Abstract

CARPENTER, JAMES GIVENS V. Time Accurate Unstructured Grid Adaption in Two and Three Dimensions. (Under the direction of D. Scott McRae).

The adaption algorithm of Benson et al is extended to three dimensional unstructured grids, building on the previous extension to two dimensional unstructured grids. R-refinement grid adaption is performed using a center of mass equation constructed from a weight function computed from solution gradients. Solution variables are updated using a coupled approach where the flux interface for each cell face is adjusted by the local grid velocity. Modifications to the integration scheme are incorporated to account for volume changes due to grid adaption through the introduction of an unsteady residual term which is resolved using sub-iterations at each timestep. The previous structured grid definition of grid velocity is shown to be inadequate for unstructured grid motion, and a new conservation based grid velocity equation is constructed from the local face displacement, which is designed to capture the volume change and preserve geometric conservation. Time accuracy is demonstrated for two and three dimensions using a shock tube simulation.

Implementation for three dimensions is accomplished using a parallel, point implicit commercial flow solver. Incorporation of the gridspeed terms in the flux interface equations is presented along with the modifications to the implicit integration scheme required to account for the volume change as the grid is displaced. Extension to three dimensions required development of smoothing routines designed to preserve or recapture grid quality for arbitrary tetrahedral grids based on a geometric quality
definition. A wing section under a prescribed sinusoidal motion is presented as a demonstration case to show the efficacy of the method. Computational results are compared to experimental data and solutions obtained using CFL3D.
Time Accurate Unstructured Grid Adaption in Two and Three Dimensions

by

James Givens Carpenter V

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Aerospace Engineering

Raleigh, NC

2007

Approved by:

________________________  _________________________
Dr. J. R. Edwards          Dr. H. A. Hassan

________________________  _________________________
Dr. C. T. Kelley           Dr. D. S. McRae
Chair of Advisory Committee
Biography

James Givens Carpenter V was born in Charleston, South Carolina on the 24th of November, 1963. After graduating from Wren High School in 1982, James earned a Bachelor of Science in Computer Science from Clemson University in 1985. During his tenure at Clemson, he began working as a consultant to a textile mill in the Greenville, South Carolina area in the information systems department. After graduation, he assumed a lead role in the company’s information systems group for the next four years. In 1989, he moved to the Raleigh-Durham area and continued working as a consultant to several companies including IBM. In 1993 he began attending the graduate school at North Carolina State University, completing his Master of Science thesis, *Unstructured Grid Adaption Using Node Movement*, in 1997. Subsequently, James took a position in a commercial software company as a developer, eventually rising to the position of VP of Development, while working to complete the PhD program. James has been employed at Corvid Technologies, Inc., an engineering consulting and software development company, in Mooresville, North Carolina, since 2002, and will continue with Corvid Technologies after the completion of his PhD.
Acknowledgements

I would first like to thank my committee members, Dr. D. S. McRae, Dr. J. R. Edwards, Dr. H. A. Hassan, and Dr. C. T. Kelley for their considerable patience and assistance in the completion of the current research. A special thanks also to Dr. Stephen Campbell for standing in for my defense. Dr. McRae has always been accommodating of my work schedule and I cannot thank him enough for his encouragement and persistence. Development of the methods at the foundation of this research benefited greatly from his assistance. Dr. Edwards has been an invaluable resource over the course of this work, and I owe him a debt of gratitude for the many times he has managed to point me in the right direction. I could not have overcome many of the obstacles I faced in the course of this research without his assistance. Dr. Hassan has taught me more engineering fundamentals than I could ever hope to retain and has always been a source of encouragement. I would also like to pay special thanks Dr. Robert Silber, who challenged me early in my graduate career. I doubt he remembers me as a student in his classes, but his straightforward criticisms and high expectations had a significant impact on my graduate studies.

There were many reasons why I decided to branch out from computer science into engineering, and it was probably with more enthusiasm than reason that I began my graduate career. Balancing a job, family, and my graduate studies seemed overwhelming at times, and I would have surely given up long ago had it not been for the support of teachers, family, and friends. So many people along the way have helped and encouraged
me that there would not be space to list them all. I have received so much beyond an 
education out of my graduate experience through the friends I have made along the way 
that the journey alone has made the whole experience invaluable.

There are a number of individuals who deserve special recognition. First I want to 
thank Dr. Neal Frink at NASA Langley for providing me with a copy of his unstructured 
flow solver TetCC. Although I ultimately used a code associated with my job, I cannot 
thank Dr. Frink enough for his willingness to sort through the paperwork necessary to 
provide me with a copy of his code. Special thanks are due to Dr. David Robinson for 
providing substantially all the materials and resources used to complete the bulk of the 
work contained herein. Through Dr. Robinson I have gained access to the three 
dimensional flow solver RavenCFD, and access to abundant computational resources. 
None of this work would have been completed without his support, and I am indebted to 
him for his patience and support.

I want to thank Dr. Robert Nance for his support over the long course of this research, 
and his willingness to listen to my ramblings and incoherent ideas. I have benefited 
greatly from my many conversations with him discussing aspects of this research and I 
cannot thank him enough his assistance. I also want to thank Dr. Mike Neaves for his 
friendship and assistance over the course of my graduate career. Mike has been a 
consistent source of encouragement and help through my graduate studies and as I 
worked to complete my research. I would not have made it through most of the class 
work without Mike as a study partner. Thanks also to Cameron Dempster for his 
assistance with sample figures and CAD models used in this work.
I would also like to pay tribute to three people who inspired me to go back to school, but unfortunately did not live to see me complete this work. First to my grandmother, Laverne Shaw Carpenter, and my grandfather, James Givens Carpenter III, I owe an immeasurable debt of gratitude not only for encouraging my education, but all of the many life lessons they provided along the way. Both grew up during the depression and were never able to attend college. It was their greatest desire that I complete a college education, and it is as much for them as for me that I pursued it as far as I have. Last, I want to thank Cindy Erickson for her support during my graduate studies. Cindy provided me with a job while attending graduate school, and showed great patience with my studies and provided endless encouragement when the task seemed overwhelming. I could not have completed this work without her help.

I also want to thank my parents, Jim and Jane Carpenter, for their support, and my brother Bryan and sisters Angela and Amy for their support and encouragement. I know my mother wondered if I would ever complete this degree, but she never gave up hope and was always persistent that I finish.

Last, but certainly not least, I have to thank Pat Oliver. I cannot imagine what living with me must have been like over the long years of working and going to school, not to mention the insanity of leaving your job and home to move to another state to start graduate school. I appreciate your patience through all the long nights as I studied, and tolerating all of the times when I became frustrated or exhausted with trying to complete this degree and work. I cannot thank you enough for your support, encouragement and understanding.
Table of Contents

List of Figures ........................................................................................................ viii
List of Symbols ......................................................................................................... x

1 Introduction ........................................................................................................ 1

2 Adaption Algorithm ............................................................................................ 9
   2.1 Unstructured r-Refinement Adaption ............................................................ 13
   2.2 Weight Function ........................................................................................... 15
   2.3 Node Movement and Restriction ................................................................. 22

3 Unstructured r-Refinement in Two Dimensions ............................................. 24
   3.1 Modified Integration Scheme ....................................................................... 26
   3.2 Flux Corrections ............................................................................................ 27
   3.3 Deficiencies in Gridspeed Definition .......................................................... 30
   3.4 New Gridspeed Definition ........................................................................... 35
   3.5 Movement Restriction .................................................................................. 43
   3.6 Time Accurate Validation ............................................................................ 45

4 Unstructured r-Refinement in Three Dimensions ......................................... 50
   4.1 Numerical Approach .................................................................................... 51
   4.2 Modified Integration Scheme ....................................................................... 52
   4.3 Flux Scheme Modifications .......................................................................... 57
   4.4 Implicit Jacobian Modifications ................................................................... 63
   4.5 Time Accurate Validation ............................................................................ 65

5 Plunging Airfoil ................................................................................................. 78
   5.1 Grid Resolution and Motion ......................................................................... 79
   5.2 Results Comparison ..................................................................................... 84
   5.3 Three Dimensional and Modeling Effects .................................................... 93

6 Conclusions ........................................................................................................ 99

7 References .......................................................................................................... 103

A Three Dimensional Grid Smoothing ............................................................... 109
   A.1 Poor Tetrahedral Quality and the Quality Function ................................... 109
   A.2 Laplacian Based Smoothing ....................................................................... 113
   A.3 Optimization Based Smoothers ................................................................. 114
A.4  Face and Edge Swap ................................................................. 117
A.5  Zonal Boundary Treatment .................................................... 120
A.6  Best Practices ........................................................................ 123
List of Figures

Figure 1 - Definition of the Dual Grid. ................................................................. 25
Figure 2 - 2D Unstructured Grid Before Adaption. .............................................. 31
Figure 3 - Adapted 2D Unstructured Grid without Weight Function ............... 31
Figure 4 - Density Contours of Perturbations After Adaption....................... 32
Figure 5 - Adapted Grid with Artificial Sink. ..................................................... 33
Figure 6 - Density Contours from Adapted Sink Grid. ...................................... 33
Figure 7 - Definition of Medial Dual Grid. .......................................................... 36
Figure 8 - Volume Displacement from Adaption. .............................................. 37
Figure 9 - Motion Vector to Define Face Displacement. .................................... 41
Figure 10 - Geometry of 2D Dual Grid with Motion Vector. ............................ 44
Figure 11 - Artificial Sink using New Gridspeed Definition. ............................ 46
Figure 12 - Adapted Shock Tube Grid. ............................................................... 46
Figure 13 - Adapted Solution Compared to Theory. .......................................... 48
Figure 14 - Unadapted Grid Compared to Theory. ........................................... 49
Figure 15 - Adapted 3D Sub-Volume. ................................................................. 59
Figure 16 - Volume change from 3D face displacement. ................................... 60
Figure 17 - Sub-Volume Calculation. ................................................................. 61
Figure 18 - Transient Grid Wave Moving in Positive X. .................................... 67
Figure 19 - Transient Grid Wave Moving in Negative X. ................................. 68
Figure 20 - Shock Tube Results for Starting, Middle, and Intermediate Solution... 73
Figure 21 - Unadapted Shock Tube Results. ..................................................... 74
Figure 22 - Adapted Shock Tube Results. ......................................................... 75
Figure 23 - Airfoil Surface and Volume Resolution. .......................................... 80
Figure 24 - Near Body Resolution. ................................................................. 80
Figure 25 - Trailing Edge Resolution. .............................................................. 81
Figure 26 - Grid Motion Region. ................................................................. 82
Figure 27 - Grid Displacement for Top and Bottom of Cycle. ......................... 83
Figure 28 - Vertical and Horizontal Plot of Pressure Across Motion Region. .... 86
Figure 29 - PIV Velocity Data (Left) Compared to Laminar Results (Right) ....... 87
Figure 30 - CFL3D Laminar Velocity (Left) Compared to Laminar Results (Right)... 89
Figure 31 - PIV Velocity Data (Left) Compared to Spalart-Allmaras Results (Right)... 91
Figure 32 - PIV Vorticity Data (Left) Compared to Laminar Vorticity (Right) .... 92
Figure 34 - Comparison of 2D (Left) and 3D (Right) Vorticity Results ......... 94
Figure 33 – Iso-Surface of Vorticity Magnitude. ............................................. 96
Figure 35 - Comparison of 3D (Left) and Delta (3D-2D Right) Vorticity Results .... 97
Figure 36 - Leading Edge Grid and Spanwise Facetization............................. 98
Figure 37 - Linear Forms of Poor Quality Tetrahedra. ..................................... 110
Figure 38 - Planar Forms of Poor Quality Tetrahedra. ..................................... 111
Figure 39 - Examples of Circumcircle of a 2D Triangle. ............................... 112
Figure 40 - Face Swap Example. ................................................................. 118
Figure 41 - Canonical Form for 4 and 5 Cell Configurations. ......................... 119
Figure 42 - Canonical Form for 6 Cell Configurations .................................... 119
Figure 43 - Canonical Form for 7 Cell Configurations ................................. 120
List of Symbols

\( a \)  \hspace{1em} \text{Speed of Sound}
\( A_f \)  \hspace{1em} \text{Face Area}
\( A_i^+, A_m^+ \)  \hspace{1em} \text{Jacobean Matrices}
\( c \)  \hspace{1em} \text{Chord Length}
\( e \)  \hspace{1em} \text{Internal Energy}
\( E \)  \hspace{1em} \text{Split Flux}
\( E_T \)  \hspace{1em} \text{Total Energy}
\( f, g, h \)  \hspace{1em} \text{Inviscid Flux Vectors}
\( f_p \)  \hspace{1em} \text{Frequency for Plunging Airfoil}
\( F \)  \hspace{1em} \text{Total Flux}
\( H \)  \hspace{1em} \text{Total Enthalpy}
\( h \)  \hspace{1em} \text{Reduced Plunge Amplitude } l/c
\( i, j, k \)  \hspace{1em} \text{Direction Unit Normal Vectors}
\( I \)  \hspace{1em} \text{Identity Matrix}
\( k \)  \hspace{1em} \text{Coefficient of Thermal Conductivity}
\( k_f \)  \hspace{1em} \text{Reduced Frequency}
\( l \)  \hspace{1em} \text{Physical Plunge Amplitude}
\( M \)  \hspace{1em} \text{Mach Number}
\( \hat{n} \)  \hspace{1em} \text{Cell Face Unit Normal}
\( N_n, N_i \)  \hspace{1em} \text{Number of Neighbors of a Node and Cell}
\( P \)  \hspace{1em} \text{Point or Node Location or Pressure}
\( Q \)  \hspace{1em} \text{Solution Vector of Dependent Variables}
\( r, s, t \)  \hspace{1em} \text{Viscous Flux Vectors}
\( \hat{r}, \hat{p} \)  \hspace{1em} \text{Direction Vectors for Movement or Interpolation}
\( S \)  \hspace{1em} \text{Cell Face Area}
\( t \)  \hspace{1em} \text{Time}
\( T \)  \hspace{1em} \text{Temperature}
\( u, v, w \)  \hspace{1em} \text{Velocity Components}
\( U_n \)  \hspace{1em} \text{Velocity Normal to a Face}
\( V \)  \hspace{1em} \text{Volume}
\( \hat{V} \)  \hspace{1em} \text{Velocity}
\( \hat{X}_f \)  \hspace{1em} \text{Cell Face Velocity}
\( \hat{z} \) Cell Face Motion Normal Vector
\( \alpha \) Integration Scheme Coefficient
\( \Delta \) Difference Operator
\( \varepsilon \) Small Number on the Order of Machine Zero
\( \gamma \) Ratio of Specific Heats
\( \eta_1, \eta_2 \) First and Second Order Coefficients
\( \phi \) Solution Difference Vector
\( \rho \) Density
\( \sigma \) Weight Function Biasing Coefficient
\( \theta \) Implicit Operator
\( \tau_{xx}, \tau_{xy}, \tau_{yy} \) Shear Stress Components
\( \omega \) Weight Function

**Subscripts**

- \( f \) Face
- \( k \) Weight Function Variable, Runge-Kutta Weight
- \( n \) Node
- \( m \) Neighbor Node or Cell
- \( n \) Normal to a Cell Face
- \( 0 \) Previous Time Level
- \( i \) Cell
- \( r \) Ratio
- \( s \) Sub-iteration
- \( t \) Time Derivative
- \( x, y, z \) Coordinate Directions
- \( \min, \max \) Minimum or Maximum Value
- \( \text{cm} \) Center of Mass

**Superscripts**

- \( 0 \) Previous Time Level
- \( C \) Convective Flux Components
- \( k \) Newton Sub-iteration Level
- \( n \) Time Level
- \( P \) Pressure Flux Components
- \( +, -, \pm, \mp \) Upwind Direction for Split Fluxes
1 Introduction

Advances in unstructured flow solvers in recent years have enabled engineers to obtain solutions on increasingly complex geometries, thereby moving unstructured methods to the forefront of applied computational engineering. Advantages of unstructured methods lie in the ability to generate grids of reasonable quality around complex shapes, owing in large part to the lack of implicit ordering and connectivity. Implicit in this advantage, however, is the need to explicitly define grid connectivity, thus increasing the overhead of unstructured methods. Control of element count is therefore desirable to reduce the resource requirements associated with the method for a given solution, without sacrificing resolution and accuracy. Resolution can be addressed to a limited degree in the grid generation process by pre-clustering elements in regions of known interest, such as a body wake. In cases where a priori knowledge of the flowfield is unavailable, adaption techniques can be used to resolve areas of interest dynamically as the solution evolves. Adaption techniques can be grouped generally into two types: enrichment and movement. Enrichment techniques add elements where resolution is required and potentially remove elements where reduced resolution would not adversely affect the solution. Aside from the complexity of implementing an enrichment scheme in a general unstructured solver, enrichment carries the extra burden of increasing the computational overhead as cells are added. In the case of transient flows, the increase in cell count can be large if not coupled with a robust coarsening scheme. Movement methods, by contrast, generate no additional overhead to the solution in terms of
increased element count, and add only a modest overhead in computing the grid movement. Although each method has advantages and disadvantages, movement methods are generally easier to implement and result in less overhead.

One current shortcoming of both methods, however, is the inability to provide accurate solutions to transient problems without introducing perturbations into the solution as the grid is adapted. Development of a conservative, time accurate adaption method for unstructured grids would expand the scope of solutions obtainable for transient flows.

In a recent survey of applied CFD methods, Rumsey and Ying\textsuperscript{1} concluded that numerical error was often the primary cause of inaccurate lift predictions for high lift configurations, frequently exceeding modeling error. Venditti and Darmofal\textsuperscript{2} in their work on adaption methods noted that the dominant reason for inaccurate predictions of computational results has been identified as grid resolution. It is agreed generally that sufficient grid resolution is necessary for the consistent and accurate prediction of flows and forces on aerodynamic bodies. Unfortunately, grid resolution as viewed from the grid generation process is analyst dependent, or at least dependent on the principles involved in the grid generation process. At best it requires \textit{a priori} information about the flowfield to develop a grid of sufficient resolution. Resolution thus becomes more of a subjective and less objective process by which the flowfield is resolved. Adaption has been developed as a way to mitigate problems associated with resolution of the initial grid by adding resolution where and, in the case of transient problems, when needed.
Unstructured adaption algorithms are often favored over structured grid approaches in much of the current research because structured grid approaches cannot support the dual needs of increased geometric complexity and localized resolution without upsetting the natural ordering of the grid. Any fully anisotropic adaption scheme for a structured grid would require explicit ordering, and result in a method comparable in complexity and overhead to unstructured methods. Unstructured methods, by definition, maintain data structures to explicitly define grid connectivity which results in substantial overhead but provides a method to generate a grid for most geometries of interest. These methods also provide a framework whereby selected areas of the flow can be resolved independently of the local topology to increase resolution and accuracy through the use of adaption.

The three methods available to adapt unstructured grids are r-refinement (movement), h-refinement (enrichment), and p-refinement (reconnection). Most of the research in unstructured grid adaption has focused on enrichment schemes represented primarily by h-refinement approaches. H-refinement enrichment schemes use an error estimator to determine regions of the grid where the solution is under resolved, and add elements locally to improve the resolution and (hopefully) the accuracy of the solution. While enrichment schemes have demonstrated the ability to improve solution accuracy through local refinement, there is added overhead in the form of increased cell count. For steady state solutions, the increased overhead is incidental assuming the resolution is needed to provide the spatial accuracy required for convergence. For transient flows, enrichment can increase the overhead unnecessarily unless coupled with a robust coarsening algorithm.
Besides the obvious issues of increased overhead, there are more subtle issues of grid quality that can arise from enrichment. Speares and Berzins\(^{13}\) have developed a time dependent 3D unstructured adaption algorithm built strictly upon h-refinement. In their approach, they subdivide selected tetrahedral elements by either edge bisection or introduction of a new node at the center of the selected tetrahedral. A drawback of this approach is that the grid quality, where quality in this case is defined only in terms of cell geometry, will decay over time as a given element is repeatedly subdivided. Without a smoothing algorithm to recover the grid quality, the number refinement iterations are artificially restricted, limiting the amount of resolution which can be added to the grid. Cavallo and Baker\(^{4}\) in the description of their edge splitting technique noted the need to implement a p-refinement algorithm for edge and face swapping to remove elements contributing to poor quality. In their approach, a locally constrained Delaunay approach is implemented which effectively results in local mesh reconstruction to increase resolution. Local reconstruction usually improves grid quality, but does not prohibit poor quality.

Grid quality, both for grid generation and adaption algorithms, has proven to be a difficult subject and one that has been the focus of much research. Freitag and Knupp\(^{5}\) outline nine independent tetrahedral cell configurations which can result in poor quality cells and affect accuracy. Each of these cell types is a possible result of successive iterations of enrichment algorithms if applied without mesh smoothing to improve quality. Structured and 2D unstructured methods have typically used Laplacian smoothers to recover element quality with success, but for unstructured 3D grids
Laplacian smoothers may not only fail to improve poor quality cells, but can degrade quality, especially in regions of varying connectivity. To improve cell quality, the use of an optimized mesh smoother based on a quality function is necessary. Frietag and Ollivier-Gooch\textsuperscript{6} have developed a best practices approach which incorporates p-refinement in the form of edge and face swap, with r-refinement performed by a Laplacian and an optimization based smoother to improve grid quality. Application of these two methods in combination has demonstrated improved mesh quality on grids where Laplacian based smoothers fail. In this context, however, it should be noted that quality is measured in purely geometric terms, which may not describe the best structure to resolve flow gradients, as shown by Laflin\textsuperscript{14}.

Measures of grid quality are typically restricted to a geometric description of quality based on quantities such as minimum angle or aspect ratio, for example. Viewed in a broader context, quality is best described in terms of errors produced from the discretization relative to the underlying solution. McRae and Bond\textsuperscript{15} have proposed an alternate view of quality which includes elements of the underlying solution in determining grid quality. The result is a universal grid quality metric based on an equidistribution principle that defines quality in terms of cell face alignment with the local gradient. By aligning the cell face with the local gradient, errors in the flux calculation are reduced resulting in improved accuracy. In terms of this metric, the typical aesthetic view of quality, where skewness and stretching are considered to result in poor grid quality is misleading when taken out of context with the solution. Regions of
high gradient, such as shock waves, actually benefit from skewness if face alignment with the local Riemann problem is improved.

In order to maintain quality, research has shown that enrichment methods require either a limited form of adaption or some form of smoothing algorithm to improve grid quality after enrichment. Restricting adaption presents obvious issues of restricted resolution and limited improvement in accuracy. Addition of a smoothing algorithm incorporates r-refinement, and reintroduces the need to properly account for grid movement in the flux terms to prevent perturbations to the flow\textsuperscript{16}. Therefore, even when using enrichment to perform adaption there is a need to develop a time accurate adaption algorithm to solve transient flows accurately.

Adaption algorithms using r-refinement have received little research focus as noted by Baker\textsuperscript{3}. Zegeling\textsuperscript{17} presents development of a r-refinement algorithm for finite differences, but the application was limited to 1D and 2D solutions. Baines et al\textsuperscript{18,19} demonstrated r-refinement methods for partial differential equations in 2D, and outlined the steps necessary to implement a time accurate movement scheme through the development of gridspeed terms and corrected velocities as applied to elliptic problems. Movement algorithms generally operate through the equidistribution of a weight function. The weight function, much the same as the error function, is designed to identify areas of the grid where more resolution is required. Once the weight function is defined, the weights are distributed throughout the grid resulting in higher resolution in areas identified by the weight function. Areas where the weight function is small and evenly distributed are smoothed for structured and 2D unstructured grids, while nodes are
clustered to increase resolution in areas where the weight function is large. Determining an appropriate weight function distribution can be more problematic for 3D unstructured grids.

Tezduyar\textsuperscript{20,21} has performed considerable research in the area of moving bodies, such as a rotating propeller blade, where body surfaces are translated in the grid to simulate movement. The focus of the approach is to simulate movement rather than to improve accuracy, but the principles involved are the same in as much as the grid is deformed via node movement as the body surface is moved, and a method to recover the solution after movement is required. In this case, interpolation to the new grid was used to transform the solution from the old grid to the new grid.

Research on methods to maintain accuracy and conservation of the solution during node movement has primarily centered on structured grids, using either a coupled or decoupled approach to advance the solution. Decoupled approaches require an interpolation scheme to update the grid after adaption, and enjoy the advantage of being independent of the integration scheme and the flow solver in general. A decoupled approach was developed by Benson and McRae\textsuperscript{22,23} and expanded by Laflin\textsuperscript{14} where the computational domain was utilized to perform movement, and the solution was then interpolated to the new grid level. The algorithm proved to be very efficient, but use of the computational domain precluded its use for any unstructured grid approach. Moreover, the use of interpolation on unstructured grids is more problematic, and involves the use of computationally expensive algorithms to determine the interpolation stencil.
A coupled approach was previously investigated by Klopfer and McRae\textsuperscript{24} for explicit solvers and by Orkwis and McRae\textsuperscript{25} for implicit solvers. Neaves\textsuperscript{26} successfully implemented the coupled approach in a 3D implicit structured code to analyze inlet unstarts. In the coupled approach, the grid is moved using the standard node movement algorithm, then the flux terms are adjusted to account for the node movement in the interface velocity. The integration algorithm is modified to account for the resultant volume change, and, in this case, the time accurate sub-iterations of Rai\textsuperscript{27} were used to recover time accuracy at each timestep.

Despite the lack of portability, a coupled approach presents several advantages over the decoupled approach for unstructured grids. Issues with defining a separate interpolation method are avoided using the coupled approach, and the robustness of the integration scheme is utilized to update the solution after each adaption step. Development of appropriate gridspeed terms can ensure conservation and thereby prevent additional perturbations to the flow. Once gridspeed terms have been defined, incorporation of the terms into the flux scheme is straightforward. Modification of the integration scheme to define an unsteady residual and sub-iterations to recover time accuracy complete the algorithm.
2 Adaption Algorithm

The goal of adaptive grid algorithms is to improve the accuracy of the solution and reduce analyst dependence through the appropriate resolution of salient features in the flowfield. Most adaption methods incorporate an error estimator to determine areas of the solution where there are large values of the error function as defined by the integration method. Determination of solution error is an ongoing field of research, but a common approach is to approximate the error through the use of solution gradients. All numerical approximations to governing equations result in some form of truncation error consisting of solution derivatives multiplied by powers of the local grid spacing. Truncation error can be used to estimate errors in the solution and identify regions of the flowfield that require additional resolution. The goal of additional resolution by any means is to reduce the local spacing and thereby reduce the local truncation error.

Several methods are available to estimate the location truncation error. A precise approach would be to determine the exact form of the truncation error terms based on the numerical method used in the flow solver. Although the first truncation error term is generally sufficient to approximate the error, a more precise determination of the error term would require determination of many truncation error terms, which must then be approximated and evaluated locally to determine the local error. A drawback of this approach is that it cannot be generalized as the truncation error is unique to each numerical method and is thus restricted to the flow solver for which the error terms are developed. A more generalized approach is to use an error estimator consisting of
solution gradients formed from first order gradients in the flowfield. First order gradients can be used to approximate higher order solution derivatives by taking the differences of the first order terms, which vary at a rate proportional to the higher order terms\textsuperscript{28}. Increased computational efficiency and reduced round-off can be accomplished through the use of undivided differences in the error estimator, and normalized by the local values. Normalized undivided solution differences form the basis of the error estimator used in the current work.

Once an error estimator is defined and implemented, a method to reduce the local spacing based on the local error function is needed. Two general approaches to this problem are enrichment and movement. As noted earlier, the typical approach is to use enrichment to locally refine the grid where the solution error is large, however there are many complications associated with enrichment algorithms that make their implementation difficult for unstructured grids. One of the biggest obstacles is the local data structure of the flow solver. Unlike structured grid methods, which generally employ an explicit i,j,k ordering scheme, unstructured solvers must explicitly define connectivity between each node and each cell in the grid. Unstructured grid generators and flow solvers employ a number of different techniques to construct the grid and compute the solution. For example, grid generators can use a hyperbolic advancing front technique, or a top down octree type approach. Flow solvers can be cell centered, or vertex based with a dual grid that can be defined a number of different ways, such as a median dual. Differences in the grid generation and flow solver approaches results in a number of different data structures used to implement each method, which are typically tailored to
each approach. An enrichment scheme can be developed for each of these methods, but generalization is difficult and will require modification of the local data structure at some level.

In addition to issues with the local data structure, parallelization can pose a number of problems in the context of enrichment schemes. Implementation issues arise at inter-zonal processor boundaries that are dependent on the enrichment scheme being employed. Face splitting schemes can be the easiest to implement since the face split is independent of information on each processor. Edge splitting is difficult because the order of operations determines the resulting face geometry if the edges of a face are split more than once during a cycle. Care must be taken to ensure the order of operation at the inter-zonal boundary is consistent for each processor so that the same adapted grid structure is produced on each processor. In addition to cell splitting issues, smoothing algorithms can become complicated at the inter-zonal boundary. Laplacian and optimized smoothing algorithms are node based methods which can be implemented with an acceptable amount of overhead at an inter-zonal boundary. Edge swap techniques, however, become increasingly complex at an inter-zonal boundary since they necessitate a new local definition of the inter-zonal connections between processors, due to changes in the local grid structured as a result of the swap.

An important limitation of enrichment schemes is the added computational overhead incurred, especially for parallel implementations. Enrichment schemes by nature add cells to the grid structure to provide increased resolution of the flowfield. In a parallel environment, no general way exists to predict which regions of the flowfield will be
resolved as a result of the adaption process, leading to an increased load on processors where there are concentrations of enrichment, and leaving other processors idle while the adaption cycle completes. If left unchecked, this can lead to a severe load imbalance and requires rebalancing of the parallel problem. Some parallel distribution codes such as ParMETIS\textsuperscript{29} contain routines designed to provide a new distribution to rebalance the problem after enrichment, but rebalancing can require a substantial cost in communications overhead, and thus increased solution time. For flows with transient features, the need to adapt frequently can make enrichment schemes prohibitively expensive and can require a robust coarsening scheme to prevent the cell count from becoming unmanageable.

Node movement schemes, by contrast, can be generalized to accommodate multiple flow solver variants and formed independent of the grid distribution across multiple processors. Since no cells are added or removed from the domain, the parallel problem never becomes unbalanced and there is no need to redistribute the grid as the adapted solution evolves. Construction of a local mesh, including a node and its surrounding cells, allows generalization of the node movement algorithm independent of the underlying flow solver. The result of the node movement scheme is a new node location based on the movement algorithm, which is updated independent of the flow solver and does not require alteration of the underlying data structures. Some modifications are most likely required to produce a local grid structure for a node, but these modifications are in addition to the data structure native to the flow solver and do not require alteration of that data structure. Use of the local mesh also generalizes the approach since node
movement is dependent only on the local data structure of the movement algorithm, which is true for inter-zonal nodes as well as nodes that lie within the grid domain of each processor. The ability to define a generalized approach with low computational overhead make movement schemes an attractive alternative to enrichment schemes for adaption based algorithms. Therefore the current research is focused on an unstructured node movement scheme which is both robust and preserves time accuracy.

2.1 Unstructured r-Refinement Adaption

Development of the current unstructured adaption algorithm is based on the structured grid adaptive algorithm originally developed by Benson and McRae\textsuperscript{22,23,30,31}. Extension of the Benson and McRae method to unstructured grids required modification of the original method for use in a generalized unstructured approach\textsuperscript{33}. Specifically, the original method applied grid motion to the computational domain, which was then mapped onto the physical domain. Solution variables from one grid level to the next were interpolated based on the previous grid and solution applied to the new grid after adaption. Since no computational space exists for an unstructured grid, the method is modified to perform movement in physical space. In place of an interpolation scheme, grid motion was coupled to the flow solver via gridspeed terms developed from the grid motion. Coupling the grid motion to the flow solver allowed for development of an adaption scheme which was both conservative and maintained the accuracy of the original flow solver\textsuperscript{26}. 

Most concepts of the original adaption algorithm, however, are maintained in the current approach. As with the original method, node movement is based on a center of mass calculation using a weight function developed from solution differences. Weight function construction is based on solution differences by means of user supplied biasing coefficient that select flow variables of interest to form the solution weights. Solution differences are combined to form the weight function used to perform the center of mass calculation, where the weight function is analogous to the monitor surfaces described by Eiseman\textsuperscript{34}. Changes are required to the original weight function and center of mass definitions to account for the non uniform connectivity associated with unstructured grids. Clipping operations described with the original method\textsuperscript{33} are maintained to provide an even distribution of resolution to strong and weak features in the flowfield. A weight function spreading operation is performed to determine how far a point is moved in each adaption cycle. The result is a generalized movement algorithm which can be applied to unstructured grids in two and three dimensions.

Application to three dimensional grids revealed limitations of the original 2D method which must be addressed. The Laplacian operator used to perform the center of mass calculations acted as a natural smoother for structured and two dimensional unstructured grids. When applied to three dimensional grids, however, the Laplacian operator is unable to detect many of the grid quality deficiencies, such as spikes, slivers, and spires, associated with tetrahedral meshes and is therefore unable to provide adequate smoothing in these regions and in many instances will invert cells if care is not taken. Motion limiting is sufficient to prevent inversion in many cases, but poor quality tetrahedral cells
such as spikes and slivers are difficult to detect with an elliptical smoother, and there is as yet no common approach which will identify and properly handle all of the geometrically poor cell types\textsuperscript{6}. Extension to three dimensions therefore necessitated the addition of an optimization based smoother that can address all types of grid deficiencies in the tetrahedral mesh, providing improved cell quality without inverting the grid. Details of the smoother are included in Appendix A.

The following sections describe details of the new adaptive grid algorithm with the modifications necessary for application to unstructured grids. Details of the weight function are developed first, followed by development of the center of mass routine. Node movement restriction routines are described in later sections pertaining to implementation in two and three dimensions, since different techniques were required for each application.

### 2.2 Weight Function

Formation of the weight function is based on solution differences which are used to approximate solution error for each cell. In this instance, differences are the absolute undivided differences of the flow quantities stored at each cell. Two dimensional solutions for the current work are based on a cell vertex description of the flowfield, meaning that the flow quantities are stored at the node locations. Difference equations are formed by differences between a node and all the associated neighbor nodes based on the local grid connectivity, multiplied by the unit direction vector connecting the nodes to
break the weight into component directions. The resulting difference equation is expressed as

$$\phi_{k,n,(x,y)} = \frac{\sum_{m=1}^{N_n} |Q_{k,n} - Q_{k,m}| \hat{r}_{(x,y)}}{N_n |Q_{k,n}| + \varepsilon}$$

(1)

where $$\hat{r}_{(x,y)}$$ represents a unit direction vector connecting the nodes, $$Q$$ represents the flow quantity of interest for the difference operator, $$N_n$$ is the number of neighbors associated with each node, and $$\varepsilon$$ is a small quantity to prevent divide by zero. The local quantity of interest is included in the denominator to prevent overweighting of flow quantities in regions where very large flow gradients are present. Inclusion of the neighbor node count in equation 1 is necessary to ensure that the weight function is not biased by the local grid connectivity. Unlike structured grids, unstructured grids do not contain uniform connectivity. Therefore the summation of difference terms must be normalized to remove the influence of changes in grid topology that are not related to actual solution differences. Division by the count of neighbor nodes will thereby reduce biasing due to grid topology. Currently, undivided differences of density, pressure, Mach number, and velocity components are used to form the weight function.

For three dimensional calculations, a different flow solver was used that required storage of the solution variables at the cell centers instead of the cell vertices. In the three dimensional case, solution differences are based on a cell and all of the neighbor cells as
defined by the local cell face connectivity, giving a new form of the difference equation as

$$
\phi_{k,i,(x,y,z)} = \frac{\sum_{m=1}^{N_i} |Q_{k,i} - Q_{k,m}| \hat{r}_{(x,y,z)}}{N_i (|Q_{k,i}| + \varepsilon)}
$$

where $\hat{r}_{(x,y)}$ now represents a unit direction vector connecting two cells through a common face, and $N_i$ is the number of cell neighbors associated with cell. Using cell differences implies that solution variations are measured only through connecting cell faces, and not through connecting nodes, potentially reducing the topology used for constructing the difference equation. Use of node based differences, however, would require averaging cell data to each of the nodes associated with a cell, and would contain no more information than is contained in the cell centered definition of the grid. Basing differences on the cell center data is more consistent with the definition of flow quantities in the grid, and how the governing equations are solved for a given topology.

After the solution differences are formed, the initial raw weight function is formed from a linear combination of the difference equations for each coordinate direction, representing a maximal collapse in dimensionality\textsuperscript{36}. Weights for each solution difference equation are included through a user specified biasing coefficient. Biasing coefficients select which flow quantities are used to define the unified weight function, where the relative weight of each coefficient determines how strongly each flow variable influences the total weight. Users can tailor the weight function through biasing the
coefficients to resolve features of interest. The raw form of the weight function for two and three dimensional flows is thus defined as

\[
\omega_{(x,y,z)} = \sum_{k=1}^{Nv} \sigma_{k,(x,y,z)} \phi_{k,(x,y,z)}
\]  

(3)

where the \( z \) component is zero for two dimensional solutions. \( \sigma_k \) represents the user defined biasing coefficient used to select solution differences of interest.

The raw formulation of the weight function does not generally provide balanced resolution of all flow features in a given solution. Features such as shock waves have very strong gradients, while features such as expansion waves have smaller cell to cell gradients. Gradients in the vicinity of shock waves vary according to the local shock strength, which can vary greatly throughout the domain\(^{33}\). Variations in solution gradients require that a means to balance the weight function is needed if all features of the flow solution are to be resolved evenly. Although this approach does not minimize the error function defined from the solution gradients, it will tend to increase the accuracy of all selected features of the flow solution, weak and strong, not just the strongest features. Implicit in this assumption is the fact that the truncation error is not the only determining factor in solution accuracy.

A clipping operation is performed on the raw weight function to enforce a balance between strong and weak features in the flow. The purpose of the clipping operation is to reduce the relative weight of stronger features in the flowfield, while increasing the relative weight of weaker features. Two user specified input parameters are defined to specify minimum and maximum ranges for the weight function. Since the range of
weight function values can vary from one solution to another, the maximum and minimum values are specified as percentages of selected global measures of the current solution. Percentages are applied to the global range of the current weight function, with any values above the maximum percentage or below the minimum percentage eliminated. Clipping the weight function can also eliminate some of the numerical noise associated with the solution, and thereby focus the adaption on significant solution differences. Care must be taken in selection of the percentages to not over damp the raw weight function. Specifying a maximum that is too low will tend to eliminate adaption to any strong flow features, while a high minimum value will treat weak features the same as regions with a zero gradient. The equation for the clipping function is given as

$$\omega_{(x,y,z)} = \min(\max(\omega_{(x,y,z)}, \omega_{\min(x,y,z)}), \omega_{\max(x,y,z)})$$

where the min and max values are defined from the user defined percentages applied to the global weight function.

The resulting weight function contains discontinuous first derivatives throughout the flowfield as a result of the clipping operation. These discontinuities are eliminated by applying a Laplacian smoothing operator to the weight function. Smoothing also blends the weight function so the grid motion is not localized to a feature, such as a shock wave, resulting in a discontinuous change in the volume ratio between cells. Smoothing leads to increased resolution of the feature over a wider region of the flowfield and provides better blending to the unresolved portions of the flowfield. To ensure resolution of the original weight function is not lost, the smoothed weight function is combined with the
original values in the method defined by Laflin\textsuperscript{14}, resulting in the following form of the smoothing operator in two dimensions

\[
\omega_{n,(x,y)} = \frac{2\omega_{n,(x,y)} + \sum_{m=1}^{N_{n}} \omega_{m,(x,y)}}{2 + N_{n}}
\]  

(5)

with the three dimensional form given by

\[
\omega_{i,(x,y,z)} = \frac{2\omega_{i,(x,y,z)} + \sum_{m=1}^{N_{i}} \omega_{m,(x,y,z)}}{2 + N_{i}}
\]  

(6)

where summation occurs between a node and all neighbor nodes for two dimensional solutions, and between a cell and all neighbor cells for three dimensional solutions.

After formation, the individual solution weights are clipped for balanced resolution, smoothed and then combined in a linear fashion to form a unified weight function. This scalar function is applied equally to each coordinate direction. While it is possible to generate a weight function for each coordinate direction for a structured grid approach, which can then be applied sequentially to promoted alignment of the structured grid with strong features during the adaption process, the lack of preferred direction in an unstructured grid makes this process inappropriate. Local grid adaption for unstructured grids depends on the local grid connectivity, so no \textit{a priori} prediction of a preferred direction for the grid adaption can be made.

The last operation applied to the solution weights is a rescaling function. Grid motion for the current algorithm is determined by a center of mass calculation described in a subsequent section. The basic function of the center of mass calculation is to relocate the
central node based on the local weight function values. Overweighting one region of the
grid forces the node closer to the higher weighted regions of the grid while moving away
from regions with lower weights. Implicit in this calculation is that the displacement
distance experienced by each node will be a function of the weight distribution. Regions
with larger local variation of the weight function will experience more relative motion
compared to regions where the weights are nearly uniform. Rescaling the weight function
can take advantage of the center of mass calculation by changing the distribution of the
weight function to control how far a cell is displaced during each adaption cycle.
Ramping of the scale can add further robustness to the calculations by allowing the user
to control the amount of motion performed by the adapter as the solution develops. High
initial perturbations at solution startup can be damped by scaling down the weight
distribution initially, and ramping up to an increased distribution as the solution evolves.
The rescaling operation is defined as

\[
\omega = \frac{(\omega_{(x,y,z)} - \omega_{\min,(x,y,z)})}{(\omega_{\max,(x,y,z)} - \omega_{\min,(x,y,z)})} (\omega_{\max,(x,y,z)} - 1) + 1 \tag{7}
\]

where here the min and max values represent the range of weight function values defined
by the rescaling operation, and the new range is defined as from 1 to the maximum
current value specified by the user. Equation 7 represents the final form of the weight
function used in the node movement algorithm described in the next section.
2.3 Node Movement and Restriction

Node movement is accomplished via a center of mass calculation analogous to the form used for structured grids. For structured grids, node movement is performed for each coordinate direction independently in computational space. Unstructured node movement uses a unified, rescaled weight function defined in the previous section in the center of mass calculation applied to physical space. New node locations are determined by the sum of the product of the weight function with the coordinate locations for each node and connecting node in two dimensions, and by each cell attached to a node in three dimensions. In each case, the sum is normalized by the total weight function value of a node and all neighbors. In two dimensions, the center of mass calculation is expressed as

\[
P_{cm_n} = \frac{\omega_n P_n + \sum_{m=1}^{N_n} \omega_m P_m}{\omega_n + \sum_{m=1}^{N_n} \omega_m}
\] (8)

and the three dimensional form is given by

\[
P_{cm_n} = \frac{\sum_{m=1}^{N_i} \omega_m P_m}{\sum_{m=1}^{N_i} \omega_m}
\] (9)

where in the two dimensional case the summation is performed over a node and all the connecting nodes, while the summation occurs over all of the cells incident on the node in the three dimensional case. In the three dimensional case, weight values are assigned to the cell centers and not to the nodes. Therefore the center of mass calculation relocates the node based on weights of the associated cells and does not include a value for the
current node. Additionally, since the weights are located at cell centers, a natural form of motion restriction is implemented since the volume defined by the center of mass routine is restricted to the region defined by the cell centers and not by the node locations. Barring large distortions in the grid, such as sharp corners, application of the center of mass calculation should not generate grid crossover. In practice, unfortunately, regions of poor grid quality produce localized geometry that is similar in structure to sharp corners or other types of problematic geometry. Issues related to poor grid quality and crossover are addressed in the section dealing with node movement restriction in three dimensions.
3 Unstructured r-Refinement in Two Dimensions

Initial development of the time accurate adaption scheme focused on two dimensional flows using an explicit two dimensional flow solver for the Euler equations developed as part of the current research. Development of the two dimensional flow solver begins with the integral form of the Euler equations given by

\[
\frac{\partial}{\partial t} \int_V Q dV + \int_S (\mathbf{f} \cdot \mathbf{n}) dS = 0
\]  

(10)

where

\[
Q = \begin{bmatrix} P \\ u \\ v \\ T \end{bmatrix}
\]

\[
f = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ u(\rho e + P) \end{bmatrix} \quad g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ v(\rho e + P) \end{bmatrix}
\]

Integration of the flow solution is accomplished via a four stage Runge-Kutta method defined as

\[
Q^{n+1,k+1} = Q^n - \frac{1}{\alpha_k} \Delta t \frac{\widetilde{F}_{net}(Q^{n+1,k})}{V}
\]  

(11)

where \( \widetilde{F}_{net}(Q^{n+1,k}) \) represents the net flux evaluated at the \( k^{th} \) stage of the Runge-Kutta step, and \( V \) represents the volume of the cell. The coefficient \( \alpha \) is defined by the Runge-Kutta method, with values corresponding to \( \alpha_k = 4, 3, 2, 1 \) for values of \( k = 1, 2, 3, 4 \). Flow
variables are stored at cell vertex locations, and an edge based implementation is used to perform flux calculations and solution integration. In the cell volume context, this implementation requires definition of a new volume cell based on the local grid construction, called the dual grid. Several options are available for the definition of the dual grid, and the current implementation uses a median dual grid where cell faces are formed from endpoints defined by the edge center between two nodes, and the cell center (area center in two dimensions) of the associated cell. The median dual results in two faces associated with each grid edge defined by the original grid connectivity, one face connecting to each cell center to the left and right of the edge as shown in Figure 1.

![Figure 1 - Definition of the Dual Grid.](image)

The solution is advanced in time to a steady state for steady state solutions, or to a specified time or iteration count for time accurate solutions. In the initial solver, grid motion was not accounted for in the solution evolution, and grid motion resulted in a perturbation to the solution. Perturbations from grid motion are iterated out over time for
steady calculations, but these perturbations prevented accurate prediction of unsteady flows. Extension to time accurate simulations required modification of cell face fluxes to account for the motion of the grid. Flux calculations at the cell interface are updated to account for the convective velocity relative to the cell face motion and the integration scheme is updated to account for the unsteady residual resulting from the volume change in each cell as the grid moves. The following sections outline the changes to the base two dimensional flow solver required to produce time accurate predictions for transient flows.

3.1 Modified Integration Scheme

Adaption of two dimensional grids results in volume changes in each cell which must be accounted for in the integration of the conservation law. The volume is usually constant for the integral of equation 10, resulting in the form shown in equation 11. When taken in the context of grid adaption the cell volume is no longer constant and must be retained as such in the time integration. The result is a non-linear term consisting of the volume multiplied by the solution vector represented as

\[
\frac{(V^{n+1}Q^{n+1} - V^nQ^n)}{\Delta t} = -\tilde{F}_{net}(Q^{n+1})
\]  

(12)

Linearization is achieved by expanding the equation at the new time level and defining an unsteady residual term which incorporates the variation of volume and the solution as the equations are evolved in time. The resulting unsteady residual is solved by incorporating sub-iterations to remove the effect of volume change as the solution evolves in time. The form of the unsteady residual is given by
\[ R_{\text{unsteady}} = \frac{V^{n+1}Q^{n+1,k} - V^n Q^{n,0}}{\Delta t} + \bar{F}_n(Q^{n,k}) \]  

(13)

which must be driven to zero to maintain time accuracy. The sub-iterations are performed using the four stage Runge-Kutta method applied to the unsteady residual. The modified Runge-Kutta is defined as

\[ Q^{n+1,k+1} = Q^n - \frac{1}{\alpha} \frac{\Delta t}{V^{n+1}} R_{\text{unsteady}} \]  

(14)

where the flux value in equation 11 is replaced by the unsteady residual defined in equation 13. A new sub-iteration time step has also been introduced. The new timestep is a pseudo timestep introduced to perform sub-iterations to advance the solution from time level n to time level n+1. The sub-iteration timestep is defined as

\[ \Delta t_i = \frac{1}{1 - \frac{1}{\Delta t} + \frac{1}{CFL_i \Delta t_i}} \]  

(15)

where CFL\_i represents the CFL value for the sub-iterations and \( \Delta t_i \) represents the pseudo timestep for the sub-iterations. Converging the sub-iterations to a specified tolerance at each time step regains the time accuracy after the grid movement.

### 3.2 Flux Corrections

The flowfields under consideration are governed by the unsteady two dimensional Euler equations. The equations are modified to include the grid movement terms resulting from the grid motion and the associated cell face and volume changes. The resulting
equations represent the governing equations for unsteady inviscid flow with grid movement.

In the context of finite volume solvers, the result of the above modification is an adjustment to the inviscid interface flux to account for the cell face movement. The modification of the cell face flux is easily made to any upwinding scheme which splits the inviscid interface flux into convective and pressure contributions, such as AUSM$^{39}$, AUSMV$^{40}$, and LDFSS$^{41}$. For example consider the flux defined by

$$E = E^c + E^p = A_f \rho V_{int} \cdot \hat{n} E^c + A_f P E^p$$

where the flux is split in convective and pressure contributions. $A_f$ represents the area of the face and $V_{int}$ is the convective velocity defined at the cell face based on the cell face outward normal $\hat{n}$. The convective and pressure terms are defined as

$$\vec{E}^c = \rho \begin{bmatrix} 1 \\ u \\ v \\ H \end{bmatrix}; \vec{E}^p = P \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The convective velocity at the cell face must be adjusted to account for the movement of the cell face after each adaption step. The first attempt to correct the cell face velocity used the method developed for structured grids. First the cell face velocity is calculated using a first order difference equation given by

$$x_t = \frac{x_{f}^{n+1} - x_{f}^{n}}{\Delta t}$$

$$y_t = \frac{y_{f}^{n+1} - y_{f}^{n}}{\Delta t}$$
for the x and y coordinate directions respectively. Values of $x_i$ and $y_i$ represent the velocity of the cell face in the x and y coordinate directions respectively, while $x_f$ and $y_f$ represent the locations of the cell face midpoint at the n and n+1 timesteps. These differences are calculated using the cell face centers of the median dual grid, instead of the coordinate locations. This was found to give a more accurate representation of the cell face movement for an unstructured grid. Modifying the cell face convective velocity to account for the grid movement results in the cell interface flux velocity

$$\mathbf{V}_{int} \cdot \mathbf{\hat{n}} A_f = (u - x_i) n_x A_f + (v - y_i) n_y A_f$$

(20)

where $\mathbf{\hat{n}} = n_x \mathbf{i} + n_y \mathbf{j}$.

Using the interface velocity defined by equation 20 introduces an error in the energy equation arising from the convection of enthalpy term in equation 17, where enthalpy is a combination of energy and pressure defined as

$$H = \frac{(E_x + P)}{\rho}$$

(21)

The modified convective velocity produces a non-physical reduction in enthalpy due to the pressure term in the convection of enthalpy equation in the flux calculation. A modification to the enthalpy term in the flux calculation is needed to remove the pressure correction by subtracting

$$PA_f (x, n_x + y, n_y)$$

(22)

This single point definition of cell face velocity has been shown to give temporally accurate results for a shock tube using a structured grid, where the cell faces primarily
undergo pure translation during adaption. However, for unstructured grids, and structured grids for more general problems, the cell faces will translate, rotate, and stretch as a result of node movement. In this case, assuming pure translation of the cell face will not give an accurate calculation of the cell volume change or the cell face flux. Therefore a new gridspeed definition is needed which will account for all modes of motion.

### 3.3 Deficiencies in Gridspeed Definition

To illustrate the conservation problems observed using the previous method, a closed volume with quiescent initial conditions will be examined. The domain is the same domain used for the shock tube problems with the pressure ratio set to 1:1. When the 1:1 shock tube is started, there should be no perturbation of the quiescent conditions and all state variables should remain constant. Using the adaption process in this situation causes small movements in the grid as nodes are relocated by the center of mass algorithm as the grid is effectively smoothed. This relaxes the grid to a more uniform distribution of areas. It also results in small distortions and rotations of the cell faces.

Figure 2 shows the unadapted initial grid. The distribution of cell area in the initial grid is generally uniform and there are no significant distortions in the cell or node distributions. Figure 3 shows the same section of the domain after five iterations through the adapter and flow solver, using the initial method of calculating the gridspeed terms without the averaging process. Since the pressure ratio was 1:1, there are no features within the flow for the adapter to resolve. To further eliminate any influence from the adaption process, the weight function for this simulation was set to a constant value, so
that every node would see the same weights, which typically has the effect of smoothing the volume distribution within the grid. However not enough adapter iterations have been performed to significantly displace the original grid and there is little change between the initial and adapted grids, as seen in the two figures.

Figure 2 - 2D Unstructured Grid Before Adaption.

Figure 3 - Adapted 2D Unstructured Grid without Weight Function.

Figure 4 shows the density contours for the adapted domain. Perturbation magnitudes have been exaggerated in the figure and are a result of using the original algorithm, with the variances generally ranging between $10^{-3}$ and $10^{-7}$ kg/m$^3$. Perturbations of this magnitude, however, are significant in the context of an unsteady flow solver, and are
well above tolerance values used for convergence of steady flows. These errors were reduced in magnitude by using the average cell face area in the flux calculation, but small perturbations persisted. These errors stem from the inability of the approach to correctly account for stretching and rotation produced by the adaption process and lead to the lack of conservation of the method. These errors were not of sufficient magnitude to generate significant errors in previous shock tube solutions, but could become significant in other situations. Adaption for flows requiring increased distortion and/or rotation would lead to an increase in the magnitude of these perturbations.

![Density Contours of Perturbations After Adaption.](image)

**Figure 4 - Density Contours of Perturbations After Adaption.**

Comparison of Figure 4 to Figure 2 and Figure 3 shows that the largest perturbations occur in locations with the greatest volume change. For unstructured grids, volume changes necessarily lead to changes in the cell face geometry, which cause the largest rotation, and therefore lead to the largest perturbations. More difficult to see is the distortion, cell face stretching or compression, caused as a result of the volume changes. Both of these effects combine to cause the perturbations seen in Figure 4.
The impact of these errors depends on the magnitude of the motion, or the amount of displacement for each node. In previous work on steady problems, no correction was made for the grid adaption and the grid movement acted as a perturbation to the solution. These perturbations were completely damped in the final steady solution as the grid motion ended. For unsteady problems, these errors still permit accurate solutions, as shown in previous work for structured and unstructured grids, if the magnitude of the errors is not significant compared to the errors associated with the method. Since the magnitude of the errors depends on the magnitude of the movement, the lack of conservation acts as a constraint on the node movement.

Figure 5 - Adapted Grid with Artificial Sink.

Figure 6 - Density Contours from Adapted Sink Grid.
To illustrate the constraint imposed by the process, another simulation is performed using the same domain with quiescent conditions. The flow solver is executed for five iterations, with three adaption steps per iteration. An artificial sink is added to the weight function to force an arbitrary clustering of the grid to an artificial feature. The adapted grid is shown in Figure 5. The most prominent feature is the clustered region of cells at the top of the domain at an X location of about 1.25. The corresponding density contours are shown in Figure 6.

Figure 5 and Figure 6 illustrate the problems that occur for unstructured grids in the presence of large grid movement. The adapter has produced a feedback loop, where the perturbations due to the lack of conservation in the approach are seen as flow features by the weight function. The adapter then tries to resolve the feature, producing more grid movement in the same region and thus larger perturbations. If the solution continues to be strongly adapted, the solver will eventually produce perturbations large enough to cause the solution to fail. It is important here to note two points. First, the perturbations produced in this solution are very large due to the use of the structured method for calculating the gridspeed terms. The averaging method reduced these errors to a manageable level, although it did not eliminate them. Second, when the same approach is applied to a structured shock tube, no perturbations are produced. This is a problem particular to unstructured grids.
3.4 New Gridspeed Definition

The interface velocity correction had previously been determined by calculating the average displacement of the cell face center as defined by the median dual grid\textsuperscript{37}. The displacement of the cell centers in the dual grid divided by the time step defined the cell face velocity. This calculation fails to introduce the correct information about distortion and rotation of the cell face as the grid is adapted. In contrast with structured grids, an unstructured grid is much more likely to produce rotation and distortion due to the random orientation of the cell faces and non-uniform connectivity of the nodes. In an initial attempt to reduce the distortion due to face rotation, the flux correction was applied to the average values of the cell face areas and normals, thus reducing the effect of large amounts of rotation or stretching caused by the adaption process. However, errors in conservation remained.

In order to correctly represent the movement of the grid and its effect on the solution, a new method is required that will account for the grid movement in an accurate and conservative manner. One approach is to base the correction on the actual volume change resulting from adaption. Towards this goal, the first step is to accurately determine the volume change produced by adaption. To determine this, the geometry of the grid must be examined in the context of the adaption process.

The unstructured grids used for this work are defined by a median dual as represented by Figure 7, where the solid lines represent the node connectivity of the base grid, and the dashed lines represent the imposed definition of the dual grid. The shaded regions
represent a left and right cell within the median dual grid cell definition. The flow solver and adaption process are based on this definition of the computational grid.

![Definition of Medial Dual Grid](image)

**Figure 7 - Definition of Medial Dual Grid.**

Figure 8 represents the median dual cell after an adaption step based on displacement of the shaded regions in Figure 7. The heavy lines correspond to the shaded areas shown in Figure 7 and the dashed lines represent the same dual cell after the adaption step. The three shaded regions represent the volume changes of the left and right dual cells, with the dotted shading representing the total volume change in the left dual cell and the two diagonal lined regions representing the volume change of the right dual cell. The dotted region represents the exact volume change due to adaption and the calculation of this volume forms the basis of the new velocity correction.

To accurately calculate the change, the volume is divided by a diagonal represented as the dividing line between the diagonal shaded regions on the right dual cell. The volume
change is thus reduced to two triangulated cells, and the volume of each may be calculated using the same methods used to determine the volume of the dual cells. The sum of these two volume changes represents the total volume change resulting from grid movement. This procedure is performed for each subcell within the dual cell. Thus the total volume change for the dual cell is accounted for by summing the volume changes for each subcell. Figure 8 shows the volume change associated with two subcells of the dual cell. The volumes to either side of the subcells show would be accounted for by the calculation of that subcells volume change.

![Figure 8 - Volume Displacement from Adaption.](image)

Now that the volume change is known it can be used to calculate the interface velocity correction. Using the volume change to calculate a cell face velocity is a departure from previous methods, and requires some discussion to understand the differences and their impact.
As previously discussed, problems of conservation arose when applying the velocity correction calculated from the displacement of cell face centers. These problems arose because the gridspeed velocity failed to correctly recapture the volume change. The most accurate representation of the grid movement would be to model the instantaneous changes in the grid as it is adapted. Unfortunately, the only information available to the solver are the initial and adapted grids. Application of the velocity correction to the adapted grid cells was shown to be too inaccurate to permit an unsteady solution. The use of average values for the cell face areas and normals reduced these errors to the point where an unsteady solution was possible, but was not completely conservative and did not accurately reflect the geometry changes.

Given the only information available are the initial and adapted grids, a method is developed based on the adapted grid at the new time level. At this point the idea of an equivalent motion is introduced. The velocity correction is determined in such a way that it yields a velocity value equivalent to the adapted dual face sweeping a volume along a direction vector equal to the volume change of the median dual cell. Thus when the velocity correction is applied to the adapted grid cell face and normal, the volume change produced by adaption is recaptured.

To determine a form of the gridspeed term based on volume change, a conservation equation was examined on a moving grid for a quiescent flow. Since the state variables are constant, the only nonzero terms are due to the grid movement. Any method which seeks to maintain conservation should reduce the grid movement terms to correctly reflect
the volume change. Equation 23 represents the integral form of the continuity equation with grid movement.

\[
\frac{\partial}{\partial t} \int_V \rho dV + \int_S (\rho V - \dot{X}_f) \cdot \hat{n} dA = 0
\]  

(23)

where \( \dot{X}_f \) represents gridspeed term to be defined. For static grid solutions, the volume term in equation 23 is constant, and the time derivative can be moved inside the integral. In the case of moving grids, the volume term is no longer constant and the time derivative must be rewritten according to Leibnitz’s rule

\[
\frac{\partial}{\partial t} \int_V \rho dV = \int_V \frac{\partial \rho}{\partial t} dV + \int_S \rho \dot{X}_f \cdot \hat{n} dA
\]  

(24)

For the specific case of a constant density represented by the quiescent flow\(^{42}\), equation 24 reduces to the following form when the density is taken to be a value of 1

\[
\frac{\partial V(t)}{\partial t} = \frac{\partial}{\partial t} \int_V dV = \int_S \dot{X}_f \cdot \hat{n} dA
\]  

(25)

Substitution into equation 24 gives

\[
\int_V \frac{\partial \rho}{\partial t} dV + \int_S \frac{\partial V(t_2)}{\partial t} \rho(t_2) - \frac{\partial V(t_1)}{\partial t} \rho(t_1)
\]

(26)

which for the case of constant density in a discrete form gives

\[
\rho \left[ \frac{\partial V}{\partial t} \right]_{t_1}^{t_2} = \frac{\rho \Delta V}{\Delta t}
\]

Substitution into equation 23 for the case of a quiescent flow with zero velocity, and rewriting in discrete form gives
\[
\frac{\rho \Delta V}{\Delta t} - \sum_{m=1}^{N_i} \left[ \rho (\hat{X}_f \cdot \hat{n}) A_f \right]_m = 0
\]  

(27)

where \(N_i\) represents the number of faces associated with the dual cell and \(\Delta V\) represents the total change in volume of the dual cell as a result of adaption. Dividing out the density term gives the quiescent, constant property discrete form of the continuity equation as

\[
\frac{\Delta V}{\Delta t} = \sum_{m=1}^{N_i} \left[ \hat{X}_f \cdot \hat{n} A_f \right]_m
\]  

(28)

which states that the total volume change over time must be represented by the sum of the projections of the gridspeed term onto the cell face normal. Equation 28 relates the contribution of each gridspeed term for each cell face to the total volume change for the cell. Referring to Figure 8, we can further break down the contributions of each subcell to the total volume change represented as

\[
\frac{1}{\Delta t} \sum_{m=1}^{N_i} \Delta V = \sum_{m=1}^{N_i} \left[ \hat{X}_f \cdot \hat{n} A_f \right]_m
\]  

(29)

where the total volume change of the cell is equal to the sum of the changes of each of the subcells. Examining one of these subcells, we can form a relation between the change in volume of the subcell and the projection of the gridspeed vector onto the cell face normal given by

\[
\frac{\Delta V_m}{A_f \Delta t} = (\dot{X}_f \cdot \hat{n})_m
\]  

(30)

where the subscript \(m\) denotes the contribution of a single subcell to the associated subcell volume change.
Figure 9 - Motion Vector to Define Face Displacement.

Note that after dividing out the area term, the left hand side now has units of velocity, and thus represents the magnitude of the velocity of the gridspeed term projected onto the subcell face. Equation 30 is an exact equation representing the magnitude of the projected gridspeed term realizing that the cell face area is a function of time $A_f = A_f(t)$. An exact definition of the gridspeed term would incorporate the instantaneous change in the cell face as a function of time. However in the discrete representation only the initial and adapted cell face geometries are known. Since the coupled approach incorporates the gridspeed term in the integration to the new time level, the adapted cell face geometry is used to define the gridspeed. To accomplish this, the sweep volume is defined as a quadrilateral volume assuming a constant cell face area equal the adapted grid cell face area. This introduces a sweep volume equal to, but not the same as, the original sweep volume defined in terms of the final grid geometry.
Equation 30 defines the magnitude of the projection of the gridspeed term onto the cell face normal in terms of the sweep volume and adapted cell face geometry. To determine the magnitude of the gridspeed term, $\hat{X}_f$ in Equation 30 must be defined in terms of the magnitude times a unit vector in the direction of the cell face displacement. Referring to Figure 9, the gridspeed term is defined as the displacement of the centroids of the initial and adapted cell face, where $\hat{n}$ represents the adapted cell face normal and $\hat{z}$ represents the unit displacement vector for the two cell face centers. Dropping the $m$ subscripts and rewriting Equation 30 in terms of a magnitude and unit displacement vectors gives

$$|\hat{X}_f| \hat{z} \cdot \hat{n} = \frac{\Delta V}{A_f \Delta t} \quad (31)$$

Dividing out the projection term from the left hand side gives the magnitude of the gridspeed term. If Equation 31 is then divided by the projected unit displacement vector $\hat{z}$, the resulting equation represents the gridspeed as defined by the conservation equation

$$|\hat{X}_f| = \frac{\Delta V}{A_f \Delta t (\hat{z} \cdot \hat{n})} \quad (32)$$

The gridspeed components can now be defined by equating the components for each coordinate direction from the right hand side of Equation 32. Since no restriction on the dimensionality has been used in the derivation of Equation 32, the equation will apply for two and three dimensional flows. The resulting components of the gridspeed terms are defined in two dimensions as
\[ x_r = \frac{\Delta V}{A_f \Delta t (\hat{z} \cdot \hat{n})} z_x \]  
(33)

\[ y_r = \frac{\Delta V}{A_f \Delta t (\hat{z} \cdot \hat{n})} z_y \]  
(34)

where \( x_r \) and \( y_r \) are the \( x \) and \( y \) components of the gridspeed term respectively, and 
\( z_x \) and \( z_y \) represent the \( x \) and \( y \) components of the displacement vector respectively. Using 
the new definitions of the inviscid flux velocity correction, the new interface flux velocity 
for a single subcell in two dimensions is defined as

\[ V_{in} \cdot \hat{n} A_f = (u - x_r) n_x A_f + (v - y_r) n_y A_f \]  
(35)

This definition of the flux velocity based on the new gridspeed terms produces a 
correction which preserves the geometric conservation of the governing equations. Use of 
the redefined interface flux combined with the coupled adaption algorithm results in a 
method which is conservative and maintains the accuracy of the method.

### 3.5 Movement Restriction

While application of the new gridspeed definition yields a method to maintain 
conservation and accuracy through the coupled approach, it does not address the issue of 
grid crossover or inversion. In two dimensions, the likelihood that cell inversion will 
occur is greatly reduced through the use of the dual cell. The center of mass algorithm 
defines a new node location for the center cell of medial dual grid, or subcells, associated 
with the node as shown in Figure 10. Using information from the new and old locations,
a movement direction vector is easily defined which describes the magnitude and direction of the displacement. The equation for the displacement vector is given by

\[ \mathbf{r} = (x - x_o) \mathbf{i} + (y - y_o) \mathbf{j} \]

(36)

where the subscript “o” represents the old node coordinates.

Figure 10 - Geometry of 2D Dual Grid with Motion Vector.

In order to limit the motion, values of the edge centers as defined by the median dual grid are used to define limits on the motion, so that the magnitude of the node displacement is not allowed to exceed the local edge center. Since cells become increasingly skewed in the presence of large gradients, restricting motion to the nearest edge center can over constrain the motion and reduce resolution. Over constraint can be mitigated to large degree by projecting the movement vector onto the direction vectors defined by a node and its neighbors. Taking the smallest projection as the direction of motion ensures that the motion is constrained only as much as is required by the local topology.
3.6 Time Accurate Validation

Validation of the new approach is accomplished by application of the method to two distinct flows. First, quiescent flow in a box is revisited to validate that perturbations observed from the previous definitions of gridspeed have been removed from the solution. Application to a quiescent flow validates conservation, but does not address the issue of time accuracy, which is required for accurate predictions of transient flows.

To validate conservation, the problem of a quiescent flowfield inside a closed domain, namely a box, is revisited. An artificial weight function sink is again used to cluster points at the center of the domain as shown in Figure 11. Twenty flow solver iterations are executed, with five adaption steps are performed for each timestep. Although the location of the sink has changed from Figure 5, the degree of adaption to the artificial feature is comparable in terms of the amount of clustering and the magnitude of displacement. Density contours resulting from application of the new gridspeed term, by contrast, show none of the perturbation which resulted from the previous gridspeed definition. In fact, there are no density perturbations in the flowfield to a double precisions measure of the fluctuations. The initial normalized density value of 1.0 is maintained in the entire flowfield as the grid is deformed to resolve the artificial sink over each of the timesteps. Thus the new approach has maintained conservation and completely eliminated the perturbations noted in the earlier solutions.
Application of the artificial sink demonstrates that geometric conservation has been maintained in the new gridspeed equation. To validate time accuracy, the gridspeed equation is applied to a 10:1 pressure ratio shock tube defined as a 2D square channel that is 5m long and 1m tall. The shock tube is an ideal case to measure time accuracy since there is an analytical solution to the problem which will describe the flow variables at a specified time. The simulation is now executed again to verify that the new adaption method is time accurate.

Figure 11 - Artificial Sink using New Gridspeed Definition.

Figure 12 - Adapted Shock Tube Grid.
Initial conditions for the shock tube are a pressure and density ratio of 10:1, with the baffle located at x=2.5m, or the longitudinal center of the channel. The solution is run to a non-dimensional time of 1, and then compared to the solution predicted by theory for the same conditions and time. Solution differences of density are used to form the weight function that will define the adapted node movement. Density contours were selected since they should identify the contact surface as well as the expansion region and the shock wave.

Figure 12 is a plot of the adapted grid associated with the shock tube at a non-dimensional time of 1. As expected, the largest degree of clustering occurs in the vicinity of the shock wave, which is the strongest feature of the flow. The contact surface location is also apparent from the adapted grid, but not as well resolved as the shock wave. Weak flow features like the expansion region have show only moderate resolution improvement, consistent with previous simulations.

Figure 13 is a plot of the adapted solution compared to theory. Most importantly, the adapted solution matches the contact and shock location predicted by theory, implying that the new approach has maintained time accuracy by matching the wave speeds of both features. Changes made to maintain conservation, while making the approach geometrically conservative, have not impacted the time accuracy of the method. There are some differences in the adapted solution compared to the exact solution in the region of the contact surface and the expansion region, but the differences are in the resolution of the feature and the location of the feature as defined by the midpoint of the adapted
contours. The differences in the contact surface are likely due to the dissipation in the upwinding method. Under prediction and over prediction at the ends of the expansion surface are due to lack of resolution of the expansion surface in general and the edges of the expansion wave specifically. Expansion waves represent weak features in the flowfield, and while clipping operations have shown improvements in the resolution of weak features, the level of resolution obtained for weaker flowfield features remains disappointing, but are comparable to results obtained previously for this and other methods.

![Graph](image)

**Figure 13 - Adapted Solution Compared to Theory.**

Figure 14 is a comparison of the unadapted grid shock tube versus theory. Contrasting with Figure 13 shows that the expansion contact surface regions are not significantly different, and the edges of the expansion region are captured better in the unadapted solution than in the adapted solution. From Figure 12 it would not be expected that there would be significant change in the expansion region due to adaption, since this
area has been largely ignored by the adapter, and the grid density is high in the expansion region. Differences in the expansion region edges are due to a slight stretching of the grid as nodes are moved towards the stronger features of the contact surface and shock wave. There has been some adaption to the contact surface, but dissipation in the method prevents significant resolution of this feature. The shock wave region, however, has shown a marked improvement over the unadapted results. Grid clustering in the region of the shock wave as seen in Figure 12 has reduced the transition region of shock wave and improved the resolution. Although not plotted, it has also reduced the oscillations in the contours of the dependent variables at the shock wave due to the increased resolution.

Figure 14 - Unadapted Grid Compared to Theory.
4 Unstructured r-Refinement in Three Dimensions

As shown in the gridspeed equation development, conservation and time accuracy for 2D solutions was maintained, and no restrictions on dimensionality were imposed on the gridspeed definition so that application of the method is easily extensible to three dimensions. Equation 32 is used to determine the local velocity of each face due to grid motion, which must then be incorporated into the flux calculation. Once the flux terms have been modified to account for grid motion, the integration scheme must be modified to account for volume variations, as was demonstrated for the 2D equation set. Extension of the unstructured r-refinement algorithm to three dimensions is accomplished by modification of the RavenCFD flow solver. RavenCFD is a three dimensional, parallel, general polyhedral commercially available perfect gas flow solver, derived from the government flow solver Cobalt6043,44 developed by the Air Force. RavenCFD is a derivative product of Cobalt60 provided to Corvid Technologies, Inc. under the technology transfer program. Modification of RavenCFD requires not only an extension of the adaption algorithm to three dimensions, but modifications necessary for a parallel implementation using MPI. Development will begin with modification of the integration scheme, followed by modifications to the flux scheme, and extension of the gridspeed terms to second order temporal calculations.

A brief overview of the numerical approach used by the current three dimensional flow solver is given in the following section based on references 43 and 44. Further details on the numerical approach can be found in the references.
4.1 Numerical Approach

The numerical approach involves solutions of the Euler and Navier-Stokes equations using a cell centered, finite volume approach based on the integral form of the conservation equations given by

\[
\frac{\partial}{\partial t} \iiint_V Q dV + \iiint_S (f_i + g_j + h_k) \cdot n dS = \iiint_S (r_i + s_j + t_k) \cdot n dS \tag{37}
\]

where

\[
Q = \begin{bmatrix}
\rho \\
u \\
v \\
w \\
e 
\end{bmatrix}
\]

\[
f = \begin{bmatrix}
\rho u \\
\rho u^2 + P \\
\rho uv \\
\rho uw \\
\rho \rho e + P 
\end{bmatrix}, \quad g = \begin{bmatrix}
\rho u \\
\rho u v \\
\rho v^2 + P \\
\rho vw \\
v (\rho e + P) 
\end{bmatrix}, \quad h = \begin{bmatrix}
\rho u \\
\rho uw \\
\rho vw \\
\rho w^2 + P \\
w (\rho e + P) 
\end{bmatrix}
\]

\[
r = \begin{bmatrix}
0 \\
\tau_{xx} \\
\tau_{xy} \\
\tau_{xz} \\
\mu \tau_{xx} + \nu \tau_{xy} + w \tau_{xz} + k \frac{\partial T}{\partial x} 
\end{bmatrix}
\]
Solution of the governing equations is accomplished by use of a point implicit Symmetric Gauss Seidel \(^{45,46}\) (LU-SGS) approximate matrix inversion method, with Newton sub-iterations employed to maintain time accuracy. Flux calculations are performed using either the approximate Riemann solver of Gottlieb and Groth \(^{47}\), or the flux vector splitting scheme of Edwards \(^{41}\). Jacobian matrices are formed using the split difference scheme of Van Leer \(^{48}\). Turbulent solutions are obtained using either the Spalart-Allmaras \(^{49}\) one equation model, Menter’s SST \(^{49}\) model or Wilcox’s \(k-\omega\) \(^{51}\) two equation models, solved in a fully or semi-coupled manner with the laminar viscous equations.

### 4.2 Modified Integration Scheme

Modification of the integration scheme requires accounting for the volume change in each cell as the grid is deformed due to grid motion. Newton sub iterations are required
to recover time accuracy by iterating out the effects of the volume change over the
timestep. To accomplish this, the base integration scheme, given by

\[ \theta \left\{ \frac{\partial Q}{\partial t} + \nabla \cdot f \right\}^{n+1} + (1 - \theta) \left\{ \frac{\partial Q}{\partial t} + \nabla \cdot f \right\}^n = 0 \] (41)

is modified to include the volume term in the time derivative. In equation 41, \( n + 1 \) and \( n \) represent values at the new and previous time level, respectively, and \( \theta \) is the implicit operator which ranges from 0 to 1, where 0 is fully explicit and 1 is fully implicit. Inclusion of the volume term leads to the new form of the basic equation given by

\[ \theta \left\{ \frac{\partial (VQ)}{\partial t} + \nabla \cdot f \right\}^{n+1} + (1 - \theta) \left\{ \frac{\partial (VQ)}{\partial t} + \nabla \cdot f \right\}^n = 0 \] (42)

where volume changes over time are now included in the time derivative terms. Expanding the time derivative term for each time level including the volume change gives a new discrete form of the time derivatives as

\[ \frac{\partial (VQ)}{\partial t}^{n+1} = \frac{\alpha_{1,1} \left( V^{n+1}Q^{n+1} - V^nQ^n \right) + \alpha_{1,2} \left( V^nQ^n - V^{n-1}Q^{n-1} \right)}{\Delta t} \] (43)

\[ \frac{\partial (VQ)}{\partial t}^n = \frac{\alpha_{2,1} \left( V^{n+1}Q^{n+1} - V^nQ^n \right) + \alpha_{2,2} \left( V^nQ^n - V^{n-1}Q^{n-1} \right)}{\Delta t} \] (44)

where \( \alpha_{1,1}, \alpha_{1,2}, \alpha_{2,1}, \alpha_{2,2} \) are coefficients used to define the temporal order of the scheme. For first order temporal accuracy, \( \alpha_{1,1} = \alpha_{1,2} = 1, \alpha_{2,1}, \alpha_{2,2} = 0 \), and for second order temporal accuracy, \( \alpha_{1,1} = 3/2, \alpha_{1,2} = -1/2, \alpha_{2,1}, \alpha_{2,2} = 1/2 \).

Flux values at the new \( n + 1 \) timestep are obtained by expanding flux term in a Taylor series expansion at time level \( n \), represented by
\[ f^{n+1} = f^n + \left( \frac{\partial f}{\partial Q} \right)^n (Q^{n+1} - Q^n) + O(\Delta Q^2) \]

\[ = f^n + A(Q^{n+1} - Q^n) + O(\Delta Q^2) \]  

(45)

where \( A \) represents the flux Jacobian matrix. To properly account for upstream and downstream signal propagation, the Jacobian must be split in two components defining the upstream and downstream values represented by

\[ \frac{\partial f}{\partial Q} = A\Delta Q = A^+\Delta Q^+ + A^-\Delta Q^- \]  

(46)

Substitution of the derivative and flux terms into the original equations gives the discrete form of the integration scheme as

\[ \theta \alpha_{1,1} \left( V_{n+1}^i Q_{n+1}^i - V^n Q^n \right) + \alpha_{1,2} \left( V^n Q^n - V^{n-1} Q^{n-1} \right) \frac{\Delta t}{\Delta t} \]

\[ + \theta \sum_{m=1}^{N_i} A^\top_m \hat{n}_m S_m (Q^{n+1} - Q^n)_m + \theta \sum_{m=1}^{N_i} A^\top_m \hat{n}_m S_m (Q^{n+1} - Q^n)_m \]

\[ (1 - \theta) \alpha_{2,1} \left( V_{n+1}^i Q_{n+1}^i - V^n Q^n \right) + \alpha_{2,2} \left( V^n Q^n - V^{n-1} Q^{n-1} \right) \frac{\Delta t}{\Delta t} \]

\[ + \theta \sum_{m=1}^{N_i} \tilde{f}_m^{n+1,k} \hat{n}_m S_m + (1 - \theta) \sum_{m=1}^{N_i} \tilde{f}_m^n \hat{n}_m S_m = 0 \]  

(47)

where \( i \) represents each cell in the grid, \( m \) represents a neighbor cell with a connecting face to cell \( i \), and \( N_i \) represents the total number of neighbors for cell \( i \). Coefficients for the time derivative terms can be reduced into two parameters, giving the simplified form of the integration equation as
\[
\frac{\eta_1}{\Delta t} \left( V^{n+1} Q^{n+1} - V^n Q^n \right) + \theta \sum_{m=1}^{N_m} A_m^\pm \hat{n}_m S_m \left( Q^{n+1} - Q^n \right)_m = \\
- \frac{\eta_2}{\Delta t} \left( V^n Q^n - V^{n-1} Q^{n-1} \right) - \theta \sum_{m=1}^{N_m} A_m^\mp \hat{n}_m S_m \left( Q^{n+1} - Q^n \right)_m \\
- \theta \sum_{m=1}^{N_m} f_m^{n+1,k} \hat{n}_m S_m - (1 - \theta) \sum_{m=1}^{N_m} f_m^n \hat{n}_m S_m 
\]

(48)

where the new coefficients \( \eta_1, \eta_2 \) are defined as

\[
\eta_1 = \alpha_{2,1} + \theta(\alpha_{1,1} - \alpha_{2,1}) \\
\eta_2 = \alpha_{2,2} + \theta(\alpha_{1,2} - \alpha_{2,2})
\]

(49)

Equation 48 represents the discrete form of the time dependent integration of the solution vector \( Q \) with volume change \( V \) over timestep \( \Delta t \). Solution of equation 48 requires linearization of the term \( \left( V^{n+1} Q^{n+1} \right) \) to solve for values of \( Q \) at the new time level \( n+1 \). Newton sub-iterations are introduced by expanding the solution at \( n+1 \), and introducing \( k+1 \) as the new values of the solution vector \( Q \) based on the solution at \( n,k \), where \( k \) represents the solution at the current Newton sub-iteration. Expansion of the term \( \Delta(VQ)^{n+1} \) is given by

\[
\Delta(VQ)^{n+1,k+1} = \Delta(VQ)^{n+1,k} + \frac{\partial(VQ)}{\partial t} + O(\Delta t^2)
\]

(50)

Since the values of \( V \) are held constant over the Newton sub-iterations, the volume term can be pulled outside the \( \Delta(VQ)^{n+1,k} \) term. Substitution into equation 48 gives
\[
\left[ \eta_i V^{n+1} I + \theta \Delta t \sum_{m=1}^{N_i} \bar{A}^{+}_m \hat{n}_m S_m \right]^{n+1,k} (Q^{n+1} - Q^n)^{n+1,k+1} = \\
- \eta_2 \left( V^n Q^n - V^{n-1} Q^{n-1} \right)^{n} - \theta \Delta t \sum_{m=1}^{N_i} \bar{A}^{+}_m \hat{n}_m S_m (Q^{n+1} - Q^n)_{m}^{n+1,k} \\
- \theta \Delta t \sum_{m=1}^{N_i} \bar{f}^{n+1,k}_m \hat{n}_m S_m - (1 - \theta) \Delta t \sum_{m=1}^{N_i} \bar{f}^{n}_m \hat{n}_m S_m - \eta_1 \left( V^{n+1} Q^{n+1} - V^n Q^n \right) (51)
\]

where the last term on the right hand side represents the unsteady part of the residual resulting from the volume change from time level \( n \) to \( n + 1 \). Equation 51 can be used in the current form to modify the integration scheme to account for grid motion at each timestep. Although it is mathematically correct, cell volumes in practice can have a wide range of values and are typically very small which can introduce numerical error into the scheme. Ideally the new form of the equations should reduce to the previous form in the absence of volume change and hence grid motion. To accomplish this, velocity ratio terms are introduced to account for the ratio on volume change due to grid motion. Substituting the appropriate volume ratio term gives the final form of the equations as

\[
\left[ \eta_i I + \frac{\theta \Delta t}{V^{n+1}} \sum_{m=1}^{N_i} \bar{A}^{+}_m \hat{n}_m S_m \right]^{n+1,k} (Q^{n+1} - Q^n)^{n+1,k+1} = \\
- \eta_2 V_r^n \left( Q^n - V^{n-1}_r Q^{n-1} \right)^{n} - \frac{\theta \Delta t}{V^{n+1}} \sum_{m=1}^{N_i} \bar{A}^{+}_m \hat{n}_m S_m (Q^{n+1} - Q^n)_{m}^{n+1,k} \\
- \frac{\theta \Delta t}{V^{n+1}} \sum_{m=1}^{N_i} \bar{f}^{n+1,k}_m \hat{n}_m S_m - (1 - \theta) \frac{\Delta t}{V^{n+1}} \sum_{m=1}^{N_i} \bar{f}^{n}_m \hat{n}_m S_m - \eta_1 \left( Q^{n+1,k} - V^{n+1}_r Q^n \right) (52)
\]

where the volume ratio terms are given as
\[ V_{r}^{n+1} = \frac{V_{n}}{V_{n+1}} \]

\[ V_{r}^{n} = \frac{V_{n-1}}{V_{n}} \]  \hspace{1cm} (53)

### 4.3 Flux Scheme Modifications

Two flux schemes are available in RavenCFD, the approximate Riemann solver of Gottlieb-Groth\(^{45}\) and the flux splitting scheme of Edwards\(^{41}\). Incorporation of gridspeed terms for the three dimensional flow solver is decoupled from the calculation of the flux interface variables, so that application of the gridspeed is independent of the flux scheme being employed. In the development of the two dimensional gridspeed terms, flux corrections were made by subtracting the grid face velocity from the convective velocity at each face, as defined by the LDFSS flux splitting scheme of Edwards. In the two dimensional case, implementation of the flux corrections was carried out in the flux calculation, tightly coupling the grid motion terms to the flux scheme. In the three dimensional case a more general approach to the flux correction has been employed where the gridspeed terms have been applied after calculation of the flux interface variables, and are therefore independent of the flux scheme. Each flux scheme defines flow quantities at the face based on the details of the flux calculation for each method, and an array of flow variables is created and stored for each face in the computational grid. Adjustments for grid velocity are made to the normal velocity component at each face independent of the flux scheme used to generate the flow variables at the face. Thus the net inviscid flux equation is given by
\[
\vec{F}_{net} = \sum_{n=1}^{N_f} \begin{bmatrix}
\rho(U_n - \dot{X}_f) \\
\rho(U_n - \dot{X}_f)u + Pn_x \\
\rho(U_n - \dot{X}_f)v + Pn_y \\
\rho(U_n - \dot{X}_f)w + Pn_z \\
\rho(U_n - \dot{X}_f)H + \dot{X}_f P
\end{bmatrix}
\]  

(54)

where \( \dot{X}_f \) represents the grid velocity of each face, and \( \dot{X}_f P \) is the pressure correction term necessary to compensate for the pressure term in the convection of enthalpy as defined in the gridspeed development.

Calculation of the gridspeed term proceeds from equation 32 developed from the volume change at each face as the face is displaced due to grid motion. In the two dimensional case face motion was defined by the displacement of an edge, which resulted in a rectangular area describing the volume change as shown in Figure 9. The resulting volume change was calculated by subdividing the volume into two sub-volume elements represented by triangles. In the three dimensional case, each face in a tetrahedra is decomposed into a separate sub-volume as defined by the cell center, the face center, and two adjacent nodes on the cell face. A tetrahedral element will contain three sub-volumes corresponding to each face, for example. New and old values of the node locations are stored in the adaption process, and are used to determine the new and old cell center and face locations before and after adaption. Thus the displacement of each face in an element can be described from the new and old cell locations as shown in Figure 15.
Face displacement during adaptation represents a volume change which is described by a prism element for a triangular face, as shown by the blue lines in Figure 16. As with the two-dimensional case, there are a number of modes of motion possible for a triangular face in three dimensions. In terms of gridspeed, however, only motions that result in displacement of the face center are of interest. If, for example, the grid face is stretched in such a way that the displacement vector for the face center is perpendicular to the face normal, as might occur at a planar boundary, the gridspeed for that face is zero even though the change in the sub-volume is non-zero. This type of displacement is generally accounted for in the $\hat{z} \cdot \hat{n}$ term of the gridspeed definition, but since it occurs in the denominator, care must be taken to ensure a finite, or in this case zero, grid velocity. Volume changes resulting from this type of displacement are accounted for in the volume
change calculation for the remaining faces in the cell, so that the net volume change associated with each face equals the total volume change of the cell.

Figure 16 - Volume change from 3D face displacement.

Calculation of the volume change described by the prism formed from the old and new face locations is determined by deconstructing prism volume into a series of sub-volume changes as shown in Figure 17. In the diagram, each face at the previous and current time level is broken down into three sub-faces as shown in blue. A center point between the old and new face locations is determined from the center points of the face at each time level as described by the blue line connecting each triangular face. Defining a tetrahedral sub-volume as two consecutive points on each face, the face mid-point, and the point at the center of the prism volume, decomposes the prism volume into 18 separate sub-volumes representing the original prism volume. An example of one of the sub-volumes is shown in green on Figure 17. Sub-volumes for each quadrilateral face are
defined by the face center of the quadrilateral, two consecutive face points, and the prism
volume center. The advantage of this approach is that each sub-volume forms a
tetrahedra, independent of the original geometry, and calculation of the prism volume
displacement follows easily from the sum of all the tetrahedral sub-volumes. Old node
locations are stored during the adaption process to calculate the volume change and are
then discarded, so that the overhead of storing old grid information is limited to the
adaption cycle.

Although adaption in the current work is applied to tetrahedral meshes, or at least to
tetrahedral elements in a mesh, the method can be applied in general to any cell geometry
provided that accurate determination of the volume change can be obtained. By
deconstructing the volume change into sub-volumes, volume changes resulting from the
motion of any arbitrary polyhedral face construction can be determined in a general
fashion as the sum of tetrahedral sub-volume changes representing the total volume
displacement.
Accurate calculation of the volume change provides all the information necessary to compute the gridspeed velocity for each face. A last issue remains in determination of the second order gridspeed equation. Equation 52 indicates the volume change must be accounted for at three time levels as defined by the three point stencil used for second order temporal computations. Since the gridspeed term was developed to exactly compensate for the volume change at each timestep, the grid velocity term must be modeled after the time operators to accurately match the grid motion with the volume change. Since the volume change is coupled with the flux correction, the gridspeed must be coupled with the volume change from the previous iteration. To accomplish this, the solution difference from the previous timestep is augmented with the flux correction at that timestep, resulting in a new residual term in the form of

$$\eta_2 V_{r}^{n+1} \left[ (Q^n - V_r^n Q^{n-1})^n - \hat{X}_f \sum_{m=1}^{N_f} \hat{f}_m \hat{n}_m S_m \right]$$

(55)

and the gridspeed term for the current time level is given by

$$\dot{X}_f = \eta_1 \dot{X}_f^{n+1}$$

(56)

where the $\eta$ coefficients match the values in the integration scheme, and the gridspeed terms are defined at the current and previous time level. The new form of the equations reduces to the original gridspeed definition for first order temporal calculations, recovering the original method, and since information for the previous time level is already stored, no additional computation overhead is incurred from the additional terms.
Given that all development of the gridspeed terms for three dimensional grids is carried out with only information from the current and previous node locations, application of this method to enrichment method is feasible. If node locations for the previous time level are stored as each node is created during enrichment, all the necessary information will be available to compute the gridspeed terms. Use of this form eliminates any issues associated with smoothing operations which must be carried out after each enrichment step. Most smoothing operators, such as Laplacian and optimized smooth, are node relocation methods which are treated in the same manner as motion due to r-refinement, since they are in effect a form of r-refinement adaption. Smoothing for unstructured meshes, however, necessitates the use of edge swap routines which create new cells as part of the edge reconstruction. Although the edge reconstruction creates new cells and faces in the grid, it does not create new nodes. Definition of the volume change relied only on current and previous node locations, making it possible to compute face velocities even though faces are reconstructed during the smoothing process. The result is a generalized approach to coupling the grid motion to the solver in a time accurate manner independent of the type of refinement that is performed, whether is it is r-refinement, h-refinement, or p-refinement.

4.4 Implicit Jacobian Modifications

Modifications to the Jacobian matrices are required to account for grid velocities defined through the gridspeed terms as adjustments to the velocity terms in the Jacobian.
As noted previously, Jacobian matrices are formed using Van Leer’s split flux scheme given for subsonic flows by

\[
F^\pm = \rho a M_n^\pm \begin{bmatrix} u \\ v \\ w \\ H \end{bmatrix} + \begin{bmatrix} 0 \\ n_x \\ n_y \\ n_z \end{bmatrix} P^\pm
\]

(57)

where

\[
M_n^\pm = \frac{1}{4}(M_n \pm 1)^2
\]

\[
P^\pm = \frac{1}{4}(2 \mp M_n)(M_n \pm 1)^2
\]

and by equation 39 for supersonic flows based on the propagation direction of waves entering or leaving the cell. In each case, the interface velocity is defined by the normal face velocity \(U_n\), which implies an interface Mach number of \(M_n = \frac{U_n}{a}\). Grid motion is accounted for in the Jacobian matrix by changing the definition of the interface velocity to include the gridspeed, written as

\[
V_{net} = U_n - x_f
\]

(58)

where \(V_{net}\) is the net normal velocity at each cell face defined as the difference of the normal fluid convection velocity minus the gridspeed. Replacing the normal velocity with \(V_{net}\) adjusts the velocity terms in the Jacobian matrix in a consistent manner with changes to the flux velocity equations on the right hand side of the equations.
Corrections to the interface velocity also entail a correction to the pressure terms through the enthalpy term in the energy equation Jacobian. Analogous to the pressure correction required for the flux scheme, a pressure correction term is required to ensure the velocity correction is applied only to convection of energy term and not to the flow work term defined by the pressure. The correction is defined by a modification of equation 57 given by

\[
F^\pm = \rho a M_n^\pm \begin{bmatrix} 1 \\ u \\ v \\ w \\ H \end{bmatrix} + \begin{bmatrix} 0 \\ n_x \\ n_y \\ n_z \\ \chi_f \end{bmatrix} P^\pm
\]

Thus equations 52, 54, and 59 represent the equation set changes necessary to the 3D flow solver to represent grid motion due to r-refinement. Implementation of these equations in RavenCFD results in a conservative, time accurate 3D motion correction algorithm capable of describing grid motion in two and three dimensions.

### 4.5 Time Accurate Validation

Initial validation was performed by application of specified grid motion to a steady state channel flow. Figure 18 shows a 2D slice of a steady state channel with an inflow boundary on the left of the figure, and an outflow boundary on the right. Slip walls were used for the side walls, and a freestream Mach number of 0.1 was used throughout the flowfield, and the overall length of the channel was 10m. An artificial transient weight function was created which started at the 2.5m location, as shown at the top of Figure 18,
and moved at 1m/s down the channel until reaching the 7.5m location as shown at the top of Figure 19. After reaching the 7.5m mark, the weight function reversed and traveled upstream in the channel until reaching the 2.5m starting point. The purpose of the calculation is to verify conservation by showing that no perturbations are introduced to the steady channel flow as a result of transient grid motion. Figure 18 and Figure 19 are 2D slices of the 3D channel centerline at equal increments in time as the weight function wave traverses the channel. Each pair in the figures consists of the grid and velocity vectors at each timestep, where the vectors are colored by pressure. Deformation of the grid is evident in both the grid and velocity vector plot, however, there are no perturbations in pressure, or any of the remaining flow properties, as a result of the grid motion. In fact, there is no variation in any flow property within the measured precision of the solution, which was double precisions in this case.

Notable in Figure 18 is the grid motion bias based on the direction of the wave propagation. As the wave propagates to the right following the downstream flow, there is an obvious tendency of the grid to skew cells in front of the wave motion, whereas cells behind the wave are more evenly distributed. As the wave reverses direction to travel upstream, shown in Figure 19, the bias also reverses direction and skews cells in the upstream direction, while the downstream cells are now more evenly distributed. Grid motion bias is not related to flow properties in the solution, given that the bias follows the artificial wave motion and not the flow direction. The bias is instead a result of the weight function and center of mass calculation, and how it is applied to a transient problem. In a transient problem, the weight function is calculated and then smoothed as
described earlier. As a result, cells from outside the influence of the local gradient are moved closer to the gradient to improve cell distribution.

Figure 18 - Transient Grid Wave Moving in Positive X.
Figure 19 - Transient Grid Wave Moving in Negative X.

As the wave function propagates, cells in front of the wave have a higher relative velocity, compared to the wave speed, than cells behind the wave, resulting in skewness.
in front of the wave and a relaxation of cells behind the wave. This effect is generally not an issue so long as the gradients are resolved, but can become an issue in very large gradient flows, as demonstrated by White. Treating the grid motion as an elliptical problem and repeating the r-refinement step until grid motion stops, or in effect until the grid is converged, will alleviate the issue. Application of this technique, however, requires interpolation to reconstruct the weight function at iteration so the weight function and gradient location is not decoupled from the grid location as the grid moves.

Of equal importance to the grid clustering at the weight function location is the smoothing of the grid away from the weight function as the wave passes through the grid. Regions outside of the weight function influence are smoothed back to their original quality, removing any residual effects of the grid deformation once the weight function propagation wave has passed. Although the original grid exhibited high cell quality and nearly uniform connectivity, application of a Laplacian based smoother is still inadequate to recover the original grid quality and distribution, especially in areas such as boundaries in general and corners in particular. Recovery of the original mesh can only be accomplished by use of an optimized smoothing technique, as described in Appendix A. As outlined earlier, a Laplacian based smoothing approach, in the form of a center of mass calculation, was sufficient to cluster grid points in regions of high gradient, while acting as a grid smoother in regions with little or no gradient. Clustering and smoothing could be accomplished through use of the center of mass routine applied to the weight function in a single operation. In the 3D case, the Laplacian smoother alone is not sufficient to smooth the grid in regions where the weight function is small, and the
optimized smoother is too expensive to use for grid clustering based on weight function gradients. Therefore a two tier approach was used to address the issues of clustering and smoothing as separate operations. First, a generalized smoothing operation, consisting of Laplacian and optimized based smoothing is used to blend grid volumes in regions where the weight function is essentially flat. A minimum quality value is used to restrict use of the optimized smooth to regions where the grid quality is exceptionally poor. Second, a Laplacian operation in the form of a center of mass algorithm is used to cluster points in regions where the weight function values are large. Thus regions near flow gradients as described by the weight function are resolved, while regions outside the influence of the weight function are smoothed to remove residual effects of the grid motion.

Steady state results for channel flow with grid deformation validate conservation of the 3D approach, but not time accuracy. To validate time accuracy, a 3D shock tube problem is constructed using the same dimensions as the 2D case, namely a 5m tube with the pressure baffle located at 2.5m. A pressure and density ratio of 10:1 is used for the 3D shock tube, but the high pressure region is set to a value of 7.142857 PSI, or a value of $10 \times 1/\gamma$. The reason for the change is the non-dimensional formulation used in the 3D flow solver. Density values are normalized by the reference density, but pressure is normalized by reference density multiplied by the reference speed of sound squared, or $1/(\rho a^2)$. Using this formulation carries a $1/\gamma$ term which changes the value of pressure relative to the value of density, but does not change the pressure ratio. In other words the pressure ratio is still 10:1, but the values of pressure and density differ by a factor of $1/\gamma$. Therefore the calculation of the exact values for the Riemann problem was changed to
account for the differences in pressure and density when comparing to the unadapted and adapted solutions. Solutions for the shock tube problem were run to a non-dimensional time of 1, using a non-dimensional timestep of 0.002s (2ms) for 500 iterations. The dimensional timestep was 5.8778e-6s, and 64 Gauss-Seidel sweeps were used with 5 Newton iterations per timestep. The solutions were run second order in space and time, including the second order equation for the gridspeed term. Theta values of 1.0, or fully implicit, and 0.75 for semi-implicit, were used and found to have no influence on the final solution for the given case and conditions. No timestep study was conducted for the shock tube solutions since time accuracy is apparent by comparison with the exact solution at a given point in time.

Figure 20 shows results for initial, intermediate, and final values of the shock to tube solution, with each instance in time represented by the grid and density contour, respectively. Solutions for the shock tube problem were produced using 3D tetrahedral mesh generated using the ICEM CFD mesh generator from Ansys, Inc. Values of the grid and solution in the figure represent cut planes perpendicular to the Z axis at the midpoint of the shock tube. The geometry is simple enough that a uniform tetrahedra mesh is produced with nearly uniform connectivity, although there are some variations in cell structure and quality near the baffle at the center of the domain. Five iterations of pre-clustering to the baffle location were performed before the solution was started to improve resolution of the initial problem, although it is difficult to discern the initial clustering from the figure. More apparent is the grid clustering for the intermediate and final values of the grid and density solution. Improved resolution of the salient features,
namely the expansion, contact, and shock locations are readily apparent in the grid results, although the degree of clustering is much less that what was obtained for the 2D solutions. The degree of grid clustering is a direct function of the weight function used to identify flow features. Gradients of density were used for the current solution since density is able to identify the contact discontinuity as well as the expansion and shock features. Improved resolution of the flow features can be obtained by increasing the weight function spreading values, particularly the shock. Increasing values of the spreading function, while improving clustering, also lead to stability issues in the solution due to very large relative gridspeeds. Stability problems are exacerbated further when the timestep is reduced, as would be normal procedure for a timestep study. The underlying cause of the stability issues results from large grid motions as the solution is adapted. Grid motion is restricted by the localized cell geometry, but generally not by any stability criteria. In a case such as the shock tube, nodes near the baffle will move as far as allowed by the local connectivity to the baffle location. Since this motion is a function of cell structure and not stability, it is independent of timestep, thus decreasing the timestep will have no effect on motion but will result in increasing values of gridspeed. As the timestep decreases the grid motion can approach supersonic velocity magnitudes, effectively changing the character of the equations. In the current work, stability issues were addressed by limiting the magnitude of the initial motion until a stable solution had developed. Ramping the grid motion as the solution started proved sufficient to maintain stability throughout the solution evolution.
Figure 20 - Shock Tube Results for Starting, Middle, and Intermediate Solution.

In general, a stability based definition of motion restriction would be desirable, where a parameter similar to a CFL value is used to define a motion restriction parameter based
solely on stability considerations. CFL itself is not sufficient to describe grid motion stability since CFL scales with timestep, whereas stability for grid motion scales with the inverse of timestep.

![Figure 21 - Unadapted Shock Tube Results.](image)

Figure 21 presents a comparison of the unadapted solution with the exact solution for density, pressure, and velocity. Resolution of the features, including the discontinuities, is reasonably good due to the density and uniformity of the starting grid. There is a fluctuation at the leading edge of the expansion wave showing the dispersive nature of the second order method. Some spreading of the contact discontinuity is present, although there are two points well aligned with the jump conditions which should be sufficient to
resolve the discontinuity. Inset into the figure is a close-up of the shock transition region for density and pressure, respectively. Blurring of the shock transition due to insufficient resolution is evident, although the shock is correctly located indicating the solution is time accurate.

![Figure 22 - Adapted Shock Tube Results.](image)

Adapted results for the shock tube solution are presented in Figure 22 using the same initial conditions as the unadapted solution. Resolution of the expansion wave, particularly at the leading and trailing edges of the expansion, show a small improved agreement with the exact solution unlike the 2D solutions where no material change in resolution was noted for adapted solutions compared to unadapted solutions, and some degradation of the expansion leading and trailing edges was noted. Similarly, resolution
of the contact surface shows some small improvement, where now there are three points in the proximity of the discontinuity. The shock location, also visible in the inset section of Figure 22 shows some improvement in the shock location commensurate with the improved resolution observed in the grid results.

It should also be noted that data points that appear to overlay each other in the plots are actually regions where the data extraction has located two points instead of a single point. Data extraction was performed using a flow visualization tool, and in some locations multiple points were extracted due to variations in the spanwise distribution of nodes due to adaption. Another side effect of the plotting software that should be noted is the way in which the data is represented in the flow visualization tool compared to the way it is stored in the flow solver. As noted earlier, the flow solver is a cell centered construction, meaning the flow properties are stored at the cell centers. Data for the flow visualization tool, however, is stored at the node location. Therefore data from the flow solver must be averaged to the nodes, resulting in apparent dissipation of the discontinuities. It is evident from the grid results in Figure 20 that there has been moderate clustering around the contact and shock discontinuities, and the adapted results indicate a moderate improvement in resolution, however the improvement in resolution is skewed by the averaging process. Likewise, unadapted results reflect more dissipation at the discontinuities than is present in the flow solution. Results for both are still comparable, but the adapted results are more impacted by the averaging process than the unadapted results. In the unadapted case, the uniformity of the grid means that the averaging process does not retain any bias from the background grid since all cells are
comparable in size and structure. In the adapted case, cells are clustered around the shock location based on the previous solution, but the solution is updated after the adaption step, lagging the adapted results. Lagging the adaption combined with the non-uniformity of the grid at the shock location leads to a small increase in spreading for the adapted case, which is to say that the adapted solution likely shows improved resolution over what is presented in the figures.

Shock tube results demonstrate that implementation of the new gridspeed terms are time accurate for moving grids. Although the grid clustering for the 3D cases was not as extreme as the 2D results, improved resolution of the salient shock tube features was present in the results, and no perturbations to the flowfield resulted from the grid motion. Contact and shock discontinuities were correctly tracked and located as the solution evolved in a time dependent manner. Shock tube results combined with steady flow results demonstrate that the method is conservative and time accurate and does not introduce any perturbations to the flowfield for first and second order discretizations in space and time. A last demonstration case of a 3D airfoil solution executing a periodic motion is presented in the next section.
5 Plunging Airfoil

As a demonstration case, the time accurate r-refinement algorithm was extended to a 3D plunging airfoil configuration which produces a temporally oscillating flowfield. A plunging airfoil was selected as a demonstration case due to the practical application of the problem and availability of comparison data. Plunging and plunging/pitching airfoils are of interest to several fields of study such as simulation of biological flight, Micro Air Vehicles (MAV), and gust response, for example. An understanding of these flow structures can yield insight into the roles of unsteady flow responses on flight mechanics, such as vortex formation and shedding, and the impact these flow structures have on flight dynamics. Solution of these types of flowfields requires grid motion, and if time accurate results are to be obtained a motion algorithm that removes perturbations due to grid motion is required.

Comparison data for the plunging airfoil is based on the experiments of Ol53, using an SD7003 airfoil section as the reference geometry. CFD solutions were developed to match the reduced plunge frequency $k$ and Reynolds number of the experiment, where the reduced plunge frequency is defined as $k = \frac{\pi c}{U_{\infty}}$. For the current solution set, values of $k = 3.93$, and $Re = 60,000$ were used. The chord length, $c$, of the airfoil section is 200mm, and the freestream Mach number is 0.1287638, which yields a freestream velocity of 43.81355m/s.

Experimental data for the plunging airfoil was collected from the Air Force Research Laboratory’s Horizontal Free-surface Water Tunnel (HFWT) using a particle image
velocimetry (PIV) system described by Ol$^{53}$. A constant section wing which spanned the water tank was used as the test geometry, with a small gap between the wing section and the tunnel side wall. The wing span of the model was 450mm. A drive motor motion system connected to the lower or high pressure side of the wing section was used to produce a sinusoidal motion, and data was collected over a large number of frames and averaged at four locations in the plunge cycle. The locations where data was collected were at the maximum peak at $h=0.05$, the centerline on the downward stroke, the minimum peak at $h=-0.05$, and the centerline on the upward stroke. These locations are referred to as $\phi = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ respectively. Results are compared to PIV data from the experiment as well as CFL3D$^{54}$ results obtained by McGowan and Gopalaratham$^{55,56}$.

### 5.1 Grid Resolution and Motion

Computational modeling of the experiment consisted of a constant section wing matching the SD7003 airfoil section used in the experiment, with symmetry boundaries located on each end of the wing so that the geometry is treated as a wing of infinite span. A far field boundary condition using Riemann invariants was used to simulate inflow and outflow conditions. The span of the computational model was 0.0125m and the outer boundaries were located 5 chord lengths away from the model in each direction. Grids with approximately 5.79 million cells were used for the computational solutions.

Grid resolution for the current model was determined by using a combination of grid spacing and density boxes, which are used to refine volume elements. Figure 23 shows a
view of the airfoil surface resolution and the symmetry plane resolution near the wing. Surface spacing of 0.0005m was used on the leading edge section of the airfoil and 0.005m spacing was used on the remaining sections. Spacing on the trailing edge of the model was 5.0e-5m. The resulting grid contained approximately 35 cells in the spanwise direction on the leading edge, and approximately 20 cells on the remaining sections of the wing.

Figure 23 - Airfoil Surface and Volume Resolution.

Figure 24 - Near Body Resolution.
Density boxes, which are used to refine the volume grid, were used near the wing surface as shown on the right of Figure 23 and in Figure 24. Grid spacing of the near body density box was 0.0015m and spacing on the outer section was 0.0075m.

Spacing on the trailing was set to 5.0e-5m resulting in between one and two cells vertically across the trailing edge, and a slow transition blending function was used to transition from the surface mesh to the volume mesh. The left side of Figure 25 shows the trailing edge region of the volume grid at the centerline, while the right side shows a close-up view of the trailing edge. The original model contained a radius on the trailing edge, which did not translate into the grid tool. Therefore the trailing edge was modeled as a straight edge with sharp corners. The region near the trailing edge appears to be well refined in the left figure, but there are large cells relative to the trailing edge spacing in the trailing edge region of the wing section. Flow around the trailing edge will be very important for the plunging airfoil solutions since the body will be in motion, and resolution of the trailing edge region may have an impact on results.

Grid motion was simulated by applying a bulk velocity parameter to the volume and surface grid near the wing body (shown inside the red box on Figure 26) while keeping
the outer sections of the grid constant, represented by the larger spacing around the perimeter of the figure outside the red box. Blending and smoothing between the moving and static portions of the grid was accomplished by applying the grid smoothing algorithm described in Appendix A. The gridspeed term defined by equation 32 was used to compute the grid face velocity for all surface and volume faces in the grid, both in the bulk motion section and blended section of the grid. As a result, higher density regions of the grid saw a constant spatial grid velocity, while the outer regions where the blending occurred saw a velocity dependent on the amount of motion required to smooth the grid in the interface region.

Figure 26 - Grid Motion Region.

Figure 27 shows the grid domain for the top and bottom of the cycle of motion for the background grid. The figure is an overlay of each grid, with the top of the cycle
represented by the black grid, and the bottom of the cycle represented by the red grid. The first frame in the figure shows the entire computational domain with each subsequent frame representing a subset of the domain closer to the wing section. The outer coarse grid region shows the far field boundaries and outer grid domain remain fixed during the motion cycle, while the inner refined region of the grid moves according to the plunge cycle. The second frame illustrates the blended region where the bulk motion of the refined grid is blended into the constant outer region of the computational domain. The third frame shows the inner refined region is moved as a rigid body according to the bulk motion velocity as defined by the plunging cycle. The fourth frame shows the offset at the top and bottom of the plunging cycle for the wing section.

Figure 27 - Grid Displacement for Top and Bottom of Cycle.
The bulk motion displacement was determined by the reduced frequency parameter at each timestep as the solution evolved, given by

\[ X_b = hc \cos(2\pi f t) \]

The displacement was divided by the timestep to give the bulk velocity over the timestep for each iteration. The bulk velocity was then applied to the refined region of the grid as a rigid body motion, while the bulk velocity was blended into the outer region of the grid based on a blending function used to transition the motion from the inner to the outer regions of the grid. Cells in the blended region were smoothed to prevent poor quality cells in the blended region due to increased skewness resulting from the bulk motion.

5.2 Results Comparison

Computational results using the bulk motion model described in the previous section are compared to experimental data and results of a similar computational solution using CFL3D\textsuperscript{55,56}. Results for the current computational solutions required selection of run parameters which would capture the results of the experiment and verify the accuracy of the calculations. To that end, the timestep was selected to produce 200 iterations per plunge cycle, with each of the measurement locations \( \phi = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4} \) occurring at 50, 100, 150, and 200 iterations, respectively. PIV data from the experiment was averaged at the measurement locations over 115 cycles of motion. It would be prohibitively expensive to produce that many samples using the current computational approach, therefore the current measurements were taken over the last 5 of the 10 total cycles executed. Data
from the experiment showed that startup transients were reduced by the second full cycle, or after 400 iterations in the computational results. Averaging computational data after 1000 iterations should be well outside the startup transients observed in the experiment, and repeatability of data after the first 5 cycles of motion confirms that assumption.

A base set of algorithmic parameters was selected for the initial time accurate results derived from previous work with time accurate solutions, and the number of Newton sub-iterations were increased to verify there were no changes in the results due to convergence of the sub-iterations. Using the reference temporal resolution of 200 iterations per cycle based on the reduced frequency implied a timestep of \(1.82452098 \times 10^{-5}\) s. Two laminar flow solutions were executed using 5 and 10 Newton sub-iterations, respectively. When examined it was found that no significant difference in forces or structure resulted from the number of Newton sub-iterations, therefore a value of 5 was selected for all computational solutions. There were 32 Gauss-Seidel sweeps performed for each solution, and the number of Gauss-Seidel sweeps was not examined as a parameter in the temporal analysis.

As an initial validation of the gridspeed terms, pressure data is extracted from the computational domain across the inner grid region defined by the bulk motion to the outer blended region where the grid is static. Pressure data extraction will verify there is no perturbation in the flow quantities as a result of the grid motion. Figure 28 shows two extraction lines with the corresponding pressure data for regions in front of and across the top of the wing section, respectively, at the centerline of the domain. Pressure data for each of the extraction lines shows no significant jump or perturbation of the pressure
values as the extraction lines cross from the bulk motion to the relative motion regions of
the grid. The pressure data shows a smooth transition from the refined to coarse regions
of the grid, indicating there is no feedback in the solution as a result of the grid motion.
This result is further confirmed by the lack of perturbations in the velocity and vorticity
plots in the relative motion region of the grid as show in subsequent figures.

![Graphs showing pressure data](image)

**Figure 28 - Vertical and Horizontal Plot of Pressure Across Motion Region.**

Figure 29 shows laminar flow computational results compared to PIV data produced
from the experiment. Overall qualitative agreement with the PIV data is good for all of
the measurement locations observed. Major structures in the flow resulting from the
body motion are captured by the computational results, although there are some differences in structure. Some differences are the result of comparing the continuous contour data from computational results to the discrete contours from the experiment.

Figure 29 – PIV Velocity Data (Left) Compared to Laminar Results (Right).
Figure 30 shows a comparison of the current computational results for laminar flow compared to laminar flow results obtained using CFL3D. CFL3D results were produced using a 2D structured grid consisting of quadrilateral elements with a freestream Mach number of 0.2, compared to a freestream Mach number of 0.12876 for the current results. Comparing data for the computational solutions again shows overall good agreement with the large structures in the flow such as the trailing edge vortex shed in the second frame of the comparison. There are apparent differences in the development of vortex regions near the leading edge of the airfoil as noted in the second and fourth frames. Although both solutions show vortex structures forming at the leading edge, the location, structure, and strength of the vortex development shows some differences between the two solutions. Overall agreement of the larger structures would indicate that these differences do not dominate the flow results, but can explain some of the differences observed between the two solutions. As with the experimental data, the CFL3D results were plotted using discrete contours, while the current results are plotted using continuous contours making a direct comparison more difficult.
Figure 30 – CFL3D Laminar Velocity (Left) Compared to Laminar Results (Right).

Figure 31 shows computational results using turbulent flow compared to PIV data. Turbulent results for the current computational solutions were obtained using the Spalart-
Allmaras one equation model with DES. No transition model was used in the turbulent results, so the flow was assumed to be turbulent throughout. Comparison of turbulent results to PIV data show similar comparisons to laminar results. Overall qualitative agreement is good and the major structures in the flow have been captured. There are regions which show differences from the experimental results, but these differences are comparable to differences observed with the laminar flow results. Comparing Figure 29 to Figure 31 shows that in fact the laminar and turbulent computational results show very few differences, except for some small differences in structure near the surface.

Viscosity ratio values showed that the viscosity ratio for the turbulent results was on the order of 1 or less, meaning that the turbulent viscosity was on the order of the molecular viscosity or less. This implies that turbulent viscosity is not dominating the flow, which is due to the low Reynolds number. For the standard Spalart-Allmaras model, the wall distance is taken as the distance of the cell center from the nearest surface point. The DES model replaces the wall distance with the minimum of the wall distance and the local characteristic cell length. The wall distance appears in the turbulence model equations as part of the dissipation term, as $1/d^2$. Therefore as the distance increases the term becomes smaller and is driven towards zero. In the DES model, the wall distance is replaced with the characteristic cell length away from the wall, which prevents the dissipation term from tending towards zero, resulting in more dissipation and less turbulent viscosity. Given the low Reynolds number of the current solutions and the amount dissipation present in the model, the turbulent viscosity remains low and dissipates quickly, resulting in solutions which are comparable to laminar solutions. A
comparison of the DES results to the base Spalart-Allmaras model is needed to validate
the maximum viscosity ratio, since the DES model should only affect viscosity ratios
outside the boundary layer. Such a comparison is outside the scope of the current work.

Figure 31 – PIV Velocity Data (Left) Compared to Spalart-Allmaras Results (Right).
Figure 32 – PIV Vorticity Data (Left) Compared to Laminar Vorticity (Right).

Figure 32 shows vorticity for the laminar computational results compared to PIV data. As with the velocity field data, results for vorticity are comparable to experimental data with major structures captured by the current computational results, but there are
differences in the smaller structures. Results at the trailing edge of the airfoil show good overall agreement with the experimental results, especially when comparing the trailing edge vortex shedding in the second and fourth frames. There is good overall agreement between the results for larger structures on the upper surface of the airfoil, but the current computational results shows a large number of smaller vortex structures not observed in the experiment. These smaller structures appear as noise in the computational solution and are believed to be the result of modeling issues with the surface grid used for the computational solutions.

Results for the plunging airfoil show overall good agreement between current computational results, experimental data, and CFL3D results. Major structures in the flowfield are captured, although some differences in smaller structures are observed. Most significantly, no perturbations due to grid motion are observed in the results, demonstrating the new form of the grid velocity term did not affect the time history of the solution. Some of the current computational solution differences may be explained by three dimensional effects resulting from the surface grid used in the computational model, which are addressed in the next section.

5.3 Three Dimensional and Modeling Effects

The current computational results showed good general agreement with experimental data for large structures in the flowfield, however there were noticeable differences in the smaller structures. It might be expected that the current computational solutions would produce consistent or improved comparison to the experimental data since the
computation results were developed using a three dimensional grid and three dimensional
effects were noted in the experiment\textsuperscript{53}. Computational results, however, showed a large
number of smaller structures which were not observed in the experiment and appear as
noise in the computational results.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image.png}
\caption{Comparison of 2D (Left) and 3D (Right) Vorticity Results.}
\end{figure}
Figure 33 shows a comparison of the 2D and 3D vorticity values for the laminar solution. Results for the 2D vorticity were obtained by excluding the velocity values normal to the symmetry plane of the solution. Results from the figure shows that the larger structures in the solution are comparable between the 2D and 3D values, but there is a significant difference in the smaller structures. Moreover these differences appear to be concentrated at the leading edge of the airfoil, with less effect noted near the trailing edge and in the shed vortices at the trailing edge. Differences in the extent of the larger structures indicate a 3D component to the larger structures as well, which also appear to be concentrated at the leading edge.

Figure 34 shows iso-surface values of vorticity for the average computational results at the $\phi = 1/2$ location in the plunge cycle for magnitudes of -36, -18, 18, and 36 respectively. The figure shows that the large shed vortex structures from the trailing edge are mainly 2D in nature, as are most of the structures close to the surface. At the leading edge, however, there are a large number of smaller 3D vortex structures which appear to emanate from the leading edge. These smaller 3D structures appear to develop and shed from the leading edge and progress downstream, and are dissipated as they move away from the airfoil surface due to the increase in grid spacing in the outer region. The grid spacing in the outer region is not sufficiently resolved to maintain the smaller vortex structures, which leads to the dissipation.
Figure 34 – Iso-Surface of Vorticity Magnitude.

Figure 35 shows a comparison of the 3D vortex values compared to the difference in the 2D and 3D values, obtained by subtracting the two results. Subtracting the 2D component of the vortex values from the 3D values gives an indication of the magnitude of the 3D component for each of the measurement locations. Comparing each frame of the figure shows there is no difference in the 2D and 3D trailing edge vortex shedding, although there are some smaller differences near the leading edge in the larger structures. The figure also indicates a significant portion of the smaller structures are a result of 3D effects. Results from out of plane vortex values, which are normal to the symmetry plane of the solution, give a further indication that the 3D effects noted in the computational solution are dominated by differences at the leading edge of the airfoil, and may lead to the differences in the smaller structures noted in the previous results.
Figure 35 - Comparison of 3D (Left) and Delta (3D-2D Right) Vorticity Results.

Comparing iso-surface results and differences in the 2D and 3D vortex values indicates that 3D effects in the computational solution result from flow differences at the leading edge of the airfoil. Figure 36 shows the grid and shaded discrete surface at the leading edge of the airfoil for the current computational results. Spanwise resolution of the leading edge appears to be reasonable, although chordwise resolution could be
improved to reduce the facetization. Facetization in the spanwise direction is the result of the grid generation process, and is the likely cause of differences in the solution noted at the leading edge. Improved results may be obtained by making the facetization in the spanwise direction at the leading edge consistent by using an extruded grid, where the symmetry plane is extruded in the spanwise direction. Extruding the symmetry plane would result in a consistent facetization pattern in the spanwise direction at the leading edge which may reduce or eliminate the 3D effects which appear to be inconsistent with the experimental data.

![Figure 36 - Leading Edge Grid and Spanwise Facetization.](image)

Although there are modeling differences which appear to have had some influence on the results, overall comparison to the experimental data is good, especially when comparing the large structures. Differences in the results do not appear to be the result of perturbations from the grid motion, indicating the new gridspeed terms have maintained time accuracy and removed any perturbations due to grid motion.
6 Conclusions

Completion of the current work represents development of a new definition of gridspeed for unstructured grids which is both conservative and accurate. Previous work on 2D unstructured grid r-refinement adaption for steady state flows was extended to time accurate solutions for 2D and 3D problems on unstructured grids.

The initial impetus for the current work was implementation of a structured grid time accurate adaption algorithm for unsteady flows on structured grids. A coupled approach was adopted where gridspeed terms were incorporated into the flux calculation to account for the grid displacement, and an unsteady residual was defined to resolve the volume change for each cell through the use of Newton sub-iterations at each timestep. A gridspeed definition, based on node displacement, was implemented in a 2D unstructured flow solver and used to solve a shock tube problem. Results for the shock tube demonstrated that flow solutions for moving grids were not time accurate using the node displacement model initially employed. It was discovered that the node displacement model only incorporated a single mode of motion, that of cell face translation. Although adequate for structured grid solutions used to develop the original method, it proved inadequate to describe unstructured grid motion where faces not only translate, but typically rotate and stretch as the grid is adapted. Study of a steady state flow, where the mesh was either relaxed or forced to deform to an artificial sink showed that the additional modes of motion present in unstructured grid movement produced perturbations in the flow proportional to the grid motion. Solution of transient problems
using an unstructured approach required a gridspeed definition that incorporated all modes of motion.

Perturbations in the flowfield due to grid motion resulted from a gridspeed definition that did not properly account for the volume displaced by the face as the grid deformed, violating geometric conservation. Development of the new gridspeed definition centered on description of the volume change resulting from grid motion based on a conservation law form of the motion equation. Taking the specialized case of grid motion in a quiescent flow, an equation for the gridspeed was developed which described the face velocity in terms of the volume displaced and the face area at the end of the motion step. Geometric conservation is enforced since the gridspeed definition was developed from a conservation form of the motion description based on the volume change. The new gridspeed equation did not place any restrictions on dimensionality; therefore the form of the equations is valid for two and three dimensional flowfields.

Validation of the gridspeed terms for 2D solutions focused on grid displacement in a box with zero velocity. An artificial sink was used to deform the grid and verify no perturbations to the flow quantities resulted from the grid motion. Time accurate validation of the method was accomplished using a 10:1 pressure ratio shock tube solution to compare against an exact shock tube solution developed from theory. Initial validation of the 3D solutions consisted of a transient grid deformation wave in the form of a moving weight function applied against a steady state flow through a channel. It was demonstrated that no perturbations were introduced to the steady state flow as a result of
grid motion. Time accurate validation of the 3D implementation was again performed using a shock tube.

A node restriction algorithm developed for 2D flows in the previous work proved sufficient to prevent grid inversion for the 2D case but was not suitable to the 3D case. Node restriction for the 3D case based on a geometric quality measure proved to be sufficient to prevent cross over, but the center of mass algorithm was not able to smooth the grid outside regions of weight function concentration. Smoothing using the center of mass calculation was shown to invert the grid in some cases depending on the local quality and connectivity. Three dimensional grid smoothing required implementation of an optimized based smoother which could address all potential types of poor cell quality in a tetrahedral mesh without inverting the grid. In addition to the Laplacian and optimized smoothers, implementation of an edge and face swap algorithm was used to further improve geometric grid quality outside large weight function gradients. Combined use of all these smoothing techniques, based on previous research, provided the means to recover the mesh after an adaption step, preventing mesh inversion and improving node distribution in the grid.

Using the time accurate r-refinement algorithm for unstructured grids developed herein, solutions for a 3D airfoil solution undergoing a sinusoidal plunging motion was used to further demonstrate time accuracy. Using a symmetry boundary condition on the sidewalls to simulate an infinite span wing, specified motion terms were applied to the grid region near the surface. The motion rates for each timestep were blended into the outer domain, allowing the surface and volume grid near the wing surface to move in a
sinusoidal motion at a prescribed reduced frequency and amplitude. Results for the plunging airfoil were compared favorably to experimental data and CFD results for large flow structures. Differences in smaller structures were noted, which are the result of spanwise facetization in the discrete surface grid used to produce the computational results. No perturbations to the flowfield variables as a result of the bulk grid motion were noted in any of the solutions.

Results from the validation and comparison cases demonstrates that the proposed gridspeed equation is time accurate and eliminates any perturbations to the flowfield quantities as a result of grid motion. Variables required for the new gridspeed equation are either known or easily calculated for a typical flow solver, and can be incorporated into the flux equations to eliminate any perturbations to the flowfield as a result of grid motion.
7 References


29 Karypis, G., Kumar, V., 1995, “METIS Unstructured Graph Partitioning and Sparse Matrix Ordering System Version 2.0,” University of Minnesota, Department of Computer Science, Minneapolis, MN 55455.


56 McGowan, G.Z., Personal Communication.

Appendix
A Three Dimensional Grid Smoothing

Extension of the $r$-refinement algorithm from two to three dimensional solutions necessitated the ability to deform and smooth tetrahedral elements. Unlike two dimensional grids constructed from triangulated faces, where Laplacian smoothers are sufficient to both increase resolution and perform grid smoothing in a single step, three dimensional tetrahedral elements require application of more sophisticated techniques to improve poor grid quality. A significant amount of development was required in the current work to both increase resolution through $r$-refinement and recover poor quality cells in the absence of large weight function values. The basis for most of the development, as well as the reference information contained in this appendix was derived from reference 6. In the following sections a definition of geometric grid quality is defined based on all cells incident on a node. A local mesh is defined which is composed of a central node and all cells incident on that node, and the quality associated with a node is defined in the context of the local mesh.

A.1 Poor Tetrahedral Quality and the Quality Function

The first step in quality improvement is determination of what constitutes a poor quality cell. Discussions of cell quality are cast almost exclusively in geometric terms, although research has shown that quality cannot be taken out of context with the underlying solution, as noted in the discussion on grid adaption. The purpose of this appendix is to outline how poor quality cells, in geometric terms, are identified in regions
where the weight function is small and where the adaption algorithm is needed to relax the grid to recover a smooth element volume transition. Smoothing in the current context is limited to discussions of tetrahedra and triangular elements, but it should be noted that quality based smoothing can be applied to any element type as long as a consistent measure of quality is maintained.

Quality of a tetrahedral element is most often defined in terms of the angles formed by the faces of the element, or dihedral angle. Mesh smoothers are designed to use some function of the dihedral angle as a quality measure which can be used to improve the element quality. The problem is particularly acute for tetrahedral elements since there are a number of configurations that can lead to either very large or very small dihedral angles. Edelsbrunner and Guoy\textsuperscript{57} have identified nine different tetrahedral element types that describe poor quality tetrahedra reproduced in Figure 37 and Figure 38.

![Figure 37 - Linear Forms of Poor Quality Tetrahedra.](image_url)
Each of the poor element types exhibits an exceedingly small or large dihedral angle or excessive skewness which results in poor cell quality. Numerical error issues in the flow solver arise when solutions are attempted on grids containing these types of poor quality elements, and perturbations in the solution or stability problems can arise. Although the dihedral angle identifies each of these types, a unified quality function based on dihedral angle may be difficult to construct since some cell types may benefit from a minimization of a large dihedral angle, while other may benefit from maximization of a small dihedral angle.

An alternate approach is to define quality in terms of the circumsphere containing the tetrahedral element or circumcircle containing a triangular element at a surface boundary. The circumsphere or circumcircle is described by the endpoints of the reference tetrahedra or sphere, where all of the points of the original element must lie on the surface of the circumscribing element. Figure 39 is an example of a circumcircle for two 2D triangular elements.
Figure 39 - Examples of Circumcircle of a 2D Triangle.

It is easily determined from inspection that the quality of the triangular element on the left of the figure is much higher than the quality of the triangular element on the right, represented by the solid lines. To quantify the element quality, the volume of the inscribed triangular element in 2D, or tetrahedral element in 3D is divided by the largest volume element that could be contained in a circumcircle or circumsphere, respectively, of equal size. Returning to Figure 39, it would be the area of the triangle represented by the solid line divided by the area of the triangle represented by the dashed line. The ratio of the two areas represents the relative quality of the cell, which can be used as a global measure of quality ranging from 0 (flat cell) to 1 (perfect isosceles). Quality of tetrahedral elements can be determined in a similar fashion.

The circumscribed definition of quality gives a quantitative metric that can be used to both smooth the grid and identify regions where the grid becomes inverted, since inversion results in a negative volume and hence a negative quality measure. References to quality in the subsequent sections refer to this definition of quality with regard to the
current implementation; however none of the subsequent sections are limited to this quality definition and can be applied equally to other quality metrics.

A.2 Laplacian Based Smoothing

Laplacian smoothers are the most common approach used to grid smoothing because they are easy to implement and are computationally inexpensive compared to other methods such as optimized smoothing. Laplacian smoothers have proved very successful at smoothing structured grids where there is constant connectivity of a node and all of its neighbors. Success in smoothing 2D unstructured grids has also been demonstrated for a numbers of cases, however they have been proven ineffective when applied to 3D unstructured grids. Failure on 3D unstructured grids is caused by the non-uniform connectivity of the grid coupled with an inability to identify all poor quality cell types in a consistent and unified manner. Weights for the Laplacian smoothers are typically constructed from the quality function, but the smoothing function itself is determined from an assumption of the motion required to improve quality and not tied to a direct measure of quality improvement. For example, consider the spire geometry from Figure 37 and the cap geometry from Figure 38. In the case of the spire, quality improvement is obtained by moving the apex of the spire towards the base of the geometry, while in the case of the cap quality improvement is achieved by moving the apex away from the base of the geometry. Each poor quality element is improved by the opposite motion that improves the quality of the corresponding element in the example. Since the element motion is disconnected from a measure of the quality change, there is no motion
algorithm available which ensures improved element quality. Given a further complication that elements of contrasting types may be incident on the same node renders Laplacian based smoothers unable to identify and correct elements of all quality type.

The examples above, however, represent the most extreme cases of poor element quality, and in most cases the majority of the poor element quality can be improved through the use of a Laplacian smoother. The relatively low cost of Laplacian smoothers makes them a viable alternative to more expensive optimized smoothers, and are used as a first pass to improve element quality before applying more expensive smoothing options. Issues of crossover, which are possible with a Laplacian smoother, are addressed by use of a constrained Laplacian. In the constrained case, the quality of all the cells incident on a node are calculated before and after the movement operation, and the movement is allowed only if there is an improvement in the minimum quality. If the minimum quality is degraded, then the movement is reversed and the poor quality elements are left for the optimized based smoothing. Basing the movement on the minimum quality does not ensure that the average quality is improved, so the overall mean quality of the grid can be reduced, but the minimum quality in the grid should only improve.

A.3 Optimization Based Smoothers

Grid cells with poor quality which cannot be corrected through use of a Laplacian smoother are passed to an optimization based smoother. Optimization based smoothers are more successful at correcting poor quality cells because the motion they induce is
determined from the quality metric and its derivatives, moving the cell in the direction of improved minimum quality. Calculations necessary to compute the movement direction and distance are very expensive, and are therefore applied after the Laplacian based smoothing and only to cells below a minimum quality threshold.

The optimized smoother implemented in the current work is based on a steepest descent method using derivatives of the objective function to determine the descent direction. The objective function in the current work is defined by the quality metric based on the previous circumsphere definition of quality for a tetrahedral element. Numerical derivatives of the quality function for each coordinate direction are computed to determine the descent direction, and an active set of directions is created and used to perform the motion. An initial step size is determined from the gradient information, and a new measure of the quality is computed to determine if step was successful in improving the quality and to ensure the quality did not degrade as well as to verify that the cell was not inverted. In cases where the cell quality is not improved, the step size is reduced and the quality function is recomputed for the new movement. Successive iterations of the movement are applied until either a specified tolerance or iteration count is reached, or until the local minimum is reached. Local cell quality either improves or remains constant at the previous minimum value.

Calculation of derivatives and the iterative nature of the optimization method make the approach very computationally expensive; however the method is successful at improving the quality or the poorest quality cells in a grid. To balance the expense of the method with the potential benefits, the minimum quality of cells where the optimized
based smoother is applied is limited. In the current work, optimized smoothing is limited to cells with quality below 0.10 or 0.05 to prevent excessive overhead.

While optimized smoothers are generally successful at improving poor cell quality, they are not successful if the poor quality is a strong function of the local cell connectivity. Optimized smoothers do not adjust the local cell connectivity and are constrained by the connectivity of the local mesh. Improvement in the efficacy of the optimized smoother can be attained if it is coupled with a face and edge swap algorithm to improve connectivity.
A.4 Face and Edge Swap

Face and edge swap operations are the only smoothing operations which can improve the local node connectivity, and are defined as a p-refinement operation. Issues with cell connectivity are identified more with h-refinement schemes than with r-refinement schemes since it is assumed that the starting grid for an r-refinement approach has been smoothed prior to the solution and all possible improvements in connectivity have been made. It is important in the context of the current work, however, since p-refinement increases the number of faces and cells in the grid and one goal of the current work is development of a method which describes a conservative, time accurate approach for r-refinement, but can also be used to remove perturbations associated with smoothing operations after an enrichment adaption cycle.

There are two basic forms of swapping defined respectively by a face swap and an edge swap. A form of face swap for a boundary element is shown in Figure 40 where the edge of the bottom two triangular elements has been swapped from one node pair to another. Face swapping inside the grid domain generally reduces to a form of edge swapping and is not treated as a separate operation outside the boundary in the current work.

Edge swapping is considerably more difficult to implement since it cannot be generalized to all possible edge constructs in the grid. In a general 3D tetrahedral grid around complex geometry, the count of cells incident on a face can vary from less than approximately 8 for simple geometries to over 20 where there are large changes in cell
distribution due to enrichment. Edge swap configurations become increasingly difficult as the number of cells incident on an edge increases, therefore edge swapping is limited in the current work to edges with 7 or less cells incident on an edge.

![Figure 40 - Face Swap Example.](image)

Implementation of an edge swap routine based on a brute force approach would require identification of 63 distinct types of cell face geometry for all possible combinations of edge swap where there are between 4 and 7 cells incident on an edge. A more elegant approach is possible if, for each of the counts of cells incident on an edge, a canonical configuration were defined. Figure 41, Figure 42, and Figure 43 show the canonical forms for edges with 4, 5, 6, and 7 incident cells respectively. Each canonical form represents a potential configuration for the plane intersecting the upper and lower grouping of cells formed when the vertical edge is swapped to the horizontal plane defined by the figures. The number below each figure represents the number of permutations possible for each canonical form, so that for each form there are a number of possible permutations which could improve cell quality. Iteration through the cell swap routine, using the canonical form as a reference, will identify the number of cells
incident on the edge, and check every permutation of each canonical form to determine
the configuration that yields the largest improvement in cell quality. The configuration
that generates the largest quality improvement is used to reconstruct the grid locally
conforming to the new topology. So an edge with 7 incident cells will check 42 possible
combinations to determine if the minimum quality can be improved, representing 7
permutations for each of the 6 canonical forms identified.

![Canonical Form for 4 and 5 Cell Configurations.](image)

Figure 41 - Canonical Form for 4 and 5 Cell Configurations.

![Canonical Form for 6 Cell Configurations.](image)

Figure 42 - Canonical Form for 6 Cell Configurations.
Figure 43 - Canonical Form for 7 Cell Configurations.

It is easy to see from the figures that the workload increases significantly as the number of cells incident on an edge increases. Based on observations in swap counts observed for complex grids, the percentage improvement in edges with 7 or more incident cells is very small and in practice it would likely not be beneficial to apply edge swapping to configurations with more than 7 incident cells.

Performance improvement in the Laplacian and optimized smoother has been noted when combined with face and edge swapping in the context of enrichment adaption. Face and edge swapping form the last method implemented in the current smoothing algorithm.

A.5 Zonal Boundary Treatment

Parallel implementation requires development of algorithms capable of collecting and processing information across zonal boundaries based on the local grid topology and distributed across processors. Zonal boundary treatment can have a significant impact on performance if the data structures and algorithms used to process inter-zonal data are not efficient. The most common approach to smoothing across zonal boundaries is to treat zonal boundary nodes and edges separately from domain nodes and edges. Nodes and
edges that lie inside a processor domain are smoothed first, followed by smoothing of the zonal boundary nodes and edges. The current implementation utilizes the separation approach applied to node operations, and freezes edge swapping at zonal boundary interfaces.

Laplacian and optimized smoothing are node based approaches where quality improvement is the result node relocation based on the quality of the local mesh. Node operations become increasingly difficult in parallel implementations of cell center based flow solvers, as in the current case, due to the fact that parallel boundaries are defined at cell face intersections, which results in nodes located on zonal boundaries. Faces are easier to manage since a face can lie on at most two processors, and defines the processor intersection. Nodes, on the other hand, can lie on multiple processors and do not contain a link between all the processors they touch. Therefore node operations require global collection and distribution operations to manage changes, whereas face operations require processor to processor communication. The result is increased overhead when dealing with node based operations on zonal boundaries. To address these issues, the current approach is to create a node based inter-zonal mesh which is a subset of the global mesh. The inter-zonal mesh contains all the nodes which lie on a processor boundary and all cells incident on those nodes. To improve parallel efficiency, each node on a zonal boundary is assigned to a processor, called a primary processor for that node, and all cell data associated with the node is communicated to the primary processor. Currently the lowest rank processor containing a node is assigned as the primary processor, which is not well balanced but is an improvement in performance. Assignment of a primary
processor for each inter-zonal node reduces the problem from a global collection and
distribution method to a processor to processor method. The result is an inter-zonal mesh
consisting of cells which are incident on a processor boundary, distributed across the
primary processor mapping for each node.

Once the inter-zonal mesh has been defined, Laplacian and optimized smooth
operations are applied to the zonal mesh, and updates to node locations are communicated
to each processor that contains one of the zonal nodes. A hierarchical structure is
required to maintain consistency in the parallel environment. Nodes that lie on a seam
between more than two processors have to be moved first since the position of a seam
node can affect the smoothing operation on incident nodes located on processors other
than the primary processor for the seam node.

Face and edge swapping present a more difficult problem. Swapping an edge node on
a processor boundary would require redefining the processor boundary since swapping
alters the face structure of the cells incident on the edge, and processor boundaries are
defined in terms of face intersections. The current work centers on r-refinement adaption,
so that the local connectivity of the grid is never altered as the grid is adapted. Constant
connectivity is one obvious advantage of r-refinement algorithms, since it allows
simplifications to the adaption and smoothing process. In cases where swapping is
necessary, such as enrichment schemes, the current approach is to force cells off of a
zonal boundary during the rebalance step where the parallel problem is redistributed after
enrichment, which is a requirement for h-refinement and not r-refinement.
A.6 Best Practices

Freitag and Ollivier-Gooch\textsuperscript{6} have defined a best practices approach to grid quality enhancement which has been generally implemented in the current work. The current implementation defines a generalized smoothing operation which consists of each of the above outlined operations and an additional edge collapse routine. A separate minimum quality value is specified for each method, where only cells below the minimum quality threshold for the corresponding method are selected for smoothing. An iteration count for each method is also used to control the number of smoothing passes for each method, and an overall smoothing iteration count is defined to determine how many passes of the combined smoothing algorithm are used in each adaption cycle.

The smoothing operation begins by performing face and edge swap for cells below the corresponding swap criteria. Swap routines can improve the local cell connectivity and result in improvement in performance of the Laplacian and optimized based smoothers. Next the Laplacian smoother is executed to attempt to improve the quality of most of the poor quality elements. A constrained Laplacian operator described earlier is used to ensure quality is only improved and not degraded. After application of the Laplacian smoother, the optimized based smoother is executed to improve the worst quality elements in the grid. The minimum quality threshold for the optimized smoother is generally very small, well below the Laplacian threshold, to reduce overhead. At planar boundaries the grid is reflected across the surface normal to form a 3D local mesh version of the 2D plane, and smoothed using the optimized based smoother if the quality is below
the threshold. Smoothing is generally limited to regions outside the prism mesh and hence the boundary layer, but planar items such as far field boxes and symmetry planes are often in proximity to the regions being smoothed. A reflected grid is a simple way to smooth a surface mesh in the special case of a planar boundary. Non-planar boundary smoothing would require interface to a boundary surface model such as a CAD package or NURBS surfaces, and is beyond the scope of the current implementation.

The last operation in a smoothing cycle is the collapse element routine. Element collapse is attempted on only the poorest quality elements since over application of collapse can degrade resolution unintentionally if care is not taken. Threshold values for element collapse are typically on the order of values used for the optimized smoother.

Application of the above represents one pass through the combine smoothing routine. Three to five passes per adaption cycle are currently used during each adaption cycle to smooth the grid. Although not presented in the current work, the flow solver contains an enrichment algorithm to locally refine the grid based on the weight function. Enrichment tends to cause greater degradation of the grid quality since it is more difficult to control skewness, and changes to the local grid connectivity are now possible. The number of smoothing passes required after enrichment is generally larger than the number of passes required after an r-refinement step where adaption is limited to motion. Smoothing after a r-refinement step can typically be accomplished with a single smoothing iteration.