marching scheme is used, thus the unsteady Euler equations are described here.

The three-dimensional unsteady Euler equations in Cartesian coordinates can be written in vector form as

\[
\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0
\] 

(2.1)

where the state vector \( U \) and the flux vectors \( F, G \) and \( H \) are defined by:

\[
U = \begin{pmatrix} 
\rho \\
\rho u \\
\rho v \\
\rho w \\
\rho E 
\end{pmatrix}, \quad F = \begin{pmatrix} 
\rho u \\
\rho u^2 + p \\
\rho uv \\
\rho uw \\
\rho uH 
\end{pmatrix}, \quad G = \begin{pmatrix} 
\rho v \\
\rho u v + p \\
\rho wv \\
\rho wh \\
\rho vH 
\end{pmatrix}, \quad H = \begin{pmatrix} 
\rho w \\
\rho uv \\
\rho uw \\
\rho w^2 + p \\
\rho wH 
\end{pmatrix}
\]

(2.2)

where \( \rho \) is density, \( u, v, \) and \( w \) are the flow velocities in the \( x, y, \) and \( z \) directions, \( p \) is static pressure, \( E \) is total energy per unit mass, and \( H \) is the total enthalpy per unit mass, given by the thermodynamic relation

\[
H = E + \frac{p}{\rho}.
\]

(2.3)

An equation of state is used to close the system of equations. Under the assumption of an ideal gas, the perfect gas law, \( p = \rho RT \), eliminates temperature from the definition of \( E = c_v T + \frac{1}{2} (u^2 + v^2 + w^2) \) and yields the closing relation

\[
\frac{p}{\rho} = (\gamma - 1)(E - \frac{1}{2} (u^2 + v^2 + w^2)),
\]

(2.4)

where the specific heat ratio \( \gamma \) is equal to 1.4 for air and is constant for all calculations performed.

### 2.2 Non-Dimensionalization

It is usually convenient to non-dimensionalize the governing equations for a problem since this makes the solutions independent of any particular system of units, clarifies
Table 2.1: Scaling Factors for Non-Dimensionalization

The scales important to the problem and can reduce the sensitivity of the solution to round-off errors. Table 2.1 lists the scaling factors and freestream values for each of the variables in the Euler equations. It can be seen that the important non-dimensional flow parameters are the freestream Mach number, $M_{\infty}$, and $\gamma$. Non-dimensionalization does not change the form of the equations, but does alter the form of the freestream boundary conditions.

### 2.3 Auxiliary Quantities

The following is a list of relevant auxiliary quantities that can be defined in terms of the primitive variables:

- **Local speed of sound:**
  \[ a = \sqrt{\frac{\gamma p}{\rho}} \]

- **Mach Number:**
  \[ M = \frac{\sqrt{u^2 + v^2 + w^2}}{a} \]

- **Total Pressure:**
  \[ P_0 = p \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\gamma/(\gamma-1)} \]

- **Total Pressure Loss:**
  \[ \Delta P_0 = \frac{P_{0\infty} - P_0}{P_{0\infty}} \]
Entropy: \[ \Delta S = \log \frac{T}{\rho^{\gamma}} \]

where the freestream entropy is defined to be zero. As mentioned previously, normalized helicity is used as one of the adaptation quantities. The normalized helicity is defined as:

\[ H_n = \frac{\vec{V} \cdot \vec{\omega}}{|\vec{V}| |\vec{\omega}|} \] (2.5)

where \( \vec{V} \) is the velocity and \( \vec{\omega} \) is the vorticity. The normalized helicity is the cosine of the angle between the velocity vector and the vorticity vector. Thus, when the vorticity and velocity vectors are aligned the normalized helicity has a value of either \( \pm 1 \). The physical interpretation of helicity and its application for adaptation will be expanded upon in Section 5.2.1.

### 2.4 Physical Boundary Conditions

Boundary conditions must be specified in order to solve for any system of differential equations. The problems addressed in this thesis use only farfield boundaries; however, solid wall boundary conditions also have been implemented and are discussed in Appendix A. The implementation of the farfield boundary conditions is discussed in Section 4.3.

The farfield boundary conditions are based on quasi-one-dimensional characteristic theory. The three-dimensional Euler equations are transformed into a system based on coordinates normal and tangential to the boundary. The transformed directions will be denoted by \((n, t, b)\), where \(n\) is normal to the boundary (positive pointing inward) and \(t\) and \(b\) are tangential to the boundary. Assuming that there are no variations in the \(t\) and \(b\) directions, i.e., the derivatives tangential to the boundaries are neglected, this reduces the three-dimensional Euler equations to a one-dimensional system of equations. Manipulation of the resulting unsteady, inviscid one-dimensional Euler equations yields
the following near-diagonal system of equations [12]

\[
\begin{pmatrix}
\frac{\partial}{\partial t} & u_t \\
\frac{\partial}{\partial t} & u_b \\
S & 0 \\
J_+ & 0 \\
J_- & 0
\end{pmatrix}
\begin{pmatrix}
u_n \\
u_n \\
u_n \\
u_n \\
u_n
\end{pmatrix}
\begin{pmatrix}
u_n \\
u_n \\
u_n \\
u_n \\
u_n
\end{pmatrix}
\begin{pmatrix}
u_t \\
u_b \\
S \\
J_+ \\
J_-
\end{pmatrix} = 0
\]

where \(S\) is the entropy, \(u_n\) is the velocity normal to the boundary, \(u_t\) and \(u_b\) are the velocities tangential to the boundary, and \(J_+\) and \(J_-\) are the Riemann invariants defined as

\[
J_+ = u_n + \frac{2a}{\gamma - 1},
\]

\[
J_- = u_n - \frac{2a}{\gamma - 1}.
\]

The system is diagonalized by assuming locally isentropic flow. For the vortex cases used the flow is not locally isentropic; however, an analysis of the off-diagonal and diagonal terms showed that the off-diagonal terms are small. Therefore, dropping these terms results in a good approximation and decouples the equations. Thus the characteristic equations are

\[
\frac{\partial J_+}{\partial t} + (u_n + a) \frac{\partial J_+}{\partial n} = 0,
\]

\[
\frac{\partial J_-}{\partial t} + (u_n - a) \frac{\partial J_-}{\partial n} = 0,
\]

\[
\frac{\partial u_t}{\partial t} + u_n \frac{\partial u_t}{\partial n} = 0,
\]

\[
\frac{\partial u_b}{\partial t} + u_n \frac{\partial u_b}{\partial n} = 0,
\]

\[
\frac{\partial S}{\partial t} + u_n \frac{\partial S}{\partial n} = 0.
\]

The characteristic variables \(u_t, u_b\) and \(S\) are convected normal to the boundary at velocity \(u_n\), while \(J_+\) is convected normal to the boundary at velocity \(u_n + a\) and \(J_-\) is convected normal to the boundary at velocity \(u_n - a\). Therefore, if \(0 < u_n < a\), the boundary is a subsonic inflow boundary, so \(J_+, S, u_t\) and \(u_b\) propagate into the domain, while \(J_-\) propagates out of the domain. A subsonic outflow boundary exists.
when $-a < u_n < 0$ in which case $J_+$ is specified from the exterior of the domain and the other characteristics come from the interior. At a supersonic inflow boundary, $u_n > a$, all the characteristics are entering the domain and thus must be specified. For a supersonic outflow boundary, $u_n < -a$, all the characteristics propagate out of the domain, therefore, nothing must be specified.
Chapter 3
3-D Finite Element Concepts

This chapter will describe some of the important concepts of the three-dimensional finite element method used in this thesis. For a broader overview of finite element methods in both two and three dimensions for the Euler equations, the work of Shapiro [27] is highly recommended. The following finite element description is based upon that work. This chapter will focus strictly on 8 node, 3-D trilinear elements. The terms element, node, edge and face are defined, and the transformations between physical and computational space are described. A brief discussion of the derivative calculation is also given.

3.1 Terminology

When using numerical methods, the physical domain of interest must be discretized in some fashion. In the finite element method the physical domain is subdivided into elements, each of which is composed of some number of nodes. In this thesis, the physical domain, which is three-dimensional, is subdivided into hexahedral elements each with 8 nodes. A hexahedral element will have 6 faces and 12 edges. Figure 3.1 illustrates a 3-D element showing the faces, edges and nodes.

The advantage of hexahedral elements is that a hexahedral mesh will have roughly five times less elements than a tetrahedral mesh. This results in significant savings in CPU time and memory since many operations are performed on elements. The disadvantages of hexahedral elements are that during grid generation hexahedral elements may be more difficult to fit to a geometry and with grid adaptation special treatment must be given to the nodes at the interface between the coarse grid and the fine grid. Specifically, at the interface the new nodes generated from grid adaptation will have
to be interpolated from the coarse grid and conservation should be maintained on the new adaptive grid. The treatment of the interface will be discussed in Section 5.1. The difficulty of fitting hexahedral elements to a geometry can be circumvented since the finite element method allows degenerate elements where a degenerate hexahedron will result in a prism. Figure 3.2 a shows degenerate hexahedra.

In the finite element method, all operations are performed at the elemental level, with element contributions distributed to the nodes. The finite element is *unstructured* in nature. It is not necessary for each grid point to be indexed by \((i, j, k)\) and a node may belong to any number of elements.
### 3.2 Finite Element Discretization

The finite element discretization will be described in three dimensions. The finite element method provides a way to make a convenient transformation between a local, computational space (natural coordinates) and a global, physical space. The natural coordinates will be denoted by \((\xi, \eta, \zeta)\) and the physical coordinates will be denoted by \((x, y, z)\). Figure 3.3 shows a typical 3-D finite element with the directions of the natural coordinates.

![Natural Coordinates](image)

In the finite element discretization one assumes that within each element, some quantity \(q^{(e)}\) is determined by its nodal values \(q_i\) and a set of shape or interpolation functions \(N_i^{(e)}\). Typically, the interpolation functions are chosen to be polynomials in the natural coordinates. The quantity, \(q^{(e)}\), is then given by

\[
q^{(e)}(\xi, \eta, \zeta) = \sum_{i=1}^{8} q_i N_i^{(e)}(\xi, \eta, \zeta), \tag{3.1}
\]

where the summation is over the nodes of the element. Similarly, the geometry of the element is interpolated in terms of nodal coordinates, i.e.,

\[
x^{(e)}(\xi, \eta, \zeta) = \sum_{i=1}^{8} x_i N_i(\xi, \eta, \zeta), \tag{3.2}
\]
\[ y^{(e)}(\xi, \eta, \zeta) = \sum_{i=1}^{8} y_i N_i(\xi, \eta, \zeta), \quad (3.3) \]
\[ z^{(e)}(\xi, \eta, \zeta) = \sum_{i=1}^{8} z_i N_i(\xi, \eta, \zeta), \quad (3.4) \]
\[ (3.5) \]

where \( x_i, y_i \) and \( z_i \) are the coordinates of node \( i \) in element \( e \) and the sum is taken over the eight nodes.

The element shape functions are summed to give global shape functions \( N_i \), so that globally \( q \) can be written
\[ q(x, y, z) = \sum_{i=1}^{M} N_i(x, y, z) q_i, \quad (3.6) \]
where \( M \) is the total number of nodes in the mesh. The relation between \( N_i^{(e)}(\xi, \eta, \zeta) \) and \( N_i(x, y, z) \) is given in the next section.

### 3.2.1 Interpolation Functions

The interpolation functions \( N_i^{(e)} \) must have certain properties for the finite element approximation to be valid. The following lists the properties of the interpolation functions used in this thesis:

1. The interpolation function \( N_i^{(e)} \) must be 1 at node \( i \) and 0 at all other nodes of the domain. This is required so that Eq. (3.6) can hold for each node.

2. The interpolation function \( N_i^{(e)} \) is 0 outside of the element \( e \). This holds for a local finite element approximation. As a result of this property the global shape function \( N_i(x, y, z) \) at node \( i \) is the sum of the elemental interpolation functions \( N_i^{(e)}(\xi, \eta, \zeta) \) for all the elements containing node \( i \).

3. In each element, the sum of all the nodal interpolation functions \( N_i^{(e)} \) are identically 1 at each point \((\xi, \eta, \zeta)\) in the element. This is required for consistency in the approximation.
3.2.2 Derivative Calculation

The derivative of a quantity in terms of the nodal values of that quantity is often desired. In order to calculate the derivatives in physical space, the Jacobian of the transformation is required. This is seen below:

\[
\begin{bmatrix}
\frac{\partial q}{\partial \xi} \\
\frac{\partial q}{\partial \eta} \\
\frac{\partial q}{\partial \zeta}
\end{bmatrix}
= J
\begin{bmatrix}
\frac{\partial q}{\partial x} \\
\frac{\partial q}{\partial y} \\
\frac{\partial q}{\partial z}
\end{bmatrix}
\tag{3.7}
\]

where \( J \) is the Jacobian matrix

\[
J = \begin{bmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\
\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\
\frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta}
\end{bmatrix}
= \begin{bmatrix}
\sum_i x_i \frac{\partial N_i^{(e)}}{\partial \xi} & \sum_i y_i \frac{\partial N_i^{(e)}}{\partial \xi} & \sum_i z_i \frac{\partial N_i^{(e)}}{\partial \xi} \\
\sum_i x_i \frac{\partial N_i^{(e)}}{\partial \eta} & \sum_i y_i \frac{\partial N_i^{(e)}}{\partial \eta} & \sum_i z_i \frac{\partial N_i^{(e)}}{\partial \eta} \\
\sum_i x_i \frac{\partial N_i^{(e)}}{\partial \zeta} & \sum_i y_i \frac{\partial N_i^{(e)}}{\partial \zeta} & \sum_i z_i \frac{\partial N_i^{(e)}}{\partial \zeta}
\end{bmatrix}
\tag{3.8}
\]

Once \( J \) has been assembled, \( J^{-1} \) can be calculated, so the derivatives of a quantity \( q \) in each element can be computed as follows:

\[
\begin{bmatrix}
\frac{\partial q^{(e)}}{\partial x} \\
\frac{\partial q^{(e)}}{\partial y} \\
\frac{\partial q^{(e)}}{\partial z}
\end{bmatrix}
= J^{-1}
\begin{bmatrix}
\sum_i q_i \frac{\partial N_i^{(e)}}{\partial \xi} \\
\sum_i q_i \frac{\partial N_i^{(e)}}{\partial \eta} \\
\sum_i q_i \frac{\partial N_i^{(e)}}{\partial \zeta}
\end{bmatrix}
, \tag{3.9}
\]

where the \( q_i \) are the nodal values of \( q \).
3.3 Trilinear Elements

The 8-node, three-dimensional element shown in Fig. 3.4 is a trilinear element. To help clarify this figure, Table 3.1 lists the nodes that make up each face of the element. The interpolation functions for a trilinear element are

\[ N_1 = (1 - \xi)(1 - \eta)(1 - \zeta)/8, \]  
\[ N_2 = (1 + \xi)(1 - \eta)(1 - \zeta)/8, \]  
\[ N_3 = (1 + \xi)(1 + \eta)(1 - \zeta)/8, \]

Table 3.1: Nodes for Each Face, Trilinear Element

<table>
<thead>
<tr>
<th>Face</th>
<th>Nodes on Face</th>
<th>Face</th>
<th>Nodes on Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-2-3-4</td>
<td>4</td>
<td>6-7-3-2</td>
</tr>
<tr>
<td>2</td>
<td>8-7-6-5</td>
<td>5</td>
<td>4-3-7-8</td>
</tr>
<tr>
<td>3</td>
<td>1-5-6-2</td>
<td>6</td>
<td>1-4-8-5</td>
</tr>
</tbody>
</table>
\[ N_4 = (1 - \xi)(1 + \eta)(1 - \zeta)/8, \]  
\[ N_5 = (1 - \xi)(1 - \eta)(1 + \zeta)/8, \]  
\[ N_6 = (1 + \xi)(1 - \eta)(1 + \zeta)/8, \]  
\[ N_7 = (1 + \xi)(1 + \eta)(1 + \zeta)/8, \]  
\[ N_8 = (1 - \xi)(1 + \eta)(1 + \zeta)/8. \]  

It will be convenient to expand Eqns. (3.1) - (3.4) as follows:

\[ z(\xi, \eta, \zeta) = a_0 + a_1 \xi + a_2 \eta + a_3 \zeta + a_4 \xi \eta + a_5 \xi \zeta + a_6 \eta \zeta + a_7 \xi \eta \zeta, \]  
\[ y(\xi, \eta, \zeta) = b_0 + b_1 \xi + b_2 \eta + b_3 \zeta + b_4 \xi \eta + b_5 \xi \zeta + b_6 \eta \zeta + b_7 \xi \eta \zeta, \]  
\[ z(\xi, \eta, \zeta) = c_0 + c_1 \xi + c_2 \eta + c_3 \zeta + c_4 \xi \eta + c_5 \xi \zeta + c_6 \eta \zeta + c_7 \xi \eta \zeta, \]  
\[ q(\xi, \eta, \zeta) = d_0 + d_1 \xi + d_2 \eta + d_3 \zeta + d_4 \xi \eta + d_5 \xi \zeta + d_6 \eta \zeta + d_7 \xi \eta \zeta. \]  

where

\[ a_0 = \frac{(q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7 + q_8)/8,}{(3.22)} \]  
\[ d_1 = \frac{(-q_1 + q_2 + q_3 - q_4 - q_5 + q_6 + q_7 - q_8)/8,}{(3.23)} \]  
\[ d_2 = \frac{(-q_1 - q_2 + q_3 + q_4 - q_5 - q_6 + q_7 + q_8)/8,}{(3.24)} \]  
\[ d_3 = \frac{(q_1 - q_2 + q_3 + q_4 + q_5 - q_6 + q_7 + q_8)/8,}{(3.25)} \]  
\[ d_4 = \frac{(q_1 - q_2 - q_3 - q_4 + q_5 + q_6 + q_7 - q_8)/8,}{(3.26)} \]  
\[ d_5 = \frac{(q_1 - q_2 + q_3 - q_4 + q_5 + q_6 + q_7 + q_8)/8,}{(3.27)} \]  
\[ d_6 = \frac{(q_1 + q_2 - q_3 - q_4 + q_5 + q_6 + q_7 + q_8)/8,}{(3.28)} \]  
\[ d_7 = \frac{(q_1 - q_2 - q_3 + q_4 - q_5 + q_6 + q_7 - q_8)/8,}{(3.29)} \]  

The coefficients \(a_i, b_i,\) and \(c_i\) are determined by the same equations as above except with the \(q_i's\) being replaced by \(z_i's, y_i's,\) and \(z_i's,\) respectively. The Jacobian matrix can be calculated by using Eq. (3.8). The three-dimensional Jacobian \(J\) is

\[
J = \begin{bmatrix}
a_2 + a_5 \eta + a_7 \zeta + a_8 \eta \zeta & b_2 + b_5 \eta + b_7 \zeta + b_8 \eta \zeta & c_2 + c_5 \eta + c_7 \zeta + c_8 \eta \zeta \\
a_3 + a_5 \xi + a_6 \zeta + a_8 \xi \zeta & b_3 + b_5 \xi + b_6 \zeta + b_8 \xi \zeta & c_3 + c_5 \xi + c_6 \zeta + c_8 \xi \zeta \\
a_4 + a_6 \eta + a_7 \xi + a_8 \xi \eta & b_4 + b_6 \eta + b_7 \xi + b_8 \xi \eta & c_4 + c_6 \eta + c_7 \xi + c_8 \xi \eta
\end{bmatrix}
\]  

where \(a_i, b_i,\) and \(c_i\) are the coefficients in the expansions of \(x, y,\) and \(z\) in the element.
Chapter 4
Solution Algorithm

This chapter describes the 3-D finite element algorithm used to obtain the steady-state solution of the Euler equations as developed by Shapiro [27]. An overview of the algorithm will first be described. The spatial discretization of the finite element method will then be described. The term "finite element method" is actually quite broad in meaning, the selection of the test functions results in quite different methods. For the 3-D calculations presented in this thesis a "cell-vertex" finite element method was used. The spatial discretization and test functions used for the cell-vertex finite element method is described within Section 4.2. The implementation of the boundary conditions is then discussed. Artificial viscosity is required for stability and is discussed in Section 4.4. Section 4.5 will then describe the time marching scheme. It should be noted here that the time marching scheme is not time accurate. Finally, consistency and conservation are briefly discussed.

4.1 Overview

The steady-state solution is obtained from the unsteady Euler equations by using a time marching technique. First, an initial condition must be imposed, then the solution is corrected by an iterative technique that resembles the solution of the unsteady problem until some level of convergence is achieved. The iterative technique used is a four-stage time integration scheme which essentially consists of the following steps: The boundary conditions are applied before each step in the multistage scheme. Then, a summation of the fluxes over the element volume is calculated. Next, artificial viscosity is computed during the first stage of the integration and "frozen". The artificial viscosity is then added to the flux summation and this quantity is termed the residual. The residual
then represents the difference between the current step and the previous step. Finally, the current solution is updated to obtain the next approximation. This process is then repeated four times. The entire process is repeated until some desired level of convergence is achieved. Convergence is signalled when the RMS of all changes divided by the RMS of all the state vectors is less than some specified value, typically around $10^{-6}$.

4.2 Spatial Discretization

The conservation form of the Euler equations (Eq. (2.1)) is used for spatial discretization and is written as

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0$$

where $U$ is the state vector and $F$, $G$ and $H$ are the flux vectors in the $z$, $y$ and $z$ directions. Within each element the state vector $U^{(e)}$ and flux vector $F^{(e)}$, $G^{(e)}$ and $H^{(e)}$ are written

$$U^{(e)} = \sum_i N_i^{(e)} U_i,$$

$$F^{(e)} = \sum_i N_i^{(e)} F_i,$$

$$G^{(e)} = \sum_i N_i^{(e)} G_i,$$

$$H^{(e)} = \sum_i N_i^{(e)} H_i,$$

where $U_i$, $F_i$, $G_i$ and $H_i$ are the nodal values of the state vector and flux vectors, and $N_i^{(e)}$ is the set of interpolation functions for element $e$.

The derivatives of these expressions can be computed by the method described in Section 3.2. The formula for the derivative in each element in terms of the nodal values follows:

$$\frac{\partial U^{(e)}}{\partial t} = \sum_i N_i^{(e)} \frac{dU_i}{dt}$$
\[
\frac{\partial F^{(e)}}{\partial z} = \sum_i \frac{\partial N_i^{(e)}}{\partial z} F_i \tag{4.7}
\]

\[
\frac{\partial G^{(e)}}{\partial y} = \sum_i \frac{\partial N_i^{(e)}}{\partial y} G_i \tag{4.8}
\]

\[
\frac{\partial H^{(e)}}{\partial z} = \sum_i \frac{\partial N_i^{(e)}}{\partial z} H_i \tag{4.9}
\]

where the summation is over the nodes \(i\) of element \(e\). Summing over all of the elements in the domain results in the following equation:

\[
N_i \frac{dU_i}{dt} = -\frac{\partial N_i}{\partial z} F_i - \frac{\partial N_i}{\partial y} G_i - \frac{\partial N_i}{\partial z} H_i \tag{4.10}
\]

where \(N_i\) is the sum of the interpolation functions of node \(i\) for each element that contains node \(i\). It should be noted here that \(N_i\) is now a global row vector and \(dU_i/dt\) is a column vector. These vectors are of length \(M\), where \(M\) is the total number of nodes in the mesh.

Equation (4.10) does not hold for all points in space since the derivatives of the interpolation functions do not exist for all points in space. Instead of asking for an equation that holds at each point we need an equation that holds for each function [28]. The right form needed is the weak form of the equations. The weak form minimizes the error in the discretization by having the error orthogonal to the space spanned by a new row vector of functions, \(\tilde{N}_j\), called test functions. \(\tilde{N}_j\) has a length equal to the number of nodes. The weak form of the equation now holds for each test function. To create the weak form, premultiply Eq. (4.10) by \(\tilde{N}_j^T\) and integrate over the entire domain. Because we are integrating over the entire domain, this allows for the introduction of discontinuous solutions as well as providing some means for obtaining the nodal values of the unknowns. The weak form of Eq. (4.10) is

\[
M_{ij} \frac{dU_i}{dt} = -\iint (\tilde{N}_j^T \frac{\partial N_i}{\partial z} F_i + \tilde{N}_j^T \frac{\partial N_i}{\partial y} G_i + \tilde{N}_j^T \frac{\partial N_i}{\partial z} H_i) dz \, dy \, dz, \tag{4.11}
\]

where \(M_{ij}\) is the consistent mass matrix. and is determined as follows

\[
M_{ij} = \iint \tilde{N}_j^T N_i dz \, dy \, dz. \tag{4.12}
\]
This results in the semi-discrete equation

\[ M_{ij} \frac{dU_i}{dt} = -(R_x)_{ij} F_i - (R_y)_{ij} G_i - (R_z)_{ij} H_i \]  

(4.13)

where \( R_x, R_y \) and \( R_z \) are the residual matrices. The matrices \( M, R_x, R_y \) and \( R_z \) involve the integration of quantities over all elements in the domain. These integrations are performed at the elemental level in natural coordinates and are assembled to give the global matrices.

At this point the choice of test functions and resulting methods from each choice will be discussed. Shapiro [27] discusses three choices of test functions in two dimensions that result in the Galerkin method, the cell-vertex method and the central difference method, all of which fit into the “finite element method”. The test functions for the Galerkin method are chosen to be the same as the interpolation functions, for the cell-vertex method the test functions are chosen to be a constant, and for the central difference method the test functions can be set to a series of Dirac delta functions. The Galerkin and cell-vertex methods were shown to be both more robust and computationally efficient than the central difference method. The cell-vertex method was less computationally expensive than the Galerkin method, although this could be due to optimizations in the residual calculation within the cell-vertex method. In three dimensions, only the cell-vertex was implemented. For trilinear elements, the test functions, \( N_j^{(e)} \), for each node within each element were chosen to be 1/8.

The calculation of each residual matrix is identical, therefore, only the calculation of the \( R_x \) matrix will be shown here. The calculation of the \( R_x \) matrix begins with

\[ R_x = \iint \tilde{N}_j^T \frac{\partial N_i}{\partial x} \, dx \, dy \, dz, \]

(4.14)

using Eq. (3.9) \( \frac{\partial N_i}{\partial x} \) and \( dx \, dy \, dz \) translated into natural coordinates become

\[ \frac{\partial N_i}{\partial x} \Rightarrow (J_{11}^{-1} \frac{\partial N_i}{\partial \xi} + J_{12}^{-1} \frac{\partial N_i}{\partial \eta} + J_{13}^{-1} \frac{\partial N_i}{\partial \zeta}) \]

(4.15)

\[ dz \, dy \, dz \Rightarrow |J| \, d\xi \, d\eta \, d\zeta. \]

(4.16)

Thus, the \( R_x \) matrix expressed in natural coordinates is

\[ R_x = \iint \tilde{N}_j^T (J_{11}^{-1} \frac{\partial N_i}{\partial \xi} + J_{12}^{-1} \frac{\partial N_i}{\partial \eta} + J_{13}^{-1} \frac{\partial N_i}{\partial \zeta}) |J| \, d\xi \, d\eta \, d\zeta. \]

(4.17)
The inverse of $J$ can be expressed as

$$J^{-1} = \frac{J^*}{|J|}$$

(4.18)

where $J^*$ is the adjoint of $J$ [29]. Substituting this formula into Eq. (4.17) yields

$$R_x = \int_{-1}^{1} \int_{-1}^{1} \bar{N}^T \left( J_{1,1}^* \frac{\partial N}{\partial \xi} + J_{1,2}^* \frac{\partial N}{\partial \eta} + J_{1,3}^* \frac{\partial N}{\partial \zeta} \right) d\xi d\eta d\zeta.$$  

(4.19)

For the mass matrix, all the quantities being integrated are also polynomials. Thus, all the element integrals can be done analytically, resulting in a significant savings in CPU time. (Note: in the implementation the element integrals are computed exactly using MACSYMA since hand generation is prone to mistakes.)

The nodal values of the state vector are solved using Eq. (4.13). The mass matrix $M$ is a sparse, positive definite matrix, but is unstructured making it computationally expensive to invert. In this thesis, we are only interested in the steady-state solution; therefore, $M$ can be replaced by a "lumped" (diagonal) matrix, $M_L$, where each diagonal entry is the sum of all the elements in the corresponding row of $M$. This allows Eq. (4.13) to be solved explicitly, thus inexpensively. For the test functions chosen, the lumped mass matrix has a value along the diagonal equal to $1/8$ of the cell volume.

4.3 Boundary Conditions

Before discussing the implementation of the boundary conditions, several definitions are needed for clarity. First, a face of an element can exist in the interior of the domain or along a boundary (a solid boundary may exist in the interior of the domain such as with a planform of a wing). The boundary face of an element can either be a solid boundary face or a farfield boundary face. The types of boundary faces and boundary nodes an element can have and the types of boundary nodes that can exist on each boundary face are summarized below.

- Boundary face types: solid, farfield
- Boundary nodes types: solid, farfield, corner
• Possible nodes on a \textit{solid} face: solid, farfield, corner

• Possible nodes on a \textit{farfield} face: farfield

Farfield faces are the simplest faces since they can only contain farfield nodes. The implementation of solid wall boundary conditions is discussed in Appendix A.

The implementation of the farfield boundary conditions will now be described. As discussed in Section 2.4, the farfield boundaries are based on one-dimensional characteristic theory. For farfield boundaries, the inward pointing unit normal vector $\hat{n}$ and the two unit tangent vectors $\hat{i}$ and $\hat{b}$ must be computed. First, the normal vector must be computed. This is computed as follows:

1. Loop over all elements in the domain.
2. Identify any farfield faces on each element.
3. Determine the farfield node numbers on the farfield face.
4. Compute the normal vector components, i.e. $n_x$, $n_y$, $n_z$, by taking the components of the cross product of the diagonals of the face.
5. Sum the normal vector components of all faces that contain the farfield node $i$.
6. Normalize the components by the magnitude of the normal vector, $\hat{n}$.

The normal vector is computed the same for all farfield nodes. The tangential velocity vector can now be computed using the following formula

$$\vec{u}_t = \vec{u} - (\vec{u} \cdot \hat{n})\hat{n}. \quad (4.20)$$

The unit tangent vector is then determined by

$$\hat{i} = \frac{\vec{u}_t}{|\vec{u}_t|}. \quad (4.21)$$

If the tangential velocity is zero, then an arbitrary tangential vector is chosen. The other unit tangential vector $\hat{b}$ is calculated by taking the cross product of $\hat{n}$ and $\hat{i}$.
The 1-D Riemann invariants and the corresponding wave speeds are now defined using the above vectors

\[
\begin{bmatrix}
\frac{2a}{\gamma - 1} + u_n \\
\frac{\gamma - 1}{2a} - u_n \\
\frac{p}{\rho^2} \\
u_t \\
u_b
\end{bmatrix}
= \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix}, \quad \text{speeds: } \begin{bmatrix} u_n + a \\ u_n - a \\ u_n \\ u_n \\ u_n \end{bmatrix}, \quad (4.22)
\]

where \(u_n = \vec{v} \cdot \vec{n}, u_t = \vec{v} \cdot \vec{t}, \) and \(u_b = \vec{v} \cdot \vec{b}.\) First, the wave speeds must be computed for each point on the boundary. This is done using the updated values of the state vector of the nodes along the boundary. Then for each point on the boundary, the characteristics are calculated using the solution state vector \(U\) and the freestream state vector \(U_{\infty}\). Next, a decision is made based upon the sign of the wave speeds whether to use the invariant based on the interior state vector or the freestream state vector. If the wave speed is positive then the characteristic based upon the freestream state vector is used.

The invariants are transformed back into primitive variables, and these primitive variables are then used to calculate the fluxes at the boundary nodes in the residual calculation. The primitive variables are calculated as follows:

\[u_n = \frac{1}{2}(C_1 - C_2), \quad (4.23)\]

\[a = \frac{\gamma - 1}{4}(C_1 + C_2), \quad (4.24)\]

\[\rho = \left(\frac{a^2}{\gamma C_3}\right)^{(\gamma - 1)/(\gamma - 1)}, \quad (4.25)\]

\[p = \frac{\rho a^2}{\gamma}, \quad (4.26)\]

\[u = u_n n_x + C_4 t_x + C_5 b_x, \quad (4.27)\]

\[v = u_n n_y + C_4 t_y + C_5 b_y, \quad (4.28)\]

\[w = u_n n_z + C_4 t_z + C_5 b_z, \quad (4.29)\]
\[ H = \frac{\gamma P}{\gamma - 1 \rho} + \frac{1}{2}(u^2 + v^2 + w^2), \quad (4.30) \]

where \( C_1 - C_5 \) are the characteristic variables above. The characteristics are also used to update the state vector at the beginning of each iteration.

### 4.4 Smoothing

Artificial viscosity is added in order to stabilize the numerical scheme as well as damp out background disturbances. The smoothing used consists of a pressure-switched second difference term and a fourth difference term, similar to that discussed by Rizzi and Erikson [25]. A Laplacian-type of second difference is used due to the unstructured nature of the grids. The smoothing is based essentially on the calculation of an elemental contribution to a second difference. The second difference smoothing method suggested by Ni [19] is used. This method is relatively fast, conservative and robust, but gives a non-zero contribution to the second difference for a linear function on a non-uniform grid, resulting in first-order accuracy. Thus, the right-hand side of Eq. (4.13) will have a non-zero contribution due to the second difference [17].

#### 4.4.1 Second Difference Smoothing

Figure 4.1 shows the contribution of a typical element to the second difference at node 1. The numbers inside the box are the node numbers, the numbers outside are the weights, denoted \( w_i \) for node \( i \). The contribution from an element to a node is obtained by subtracting the value at the node from the average value in the element. The second difference at a node is then the sum of the contributions from all elements that contain node \( i \). That is, the contribution to the second difference at node 1 from element \( e \) is

\[ V_1^{(e)} = k^{(e)} \left( \frac{U_1 + U_2 + U_3 + U_4 + U_5 + U_6 + U_7 + U_8}{8} - U_1 \right) \quad (4.31) \]

or equivalently expressed using the weights

\[ V_1^{(e)} = k^{(e)} \sum_{i=1}^{8} w_i U_i \quad (4.32) \]
where $k^{(e)}$ is some elemental weight (such as a pressure switch which is discussed below). This second difference method is conservative since the sum of the contributions over all the nodes within each element is zero. However, it is of lower accuracy since on a non-uniform mesh a linear function in $x$, $y$, and $z$ can produce a non-zero second difference at each node.

The second difference smoothing is weighted by an elemental pressure switch, $S$. In order to calculate the elemental pressure switch, the nodal second difference of pressure is calculated by the method described above. The second difference at a node is then normalized by a nodal pressure average at the node. The nodal pressure average is the sum of the pressures from all of the elements that contain node $i$ divided by the number of elements that contain node $i$. An elemental value is obtained by summing the second difference of pressure at node $i$ normalized by the average pressure at node $i$ over the nodes in the element. The resultant quantity is then scaled so that the maximum over the entire mesh is 1. Thus, the elemental pressure switch $S$ is

$$S = \frac{1}{S_{\text{max}}} \sum_i \frac{(D_2p)_i}{\bar{p}_i}$$

where $(D_2p)_i$ is the second difference of pressure at node $i$, $\bar{p}_i$ is the average pressure at node $i$, the summation is over all the nodes in the element, and $S_{\text{max}}$ is chosen so that the maximum value of $S$ over the entire mesh is 1.
4.4.2 Combined Smoothing

The complete smoothing for a time step is a combined value of a second difference term and a fourth difference term. The second difference smoothing term is the pressure switched second difference of the state vectors multiplied by a constant, $\nu_2$, between 0.00 and 0.05. The fourth difference smoothing is the second difference of the second difference of the state vector multiplied by a constant, $\nu_4$, between 0.0005 and 0.05. Thus, the combined smoothing is given by

$$V_i = \frac{M_{Li}}{\Delta t_i} (\nu_2 D_2^S U_i - \nu_4 D_2 (D_2 U_i)),$$

(4.34)

where $D_2$ indicates the second difference operator, the superscript $S$ indicates that the second difference is weighted by the pressure switch, the subscript $i$ denotes a node, $M_{Li}$ is the entry in the lumped mass matrix and $\Delta t_i$ is the local time step (this will be defined in Section 4.5). The smoothing terms are grid dependent with the second difference term being first-order accurate and the fourth difference term being third-order accurate. (The pressure switch as implemented results in the second difference term being first-order accurate. Jameson's pressure switch is first-order accurate at shocks and third-order accurate in smooth regions of the flow [14]. For the calculations in this thesis, no second difference smoothing was used; therefore, this was not an issue.)

The smoothing must be scaled by the lumped mass matrix divided by the time step to obtain the correct order of magnitude. It was realized later that the smoothing is not strictly conservative. For conservation $\frac{M_{Li}}{\Delta t}$ must be an elemental value and placed in the same location as $k(e)$ in Eq. (4.31). However, since there are no shocks in the test cases this will be inconsequential. The combined smoothing term $V_i$ is then added directly into the time integration as will be seen in the next section. In this thesis, the smoothing is computed at the first stage of the multistage time integration and "frozen" for the remaining stages in order to save CPU time.
4.5 Time Integration

Since we are solving the unsteady Euler equations to achieve a steady-state solution, we need to integrate Eq. (4.13) in time. A four-stage integration scheme is used:

\[
\begin{align*}
U_i^{(1)} & = U_i^n + \frac{1}{4} \lambda \frac{\Delta t_i}{M_{Li}} \left( - \sum Fluzes_i(U^n) + V_i^n \right), \\
U_i^{(2)} & = U_i^n + \frac{1}{3} \lambda \frac{\Delta t_i}{M_{Li}} \left( - \sum Fluzes_i(U^{(1)}) + V_i^n \right), \\
U_i^{(3)} & = U_i^n + \frac{1}{2} \lambda \frac{\Delta t_i}{M_{Li}} \left( - \sum Fluzes_i(U^{(2)}) + V_i^n \right), \\
U_i^{(4)} & = U_i^n + \lambda \frac{\Delta t_i}{M_{Li}} \left( - \sum Fluzes_i(U^{(3)}) + V_i^n \right), \\
U_i^{n+1} & = U_i^{(4)},
\end{align*}
\]

where \( \sum Fluzes_i(U) \), the flux summation, is the right-hand side of Eq. (4.13) with the fluxes based on the state vector \( U \), \( V_i \) is the combined smoothing described in Section 4.4, \( M_{Li} \) is the entry in the lumped mass matrix for node \( i \), and \( \lambda \) is the CFL number.

Local time stepping is used to accelerate convergence with the time step given by

\[
\Delta t_i = \frac{\Delta z_i}{|u| + a}
\]

where \( \Delta z_i \) is the minimum over all elements containing the node \( i \) of the average length of the opposite sides of the element, and \( u \) is the flow velocity at the node.

A von Neumann stability analysis in three dimensions is quite complicated; however, the stability limit can be estimated from the one dimensional linear analysis. The stability limit for \( \lambda \) in \( n \) dimensions can be estimated by

\[
\lambda = \frac{1 - D \text{ limit}}{\sqrt{\# \text{ of dimensions}}}
\]

This is based upon the time it takes a wave to travel the longest distance of a cell. The 1-D limit is \( 2\sqrt{2} \); therefore, the 3-D limit will be approximately 1.63.

In order to evaluate the accuracy of this approximation, the 2-D limit will be used for comparison. The wave equation in two dimensions is \( U_t + aU_x + bU_y = 0 \), if \( a = b = 1 \) then the stability limit for \( \lambda \) for the cell-vertex method in two-dimensions is 2.17 [27]. This two dimensional analysis is a worst-case analysis since if \( a/b \) is either large or small
compared to 1, the stability limit is closer to the 1-D limit. For comparison, the above method based on the 1-D limit using Eq. (4.37) would approximate $\lambda = 2$. Thus, the approximate 3-D stability limit is a conservative estimate.

4.6 Consistency and Conservation

It is necessary for a numerical scheme to be consistent and desirable for it to be conservative. This applies to the numerical scheme used to solve for the Euler equations as well. Consistency means that as the mesh is refined, the discrete equations approach the exact equations. Conservation means that the difference operator will conserve quantities in the interior region. Conservation is necessary when one attempts to capture shocks or other discontinuities. This section will state the requirements for consistency and conservation, for a more detailed discussion refer to Shapiro [27].

A sufficient condition for consistency of a finite element approximation is that the element can support a constant value of the state vector, representing a uniform flow, for all possible element shapes. This is expressed in the finite element approximation as

$$\sum_{i} N_i = 1 \quad \text{(4.38)}$$

within each element, where the sum is over all the nodes of the element.

For conservation, the sum of each column in the assembled residual matrices $R_x, R_y$ and $R_z$ must be zero at all interior points. This means that the contribution from each interior point is zero. The condition for conservation expressed in terms of the finite element approximation is that

$$\sum_{i} N_{i}^{(c)} = 1, \quad \text{(4.39)}$$

where $i$ ranges over the nodes in the element. The three-dimensional cell-vertex scheme fulfills these requirements; however, the smoothing is not conservative. Therefore, the overall algorithm is consistent but not strictly conservative.
This chapter will describe the adaptation method used in the three-dimensional finite element solver. In order to save on computational costs, one would like to adapt or refine the grid in the regions of the computational domain that have “interesting” features without having to refine the entire domain. The interesting regions of the flow often have high gradients and thus larger numerical errors, while other regions of the flow are relatively smooth. Adaptation is a method which uses these high gradients to mark a region and then increase the number of node points in that region in order to better resolve the desired flow feature and reduce the numerical errors. There are several ways in which the grid can be adapted as discussed in Section 1.1. The adaptation method used in this thesis is grid enrichment. In this method, additional nodes and elements are inserted into the grid in the regions that are of interest. Specifically the embedded grids will be aligned with the initial mesh and will have interfaces which require special treatment. The focus of this thesis is on adaptation for vortical flows.

The next section will describe the adaptation procedure used for three-dimensional elements. The adaptation criteria for vortex flows will then be discussed.

5.1 Adaptation Procedure

Adaptation is based upon knowing approximately what the flow field will look like, but not knowing exactly where the relevant features will lie. Therefore, a coarse grid is used initially. However, this grid must be fine enough to roughly capture the important features. After the solution has evolved to some point on the coarse grid, the adaptation parameter is calculated. The choice of adaptation parameter and the method by which
it is calculated will be discussed in Section 5.2. The point during the iterative solution process at which to adapt is quite arbitrary. Adaptation can be specified by the user at any point during the calculation, or alternatively, the decision to adapt can be automated. One possible choice for automatic adaptation is choosing the adaptation point as some function of the convergence parameter. Shapiro used the square root of the convergence parameter [27]. Once the adaptation parameter has been calculated, cells are flagged for refinement or unrefinement. After the initial flagging of the elements, the elements adjacent to the marked cells are also marked for division. This process of marking adjacent cells can be performed for several passes; however, for the test cases performed only one level of adjacent cells needed to be flagged for good results. The flagged cells at the coarsest level are divided first, then the cells at the next coarsest level are divided until the finest level is reached. It is at the finest level that unrefinement begins and proceeds in the same manner as the refinement process until the coarsest level is reached. After one complete pass of grid refinement and unrefinement, the grid is scanned for “holes”. A hole, essentially, is a coarse cell in the midst of a group of fine cells. This hole will reduce the accuracy in the region and defeat the purpose of adaptation. More precisely, in three dimensions an element is considered a hole when 4 or more of its faces are subdivided. If an element is a hole, then that element is subdivided. This process can result in more holes; therefore, the process of scanning for holes is repeated until no more holes occur. The analogous “island” of fine cells in the midst of coarse cells is not treated explicitly; however, the marking algorithm typically prevents this problem. In the special case where islands appear, it is not necessary to unrefine these cells since the islands of fine cells should only increase or not change the accuracy of the algorithm.

Adaptation takes about the same amount of time as several iterations, however, since adaptation is infrequent the time involved is insignificant. When adaptation is performed and all the mesh changes are completed, the pertinent grid information is recalculated. This involves renumbering the elements and nodes, recalculating the element connectivity information, and recalculating the cell lengths. The state vector is then interpolated onto the new grid. At this point, the three-dimensional interface will be discussed. In three dimensions there are two types of interface nodes, called
Figure 5.1: Cutaway View of Three-Dimensional Interface

"virtual nodes": mid-face nodes and mid-edge nodes. Figure 5.1 shows a cutaway view of the interface between the fine and coarse cells in three dimensions. In this figure, the fine cells are to the right, and the coarse region to the left. For the additional virtual nodes on the edges, the values of the state vector are initialized to the average of the two edge nodes (node 6 is the average of nodes 1 and 4). For the additional mid-face nodes, the values of the state vector are initialized to the average of the four corner nodes (node 5 is the average of nodes 1-4). Once the state vector has been initialized for all new nodes on the adapted grid, the calculation continues. On the adapted grid, conservation should be maintained since shocks may pass through an adapted region. One requirement for conservation is that the interpolation functions in each element must be continuous across the element face. This does not strictly enforce conservation, additionally what is needed for conservation is that the residuals at the mid-face and mid-edge nodes must be distributed to the "parent" nodes, i.e. the nodes before subdivision. This distribution was not implemented; however, the results from Shapiro [27] show that the typical conservation errors introduced by not performing this distribution ranged 0.1% to 0.5% for the cases used in his thesis. Since none of the cases in this
thesis have shocks, the conservation errors should be smaller.

With adaptation, cell refinement and unrefinement must obey certain rules in order for the flow solver to handle the adapted elements and remain a stable, consistent and conservative scheme. Figure 5.2 demonstrates how the refinement/unrefinement algorithm works in two dimensions, the equivalent three-dimensional figure would be more difficult to understand and not give any additional information than the two-dimensional figure. The elements marked with a \( \ast \) are marked for refinement into 4 elements, while the elements marked with a \( \circ \) are marked for unrefinement and will be fused back into one element. The main rule about refinement and unrefinement is that no more than one interface node may be on any edge of any element. Two examples of where the grid can not be adapted are shown in the figure. First, the element with the \( \times \) can not be refined since this would result in two additional nodes on the edge to the right. Similarly, the elements with the \( \circ \) can not be unrefined since two additional nodes would occur on the bottom edge. In three dimensions, the main rule is that no edge can have more than one additional node on it and that each face has only one additional mid-face node.
During adaptation, new nodes may be created at boundary surfaces. For the cases in this thesis, all boundaries are planar, therefore, simple linear fitting is sufficient for node placement on all boundaries. For the vortex cases used in this thesis the farfield boundaries will have non-uniform freestream conditions, therefore when adaptation occurs at farfield boundaries the freestream conditions must be recomputed at the new nodes on the boundaries.

The decision on how many levels of embedded adaptation and how many times to adapt is completely up to the user. The algorithm allows for any number of adaptation levels, but typically only 3 or at most 4 levels of adaptation are used. Some of the limiting factors on adaptation in three dimensions is the amount of memory available as well as the time required for the computation. How many times to adapt is a different issue. For steady problems, each succeeding level of adaptation increases the number of elements until a maximum number of elements is reached. At this point further adaptation tends to decrease the number of elements since some elements that were adapted are no longer relevant and can be unrefined. After a few adaptations, some “converged” grid will be reached. Typically, adapting twice after the maximum is reached is sufficient to obtain this converged grid.

5.2 Adaptation Criteria

This section will discuss three aspects of adaptation: adaptation quantities, adaptation measures and automatic adaptation thresholding. For clarity, all adaptation terms will be defined here. The “adaptation quantity” refers strictly to the flow quantity, the “adaptation measures” refer to the method of looking at the adaptation quantity, for example, the first difference of the adaptation quantity could be used, and the “adaptation parameter” is the value computed by an adaptation measure. Automatic thresholding is a method which determines a high and low cutoff of the adaptation parameter that will be used to determine which cells should be refined or unrefined, respectively. An optimal adaptation parameter for vortex flows would strictly identify vortices and adapt varying strength vortices equally well.
First, an adaptation quantity must be defined. The choice of adaptation quantity can be based upon the physics of the problem or upon numerics, such as the mesh-convergence [23]. The focus of this thesis is upon vortex flows. Two adaptation quantities, total pressure loss and normalized helicity, are compared to see which quantity detects vortices better and therefore will adapt the vortical flow region better. When detecting features such as shocks, the first difference or second difference of a quantity like density is used since a change in density is generated across a shock. Additionally, when multiple adaptations are performed it is important that, in the adapted region, the adaptation parameter not increase drastically thus indicating a need for further adaptation when it is not necessary [7]. For vortex flows, the use of the gradients of the adaptation quantities as well as the issue with multiple levels of adaptation will then be discussed. Finally, once the adaptation parameter has been determined, we would like to use a means of automatically selecting the adaptation thresholding limits. Autothresholding will be discussed in section 5.2.3. The next section will discuss the calculation of the adaptation quantities.

5.2.1 Adaptation Quantities

The adaptation quantities are calculated on an elemental level since adaptation refines or unrefines elements. However, the values of these quantities must also be known at the nodes for certain operations. For example, when plotting contours of the quantities the values at the nodes are needed. Similarly, if a second difference calculation is desired, the nodal values are needed. The total pressure loss is easily calculated at the nodes and then can be combined into an elemental quantity. The normalized helicity is calculated on the elemental level and then must be distributed to the nodes. The two adaptation quantities, total pressure loss and normalized helicity, will now be discussed.

Total pressure loss was chosen as one of the adaptation quantities since total pressure loss is localized to the vicinity of the vortex. Powell [22] and Batina [2] both demonstrate the use of the total pressure loss as an effective adaptation parameter for vortex flows in their 2-D calculations.
The calculation of the elemental total pressure loss will be described below. First, the total pressure loss at each node is calculated as follows:

\[ \Delta P_{0i} = \frac{P_{\infty 0} - P_{0i}}{P_{\infty 0}}. \]  

(5.1)

Then the total pressure loss in each element is determined by averaging the total pressure loss of the 8 nodes of the element.

Normalized helicity was chosen as the other adaptation quantity. The reason for choosing normalized helicity for adaptation will be discussed below. However, first the physical meaning of helicity is discussed. Helicity relates two important aspects of vortical flowfields, the velocity field and the vorticity field [16]. Helicity is defined as:

\[ H_d = \vec{V} \cdot \vec{\omega} \]  

(5.2)

where \( \vec{V} \) is the velocity vector and \( \vec{\omega} \) is the vorticity vector. Thus helicity is a measure of the alignment between the velocity and vorticity vectors. The lift generated by a body such as a delta wing is due to the production of streamwise or shed vorticity. In this case, since the vorticity is in the streamwise direction, the vorticity will be aligned with the velocity thus resulting in large values of helicity. On the other hand, the vorticity generated by shear in a boundary layer will, in general, not be aligned with the velocity. This coupled with the smaller values of velocity in the boundary layer will result in low values of helicity. For the applications of interest to us, primarily vorticity generated by lifting bodies, helicity should be a good adaptation parameter.

The magnitude of helicity indicates the strength of a vortex while the sign of helicity indicates the direction of swirl. If the vortical flowfield has a primary and secondary vortex, the helicity will be of opposite sign for each vortex while the magnitude of helicity will indicate the strength of each vortex. However, for the purposes of adaptation what is actually desired is that the two vortices look identical so that each vortex receives the same amount of adaptation. This is achieved by using normalized helicity. Normalized helicity is defined as

\[ H_n = \frac{\vec{V} \cdot \vec{\omega}}{|\vec{V}| |\vec{\omega}|} \]  

(5.3)

as previously defined in Eq. (2.5).
Normalized helicity physically represents the cosine of the angle between the velocity and vorticity and therefore has a range from -1 to 1. For a streamwise vortex, the normalized helicity will have an absolute value of unity in the core of the vortex. Thus, the normalized helicity of any number of streamwise vortices will be of comparable magnitude. In comparison, when using total pressure loss as an adaptation parameter, the total pressure losses of unequal vortices will have varying magnitudes, thus resulting in less adaptation for weaker vortices. Additionally, if total pressure loss is used as the adaptation parameter in transonic or supersonic flow applications, the total pressure loss behind a shock would indicate a need for adaptation in this region which may be unnecessary. In addition, the vortical regions in these flows would also receive less adaptation with the weaker vortices receiving little or no adaptation. Normalized helicity would be a better adaptation parameter for these cases since across a shock the vorticity is not aligned with the velocity. Thus, the magnitude of the normalized helicity will most likely not be zero but will definitely be less than one. Therefore, the streamwise vortices generated in this flow would receive the greatest amount of adaptation with other regions of the flow, such as around shocks and in the boundary layer, receiving less adaptation.

In order to calculate normalized helicity, the vorticity must first be computed. The vorticity is defined as the curl of the velocity:

$$ \vec{\omega} = \nabla \times \vec{u} $$

(5.4)

The vorticity vector can then be expanded as follows:

$$ \vec{\omega} = \left( \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right) \hat{i} + \left( \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right) \hat{j} + \left( \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \hat{k} $$

(5.5)

where $u$, $v$ and $w$ are the components of velocity. The derivative terms were computed using a finite volume approximation; however, they could have been computed using a finite element approximation by Eq. 3.9. The author chose to implement a finite volume formulation since the finite element derivative calculation was incorporated into the flow solver and therefore not accessible.
5.2.2 Adaptation Measures

Dannenhoffer [6] evaluates three adaptation measures for their effectiveness at detecting shock waves, slip lines, expansion fans and shock wakes. The three adaptation measures he uses are: the absolute value of the adaptation quantity, a first difference of the quantity and a second difference of the quantity. These three adaptation measures will be discussed for detecting vortices using total pressure loss and normalized helicity as the adaptation quantities. It should be noted that the first difference and second difference indicators examined were those implemented by Shapiro [27]. In addition, the adaptation parameter was set to have a range from 0 to 1 where a low value would indicate grid unrefinement and a high value would indicate grid refinement. This was chosen purely for convenience in implementation.

As discussed in the previous section, total pressure loss has been shown to be a good adaptation parameter for vortex flows and is evaluated in comparison to normalized helicity in this thesis. When the quantity total pressure loss was used as the adaptation parameter it was normalized in the following manner. The total pressure loss, $\Delta P_0^{(e)}$, was normalized by the maximum total pressure loss resulting in $\Delta P_0^{(e)}$. The range of $\Delta P_{0\text{norm}}^{(e)}$ should then be from 0 to 1; however, there are small regions where the total pressure loss is slightly negative (less than 5% of the maximum total pressure loss). Since these regions are non-physical and tend to be small, for adaptation any negative total pressure losses were set to zero. Therefore, it should be noted that the adaptation parameter is not the actual value of total pressure loss but a normalized value.

Normalized helicity is the other adaptation quantity and has a range from -1 to 1. However, for adaptation the sign of the normalized helicity is not important since this only indicates the direction of swirl. It is the magnitude of normalized helicity that is relevant. Therefore when normalized helicity is used as the adaptation quantity the absolute value of normalized helicity is used.

The total pressure loss and normalized helicity distributions through the core of a Lamb vortex are shown in Figs. 5.3 and 5.4. Using a first difference of either adaptation
quantity is ineffective as an adaptation parameter since the distribution of total pressure loss and normalized helicity tend to flatten out near the vortex core. When two levels of adaptation were performed using the first difference of either adaptation parameter, the second level of adaptation actually unrefined the core region. Similarly, the second difference of the adaptation quantities is ineffective at capturing the vortices since the gradient of the first difference is small throughout the flowfield.

Figure 5.3: Total Pressure Loss through a Lamb Vortex Core

The normalized adaptation quantities work best as adaptation parameters since they represent physical aspects of the vortex. Therefore, for all calculations the normalized adaptation quantities are used as the adaptation parameter.

5.2.3 Autothresholding

Automatic thresholding is a method by which the upper and lower bounds of the adaptation parameter are chosen to mark cells for refinement or unrefinement. Dannenhoffer [6] and Powell [23] have constructed automatic refinement thresholding algorithms. These algorithms will now be described and evaluated for the test cases in this thesis. Then,
an autothresholding algorithm, based upon the information from these two methods, is constructed for the test cases to follow.

First, Dannenhoffer's autothresholding algorithm will be described. Dannenhoffer's refinement thresholding consists of the following steps:

1. Construct a histogram of the adaptation parameter for all elements.

2. Construct the cumulative distribution function.

3. Determine the "knee" - arbitrarily, this is the point on the cumulative distribution function where the slope is equal to -1. This is the value of the adaptation parameter which will be used to mark cells for refinement.

Additional criteria is then imposed to guarantee that proper adaptation will occur. These include:

1. The computed threshold must exceed some specified value (one choice is that the threshold must be greater than 1.2 times the average adaptation parameter).
2. The threshold is limited to be a value which limits the number of detected elements (assume no more than 25% of the elements may be refined).

3. Unrefinement is arbitrarily set (for example, mark cells for unrefinement if the adaptation parameter is less than 1/2 the average adaptation parameter).

This algorithm was implemented and tested with both test cases. The problem with this algorithm for these test cases was that the cumulative distribution function was not smooth. Thus, when the slope was computed (by central differences) it was very irregular, i.e. the value jumped between positive and negative values often. Therefore, the point at which the slope $= -1$ was quite arbitrary and varied dramatically based on the case and the strength of the vortices used in the cases.

Now, Powell's refinement thresholding algorithm will be outlined.

1. Construct the integrated histogram (same as the cumulative distribution function) showing the number of cells flagged for refinement if the threshold is set at a certain value.

2. Do a least-squares fit of a higher-order polynomial to get a differentiable function for the histogram.

3. Take the curvature of this polynomial.

4. Set the threshold value at the lowest value of the refinement parameter that produces a local maximum in the curvature plot.

This algorithm was also implemented for both test cases as well. Smoothing the integrated histogram by using a least-squares fit removed the problem of the previous algorithm; however, again the criteria for selecting the refinement parameter tended to widely vary based upon the adaptation quantity used and whether a difference of the quantity was used. This method did not consistently produce acceptable refinement parameters for the test cases.
Neither of the above methods worked consistently for the test cases; however, some autothresholding algorithm is desired. The additional criteria that Dannenhoffer specifies in case the first method of computing the threshold limit fails worked extremely well for all test cases. The algorithm chosen is then quite simple.

1. Compute the average of the adaptation parameter over all the cells.

2. Set the refinement parameter to 1.2 times the average adaptation parameter and determine the number of cells which would be flagged for refinement.

3. If more than 25% of the cells are flagged for refinement then set the refinement parameter such that no more than 25% of the cells are refined.

4. For unrefinement, set the unrefinement parameter to 0.5 times the average refinement parameter.

This algorithm worked extremely well for all test cases, that is, there was an adequate and appropriate amount of adaptation in the vortical regions. This will be seen more clearly in the next section discussing the test cases. Thus, for all test cases the above algorithm is used for autothresholding.
Chapter 6  
Vortex Flow Results

All test cases use an analytical model of a vortex core and then propagate the vortical flow down a rectangular domain. As the vortical flow propagates downstream there is a numerical diffusion of vorticity. The two important mechanisms that cause the diffusion of vorticity are truncation error and artificial viscosity. These will be examined in the following test cases: a single vortex and two unequal vortices with opposite rotation direction. Increasing the grid resolution reduces the truncation errors and artificial viscosity, while increasing the artificial viscosity coefficients dampens out background oscillations but increases the amount of dissipation in the system.

This chapter will first describe the vortex core models used for the computations. The test cases will then be described. The effects of adaptation on each test case will then be discussed. In order to measure the effects of adaptation, total pressure loss will be used since this is a reasonably good measure of the vortex strength and location. Contour plots of total pressure loss at various stations along the domain will qualitatively show the vortex diffusion. The effects of adaptation will be measured quantitatively by examining the total pressure loss along the streamline aligned with the vortex core. Additionally, the effects of the artificial viscosity coefficient on one of the test cases will be examined.

6.1 Vortex Models

A model is needed to specify an inflow boundary condition and the freestream values at the farfield representing a vortical flow. The model consists of a core and an outer, near potential, flow. Two vortex models were used for the computations: a Rankine vortex
and a Lamb vortex. The Rankine vortex has a discontinuity in the tangential velocity at the edge of the core and has zero entropy outside of the core. The unequal vortex case used a Rankine vortex model and superposition to initialize the flowfield. Since a Rankine vortex has zero entropy outside of the core, the overall entropy distribution was assumed to simply be combination of the entropy distribution of each vortex independent of the other vortex. The vortical flowfield for the unequal vortex case will be explained in more detail when discussing the test case. A Lamb vortex was chosen for the single vortex case. The Lamb vortex is a more physically realistic model since the variations are smooth at the vortex edge as well as having an entropy distribution outside of the core. These vortex models as derived by Roberts [26] will now be described.

The flow field is initialized by specifying values for the state vector at all nodes. In order to compute the state vector, the velocity field, density and pressure must be specified at each node. The tangential velocity of a two-dimensional Rankine vortex is given by

\[
\omega_\theta = \begin{cases} 
\frac{\Gamma}{2\pi r} & \text{if } r > a \\
\frac{\Gamma}{2\pi a^2} & \text{if } r \leq a
\end{cases} \quad (6.1)
\]

where \(\omega_\theta\) is the tangential velocity, \(\Gamma\) is the circulation, \(r\) is the distance from the center of the vortex and \(a\) is the vortex core size. The tangential velocity distribution of a Rankine vortex with \(\Gamma = 0.25, a = 0.1\) is shown in Fig. 6.1. The tangential velocity of a Lamb vortex is given by

\[
\omega_\theta = \frac{\Gamma}{2\pi r} \left(1 - e^{-\left(\frac{r}{a}\right)^2}\right) \quad (6.2)
\]

and is shown for \(\Gamma = 0.25, a = 0.1\) in Fig. 6.2. For each test case, the vortex core size and the circulation will be specified. The axial velocity is assumed constant.

The density and pressure can be determined using the following assumptions. The total enthalpy is taken to be a constant equal to the freestream value throughout the domain. To determine the entropy distribution, Crocco's theorem in dimensional form is used:

\[
\nabla H = T \nabla s + \mathbf{u} \times \mathbf{j}, \quad (6.3)
\]

where \(H\) is the total enthalpy, \(T\) is the temperature, \(s\) is the specific entropy, \(\mathbf{u}\) is the
velocity, and \( \vec{\omega} = \nabla \times \vec{u} \) is the vorticity. Equation (6.3) states that the entropy is not constant, but has a has a gradient proportional to \( \vec{u} \times \vec{\omega} \). In order to determine the entropy distribution in the core, the equation of state is substituted into the non-dimensional form of Equation (6.3) to get

\[
\nabla H = \frac{p}{\rho \gamma - 1} \nabla s + \vec{u} \times \vec{\omega} \tag{6.4}
\]

where \( s \) has been non-dimensionalized by \( c_v \), the specific heat at constant volume. Now, using the assumption that the total enthalpy is a constant, Eq. (6.4) may be further simplified by using the Eqns. (2.3) and (2.4) to eliminate \( \rho \) and \( p \). The resulting equation is

\[
\nabla s = -\frac{\gamma (\vec{u} \times \vec{\omega})}{H - \frac{\vec{u} \cdot \vec{u}}{2}}.
\tag{6.5}
\]

\( H \) is determined by the freestream Mach number and is given by

\[
H = \frac{1}{\gamma - 1} + \frac{1}{2} M_c^2.
\tag{6.6}
\]

With \( H \) known and \( \vec{u} \) determined from the velocity field of the vortices of the prescribed flow field, \( s \) can be found by numerically integrating Eq. (6.5) from large \( r \) (where \( s \) is known to be zero) to \( r = 0 \). For all test cases, the flow is axisymmetric about the vortex core, thus resulting strictly in radial variations in the entropy. For a Rankine vortex, Eq.(6.5) for an axisymmetric vortex reduces to

\[
\frac{\partial s}{\partial r} = \frac{-\gamma}{2} \frac{\Gamma^2}{8\pi^2 \alpha^4 (r^2 - \alpha^2)} \tag{6.7}
\]

for \( r \leq \alpha \). Outside of the core, \( r > \alpha \), the entropy is zero. The entropy distribution is plotted in Fig. 6.3 for \( \Gamma = 0.25, \alpha = 0.1 \). For a Lamb vortex the radial entropy distribution is given by

\[
\frac{\partial s}{\partial r} = \frac{-\gamma}{2} \frac{\Gamma^2}{8\pi^2 \alpha^4} \frac{e^{-(\xi/e)} \left\{ 1 - e^{-(\xi/e)} \right\}}{1 - \frac{\Gamma^2}{8\pi^2 \alpha^4} \left\{ 1 - e^{-(\xi/e)} \right\}^2}.
\tag{6.8}
\]

and is shown in Fig. 6.4 for \( \Gamma = 0.25, \alpha = 0.1 \).

Once the entropy distribution is known, the entire prescribed state vector can be determined. To get the density, the equation of state and the constant total enthalpy
assumption are used. The equation of state describing the relation between pressure, density, and entropy is

\[ p = \rho^\gamma e^* \]  

(6.9)

It is convenient to define a new variable

\[ S = e^* \]  

(6.10)

and write the equation of state as

\[ \frac{P}{\rho^\gamma} = S. \]  

(6.11)

This definition gives \( S \) equal to 1 outside the rotational core, and greater than 1 inside the core. From the known velocity field and total enthalpy, the ratio of pressure to density is

\[ \frac{P}{\rho^\gamma} = \left( H - \frac{\vec{u} \cdot \vec{u}}{2} \right) \]  

(6.12)

Combining Eqs. (6.11) and (6.12) yields the following equation for the density

\[ \rho = \left\{ \frac{\gamma - 1}{S} \left( H - \frac{\vec{u} \cdot \vec{u}}{2} \right) \right\}^{\frac{1}{\gamma - 1}}. \]  

(6.13)

Using Eq. (6.13) the density distribution is determined, then Eq. (6.12) can be used to compute the pressure distribution. The total energy

\[ E = H - \frac{P}{\rho} \]  

(6.14)

can then be readily found. With the density, the velocity and the total energy known at each cell in the domain, the prescribed flow state vector is known.

### 6.2 Vortex Test Cases

For both test cases, the domain is a square cross-sectional area with all farfield boundaries. When using characteristic theory for the boundary conditions, the freestream values must be specified at the farfield boundaries. Therefore some approximation of the freestream values at the farfield is needed. The freestream values of the state vector at the farfield boundaries are determined by solving for the flowfield through the entire
domain. The values at the boundaries from this solution are then set to the freestream values. For the single vortex case, the exact solution is simply the vortex through the domain. For the unequal vortices case, the "exact" solution is slightly more complex and will discussed within Section 6.2.2. The physical dimensions of the domain are different for the two test cases, thus these dimensions will be given in the discussion of each case. The freestream Mach number for both cases is $M_\infty = 1.5$ with a $CFL = 0.33$ and second difference damping $\nu_2 = 0.0$ since no shocks exist in the flow. (The $CFL$ limit should be approximately 1.63 according to a 1-D stability analysis; however, the $CFL$ used was found roughly to be the maximum for all cases.) The specifics of each test case, such as the grid density, vortex size and circulation, and fourth difference damping, will be discussed with the results of each test case.

6.2.1 Single Vortex

This test case will discuss two primary issues: how adaptation can be used to reduce the diffusion errors and the effects of fourth difference damping. The first issue is addressed using a single Lamb vortex introduced into the domain through the inlet boundary condition. The vortex should propagate through the domain without any variation from the initial condition. However, due to the artificial viscosity that must be added for stability and the numerical accuracy of the algorithm, the vortex will experience numerical diffusion. Two levels of adaptation based on total pressure loss and normalized helicity were performed. The effects of adaptation are examined at each level with the errors measured by the total pressure loss along the vortex core. The effects of varying the fourth difference damping used a single Lamb vortex imposed through the entire channel with one level of adaptation performed based on total pressure loss. The adaptation was based upon the initial condition. This case was actually computed before the multiple level adaptive cases and helped determine the fourth difference smoothing coefficient for the adaptive cases. However, it will be shown that the effects of varying the fourth difference smoothing is a secondary concern and thus will be discussed after the adaptive cases.
The domain for this test case has a width and height with a unit length of 1 (with the crossflow coordinates $-0.5 < y, z < 0.5$) and the domain length is a distance of 2 (with the streamwise coordinate $0.0 < z < 2.0$). The vortex is imposed in the center of the domain with a core radius equal to 0.1 and a circulation equal 0.25 rotating counterclockwise. This results in a total pressure loss of 14.01% at the vortex center.

To examine the effects of grid adaptation five variations were run: an initial grid, one level of adaptation based on the two adaptation quantities, i.e. total pressure loss and normalized helicity, and two levels of adaptation based on these quantities. Adaptation does not need to be performed on a converged solution; however, a converged solution is desired to measure the diffusion errors. Therefore, since a converged solution is desired, all adaptations were performed on this solution. For each of these variations the fourth difference smoothing coefficient, $\nu_4$, was set to 0.005. The initial grid for this case is $20 \times 20 \times 20$ (8000 elements). Based on the core radius and grid spacing, the vortex core is contained within a $4 \times 4$ cell area in the crossflow plane. The initial grid is uniform with a $y-z$ slice shown in Fig. 6.5 and a $z-z$ slice shown in Fig. 6.6 (a $z-y$ grid slice is identical to an $z-z$ slice). The initial condition is a Lamb vortex imposed at the inlet boundary. The steady-state solution was achieved in approximately 600 iterations with the $\log(RMS) = -6.3$. The contour plots of total pressure loss at $z = 0.0$ (Fig. 6.7), $z = 0.5$ (Fig. 6.8), $z = 1.0$ (Fig. 6.9) and $z = 2.0$ (Fig. 6.10) show the effects of numerical dissipation and truncation error on the vortex. The contour plots of normalized helicity at these stations are shown in Figs. 6.11 - 6.14. The normalized helicity contours are shown since this is quantity is used as an adaptation parameter. These contours show that normalized helicity is less sensitive to numerical diffusion errors. A quantitative analysis of the diffusion errors will be discussed after all variations are introduced.

Adaptation using the total pressure loss parameter was performed based on this converged solution. When total pressure loss was used as the adaptation parameter, refinement was indicated when the adaptation parameter was greater than 0.04245 (note: this is not the total pressure loss, but the normalized total pressure loss as described in Section 5.2.2). Unrefinement is not permitted since unrefinement beyond the initial grid
is not allowed. The grid refinement algorithm marked 872 elements for refinement, thus resulting in an adaptive grid with 14,104 elements. The $y - z$ grid slices of the adaptive grid at $z = 0.0$, $z = 1.0$ and $z = 2.0$ are shown in Figs. 6.15 - 6.17. The steady-state solution was reached after approximately 800 iterations with a $\log(RMS) = -6.5$. The contour plots of total pressure loss at the various stations are shown in Figs. 6.18 - 6.21. Qualitatively we can see that the errors in total pressure loss have been reduced. When a second level of adaptation was performed, cells were marked for refinement when the adaptation parameter was greater than 0.1711 and unrefinement when less then 0.06111. This resulted in 3,184 elements being marked for refinement and 176 elements marked for unrefinement. The final grid with two levels of adaptation contained 35,160 elements with the $y - z$ grid slices at $z = 0.0$, $z = 1.0$ and $z = 2.0$ shown in Figs. 6.22 - 6.24. A converged solution was reached in approximately 1100 iterations with a $\log(RMS) = -6.6$. The contour plots at the $z = 0.0$ (Fig. 6.25) and $z = 2.0$ (Fig. 6.26) show that the diffusion errors have been greatly reduced over the initial grid and even one level of adaptation.

When normalized helicity was used as the adaptation parameter, refinement was indicated when the adaptation parameter was greater than 0.0806. The grid refinement algorithm marked 1,116 elements for refinement resulting in an adaptive grid with 15,840 elements, slightly more elements than the total pressure loss adaptive grid. The $y - z$ grid slices at the $z = 0.0$, $z = 1.0$ and $z = 2.0$ are shown in Figs. 6.27 - 6.29. The assymetry at $z = 0.0$ and $z = 2.0$ most likely is due to these values being extremely close to the threshold value with only some of the cells being marked for refinement. The steady-state solution was again reached in approximately 800 iterations with a $\log(RMS) = -6.5$. The helicity contour plots at the $z = 0.0$, $z = 1.0$ and $z = 2.0$ are shown in Figs. 6.30 - 6.32. When a second level of adaptation was performed, the algorithm marked cells for refinement when the adaptation parameter is greater than 0.13185 (3,492 elements were marked) and unrefinement when the adaptation parameter was less than 0.05494 (400 elements were marked). The resulting adaptive grid has 37,484 elements. The $y - z$ grid slices at the $z = 0.0$ and $z = 2.0$ are shown in Fig. 6.33 and Fig. 6.34. This case converged with a $\log(RMS) = -6.6$ in approximately 1100 iterations. The helicity contour plots are virtually the same through the domain.
However, the total pressure loss contour plots show the diffusion errors. Therefore, the total pressure loss contours at the $z = 0.0$ and $z = 2.0$ for this adaptive grid are shown in Fig. 6.35 and Fig. 6.36.

The convergence histories for the above five variations are shown in Fig. 6.37. As can be seen in this figure, after an adaptation is performed the error jumps over two orders of magnitude. This is because grid refinement reduces both the truncation errors and the smoothing errors. Thus, the newly interpolated solution on the adaptive grid has errors that can be reduced due to the improved accuracy.

The total pressure loss along the vortex center for these five variations is shown in Fig. 6.38. There are some odd-even oscillations near the outlet in Fig. 6.38 which most likely is an effect due to the fourth difference smoothing boundary condition. Shapiro [27] has similar oscillations near the boundaries for a two-dimensional wedge case and recommends using as little fourth difference smoothing as possible. Shapiro found that too much fourth difference smoothing resulted in a non-physical increase in entropy for the wedge case. Since the oscillations occur only near the boundaries, this indicates that the fourth difference smoothing stencil may be improved upon at the boundaries.

A summary of the errors in total pressure loss at the $z = 0.5$, $z = 1.0$ and $z = 2.0$ is given in Table 6.1. The percent error is the difference between the inlet total pressure loss (14.01% for all cases) and the station total pressure loss divided by the inlet total pressure loss. The cases with adaptation based on normalized helicity have a smaller error since there are more cells adapted thus reducing the truncations errors. Both cases with two levels of adaptation show a slight rise in total pressure loss near the outlet. The case based on normalized helicity actually shows a total pressure loss at the outlet, $z = 2.0$, slightly greater than the inlet ($z = 0.0$) total pressure loss. Since all of the errors with two levels of adaptation are less than 1%, it is presumed that the accuracy of the algorithm has been reached and that this is not significant.

The case with one level of adaptation based on normalized helicity shows an average factor of improvement of 5.7 over the unadapted case. With a second level of adaptation the average factor of improvement over one level of adaptation is 4.5 (this excludes the
z = 2.0 value due to the increase in total pressure loss). Similarly, the case with one level of adaptation based on total pressure loss shows an average factor of improvement of 4.0 over the unadapted case. With two levels of adaptation based on total pressure loss the average factor of improvement is 6.9 over one level of adaptation. The cell-vertex scheme without damping is second-order accurate, meaning that if the cell size is reduced by a factor of 2 the accuracy will increase by a factor of 4. The fourth difference damping is third-order accurate indicating that reducing the cell size by a factor of 2 will increase the accuracy by a factor of 8. Therefore, the normalized helicity cases have an order of accuracy from 2.2 to 2.5, while the total pressure loss cases have an order of accuracy from 2 to 2.8. The interpretation of the order of accuracy is important. The scheme is second-order accurate therefore it would be incorrect to say that the order of accuracy is greater than second-order. However, the order of accuracy shows the importance of the smoothing error to the truncation error of the algorithm. If the order of accuracy is significantly greater than 2 then the smoothing term is a significant contributor to the error. Thus, these order of accuracy estimates indicate that the smoothing error is significant in comparison to the truncation error.

A key result from this test case is the reduction in the diffusion errors. The initial grid had 5 points across the vortex resulting in a total pressure loss error of 25% at a distance of 20 core radii downstream. With two levels of adaptation this error was reduced to under 1% throughout the entire channel with 23 points across the vortex. An equivalent globally refined grid would have required 512,000 elements in comparison to the roughly 36,000 elements used in these cases (roughly 7% of the globally refined mesh). Thus, adaptation was able to greatly reduce the numerical errors at a relatively low cost in terms of memory and CPU time.

The effects of varying fourth difference damping will now be examined using the previously described case with the adaptation based on the initial condition and total pressure loss as the adaptation parameter. The adaptive grid has 12,480 elements. Three values of \( \nu_4 \) were used: 0.001, 0.005, 0.01. The convergence histories for each case are shown in Fig. 6.39. The \( \nu_4 = 0.001 \) case reaches a \( \log(RMS) = -6.4 \) after 1000 iterations. The \( \nu_4 = 0.005 \) case levels off to a \( \log(RMS) = -6.5 \) after approximately
650 iterations while the $\nu_4 = 0.01$ case reaches a $\log(RMS) = -8.5$ at 900 iterations and continues to decrease. The cases with smaller values of damping tend to stabilize at a higher $RMS$ value and take much longer to reach a converged solution.

The total pressure loss along the vortex center for these three cases with varying smoothing is shown in Fig. 6.40. As expected the smaller the value of $\nu_4$, the smaller the total pressure loss errors. A summary of the total pressure loss at the various stations along the domain is given in Table 6.2. If $\nu_4$ is increased by a factor of 10 (from 0.001 to 0.01), then the percent errors increase by an average factor of 3.21 and a peak increase at the outlet by a factor of 4.62 (a distance of 20 core radii). Even with a "high" value of $\nu_4$ the percent errors are less than 10%.

Table 6.2: Single Vortex - Comparison of $\Delta P_0$ Errors at Stations along Domain for Varying $\nu_4$

From this test case we can conclude that grid refinement is a much larger factor than fourth difference damping in reducing the numerical diffusion the vortex experiences. Additionally, smaller values of the smoothing coefficient resulted in a slower convergence.
rate to a steady-state solution. Thus, this would indicate that the value of the smoothing coefficient is a secondary concern and that in order to reduce numerical errors the focus should be upon grid refinement.
Figure 6.1: Tangential Velocity Distribution for a Rankine Vortex

Figure 6.2: Tangential Velocity Distribution for a Lamb Vortex
Figure 6.3: Entropy Distribution for a Rankine Vortex

Figure 6.4: Entropy Distribution for a Lamb Vortex
Figure 6.5: Single Vortex - Initial Grid, y – z Slice

Figure 6.6: Single Vortex - Initial Grid, z – z Slice
Figure 6.7: Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Initial Grid

Figure 6.8: Single Vortex - $\Delta P_0$ Contours at $z = 0.5$, Initial Grid
Figure 6.9: Single Vortex - $\Delta P_0$ Contours at $z = 1.0$, Initial Grid

Figure 6.10: Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Initial Grid
Figure 6.11: Single Vortex - \( H_n \) Contours at \( z = 0.0 \), Initial Grid

Figure 6.12: Single Vortex - \( H_n \) Contours at \( z = 0.5 \), Initial Grid
Figure 6.13: Single Vortex - $H_n$ Contours at $z = 1.0$, Initial Grid

Figure 6.14: Single Vortex - $H_n$ Contours at $z = 2.0$, Initial Grid
Figure 6.15: Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y-z$ Slice at $z = 0.0$

Figure 6.16: Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y-z$ Slice at $z = 1.0$
Figure 6.17: Single Vortex - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at $z = 2.0$
Figure 6.18: Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (1 level) Based on $\Delta P_0$

Figure 6.19: Single Vortex - $\Delta P_0$ Contours at $z = 0.5$, Adaptive Grid (1 level) Based on $\Delta P_0$
Figure 6.20: Single Vortex - $\Delta P_0$ Contours at $z = 1.0$, Adaptive Grid (1 level) Based on $\Delta P_0$

Figure 6.21: Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (1 level) Based on $\Delta P_0$
Figure 6.22: Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0, y - z$ Slice at $z = 0.0$

Figure 6.23: Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0, y - z$ Slice at $z = 1.0$
Figure 6.24: Single Vortex - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at $z = 2.0$
Figure 6.25: Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (2 levels) Based on $\Delta P_0$

Figure 6.26: Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (2 levels) Based on $\Delta P_0$
Figure 6.27: Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y-z$ Slice at $z = 0.0$

Figure 6.28: Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y-z$ Slice at $z = 1.0$
Figure 6.29: Single Vortex - Adaptive Grid (1 level) based on $H_n$, $y-z$ Slice at $z = 2.0$
Figure 6.30: Single Vortex - $H_n$ Contours at $z = 0.0$, Adaptive Grid (1 level) Based on $H_n$

Figure 6.31: Single Vortex - $H_n$ Contours at $z = 1.0$, Adaptive Grid (1 level) Based on $H_n$
Figure 6.32: Single Vortex - $H_n$ Contours at $z = 2.0$, Adaptive Grid (1 level) Based on $H_n$
Figure 6.33: Single Vortex - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at $x = 0.0$

Figure 6.34: Single Vortex - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at $x = 2.0$
Figure 6.35: Single Vortex - $\Delta P_0$ Contours at $z = 0.0$, Adaptive Grid (2 levels) Based on $H_n$

Figure 6.36: Single Vortex - $\Delta P_0$ Contours at $z = 2.0$, Adaptive Grid (2 levels) Based on $H_n$
Figure 6.37: Single Vortex - Convergence Histories for Adaptive Cases

Figure 6.38: Single Vortex - $\Delta P_0$ along Vortex Core for Adaptive Cases
Figure 6.39: Single Vortex - Convergence Histories for Varying $\nu_4$ Cases

Figure 6.40: Single Vortex - $\Delta P_0$ along Vortex Core for Varying $\nu_4$ Cases
6.2.2 Two Unequal Vortices with Opposite Rotation Directions

This case was selected to model a leading edge vortex and a secondary vortex at the trailing edge of a delta wing. The primary vortex will be much stronger than the secondary vortex as well as having an opposite direction of rotation. An example of this was shown in the Section 1.1 with the primary and secondary vortex for the NTF delta wing. The total pressure loss contours at $z = 1.2$ (120% chord) are shown in Fig. 6.41. The total pressure loss in the primary vortex core is 61.4% and in the secondary vortex core is 36.1% [18]. This case is not only much closer to a physical application, but also tests how well the adaptation works in detecting vortices of various strengths. The purpose of using adaptation is to detect and refine regions of the flow where vorticity exists. It is important that the adaptation criteria be able to detect regions of smaller vorticity since the initial grid resolution is often not fine enough to accurately capture weaker vortices. This test case will be used to evaluate the adaptation quantities, total pressure loss and normalized helicity, to determine which method is better at capturing vortex flows.

Figure 6.41: Total Pressure Loss Contour at $z = 1.20$ for the NTF Delta Wing

The physical dimensions of the domain range from $\pm 0.75$ in the crossflow plane and
from 0.0 to 1.5 in the streamwise direction. The physical dimensions of the crossflow plane were increased over the single vortex case since two vortices are now in the domain. The initial grid has $30 \times 30$ elements in the crossflow plane and 20 elements in the streamwise direction (18000 elements). A $y-z$ slice is shown in Fig. 6.42 and a $x-z$ slice is shown in Fig. 6.43. Two Rankine vortices are imposed at the inlet of the domain and allowed to propagate down the domain. The reason for choosing a Rankine vortex model will be explained shortly. Both vortices have a core radius equal to 0.1 and are located at $y = \pm 0.1, z = -0.1$, the vortex at $y = -0.1$ has the circulation set equal to $\Gamma_1 = 0.12$ (counterclockwise rotation) and the other vortex at $y = 0.1$ has the circulation equal to $\Gamma_2 = -0.08$ (clockwise rotation). The values of circulation were chosen to achieve total pressure losses in the vortex cores on the same order of magnitude as Goodsell's main vortex and roll-up vortex [13]. In Goodsell's Euler calculations for a delta wing at $M_\infty = 1.3$ and $\alpha = 10^\circ$ one station off the trailing edge the main vortex has a total pressure loss of 42% in the core and the roll-up vortex has a total pressure loss of 20%. This test case is not trying to model this case, but simply use typical values that are found in applications. The values of circulation chosen for this test case results in a total pressure loss of 40.87% for the stronger vortex and a total pressure loss of 20.61% for the weaker vortex.

A two-dimensional solution is used for the initial condition at the inlet as well as for the freestream values at the farfield boundaries. In two dimensions, the two unequal vortices in this test case would remain a constant distance apart and move in circular paths about some centroid of vorticity with constant angular velocity [1]. For these two vortices the centroid of vorticity will be to the left of the stronger vortex. The exact solution in three-dimensions is not easily determined analytically; however, we can approximate the three-dimensional solution as a collection of two-dimensional slices. Since there are streamwise variations in the solution, the two-dimensional Euler equations not do satisfy the three-dimensional Euler equations. Therefore, this method is only an approximation.

As mentioned previously, two Rankine vortices are used. The velocity fields of each vortex are superimposed as well as the entropy distributions. For the solution to
be exact in two dimensions, Crocco's theorem relates the entropy distribution to the velocity and vorticity fields. Therefore, the vorticity field of the superimposed velocity field should be calculated. An approximation was made at this point that the entropy distribution of each vortex was additive. With Rankine vortices, entropy and vorticity are zero outside of the core; therefore, since the two vortex cores are not overlapping the entropy distribution is unaffected by the other vortex. The resulting initial condition has a slight negative total pressure loss of 3% between the two vortices. This most likely is due to the entropy approximation. Since the velocity fields of the vortices interact, the entropy distribution most likely is not zero outside of the core which could result in the slight negative total pressure loss.

To compute the freestream values at the farfield boundaries, the motion of the vortices through the domain must be determined. First, the centroid of vorticity is found from equating the angular velocities about the centroid of vorticity. The angular velocities are given by:

\[ \omega = \frac{u_{s_1}}{y_{cv} + b} = \frac{u_{s_2}}{y_{cv} - b} \quad (6.15) \]

\[ u_{s_1} = -\frac{\Gamma_2}{2\pi b} \quad (6.16) \]

\[ u_{s_2} = \frac{\Gamma_1}{2\pi b} \quad (6.17) \]

where \( y_{cv} \) is the centroid of vorticity (in this case it will be on the y-axis), \( b \) is the distance between the vortices, \( \omega \) is the angular velocity and \( \Gamma_1 \) and \( \Gamma_2 \) are the circulations of the vortices. The centroid of vorticity is then

\[ y_{cv} = \frac{\Gamma_2 - \Gamma_1}{\Gamma_1 + \Gamma_2} b = -5b. \quad (6.18) \]

The angle that the vortices rotate through about the centroid of vorticity as they propagate down the domain is

\[ \theta = \frac{\omega \gamma}{M_{\infty}} = 13.68^\circ. \quad (6.19) \]

Thus the position of the vortices in the crossflow plane can be updated for each streamwise position. The flowfield can then be computed based upon the new position of the vortices and the values computed at the boundaries are set as the freestream values.
The reason the farfield model is important is that normalized helicity is extremely sensitive to velocity variations since the vorticity is the curl of the velocity. If a poor farfield model is used then the normalized helicity is large at the boundaries and adapts this region.

Similar to the single vortex case, five variations were run: an unadapted case, one level of adaptation based on total pressure loss, one level of adaptation based on normalized helicity, two levels of adaptation based on total pressure loss and normalized helicity. The fourth difference smoothing coefficient was set to 0.005 for all variations.

As mentioned above, the initial grid for this case was $20 \times 30 \times 30$ elements (18,000 elements). The steady-state solution was achieved in approximately 400 iterations with a $\log(RMS) = -5.1$. The total pressure loss contours at the inlet and outlet are shown in Figs. 6.44 and 6.45. These two vortices have large diffusion errors largely due to the inadequate grid resolution. The weaker vortex is quite diffused at the downstream boundary. Since normalized helicity is the other adaptation parameter the contour plots of this quantity at the inlet and outlet are shown in Figs. 6.46 and 6.47. These contour plots show that normalized helicity distinguishes both vortices equally well with the only difference being in the sign (indicating the direction of rotation). A quantitative analysis of the errors in total pressure loss will be discussed after all variations are introduced.

Adaptation based on total pressure loss was performed on this converged solution. Refinement was indicated with the adaptation parameter was greater than 0.0204, resulting in 1,270 elements marked for refinement. The resulting adaptive grid contained 26,960 elements. The $y-z$ grid slices at the inlet and outlet are shown in Figs. 6.48 and 6.49. The adapted region at the outlet shows the diffused solution from the initial grid. The steady-state solution on this grid was reached after approximately 600 iterations with a $\log(RMS) = -5.35$. The contour plots of total pressure loss at the inlet and outlet are shown in Figs 6.50 and 6.51. As expected, qualitatively the total pressure loss errors have been reduced. The weaker vortex is much more distinguishable at the outflow boundary. When a second level of adaptation is performed based upon this
converged solution, 5,271 elements were marked for refinement and 119 elements were marked for unrefinement. The resulting adaptive grid contained 63,388 elements. The \( y - z \) grid slices at the inlet and outlet are shown in Figs. 6.52 and 6.53. The converged solution was reached in approximately 1000 iterations with a \( \log(RMS) = -5.45 \). The contour plots of total pressure loss at the inlet (Fig. 6.54) and outlet (Fig. 6.55) show that the numerical errors have been greatly reduced over the coarser grids.

When normalized helicity was used as the adaptation parameter, the first level of adaptation marked 3,216 elements for refinement resulting in an adaptive mesh with 44,012 elements. This is approximately 1.4 times the number of elements in the adaptive mesh based on total pressure loss. The \( y - z \) grid slices at the inlet and outlet are shown in Figs. 6.56 and 6.57. When using normalized helicity as the adaptation parameter, a larger region is marked for refinement since the normalized helicity distribution spreads farther into the domain than total pressure loss. Additionally, normalized helicity is less sensitive to numerical diffusion errors, thus it will mark the vortex core throughout the length of the domain equally well. Normalized helicity is a more sensitive measure of the vortex; however, normalized helicity also is sensitive to small velocity gradients. For example, the normalized helicity is significant at the boundaries where the farfield conditions are not perfectly matched with the solution in the interior. Thus there are small regions along the boundaries that normalized helicity marked for adaptation. The total pressure loss contours at the inlet and outlet are shown in Figs. 6.58 and 6.59 while the normalized helicity contours at the inlet and outlet are shown in Figs. 6.60 and 6.61. The total pressure loss contours show that the errors through the channel have been significantly reduced. A second level of adaptation based on normalized helicity marks 7,850 elements for refinement and 1,360 elements for unrefinement. The resulting adaptive grid has 91,262 elements. The \( y - z \) grid slices at the inlet and outlet are shown in Figs. 6.62 and 6.63. Again, there is some adaptation at the boundaries due to the velocity mismatch between the boundary condition and the interior. The total pressure loss contours at the inlet (Fig. 6.64) and outlet (Fig. 6.65) show that the errors are extremely small. The reductions in the total pressure loss errors will be discussed shortly.
The convergence histories for the above five variations are shown in Fig. 6.66. The log(RMS) levels off at around -5.5 for these cases. The reason the RMS error does not drop to machine zero is most likely due to the errors at the boundaries. This was also seen in the normalized helicity calculation. For these five variations, the total pressure loss along the stronger vortex core is shown in Fig. 6.67 and the weaker vortex core is shown in Fig. 6.68. Without adaptation, the total pressure loss is significant for both vortices. However, with one level of adaptation the errors are significantly reduced. With two levels of adaptation, the total pressure loss errors become negligible.

The inlet total pressure loss for the stronger vortex is 40.87% and for the weaker vortex is 20.61%. A summary of the total pressure losses at \( x = 0.5, z = 1.0, \) and \( x = 1.5 \) for the stronger vortex is given in Table 6.3, while the total pressure losses at these stations for the weaker vortex are given in Table 6.4. It should be noted that the exact solution is not known. Therefore, the "errors" are based upon the assumption that the strengths of the vortices remain constant through the domain. The percent error is then the difference between the inlet total pressure loss and the station total pressure loss divided by the inlet total pressure loss. The total pressure loss errors for the stronger vortex are given in Table 6.5 and for the weaker vortex are given in Table 6.6. These tables show that with one level of adaptation normalized helicity is a slightly better adaptation parameter than total pressure loss. This is most likely due to normalized helicity adapting a slightly larger region. When two levels of adaptation are performed both adaptation parameters perform equally well. It will be noticed that both adaptation parameters result in a slight increase in the total pressure loss above the inlet total pressure loss (less than 2% error). This may be non-physical, but one possible explanation is that the initial condition uses a Rankine vortex core structure which does not satisfy the two or three-dimensional Euler equations. From the unadapted grid to the one level of adaptation grids, the average factor of improvement was 3.35. This is equivalent to an order of accuracy of 1.75. Thus, this indicates that the truncation errors of the algorithm are more significant than the smoothing errors. With a second level of adaptation, it is assumed that the accuracy of the algorithm has been reached.
Table 6.3: Unequal Vortices - Comparison of $\Delta P_0$ for Stronger Vortex at Stations along Domain

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Elements</th>
<th>$\Delta P_0$, $z = 0.5$</th>
<th>$\Delta P_0$, $z = 1.0$</th>
<th>$\Delta P_0$, $z = 1.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>18000</td>
<td>21.25%</td>
<td>17.30%</td>
<td>14.90%</td>
</tr>
<tr>
<td>$\Delta P_0$ (1 lvl)</td>
<td>26960</td>
<td>35.50%</td>
<td>32.46%</td>
<td>29.56%</td>
</tr>
<tr>
<td>$H_n$ (1 lvl)</td>
<td>44012</td>
<td>37.55%</td>
<td>32.56%</td>
<td>29.54%</td>
</tr>
<tr>
<td>$\Delta P_0$ (2 lvls)</td>
<td>63388</td>
<td>41.46%</td>
<td>41.19%</td>
<td>40.49%</td>
</tr>
<tr>
<td>$H_n$ (2 lvls)</td>
<td>91262</td>
<td>41.58%</td>
<td>41.54%</td>
<td>40.93%</td>
</tr>
</tbody>
</table>

Table 6.4: Unequal Vortices - Comparison of $\Delta P_0$ for Weaker Vortex at Stations along Domain

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Elements</th>
<th>$\Delta P_0$, $z = 0.5$</th>
<th>$\Delta P_0$, $z = 1.0$</th>
<th>$\Delta P_0$, $z = 1.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>18000</td>
<td>9.60%</td>
<td>8.03%</td>
<td>7.50%</td>
</tr>
<tr>
<td>$\Delta P_0$ (1 lvl)</td>
<td>26960</td>
<td>18.15%</td>
<td>15.32%</td>
<td>13.46%</td>
</tr>
<tr>
<td>$H_n$ (1 lvl)</td>
<td>44012</td>
<td>18.20%</td>
<td>15.44%</td>
<td>13.69%</td>
</tr>
<tr>
<td>$\Delta P_0$ (2 lvls)</td>
<td>63388</td>
<td>21.02%</td>
<td>21.01%</td>
<td>20.50%</td>
</tr>
<tr>
<td>$H_n$ (2 lvls)</td>
<td>91262</td>
<td>20.87%</td>
<td>20.97%</td>
<td>20.43%</td>
</tr>
</tbody>
</table>

6.3 Summary

This section presented two test cases: a single Lamb vortex and two unequal strength Rankine vortices. On the initial grid, each vortex had only 5 points across the grid. For the single vortex case, the total pressure errors at the outlet were approximately 25%. For the two unequal vortices, the total pressure loss errors were approximately 63%. The two unequal vortices have larger errors since there are streamwise variations. With one level of adaptation, the adaptive grid size was a fraction of a globally refined grid, however, the errors were reduced by more than a factor of 4 for the single vortex case and by a factor of 3.35 for the unequal vortex case. With one level of adaptation the number of nodes across the vortex core was 13. The total pressure loss errors for the
Adaptation Elements | % Error, $z = 0.5$ | % Error, $z = 1.0$ | % Error, $z = 1.5$
--- | --- | --- | ---
None | 18000 | 48.01% | 57.67% | 63.54%
$\Delta P_0$ (1 lvl) | 26960 | 13.14% | 20.58% | 27.67%
$H_n$ (1 lvl) | 44012 | 8.12% | 20.34% | 27.72%
$\Delta P_0$ (2 lvls) | 63388 | -1.44% | -0.78% | 0.94%
$H_n$ (2 lvls) | 91262 | -1.75% | -1.64% | -0.15%

Table 6.5: Unequal Vortices - Comparison of $\Delta P_0$ Errors for Stronger Vortex at Stations along Domain

| Adaptation | Elements | % Error, $z = 0.5$ | % Error, $z = 1.0$ | % Error, $z = 1.5$
--- | --- | --- | --- | ---
None | 18000 | 53.42% | 61.06% | 63.61%
$\Delta P_0$ (1 lvl) | 26960 | 11.96% | 25.67% | 34.68%
$H_n$ (1 lvl) | 44012 | 11.69% | 25.09% | 33.60%
$\Delta P_0$ (2 lvls) | 63388 | -1.98% | -1.94% | 0.56%
$H_n$ (2 lvls) | 91262 | -1.26% | -1.77% | 0.85%

Table 6.6: Unequal Vortices - Comparison of $\Delta P_0$ Errors for Weaker Vortex at Stations along Domain

The single vortex case were on the order of 4% to 5% while for the unequal vortices case the total pressure loss errors were over 25% for both the stronger and weaker vortices. After two levels of adaptation, the total pressure loss errors were reduced to within ±2%. The number of nodes across the vortex core with two levels of adaptation was approximately 23 nodes.

**Briefly comparing the adaptation parameters, the normalized helicity of each vortex is of comparable magnitude thus each vortex will receive an equal amount of adaptation. Total pressure loss will purely reflect the strength of the vortex and thus could vary quite dramatically. Additionally, the normalized helicity should be relatively insensitive to numerical diffusion errors as opposed to total pressure loss which again displays the**
diffusion errors. In concluding this section, the results from the test cases show that total pressure loss and normalized helicity work equally well at resolving the vortical fields in these test cases. This is due to the test cases being relatively simple flowfields. For the reasons explained above and in Section 5.2.1, it is believed that in realistic applications normalized helicity will be a better adaptation parameter than total pressure loss.
Figure 6.42: Unequal Vortices - Initial Grid, $y-z$ Slice

Figure 6.43: Unequal Vortices - Initial Grid, $x-z$ Slice
Figure 6.44: Unequal Vortices - $\Delta P_0$ Contours at Inlet, Initial Grid

Figure 6.45: Unequal Vortices - $\Delta P_0$ Contours at Outlet, Initial Grid
Figure 6.46: Unequal Vortices - $H_n$ Contours at Inlet, Initial Grid

Figure 6.47: Unequal Vortices - $H_n$ Contours at Outlet, Initial Grid
Figure 6.48: Unequal Vortices - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at $x = 0.0$

Figure 6.49: Unequal Vortices - Adaptive Grid (1 level) based on $\Delta P_0$, $y - z$ Slice at Outlet
Figure 6.50: Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (1 level) Based on $\Delta P_0$

Figure 6.51: Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (1 level) Based on $\Delta P_0$
Figure 6.52: Unequal Vortices - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at Inlet

Figure 6.53: Unequal Vortices - Adaptive Grid (2 levels) based on $\Delta P_0$, $y - z$ Slice at Outlet
Figure 6.54: Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (2 levels) Based on $\Delta P_0$

Figure 6.55: Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (2 levels) Based on $\Delta P_0$
Figure 6.56: Unequal Vortices - Adaptive Grid (1 level) based on $H_n, y - z$ Slice at Inlet

Figure 6.57: Unequal Vortices - Adaptive Grid (1 level) based on $H_n, y - z$ Slice at Outlet
Figure 6.58: Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (1 level) Based on $H_n$

Figure 6.59: Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (1 level) Based on $H_n$
Figure 6.60: Unequal Vortices - $H_n$ Contours at Inlet, Adaptive Grid (1 level) Based on $H_n$

Figure 6.61: Unequal Vortices - $H_n$ Contours at Outlet, Adaptive Grid (1 level) Based on $H_n$
Figure 6.62: Unequal Vortices - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at Inlet

Figure 6.63: Unequal Vortices - Adaptive Grid (2 levels) based on $H_n$, $y - z$ Slice at Outlet
Figure 6.64: Unequal Vortices - $\Delta P_0$ Contours at Inlet, Adaptive Grid (2 levels) Based on $H_n$

Figure 6.65: Unequal Vortices - $\Delta P_0$ Contours at Outlet, Adaptive Grid (2 levels) Based on $H_n$
Figure 6.66: Unequal Vortices - Convergence Histories
Figure 6.67: Unequal Vortices - $\Delta P_0$ along Stronger Vortex Core

Figure 6.68: Unequal Vortices - $\Delta P_0$ along Weaker Vortex Core
Chapter 7
Conclusions

Numerical diffusion has been determined as having a significant effect on computed vortex flows and can often be the limiting factor in the accuracy achieved in the algorithm. One method to reduce these errors is to use an adaptive algorithm which refines the grid in the vortical region. This thesis focused upon this method, namely adaptation for vortex flows. The term "adaptation" encompasses several issues. Several objectives relating to adaptation are addressed in this thesis. Specifically, this thesis examined two flow quantities, total pressure loss and normalized helicity, to evaluate their effectiveness as adaptation quantities. In addition to using these quantities for adaptation, the method of measuring these quantities was examined to determine if using a first or second difference of the quantity would a more effective adaptation parameter. Finally, an algorithm to automatically define the thresholding limits for refinement and unrefinement was developed. Two test cases were used to evaluate the above objectives: a single Lamb vortex and two unequal strength Rankine vortices with opposite rotation directions. The results from each of these studies are discussed below. In addition to these objectives, adaptation for vortex flows will be evaluated in comparison to other methods of reducing numerical diffusion errors for vortex flows.

First, the difference between total pressure loss and normalized helicity should be reiterated. Total pressure loss is a physical quantity measuring the entropy loss in the flow. Normalized helicity is the cosine of the angle between the vorticity vector and the velocity vector. For streamwise vortices, the normalized helicity will be in the range \( \pm 1 \). In a flowfield with several vortices of varying strengths, total pressure loss will only differentiate the vortices by the actual strengths while the normalized helicity of each vortex will be of comparable magnitude. Therefore, normalized helicity will theoretically be a better adaptation indicator. There is one drawback to using normalized helicity
and that is that it is extremely sensitive to noisy variations in the velocity field. The vorticity is computed in the calculation of normalized helicity and vorticity is the curl of the velocity; therefore, small variations in the velocity field can become large variations in the normalized helicity. This was seen with the unequal vortices case. Although not demonstrated it is believed that in a practical application involving shock waves, shear layers and multiple vortices the normalized helicity will be a better adaptation parameter than total pressure loss.

The second objective arose out of the adaptation methods used for shocks where a first difference of density or a divided second difference of density are commonly used adaptation parameters. The total pressure loss and normalized helicity distributions through the vortex core show that a first difference or second difference of these quantities are not accurate measures of the vortex. Both total pressure loss and normalized helicity are good indicators of the vortical regions.

The third objective of this thesis was to develop a consistently effective automatic thresholding algorithm. The autothresholding algorithms of Dannenhoffer and Powell were implemented and found not to be very consistent for the test cases in this thesis. However, a simple algorithm for automatic thresholding was found. The algorithm determines the average adaptation parameter, the high limit is then set to 1.2 times this average or a maximum of 25% of the cells are adapted. The low limit for unrefinement is the one half times the average. This algorithm was shown to be effective for all test cases.

The effectiveness of adaptation for vortex flows should be commented upon. This will be evaluated by the total pressure loss in the vortex core. Depending on the application, the total pressure loss in the vortex core may be a key flow parameter in the problem. This is true in helicopter vortex flows [9] where the effects of numerical diffusion have been studied since this diffusion is a limiting factor in the accuracy of the solutions. For the vortices coming off a helicopter rotor, a total pressure loss error of 1% can have significant effects on the blade-vortex interaction. However, for a delta wing the primary and secondary vortices formed over the wing maintain their strength due to
vorticity from the feeding sheet. The diffusion effects are prevalent downstream of the trailing edge where there is no physical phenomenon to continue feeding vorticity into the flow. Both test cases showed that with two levels of adaptation the accuracy of the algorithm was reached. The total pressure loss errors were ±2% through the channel. In order to achieve this accuracy, approximately 23 nodes across the core were needed. The adaptive grids were less than 10% of an equivalent globally refined grid. Thus, grid adaptation was shown to be effective for vortex flows using both total pressure loss and normalized helicity.

This thesis introduced the problem of numerical diffusion of vorticity with the NTF delta wing calculations in Section 1.1. The total pressure loss of the primary vortex at the trailing edge was approximately 60% while at the outflow boundary the total pressure loss dropped to 41.7% resulting in a 30.5% error [3]. The primary vortex was contained within 8 x 8 cells with 9 nodes across the core of the vortex. The second difference smoothing coefficient was \( \nu_2 = 0.07 \) and the fourth difference smoothing coefficient was \( \nu_4 = 0.012 \). A direct comparison to the total pressure loss errors from the single vortex case in Section 6.2.1 can not be made but they are of comparable magnitudes. The factor of improvement for the single vortex case with one level of adaptation using normalized helicity as the adaptation parameter was 5.7. If one level of adaptation were performed with the NTF delta wing calculation and approximating the factor of improvement to be 5 would result in a total pressure loss error of 6.1%. On an adaptive grid for the NTF delta wing, the primary vortex would be contained within 16 x 16 cells with 17 nodes across the vortex core. The original grid for the NTF calculation contained 276,000 cells. There were 25 cells in the streamwise direction over the wing and 15 cells in the streamwise direction in the wake. Assuming only the region containing the primary vortex is adapted (40 x 8 x 8 cells), this would result in 17,920 new cells and an adaptive grid with 293,920 cells. Clearly if the solution of the NTF delta wing case were adapted on more cells would be marked for refinement; however, the adaptive grid would be a small fraction of a globally refined grid (which would contain over 2 million cells). Thus, the advantage of grid adaptation is clearly seen.

Finally, some comments about other methods for correcting the numerical diffusion
of vorticity should be made. Three methods will be briefly discussed: perturbation methods, higher-order schemes and Eulerian/Lagrangian methods. Perturbation methods involve the use of a known exact or approximate vortex structure a priori in order to correct for the numerical diffusion of vorticity. Roberts [26] has successfully applied this methodology to the steady flow over a hovering rotor. He also applies this scheme to a single Lamb vortex propagating down a rectangular domain. Since values of the total pressure loss are not given, it is difficult to assess the accuracy achieved by the perturbation scheme for comparison. Rai [24] presents an implicit fifth-order upwind Euler and Navier-Stokes scheme specifically to reduce the effects of numerical diffusion on a convecting vortex. Rai achieved a 2.5% error with 8 grid points across the vortex core at 45 core radii downstream. Felici and Drela [11] present an Eulerian/Lagrangian method which consists of the addition of a Lagrangian particle tracking solution to the standard Eulerian solution for both compressible and incompressible flow. In one of their computed flows, a Lamb vortex convects downstream perpendicular to its axis of rotation over a distance of fifty core radii. The Lagrangian correction leads to a solution approximately 4 times more accurate than then Eulerian scheme alone. The results of this thesis show that grid adaptation compares favorably with these other methods for reducing the numerical diffusion of vorticity. The differences between these methods then lies in the complexity of the algorithms, the memory requirements and the CPU costs. In conclusion, this thesis has shown that grid adaptation is effective at reducing the numerical diffusion of vorticity and has demonstrated the effectiveness of total pressure loss and normalized helicity as adaptation parameters for vortex flows.
Bibliography


Appendix A

Solid Wall Boundary Conditions

A.1 Physical Boundary Condition

Solid wall boundary conditions require that there is no mass flux through the surface. Mathematically this means that the velocity normal to the boundary must be zero, i.e.,

\[ \mathbf{u} \cdot \mathbf{n} = 0, \quad \text{(A.1)} \]

where \( \mathbf{u} \) is the velocity vector and \( \mathbf{n} \) is an inward pointing unit vector normal to the surface.

A.2 Implementation

Before discussing the implementation of the solid boundary conditions, the node types on a solid surface need to be expanded upon. Solid surfaces can contain all three node types: solid, corner, and farfield. A corner node exists only when two solid surfaces of the *same* element are adjacent. An example of this is the elements that compose a channel with solid walls. The nodes where two walls meet are corner nodes. With this same channel, the elements along the solid walls at the inlet and outlet of the channel will have solid surfaces with farfield nodes at the inlet and outlet. Any other nodes along the solid walls are defined as solid nodes. The need for a distinction between corner nodes and solid nodes is to correctly enforce flow tangency along the solid surfaces.

The implementation of the solid boundary conditions will now be discussed. The fluxes at the solid boundaries are calculated by setting the normal convective flux terms through the solid face to zero before each iteration, and enforcing flow tangency after
the flux calculation. The equations for the fluxes then become

\[
\begin{align*}
F_w &= \begin{pmatrix}
\rho u_m \\
\rho u_m + p \\
\rho u_m v \\
\rho u_m w \\
\rho u_m h_0
\end{pmatrix}, & G_w &= \begin{pmatrix}
\rho v_m \\
\rho v_m + p \\
\rho v_m w \\
\rho v_m h_0
\end{pmatrix}, & H_w &= \begin{pmatrix}
\rho w_m \\
\rho w_m + p \\
\rho w_m w \\
\rho w_m h_0
\end{pmatrix}
\end{align*}
\tag{A.2}
\]

where \( u_m, v_m \) and \( w_m \) are the corrected velocities such that the total convective contribution normal to the wall is zero. These velocities are the \( z, y \) and \( z \) components of the tangential velocity which is given by

\[
\bar{u}_{\text{tan}} = \bar{u} - (\bar{u} \cdot \hat{n})\hat{n}.
\tag{A.3}
\]

Expanding \((\bar{u} \cdot \hat{n})\) gives the following expression

\[
\bar{u}_{\text{norm}} = (\bar{u} \cdot \hat{n}) = u \cdot n_x + v \cdot n_y + w \cdot n_z.
\tag{A.4}
\]

Thus the components of the tangential velocity are

\[
\begin{align*}
\bar{u}_m &= u - \bar{u}_{\text{norm}} n_x \\
\bar{v}_m &= v - \bar{u}_{\text{norm}} n_y \\
\bar{w}_m &= w - \bar{u}_{\text{norm}} n_z.
\end{align*}
\tag{A.5-7}
\]

The calculation of the normal vector will now be described. As mentioned before a solid face of an element can have three possible node types: solid, farfield, and corner nodes. If a farfield node exists on an element with a solid face, that node receives no contribution from the solid face to the normal vector. This is because the normal vector is computed with respect to a boundary, such as a farfield boundary or solid boundary, thus the normal vector from a farfield face is the only type of face that may contribute to a farfield node. If a solid node exists on a solid face the following procedure is used to calculate the normal vector:

1. Loop over all elements in the domain.
2. Identify any solid faces on each element.
3. Identify any solid nodes on the solid face.

4. Compute the normal vector components, i.e. $n_x$, $n_y$, $n_z$, for the face by taking the components of the cross product of the diagonals of the solid face.

5. For node $i$ sum the normal vector components of all faces that contain the solid node $i$.

6. Normalize the components by the magnitude of the normal vector to give $\vec{n}$.

If a corner node exists on a solid face the following procedure is used to calculate the tangential velocity (note: the tangent vector is not actually computed). The tangential velocity vector is computed to guarantee a unique, correct direction.

1. Loop over all elements in the domain.
2. Identify any solid faces on each element.
3. Identify any corner nodes on the solid face.
4. Compute the normal vector components, i.e. $n_x$, $n_y$, $n_z$ for the face, by taking the components of the cross product of the diagonals of the solid face.
5. Label the normal vector components with the boundary surface (global surface, not on the elemental level) from which it was calculated (for solid edges typically there are two surfaces, a case with three solid surfaces intersecting is a degenerate case which was not tested).
6. Sum the normal vector components of all faces on the same boundary surface that contain corner node $i$.
7. Normalize the components by the magnitude of the normal vector to give $\vec{n}$.

The tangential velocity is then calculated by the following

\[
\vec{v}_1 = \vec{v} - (\vec{v} \cdot \vec{n}_1)\vec{n}_1 \tag{A.8}
\]

\[
\vec{v}_2 = \vec{v}_1 - (\vec{v}_1 \cdot \vec{n}_2)\vec{n}_2 \tag{A.9}
\]
where the subscripts 1 and 2 refer to the two boundary surfaces that form the solid edge. This procedure computes the tangential velocity vector to boundary surface 1 and then dots this vector with the normal vector of boundary surface 2, thus the resulting vector, \( \vec{v}_2 \), is tangential to both boundary surfaces.