AIAA 2013-2442

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

M. J. Williamschen* and C. P. T. Groth †
University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Toronto, Ontario, M3H 5T6, Canada

A parallel, anisotropic, block-based, adaptive mesh refinement (AMR) algorithm is proposed and described for the solution of physically complex flow problems with both highly disparate and anisotropic spatial scales and flow features on three-dimensional, multi-block, body-fitted, hexahedral meshes. The block-based AMR is used to allow local refinement of the mesh and for its efficient and highly scalable parallel implementation. The body-fitted hexahedral grid blocks with unstructured root block topology and connectivity are used to afford the treatment of complex geometries. Instead of using more traditional isotropic mesh refinement strategies, the proposed AMR scheme uses a binary tree hierarchical data structure to permit anisotropic refinement of the grid blocks in a preferred coordinate direction as dictated by appropriately selected physics-based refinement criteria. The anisotropic coarsening of the grid blocks in a manner that is independent of the refinement history allows the mesh to rapidly re-adapt for unsteady flow applications. Overall, the proposed anisotropic AMR procedure allows for more efficient and accurate capturing of complex flow features such as shocks, boundary layers, or flame fronts. The AMR scheme is applied in conjunction with an upwind finite-volume spatial discretization scheme to the solution of the Euler equations for inviscid compressible gaseous flow. Steady-state and time-varying flow problems are considered on anisotropic adapted meshes. Anisotropic adapted cubed-sphere grids are investigated. The potential of anisotropic AMR for simulation of complex flows in an efficient and generalized manner is demonstrated.

I. Introduction

I.A. Motivation

As computational fluid dynamics (CFD) evolves and its use becomes more widespread in various science and engineering fields, problems involving increasingly complex flow physics, often coupled with complicated geometries, arise. Typically, these flow problems are difficult to solve due to numerical stiffness associated with disparate spatial and temporal scales. Moreover, they often require the solution of additional sub-physics equations and/or mathematical models. Specific classes of flows exhibiting these complex features include: (*i*) turbulent flows around complex geometries; (*ii*) chemically reacting flows, especially combustion; (*iii*) electrically conductive flows; and (*iv*) micro-scale flows.

One way to decrease the computational burden of solving complex flow problems without resorting to simplified or reduced-order physical models and while still maintaining the desired solution accuracy is to use adaptive mesh refinement (AMR). Increasing the number of cells in only the areas of the computational domain that need to be resolved while leaving other areas unchanged, AMR adapts the mesh to treat the disparate spatial scales without placing extreme demands on computing resources that would be caused by over resolving the entire solution domain. Parallel block-based AMR[1–3] methods have been recently proposed by Groth and co-workers[1,4–8] for the solution of complex flow problems on multi-block body-fitted meshes. They have been applied very successfully to the solution of complex flow problems such as non-premixed laminar[5] and turbulent[6] flames as well as turbulent multi-phase rocket core flows,[7] magnetohydrodynamics (MHD) simulations,[1,4] and micron-scale flows.[9]

*M. A. Sc. Candidate, mike.williamschen@mail.utoronto.ca
†Professor, Senior Member AIAA, groth@utias.utoronto.ca

One limitation of the block-based approaches mentioned above is that they are all restricted to isotropic refinement of the mesh, where the grid blocks are coarsened or refined equally in all directions. While the use of body-fitted grid blocks with grid line stretching does permit the use of grids with anisotropic mesh spacing, further improvements to the block-based AMR procedure are certainly possible by considering fully anisotropic mesh refinement strategies. Anisotropic mesh refinement may reduce further the cost of CFD simulations by allowing the creation of high aspect ratio cells aligned with the regions of the solution that require higher resolution in a preferred direction (e.g., for flow regions with thin shear and boundary layers and shocks) while still placing larger cells in regions with less strict resolution requirements without *a priori* knowledge of the solution and/or prescribed grid point clustering.

Three-dimensional anisotropic AMR methods have been considered in previous studies. Examples include the Cartesian 2^n tree[10,11] and the fully unstructured Immersed Boundary approach with conforming cells.[12,13] However, these methods can lead to complex data structures, additional overhead with unstructured computations and connectivity, large computational complexity when refining and coarsening (or de-refining) with unsteady flow problems, and difficulty scaling to large parallel simulations. It is for these reasons that a block-based approach to three-dimensional anisotropic AMR is considered here. A block-based anisotropic AMR scheme was proposed recently by Zhang and Groth[8,14] for the computation of two dimensional (2D) problems. Nevertheless, most complex flow problems are inherently three-dimensional (3D), thus the extension of the 2D algorithm to three dimensions is a natural evolution of the anisotropic approach and is addressed in this study.

I.B. Scope

A parallel, anisotropic, block-based, adaptive mesh refinement algorithm is therefore proposed for the solution of physically complex flow problems with disparate spatial scales exhibiting highly anisotropic features on three-dimensional, multi-block, body-fitted, hexahedral meshes. Block-based AMR is considered here over cell-based algorithms to allow local refinement while maintaining an efficient and highly scalable parallel implementation.[8,15–17] The body-fitted hexahedral grid blocks with unstructured root block topology and connectivity also support edge and corner degeneracies are used for the treatment of complex geometries which are generally encountered when solving the flow problems listed above. Traditional isotropic mesh refinement strategies have shown success in reducing computational complexity but further savings can be afforded by considering anisotropic mesh refinement. The proposed AMR scheme uses a binary tree hierarchical data structure to permit anisotropic refinement of the grid blocks in a preferred coordinate direction as dictated by appropriately selected physics-based refinement criteria. For the proposed algorithm to be efficiently applied to time-varying flow problems, the AMR procedure allows for mesh coarsening, in addition to refinement, while contributing little to the overall computational complexity of the problem. Furthermore, the anisotropic coarsening of the grid blocks is carried out in a fashion that is independent of the refinement history thereby permitting the mesh to rapidly adapt to the evolving unsteady flow. Overall, the proposed anisotropic AMR procedure allows for more efficient and accurate capturing of complex flow features such as shocks, boundary layers, and/or flame fronts. The AMR scheme is applied here in conjunction with an upwind finite-volume spatial discretization scheme to the solution of the Euler equations for inviscid compressible gaseous flow. Steady-state and time-varying flow problems are considered on anisotropic adapted meshes including the cubed-sphere.[18–20] The potential of anisotropic AMR for simulation of complex flows in an efficient and generalized manner is demonstrated.

II. Finite-Volume Scheme

As indicated above, the proposed anisotropic AMR scheme in this study is applied to the solution of the Euler equations governing compressible three-dimensional flows of a polytropic gas. The Euler equations are considered here for a first proof of concept of the proposed anisotropic AMR algorithm in three space dimensions and to illustrate its potential for the solution of more complex flow applications. Moreover, the algorithm presented in this paper has been implemented in a generalized way such that extensions to other systems of equations is rather straightforward.

II.A. Three-Dimensional Euler Equations

The conservation form of the Euler equations for a compressible polytropic gas can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = 0, \tag{1}$$

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

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} = 0, \tag{2}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{bmatrix}, \ \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \rho uh \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ \rho vh \end{bmatrix}, \ \mathbf{H} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ \rho wh \end{bmatrix}, \tag{3}$$

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 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.

II.B. Upwind Finite Volume Scheme

Application of a high-resolution Godunov-type[21] upwind finite-volume scheme to the integral form of the conservation equations given above applied to a hexahedral computational cell as shown in Figure 1a results in the following semi-discrete form:

$$\frac{d\mathbf{U}_{i,j,k}}{dt} = -\frac{1}{V_{i,j,k}} \sum_{m=1}^{N_f} \vec{\mathbf{F}}_{i,j,k,m} \cdot \vec{n}_{i,j,k,m} \, \Delta A_{i,j,k,m} = \mathbf{R}_{i,j,k}, \tag{4}$$

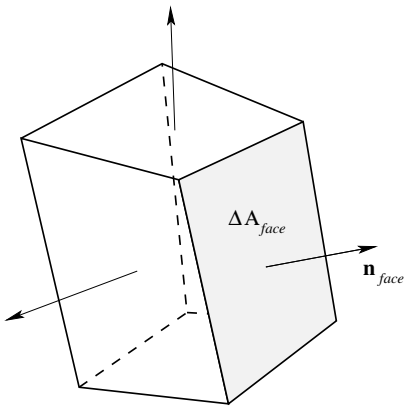
where $\mathbf{U}_{i,j,k}$ is the cell average of the conserved solution, $V_{i,j,k}$ is the volume of the cell, $\vec{\mathbf{F}}_{i,j,k,m}$ is the flux through the cell face, $\vec{n}_{i,j,k,m}$ is the outward pointing unit normal of a cell face, and $A_{i,j,k,m}$ is the area of a cell face. The variable N_f is the number of faces of the cell which for a hexahedral cell (as used in this work) is 6. The indices i, j, k correspond to a cell's computational coordinates within a structured mesh. Each structured mesh is contained inside a single block with that block being a part of a multi-block body-fitted mesh with general unstructured connectivity between blocks as shown in Figure 1b.

Limited linear least-squares solution reconstruction[22] and Riemann-solver based flux functions are used in the evaluation of the numerical flux integrals of Equation 4. The Roe linearized approximate Riemann solver[23] is used in the flux evaluation. Solutions of the semi-discrete form of the governing equations given in Equation 4, represented by the averaged solution quantities within each computational cell, $\mathbf{U}_{i,j,k}$, is obtained herein by applying a standard second-order accurate, Runge-Kutta, explicit time-marching scheme to the resulting coupled non-linear ordinary differential equations (ODEs). Steady-state solutions are obtained by advancing the solution in time until a converged time-invariant solution is achieved. While the latter is certainly non-optimal, and implicit time integration schemes may be preferred,[24–26] it proved sufficient for the purposes of the present work.

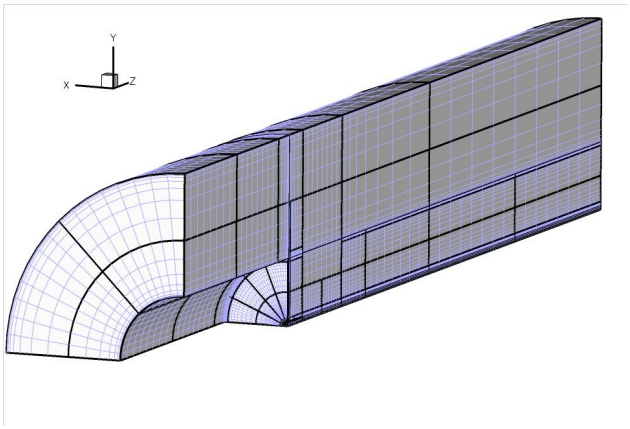
III. Anisotropic Adaptive Mesh Refinement

III.A. Block-Based Adaptive Mesh Refinement

The proposed scheme adopts a three-dimensional block-based approach to AMR as proposed by Groth and co-workers[15–17] for a flexible block-based AMR scheme allowing automatic solution-directed mesh adaptation



(a) Hexahedral cell at grid location i, j, k showing face normals.



(b) Body-fitted adapted mesh after several refinements. Grid blocks are shown with bold lines.

Figure 1: Three-dimensional hexahedral cell and example mesh.

on multi-block body-fitted (curvilinear) meshes consisting of quadrilateral (two-dimensional, 2D, case) and hexahedral computational cells (three-dimensional, 3D, case). The latter is shown in Figure 1a. This block-based approach has been shown to enable efficient and scalable parallel implementations for a variety of flow problems with anisotropic stretching. The stretching aids in the treatment of complex flow geometry and flows with thin boundary, shear, and mixing layers and/or discontinuities and shocks. Applications of the block-based AMR scheme have included laminar flames[5,27] and soot prediction,[25] turbulent non-premixed flames[6,27,28] as well as turbulent multi-phase rocket core flows,[7,29] magnetohydrodynamics (MHD) simulations,[1,4,20] and micron-scale flows.[9] Extensions of the block-based body-fitted AMR approach for embedded boundaries not aligned with the mesh[30] and with an anisotropic refinement strategy[14] are also possible and have been developed. Figure 1b illustrates the application of the block-based AMR technique to a body-fitted mesh.

In general, block-based AMR approaches tend to locally over resolve the domain in areas flagged for refinement, since every cell in a block is refined compared to individual cells in an alternative cell-based method. The advantage of using a block-based approach is that it requires an overall light data structure due to grid connectivity being computed on a block level as opposed to the cell level, which greatly simplifies nearest neighbor computation and solution communication. Solution communication is carried out at the block level, rather than between every cell. In addition, the domain of a block-based adaptively refined mesh is naturally decomposed, greatly simplifying the parallel implementation of the algorithm and leading to highly scalable solution methods.

In this work, the multi-block body-fitted mesh as shown in Figure 1b allows the successively refined grids to smoothly conform to the problem geometry. Each block contains a structured mesh with $N_i \times N_j \times N_k$ cells with two ghost cell layers that overlap with neighboring blocks for solution transfer between the blocks. The block dimensions, N_i , N_j , N_k are required to be even integers but do not need to be the same. At each time step or iteration level, blocks are independently solved and then ghost cell information is sent and received to or from neighboring blocks which may require solution prolongation or restriction depending on the refinement level difference between the adjacent blocks.

As with the isotropic block-based AMR schemes described above, block connectivity and refinement history in the proposed anisotropic AMR scheme is again stored within a hierarchical tree data structure allowing individual blocks to be refined or coarsened without having to compute an entirely new mesh over the whole domain. Each refinement produces new blocks called "children" from a "parent" block and the children can further refine creating new branches in the tree. An initial mesh can contain a single block or so-called "forest" of tree data structures each with their own connectivity and refinement history. Blocks that have been refined and exist in the binary tree as a history of the refinement are called "ancestors" of the leaf belonging to the same tree.

Block-based anisotropic AMR selectively refines and coarsens blocks based on refinement criteria for

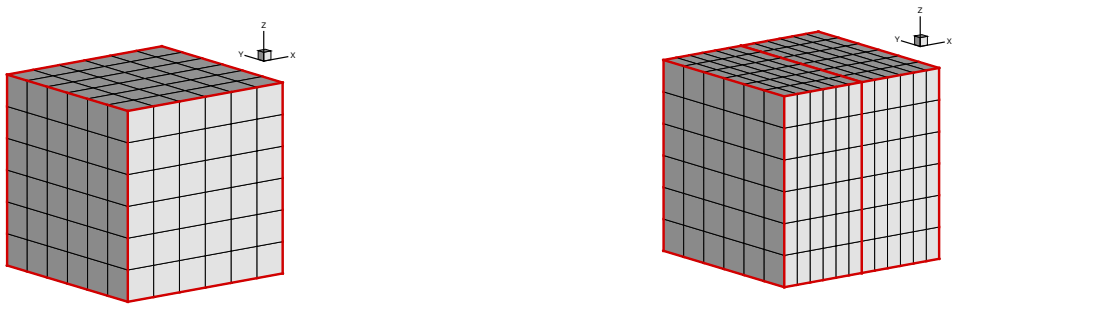
each computational direction. Anisotropic refinement, as shown in Figures 2a and 2b, doubles resolution in a single direction rather than only doubling refinement in all directions as in isotropic AMR. The proposed anisotropic AMR algorithm in this paper extends two-dimensional anisotropic AMR, previously considered by Zhang and Groth,[8,14] to three dimensions and follows the block-based framework described above.

III.B. Binary Tree Hierarchical Data Structure

Unlike the 3D isotropic AMR scheme that makes use of an octree data structure in which each refined block produces 8 children, the proposed anisotropic AMR applied to a three-dimensional body-fitted mesh uses a binary tree data structure with each block having just two children. Children are assigned a sector: west or east, south or north, bottom or top that correspond to a specific half of the parent block in the ξ, η, or ζ computational (logical) directions of that block. Each binary tree has a root which is assigned to one of the blocks created during initial mesh generation. As the AMR proceeds, branches are formed by each subsequent refinement and the leaves of the tree are the actively used blocks, having no children. However, the tree does not proceed purely in one direction since leaves may be removed and connectivity rearranged during the coarsening process. Every ancestor block in the tree is assigned a split direction which is associated with any of the three possible computational directions of that block and correspond to the direction in which the block was refined. Figure 3 shows the resulting binary tree after several refinements of an initial mesh consisting of a single block.

III.B.1. Neighbor Searching

An efficient neighbor searching algorithm is particularly critical for unsteady simulations where AMR needs to be performed frequently to adapt to unsteady flow features. In addition, to allow coarsening of blocks independently of the refinement history, the search algorithm must not rely on previous neighbor information. The neighbor searching algorithm in this work is based on the previous algorithm by Zhang and Groth for 2D anisotropic AMR[8,14] and 3D isotropic AMR by Gao *et al.*[6] as well as tree based searching by other authors.[31,32] Neighbor searching can be split into two parts: searches associated with ascending and descending the tree. First, a "bridge" must be found that links the branch of the searching block with that of the neighbor block. To find the bridge, the algorithm ascends the binary tree until a common ancestor in the search direction is found. If the ancestor is not found, there is either no neighbor in that direction or the neighbor is an ancestor of another root in which case the pre-computed root connectivity is used to jump to the neighboring tree. Once the bridge is found, it is descended by traversing the binary tree in the opposite direction and moving towards the sector of the searching block. This descending process ends when an intermediate block is reached that has the same or lower refinement level in a direction other than the search direction. For a face neighbor search, once the intermediate block is found, it is searched for leaves that are face neighbors to the searching block by descending the intermediate block's branches. For an edge or corner search, instead of descending the intermediate block, one or two more searches must be performed in the remaining search direction(s) from the first intermediate block. Following the additional searches, the resulting intermediate block can be descended in a similar manner to that of the face neighbor search. Care



(a) Coarse grid block with $8 \times 8 \times 8$ cells.

(b) Resulting blocks after anisotropic refinement in the x direction.

Figure 2: Example anisotropic refinement in one direction.

must be taken to ensure that the search moves toward the sectors of the first intermediate block and the searching block, making the implementation of the 3D anisotropic search significantly more involved than that of the original 2D anisotropic search implementation.

III.B.2. Refinement Criteria and Conflict Checking

In this work, a block is marked for refinement or coarsening by using physics-based refinement criteria. While refinement or coarsening criteria for AMR based on adjoint error estimation strategies, such as the proposed method by Venditti and Darmofal in one and two dimensions[33,34] and for anisotropic refinement,[35,36] as well as *hp* adaptivity by Heuveline and Rannacher[37] and *a priori* error estimation by Rannacher and Vexler,[38] are expected to be more optimal, the refinement criteria here are restricted to a gradient-based approaches for their relative simplicity. For the Euler equations, possible refinement criteria include the density gradient, compressibility, and vorticity in each direction. For example, the refinement criteria for the density gradient are[8]

$$\epsilon_\xi = \frac{1}{\rho}\left(\vec{\nabla}\rho \cdot \Delta\tilde{\mathbf{X}}\right), \quad \epsilon_\eta = \frac{1}{\rho}\left(\vec{\nabla}\rho \cdot \Delta\tilde{\mathbf{Y}}\right), \quad \epsilon_\zeta = \frac{1}{\rho}\left(\vec{\nabla}\rho \cdot \Delta\tilde{\mathbf{Z}}\right), \tag{5}$$

where ρ is the density, $\Delta\tilde{\mathbf{X}}$, $\Delta\tilde{\mathbf{Y}}$ 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. Refinement criteria for each cell is calculated and the maximum value over all cells in a particular computational direction is assigned as the block's refinement criteria. A block is flagged to coarsen or refine based on user chosen values for the maximums and minimums of ϵ_ξ, ϵ_η and ϵ_ζ.

The conflict checking procedure modifies the refinement flags in order to remove refinement or coarsening scenarios that are not allowed. There are three scenarios that are not allowed in the 3D anisotropic AMR algorithm:

1. adjacent blocks having a resolution change greater than two;
2. blocks flagged to coarsen that do not have a sibling/neighbor flagged to coarsen or after refinement the sibling/neighbor is not at the same refinement level; or
3. an invalid binary tree connectivity would result from refinement or coarsening.

Where siblings are defined as actively used blocks that were created from the same parent block. Each one of the conflicts above is checked for in an iterative conflict checking procedure based on the 2D anisotropic AMR procedure by Zhang and Groth.[8,14] The procedure is split into four parts: (*i*) first sibling checks; (*ii*) level checks; (*iii*) second sibling checks; and, (*iv*) invalid tree checks.

First sibling checks simply make sure that all blocks that have been flagged to coarsen have a neighbor in the coarsening direction that is also flagged to coarsen and is at the same refinement level in that direction. In addition, if the neighbor is at the same level in the direction of coarsening but coarser in the other direction(s), the neighbors in the other direction(s) must also be at the same refinement level in the coarsening direction.

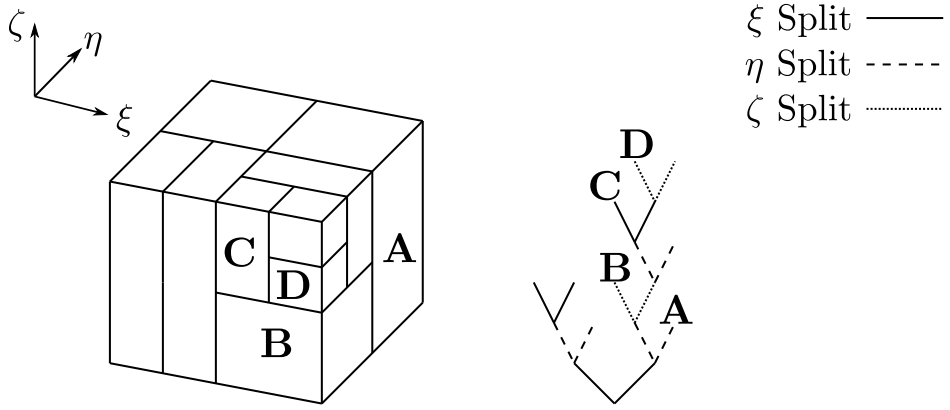


Figure 3: 3D binary tree and the corresponding blocks after several refinements

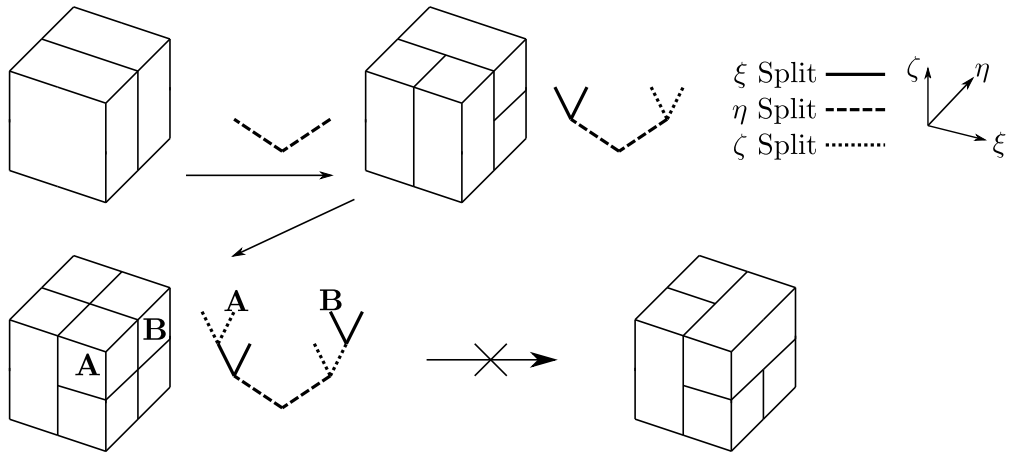


Figure 4: Sequence of refinements that create a mesh that cannot be represented by the 3D binary tree data structure.

If either of these checks are violated, then the block is flagged to not change its refinement level. The process is repeated for every block that is flagged to coarsen.

Next, level checks are performed to ensure that no adjacent block has a refinement level difference greater than two. The checks are performed sequentially for the ξ, η, and ζ directions with each one comparing resolution changes in all 26 neighboring directions if the block is flagged to refine or coarsen. First, if a resolution change of 3 would result, both the block and its neighbor are forced to not change their refinement level. Otherwise, if a block flagged to coarsen conflicts with a neighbor, its refinement flag is set to not change the refinement level. If instead a block flagged to not change its refinement level conflicts with a neighbor that is flagged to coarsen, the block is forced to refine. This process is again repeated for each block used in the simulation.

The second sibling check ensures that, following the coarsening of a block, its siblings in the direction other than the coarsening direction are at the same level. First, the levels not in the direction of coarsening of two blocks flagged to coarsen are compared. Refinement is favored over coarsening and thus blocks that require refinement to be at equal level in the directions other than the coarsening direction are flagged to refine while in the opposite case where coarsening is required, the blocks are flagged to not change their refinement level.

Finally, cases where an invalid binary tree connectivity would result from refinement or coarsening of the grid blocks are eliminated. An example of an invalid series of adaptive mesh refinements is shown in Figure 4. Following the last refinement, if the coarsening of blocks A and B is attempted, there is no way to rearrange the connectivity of the binary tree to reflect the change in the mesh. These cases are identified and then eliminated by changing the refinement flags of the blocks involved in the conflict from coarsen to no change. Identifying the block coarsening cases that create an invalid binary tree is done by:

1. ascending the tree until an ancestor that is split in the coarsening direction is found;

2. descending the ancestor block and determining the lowest refinement level of the active blocks in the directions other than the coarsening direction;

3. checking if the lowest refinement levels of the active blocks are less than the corresponding refinement levels of the coarsening block; and,

4. if both refinement levels are lower than the coarsening block's refinement levels, do not allow it to coarsen.

The first sibling checks and the level checks are repeated until there are no changes in the refinement flags. The second sibling checks as well as the checks for invalid trees are performed and then the first sibling checks and the level checks are converged again. The procedure ends when the refinement flags for all grid blocks cease to change.

III.C. Refinement and Coarsening of Grid Blocks

Ordering of the anisotropic refinement and coarsening procedures is important to ensure that no violations in adjacent block resolution change occur and that coarsening is allowed to occur in an independent manner of the refinement history. The proposed procedure is as follows:

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 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 found and stored for each block; and
6. solution and geometry information is shared between adjacent blocks through message passing procedures.

While the connectivity rearrangement procedure above requires accurate neighbor information, neighbor information is destroyed by choosing to refine blocks before coarsening. This is in contrast to the 2D anisotropic refinement and coarsening procedure where connectivity rearrangement for all blocks that require it is completed before coarsening and refinement. Adopting a similar procedure for 3D anisotropic AMR would be much more involved since three possible coarsening directions make multiple connectivity rearrangements necessary for coarsening in more than one direction and, furthermore, the special case of connectivity rearrangement outlined in Section III.C.1 would destroy the connectivity rearrangements performed earlier. To avoid rearranging the connectivity of all blocks at once, connectivity rearrangement is only performed while coarsening and the required neighbor information is computed only for the blocks that are being coarsened.

Refined node locations are computed from the coarse nodes in the parent mesh. Figure 5 shows the computation of fine cells from coarse cells in one computational direction. Fine cells can be computed by averaging the coarse node locations composing an edge, face, or a volume of interest or they can be computed more accurately by using grid metrics which map the physical node locations with those in a uniform computational mesh. A second-order Taylor series expansion is used for the computation of the metrics shown in Equation 6, where $\vec{x}(\xi, \eta, \zeta)$ is the physical node location mapped from the computational location (ξ, η, ζ).

$$\begin{aligned}\vec{x}(\xi+\Delta\xi, \eta+\Delta\eta, \zeta+\Delta\zeta) = \vec{x}(\xi,\eta,\zeta) &+ \left.\frac{\partial \vec{x}}{\partial \xi}\right|_{\xi,\eta,\zeta}\Delta\xi + \left.\frac{\partial \vec{x}}{\partial \eta}\right|_{\xi,\eta,\zeta}\Delta\eta + \left.\frac{\partial \vec{x}}{\partial \zeta}\right|_{\xi,\eta,\zeta}\Delta\zeta \\ &+ \frac{1}{2}\left(\left.\frac{\partial^2 \vec{x}}{\partial \xi^2}\right|_{\xi,\eta,\zeta}(\Delta\xi)^2 + 2\left.\frac{\partial^2 \vec{x}}{\partial \xi \partial \eta}\right|_{\xi,\eta,\zeta}\Delta\xi\Delta\eta + \left.\frac{\partial^2 \vec{x}}{\partial \eta^2}\right|_{\xi,\eta,\zeta}(\Delta\eta)^2\right. \\ &\left. + 2\left.\frac{\partial^2 \vec{x}}{\partial \xi \partial \zeta}\right|_{\xi,\eta,\zeta}\Delta\xi\Delta\zeta + \left.\frac{\partial^2 \vec{x}}{\partial \zeta^2}\right|_{\xi,\eta,\zeta}(\Delta\zeta)^2 + 2\left.\frac{\partial^2 \vec{x}}{\partial \eta \partial \zeta}\right|_{\xi,\eta,\zeta}\Delta\eta\Delta\zeta\right) \\ &+ \mathcal{O}\left((\Delta\xi)^3, (\Delta\eta)^3, (\Delta\zeta)^3\right) \quad (6)\end{aligned}$$

The first and second order partial derivatives are computed using second order centered differences for the interior nodes and third order forward or backward differences for the nodes residing on the block boundaries. The fine node locations are then computed by averaging the two locations produced by Equation 6 for each edge. For refinements in two directions, where new nodes at the cell faces are required, the node location at the face center is approximated by averaging Taylor expansions from the four coarse nodes that define the face. Similarly, for refinements in three directions, nodes in the cell volume are computed by averaging Taylor expansions from the eight coarse nodes that define the coarse cell and are averaged. Coarsening of the mesh is done by removing every second node from the fine mesh in the required computational direction(s).

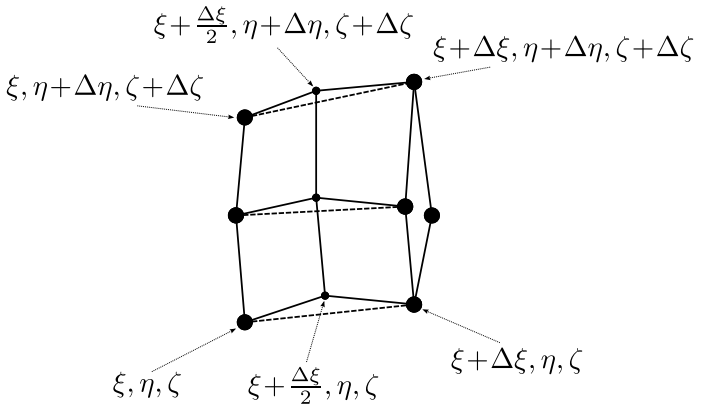


Figure 5: Anisotropically refining a cell using grid metrics in the ξ computational direction. Fine node locations are indicated with small dots and coarse node locations with large dots.

At the block level, refinement takes the parent block and gives it to the child which has a west, south, or bottom sector. The remaining child is given a block taken from a pool of available unused blocks which can come from any processor used in the simulation. For coarsening, the west, south, or bottom sector child block is given back to its parent while the east, north, or top sector child block is placed back into the pool of available resources.

To allow for refinements in multiple directions in one AMR sequence, e.g., refinement in two or three computational directions, the grid blocks are refined and the solution is prolonged in one step rather than the sequential approach proposed by Zhang and Groth.[8,14] The two-step approach is computationally more expensive and leads to unnecessary errors in the solution prolongation. Since refinement happens in one step and the binary tree datastructure naturally supports only two children for each block, the binary tree is grown as if the one step refinement had actually occurred as a sequential refinement. For example, following a block refinement in the ξ and η computational directions, the binary tree first creates children in ξ and then branches from those children are created in η. The four new children are each assigned a pointer to the solution and grid blocks created previously based on the sectors of the blocks.

III.C.1. Connectivity Rearrangement

To maintain block connectivity, coarsening of grid blocks logically requires both leaves of the tree to share the same parent block, effectively reversing a previous refinement. While this process works well for isotropic AMR, it is too restrictive for anisotropic AMR. The standard isotropic AMR coarsening strategy would lead to a deadlock situation in the AMR where areas of the mesh remain highly refined even though the solution has evolved and coarsening or de-refinement is called for in a direction other than the one in which they were originally refined.

In order to both maintain the block connectivity and to ensure that blocks can coarsen independently of their previous refinement, Zhang and Groth[8,14] suggest altering the connectivity of the binary tree to force leaves to share the same parent even if they were created by a different refinement sequence. Connectivity rearrangement works by first identifying a common ancestor with the desired split, called the bridge. If the grandparent of both blocks is not the bridge, then the function is called recursively with the block's parents until the bridge is found. Once the bridge is found, the split directions of the bridge and its children are swapped, then the children and grandchildren split directions are swapped, and on until the desired split is achieved. An example of this process is shown in Figure 6 where blocks A and B do not share the same parent until connectivity rearrangement is performed.

A key difference in connectivity rearrangement between 3D and 2D versions of the anisotropic AMR algorithm is that the parents of blocks undergoing connectivity rearrangement are not guaranteed to be at the same refinement levels due to the complex refinement history created when considering the additional third refinement direction. An example of this connectivity rearrangement problem is illustrated in Figure 7. Blocks **A** and **B** are flagged to coarsen in ξ but their parents are at different refinement levels in the η and ζ directions causing the connectivity rearrangement to fail. A solution to this problem is to introduce an

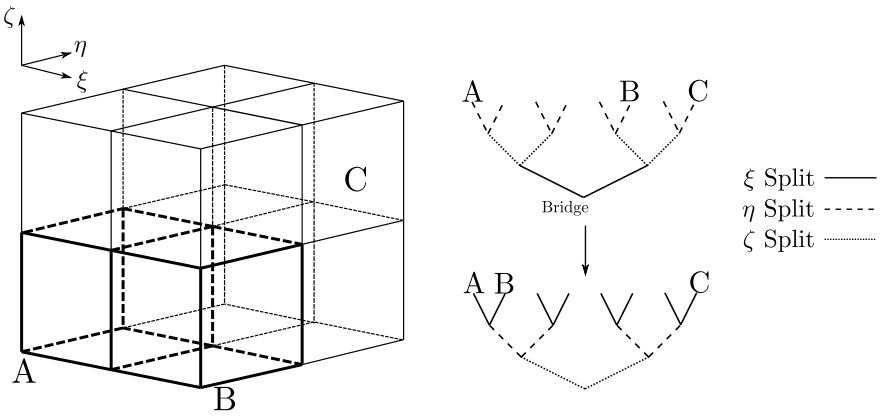


Figure 6: Left: blocks **A** and **B** need to coarsen in the ξ direction but do not share a common parent. Right: binary trees showing the connectivity rearrangement required to force **A** and **B** to share a common parent.

intermediate connectivity rearrangement that rearranges the connectivity of block **A** or **B** with its neighbor before rearranging the connectivity of blocks **A** and **B**. This forces the parents of blocks **A** and **B** to have the same refinement level so that **A** and **B** can rearrange to become siblings. In practice, more than one intermediate rearrangement can be required during a connectivity rearrangement and can be significantly more complicated. The intermediate rearrangement process is generalized as follows:

1. ascend the tree from block **A** until a bridge is found with the same split as **B**'s parent;
2. cross the bridge and descend the tree toward block **A** until the descended block 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** instead of block **A**;
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.

While the entire connectivity rearrangement procedure is rather complex, it adds little to the computational complexity of the problem since there are no floating point operations involved and it is only used during the coarsening procedure for those blocks that require it.

III.D. Solution Transfer

Following grid refinement, the solution information in the original coarse block needs to be transferred to the newly created fine children. This coarse to fine solution transfer (prolongation) is done either by direct injection, where the cell averaged solution in the coarse cells is assigned to the cell averages of the the fine children, or by a piecewise linear method where the solution is reconstructed in the coarse cells and the solution gradients are then used to determine the solution at the centroids of the fine cells. Direct injection is typically used for the steady-state problems where the transient solution is not important while the piecewise linear method is used for time-accurate problems to accurately capture the unsteady behavior.

The solution transfer methods previously described are sufficient for the second order accurate scheme used in this work but for higher order schemes it will be insufficient and a prolongation method that preserves the higher order accuracy will need to be used. After coarsening, the solution information in the fine blocks needs to be transferred to the coarse block (restriction). The restriction process is done by using a volume weighted average of the cell-averaged solution in the fine cells:

$$\mathbf{U}_C = \frac{1}{V_C} \sum_{n=1}^{N_F} \mathbf{U}_n V_n, \tag{7}$$

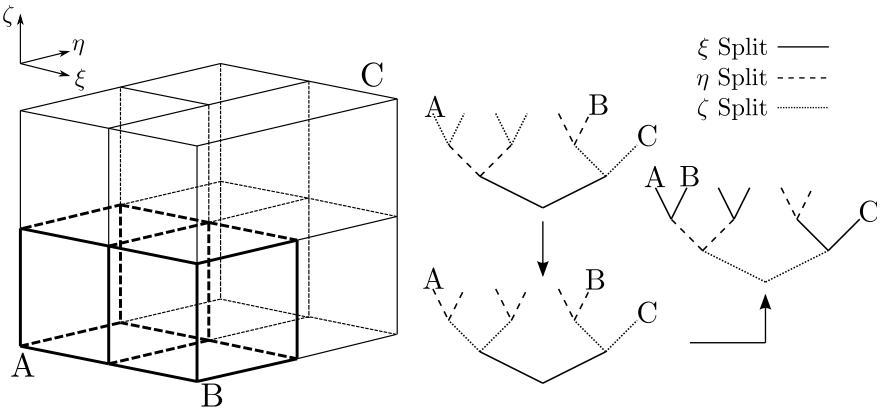


Figure 7: Blocks A and B need to rearrange connectivity but their parents are at different levels, causing the rearrangement to fail.

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

III.E. Inter-Block Exchange of Solution Information

Each solution block uses two layers of ghost cells that overlap the neighboring block to facilitate transfer of solution information between adjacent blocks. In the standard 3D isotropic AMR, there are only three possible message passing scenarios: no resolution change, coarse to fine resolution change, or fine to coarse resolution change for each of the 26 faces, edges, or corners. For 3D anisotropic AMR, the number of possible message passing scenarios increases to 27 for each of the 26 faces, edges, or corners. These extra scenarios arise by allowing a block to have any refinement level in each computational direction provided that there are no resolution changes between adjacent blocks in any of the computational directions that are greater than two. As a result of the significantly increased message passing scenarios, a more generalized implementation of the solution exchange to capture all possible scenarios is required.

For adjacent grid blocks with isotropic resolution changes, message passing for anisotropic AMR works in a similar way to that of isotropic AMR. For no resolution change, the solution is directly assigned to the neighbor's ghost cells. For coarse to fine or fine to coarse resolution changes, solution information is prolonged or restricted in the same way as with the solution transfer to interior cells described in Section III.D. Because anisotropic AMR has the freedom to allow resolution changes in each refinement direction, prolongation or restriction in one direction or multiple directions and a combination of prolonging and restricting may be required to fill neighboring ghost cells. For example, sending solution information from a block at a refinement level in (ξ, η, ζ) of $(1, 0, 1)$ to an adjacent block with a refinement level of $(0, 1, 0)$ is shown in Figure 8. Prolongation is completed first followed by restriction.

In addition to the transfer of solution information to neighboring ghost cells, the flux must be corrected at cell interfaces with resolution changes across the interface to ensure the conservation properties of the finite-volume scheme.[39, 40] For resolution changes across cell interfaces in only one of the two tangential computational directions, flux corrections are done in a straightforward way by correcting the flux through the coarse cell face by the difference between the coarse flux and the fine fluxes. The residual in the coarse cell is then updated with the flux change using

$$\vec{\mathbf{R}}_{i,j,k} = \vec{\mathbf{R}}_{i,j,k} - \frac{A_{\text{coarse}} \Delta\vec{\mathbf{F}}}{V_{i,j,k}}, \tag{8}$$

where $\vec{\mathbf{R}}_{i,j,k}$ is the solution residual (RHS of Equation 4), $V_{i,j,k}$ 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 resolution changes in each of the tangential computational directions to the cell faces, an averaged flux correction is used based on the combined area of the neighboring cells as illustrated in Figure 9. An average flux can be defined at the cell faces as

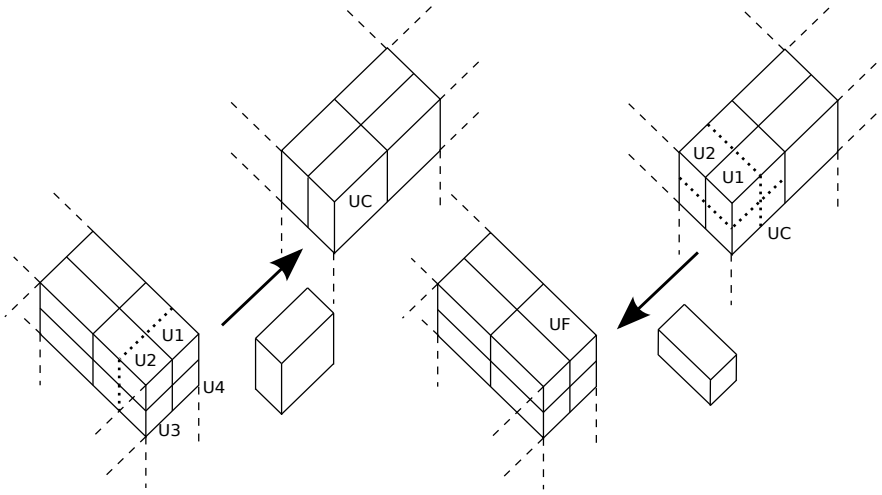


Figure 8: Left: Interior cells are divided into two sub-cells and then restricted and assigned to the coarse ghost cell, UC, in the neighboring block. Right: Interior cells are divided into four sub-cells and then pairs are restricted and assigned to the fine ghost cell, UF, in the neighboring block.

$$\vec{\mathbf{F}}_{\text{Avg}} = \frac{\vec{\mathbf{F}}_A + \vec{\mathbf{F}}_B}{2}, \tag{9}$$

and then the flux corrections are

$$\Delta\vec{\mathbf{F}}_A = \vec{\mathbf{F}}_{\text{Avg}} - \vec{\mathbf{F}}_A \text{ and } \Delta\vec{\mathbf{F}}_B = \vec{\mathbf{F}}_{\text{Avg}} - \vec{\mathbf{F}}_B. \tag{10}$$

Updating the residual is identical to the single resolution change case in Equation 8 but with $\Delta\vec{\mathbf{F}}$ being replaced by $\Delta\vec{\mathbf{F}}_1$, $\Delta\vec{\mathbf{F}}_2$, $\Delta\vec{\mathbf{F}}_3$, or $\Delta\vec{\mathbf{F}}_4$ and A_{Coarse} by the area of the face of the cell whose flux is being corrected. The error introduced by this average flux correction has yet to be fully quantified. Another possible solution would be to prohibit cases where there are greater than one tangential refinement level difference between adjacent solution blocks. While not allowing these cases has been shown to work well for cell-based 3D anisotropic AMR approaches,[12] in the proposed block-based anisotropic AMR it significantly reduces the effectiveness of anisotropic refinement. It was found to have the effect of not only increasing local mesh refinement levels but also neighboring areas of the mesh in order to maintain refinement level constraints between adjacent blocks.

III.F. Parallel Implementation

An efficient parallel implementation of anisotropic AMR is required to take advantage of the block-based approach. This work uses the message passing library, Message Passing Interface (MPI) for the task of

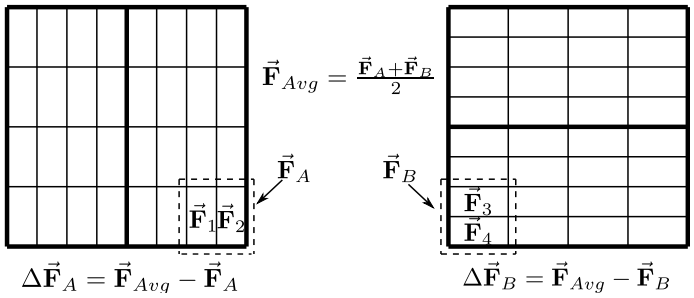


Figure 9: Flux correction procedure for resolution changes in two directions across the block interfaces. The dotted line shows the corresponding cell faces used in the correction.

communicating solution information across processors. Since cells are already grouped into equal sized blocks, the domain is naturally decomposed and the blocks are distributed equally across all processors without needing a computationally expensive domain decomposition algorithm. Increased message passing performance can be improved by using load balancing to distribute blocks based on the speed of each processor and ordering the blocks such that nearest neighbors are on the same processor.[6,17,25,27] However, these additional strategies are not considered in this research.

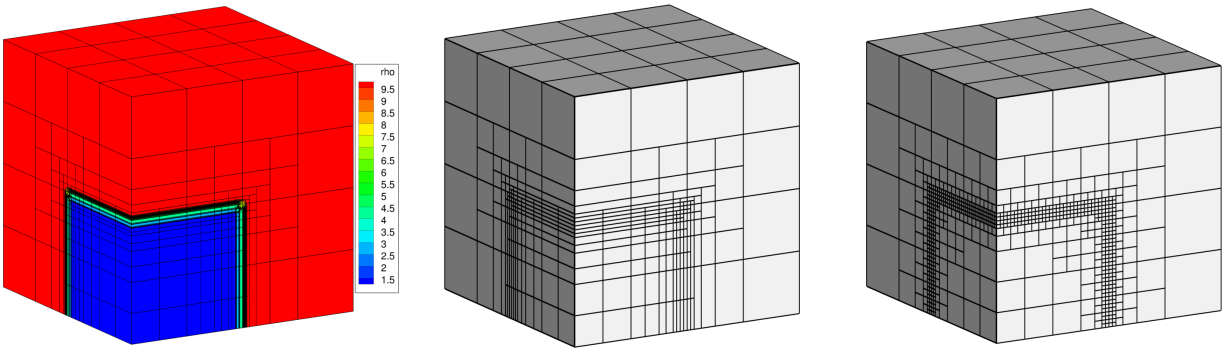
IV. Results

Results for several flow problems will now be discussed. Numerical results are described for an unsteady three-dimensional shock-cube problem and steady-state supersonic channel flow with a bump. Results using a cubed-sphere mesh[18–20] are presented for a steady-state supersonic radial flow and steady-state supersonic sphere. An accuracy assessment is performed for the supersonic radial flow problem. The flow problems have been chosen to show the potential of the proposed 3D anisotropic AMR algorithm for significantly reducing computational complexity and the validity of the algorithm for solving both steady-state and time-varying problems. Comparing the results for anisotropic AMR with 3D isotropic AMR and 2D anisotropic AMR illustrates the potential of the AMR scheme for reducing computational costs.

IV.A. Unsteady Shock Cube

The shock cube flow problem provides an excellent evaluation of AMR algorithms, requiring high mesh resolution to capture the wave interactions and fast adaptation to the unsteady solution. Due to the highly anisotropic nature of the shock cube solution, it is a good evaluation of 3D anisotropic AMR. The initial conditions for the shock-cube problem are given as standard atmospheric conditions ($\rho = 1.225$ kg/m$^3, p =$ 101.325 kPa, $\vec{V} = 0$ m/s, $\gamma = 1.4$) for $x < 0$, $y < 0$, and $z < 0$ and eight times standard atmospheric density and ten times standard atmospheric pressure elsewhere. Reflection boundary conditions are imposed on all walls of the cube. Before the simulation begins, the initial mesh with a single block containing $8 \times 8 \times 8$ cells is refined based on the initial conditions six times in order to resolve the discontinuity between the two states and assigned a maximum refinement level of seven. Following the initial refinements, the anisotropic AMR refinement and coarsening procedure is repeated every 15 time steps until the maximum times shown in Figure 10.

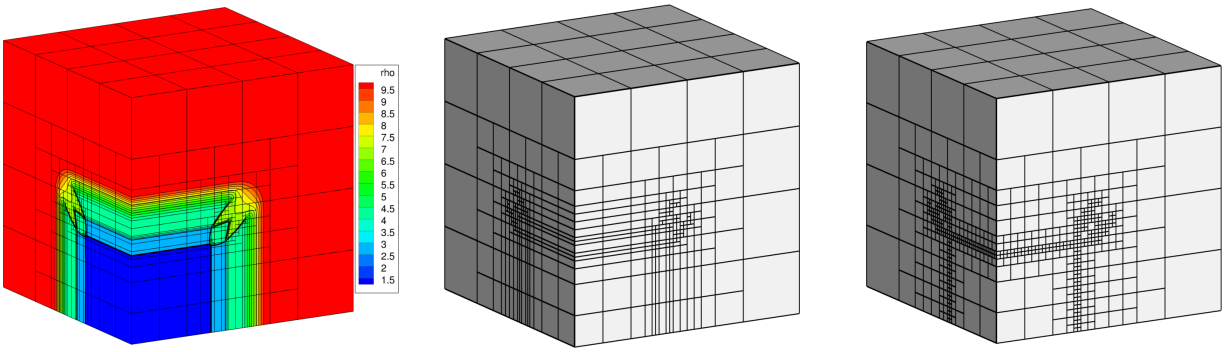
Throughout the simulation, the anisotropic and isotropic AMR show similar spatial resolution and track the solution well. At $t = 0.05$ ms, there are large anisotropic features in the solution which are exploited by anisotropic AMR using blocks with high aspect ratio cells and with an overall reduction of 89% in the number of cells compared to that of the usual 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 the anisotropic AMR algorithm to produce a more isotropically refined mesh. 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 of 53% on the mesh.



(a) Anisotropic AMR solution at $t = 0.05$ ms

(b) Anisotropic AMR $802,816$ cells

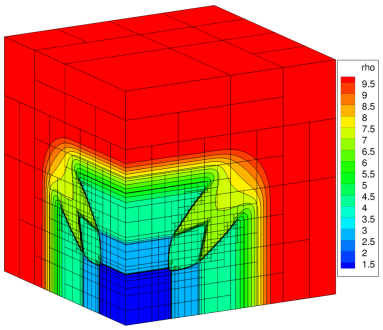
(c) Isotropic AMR $7,620,096$ cells



(d) Anisotropic AMR solution at $t = 0.25$ ms

(e) Anisotropic AMR $1,323,520$ cells

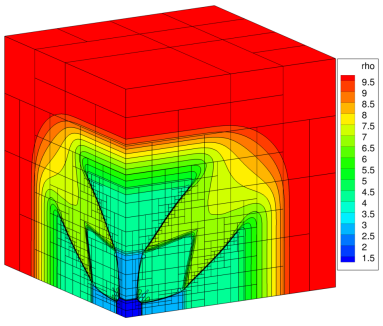
(f) Isotropic AMR $6,512,640$ cells



(g) Anisotropic AMR solution at $t = 0.50$ ms

(h) Anisotropic AMR $2,684,928$ cells

(i) Isotropic AMR $7,795,712$ cells

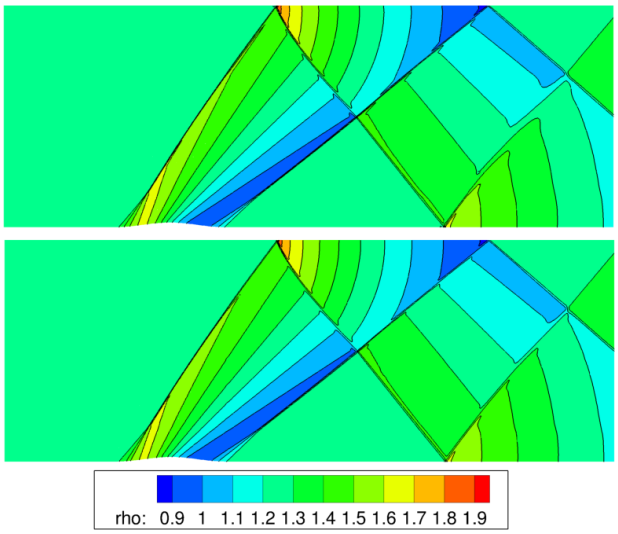


(j) Anisotropic AMR solution at $t = 0.75$ ms

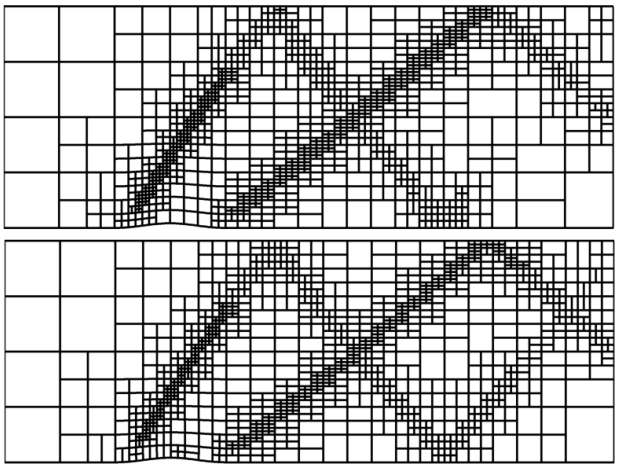
(k) Anisotropic AMR $3,079,168$ cells

(l) Isotropic AMR $6,580,736$ cells

Figure 10: 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.



(a) Density contours in kg/m^3 for $M = 1.4$ flow over a smooth bump in a channel. Top: 2D Anisotropic AMR. Bottom: 3D Anisotropic AMR



(b) Blocks shown for $M = 1.4$ flow over a smooth bump in a channel. Top: 2D Anisotropic AMR with 1222 blocks. Bottom: 3D Anisotropic AMR with 1215 blocks.

Figure 11: Supersonic, $M = 1.4$ flow over a smooth bump in a channel.

IV.B. Steady Supersonic Channel Flow Over a Bump

To demonstrate the applicability of 3D anisotropic AMR to steady-state problems and to compare to 2D anisotropic AMR, flow over a smooth bump in a channel has been performed. The particular problem consists of a rectangular cross section channel with a smoothed circular bump. Air at $p = 101.325$ kPa, $\rho = 1.225$ kg/m^3, $M = 1.4$, $\gamma = 1.4$ enters the channel through the left boundary and exits through the right boundary.

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 optimal smoothing scheme[41] with local time stepping and a CFL of 0.5 is used for the time integration.

Six anisotropic adaptive mesh refinements are performed with the final mesh containing $1,215$ blocks shown at the bottom in Figure 11b. After undergoing six mesh refinements, the number of blocks in the out of plane direction remains one, which is an expected result considering the two-dimensionality of the flow. Solving this highly 2D problem with isotropic AMR would require an excessive additional amount of cells to achieve a similar spatial resolution since all refinements would be carried out in an isotropic fashion.

Comparing the refined mesh to the solution density contours in Figure 11a for 3D anisotropic AMR, it is clear that the mesh is adapted well to the shock waves in the converged solution. The solution to the same problem in two dimensions using 2D anisotropic AMR[8,14] is given in Figure 11a with the refined mesh shown in Figure 11b. Both 3D and 2D anisotropic AMR achieve similar spatial resolution over the majority of the flow field and the density contours are in agreement.

IV.C. Steady Supersonic Spherical Outflow

A supersonic spherical outflow problem is considered here to provide assessment of the accuracy of the proposed anisotropic AMR algorithm on the cubed-sphere in comparison to both isotropic AMR and uniform meshes. The flow problem domain consists of an inner sphere with radius $R_i = 1$ and an outer sphere of radius $R_o = 4$. Gas enters the domain through the inner sphere supersonically, expands, and then exits the outer sphere supersonically. Only the radial velocity, V_r, is non-zero. The analytical solution to this flow

problem is described by Ivan[20] as the numerical solution at any radial location, r, to the equation

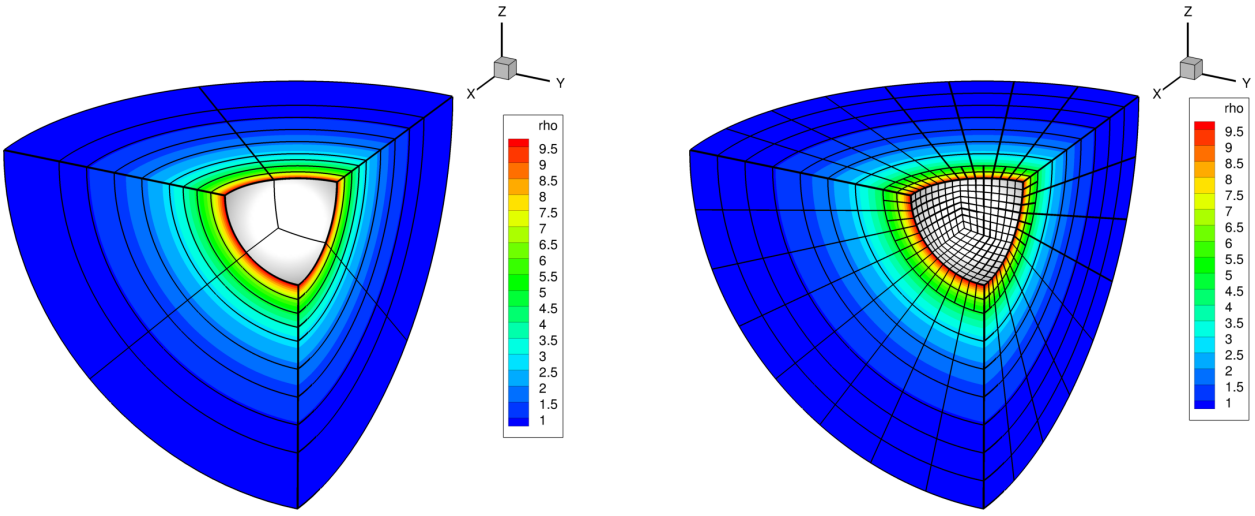
$$C_3 - \frac{1}{r^2 V_r \left[(C_2 - V_r^2)^{\frac{1}{\gamma-1}} \right]} = 0, \tag{11}$$

where

$$C_3 = \frac{1}{\left(\frac{2\gamma}{\gamma-1} \frac{p_i}{\rho_i}^{\frac{1}{\gamma-1}} \right) R_i^2 V_{r,i}} \text{ and } C_2 = \frac{2\gamma}{\gamma-1} \frac{p_i}{\rho_i} + V_{r,i}^2. \tag{12}$$

The initial mesh consists of 24 blocks with $32 \times 32 \times 16$ cells in each of the blocks. The inner sphere boundary condition is fixed at $\rho_i = 10.0$, $V_{r,i} = 4.5$, and $p_i = 26.0$. Linear extrapolation is used for the outer sphere boundary condition since the flow exits supersonically. The mesh undergoes 3 adaptive mesh refinements and is converged to machine zero using the multi-stage optimal smoothing scheme[41] with local time stepping. The solution density and Mach number contours are shown for a quarter slice of the domain in Figures 12 and 13 for anisotropic and isotropic AMR respectively. As expected, anisotropic AMR refines only in the radial direction where the largest gradient in density occurs which leads to savings in cell count of over 96% compared to isotropic AMR.

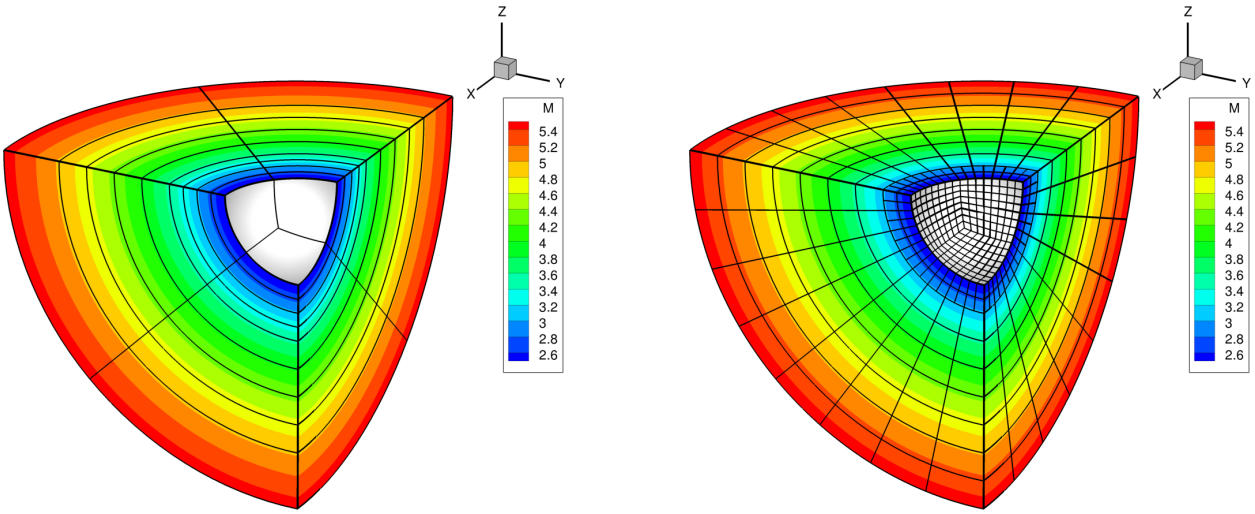
Accuracy is assessed by comparing the L_1, L_2, and L_∞ density error norms on successively refined meshes. Results of the accuracy assessment are shown in Figure 14 for the initial mesh and 3 levels of anisotropic and isotropic AMR. The error reduction on the uniform mesh achieves the expected second-order accuracy. Isotropic AMR error reduction is on-par or better than that of the uniform mesh and it is expected that increasing the number of mesh refinements will allow it to perform better. The reduction of error using anisotropic AMR is shown to be much greater than that of isotropic AMR and uniform refinement. Although, anisotropic AMR shows much greater performance than isotropic AMR for this accuracy assessment, increasing the number of refinements past 3, the L_1, L_2, and L_∞ error norms start to level off. This levelling off behavior is believed to be caused by the limited capabilities of the gradient based refinement criteria used in this work. Since the gradient of density is much larger in the radial direction than all other directions, anisotropic AMR only refines in the radial direction. As the refinement in the radial direction increases, the error caused by ignoring refinement in the angular directions begins to dominate, polluting the overall accuracy. This is illustrated in Figure 14 by the slope of the L_1, L_2, and L_∞ error norms for anisotropic AMR decreasing as the refinement level increases. A better refinement indicator for this problem would be based on error estimation that takes into account actual solution errors[33–38] instead of only the largest gradients in solution variables. The use of physics-based refinement criteria is a clear limitation of the present approach.



(a) Anisotropic AMR with 442,368 cells.

(b) Isotropic AMR with 11,796,480 cells.

Figure 12: Quarter slices of the solution showing density and block outlines for anisotropic and isotropic AMR.



(a) Anisotropic AMR with 442,368 cells.

(b) Isotropic AMR with 11,796,480 cells.

Figure 13: Quarter slices of the solution showing Mach number and block outlines for anisotropic and isotropic AMR.

IV.D. Steady Supersonic Flow Past a Sphere

The supersonic radial outflow problem considered above only offers a limited evaluation of the capabilities of anisotropic AMR on the cubed-sphere due to the flow problem's inherent spherical symmetry. Therefore, a solution of supersonic flow over a sphere is presented as both a test of the performance of anisotropic AMR on the cubed-sphere mesh for a fully three-dimensional problem as well as an assessment of the computational savings as compared to solving the same problem with isotropic AMR.

The geometry consists of a half sphere of radius of 1 m with a far-field boundary at 10 m shown in Figure 15. The cubed-sphere mesh for this case has only 5 sectors to reduce the problem size and 3 blocks stacked in the radial direction for each of the five sectors forming a total of 15 initial blocks with $10 \times 10 \times 10$ cells each. Free-stream air at $p = 101.325$ kPa, $\rho = 1.225$ kg/m^3, $M = 2.0$, and $\gamma = 1.4$ enters through the far-field in the positive x direction and exits through the outflow boundary. The far-field boundary condition is fixed at the inflow conditions and the outflow boundary condition is imposed as linear extrapolation since the air exits supersonically. A reflection boundary condition is imposed on the surface of the inner sphere. The multi-stage optimal smoothing scheme[41] with local time stepping is used for integrating the governing equations and the HLLE approximate Riemann solver[42] is used for the flux evaluation at cell boundaries.

Figures 16a and 16b show the solution density and Mach number contours after 5 refinements for anisotropic AMR and isotropic AMR respectively. For both cases, the adapted mesh is only well aligned with the bow shock near the stagnation point and then rapidly loses alignment moving toward the outer regions of the bow shock. Nevertheless, anisotropic AMR still reduces the total number of cells compared to isotropic AMR by 91%. These significant savings are more apparent by looking at an axial slice of the mesh for anisotropic and isotropic AMR in Figures 17a and 17b showing far less refined blocks in the angular directions for anisotropic AMR.

To compare the performance of 3D anisotropic AMR with 2D anisotropic AMR, a 2D axisymmetric simulation of the supersonic sphere flow problem is presented. Referring to Figures 18a for the 2D simulation and 16a for the 3D simulation, both 2D and 3D anisotropic AMR show agreeable refinements throughout the domain. A comparison of the center line density and Mach number from the stagnation point to the free-stream is given in Figure 18b. Both methods predict very a similar flow field including the shock location and post-shock stagnation conditions.

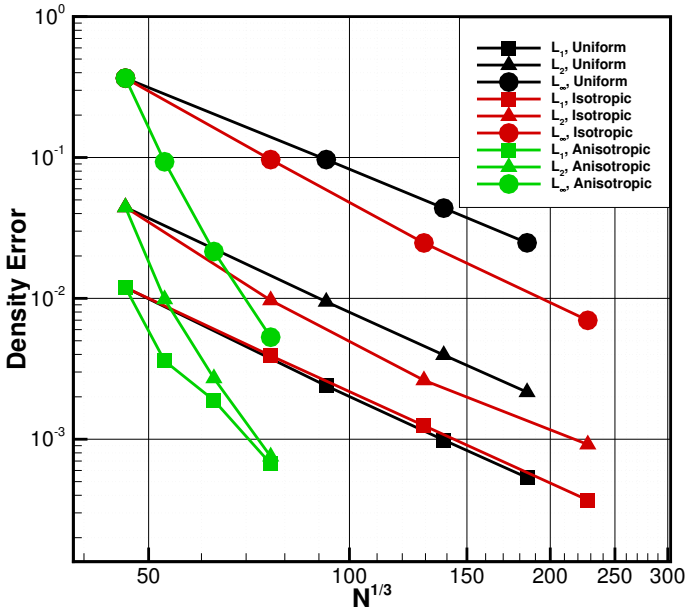


Figure 14: Accuracy assessment based on the exact solution density of the supersonic radial outflow problem comparing anisotropic AMR, isotropic AMR, and the solution on a uniformly refined mesh.

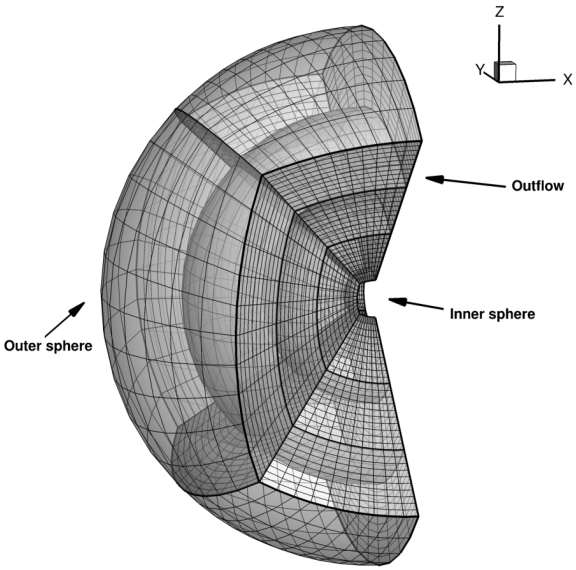
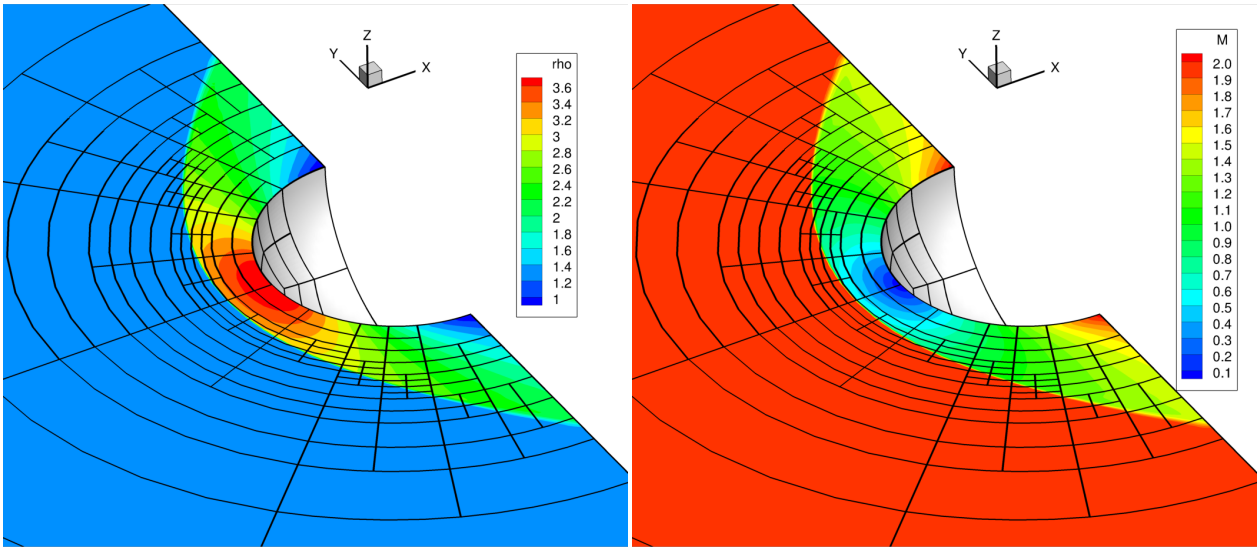
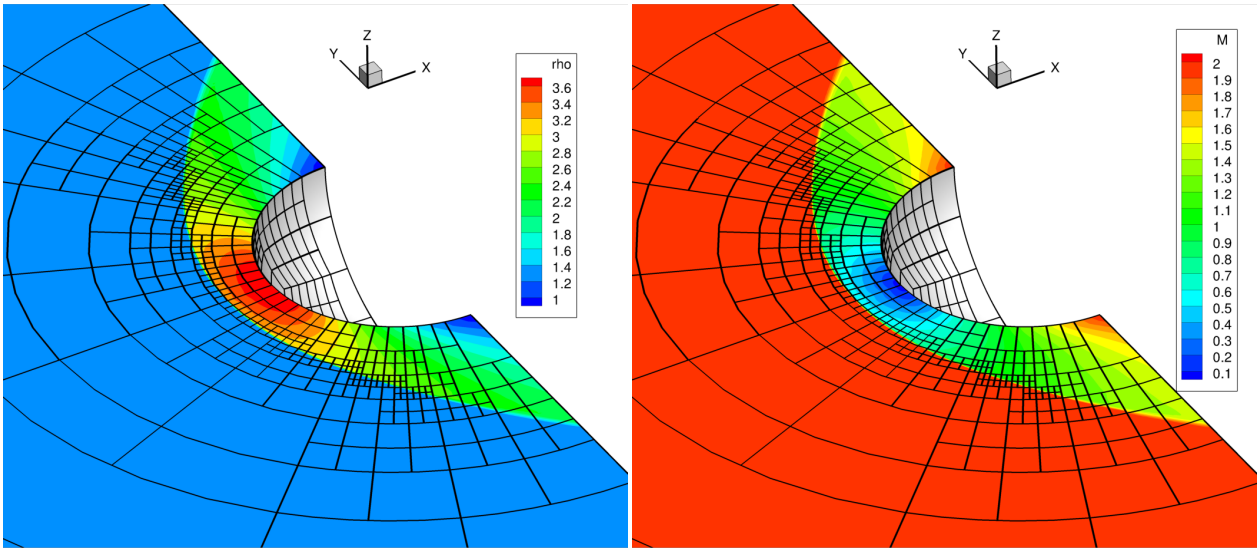


Figure 15: The initial mesh for the steady-state supersonic sphere problem with one of the five sectors removed for clarity. The block outlines are shown in dark black and the grid lines are shown in light black.

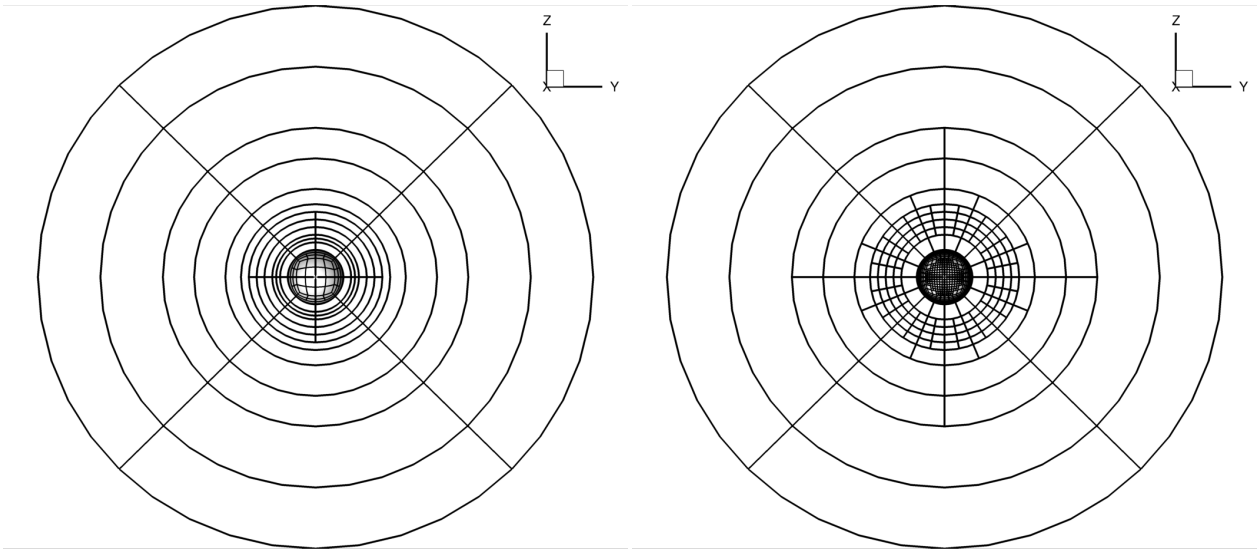


(a) Anisotropic AMR half slice of solution showing density and mach number for the supersonic sphere problem with $1,176,000$ cells.



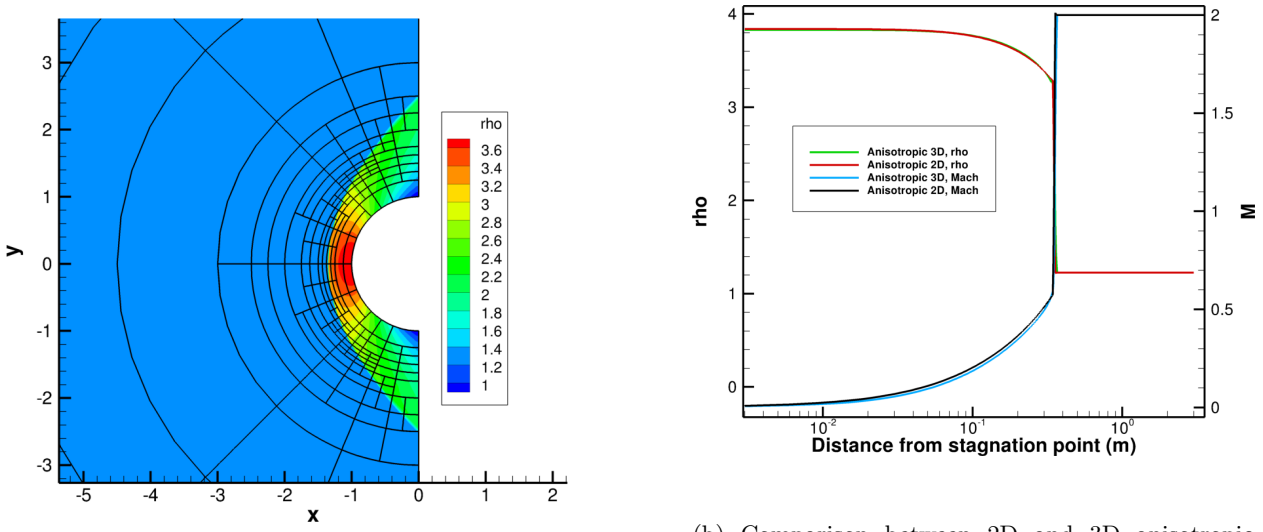
(b) Isotropic AMR half slice of solution showing density and mach number for the supersonic sphere problem with $13,110,000$ cells.

Figure 16: Steady-state $M = 2.0$ sphere problem comparison between anisotropic and isotropic AMR. Blocks are shown outlined in black.



(a) Anisotropic AMR (b) Isotropic AMR

Figure 17: Slice of the supersonic sphere mesh in the axial direction at $x = 0.0$ showing blocks outlined in black.



(a) Axisymmetric 2D anisotropic AMR simulation of a $M = 2$ supersonic sphere.

(b) Comparison between 2D and 3D anisotropic AMR of center line density and Mach number from the stagnation point to the free-stream.

Figure 18: Supersonic $M = 2$ sphere solution with 2D anisotropic AMR and comparison with 3D anisotropic AMR.

V. Conclusion

An extension of parallel block-based two-dimensional anisotropic AMR to three-dimensional anisotropic AMR has been proposed. A detailed description of the proposed AMR scheme has been described and a variety of flow problems have been solved, illustrating the capabilities of anisotropic AMR. The cubed-sphere mesh has also been shown to work well with the proposed algorithm and an initial accuracy assessment has shown increased performance over isotropic AMR. The potential for the computation of complex flow problems with significantly lower computational complexity has been demonstrated.

Future work includes applications of the proposed algorithm to more complex flow problems, including viscous flows with shear and boundary layers. The efficient and generalized implementation should allow this extension to be performed with minimal modification to the algorithm. In addition, the results of the study have also highlighted the need for improved mesh refinement criteria. To this end, the development of error-based anisotropic refinement criteria[33–38] will be considered. Extension of the finite-volume scheme to arbitrary higher-order to explore the additional savings afforded by *hp* adaptivity will also be examined.

Acknowledgements

Financial support for the research described herein 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. Both of these funding sources are gratefully acknowledged with many thanks. Computational resources for performing all of the calculations reported herein were 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.

References

[1]C. P. T. Groth, D. L. D. Zeeuw, K. G. Powell, T. I. Gombosi, and Q. F. Stout, "A parallel solution-adaptive scheme for ideal magnetohydrodynamics," Paper 99-3273, AIAA, June 1999.

[2]M. J. Berger and J. S. Saltzman, "AMR on the CM-2," *Applied Numerical Mathematics*, vol. 14, pp. 239–253, 1994.

[3]J. J. Quirk and U. R. Hanebutte, "A parallel adaptive mesh refinement algorithm," Report 93-63, ICASE, August 1993.

[4]C. P. T. Groth, D. L. De Zeeuw, T. I. Gombosi, and K. G. Powell, "Global three-dimensional MHD simulation of a space weather event: CME formation, interplanetary propagation, and and interaction with the magnetosphere," *Journal of Geophysical Research*, vol. 105, no. A11, pp. 25,053–25,078, 2000.

[5]S. A. Northrup and C. P. T. Groth, "Solution of laminar diffusion flames using a parallel adaptive mesh refinement algorithm," Paper 2005-0547, AIAA, January 2005.

[6]X. Gao, *A Parallel Solution-Adaptive Method for Turbulent Non-Premixed Combusting Flows*. PhD thesis, University of Toronto, August 2008.

[7]J. Sachdev, C. Groth, and J. Gottlieb, "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, pp. 159–177, 2005.

[8]J. Z. Zhang, "Parallel anisotropic block-based adaptive mesh refinement finite-volume scheme," Master's thesis, University of Toronto, November 2011.

[9]J. G. McDonald and C. P. T. Groth, "Numerical modeling of micron-scale flows using the gaussian moment closure," Paper 2005-5035, AIAA, June 2005.

[10]Z. J. Wang, R. F. Cphen, N. Hariharan, A. Przekwas, and D. Grove, "A 2n tree based automated viscous cartesian grid methodology for feature capturing," *AIAA paper*, no. 99–3300, 1999.

[11]Z. J. Wang and R. F. Chen, "Anisotropic solution-adaptive viscous cartesian grid method for turbulent flow simulation," *AIAA Journal*, vol. 40, pp. 1969–1978, October 2002.

[12]G. Iaccarino and F. Ham, "Automatic mesh generation for les in complex geometries," *Annual Research Briefs*, pp. 460–490, 2005.

[13]G. Iaccarino and F. Ham, "Les on cartesian grids with anisotropic refinement," in *Complex Effects in Large Eddy Simulations*, vol. 56, pp. 219–233, 2007.

[14]Z. J. Zhang and C. P. T. Groth, "Parallel high-order anisotropic block-based adaptive mesh refinement finite-volume scheme," 2011.

[15]C. P. T. Groth, D. L. De Zeeuw, T. I. Gombosi, and K. G. Powell, "A parallel adaptive 3D MHD scheme for modeling coronal and solar wind plasma flows," *Space Science Reviews*, vol. 87, pp. 193–198, 1999.

[16]X. Gao and C. P. T. Groth, "Parallel adaptive mesh refinement scheme for three-dimensional turbulent non-premixed combustion," Paper 2008-1017, AIAA, January 2008.

[17]X. Gao and C. P. T. Groth, "A parallel solution-adaptive method for three-dimensional turbulent non-premixed combusting flows," *Journal of Computational Physics*, vol. 229, no. 5, pp. 3250–3275, 2010.

[18]C. Ronchi, R. Iacono, and P. S. Paolucci, "The cubed sphere: a new method for the solution of partial differential equations in spherical geometry," *Journal of Computational Physics*, vol. 124, no. 1, pp. 93–114, 1996.

[19]L. Ivan, H. De Sterck, S. A. Northrup, and C. P. T. Groth, "Three-dimensional MHD on cubed-sphere grids: Parallel solution-adaptive simulation framework," No. 2011-3382, AIAA, June 2011.

[20]L. Ivan, H. De Sterck, S. A. Northrup, and C. P. T. Groth, "Hyperbolic conservation laws on three-dimensional cubed-sphere grids: A parallel solution-adaptive simulation framework." Submitted, 2012.

[21]S. K. Godunov, "Finite-difference method for numerical computations of discontinuous solutions of the equations of fluid dynamics," *Matematicheskii Sbornik*, vol. 47, pp. 271–306, 1959.

[22]T. J. Barth, "Recent developments in high order k-exact reconstruction on unstructured meshes," Paper 93-0668, AIAA, January 1993.

[23]P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[24]C. P. T. Groth and S. A. Northrup, "Parallel implicit adaptive mesh refinement scheme for body-fitted multi-block mesh," Paper 2005-5333, AIAA, June 2005.

[25]M. R. J. Charest, C. P. T. Groth, and Ö. L. Gülder, "A computational framework for predicting laminar reactive flows with soot formation," *Combustion Theory and Modelling*, vol. 14, no. 6, pp. 793–825, 2010.

[26]M. R. J. Charest, C. P. T. Groth, and Ö. L. Gülder, "Solution of the equation of radiative transfer using a newton-krylov approach and adaptive mesh refinement," *Journal of Computational Physics*, vol. 231, pp. 3023–3040, 2012.

[27]X. Gao, S. Northrup, and C. P. T. Groth, "Parallel solution-adaptive method for two-dimensional non-premixed combusting flows," *Progress in Computational Fluid Dynamics, an International Journal*, vol. 11, no. 2, pp. 76–95, 2011.

[28]X. Gao and C. P. T. Groth, "A parallel adaptive mesh refinement algorithm for predicting turbulent non-premixed combusting flows," *International Journal of Computational Fluid Dynamics*, vol. 20, no. 5, pp. 349–357, 2006.

[29]J. S. Sachdev, C. P. T. Groth, and J. J. Gottlieb, "Parallel solution-adaptive scheme for multi-phase core flows in rocket motors," *AIAA Paper*, no. 03–4106, 2003.

[30]J. Sachdev and C. Groth, "A mesh adjustment scheme for embedded boundaries," *Communications in Computational Physics*, 2007. accepted in April 2007.

[31]W. J. Coirier, *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, University of Michigan, 1994.

[32]D. L. De Zeeuw, *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*. PhD thesis, University of Michigan, September 1993.

[33]D. A. Venditti and D. L. Darmofal, "Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow," *Journal of Computational Physics*, vol. 164, pp. 204–277, 2000.

[34]D. A. Venditti and D. L. Darmofal, "Grid adaptation for functional outputs: Application to two-dimensional inviscid flows," *Journal of Computational Physics*, vol. 176, pp. 40–69, 2002.

[35]D. A. Venditti and D. L. Darmofal, "Anisotropic adaptation for functional outputs of viscous flow simulations," Paper 2003-3845, AIAA, June 2003.

[36]D. A. Venditti and D. L. Darmofal, "Anisotropic grid ddaptation for functional outputs: Application to two-dimensional viscous flows," *Journal of Computational Physics*, vol. 187, pp. 22– 46, 2003.

[37]V. Heuveline and R. Rannacher, "Duality-based adaptivity in the *hp*-finite element method," *Journal of Numerical Mathematics*, vol. 11, pp. 95–103, 2003.

[38]R. Rannacher and B. Vexler, "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, pp. 1844–1863, 2005.

[39]M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, pp. 484–512, 1984.

[40]M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *Journal of Computational Physics*, vol. 82, pp. 67–84, 1989.

[41]B. van Leer, C. H. Tai, and K. G. Powell, "Design of optimally-smoothing multi-stage schemes for the Euler equations," Paper 89-1933-CP, AIAA, June 1989.

[42]A. Harten, P. D. Lax, and B. van Leer, "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws," *SIAM Review*, vol. 25, no. 1, pp. 35–61, 1983.