PARALLEL ANISOTROPIC BLOCK-BASED ADAPTIVE MESH
REFINEMENT ALGORITHM FOR THREE-DIMENSIONAL FLOWS

by

Michael Williamschen

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

Copyright © 2013 by Michael Williamschen

Abstract

Parallel Anisotropic Block-Based Adaptive Mesh Refinement Algorithm For
Three-Dimensional Flows

Michael Williamschen

Master of Applied Science

Graduate Department of Aerospace Engineering

University of Toronto

2013

A three-dimensional, parallel, anisotropic, block-based, adaptive mesh refinement (AMR) algorithm is proposed and described for the solution of fluid flows on body-fitted, multi-block, hexahedral meshes. Refinement and de-refinement in any grid block computational direction, or combination of directions, allows the mesh to rapidly adapt to anisotropic flow features such as shocks, boundary layers, or flame fronts, common to complex flow physics. Anisotropic refinements and an efficient and highly scalable parallel implementation lead to a potential for significant reduction in computational cost as compared to a more typical isotropic approach. Unstructured root-block topology allows for greater flexibility in the treatment of complex geometries. The AMR algorithm is coupled with an upwind finite-volume scheme for the solution of the Euler equations governing inviscid, compressible, gaseous flow. Steady-state and time-varying, three-dimensional, flow problems are investigated for various geometries, including the cubed-sphere mesh.

Acknowledgements

I would like to thank Professor Clinton P. T. Groth for his support and guidance. Our enthusiastic discussions helped motivate me throughout my Master's research. In addition, he gave me the freedom to pursue my own research direction.

I also would like to thank my friends and family for their support during my studies. I especially want to thank my parents for their never-ending encouragement. The UTIAS community deserves strong thanks as well. I do not think I could have had a more enjoyable Master's education anywhere else.

Lastly, all computations in this thesis made use of the computational resources provided by the SciNet High Performance Computing Consortium at the University of Toronto and Compute/Calcul Canada through funding from the Canada Foundation for Innovation (CFI) and the Province of Ontario, Canada. Financial support for the research described in this thesis was provided through a Space Science and Technology Cluster Pilots Program Grant from the Canadian Space Agency and from the MITACS (Mathematics of Information Technology and Complex Systems) Network. The latter part of the Networks of Centres of Excellence (NCE) program funded by the Canadian government. I wish to thank these aforementioned computational and funding resources for providing support for my research.

Michael Williamschen
Toronto, September 2013

Contents

1	Introduction	1
1.1	Solution Adaptive Schemes	2
1.2	Anisotropic AMR Techniques	7
1.3	Motivation	7
1.4	Research Objective and Scope	8
1.4.1	Outline of Thesis	9
2	Governing Equations and the Numerical Solution Scheme	10
2.1	Three-Dimensional Euler Equations	10
2.2	Upwind Finite Volume Method	11
2.2.1	Semi-Discrete Form of the Euler Equations	12
2.2.2	Numerical Flux Evaluation	13
2.2.3	Higher Resolution Finite Volume Scheme	16
2.2.4	Time Marching Schemes	18
2.3	Parallel Implementation	18
3	Three-Dimensional Anisotropic Block-Based AMR	20
3.1	Block-Based AMR	20
3.1.1	Anisotropic AMR Procedure	23
3.2	Block Connectivity	25
3.2.1	Binary Tree Hierarchical Data Structure	26
3.2.2	Computation of Connectivity	27
3.2.3	Connectivity Rearrangement	29
3.3	Refinement Flagging and Conflict Checking	32
3.3.1	Refinement Criteria	32
3.3.2	Conflict Checking	32

3.4	Anisotropic Mesh Refinement and Coarsening	35
3.4.1	Refinement	36
3.4.2	Coarsening	39
3.5	Inter-Block Solution Transfer	39
3.5.1	Ghost-Cell Solution Information	39
3.5.2	Interface Flux Corrections	42
4	Numerical Results	45
4.1	Unsteady Shock-Cube Initial Value Problem	45
4.2	Steady Supersonic Channel Flow Over a Bump	48
4.3	Steady Supersonic Radial Outflow	51
4.3.1	Accuracy Assessment	53
4.4	Steady Supersonic Flow Past a Sphere	55
5	Conclusions	61
5.1	Main Contributions	62
5.2	Future Work	63
	Bibliography	64

List of Tables

2.1	Coefficients and optimal CFL numbers for the multi-stage optimally smoothing scheme for a second-order spatial discretization.	18
-----	--	----

List of Figures

1.1	Adapting a mesh to a solution feature using mesh movement with 25 cells before and after adaptation.	3
1.2	Adapting a mesh to a solution feature using mesh refinement with 25 cells before refinement and 48 cells after refinement.	3
1.3	Illustration of cell-based, patch-based, and block-based AMR applied to an arbitrary discontinuity.	4
2.1	Hexahedral cell at grid location i, j, k showing face normals.	13
2.2	HLLC wave pattern showing the integration path.	16
2.3	Unlimited piecewise-linear reconstruction in one dimension superimposed over a piecewise-constant solution representation. Notice the solution overshoot at $i + \frac{1}{2}$	17
3.1	Three-dimensional hexahedral cell and example mesh.	21
3.2	Refined blocks with overlapping ghost cells.	22
3.3	The “deadlock” arising as a consequence of restricting the block coarsening to one direction per application of AMR	24
3.4	Multi-block mesh with unstructured block connectivity. Grid block boundaries are outlined in black lines and the orientations are shown with arrows.	25
3.5	3D binary tree and the corresponding blocks after several refinements	26
3.6	An example southeast edge neighbor search from the block shown in green to the block shown in red. Paths B1 and B2 are the bridges.	28
3.7	Left: blocks A and B need to coarsen in the ξ direction but do not share a common parent. Right: binary trees showing the connectivity rearrangement required to force A and B to share a common parent.	30

3.8	Blocks A and B need to rearrange connectivity but their parents are at different levels, causing the rearrangement to fail. After an intermediate rearrangement, the connectivity of blocks A and B can be rearranged. . .	31
3.9	Example conflicts resolved through first and second sibling checks.	34
3.10	Example refinement level conflict resolved with level checks.	34
3.11	Sequence of anisotropic mesh adaptations that create a mesh which cannot be represented by the 3D binary tree data structure.	35
3.12	All possible single-step anisotropic refinements.	36
3.13	Anisotropic refinement of a cell using the grid metrics approach in the ξ computational direction. Fine node locations are indicated with small dots and coarse node locations with large dots.	38
3.14	Several message passing scenarios shown for anisotropic AMR.	41
3.15	Proposed message passing procedure to handle anisotropic resolution changes more generally. Sent cells are shown in green, and the reconstruction stencil for the blue cell is shown in red.	42
3.16	Flux correction procedure for resolution changes in two directions across the block interfaces. The dotted line shows the corresponding cell faces used in the correction.	43
4.1	Comparison between anisotropic and isotropic AMR for $t = 0.05$ ms, $t = 0.25$ ms, $t = 0.5$ ms, and $t = 0.75$ ms. The contour plots show the gas density in kg/m ³ and the blocks are outlined with solid lines.	47
4.2	Supersonic, $M = 1.4$ flow over a smooth bump in a channel at refinement levels one through three for 3D and 2D anisotropic AMR. Predicted solution density contours in kg/m ³ and block outlines are shown.	49
4.3	Supersonic, $M = 1.4$ flow over a smooth bump in a channel at refinement levels four through six for 3D and 2D anisotropic AMR. Predicted solution density contours in kg/m ³ and block outlines are shown.	50
4.4	Density along the line $y = 0.25$ m at refinement levels one through six. .	51
4.5	Quarter slices of the predicted solution showing density in kg/m ³ and block outlines for anisotropic and isotropic AMR.	52
4.6	Quarter slices of the predicted solution showing Mach number and block outlines for anisotropic and isotropic AMR.	53

4.7	Accuracy assessment based on the exact solution density of the supersonic radial outflow problem comparing anisotropic AMR, isotropic AMR, and the solution on a uniformly refined mesh.	54
4.8	Radial density profile comparison between 3D anisotropic and isotropic AMR for refinement levels one through four.	55
4.9	The initial mesh for the steady-state supersonic sphere problem with one of the five sectors removed for clarity.	56
4.10	Slices of $M = 2.0$ air flow over a sphere solution density contours. Isotropic adapted solutions shown in the left column, anisotropic shown in the right column. Meshes are shown from a refinement level of one to five.	58
4.11	Slice of $M = 2.0$ air flow over a sphere solution contours. Isotropic adapted solution density and Mach number shown at a refinement level of six with 26,753 blocks on the left. Anisotropic adapted solution density and Mach number shown at a refinement level of six with 3,789 blocks on the right.	59
4.12	Slice of the supersonic sphere mesh in the axial direction at $x = -0.2$ m showing blocks outlined in black.	60
4.13	Comparison between 3D anisotropic and isotropic AMR of center line density and Mach number from the stagnation point to the free-stream. The center line density for anisotropic AMR at each refinement level is also shown.	60

Chapter 1

Introduction

The use of computational fluid dynamics (CFD) tools has become ubiquitous in science and engineering fields with an ever increasing array of applications. These tools are allowing researchers and engineers to obtain greater insight into the driving physics of their problems, allowing detailed quantitative results or fast design iterations, without performing numerous costly or even fully impractical experiments. Furthermore, advances in computing power and algorithm design has shaped the progression of CFD into an accurate and efficient analysis tool.

High performance computing systems, in the form of massively parallel clusters with thousands of processors, are now commonly available to researchers. Furthermore, these systems have achieved peta-scale performance and are moving towards exa-scale performance in the near future. While the increasing computational power available to researchers has allowed the solution of fluid-physics problems faster and to a higher degree of accuracy, the desire to increase the size of the problems or the included physics, has grown accordingly. Moreover, complex, highly-coupled, multi-physics solutions are sought, that remain far more computationally complex to solve than the available computing resources into the foreseeable future. Many of the complex problems of interest to the scientific and engineering communities involve: *(i)* turbulent flows around complex geometries; *(ii)* chemically reacting flows, especially combustion; *(iii)* acoustically driven flows, *(iv)* electrically conductive flows; and *(v)* micro-scale, non-equilibrium flows. These complex, three-dimensional (3D), flows are highly computationally expensive to solve due to numerical stiffness of the governing non-linear, partial differential equations (PDEs), associated with disparate spatial and temporal scales. In addition, sub-physics equations and/or mathematical models must typically be solved in conjunction with the

governing equations and often, more than one of these complex features can be combined in a single simulation.

For these reasons, focusing on the development of novel CFD algorithms to more efficiently solve the equations governing physically-complex flows is necessary. One way to reduce the computational cost of solving these complex flow problems is to use adaptive mesh refinement (AMR) [1–7] methods that scale the local spatial discretization size only to the level necessary to resolve the fundamental physical features, rather than treating the entire problem in a uniform fashion. Additionally, these complex flow problems often exhibit anisotropic physical features such as shock waves, flame fronts, or shear layers, that have large variations only in particular spatial directions. Such solution structures can be exploited to further reduce the computational cost in an adaptive scheme. Finally, considering the recent advances in high performance computing in the form of large scale computing clusters with thousands of processors, it is imperative that emerging CFD algorithms are designed to efficiently utilize these resources. Therefore, the potential to significantly reduce the computational cost of solving flow problems involving complex physics by the coupling of a solution-adaptive scheme, particularly with anisotropic mesh adaptation, with an efficient parallel solution framework, forms the primary motivation for this research.

1.1 Solution Adaptive Schemes

As mentioned previously, solution-adaptive meshing can be used to reduce the computational complexity of solving complex flow problems by increasing the amount of local spatial discretization only to the extent necessary to resolve the primary features of the solution. Mesh adapting schemes alter the discretization of the computational domain in areas of the domain that do not sufficiently resolve the solution or that over resolve the solution. Existing solution-adaptive mesh techniques can be categorized into two broad classes: mesh movement, and mesh refinement techniques. With a mesh movement technique, the total number of computational elements remains constant, and the physical location of the elements change when the mesh is regenerated. Many of the common mesh movement techniques are discussed in [8]. Mesh movement is illustrated in Figure 1.1, where the mesh is adapted to the circular discontinuity, and the number of elements remains constant. In contrast, mesh refinement techniques alter the number of elements,

but the locations of the initial grid lines remain unchanged. Figure 1.2 shows adaptive mesh refinement on a Cartesian initial mesh.

Figure 1.1: Adapting a mesh to a solution feature using mesh movement with 25 cells before and after adaptation.

Figure 1.2: Adapting a mesh to a solution feature using mesh refinement with 25 cells before refinement and 48 cells after refinement.

Standard adaptive mesh refinement techniques involving cell enrichment have a disadvantage over mesh movement techniques in that the computational complexity increases when more computational elements are added. However, the total number of elements or cells does not need to be known *a priori* and maximum refinement levels can be defined to control the adapted mesh size. Furthermore, mesh movement techniques can suffer from poor mesh quality due to over-stretching of the computational elements to adapt to the changing solution. Therefore, only adaptive mesh refinement techniques involving cell addition will be considered here.

There are three main types of adaptive mesh refinement techniques: cell-based, patch-based, and block-based, all of which can be visualized in Figure 1.3 for AMR on a two-dimensional, Cartesian mesh. Cell-based AMR methods selectively refine individual computational cells and typically results in the fewest number of refined cells. These refined cells are created as perfectly overlapping of their parent, or coarse cell. Storage of connectivity is typically maintained using a hierarchical quadtree (two space dimensions) or octree (three space dimensions) data structure, where for each refined coarse

(a) Solution showing cells flagged to refine.

(b) Example of cell-based mesh refinement.

(c) Example of patch-based mesh refinement.

(d) Example of block-based mesh refinement.

Figure 1.3: Illustration of cell-based, patch-based, and block-based AMR applied to an arbitrary discontinuity.

cell, there are four or eight new fine cells created, respectively. Powell, De Zeeuw, and coworkers [4, 5, 9] developed cell-based AMR methods for a Cartesian initial mesh, as well as Berger [3] and Aftosmis [10, 11], for an efficient parallelized cut-cell approach for handling of complex geometries. Ham *et al.* [12] proposed a two-dimensional Cartesian cell-based anisotropic AMR algorithm with unstructured connectivity allowing transient

mesh adaptation. Iaccarino and Ham proposed a fully-unstructured, anisotropic approach to cell-based AMR [13, 14] and Wang *et al.* [15, 16] proposed a Cartesian 2^n tree, projected boundary layer approach for viscous computations. Typically, cut cell-based AMR methods suffer from varied cell-quality in near-geometry regions where high-quality cells are desired for viscous computations. Moreover, the cut-cell process may have more computational complexity than other approaches as many intersections with geometry must be computed. In addition, storage of element connectivity generally requires larger data structures and computation of connectivity, exchanging solution information, and parallelization is generally more complex than other AMR methods.

Patch-based AMR creates regular patches of cells that overlap areas of the domain that require additional resolution. An initial patch-based approach was proposed by Berger and Oliger [1], where arbitrarily oriented, rectangular patches of cells overlap coarser regions. Data-structures for the dynamic, patch-based approach were developed by Berger [17]. Berger and Colella [2] further improved this technique by restricting to rectangular patches of cells that conform to the original coarse mesh and allow unsteady simulations on a supercomputer.

Block-based AMR combines elements of both cell-based and patch-based AMR into a relatively simple and efficient algorithm. Contrary to individual cells being refined, block-based AMR refines full blocks, each containing a number of cells. Much like cell-based AMR, blocks are typically refined to produce eight children blocks, for isotropic AMR in three dimensions, and the refinement history is stored within a tree-based data structure. While block-based AMR can sometimes locally over-refine, as shown in Figure 1.3, this negative aspect is outweighed by the efficient use of computational resources when communicating solution information between neighboring computational elements, refining, updating connectivity, and tracking refinement history, for blocks, not for individual cells. In addition, some block-based AMR methods can use body-fitted, curvilinear, grid blocks, allowing accurate representation of the geometry and more preferable flow-aligned cells.

The block-based approach for AMR on a Cartesian mesh was developed by Quirk [6] and Berger [18]. The approaches by Quirk and Berger show some similarities to the block-based approach developed by Groth and coworkers in two and three dimensions [7, 19–24] for magnetohydrodynamics (MHD) simulations using a high-performance, flexible, data structure. This flexible, Cartesian, block-based approach has been extended for application to two-dimensional body-fitted, curvilinear blocks by Sachdev, Northrup,

and Groth [25,26] and then further to three dimensional, body-fitted, blocks by Gao and Groth [27,28]. Other extensions have included high-order central-ENO (CENO) solution reconstruction by Ivan and Groth [29–31], a parallel implicit Newton-Krylov solver by Northrup and Groth [26], and anisotropic block-based AMR by Zhang and Groth [32,33] in two dimensions.

The parallel, block-based adaptive mesh refinement of Groth *et al.* has been successfully applied to the solution of a wide range of flow problems including laminar flames [34,35], turbulent non-premixed flames [27,35,36], and micron-scale flows [37]. Other applications of block-based AMR include soot prediction [38], radiative heat transfer [39], multi-phase turbulent rocket core flows [25,40], and the aforementioned MHD simulations [7,20–22,41].

An overset, or Chimera mesh [42] can also be used with AMR, where near geometry regions use high-quality, body-fitted, meshes which are then “cut” out of a Cartesian volume mesh and solution information is interpolated across the interface of the two regions. This approach was used with AMR by Boden and Toro [43], Brislawn [44], Chesshire [45], and Henshaw [46]. These approaches can be very effective at treating problems with complex or even moving geometries in an efficient and relatively automatic manner. Disadvantages of the above overset approaches are the necessity to recreate the entire mesh and compute connectivity after refinement, as well as difficulty interpolating the solution across overlapping interfaces. Furthermore, conservation is not necessarily well-enforced due to the non-conforming, overlapping regions.

Lastly, there exists some motivation for hybrid methods to allow for greater flexibility, although their use is less widespread. One example is the hybrid block-based method by van der Holst and Keppens [47], where a combination of block-based and patch-based AMR is used. Instead of creating four or eight fine blocks for every refinement, as in the case of isotropic refinement in two and three dimensions, incomplete block families can be created with less than the four or eight fine blocks. Although this approach seems to offer greater flexibility than the patch or block-based approaches, connectivity requires a doubly linked list of grid nodes per refinement level in addition to the block connectivity. Also, the incomplete block families complicates the neighbor computation.

A hybrid structured and unstructured, block-based AMR method has been developed by Zheng and Groth [48] for the two-dimensional form of the Euler equations. The main advantage of such an approach is the ability to use high-quality, anisotropic-stretched,

body-fitted, structured blocks in the near-geometry regions for accuracy and efficiency, and then use unstructured elements for efficient space-filling of the volume regions.

1.2 Anisotropic AMR Techniques

Previous studies of anisotropic adaptive mesh refinement techniques based on data structures that store complete hierarchies of refinements are somewhat scarce. As previously mentioned, a two-dimensional, block-based approach to anisotropic AMR using hexahedral computational cells has been proposed by Zhang and Groth [32, 33] based on a binary tree hierarchical data structure. A cell-based anisotropic AMR algorithm for a Cartesian mesh with a projected boundary layer technique was proposed by Wang *et al.* [15, 16] that uses a 2^n [15, 16] tree data structure, permitting both anisotropic and isotropic refinements.

Other cell-based anisotropic AMR approaches have been proposed that rely on fully unstructured techniques and do not use hierarchical data structures. Ham *et al.* [12] proposed a two-dimensional Cartesian cell-based anisotropic AMR algorithm for transient refinement. The algorithm uses fully unstructured connectivity and permits cell coarsening in a direction other than the previous refinement. Iaccarino and Ham proposed a fully-unstructured, three-dimensional anisotropic approach to cell-based AMR [13, 14] using an immersed boundary approach.

While cell-based, three-dimensional, anisotropic AMR algorithms have been developed, to the best of the author's knowledge there are no previous studies pertaining to the development of three-dimensional, block-based, anisotropic AMR algorithms. However, the advantages of a block-based approach to AMR as outlined above, would be beneficial. Therefore a three-dimensional, block-based, anisotropic AMR algorithm is considered in this thesis rather than following what has been developed for cell-based anisotropic AMR.

1.3 Motivation

This work will therefore focus on the use of a block-based approach to AMR, as defined above, as it has been applied successfully to the simulation of complex, multi-scale, flow problems in an efficient and flexible manner, while utilizing high performance

computational resources more efficiently than other approaches. The addition of three-dimensional anisotropic refinement capabilities to further reduce computational costs, is a natural evolution of the block-based approach and has already been shown to offer significant advantages in two space dimensions [32,33]. For these reasons, a block-based AMR approach in combination with an anisotropic refinement strategy is directly in-line with the motivation for AMR outlined in the beginning of this chapter.

1.4 Research Objective and Scope

With the above motivation for a fully three-dimensional, anisotropic, parallel, block-based approach to AMR in mind, the objective and scope for this thesis can be defined. Since this is an initial investigation into the capabilities of a 3D, block-based, anisotropic AMR algorithm, this work is limited to the solution of the Euler equations governing inviscid compressible gaseous flows with flow fields with relatively simplistic geometries, but developed in a generalized way, such that the extension to more complex cases will be supported. Both steady-state and unsteady simulations are supported by the proposed scheme. The previous studies and framework of two-dimensional (2D), block-based, anisotropic AMR [32,33] supplies the foundation for the development of the three-dimensional algorithm. However, development of a robust and efficient algorithm has required significant deviations from both the 2D anisotropic AMR and the 3D isotropic AMR schemes.

For accurate treatment of complex geometries and high quality flow-aligned cells, body-fitted grid blocks with hexahedral cells are used as part of the multi-block mesh approach, while allowing full unstructured root block connectivity. In the proposed anisotropic AMR scheme, a flexible binary-tree data structure is employed, permitting solution-directed refining and coarsening in each of the three computational directions, or combinations of directions, of the grid blocks to rapidly adapt the mesh to evolving flow features. In addition, coarsening of blocks is allowed in any direction irrespective of the refinement history. The data structure includes an efficient neighbor searching and connectivity rearrangement algorithm, similar in design to those of 2D anisotropic AMR [32,33], but with extensive changes required to handle the additional complexity of a third refinement and coarsening direction. Anisotropic refinement or coarsening is carried out in a single step, rather than the multi-step process of the aforementioned 2D scheme of Zhang and Groth, removing the associated loss in accuracy, and generality, of

such an approach. The resulting algorithm is efficiently parallelized to take full advantage of today's high-performance computing resources.

These contributions lead to a more efficient way to simulate complex, three-dimensional flows that include anisotropic features, than an identical simulation with isotropic AMR. Furthermore, the flexibility of the proposed scheme allows 3D anisotropic AMR to retain the same capabilities, including refinements, as isotropic AMR. This scheme is applied to several flow problems and meshes, including the cubed-sphere mesh [41, 49, 50], to assess the performance and accuracy of the developed scheme.

1.4.1 Outline of Thesis

Subsequent chapters of this thesis seek to build a complete understanding of the proposed anisotropic AMR algorithm in a logical manner, as well as display an accurate evaluation of its performance. Chapter 2 presents the governing equations considered in this work and gives a broad overview of the numerical techniques used by the proposed algorithm to solve the governing equations. Chapter 3 details each component of the proposed algorithm and covers the technical aspects of its implementation in the overall solution framework. The proposed anisotropic AMR algorithm is evaluated in Chapter 4 and compared to implementations of 3D isotropic AMR and 2D anisotropic AMR for a variety of flow problems. Finally, Chapter 5 discusses the results and summarizes the main contributions of the thesis while providing a road map for future research.

Chapter 2

Governing Equations and the Numerical Solution Scheme

The anisotropic AMR algorithm proposed in this thesis is applied to the solution of the three-dimensional Euler equations for which solutions are sought using an upwind finite-volume scheme. This chapter presents the Euler equations and describes in detail the finite-volume solution methodology used to solve the governing hyperbolic system of conservation laws for both steady-state and time-dependent problems. With appropriate treatments for the elliptic or diffusive fluxes, a similar solution methodology can be followed for the solution of other sets of governing equations, such as the Navier-Stokes equations governing laminar viscous compressible gaseous flows. The latter would represent a natural avenue for future follow-on studies.

2.1 Three-Dimensional Euler Equations

The Euler equations are a set of PDEs governing inviscid, compressible, continuum, fluid flows. The three-dimensional form of the Euler equations for an inviscid, polytropic gas are given as,

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = 0, \quad (2.1)$$

where $\mathbf{U}$ is the vector of conserved solution variables and $\vec{\mathbf{F}}$ is the solution flux dyad. For a three-dimensional Cartesian coordinate system, the Euler equations can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} = 0, \quad (2.2)$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \rho uh \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ \rho vh \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ \rho wh \end{bmatrix}, \quad (2.3)$$

and where ρ is the gas density, u , v , and w are the components of the velocity vector in the x , y , and z directions, $e = p/(\rho(\gamma - 1)) + u^2/2$ is the specific total energy, and γ is the ratio of specific heats. The vectors $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$, are the inviscid flux in the x , y , and z directions, respectively with $\vec{\mathbf{F}} = [\mathbf{F}, \mathbf{G}, \mathbf{H}]$. The ideal gas equation of state, $p = \rho RT$, is used to close the system, where p is the gas pressure, T is the gas temperature and R is the ideal gas constant. This set of PDEs forms a hyperbolic system of conservation equations that fully describes the behavior of an inviscid, compressible, polytropic, continuum, fluid by the conservation of mass, momentum, and energy.

Numerical solution of the Euler equations is generally made difficult due to the occurrence of nonlinear wave propagation and the possible formation of shocks or discontinuities. A way to solve these equations while capturing shocks or discontinuities is considered in the following sections. While solutions to the full Navier-Stokes equations would certainly be desirable for the treatment of many complex flow problems, the Euler equations are considered exclusively in this work as proof of concept of the proposed anisotropic AMR algorithm.

2.2 Upwind Finite Volume Method

The proposed anisotropic AMR algorithm developed in this work employs a Godunov-type [51] upwind finite volume spatial discretization scheme for the solution of the three-dimensional Euler equations. Upwind finite-volume methods are considered superior to other solution methods for problems involving complex flow physics, due to their intrinsic conservation, natural handling of shocks and discontinuities, and their flexibility in handling both structured and unstructured meshes, aiding in the treatment of complex geometries. Finite-volume methods are routinely used for the solution of partial differential equations arising from hyperbolic conservation laws having the weak conservation

form

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \mathbf{S}(\mathbf{U}), \quad (2.4)$$

where $\mathbf{U}$ is the solution vector, $\vec{\mathbf{F}}$ is the corresponding flux dyad, and $\mathbf{S}$ is the source vector. Domain discretization is accomplished by dividing the domain volume into conforming, non-overlapping, control volumes, or computational cells. The equation can be integrated over the control volume, $\mathcal{V}$, to obtain the integral form of the equation, setting the source term, $\mathbf{S}$, equal to zero,

$$\frac{d}{dt} \iiint_{\mathcal{V}} \mathbf{U} \, d\mathcal{V} + \iiint_{\mathcal{V}} \vec{\nabla} \cdot \vec{\mathbf{F}} \, d\mathcal{V} = 0, \quad (2.5)$$

then applying the Gauss divergence theorem and defining $\frac{1}{\mathcal{V}} \iiint_{\mathcal{V}} \mathbf{U} \, d\mathcal{V} = \bar{\mathbf{U}}$ as the average solution over the control volume and rearranging, Equation (2.5) becomes,

$$\frac{d\bar{\mathbf{U}}}{dt} = -\frac{1}{\mathcal{V}} \oint_{\mathcal{S}} \vec{\mathbf{F}} \cdot \vec{n} \, d\mathcal{S}, \quad (2.6)$$

where $\vec{n}$ is outward pointing unit vector, normal to the control volume surface, $\mathcal{S}$.

In 1959, Godunov proposed the first-order, upwind, shock capturing scheme for the one-dimensional form of the preceding conservation equations [51]

$$\bar{\mathbf{U}}_i^{n+1} = \bar{\mathbf{U}}_i^n - \frac{\Delta t}{\Delta x} \left[\mathcal{F}_{i+\frac{1}{2}}^n - \mathcal{F}_{i-\frac{1}{2}}^n \right], \quad (2.7)$$

where the piecewise constant average solution in cell i , $\bar{\mathbf{U}}_i^{n+1}$, is equal to the current solution, $\bar{\mathbf{U}}_i^n$, minus the sum of the numerical fluxes, $\mathcal{F}_{i+\frac{1}{2}}^n$ and $\mathcal{F}_{i-\frac{1}{2}}^n$ through the cell boundaries. Evaluating the numerical fluxes involves a solution to a Riemann problem [52] at each of the cell boundaries, which provides a natural upwinding of the solution and is described in later sections.

2.2.1 Semi-Discrete Form of the Euler Equations

Applying the finite-volume discretization described above to the conservation form of the three-dimensional Euler equations to a hexahedral computational cell as depicted in

Figure 2.1: Hexahedral cell at grid location i, j, k showing face normals.

Figure 2.1 results in the semi-discrete form

$$\frac{d\bar{\mathbf{U}}_{i,j,k}}{dt} = -\frac{1}{V_{i,j,k}} \sum_{m=1}^{N_f} \vec{\mathbf{F}}_{i,j,k,m} \cdot \vec{n}_{i,j,k,m} \Delta A_{i,j,k,m} = \mathbf{R}_{i,j,k}, \quad (2.8)$$

where $\bar{\mathbf{U}}_{i,j,k}$ is the cell average of the conserved solution at cell i, j, k , $V_{i,j,k}$ is the volume of the cell, $\vec{\mathbf{F}}_{i,j,k,m}$ is the flux through the m -th cell face, $\vec{n}_{i,j,k,m}$ is the outward pointing unit normal of the m -th cell face, and $A_{i,j,k,m}$ is the area of the m -th cell face. The variable, N_f , is the number of faces of the element, which is six for a hexahedral computational cell as illustrated in Figure 2.1.

2.2.2 Numerical Flux Evaluation

The semi-discrete form of the Euler equations of Equation (2.8) presents difficulties in the evaluation of the flux through cell faces, $\vec{\mathbf{F}}_{i,j,k,m}$, due to discontinuities in values of the solution that may be present at the cell boundaries caused by the discrete representation of the solution. To evaluate the flux through cell faces, a one-dimensional Riemann initial-value problem in the direction, x , normal to the cell face can be solved based on the initial conditions

$$\mathbf{U} = \begin{cases} \mathbf{U}_L, & \text{for } x < 0 \\ \mathbf{U}_R, & \text{for } x > 0, \end{cases} \quad (2.9)$$

where the solution, $\mathbf{U}(x=0)$, at the interface of the left and right states, $\mathbf{U}_L$ and $\mathbf{U}_R$, is desired. The solution of this Riemann problem should be as computationally inexpensive as possible, since flux evaluations are required for all faces of every cell, multiple times, at each time step or iteration.

Although an exact Riemann solver for an inviscid, compressible, gas can be used, such as one by Godunov [51] and the more efficient method by Gottlieb and Groth [52], they are relatively computationally expensive. Furthermore, for many fluid flows with non-ideal equations of state, there may exist no exact solution to the Riemann problem. Therefore, approximate Riemann solvers are used here in the evaluation of the numerical flux at the cell interfaces. For this work, the one-dimensional problem, normal to the cell faces is solved and then rotated back to the true frame of reference.

Roe Approximate Riemann Solver

A popular choice for an approximate Riemann solver is the Roe flux function [53]. Instead of solving for the exact solution to the Riemann problem based on the one-dimensional nonlinear Euler equations given by

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{U}}{\partial x} = 0, \quad (2.10)$$

where $\mathbf{A} = \partial \mathbf{F} / \partial \mathbf{U}$, the local solution is approximated by the exact solution to a linear problem given by,

$$\frac{\partial \mathbf{U}}{\partial t} + \tilde{\mathbf{A}} \frac{\partial \mathbf{U}}{\partial x} = 0, \quad (2.11)$$

where $\tilde{\mathbf{A}} = \mathbf{A}(\mathbf{U}_*)$ is a matrix with an average state, $\mathbf{U}_*(\mathbf{U}_L, \mathbf{U}_R)$ chosen such that $\tilde{\mathbf{A}}$ is constant [53]. The flux can then be approximated by,

$$\mathcal{F} = \begin{cases} \mathbf{F}(\mathbf{U}_L) + \sum_{\lambda_k^* < 0} r_{c,k}^* \alpha_k^* \lambda_k^*, & \alpha_k^* = l_{p,k}^* (\mathbf{W}_R - \mathbf{W}_L) \\ \mathbf{F}(\mathbf{U}_R) - \sum_{\lambda_k^* > 0} r_{c,k}^* \alpha_k^* \lambda_k^*, & \alpha_k^* = l_{p,k}^* (\mathbf{W}_R - \mathbf{W}_L), \end{cases} \quad (2.12)$$

where $r_{c,k}^*$, $l_{p,k}^*$, and λ_k^* are the right conservative, left primitive eigenvectors, and eigenvalues of $\tilde{\mathbf{A}}$ evaluated with the Roe averages of density, velocity, specific enthalpy, and

sound speed:

$$\begin{aligned}
 \rho_* &= \sqrt{\rho_L \rho_R}, \\
 \vec{v}_* &= [u_*, v_*, w_*]^T = \frac{\sqrt{\rho_L} \vec{v}_L + \sqrt{\rho_R} \vec{v}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
 h_* &= \frac{\sqrt{\rho_L} h_L + \sqrt{\rho_R} h_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \text{ and} \\
 a_* &= (\gamma - 1) \left[h_* - \frac{1}{2} (u_*^2 + v_*^2 + w_*^2) \right].
 \end{aligned} \tag{2.13}$$

The primitive solution change across the cell boundary is given by $\mathbf{W}_R - \mathbf{W}_L$. A limitation with the original Roe Riemann solver is the nonphysical solution of problems with rarefaction waves through a sonic point. Harten devised an entropy correction [54] for this problem that is included in all results in this thesis obtained using the Roe solver.

HLLE Approximate Riemann Solver

A simpler numerical flux function, developed by Harten, van Leer, Lax, and Einfeldt [55], considers the solution to the Riemann problem having the form,

$$\mathbf{U}(x, t) = \begin{cases} \mathbf{U}_L, & \text{if } \frac{x}{t} \leq \lambda^- \\ \mathbf{U}_*, & \text{if } \lambda^- < \frac{x}{t} < \lambda^+ \\ \mathbf{U}_R, & \text{if } \frac{x}{t} \geq \lambda^+, \end{cases} \tag{2.14}$$

where $\mathbf{U}_*$ is found by integrating Equation (2.1) around the closed path shown in Figure 2.2, and λ^+ and λ^- are the largest and smallest wave speeds defined as:

$$\begin{aligned}
 \lambda^+ &= \max(\lambda_R^{\max}, \lambda_{\text{Roe}}^{\max}) = \max(u_R + a_R, u_* + a_*), \\
 \lambda^- &= \min(\lambda_L^{\min}, \lambda_{\text{Roe}}^{\min}) = \min(u_L + a_L, u_* - a_*),
 \end{aligned} \tag{2.15}$$

where u_* and a_* are the Roe averaged states given in Equation (2.13).

The flux through the intermediate state, $\mathbf{F}_*$ is determined by considering a closed path integral around each half of the Riemann problem separately and then substituting in the expression for $\mathbf{U}_*$ to determine the numerical flux, $\mathcal{F}$, given by

$$\mathcal{F} = \begin{cases} \mathbf{F}_L = \mathbf{F}(\mathbf{U}_L), & \text{if } \frac{x}{t} \leq \lambda^- \\ \mathbf{F}_R = \mathbf{F}(\mathbf{U}_R), & \text{if } \lambda^- < \frac{x}{t} < \lambda^+ \\ \mathbf{F}_* = \frac{\lambda^+ \mathbf{F}_L - \lambda^- \mathbf{F}_R}{\lambda^+ - \lambda^-} + \frac{\lambda^+ \lambda^-}{\lambda^+ - \lambda^-} (\mathbf{U}_R - \mathbf{U}_L), & \text{if } \frac{x}{t} \geq \lambda^+, \end{cases} \tag{2.16}$$

Figure 2.2: HLLE wave pattern showing the integration path.

The HLLE flux function does not include a contact surface in its derivation, which may give unfavorable results for problems exhibiting contact surfaces. This problem can be rectified however, by using the modified form of the HLLE flux function, called HLLL, developed by Linde [56] which includes a contact surface in its derivation.

Choice of Flux Function

Choosing which flux function to use depends highly on what flow problem is being solved and the computational cost that can be afforded. Using the Roe flux function will result in a less diffusive solution, since more information about the wave structure is included, but at an increased computational cost. For flow problems exhibiting strong normal shock waves aligned with the grid lines, instabilities known as the “carbuncle phenomenon” can develop when using the Roe flux function [57].

2.2.3 Higher Resolution Finite Volume Scheme

For linear schemes, Godunov’s theorem states that the only way to ensure solution monotonicity is by a piecewise constant representation of the solution in each cell. This limitation is certainly not ideal since the spatial error of such representation is of $\mathcal{O}(\Delta h)$, where h is the representative cell size, forcing the use of an extremely fine mesh. Although solution monotonicity cannot be naturally achieved with a higher than zero-order solution representation and a linear scheme, higher solution accuracy and monotonicity can be achieved by adopting a non-linear scheme with non-linear slope limiting, resulting in a more accurate scheme such as the MUSCL-type schemes of van Leer [58]. Figure 2.3 shows an example of a piecewise linear solution representation superimposed over a piecewise constant solution representation.

In three dimensions, a second-order accurate solution in cell i can be represented by

$$\mathbf{U}_i(x) = \bar{\mathbf{U}}_i + \Phi_i \vec{\nabla} \mathbf{U}_i (\vec{x} - \vec{x}_i), \quad (2.17)$$

where $\vec{x}_i$ is the midpoint, or centroid, of cell i and Φ_i is a slope-limiting function that takes a value between 0 and 1 for each variable to ensure that no new maxima or minima are introduced. An example of the necessity for slope-limiting is given in Figure 2.3. If a piecewise-linear solution representation is used in cell i , a new maximum value would be introduced. The slope limiter takes a value of 0 for this case, effectively reverting to a piecewise constant representation. The slope-limiter used in this work is due to Venkatakrishnan [59] unless otherwise noted.

The primitive solution gradients, $\vec{\nabla} \mathbf{W}$, for cell i in equation (2.17), can be reconstruction using the 26 nearest neighbors by a least squares reconstruction that minimizes the second order error fit:

$$\epsilon_i^2 = \sum_{k=1}^N \epsilon_{ik}^2 = \sum_{k=1}^N \left[\Delta \mathbf{W}_{ik} - \vec{\nabla} \mathbf{W}_i \cdot \vec{r}_{ik} \right]^2, \quad (2.18)$$

where $\vec{r}_{ik} = \vec{x}_k - \vec{x}_i$ with $\vec{x}_i$ and $\vec{x}_k$ being the centroid of cell i and one of its N neighbors, cell k . For solution reconstruction in cells laying near block degeneracies, a stencil other than the 26 nearest neighbors may be required and the approach outlined by Gao and Groth [60] is used. The minimization of the previous equation results in a 3×3 system of linear equations which can then be solved for the solution gradients using Cramer's rule.

Figure 2.3: Unlimited piecewise-linear reconstruction in one dimension superimposed over a piecewise-constant solution representation. Notice the solution overshoot at $i + \frac{1}{2}$.

2.2.4 Time Marching Schemes

Integration of the semi-discrete form of the Euler equations in Equation (2.8) is accomplished herein by the application of explicit time-marching schemes. For time dependent solutions, the second-order Runge-Kutta time marching scheme [61] is used and for steady-state solutions, the multi-stage optimally-smoothing time marching scheme of van Leer *et al.* [62] is used. The multi-stage optimally-smoothing scheme uses specifically tuned coefficients to achieve optimal damping of high frequencies for the second order upwind discretization of the Euler equations and is shown to improve convergence over a Runge-Kutta scheme [62]. The multi-stage optimally-smoothing scheme takes the following form for a M -stage scheme:

$$\begin{aligned}\mathbf{U}^0 &= \mathbf{U}^n \\ \mathbf{U}^m &= \mathbf{U}^0 + \alpha_m \Delta t \mathbf{R}(\mathbf{U}^{m-1}), m = 1, \dots, M \\ \mathbf{U}^{n+1} &= \mathbf{U}^M,\end{aligned}\tag{2.19}$$

where $\mathbf{U}^n$ is the solution in one cell at the end of iteration n and $\mathbf{U}^{n+1}$ is the solution in the same cell at the end of iteration $n + 1$. The coefficients, α_m , are given in Table 2.1 for two through six stage schemes as well as the corresponding optimal CFL numbers.

	Number of Stages				
	2	3	4	5	6
α_1	0.4242	0.1918	0.1084	0.0695	0.0482
α_2	1	0.4929	0.2602	0.1602	0.1085
α_3		1	0.5052	0.2898	0.1885
α_4			1	0.5060	0.3050
α_5				1	0.5063
α_6					1
Optimal CFL	0.4693	0.6936	0.9214	1.1508	1.3805

Table 2.1: Coefficients and optimal CFL numbers for the multi-stage optimally smoothing scheme [62] for a second-order spatial discretization.

2.3 Parallel Implementation

As mentioned in the introduction, an efficient implementation of the anisotropic AMR framework is crucial to take advantage of the current high performance computing re-

sources necessary to solve complex flow problems. The parallel anisotropic AMR algorithm in this work uses the same block-based efficient and highly scalable implementation as developed by Gao and Groth [35]. With their implementation, the domain is naturally decomposed, owing to the block-based approach, with equally sized blocks. Blocks are divided equally across all available processors and solution information at block boundaries is exchanged with a neighboring block's overlapping ghost cells via a message passing procedure that sends or receives boundary solution and geometry information. Messages are sent using the standardized Message Passing Interface (MPI) [63] when communicating with blocks residing on different processors. Further details on the message passing procedure for anisotropic AMR will be discussed in the next chapter. The proposed 3D anisotropic AMR algorithm has been developed using the C++ programming language and an object oriented approach to be as general as possible, allowing for the solution of a wide range of flow problems and physics.

Chapter 3

Three-Dimensional Anisotropic Block-Based AMR

Adaptive mesh refinement (AMR) techniques offer a way to reduce the computational effort of solving a complex flow problem, by refining only the areas of the mesh requiring higher resolution to resolve pertinent flow features. This leads to a more efficient use of computational resources, allowing researchers to obtain results faster or to increase the resolution of their simulation, without incurring additional computational costs. Additional savings can be afforded by considering anisotropic AMR, which increases or decreases mesh resolution only in locations and directions of the mesh required to resolve the solution. Anisotropic flow features such as shock waves, flame fronts, or thin shear layers, can be resolved with lower computational cost with an anisotropic AMR approach. This chapter provides a complete description of the proposed three-dimensional anisotropic AMR algorithm and mesh refinement strategy.

3.1 Block-Based AMR

The proposed adaptive scheme adopts and then extends the three-dimensional block based approach as previously proposed by Groth and co-workers [7, 28, 64]. This AMR approach allows the adaptation of structured, body-fitted, meshes consisting of hexahedral computational cells as shown in Figures 3.1b and 3.1a. The body-fitted grid blocks permit anisotropic stretching and are key to conforming to complex geometries, while the block-based AMR approach lends itself to an efficient and highly scalable parallel

Figure 3.1: Three-dimensional hexahedral cell and example mesh.

implementation. The parallel, block-based approach has shown success in solving a wide range of complex flow problems with several examples mentioned in Section 1.1.

The block-based AMR scheme refines or coarsens entire grid blocks rather than individual computational cells or patches of cells. While this may lead to local over-refinement of the mesh, the benefits of a light, tree-based, data structure and computing and storing connectivity information as well as transferring solution information only on the block level, overpowers the disadvantage of over-refinement. A flexible, tree-based, data structure is used to store the connectivity and refinement history of grid blocks. Since individual blocks are refined, only the nearest neighbor information needs to be computed and stored, regardless of the order of solution reconstruction. The initial grid blocks are defined as ‘root’ blocks, each representing the base of their own tree. As the root blocks are refined, the trees ‘branch’ off from the root blocks, creating refined ‘children’ blocks that exactly overlap and replace their ‘parent’ block. The tree efficiently stores information about the refinement history of blocks, including the inactive, previously refined, ‘ancestor’ blocks.

Each body-fitted grid block contains a three-dimensional structured mesh with $N_\xi \times N_\eta \times N_\zeta$ total computational cells, where N_ξ , N_η , and N_ζ are the numbers of computational cells in the ξ , η , and ζ computational or logical directions of the block. These block directions correspond to logical ordering of the computational cells in each block. The block dimensions, N_ξ , N_η , and N_ζ must be even integers, but can have different

Figure 3.2: Refined blocks with overlapping ghost cells.

values. At each iteration level, the solution is obtained for each block independently, and then solution information is sent to or received from neighboring blocks. To facilitate the transfer of solution information across block boundaries, two layers of ghost cells, overlapping with adjacent block interior cells, are used. An example of refined blocks with overlapping ghost cells is illustrated in Figure 3.2. Grid blocks are restricted to having no greater than one refinement level difference between adjacent blocks. This restriction enforces a smooth transition from the finest refined blocks to the coarser blocks, and makes the implementation of the AMR algorithm less complicated and more efficient.

The proposed anisotropic AMR algorithm follows from the 2D anisotropic AMR previously described by Zhang and Groth [32, 33] for the Euler equations and the model advection-diffusion equations. The 2D scheme was coupled with high-order CENO finite-volume [29–31] and a Newton-Krylov-Schwartz implicit solver [26]. The anisotropic AMR method developed herein allows refinement or coarsening (de-refinement) in any computational direction of a block, irrespective of the refinement history. Instead of one refinement producing eight refined blocks, as is the case for a more typical isotropic AMR algorithm, 3D anisotropic AMR can produce two, four, or eight refined blocks, each summarized in Figure 3.12. A flexible binary-tree data structure stores the connectivity and

refinement history information and permits coarsening in any direction(s), independent of the previous refinement history.

While many of the ideas and much of the theory for the 2D anisotropic AMR scheme could be adopted directly in the implementation of the proposed 3D anisotropic AMR approach, many of the features of the two-dimensional method do not readily extend to the three-dimensional case in a straightforward manner. These differences and difficulties are identified and solved in the subsequent sections of this chapter.

3.1.1 Anisotropic AMR Procedure

Anisotropic AMR can be applied differently depending on the problem being solved. For example, when solving a steady state problem, it may be advantageous to begin the simulation with a coarse initial mesh and then apply the anisotropic AMR procedure only after a certain level of convergence has been achieved on the coarse mesh. Anisotropic AMR can then be applied to increase the refinement level as the solution develops. This process can then be repeated until the solution is sufficiently resolved. An alternative way to use anisotropic AMR, is to employ the procedure after some specified number of iterations or time steps, or duration in the simulation. For an unsteady problem, this is required so that the mesh is allowed to track the solution progression. In addition, given an initial solution field, the initial mesh can be adapted to any level of refinement before a simulation begins. In each instance that the 3D anisotropic AMR procedure is applied:

1. blocks are flagged to refine, coarsen, or to not change their refinement level for each computational direction (ξ, η, ζ) based on refinement criteria calculated for each block;
2. refinement flags are modified with a conflict checking procedure to ensure that there are no violations in refinement or coarsening rules;
3. all blocks flagged to refine, are refined in the flagged direction(s) and the binary tree is updated to reflect the refinements;
4. all blocks flagged to coarsen, are coarsened in the flagged direction(s) and connectivity is rearranged if the block's ancestors are not split in the coarsening direction(s);
5. neighbor information is re-computed and stored for each block; and

Figure 3.3: The “deadlock” arising as a consequence of restricting the block coarsening to one direction per application of AMR [33].

6. solution and geometry information is shared between adjacent blocks through message passing procedures.

Each of the steps presented above are outlined in detail in the following sections.

While it may seem that a more simplistic solution would be to allow each block to be adapted in only one coarsening or refinement direction per application of AMR, this in fact does not work well or easily for anisotropic AMR. As there are restrictions on the maximum refinement level difference between adjacent blocks, if the anisotropic AMR is limited to coarsen in only one direction per application of AMR, blocks can become stuck or trapped in “deadlock” situations. Such deadlocks were identified by Zhang [33] and an example is illustrated in Figure 3.3. If blocks A1 and A2 or B1 and B2 are coarsened, they would violate the restrictions on maximum resolution change across neighboring blocks and would be flagged to not coarsen by the conflict checking procedure. Since neither blocks can coarsen, the additional unwanted levels of refinement will be effectively locked in and the desired mesh coarsening would not be possible, even with subsequent applications of AMR.

Figure 3.4: Multi-block mesh with unstructured block connectivity. Grid block boundaries are outlined in black lines and the orientations are shown with arrows.

3.2 Block Connectivity

The anisotropic AMR algorithm considered in this work makes use of an unstructured root block connectivity identical to the one developed by Gao and Groth [27, 28] for 3D isotropic AMR. In standard structured block connectivity, four blocks must meet at an edge, and eight blocks must meet at a corner. In the unstructured case, corner and edge degeneracies are allowed. For example, three blocks could meet at an edge and only five blocks at a corner. Additionally, this work allows arbitrarily oriented root blocks. An example of a mesh with an unstructured root block connectivity is shown in Figure 3.4. In this example, only three blocks meet at an edge and the orientations of the individual grid blocks are different as indicated in the figure.

The root block connectivity is computed by an algorithm that compares nodal locations at the corners of the initial grid blocks to determine and then store the block face, edge, and corner neighbor information. Block orientation information is computed and then stored in a transformation matrix for each neighbor of a grid block. Further details of the implementation of the unstructured root-block connectivity are given in the work by Gao and Groth [27, 28].

Figure 3.5: 3D binary tree and the corresponding blocks after several refinements

3.2.1 Binary Tree Hierarchical Data Structure

As opposed to the octree data structure employed in the previous three-dimensional isotropic AMR algorithm [27, 28], a binary tree data structure is used. The binary tree data-structure permits division of the grid into just two children after each refinement, compared to the eight children produced by isotropic AMR. However, several refinements of the binary tree can be combined to create two, four, or eight grid blocks in one AMR step, allowing for greater flexibility. Each node of the binary tree, which can represent an ancestor or currently used block, has a particular split direction corresponding to the computational direction of refinement. Location information is represented as a sector: west or east, south or north, bottom or top that represents a specific refined half of the block's parent in the ξ , η , or ζ computational directions. Refinement level information, one for each computational direction, is stored for each node of the tree. Therefore, it is possible for binary tree nodes to be at three different refinement levels, each corresponding to a different direction, at one time.

An example of a refined mesh showing the corresponding grid blocks and the corresponding binary tree is depicted in Figure 3.5. The refinement level of a block can be found by ascending the tree from the node representing that block and counting the number of splits for each refinement direction. In the case of block **D** as shown in the figure, the number of splits for the ξ , η , and ζ directions are equal. Therefore, the block is at the same refinement level in all directions.

3.2.2 Computation of Connectivity

Efficient and rapid computation of block connectivity, or neighbor information, is essential for unsteady AMR since the neighbor information must be computed after each mesh adaptation. In unsteady flow applications with rapidly evolving flow features, AMR must be applied frequently and therefore connectivity must be re-computed frequently. Neighbor searching algorithms that rely on comparing node locations suffer from the computational overhead of using floating-point calculations. Therefore, a tree-based neighbor searching algorithm is preferred, where only the level and sector information in the tree is required and no floating-point calculations are necessary.

A strategy for efficient computation of block connectivity has been developed by Zhang and Groth [32,33] for 2D anisotropic AMR, which follows from tree-based neighbor searching algorithms by others [9,65]. A neighbor searching algorithm similar to the 2D anisotropic AMR implementation was developed for this work. Nearest neighbors are found for each of the 26 possible neighbor directions corresponding to face, edge, and corner neighbors. Due to refinement level constraints between adjacent blocks, the maximum number of neighbors for faces, edges, and corners are four, two, and one, respectively. Searches progress in one computational direction at a time, therefore edge and corner neighbors are found by a combination of two and three searches.

The neighbor searching process begins by first ascending the tree towards the root node until an ancestor block is found that is split in the direction of the search. This block is called the bridge and is a common ancestor of both the searching block and the neighbor block. If a root block is reached before a bridge is found, there are either no neighbors in the search direction, or the neighbor block is a descendent of a different root block, in which case the pre-computed root block connectivity information is used as the bridge. The root block connectivity is computed during generation of the initial mesh and is stored for the remainder of the simulation.

After the bridge is found, the tree is then descended from the bridge, based on the block sector information and split direction. For example, a block that is an east sector of its parent or ancestor and is searching for an east neighbor will have a western descending direction in ξ from the east child of the bridge. The tree is descended until an intermediate node in the tree is reached that is at the same level as the searching block in a direction other than the searching direction. For a face neighbor search, this node is descended further to get the individual neighbors. For an edge neighbor search, this searching

Figure 3.6: An example southeast edge neighbor search from the block shown in green to the block shown in red. Paths B1 and B2 are the bridges.

process is repeated in the remaining search direction(s) but this time, the search begins from the intermediate node. Again, once an edge node has been descended until it is at the same level as the searching block in a direction other than the searching direction, the individual neighbors are found. This process is repeated again for a corner neighbor search in the remaining direction starting from the edge neighbor intermediate node.

The searching process for an edge neighbor in $(\xi, -\eta, 0)$ can be visualized by referring to Figure 3.6. The first search looks for an intermediate neighbor in $(0, -\eta, 0)$. From the node indicated in green, the binary tree is ascended until a bridge in the η direction (B1) is found. The bridge is crossed, arriving at the intermediate node shown in blue. Another search is then computed from the intermediate node in the ξ direction and the tree is ascended until bridge, B2, is found. The tree is then descended towards the intermediate node and stops when a node is reached that is at an equal level to the intermediate node in η or ζ . This new node is called the second intermediate node. The second intermediate node is then descended towards the searching block until the edge neighbor is found. Unique to 3D anisotropic AMR, the descending steps can require extra logic when the intermediate node is at a lower refinement level in a direction other

than the searching direction(s). In these cases, it is key that the direction of descent be determined by comparing sector information for nodes at the same refinement level. This may require ascending the tree until both nodes are at the same level, before determining the descending direction.

3.2.3 Connectivity Rearrangement

The simplest way to coarsen blocks is to reverse the previous refinement for that block. While this method works in an unaltered fashion for isotropic AMR [27,28], where refinement and coarsening is not directionally dependent, it unfortunately fails for anisotropic AMR. Often, blocks are flagged to coarsen in directions that are not simply reverse-refinements, in order to adapt to rapidly changing flow features. If the highly restrictive coarsening process for isotropic AMR were in fact adopted, AMR deadlocks could develop, such that areas of the mesh would remain highly refined even though the solution has evolved and now demands de-refinement of the mesh. Because coarsening is called for in a direction other than the one in which the mesh was originally refined, there is no way to combine the children back into their original parent block and coarsening ceases in that area of the mesh.

Zhang and Groth devised a solution to this problem [32,33] by re-forming the binary tree refinement history in order to force blocks that are flagged to coarsen, to have the same parent. They observed that while the refined mesh is unique, the corresponding binary tree is not. Referring to Figure 3.5, it is apparent that the binary tree can be re-written in several different ways to result in the same final refined mesh. The basic connectivity rearrangement process works by first identifying a bridge between the two blocks. If the bridge is the parent of both blocks, then no connectivity rearrangement is necessary. Otherwise, if the grandparent of both blocks is the bridge, then the split of the bridge is swapped with the split of the parent blocks and the grandchildren of the bridge are swapped between the two children of the bridge. If the grandparent is not the bridge, then connectivity rearrangement is called recursively, with the parents of the coarsening blocks, until the bridge is found and the split direction has been propagated down the tree to the desired or correct location. An example of this rearrangement process is shown in Figure 3.7, where blocks A and B do not initially share the same parent but can be made to once the connectivity rearrangement as outlined above is performed.

Figure 3.7: Left: blocks **A** and **B** need to coarsen in the ξ direction but do not share a common parent. Right: binary trees showing the connectivity rearrangement required to force **A** and **B** to share a common parent.

Unfortunately, an identical connectivity rearrangement procedure cannot be adopted for 3D anisotropic AMR because of the possibility that parents of the rearranging blocks are at different refinement levels. While the conflict checking procedure ensures that blocks must be at the same refinement levels in order to coarsen, there are no restrictions on the refinement levels of their parents. For example, two blocks flagged to coarsen in the ξ -direction could have parents split in the η and ζ directions. A solution to this problem is to introduce an intermediate connectivity rearrangement, where at each level of connectivity rearrangement, the parent connectivity is re-written to ensure that both are at the same refinement levels. The intermediate rearrangement process is generalized as follows for two blocks, **A** and **B**, flagged to coarsen with parents at different refinement levels:

1. ascend the tree from block **A** until a bridge is found with the same split direction as **B**'s parent;
2. cross the bridge and descend the tree toward block **A** until the block descended to is at the same level as block **A** in at least one direction;
3. if the descended block is not split and is at a lower refinement level than **A** in any direction, start from step (1) with block **B** switched with block **A**;

Figure 3.8: Blocks **A** and **B** need to rearrange connectivity but their parents are at different levels, causing the rearrangement to fail. After an intermediate rearrangement, the connectivity of blocks **A** and **B** can be rearranged.

4. if the descended block is at a higher refinement level than block **A**, perform another intermediate rearrangement with the descended block in the direction of the higher refinement level; and finally,
5. rearrange the connectivity of block **A** and the descended neighbor to force blocks **A** and **B** to have parents at equal refinement levels.

An example scenario for block rearrangement in the fully 3D anisotropic case using the above algorithm is given in Figure 3.8. Blocks **A** and **B** are flagged to coarsen in ξ but their parents have different refinement levels in the η and ζ directions. In applying the above algorithm, the connectivity of block **A** and its neighbor in η are first rearranged so that the parents of **A** and **B** are at the same refinement levels. Following the intermediate rearrangement, blocks **A** and **B** are then free to rearrange connectivity in ξ , forcing them to share the same parent. While this example depicts only one intermediate rearrangement, it is also possible that multiple intermediate connectivity rearrangements are required to rearrange the connectivity of two blocks. This adds significant complication to the implementation of connectivity rearrangement, although in practice, connectivity rearrangement only happens during coarsening and only for the blocks that require it, adding little to the actual overall computational cost of the problem.

3.3 Refinement Flagging and Conflict Checking

3.3.1 Refinement Criteria

Before each anisotropic AMR, blocks are flagged to refine, coarsen, or not change their refinement levels in each computational direction. While refinement and coarsening criteria for AMR based on adjoint error estimation strategies, such as the methods proposed by Venditti and Darmofal in one and two dimensions [66,67] and for anisotropic refinement [68,69], as well as *hp* adaptivity by Heuveline and Rannacher [70] and *a priori* error estimation by Rannacher and Vexler [71], are expected to be more optimal, the refinement criteria here are restricted to heuristic or physics-based approaches for their relative simplicity [5,72], as well as because the focus of the thesis has been of the anisotropic AMR procedure itself, not improved refinement criteria. Physics-based refinement criteria have been used extensively by Groth and co-workers [7,20–22,25–28,30,31,34,37–39] when applying block-based AMR techniques to a wide variety of flow problems. Future, follow-on studies will undoubtedly consider error-based refinement strategies for use with the proposed anisotropic AMR method.

For the Euler equations of interest here, the refinement criteria for the ξ , η , and ζ computational directions based on the gradient of density, ρ , are

$$\epsilon_\xi = \frac{1}{\rho} \left(\vec{\nabla} \rho \cdot \Delta \tilde{\mathbf{X}} \right), \quad \epsilon_\eta = \frac{1}{\rho} \left(\vec{\nabla} \rho \cdot \Delta \tilde{\mathbf{Y}} \right), \quad \epsilon_\zeta = \frac{1}{\rho} \left(\vec{\nabla} \rho \cdot \Delta \tilde{\mathbf{Z}} \right), \quad (3.1)$$

where $\Delta \tilde{\mathbf{X}}$, $\Delta \tilde{\mathbf{Y}}$ and $\Delta \tilde{\mathbf{Z}}$ are the vector differences between the midpoints of the east and west, north and south, and top and bottom cell faces, respectively.

The above refinement criteria are evaluated and normalized by the cell averaged solution for every cell in each block and the maximum value over all cells for each direction is assigned as the refinement criteria for a block. Using the maximum value and user specified thresholds for refinement or coarsening, a given block is then flagged to refine, coarsen, or to not change its refinement level.

3.3.2 Conflict Checking

The preceding refinement criteria give an indication of which blocks should be flagged to refine or coarsen so as to better resolve the solution, but further modification of the AMR flags is necessary to ensure that none of the externally imposed restrictions on

the mesh adaptation are violated in the subsequent refinement and coarsening process. These imposed restrictions on refinement flags identify and eliminate and prevent the various combinations of refinement flags that would cause:

1. adjacent blocks to have a resolution change greater than two;
2. the refinement level of a block to exceed user-defined maximum or minimum refinement levels;
3. a block flagged to coarsen that does not have a sibling/neighbor flagged to coarsen or after refinement the sibling/neighbor is not at the same refinement level; or
4. an invalid binary tree connectivity.

In general, these restrictions are imposed on the flagged blocks for numerical considerations, e.g., a smooth transition from the finest refined blocks to the coarser blocks, or to make the implementation of the algorithm less involved and more efficient, e.g., limiting refinement level differences to reduce the complexity of solution transfer and connectivity computation. Overall, preference is given to refinement and blocks are coarsened only if the coarsening does not affect the refinement of neighboring blocks.

The conflict checking procedure looks for violations of the restrictions outlined above in four steps: (i) first sibling checks; (ii) level checks; (iii) second sibling checks; and, (iv) invalid tree checks. In each step, if a violation is found, the refinement flags of the offending blocks are altered. All conflicts are resolved in an iterative procedure that repeats until no more conflicts are found.

The first sibling checks ensure that if a block is flagged to coarsen, it has a neighboring block at the same refinement level also flagged to coarsen in the same direction. In addition, if the neighboring block is coarser in a direction other than the coarsening direction, the neighbors in that direction must be checked as well. An example of a first sibling check is provided at the top of Figure 3.9, where coarsening is not permitted since there are no neighbor blocks flagged to coarsen in the coarsening direction.

Level checks ensure that the maximum refinement level difference between adjacent blocks is no greater than one for each computational direction. For example, a block can be one refinement level lower or higher than the neighboring block in each of the computational directions. Figure 3.10 shows a block flagged to refine, that if refined, it would exceed the maximum refinement level difference. To avoid violating the maximum refinement level difference constraint, the neighboring block is forced to refine.

Figure 3.9: Example conflicts resolved through first and second sibling checks.

The second sibling checks make sure that if a block is flagged to coarsen, it has a neighboring block that will be at the same refinement level in the directions other than the coarsening direction prior to coarsening. The case at the bottom of Figure 3.9 illustrates the application of a second sibling check. Both blocks are flagged to coarsen in the x direction, but only one block is flagged to refine in the z direction. Since refinement is preferred over coarsening, both blocks are forced to refine in z and each refined block is then coarsened in the x direction.

Figure 3.10: Example refinement level conflict resolved with level checks.

Figure 3.11: Sequence of anisotropic mesh adaptations that create a mesh which cannot be represented by the 3D binary tree data structure.

One of the challenges of developing a 3D anisotropic AMR algorithm, is the treatment of and logic for conflict checking for the additional block configurations that do not arise in 2D anisotropic AMR. Considering only the physical refinement and coarsening conflicts described above is not sufficient. Figure 3.11 illustrates a sequence of refinements and coarsenings that lead to a final mesh which can not be represented by the binary tree and must therefore be prevented. These cases are detected before they happen by first ascending the binary tree until a bridge is reached that is split in the direction of coarsening. Next, all active blocks that are descendants of the bridge are recorded. If two descendents are at a lower refinement level than the searching block in both computational directions other than the coarsening direction, the block is flagged so as not to coarsen. For the example in Figure 3.11, descending from the bridge, it is apparent that one active block is at a lower level in ζ and one is at a lower level in ξ . If blocks **A** and **B** were allowed to coarsen in η , there would be no way to create a binary tree to represent the coarsened mesh.

3.4 Anisotropic Mesh Refinement and Coarsening

Adapting the mesh to evolving unsteady flow features in an anisotropic manner requires a robust refinement algorithm to handle shock interactions, turbulent flame sheets, and/or

flow separation. To achieve this, a single block undergoing refinement can result in one of the seven distinct refinements as shown in Figure 3.12, and leads to the creation of two, four, or eight newly refined blocks, respectively. Following refinement, the number of cells comprising the same volume of the original block increases by a factor of two, four, or eight, depending on how the original block was flagged to refine. The mesh resolution is increased in each direction accordingly. Similarly, coarsening can be done for each computational direction as well as for all possible combinations of directions.

3.4.1 Refinement

In the present implementation, refined node locations can be computed in one of two ways: by using a midpoint method or by using a method based on the grid metrics. For the midpoint method, new node locations are computed by taking the average of the coarse nodes defining an edge, a face, or a volume. For refinement in one direction, new nodes are computed only on edges; for refinement in two directions, new nodes are computed on edges and faces; and for refinement in three directions, new nodes are computed on edges, faces, and volumes. While this simple midpoint method requires the fewest computations, it may in many cases result in poor representation of curved boundaries and a grid metrics based method may be preferred.

The computation of refined node locations based on grid metrics is effective in preserving the grid line topology and clustering, particularly important near curved bound-

Figure 3.12: All possible single-step anisotropic refinements.

aries, and results in a smoother refined mesh than that provided by the midpoint approach [27, 28]. The grid metrics approach used in this work is based on the previous ideas for 3D isotropic AMR [27, 28] and 2D anisotropic AMR [32, 33]. The refined node locations are computed using the second order Taylor expansion

$$\begin{aligned}
\vec{x}(\xi + \Delta\xi, \eta + \Delta\eta, \zeta + \Delta\zeta) = & \vec{x}(\xi, \eta, \zeta) + \left. \frac{\partial \vec{x}}{\partial \xi} \right|_{\xi, \eta, \zeta} \Delta\xi + \left. \frac{\partial \vec{x}}{\partial \eta} \right|_{\xi, \eta, \zeta} \Delta\eta + \left. \frac{\partial \vec{x}}{\partial \zeta} \right|_{\xi, \eta, \zeta} \Delta\zeta \\
& + \frac{1}{2} \left(\left. \frac{\partial^2 \vec{x}}{\partial \xi^2} \right|_{\xi, \eta, \zeta} (\Delta\xi)^2 + 2 \left. \frac{\partial^2 \vec{x}}{\partial \xi \partial \eta} \right|_{\xi, \eta, \zeta} \Delta\xi \Delta\eta + \left. \frac{\partial^2 \vec{x}}{\partial \eta^2} \right|_{\xi, \eta, \zeta} (\Delta\eta)^2 \right. \\
& \left. + 2 \left. \frac{\partial^2 \vec{x}}{\partial \xi \partial \zeta} \right|_{\xi, \eta, \zeta} \Delta\xi \Delta\zeta + \left. \frac{\partial^2 \vec{x}}{\partial \zeta^2} \right|_{\xi, \eta, \zeta} (\Delta\zeta)^2 + 2 \left. \frac{\partial^2 \vec{x}}{\partial \eta \partial \zeta} \right|_{\xi, \eta, \zeta} \Delta\eta \Delta\zeta \right) \\
& + \mathcal{O}((\Delta\xi)^3, (\Delta\eta)^3, (\Delta\zeta)^3) \quad (3.2)
\end{aligned}$$

from the coarse nodes, where $\vec{x}(\xi, \eta, \zeta)$ is the location in physical space that is mapped from the computational space (ξ, η, ζ) by means of the metrics of the coordinate transformation.

In Equation (3.2) above, the first-and second-order partial derivatives associated with the coordinate transformation are approximated using second order centered differences for the block interior nodes and third order forward or backward differences for the nodes residing on the block boundaries. If refining in one direction, fine node locations on cell edges are evaluated using the average of the Taylor expansions from the coarse node locations on the edges. Similarly, for refinement in two directions, fine nodes locations on the cell faces are computed from the coarse node locations using the average of the Taylor expansions from the coarse node locations on the cell faces. For refinement in three directions, the Taylor expansions from the coarse node locations are averaged to compute the fine node location in the center of the cell volumes. An example of refinement in the ξ direction is given in Figure 3.13, showing the newly introduced nodes of the refined grid.

During refinement, one of the newly created grid and solution blocks takes the coarse parent block and the remaining new blocks are taken from a pool (or list) of available resources which can come from any processor involved in the computation. Following refinement of the mesh, the solution is transferred from the coarse interior cells of the original block to the fine interior cells of the refined blocks via one of two prolongation procedures. For steady-state simulations, where the transient solution is not important,

Figure 3.13: Anisotropic refinement of a cell using the grid metrics approach in the ξ computational direction. Fine node locations are indicated with small dots and coarse node locations with large dots.

the prolongation procedure uses direct injection, where the cell-averaged solution in the coarse cells is directly assigned as the cell average of the overlapping fine cells. When performing unsteady simulations, the solution in the coarse cells is reconstructed using linear least squares and then evaluated at the centroid of each overlapping fine cell to get the cell average.

After creating the refined blocks and transferring the solution content, the binary tree data structure must be updated to reflect the mesh refinement and new grid block topology. By definition, the binary tree supports only two children blocks for each parent block, and cannot natively handle refinements with four or eight children, such as the examples shown in Figure 3.12. It is possible to create refinements in more than one direction by refining each block sequentially, e.g., for a refinement in ξ and η , blocks are refined in ξ and then each of the children are refined in η as in the implementation of 2D anisotropic AMR [32, 33]. Unfortunately, if such a refinement procedure is applied directly with multiple successive prolongations, it would suffer both from some inefficiencies as well as a loss of solution information between refinements. To circumvent these limitations, the new blocks are created and the solution is prolonged in a single step. Only then is the binary tree updated to appear as if the refinement process occurred as a sequence of individual steps.

3.4.2 Coarsening

Anisotropic coarsening of the mesh simply removes every other node in the computational direction(s) of interest. One of the blocks in the data structure is assigned or becomes the new coarse block, and the remaining unused blocks are given back to the pool of available resources to be used when another block undergoes refinement. Following coarsening of the mesh, the solution in the original fine blocks is transferred to the new coarse block by an area-weighted average restriction process, where the new coarse cell averaged solution is

$$\mathbf{U}_C = \frac{1}{V_C} \sum_{n=1}^{N_F} \mathbf{U}_n V_n, \quad (3.3)$$

where $\mathbf{U}_n$ and V_n are the fine cell average solution and volume, respectively, and N_F is the number of fine cells overlapping the coarse cell.

Like the refinement procedure outlined above, coarsening in any number of the three directions can also be completed in a single step, separating the block representations in the binary tree from that of the actual grid blocks. In addition, coarsening is allowed in any direction(s) irrespective of the refinement history. For coarsening of a single block, coarsening the binary tree is completed first, and a list of the blocks involved in the coarsening process is stored. Coarsening in multiple directions is done with a similar process to refinement, where the tree is coarsened in each direction sequentially, but the physical blocks are coarsened in one step. If blocks do not share the same parent, the connectivity rearrangement procedure, outlined in section 3.2.3, is performed. Following coarsening of the tree, the list of all blocks involved in coarsening of the single block are sent to the procedure that coarsens the physical grid blocks as outlined above.

3.5 Inter-Block Solution Transfer

3.5.1 Ghost-Cell Solution Information

In the proposed multi-block approach, each grid block requires updated solution information at the block boundaries to provide the correct computed flux into and out of the block and to reconstruct the solution for each residual update. To facilitate this exchange of information between blocks, two layers of ghost cells are used to overlap the neighboring blocks on each face, edge, and corner as shown in Figure 3.2. For two neighboring blocks, the solution is transferred from the interior cells of one block to the overlapping

ghost cells of its neighboring block. Ghost cells are grouped into 26 boundary elements representing the sets overlapping ghost cells on block faces, edges, or corners of the block.

The parallel implementation is more complicated, since blocks involved in the solution transfer may reside on different processors, requiring the use of message passing routines to transfer the necessary solution information. To send solution information, a one-dimensional buffer is loaded with the prolonged, restricted, or directly-injected solution information in the order it will be unloaded. The order of unloading is dependent on the orientation of the neighboring block as defined in Section 3.2. Once the buffer is loaded, it is tagged with a unique identifier and sent using MPI [63]. For a block to receive solution information, a receive request is sent, and the buffer is tagged with the unique identifier of the sending block. In addition, geometry information is sent and received to update the ghost cells, ensuring that they conform to the neighbor's overlapped interior cells.

This process is complicated by the creation of resolution changes between neighboring blocks after AMR. With the isotropic AMR approach, these resolution changes can be coarser, finer, or no resolution change, for each of the 26 block faces, edges, or corners. While there are many resolution changes for the isotropic case, there are only three for each boundary element of a block, greatly simplifying the solution transfer procedure.

If one considers now the anisotropic case in three-dimensions, the possible resolution changes are coarser, finer, or no resolution change in each of the three computational directions and for each of the 26 block boundary elements. That leads to 27 possible resolution changes for each boundary element. Furthermore, since a block can be at a coarser level in one or more directions, but at a finer level in others, both prolongation and restriction may be necessary for one solution transfer. Figure 3.14 shows several of the many possible message passing scenarios. Scenario 1 shows message passing for a resolution change of coarse-to-fine in one direction and the required prolongation in one direction. In scenario 2, restriction in two directions is performed before transferring the solution to the neighbor's ghost cells. Conversely in scenario 3, prolongation in two directions is performed before transferring the solution to the neighbor's ghost cells. Scenario 4 shows a more complicated message pass for a resolution change of coarse-to-fine in one direction, as well as fine-to-coarse in another direction. As there are effectively two opposite resolution changes in scenario 4, prolongation followed by restriction is required to send solution information to the neighbor's ghost cells.

Figure 3.14: Several message passing scenarios shown for anisotropic AMR.

Unique to cases similar to that shown in Scenario 4, which require both solution prolongation and restriction, the solution information transferred from both adjacent blocks involved must first be evaluated by performing solution reconstruction. This presents a problem, however, since the reconstruction stencil requires updated ghost cell solution information from both blocks, before any messages can be exchanged. With no correction, these cases introduce errors during message passing by using the previous time or iteration level ghost cell values in the reconstruction. This problem was first identified by Zhang and Groth [33] and was found to have little effect on the solution of steady-state flow problems using explicit iterations, but was found to be detrimental to convergence for implicit iterations and no doubt introduces unrecoverable error into time-accurate simulations.

While not considered in the present implementation, one possible fix for scenario 4 is to iteratively send and receive the messages until the error between the values of the interior cells of the neighbor block and the overlapping ghost cells, reduces to an acceptable value as proposed by Zhang and Groth [33]. While iterative message passing

Figure 3.15: Proposed message passing procedure to handle anisotropic resolution changes more generally. Sent cells are shown in green, and the reconstruction stencil for the blue cell is shown in red.

was shown to be a passable solution [33], it adds a layer of unacceptable computational expense and greatly increases the ratio of inter-block communication to computation. A far more optimal solution would be to change the way the message passing procedure is currently implemented. Instead of prolonging and/or restricting solution information to fit the cells of the neighboring blocks, sending this information, and then unloading it, the necessary interior solution and geometry information can be sent to the neighbor directly. Once the neighbor receives the solution information, it can then use this information directly in the required solution reconstruction. This proposed message passing procedure would eliminate a layer of complexity in the implementation of message passing, although solution reconstruction would become more complicated on block boundaries, sharing similarities with an unstructured approach. The proposed solution is illustrated in Figure 3.15 for a similar case to scenario 4.

3.5.2 Interface Flux Corrections

Finally, as a natural consequence of refining and coarsening blocks, many adjacent blocks will have resolution changes, and therefore non-conforming cells, where the faces of both blocks meet. To maintain the conservation properties of the finite-volume scheme, the flux through the non-conforming faces must be corrected [1, 2]. Typical procedures for flux correction involve correcting the flux through the coarse cell faces by the difference

between the coarse flux and the flux through the two or four fine cell faces. The residual in the coarse cell is then updated with

$$\vec{\mathbf{R}}_{i,j,k} = \vec{\mathbf{R}}_{i,j,k} - \frac{A_{\text{coarse}} \Delta \vec{\mathbf{F}}}{V_{i,j,k}}, \quad (3.4)$$

where $\vec{\mathbf{R}}_{i,j,k}$ is the solution residual (RHS of Equation (2.8)), $V_{i,j,k}$ is the coarse cell volume, A_{coarse} is the coarse face area, and $\Delta \vec{\mathbf{F}}$ is the difference between the combined fine flux and the original coarse mesh flux: $\Delta \vec{\mathbf{F}} = \vec{\mathbf{F}}_{\text{Fine}} - \vec{\mathbf{F}}_{\text{Coarse}}$.

For tangential resolution changes across block boundaries where there is no strictly fine or coarse cell, as in scenario 4 of Figure 3.14, another approach to flux correction is necessary and is illustrated in Figure 3.16. Instead of correcting the coarse cell residual, an average flux is calculated associated with the total conforming area and given by

$$\vec{\mathbf{F}}_{\text{Avg}} = \frac{\vec{\mathbf{F}}_A + \vec{\mathbf{F}}_B}{2}, \quad (3.5)$$

and then the flux through the cell faces that comprise the conforming area is corrected in each block to be equal to this average flux. The flux corrections for area A and B are then

$$\Delta \vec{\mathbf{F}}_A = \vec{\mathbf{F}}_{\text{Avg}} - \vec{\mathbf{F}}_A \text{ and } \Delta \vec{\mathbf{F}}_B = \vec{\mathbf{F}}_{\text{Avg}} - \vec{\mathbf{F}}_B. \quad (3.6)$$

and the residual in all of the cells on the interface are updated accordingly using Equation (3.5) above.

Figure 3.16: Flux correction procedure for resolution changes in two directions across the block interfaces. The dotted line shows the corresponding cell faces used in the correction.

As the proposed flux correction averages the flux contributions from adjacent cells, some solution inaccuracies are introduced by this procedure. Furthermore, an accurate representation of this flux correction may be an important impediment to extending the anisotropic AMR procedure to higher order accuracy. The possible inaccuracy of the flux correction also provides more motivation for the proposed alternate message passing procedure outlined above, since flux corrections would then be entirely eliminated by naturally handling the non-conforming cells in the finite-volume solution method directly.

It is also worth mentioning that instead of correcting the flux or changing the message passing procedure as outlined for these latter cases, these particular grid-block topologies can be avoided by adding another rule in the conflict checking procedure so as to prevent grid-block connectivity of the type shown in Figure 3.16. This restriction has been shown to work well for cell-based 3D anisotropic AMR approaches [13]. However, an investigation performed here into this possible fix showed that not allowing cases similar to scenario 4 in Figure 3.14, results in the significantly reduced effectiveness of the anisotropic AMR algorithm. It was found to have the effect of not only increasing local levels of mesh refinement, but also the levels of mesh refinement in neighboring areas in order to maintain refinement level constraints between adjacent blocks.

Chapter 4

Numerical Results

This chapter applies the three-dimensional, anisotropic AMR algorithm, developed and described in the preceding chapter in combination with the finite-volume scheme of Chapter 2, to the solution of several flow problems. Each flow problem has been chosen to assess and evaluate the proposed algorithm, as well as assess performance gains as compared to an isotropic approach, under a variety of situations. Steady-state and unsteady flow problems with multiple geometries are solved and solutions are compared to three-dimensional isotropic AMR results as well as analytical solutions where possible. Results are shown for an unsteady shock-cube, supersonic flow over a bump, supersonic radial outflow, and supersonic flow over a sphere.

4.1 Unsteady Shock-Cube Initial Value Problem

An important capability of the anisotropic AMR algorithm is the ability to adapt to evolving unsteady flow features, such as shock waves, in an efficient manner and to allow both refinement and coarsening. The shock-cube initial value problem is an excellent evaluation of these features, requiring high mesh resolution in regions of wave interactions and fast adaptation to the quickly evolving solution.

The initial conditions for the shock-cube problem in a cubed domain filled with air having 1 m long edges and centered at the origin were taken to be standard atmospheric conditions with the density, $\rho = 1.225 \text{ kg/m}^3$, pressure, $p = 101.325 \text{ kPa}$, and the velocity, $\vec{V} = 0 \text{ m/s}$ for $x < 0$, $y < 0$, and $z < 0$ and eight times the standard atmospheric density and ten times standard atmospheric pressure elsewhere. A ratio of specific heats for the air of $\gamma = 1.4$ was assumed. Reflection boundary conditions were imposed on all walls of

the cube. Before the simulation begins, initial single block meshes containing $8 \times 8 \times 8$ cells for both anisotropic and isotropic AMR were sequentially refined six times based on the initial conditions in order to resolve the discontinuity between the two states.

Following the initial refinement, the simulations were commenced and the anisotropic and isotropic AMR refinement and coarsening procedures were repeated every 15 time steps until a maximum time of 0.75 ms was reached. The AMR procedure was limited to a maximum of seven refinement levels for each of the computational directions, so as to limit the total number of computational cells used in the simulations. The anisotropic and isotropic adapted grid blocks and solution density contours were compared at four different times [$t = 0.05$ ms, 0.25 ms, 0.5 ms, and 0.75 ms] in Figure 4.1.

It is evident from the results of Figure 4.1 that, throughout the simulation, both anisotropic and isotropic AMR show similar spatial resolution and track the solution well. At $t = 0.05$ ms, there are large anisotropic features in the solution which are exploited by anisotropic AMR using blocks with high aspect ratio cells, resulting in a reduction of 89% in the number of cells compared to that of isotropic AMR. From $t = 0.25$ ms to $t = 0.50$ ms, the solution begins to move out of alignment with the grid line orientation of the computational blocks, causing anisotropic AMR to produce a more isotropically refined mesh in some regions. Even though there is less anisotropic refinement, the anisotropic procedure still offers a mesh size reduction of 79% for $t = 0.25$ ms and 65% for $t = 0.50$ ms. At $t = 0.75$ ms, the solution has moved further from alignment with the mesh. Nevertheless, anisotropic refinement still manages to achieve a reduction in the overall size of the mesh of 53%.

Figure 4.1: Comparison between anisotropic and isotropic AMR for $t = 0.05$ ms, $t = 0.25$ ms, $t = 0.5$ ms, and $t = 0.75$ ms. The contour plots show the gas density in kg/m^3 and the blocks are outlined with solid lines.

4.2 Steady Supersonic Channel Flow Over a Bump

In order to demonstrate the applicability of 3D anisotropic AMR to the solution of steady-state flow problems and to compare its performance to the original 2D anisotropic AMR scheme of Zhang and Groth [32, 33], supersonic flow over a smooth bump in a channel was analyzed. The particular problem consists of a rectangular cross section channel, 5.5 m long, 2 m high, and 1 m wide with a smoothed circular bump at the bottom of the channel. Air with a flow Mach number of $M = 1.4$ and standard atmospheric density and pressure of $p = 101.325$ kPa and $\rho = 1.225$ kg/m³, respectively, enters the channel through the left boundary and exits through the right boundary of the domain. A ratio of specific heats of $\gamma = 1.4$ was assumed.

The initial mesh consists of 4 blocks along the length of the channel, 2 blocks along the height of the channel, and 1 block along the out of plane direction, with each block containing $8 \times 8 \times 2$ cells. Reflection boundary conditions are imposed on the smooth bump boundary as well as the top of the channel. The $M = 1.4$ flow is fixed at the inlet and linear extrapolation is imposed at the outlet. The multi-stage optimally smoothing scheme by van Leer *et al.* [62] with local time stepping and a CFL of 0.5 is used for the time integration.

Six anisotropic adaptive mesh refinements were performed with the 3D anisotropic AMR method and compared with the 2D anisotropic AMR scheme. Following the sixth and final mesh refinement, the adapted meshes for the 3D and 2D anisotropic AMR schemes contained 1,191 and 1,165 blocks, respectively. The predicted density contours and the adapted blocks at refinement levels of one, two, and three are shown in Figures 4.2a, 4.2e, and 4.3e for 3D anisotropic AMR and in Figures 4.2b, 4.2f, and 4.3f for 2D anisotropic AMR. The predicted density contours and the adapted blocks at refinement levels of four, five, and six are shown in Figures 4.3a, 4.3c, and 4.3e for 3D anisotropic AMR and in Figures 4.3b, 4.3d, and 4.3f for 2D anisotropic AMR.

For each level of refinement in Figures 4.2 and 4.3, the the density contours and adapted blocks for 3D anisotropic and 2D anisotropic AMR are similar. Comparing the refined meshes at the sixth refinement level, it is clear that each mesh is adapted well to the shock waves in the converged solution. Furthermore, both 3D and 2D anisotropic AMR achieve similar spatial resolution over the majority of the flow field and the density contours are in agreement. Figures 4.4a and 4.4b give the solution density along the line $y = 0.25$ m for 3D and 2D anisotropic AMR at refinement levels one through six. As the

refinement level increases, the solution density rapidly approaches a converged solution and both the 3D and 2D solutions are very similar along the line for each refinement level.

Figure 4.2: Supersonic, $M = 1.4$ flow over a smooth bump in a channel at refinement levels one through three for 3D and 2D anisotropic AMR. Predicted solution density contours in kg/m^3 and block outlines are shown.

Figure 4.3: Supersonic, $M = 1.4$ flow over a smooth bump in a channel at refinement levels four through six for 3D and 2D anisotropic AMR. Predicted solution density contours in kg/m^3 and block outlines are shown.

Figure 4.4: Density along the line $y = 0.25$ m at refinement levels one through six.

Following the six refinements for 3D anisotropic AMR, the number of blocks in the out of plane direction remains one, which is an expected result considering the two-dimensionality of the flow. Note that solving this highly 2D problem with isotropic AMR would require an excessive additional number of cells to achieve a similar spatial resolution, since refinement would be necessary in the out of plane direction.

4.3 Steady Supersonic Radial Outflow

Many physical phenomena of interest to researchers, especially ones involving Earth or space physics, are spherical in nature. Spherical computational domains are therefore necessary to solve these problems, but standard structured grids cannot be created with a spherical shape. Fortunately, a cubed-sphere [49] multi-block mesh can be created, which consists of six sectors, or blocks, connected with edge degeneracies, forming an inner, hollow sphere, and an outer spherical shell. Originally proposed by Ronchi *et al.* [49] for the solution of partial differential equations in spherical geometry, it has since been incorporated into an isotropic block-based AMR framework by Ivan *et al.* [41, 50].

As an initial investigation into anisotropic AMR on the cubed-sphere mesh [41, 49, 50], a simple steady-state flow problem with an analytical solution was solved. The problem consists of a spherical inflow with a radius of $R_i = 1$ m and a spherical outflow at a radius of $R_o = 4$ m. Supersonic air enters through the inner sphere with velocity only

in the radial direction, V_r , expands, and exits supersonically through the outflow sphere. The inflow was fixed with flow density, radial velocity, and pressure of $\rho_i = 10 \text{ kg/m}^3$, $V_{r,i} = 4.5 \text{ m/s}$, and $p_i = 26 \text{ Pa}$, respectively. A ratio of specific heats of $\gamma = 1.4$ was assumed. Linear extrapolation is used at the outflow boundary, since it is assumed that gas will be exiting at supersonic speed. The analytical solution to this flow problem is described by Ivan [41] as the numerical solution at any radial location, r , to the equation

$$C_3 - \frac{1}{r^2 V_r \left[(C_2 - V_r^2)^{\frac{1}{\gamma-1}} \right]} = 0, \quad (4.1)$$

where

$$C_3 = \frac{1}{\left(\frac{2\gamma}{\gamma-1} \frac{p_i}{\rho_i} \right)^{\frac{1}{\gamma-1}} R_i^2 V_{r,i}} \text{ and } C_2 = \frac{2\gamma}{\gamma-1} \frac{p_i}{\rho_i} + V_{r,i}^2 \quad (4.2)$$

are constants depending on the inflow conditions.

The initial mesh consisted of a refined cubed-sphere with 24 blocks, each containing $32 \times 32 \times 16$ cells. The integration of the governing equations to machine-zero is done by the multi-stage optimally smoothing scheme by van Leer *et al.* [62] with local time-stepping. Three further adaptive refinements were completed and the solution was converged at each refinement level. The predicted flow density and Mach number obtained using both anisotropic and isotropic AMR at the final level of refinement are shown on a quarter-slice of the cubed-sphere in Figures 4.5 and 4.6, respectively.

Figure 4.5: Quarter slices of the predicted solution showing density in kg/m^3 and block outlines for anisotropic and isotropic AMR.

Figure 4.6: Quarter slices of the predicted solution showing Mach number and block outlines for anisotropic and isotropic AMR.

It can be observed that the solution varies only along the radial direction and that anisotropic AMR exploits this by refining only in the radial direction, while isotropic AMR refines more uniformly. Therefore, the large reduction of 96% in the total cell count compared to isotropic AMR is expected.

4.3.1 Accuracy Assessment

To assess whether the large reduction in the total cell count also corresponds to a large reduction in solution error compared to the isotropic AMR solution, an accuracy assessment has also been performed for the supersonic radial outflow problem. After obtaining the converged solution for each refinement level, the solutions are compared to the exact solution given above, and the L_1 , L_2 , and L_∞ error norms are calculated. The results of this accuracy assessment are shown for anisotropic AMR, isotropic AMR, and a uniformly refined mesh in Figure 4.7. The uniformly refined mesh shows the expected second-order reduction in error. Referring to the isotropic AMR mesh in Figures 4.5 and 4.6, much of the mesh is refined uniformly, leading to marginal improvement over uniform refinement of the mesh. On the contrary, anisotropic AMR shows a large reduction in the error norms compared to uniformly refining the mesh. By the final anisotropic refinement level, the error norms have reached the same level as the final isotropic refined mesh, but with a total number of cells equal to the first isotropic refined mesh. Figure 4.8 showing the radial density profile for anisotropic and isotropic AMR further illustrates this

trend. In particular, the density profiles at each refinement level for anisotropic AMR are similar to that of isotropic AMR, even though anisotropic AMR is using over 90% less computational cells to achieve the same solution.

Although anisotropic AMR shows superior performance over isotropic AMR for the accuracy assessment depicted in Figure 4.7, if the number of refinements are increased beyond three, the L_1 , L_2 , and L_∞ error norms do not continue to decrease as would be desired and begin leveling off. This behavior is attributed to the use of the physics-based refinement criteria that do not accurately reflect the actual solution errors. Since the solution varies only in the radial direction, anisotropic AMR does not refine in the angular directions of the sphere. Therefore, as the refinement level increases and the errors in the radial direction decrease, the errors caused by low refinement in the angular directions, and therefore poor representation of the sphere, begin to dominate. This behavior demonstrates the need for error-based refinement indicators [66–71], and in particular, efficient, directional error-based refinement indicators. The heuristic refinement criteria used in this work are therefore a clear limitation.

Figure 4.7: Accuracy assessment based on the exact solution density of the supersonic radial outflow problem comparing anisotropic AMR, isotropic AMR, and the solution on a uniformly refined mesh.

Figure 4.8: Radial density profile comparison between 3D anisotropic and isotropic AMR for refinement levels one through four.

4.4 Steady Supersonic Flow Past a Sphere

While the supersonic radial outflow problem offers a good initial evaluation of anisotropic AMR for the cubed-sphere [41, 49, 50] mesh, the inherent spherical symmetry of the problem leads to refinement only in one computational direction. A better evaluation of anisotropic AMR would be a flow problem requiring refinements in all computational directions, yet still containing significant anisotropic features. Therefore, a more three-dimensional flow problem of a steady-state, supersonic flow past a sphere is presented.

The flow problem geometry consists of a solid sphere of radius 1 m and a far-field spherical boundary of radius 10 m. To focus refinements entirely on the bow shock, the problem has been reduced by considering only a partial sphere. A five sector cubed-sphere initial mesh is used with four blocks stacked in the radial direction for each of the sectors as shown in Figure 4.9, with each block containing $8 \times 8 \times 8$ cells. A reflection boundary condition is imposed on the inner sphere surface. The outer sphere is fixed at free stream conditions for air with a flow Mach number of $M = 2.0$ and a density and pressure of $\rho = 1.225 \text{ kg/m}^3$ and $p = 101.325 \text{ kPa}$, respectively. A ratio of specific heats of $\gamma = 1.4$ is assumed. Air exits the domain through the outflow boundary supersonically, therefore a linear extrapolation boundary condition is used. Integration of the governing equations is completed by the multi-stage optimally smoothing scheme by van Leer *et*

Figure 4.9: The initial mesh for the steady-state supersonic sphere problem with one of the five sectors removed for clarity.

al. [62] with local time stepping. The HLLE approximate Riemann solver [55] is used for the flux evaluation at cell boundaries.

Solutions to the preceding supersonic flow problem are found using both the anisotropic and isotropic AMR algorithms, each with six levels of refinement of the initial mesh. Density contours and grid blocks are shown for each refinement level in Figure 4.10 for anisotropic and isotropic AMR at refinement levels one through five. The solutions at the sixth and final refinement level are given in Figure 4.11. It is apparent that anisotropic and isotropic AMR achieve similar spatial resolution across the bow shock, near the stagnation region, but towards the outer region of the bow shock near the outflow, isotropic AMR refines more. This discrepancy is caused by different ways of calculating the refinement criteria between the two methods. A lower refinement threshold is required with anisotropic AMR for this flow problem so that the weaker portion of the bow shock can be resolved, although the optimal combination of refinement and coarsening thresholds for this is unknown. Since it is recognized that the gradient-based refinement criteria used in this work is far from ideal, further solutions with lower refinement thresholds to attempt to get more comparable refinements with isotropic AMR in the outer regions of the bow shock, were not sought.

In the region near the bow shock, where the solution is well-aligned with the block computational directions, anisotropic AMR refines significantly less than isotropic AMR tangential to the shock-surface. Furthermore, in the angular directions of the sphere, anisotropic AMR shows the greatest reduction in the number of computational cells. The

large savings can be visualized by comparing slices of the final refined mesh of anisotropic AMR and isotropic AMR near the outflow plane, perpendicular to the free-stream flow in Figures 4.12a and 4.12b. In the radial direction of the slice, both anisotropic and isotropic AMR show similar levels of refinement, but in the angular direction, anisotropic AMR refines substantially less. This leads to a reduction in total cell count of approximately 85% compared to isotropic AMR. To assess the performance of 3D anisotropic AMR compared to isotropic AMR, the center line density and Mach number solutions are compared. Comparing the center line density and Mach number from the stagnation point to the free stream in Figure 4.13a, both predict similar shock location and post-shock stagnation conditions. Figure 4.13b illustrates the center line density from the stagnation point to the free stream at each refinement level for anisotropic AMR. Successive anisotropic refinements of the mesh allow the solution to rapidly approach a converged solution, clearly demonstrating the effectiveness of the anisotropic AMR strategy. Since there are no exact solutions for this flow problem, an accuracy assessment was not performed.

Figure 4.10: Slices of $M = 2.0$ air flow over a sphere solution density contours. Isotropic adapted solutions shown in the left column, anisotropic shown in the right column. Meshes are shown from a refinement level of one to five.

Figure 4.11: Slice of $M = 2.0$ air flow over a sphere solution contours. Isotropic adapted solution density and Mach number shown at a refinement level of six with 26,753 blocks on the left. Anisotropic adapted solution density and Mach number shown at a refinement level of six with 3,789 blocks on the right.

Figure 4.12: Slice of the supersonic sphere mesh in the axial direction at $x = -0.2$ m showing blocks outlined in black.

Figure 4.13: Comparison between 3D anisotropic and isotropic AMR of center line density and Mach number from the stagnation point to the free-stream. The center line density for anisotropic AMR at each refinement level is also shown.

Chapter 5

Conclusions

The motivation for the development of a 3D anisotropic AMR algorithm so as to solve flow problems of greater complexity in a more computationally efficient manner than a traditional isotropic approach to AMR was clearly demonstrated in the introduction of this thesis. Block-based AMR, using body-fitted grid blocks for accurate geometry representation, allows for large reductions in computational complexity by only refining areas of the mesh required to resolve the solution. Further reduction in computational complexity is obtained with the use of anisotropic AMR, allowing solution-directed, directional refinement of body-fitted grid blocks. Focusing on directionally refining a mesh to exploit anisotropic flow features, such as shock waves, thin shear layers, boundary layers, or flame fronts, common to most complex flow physics, offers a way to significantly reduce the high computational cost of solving such problems. Moreover, a two-dimensional, anisotropic, block-based AMR algorithm was recently developed by Zhang and Groth [32,33] and was shown to afford significant reductions in mesh sizes.

An extension of parallel, block-based, anisotropic AMR [32,33] to three spatial dimensions was therefore considered in this thesis for the solution of the Euler equations. More traditional block-based AMR approaches refine blocks uniformly, *i.e.* eight blocks are created for each refinement. With an anisotropic approach, two, four, or eight refined blocks can be created, resulting in five distinct possible refinements for a single block, to better adapt to evolving flow features. Additionally, coarsening of grid blocks is allowed in any direction, or combination of directions, irrespective of the refinement history. The proposed algorithm makes use of a flexible binary tree data structure to store the refinement history and connectivity of the mesh. The binary tree is coupled with computationally efficient algorithms for connectivity computation, refinement, and connectiv-

ity rearrangement. Solutions to three-dimensional, inviscid, compressible flow problems have been computed on body-fitted adapted meshes, including the cubed-sphere, and compared to isotropic AMR solutions.

5.1 Main Contributions

The two-dimensional, block-based, anisotropic AMR algorithm provided a strong foundation upon which the proposed three-dimensional algorithm has been built. However, a large part of the development of the proposed three-dimensional anisotropic AMR algorithm has involved determining what complexities are introduced by extending the two-dimensional algorithm to three dimensions and how they can be dealt with and incorporated. In addition, the similarities between the two and three-dimensional anisotropic AMR algorithms were initially over-estimated. Significant changes to the two-dimensional anisotropic connectivity computation, connectivity rearrangement, conflict checking, message passing, refinement, and coarsening algorithms have been required to incorporate the added complexities of the three-dimensional algorithm.

Developing a three-dimensional connectivity computation algorithm has required extra logic to handle ambiguities in descending directions for searches in multiple directions. Connectivity rearrangement required the addition of an intermediate rearrangement process to handle parents of coarsening blocks with differing refinement levels. An additional conflict check needed to be devised to detect coarsening scenarios that would result in the creation of an invalid binary tree. Due to the substantially increased number of resolution changes between adjacent blocks, compared to two-dimensional anisotropic AMR, a more generalized message passing procedure was developed including an average flux correction for non-conforming blocks.

A new refinement and coarsening procedure has also been developed that refines and coarsens grid blocks in one-step, for any combination of refinement/coarsening directions, instead of the sequential refinement and coarsening of two-dimensional anisotropic AMR. This permits prolongation using piecewise-linear reconstruction and avoids the addition of error into the solution by sequentially prolonging and restricting the solution for each refinement/coarsening direction. Furthermore, it is consistent with isotropic AMR for isotropic refining and coarsening. Unstructured root block connectivity is also permitted, allowing the treatment of complex geometries. The implementation of the proposed algorithm has been assessed by obtaining solutions for several three-dimensional flow

problems. It was found that large reductions in the number of computational cells could be achieved for highly anisotropic flow problems, with features well aligned with the computational directions of the mesh. Significant computational savings, as measured in terms of the sizes of the computational mesh, were realized for flow problems exhibiting more of a mixture of isotropic and anisotropic flow features, or features not well-aligned with grid block computational directions. Additionally, an initial accuracy assessment has highlighted the need for a better anisotropic refinement criteria, although more rigorous accuracy assessments should be completed in the near future.

5.2 Future Work

As indicated in Chapter 3, the current message passing implementation has limitations with message passing scenarios involving both prolongation and restriction, requiring the use of an iterative message passing fix for accurate computation of unsteady flow problems. In addition, the average flux correction procedure required for non-conforming blocks, could be another possible source of error. Therefore, an alternate implementation of the message procedure should be investigated, such as the procedure outlined in Section 3.5 that sends all the information required to reconstruct the solution in the receiving block.

While this thesis considers only solutions to relatively simple flow problems and geometries, more complicated flow physics and geometries need to be considered to truly evaluate the computational savings afforded by using anisotropic AMR. Another key limitation of the current implementation is the lack of an error-based refinement criteria [66–71]. While the gradient-based criteria used in this work is capable of efficiently detecting flow features which may be large sources of error, it provides no indication of how well the solution is resolved. Furthermore, in situations such as described in Section 4.3, a gradient-based criteria can fail to direct the solution refinement so as to improve solution accuracy. Extension to higher-order accuracy and an implicit solver, such as the CENO scheme [29–31] and Newton-Krylov-Schwartz [26], would also appear to be desirable so as to introduce further computational cost savings. Since these have already been developed and studied for use with block-based isotropic AMR [27, 28] and 2D block-based anisotropic AMR [32, 33] and for complex flow problems [27, 34–36, 38, 39, 41], the extension to 3D anisotropic AMR should therefore be straightforward.

Bibliography

- [1] Berger, M. J. and Oliger, J., “Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations,” *Journal of Computational Physics*, Vol. 53, 1984, pp. 484–512.
- [2] Berger, M. J. and Colella, P., “Local Adaptive Mesh Refinement for Shock Hydrodynamics,” *Journal of Computational Physics*, Vol. 82, 1989, pp. 67–84.
- [3] Berger, M. J. and LeVeque, R. J., “An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries,” Paper 89-1930, AIAA, June 1989.
- [4] De Zeeuw, D. and Powell, K. G., “An Adaptively Refined Cartesian Mesh Solver for the Euler Equations,” *Journal of Computational Physics*, Vol. 104, 1993, pp. 56–68.
- [5] Powell, K. G., Roe, P. L., and Quirk, J., “Adaptive-Mesh Algorithms for Computational Fluid Dynamics,” *Algorithmic Trends in Computational Fluid Dynamics*, edited by M. Y. Hussaini, A. Kumar, and M. D. Salas, Springer-Verlag, New York, 1993, pp. 303–337.
- [6] Quirk, J. J. and Hanebutte, U. R., “A Parallel Adaptive Mesh Refinement Algorithm,” Report 93-63, ICASE, August 1993.
- [7] Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., “A Parallel Adaptive 3D MHD Scheme for Modeling Coronal and Solar Wind Plasma Flows,” *Space Science Reviews*, Vol. 87, 1999, pp. 193–198.
- [8] Hawken, D. F., Gottlieb, J. J., and Hansen, J. S., “Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations,” *Journal of Computational Physics*, Vol. 95, 1991, pp. 254–302.

- [9] De Zeeuw, D. L., *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*, Ph.D. thesis, University of Michigan, September 1993.
- [10] Aftomis, M. J., Berger, M. J., and Melton, J. E., “Robust and Efficient Cartesian Mesh Generation for Component-Base Geometry,” *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.
- [11] Aftomis, M. J., Berger, M. J., and Adomavicius, G., “A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries,” Paper 2000-0808, AIAA, January 2000.
- [12] Ham, F., Lien, F.-S., and Strong, A. B., “A Cartesian Grid Method with Transient Anisotropic Adaption,” *Journal of Computational Physics*, Vol. 179, 2002, pp. 469–494.
- [13] Iaccarino, G. and Ham, F., “Automatic mesh generation for LES in complex geometries,” *Annual Research Briefs*, 2005, pp. 460–490.
- [14] Iaccarino, G. and Ham, F., “LES on Cartesian Grids with Anisotropic Refinement,” *Complex Effects in Large Eddy Simulations*, Vol. 56, 2007, pp. 219–233.
- [15] Wang, Z. J., Cphen, R. F., Hariharan, N., Przekwas, A., and Grove, D., “A 2n tree based automated viscous Cartesian grid methodology for feature capturing,” Paper 99-3300, AIAA, 1999.
- [16] Wang, Z. J. and Chen, R. F., “Anisotropic Solution-Adaptive Viscous Cartesian Grid Method for Turbulent Flow Simulation,” *AIAA Journal*, Vol. 40, No. 10, October 2002, pp. 1969–1978.
- [17] Berger, M. J., “Data Structures for Adaptive Grid Generation,” *SIAM Journal for Scientific and Statistical Computing*, Vol. 904, 1986.
- [18] Berger, M. J. and Saltzman, J. S., “AMR on the CM-2,” *Applied Numerical Mathematics*, Vol. 14, 1994, pp. 239–253.
- [19] Stout, Q. F., De Zeeuw, D. L., Gombosi, T. I., Groth, C. P. T., Marshall, H. G., and Powell, K. G., “Adaptive Blocks: A High-Performance Data Structure,” *Proceedings of SC97*, San Jose, California, U.S.A., November 1997.

- [20] Groth, C. P. T., Zeeuw, D. L. D., Powell, K. G., Gombosi, T. I., and Stout, Q. F., “A Parallel Solution-Adaptive Scheme for Ideal Magnetohydrodynamics,” Paper 99-3273, AIAA, June 1999.
- [21] Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., “Three-Dimensional MHD Simulation of Coronal Mass Ejections,” *Advances in Space Research*, Vol. 26, No. 5, 2000, pp. 793–800.
- [22] Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., “Global Three-Dimensional MHD Simulation of a Space Weather Event: CME Formation, Interplanetary Propagation, and Interaction with the Magnetosphere,” *Journal of Geophysical Research*, Vol. 105, No. A11, 2000, pp. 25,053–25,078.
- [23] Gombosi, T. I., De Zeeuw, D. L., Groth, C. P. T., and Powell, K. G., “Magnetospheric Configuration for Parker-Spiral IMF Conditions: Results of a 3D AMR MHD Simulation,” *Advances in Space Research*, Vol. 26, No. 1, 1999, pp. 139–149.
- [24] Liu, Y., Nagy, A. F., Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., “3D Multi-fluid MHD Studies of the Solar Wind Interaction with Mars,” *Geophysical Research Letters*, Vol. 26, No. 17, 1999, pp. 2689–2692.
- [25] Sachdev, J., Groth, C., and Gottlieb, J., “A Parallel Solution-Adaptive Scheme for Predicting Multi-Phase Core Flows in Solid Propellant Rocket Motors,” *International Journal of Computational Fluid Dynamics*, Vol. 19, No. 2, 2005, pp. 159–177.
- [26] Groth, C. P. T. and Northrup, S. A., “Parallel Implicit Adaptive Mesh Refinement Scheme for Body-Fitted Multi-Block Mesh,” Paper 2005-5333, AIAA, June 2005.
- [27] Gao, X., *A Parallel Solution-Adaptive Method for Turbulent Non-Premixed Combusting Flows*, Ph.D. thesis, University of Toronto, August 2008.
- [28] Gao, X. and Groth, C. P. T., “A Parallel Solution-Adaptive Method for Three-Dimensional Turbulent Non-Premixed Combusting Flows,” *Journal of Computational Physics*, Vol. 229, No. 5, 2010, pp. 3250–3275.
- [29] Ivan, L. and Groth, C. P. T., “High-Order Central ENO Scheme for Hyperbolic Conservation Laws on Body-Fitted Multi-Block Mesh,” *Proceedings of the 14th Annual Conference of the CFD Society of Canada, Kingston, Ontario, Canada, July 16–18, 2006*, No. paper 1181, CFD Society of Canada, 2006, pp. 317–324.

- [30] Ivan, L. and Groth, C. P. T., “High-Order Central ENO Finite-Volume Scheme with Adaptive Mesh Refinement,” Paper 2007-4324, AIAA, June 2007.
- [31] Ivan, L., *Development of High-Order CENO Finite-Volume Schemes with Block-Based Adaptive Mesh Refinement*, Ph.D. thesis, University of Toronto, October 2010.
- [32] Zhang, Z. J. and Groth, C. P. T., “Parallel high-order anisotropic block-based adaptive mesh refinement finite-volume scheme,” Paper 2011-3695, AIAA, June 2011.
- [33] Zhang, Z. J., *Parallel Anisotropic Block-Based Adaptive Mesh Refinement Finite-Volume Scheme*, Master’s thesis, University of Toronto, November 2011.
- [34] Northrup, S. A. and Groth, C. P. T., “Solution of Laminar Diffusion Flames Using a Parallel Adaptive Mesh Refinement Algorithm,” Paper 2005-0547, AIAA, January 2005.
- [35] Gao, X., Northrup, S., and Groth, C. P. T., “Parallel solution-adaptive method for two-dimensional non-premixed combustng flows,” *Progress in Computational Fluid Dynamics, an International Journal*, Vol. 11, No. 2, 2011, pp. 76–95.
- [36] Gao, X. and Groth, C. P. T., “A Parallel Adaptive Mesh Refinement Algorithm for Predicting Turbulent Non-Premixed Combusting Flows,” *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 5, 2006, pp. 349–357.
- [37] McDonald, J. G. and Groth, C. P. T., “Numerical modeling of micron-scale flows using the Gaussian moment closure,” Paper 2005-5035, AIAA, June 2005.
- [38] Charest, M. R. J., Groth, C. P. T., and Gülder, Ö. L., “A computational framework for predicting laminar reactive flows with soot formation,” *Combustion Theory and Modelling*, Vol. 14, No. 6, 2010, pp. 793–825.
- [39] Charest, M. R., Groth, C. P. T., and Gülder, Ö. L., “Solution of the equation of radiative transfer using a Newton–Krylov approach and adaptive mesh refinement,” *Journal of Computational Physics*, Vol. 231, No. 8, 2012, pp. 3023–3040.
- [40] Sachdev, J. S., Groth, C. P. T., and Gottlieb, J. J., “Parallel solution-adaptive scheme for multi-phase core flows in rocket motors,” Paper 03–4106, AIAA, 2003.

- [41] Ivan, L., De Sterck, H., Northrup, S. A., and Groth, C. P. T., “Hyperbolic conservation laws on three-dimensional cubed-sphere grids: A parallel solution-adaptive simulation framework,” Submitted.
- [42] Lin, C.-W., Percival, S., and Gotimer, E. H., “Chimera Composite Grid Scheme,” <http://www.nswccd.navy.mil/hyd/tec-rep/chi-com-gri/index.html>.
- [43] E.P.Boden and Toro, E. F., “A Combined Chimera-AMR Technique for Computing Hyperbolic PDEs,” *Proceedings of the Fifth Annual Conference of the CFD Society of Canada, Toronto, Ontario, Canada*, CFD Society of Canada, 1997, pp. 5.13–5.18.
- [44] Brislawn, K. D., Brown, D. L., Chesshire, G., and Saltzman, J., “Adaptively-Refined Overlapping Grids for the Numerical Solution of Hyperbolic Systems of Conservation Laws,” Report LA-UR-95-257, Los Alamos National Laboratory, 1995.
- [45] Chesshire, G. and Henshaw, W., “Composite Overlapping Meshes for the Solution of Partial Differential Equations,” *Journal of Computational Physics*, Vol. 90, 1990, pp. 1–64.
- [46] Henshaw, W. D., “Adaptive Mesh Refinement on Overlapping,” *Adaptive Mesh Refinement – Theory and Applications*, Springer Berlin Heidelberg, 2005.
- [47] van der Holst, B. and Keppens, R., “Hybrid Block-AMR in Cartesian and Curvilinear Coordinates:MHD Applications,” *Journal of Computational Physics*, Vol. 226, 2007, pp. 925–946.
- [48] Zheng, J. Z. X. and Groth, C. P. T., “Block-Based Adaptive Mesh Refinement Finite-Volume Scheme for Hybrid Multi-Block Meshes,” July 2012.
- [49] Ronchi, C., Iacono, R., and Paolucci, P. S., “The cubed sphere: a new method for the solution of partial differential equations in spherical geometry,” *Journal of Computational Physics*, Vol. 124, No. 1, 1996, pp. 93–114.
- [50] Ivan, L., De Sterck, H., Northrup, S. A., and Groth, C. P. T., “Three-dimensional MHD on cubed-sphere grids: Parallel solution-adaptive simulation framework,” No. 2011-3382, AIAA, June 2011.

- [51] Godunov, S. K., “Finite-Difference Method for Numerical Computations of Discontinuous Solutions of the Equations of Fluid Dynamics,” *Matematicheskii Sbornik*, Vol. 47, 1959, pp. 271–306.
- [52] Gottlieb, J. J. and Groth, C. P. T., “Assessment of Riemann Solvers for Unsteady One-Dimensional Inviscid Flows of Perfect Gases,” *Journal of Computational Physics*, Vol. 78, 1988, pp. 437–458.
- [53] Roe, P. L., “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.
- [54] Harten, A., “High Resolution Schemes for Hyperbolic Conservation Laws,” *Journal of Computational Physics*, Vol. 49, 1983, pp. 357–393.
- [55] Harten, A., Lax, P. D., and van Leer, B., “On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws,” *SIAM Review*, Vol. 25, No. 1, 1983, pp. 35–61.
- [56] Linde, T., “A Practical, General-Purpose, Two-State HLL Riemann Solver for Hyperbolic Conservation Laws,” *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 391–402.
- [57] Pandolfi, M. and D’Ambrosio, D., “Numerical Instabilities in Upwind Methods: Analysis and Cures for the “Carbuncle Phenomenon”,” *Journal of Computational Physics*, Vol. 166, No. 2, 2001, pp. 271–301.
- [58] van Leer, B., “Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov’s Method,” *Journal of Computational Physics*, Vol. 32, 1979, pp. 101–136.
- [59] Venkatakrishnan, V., “On the Accuracy of Limiters and Convergence to Steady State Solutions,” Paper 93-0880, AIAA, January 1993.
- [60] Gao, X., *A Parallel Solution-Adaptive Method for Turbulent Non-Premixed Combusting Flows*, Ph.D. thesis, University of Toronto, September 2008.
- [61] Lomax, H., Pulliam, T., and Zingg, D., *Fundamentals of Computational Fluid Dynamics*, Springer-Verlag Berlin Heidelberg, 2001.

- [62] van Leer, B., Tai, C. H., and Powell, K. G., “Design of Optimally-Smoothing Multi-Stage Schemes for the Euler Equations,” Paper 89-1933-CP, AIAA, June 1989.
- [63] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI*, MIT Press, Cambridge, Massachusetts, 1999.
- [64] Gao, X. and Groth, C. P. T., “Parallel Adaptive Mesh Refinement Scheme for Three-Dimensional Turbulent Non-Premixed Combustion,” Paper 2008-1017, AIAA, January 2008.
- [65] Coirier, W. J., *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Michigan, 1994.
- [66] Venditti, D. A. and Darmofal, D. L., “Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow,” *Journal of Computational Physics*, Vol. 164, 2000, pp. 204–277.
- [67] Venditti, D. A. and Darmofal, D. L., “Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows,” *Journal of Computational Physics*, Vol. 176, 2002, pp. 40–69.
- [68] Venditti, D. A. and Darmofal, D. L., “Anisotropic Adaptation for Functional Outputs of Viscous Flow Simulations,” Paper 2003-3845, AIAA, June 2003.
- [69] Venditti, D. A. and Darmofal, D. L., “Anisotropic Grid Dadaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows,” *Journal of Computational Physics*, Vol. 187, 2003, pp. 22– 46.
- [70] Heuveline, V. and Rannacher, R., “Duality-Based Adaptivity in the *hp*-Finite Element Method,” *Journal of Numerical Mathematics*, Vol. 11, 2003, pp. 95–103.
- [71] Rannacher, R. and Vexler, B., “A priori Error Estimates for the Finite Element Discretization of Elliptic Parameter Identification Problems with Pointwise Measurements,” *SIAM Journal on Control and Optimization*, Vol. 44, 2005, pp. 1844–1863.
- [72] Paillère, H., Powell, K. G., and De Zeeuw, D. L., “A Wave-Model-Based Refinement Criterion for Adaptive-Grid Computation of Compressible Flows,” Paper 92-0322, AIAA, January 1992.