Journal of Computational Physics 458 (2022) 111092

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Enhanced anisotropic block-based adaptive mesh refinement for three-dimensional inviscid and viscous compressible flows

Lucie Freret *, Michael Williamschen [1], Clinton P.T. Groth

University of Toronto Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, M3H 5T6, Canada

ARTICLE INFO

Article history:
Received 3 July 2021
Received in revised form 31 December 2021
Accepted 20 February 2022
Available online 2 March 2022

Keywords:
Anisotropic adaptive mesh refinement
Multi-scale flows
Compressible Euler and Navier-Stokes equations
Finite-volume methods

ABSTRACT

A parallel anisotropic block-based adaptive mesh refinement (AMR) algorithm is proposed for the prediction of physically complex flow problems having disparate spatial and temporal scales and exhibiting highly anisotropic features on three-dimensional multi-block body-fitted hexahedral meshes with non-uniform grid blocks. The proposed AMR scheme makes use of a binary tree hierarchical data structure to permit anisotropic refinement of 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 evolving unsteady flow applications. Moreover, the proposed anisotropic AMR scheme adopts a non-uniform representation of the cells within each block by directly using the neighboring cells as the ghost cells, even at a grid resolution change. This affords a number of computational advantages especially related to evaluating the solution in the ghost cells as well as to ensuring flux conservation at block interfaces. A modified upwind-based finite-volume spatial discretization scheme is applied in conjunction with the AMR scheme to the solution of Euler and Navier-Stokes equations for inviscid and viscous compressible gaseous flows. Steady-state and time-varying flow problems are considered on anisotropic adapted meshes. The potential flexibility and efficiency of the proposed enhanced anisotropic AMR scheme are demonstrated for the simulation of a number of representative flows of varying complexity.

1. Introduction

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

* Corresponding author.
E-mail address: lfreret@utias.utoronto.ca (L. Freret).
[1] Currently at KeyW Holding Corp., Hanover, MD, 21076, USA.

https://doi.org/10.1016/j.jcp.2022.111092
0021-9991/

One way to decrease the computational burden when solving multi-scale physically-complex flows without simplifying the physics and still maintaining solution accuracy is to use adaptive mesh refinement (AMR). By increasing the number of cells making up the computational grid only in those areas of the domain that require increased resolution while keeping other regions unchanged, solution-dependent AMR algorithms can adapt the mesh for treating disparate spatial scales while reducing significantly the extreme demands on computing resources. Grids that automatically adapt to the solution of the governing equations have proven to be very effective in treating problems with disparate length scales, providing the required spatial resolution while minimizing memory and storage requirements [1–10].

Hierarchical-based mesh adaptation schemes can be classified into one of two broad categories depending on the mesh partitioning/refinement algorithm used: either cell-based or block-structured AMR. Cell-based AMR methods selectively refine the individual computational cells of a mesh and typically results in fewer number of additional cells or mesh points. The newly introduced refined cells perfectly overlap of the original parent, or coarse cell. Storage of the cell connectivity is typically maintained using a hierarchical tree data structure. In three space dimensions, an octree data structure is typically used for isotropic refinement of each refined coarse cell, where there are eight new fine cells created. Alternatively, a binary tree structure can be used for anisotropic refinement in which two, four, or eight new fine cells are created and overlap the original coarse cell. Berger and Leveque [11], De Zeeuw and Powell [12,13], Powell et al. [14], Coirier and Powell [15,16], Aftosmis et al. [4,7], and Murman et al. [17] are examples of cell-based AMR methods developed for a Cartesian initial mesh with an efficient parallelized cut-cell approach for handling of complex geometries not aligned with the mesh.

Typically, these cell-based AMR methods can suffer from varying cell quality in regions near solid walls and boundaries where high-quality cells are desired for viscous computations and the discretization of elliptic operators on Cartesian cut cells can be challenging [18]. Moreover, cut-cell approaches may introduce additional computational complexity as many intersections with geometry must be computed. Nevertheless, very efficient schemes for the latter have been devised and fully three-dimensional meshes around extremely complex objects can be generated automatically and routinely [4,7,19]. It is also noted that the storage of element connectivity in cell-based AMR techniques generally requires larger data structures and the computation of connectivity, exchanging solution information, and parallelization is generally more complex than the alternative block-structured AMR methods, as will be discussed below. Despite these drawbacks, successful anisotropic implementations of cell-based AMR methods have also been achieved for two-dimensional (2D) unsteady flows [20], using an immersed boundary approach [21,22], as well as for three-dimensional (3D) laminar aerodynamics flows [23].

The second family of AMR methods is the block-structured AMR techniques and they may be divided into two subcategories: patch-based and block-based AMR methods. The former creates regular patches of cells that overlap areas of the domain that require additional resolution. The first patch-based approach was proposed by Berger and Oliger [1,24] and was subsequently extended by Berger [2] and Berger and Colella [3]. In this approach, arbitrarily oriented, rectangular patches of cells overlap coarser regions of the mesh. As the mesh hierarchy consists of nested, overlapping grids of increasingly finer resolution, a tree data structured is not required to describe the block connectivity in this case [25]. The overall adaptivity of the patch-based meshes is not as flexible as the cell-based AMR, but the more regular organization of the solution content in memory leads to more efficient and faster code execution. The most well-known patch-based AMR libraries are the Chombo [26] and Samrai [27] software libraries. Additionally, the Enzo [28] code is designed for simulations of cosmological structured formation, and the Pluto [29], Nirvana [30] and Astrobear [31] software are dedicated to astrophysical problems, with the latter making use of a distributed tree data structure and algorithm unlike the other patch-based AMR. A recent review of patch-based codes is provided in the survey by Dubey et al. [32]. All of the aforementioned libraries are restricted to isotropic refinement strategies; however, Christopher et al. [25] have recently implemented an anisotropic approach to the refinement of the patch-based mesh for Chombo, leading to a complex set of procedures involving intermediate patches for the exchange of "ghost" cell ("halo" or "guard" cells) information between adjacent patches. The Chombo software has also been extended for use with curvilinear or mapped grids and high-order discretization methods in applications to transient, compressible, turbulent, reacting and non-reacting fluid flows with complex geometry as described by Guzik et al. [33], Gao et al. [34], and Owen et al. [35].

As originally proposed by Quirk [36], Berger and Saltzman [37], and Groth et al. [5,6], block-based AMR is a simplified variant of the patch-based AMR scheme. Block-based AMR considers a multi-block grid and refines full non-overlapping grid blocks, each containing a fixed number of cells. Similar to cell-based AMR, blocks are typically refined to produce offspring and the refinement history is stored within a tree-based data structure. The main difference in this case is that the smallest divisible element of the grid is the block and not the cell. As with patch-based methods, the block-based AMR strategy is not as optimal from a refinement perspective as cell-based methods and can produce some local over-refinement of the mesh, this negative aspect is generally 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, particularly for implementations on parallel high-performance computing systems. The Paramesh [38] and p4est [39] libraries are examples of block-based AMR software tools that consider a hierarchy of non-overlapping grids, organized as leaves in either quadtree or octree data structures for 2D and 3D meshes, respectively. Some examples of simulation codes using block-based AMR are the MPI-AMRVAC software [40,41], dedicated to hydrodynamics and magnetohydrodynamics (MHD), the relativistic counterpart of the AMRVAC software, BHAC [42], for solving the equations of ideal general-relativistic magnetohydrodynamics and studying astrophysical phenomena such as black holes, the Flash [43] code, a multi-physics software tool originally based on Paramesh [38] and recently extended for use with

Chombo [26], the SWMF [44] code, dedicated to space weather simulations, and ForestCLAW [45], another example of a block-based simulation tool based on the p4est library.

The preceding block-based AMR schemes were largely restricted to multi-block Cartesian mesh. More recently, Groth and co-researchers [46,47,9,10,48] have developed a rather flexible block-based AMR scheme allowing automatic solution-directed mesh adaptation on multi-block body-fitted (curvilinear) meshes consisting of quadrilateral (2D case) and hexahedral computational cells (3D case). This block-based approach has been shown to enable efficient and scalable parallel implementations for a variety of flow problems, as well as allow local refinement of body-fitted mesh with anisotropic stretching. The latter aids in the treatment of complex flow geometry and flows with thin boundary, shear, and mixing layers and/or discontinuities and shocks. Extensions of the block-based body-fitted AMR approach for the treatment of embedded boundaries not aligned with the mesh [49] as well as for cubed-sphere grids [48,50] have also been considered. The block-based approach for multi-block, body-fitted meshes has been applied to the prediction of a wide range of flow problems including multi-phase turbulent core flows in rocket motors [51,46], laminar flames [52,53,10,54], turbulent non-premixed flames [47,10], micron-scale flows [55,56], radiative heat transfer [57], and global MHD simulations [48,50].

Variants of the block-based AMR approach for treating more complex geometries with curved boundaries have also been considered based on a combination of Chimera-type overlapping grids with AMR. In this approach, overlapping body-fitted or curvilinear grids are used with one or more background Cartesian grids which fill computational domain. Boden and Toro [58] and Henshaw et al. [59–61] have shown that AMR on overlapping grids can provide an efficient approach for solving problems with multiple space and time scales for complex geometry. Challenges for this variant of block-based AMR approaches are associated with determining and/or re-evaluating block connectivity as well as blocks hidden following mesh refinement (this information must be re-computed and stored) and interpolation of solution quantities between different base grids and/or between grids with different levels to ensure accurate results and global conservation properties of the discretization solution. Additionally, hybrid patch/block-based AMR approaches have also been considered. Holst and Keppens [62] applied a hybrid approach to general curvilinear coordinate systems, modifying the full tree data structure to allow for incomplete block refinement and incorporate ideas from patch-based strategies. However, the mixed hierarchical data structure can complicate the neighbor search algorithm. Holst and Keppens [62] compared the three AMR strategies, i.e., patch-based, tree block-based, and hybrid patch/block-based methods, and their findings suggest that a block-based AMR approach is the most efficient, in terms of the execution speed for the same solution accuracy.

The cell-based and block-structure approaches mentioned above were all largely restricted to isotropic refinement of the mesh; that is, where the mesh is coarsened or refined equally in all coordinate directions. While such strategies have shown success in reducing computational costs, further savings can be afforded by AMR strategies based on the anisotropic refinement of the mesh, i.e., where the mesh is coarsened or refined in a preferred direction [63–66]. Anisotropic AMR methods can reduce further the cost of CFD simulations by allowing the creation of high-aspect-ratio cells aligned with solution content requiring higher resolution in a preferred direction (e.g., for flow regions with thin shear and boundary layers as well as shocks) while still having larger cells in regions with lower resolution requirements and this can be accomplished without *a priori* knowledge of the solution and/or pre-prescribed grid point clustering. For problems involving shock waves and or thin boundary layers which are strongly aligned with a coordinate direction of the grid and for meshes involving more than 5-6 levels of refinement, the reduction in the mesh size can be considerable and easily more than 95% as shown previously by Zhang and Groth [63,64] in 2D cases. Even for multi-scale flows in which the anisotropic AMR reverts to isotropic AMR in some regions because the solution variation is not strongly aligned with the grid coordinate directions, reductions in the mesh size can be substantial. For 2D cases, 50% reductions were achieved in many cases and mesh-size reductions were estimated to be 50–75% or more for equivalent 3D problems [63,64].

All of the patch- and block-based AMR approaches described above also have in common the use of ghost cells to store information for the exchange of solution content between adjacent or neighboring patches or blocks. The size of a patch or block in terms of the number of cells in each logical or coordinate direction is extended by a surrounding layer of ghost cells, all at the same level of refinement as that of the core cells for the patch or block, with the ghost cells overlapping the neighboring patches or blocks. This results in an overall enlarged uniform local block or unit that contains cells associated with the patch or block of interest as well as those of adjacent patches or blocks needed for the update of the solution. Standard multigrid-type restriction and prolongation operators are required and used to evaluate or "fill" the solution in the ghost cells of neighboring patches or blocks with resolution changes [6,46,9,32,25]. For an isotropic methodology, this can force the message passing to occur in a particular order [64,25]. The finer ghost cells have to be filled before the coarser ghost cells with the restriction occurring prior to the prolongation. For an anisotropic methodology with cells not permitted to span a neighboring cell, the ghost cell filling becomes quite complicated. An iterative combination of restriction and prolongation can be implemented as considered by Zhang and Groth [64] or a temporary new common fine grid can be computed on each side of the block boundary before switching back to the original mesh as in [25]. While doable for standard second-order schemes on both 2D and 3D meshes, the uniform representation of the cells within each extended patch or block introduces a number of possible computational inefficiencies. Furthermore, for finite-volume solution methods, a flux correction strategy is required to maintain the flux conservation properties of the scheme at patch or block interfaces with resolution changes [6,46,9,32,25]. Additionally, the complications associated with filling of the ghost cells and correcting the interface fluxes make the implementation of an anisotropic structured AMR scheme for

use in conjunction with a high-order spatial discretization method, such as the high-order finite-volume scheme recently developed by Ivan et al. [67,50], Susanto et al. [68], and Charest et al. [69], particularly problematic.

With the preceding in mind, an enhanced and affordable anisotropic block-based AMR scheme is proposed and described. The proposed approach represents a modified extension of 2D anisotropic AMR of Zhang and Groth [63,64] to the 3D case. As opposed to using a uniform treatment of the core and ghost cells within a block, the proposed 3D block-based AMR scheme adopts a non-uniform representation of the grid blocks by directly using the adjacent or neighboring cells as the ghost cells, even for those with different levels of refinement as found at mesh resolution changes. This both eliminates the need for the restriction and prolongation of solution information to fill the ghost cells and avoids the requirement for the conservative flux corrections at the block interfaces. The proposed anisotropic AMR scheme is therefore more tractable and, without extra cost, well-suited for use with high-order spatial discretization schemes, as has been shown recently by Freret et al. [70,71]. The proposed 3D 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 very little overhead to the overall computational complexity of the problem solution. 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.

To illustrate the performance of the proposed anisotropic block-based AMR scheme with non-uniform grid block treatment, the approach is applied herein, in conjunction with a second-order finite-volume scheme, to the solution of the Euler and Navier-Stokes equations for inviscid and viscous compressible gaseous flows for a number of representative flow problems. The remainder of the paper is then organized as follows. The finite-volume spatial discretization scheme and the application of the solution methodology to both steady-state and time-varying problems are described in Section 2. The non-uniform cell block treatment and its use within the finite-volume scheme are discussed in Section 3. The extension and description of the anisotropic AMR scheme for three-dimensional computational domains are studied in Section 4. Finally, numerical results are provided in Section 5 for a number of inviscid and viscous flows to demonstrate the potential of the proposed enhanced anisotropic AMR scheme.

2. Finite-volume scheme

The proposed anisotropic block-based AMR scheme is applied herein to the solution of Euler and Navier-Stokes equations governing inviscid and viscous compressible three-dimensional flows of a polytropic gas. For viscous applications considered herein, the flow Reynolds numbers are restricted to ensure that the flows remain laminar. Applications to turbulent flows and any associated additional modeling of any unresolved solution content are not considered herein. The latter will be the subject of future follow-on studies.

2.1. Conservation equations for three-dimensional gaseous flows

The conservation form of the governing equations for a compressible polytropic gas, either inviscid or viscous, can be written as

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

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, $\mathbf{U}$ is given by

$$\mathbf{U} = [\rho, \rho u, \rho v, \rho w, \rho e]^{\mathrm{T}}\,, \tag{2}$$

where ρ is the gas density, u, v, w are the components of the velocity vector in the x, y, z directions, $e = p/(\rho(\gamma - 1)) + u^2/2$ is the specific total energy, $p = \rho RT$ is the gas pressure, T is the gas temperature, R is the ideal gas constant, and γ is the ratio of specific heats and $h = e + p/\rho$ is the specific enthalpy. The corresponding solution flux dyad, $\vec{\mathbf{F}}$, is given by $\vec{\mathbf{F}} = [\mathbf{F}_h - \mathbf{F}_v, \mathbf{G}_h - \mathbf{G}_v, \mathbf{G}_h - \mathbf{G}_v]$ where $\mathbf{F}_h$, $\mathbf{G}_h$, $\mathbf{H}_h$ are the inviscid or hyperbolic flux vectors associated with the x, y, z coordinate directions, respectively, and $\mathbf{F}_v, \mathbf{G}_v, \mathbf{H}_v$ are the corresponding viscous flux vectors in the x, y, and z coordinate directions, respectively. Definitions of the latter are given elsewhere. Please refer to, for example, the textbooks by Schlichting and Gersten [72] or Hirsch [73].

2.2. Upwind-based spatial discretization scheme

The high-resolution upwind-based finite-volume scheme applied herein follows from the application of the integral form of the conservation equations described above to a hexahedral computational cell and results in the following semi-discrete form of the governing equations:

$$\frac{\mathrm{d}\overline{\mathbf{U}}_i}{\mathrm{d}t} = -\frac{1}{V_i}\sum_{m=1}^{N_f} \vec{\mathbf{F}}_{i,m} \cdot \vec{n}_{i,m}\, A_{i,m} = -\mathbf{R}_i\,, \tag{3}$$

where $\overline{\mathbf{U}}_i$ is the volume-averaged value of the conserved solution vector for cell, i, V_i is the volume of cell i, $\vec{\mathbf{F}}_{i,m}$, $\vec{n}_{i,m}$ and $A_{i,m}$ denotes the solution flux, outward pointing unit normal, and area for the m^{th} face of cell i, respectively. The integer value, N_f, represents the number of faces for the cell and, for a non-adapted or uniform-resolution mesh, $N_f = 6$ with $m \in \{W, E, S, N, B, T\}$ corresponding to the six faces (west, east, south, north, bottom and top) of a hexahedral cell. Each cell, i, is contained within a single structured grid block. The complete multi-block body-fitted mesh is then composed of a number of these structured grid blocks and a general unstructured connectivity is permitted between the grid blocks making up the original non-adapted or unmodified mesh [9].

In the upwind-based finite-volume scheme, limited linear least-squares solution reconstruction [74,9] and Riemann-solver based flux functions are used in combination with the standard mid-point quadrature rule in the evaluation of the surface integrals of the numerical fluxes appearing in (3). The limiters of Barth and Jespersen [75] and Venkatakrishnan [76] are both considered here to ensure monotonicity of the solution reconstruction. Either the so-called HLLE flux function [77, 78] or Roe linearized approximate Riemann solver [79] is used in the evaluation of the inviscid solution fluxes at the faces of the cell. The viscous solution fluxes are not upwinded and determined using a centrally-weighted scheme based on a diamond-path Green-Gauss integration procedure for the evaluation of the average value of solution gradients at each cell interface [9,80,54].

2.3. Time-marching schemes

The semi-discrete from of (3) represents a system of coupled, non-linear, first-order, ordinary differential equations (ODEs) for the cell-averaged quantities, $\overline{\mathbf{U}}_i$. For solution of unsteady problems related to the Euler equations considered herein, a standard, second-order accurate, two-stage, explicit, Runge-Kutta, time-marching scheme [81] is used to evolve the solution of (3) forward in time while, for the steady-state or time-invariant problems pertaining to the Euler equations, the multi-stage optimally smoothing schemes due to van Leer et al. [82,83] with local time-stepping are used. A fully-implicit time-marching scheme, based on Newton's method, is used for the rapid and efficient solution of the steady-state problems related to the Navier-Stokes equations. The latter follows the algorithm previously developed by Groth and co-workers [84,57,54] which has been well adapted for performing CFD simulations on large multi-processor parallel clusters. The implementation of the implicit time-marching scheme is based on a Jacobian-free inexact Newton method coupled with an iterative Krylov subspace linear solver. The generalized minimal residual (GMRES) algorithm of Saad and co-workers [85–88] is used to solve the resulting large sparse non-symmetric system of linear equations at each Newton step and a combination of an additive Schwarz global preconditioner and a block incomplete lower-upper local preconditioner is used to speed-up convergence. Additionally, the implicit Euler startup procedure of Mulder and Van Leer [89] with successive evolution/relaxation (SER) is used to improve the global convergence of the Newton method. Further details of the Newton method for steady viscous flows are provided elsewhere [84,57,54].

3. Non-uniform representation of grid blocks

The finite-volume scheme described in the previous section is applied to the hexahedral computational cells of a multi-block body-fitted mesh in which the complete grid is composed of a number of structured sub-blocks. Each of these structured blocks of the mesh consists of $N_i \times N_j \times N_k$ hexahedral cells where N_i, N_j, and N_k are even, but not necessarily equal, integer values. To allow solution information transfer between neighboring sub-blocks, each of the structured uniform grid blocks is extended by a number of ghost cell layers, N_g (taken to be 2 here), while still preserving the mesh resolution of the neighboring blocks. For adjacent grid blocks with and without resolution changes (the latter can occur in the case of mesh adaptation), this results in an extended non-uniform sub-block composed of a single interior block containing the core computational cells of the grid block and a minimum of at least 26 block boundary elements (one for each 6 faces, 12 edges and 8 corners in the case of no resolution changes between neighboring blocks) and a maximum of at most 56 block boundary elements (4 neighbors per face, 2 per edge and 1 per corner in the case of an adapted computational mesh). Examples of two such non-uniform blocks are depicted in Fig. 1 for $N_i = 8$, $N_j = 8$, $N_k = 8$, and $N_g = 2$. On the right of this figure, a grid block with 26 boundary elements is shown corresponding to the non-adapted case while, on the left, a grid block with 28 boundary elements is shown corresponding to a mesh with a resolution change at one of the faces and edges of the block. Hence, a single non-uniform grid block is then represented logically by a minimum of at least 27 3D arrays of storage in memory and by a maximum of at most 57 3D arrays containing the solution and geometry information for the block and its adjacent neighbors. The storage array for the single core block will have dimensions $N_i \times N_j \times N_k$, while the storage for the boundary elements will be reduced in size and depend on the value of N_g.

Two major advantages arise from the proposed non-uniform treatment of the grid blocks making up an adapted multi-block body-fitted mesh. First of all, the usual prolongation and restriction procedures are no longer required to determine the solution content in the ghost cells at grid blocks with resolution changes. Instead, in the proposed implementation, the ghost cell solution information is directly evaluated by injection of the solution from the cells of the neighboring blocks. Secondly, the additional procedures for correcting the solution flux at non-conforming grid-block faces with resolutions changes in order to maintain the discrete conservation properties of the finite-volume scheme, as typically required in other block-based AMR implementations [38,29,32,9,65,25] are no longer required. Flux conservation is naturally handled by the proposed modifications to the finite-volume method which account for the ghost cells of the boundary elements

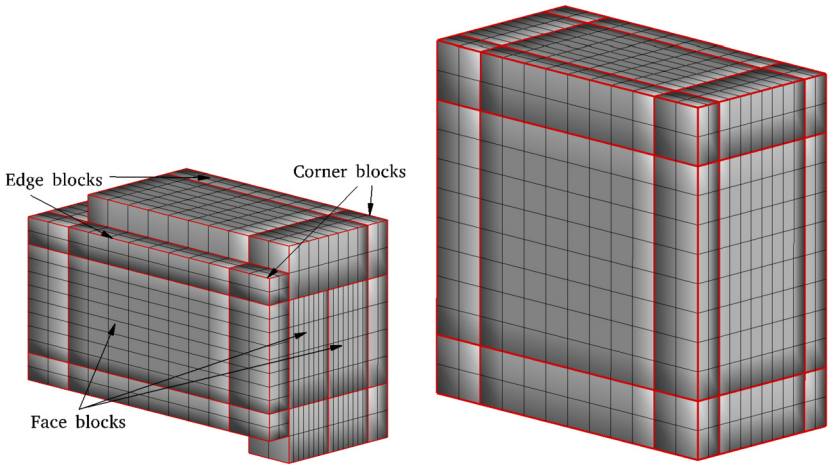


Fig. 1. Illustration of non-uniform block structure used in representing the multi-block body-fitted mesh within the proposed block-based AMR scheme showing a sub-block 26 boundary elements corresponding to the non-adapted case (left) as well as a grid block with 28 boundary elements corresponding to mesh with a resolution change at one of the faces and edges of the block (right).

now possibly being at different levels of resolution from the core cells within the grid block in the case of adapted mesh as described in Section 3.1 to follow. Additionally, a direct consequence of these two desirable features of the proposed non-uniform grid-block treatment is that the message passing procedure used to exchange the solution content between adjacent blocks, and thereby fill the ghost cells, can now be performed without any preferred ordering of the messages based the change in the level of refinement between neighboring blocks. On the other hand, the procedures for the evaluation of the numerical fluxes and piecewise solution reconstruction becomes slightly more complicated for some of the hexahedral cells located at an interior boundary of the block. In particular, the proposed spatial discretization scheme must be modified to allow for the non-uniform nature of the grid and the possible presence of so-called "non-conforming faces" and "hanging" nodes. The proposed modifications to the flux evaluation and solution reconstruction procedures are discussed in the next two sections.

3.1. Numerical flux evaluation

The evaluation of the solution residual, $\mathbf{R}_i$, of (3) requires the integration via quadrature of the solution fluxes through the faces of each computational cell which, in turn, necessitates the evaluation of the numerical fluxes at the centroid of each face element for the cell. In the proposed non-uniform block treatment, the number of face elements, N_f, for the cell varies depending on the local refinement of the adapted mesh and, in particular, depends on the relative location of the cell within the grid block and relative refinement level of the associated ghost cells. For all interior core cells that are not located on the boundary of the grid block, the cell connectivity with its neighbors is then fully conforming and $N_f = 6$. Conversely, for core cells located at a block boundary and whose surface area includes a portion of the grid block boundary, $N_f \geq 6$ in general and the number of face elements depends on resolution of the neighboring ghost cells sharing the face relative to the cell. As will be discussed, the maximum change or difference in refinement levels for each coordinate direction between adjacent blocks is restricted in the proposed block-based AMR scheme to be one such that the cell resolution ratio is at most 2:1 between core and ghost cells. At the block boundary, the value of N_f can therefore range from a minimum of $N_f = 6$ (i.e., for a cell with fully conforming neighbors and no resolution change or for the case where the resolution of the neighbor ghost cells is coarser and at a lower level of refinement) up to a maximum of $N_f = 15$ (i.e., $N_f = 3 \times 1 + 3 \times 4 = 15$ for a core cell located at the corner of a block with three faces all associated with non-conforming ghost cells with hanging nodes arising from the isotropic refinement of the neighboring blocks to a higher level of refinement). There are three different basic possible configurations or face connectivities for a core cell with its neighbors, depending on the ghost cell geometry. These three connectivity types are depicted in Fig. 2. For fully anisotropic refinement of the grid block, the resolution of the core and ghost cells in each of the computational coordinate directions are not necessarily the same and are independent of each other. As a consequence, there are then up to nine (9) different combinations of resolution change at the block boundaries between the core cells and their adjacent ghost cells yielding one of these three possible face connectivities (face connectivity type "A", "B", or "C") as summarized in Table 1.

Referring to Fig. 2, face connectivity type A involves a core cell that is at an equal or higher level of refinement in both face normal coordinate directions compared to its neighboring ghost cell, as depicted in Fig. 2(a). In this case all of the terms required for evaluation of the solution of (3) are known and $\mathbf{R}_i$ can be computed directly without any additional assumptions or modifications. Face connectivity type B is an example of multiple non-conforming ghost cells at higher levels

Table 1
Summary of the 9 different combinations of resolution change at the block boundaries between a interior core cell at refinement level l_ξ and l_η and their adjacent ghost cells yielding one of three possible face connectivity types: A, B, or C, where l_ξ and l_η are refinement levels associated with the two directions normal to the face.

Interior Cell Refinement Level	Ghost Cell Refinement Level	Connectivity Type
l_ξ, l_η	l_ξ, l_η	A
l_ξ, l_η	l_ξ+1, l_η	B
l_ξ, l_η	l_ξ, l_η+1	B
l_ξ, l_η	l_ξ+1, l_η+1	B
l_ξ, l_η	l_ξ-1, l_η	A
l_ξ, l_η	l_ξ, l_η-1	A
l_ξ, l_η	l_ξ-1, l_η-1	A
l_ξ, l_η	l_ξ+1, l_η-1	C
l_ξ, l_η	l_ξ-1, l_η+1	C

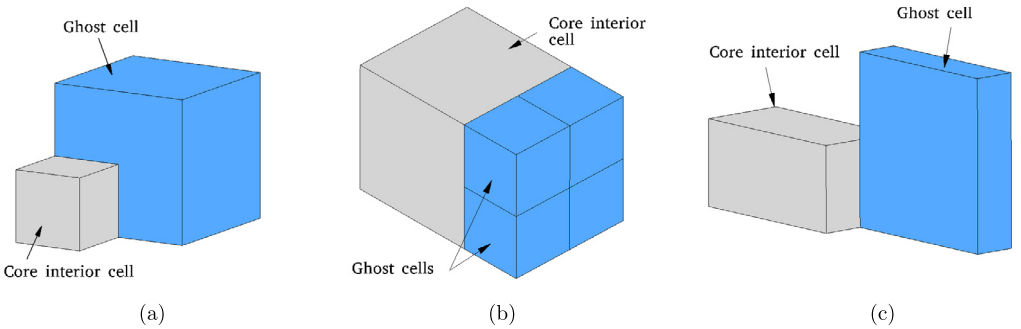


Fig. 2. Schematic diagram illustrating the three different types of face connectivities possible at the interface between a core interior cell and the adjacent neighboring ghost cells depending on the relative level of refinement between the cells.

of refinement in both face normal coordinate direction as shown in Fig. 2(b). In this case, the flux $\vec{\mathbf{F}}_{i,m}$ through the m$^{\text{th}}$ face of cell i is taken to be the sum of all ghost faces, $N_g = 4$, having face areas, A_g, with $\bigcup_{g \in N_g} A_g = A_{i,m}$ such that

$$\vec{\mathbf{F}}_{i,m} \cdot \vec{n}_{i,m} A_{i,m} = -\sum_{g=1}^{N_g} \vec{\mathbf{F}}_{i,m,g} \cdot \vec{n}_g A_g, \tag{4}$$

where $\vec{\mathbf{F}}_{i,m,g}$ is the flux through the adjacent ghost cell, g, with face area, A_g. Finally, face connectivity type C involves two cells with two opposite grid resolution changes in the two face normal directions as shown in Fig. 2(c). For this case, $N_g = 2$ and surface areas $A_{i,m,g}$ of the ghost cells are defined such that

$$\bigcup_{g \in N_g} A_{i,m,g} = A_{i,m} \text{ with } A_{i,m,g} = \left(A_g \cap A_{i,m}\right). \tag{5}$$

Normal vectors $\vec{n}_{i,g}$ of these new faces are computed and finally, for a particular face m, $\vec{\mathbf{F}}_{i,m}$ is given by

$$\vec{\mathbf{F}}_{i,m} \cdot \vec{n}_{i,m} A_{i,m} = \sum_{g=1}^{2} \vec{\mathbf{F}}_{i,m,g} \cdot \vec{n}_{i,g} A_{i,m,g}. \tag{6}$$

3.2. Solution reconstruction

As noted previously, piecewise limited linear polynomial interpolation is used in the reconstruction of the solution within each computational cell of the non-uniform grid blocks such that the spatial variation of any component of the solution, $u(\vec{x})$, in cell, i, is represented as

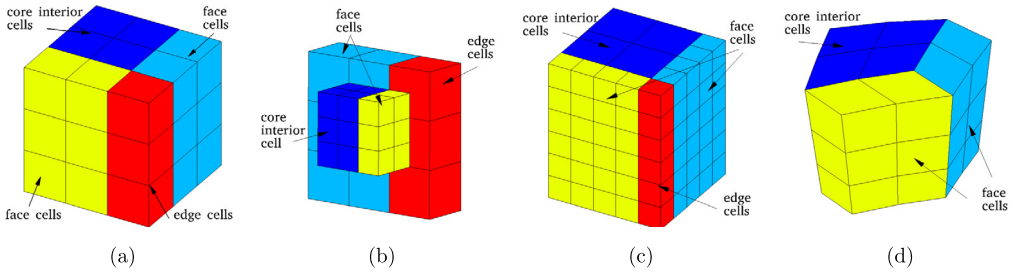


Fig. 3. Examples of four possible different stencils associated with an interior core cell (shaded in dark blue) adjacent to a grid-block edge that include ghost cells of the neighboring blocks for a regular non-degenerate logically Cartesian mesh (a, b, c) and a cubed-sphere grid (d). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$u_i(\vec{x}) = \overline{u}_i + \Phi_i \vec{\nabla} u_i \cdot \left(\vec{x} - \vec{x}_i\right) , \tag{7}$$

where $\overline{u}_i$ is the mean or average value of the solution for cell i, $\vec{x}_i = (x_i, y_i, z_i)$ is the 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. The solution gradients, ∇u_i, for cell, i, in (7) is reconstructed using the N nearest neighbors via a least squares approach that minimizes the second-order accurate error for the fit given by

$$\epsilon_i^2 = \sum_{k=1}^{N} \epsilon_{ik}^2 = \sum_{k=1}^{N} \left[u_k - \overline{u}_i - \vec{\nabla} u_i \cdot \left(\vec{x}_k - \vec{x}_i\right) \right]^2 , \tag{8}$$

where u_k is the value of the solution variable at the centroid of the nearest neighboring cell, k, with $k \in [1, N]$

The number of neighbors, N, used in the reconstruction procedure for the cell defines the reconstruction stencil for the cell and this stencil is not fixed here and depends on the geometry and topology of the adapted mesh and adjacent cells. The neighbors within the first layer of surrounding cells for core cell i within the unstructured grid block are always used here to define the stencil, no matter the mesh topology. For all interior core cells not adjacent to the boundary or edge of the grid, which then by construction have neighboring cells all at the same resolution, the stencil is equivalent to that for a uniform mesh and $N = 26$. Conversely, Fig. 3 provides examples of the different stencils that are possible for a given cell of interest (shaded in dark blue) located close to a block edge for which the stencils now include cells from adjacent neighboring blocks. For all ghost cells at the same resolution as the interior cells (i.e., the uniform case), $N = 26$ neighbors are again included as shown in Fig. 3(a). Additionally, $N = 26$ for ghost cells which are either at an equal or lower level of mesh refinement (i.e., coarser mesh) as depicted in Fig. 3(b). Fig. 3(c) depicts a stencil in which the ghost cells are all at higher level of refinement (i.e., finer mesh). In this case, the number of neighbors for the reconstruction stencil is then $N = 65$. The minimum number of neighbors in a reconstruction stencil results from a multi-block cubed-sphere grid [48,50] that presents both corner and edge degeneracies in the grid block structure. For this stencil, $N = 23$, as is shown in Fig. 3(d). For all four of these configurations, there is always a sufficient number neighbors such that Eq. (8) remains over-determined. It is also worth mentioning that solution of the least-square problem of Eq. (8) ensures the exact reconstruction of a linear function and second-order spatial accuracy, even for non-uniform mesh spacing, as shown in Section 5.2.3 and previously by Freret et al. [71] and Ivan et al. [48].

4. Anisotropic block-based AMR scheme

In the proposed anisotropic block-based AMR scheme, mesh adaptation is accomplished by refining and coarsening existing grid blocks in a preferred logical coordinate direction. In an analogous fashion to conventional isotropic block-based AMR methods, each refinement produces new blocks called “children” from a “parent” block via a division process in which the parent block is split into two children in a direction specified by the prescribed solution-dependent refinement criteria and the resulting child block has the same number of interior core cells as the parent. This process effectively doubles the local resolution of mesh in the preferred direction of interest, producing a grid with anisotropic resolution. The children can be refined further in subsequent adaptation of the mesh and the refinement process can be reversed in regions that are deemed over-resolved whereby two, four, or eight child blocks are coarsened or re-merged into a single parent block. A flexible hierarchical binary tree data structure is used for tracking block connectivity and mesh refinement history, based on an extension to the octree data structure adopted by Gao and Groth [9]. Please refer to Fig. 4.

Similar to the tree data structure of Gao and Groth [9], the proposed binary tree structure allows for multiple root blocks representing the initial non-adapted multi-block mesh and these root grid blocks may have a general unstructured connectivity. Each node of the binary tree is either an active computational grid block or references two child nodes, corresponding to the refinement of the original grid block in one of the three logical coordinate directions. At every node, the refinement level for each coordinate direction, and the coordinate direction in which the block is refined are all stored. This refinement level information is used when flagging blocks to refine or coarsen and aids in efficient searching for nearest

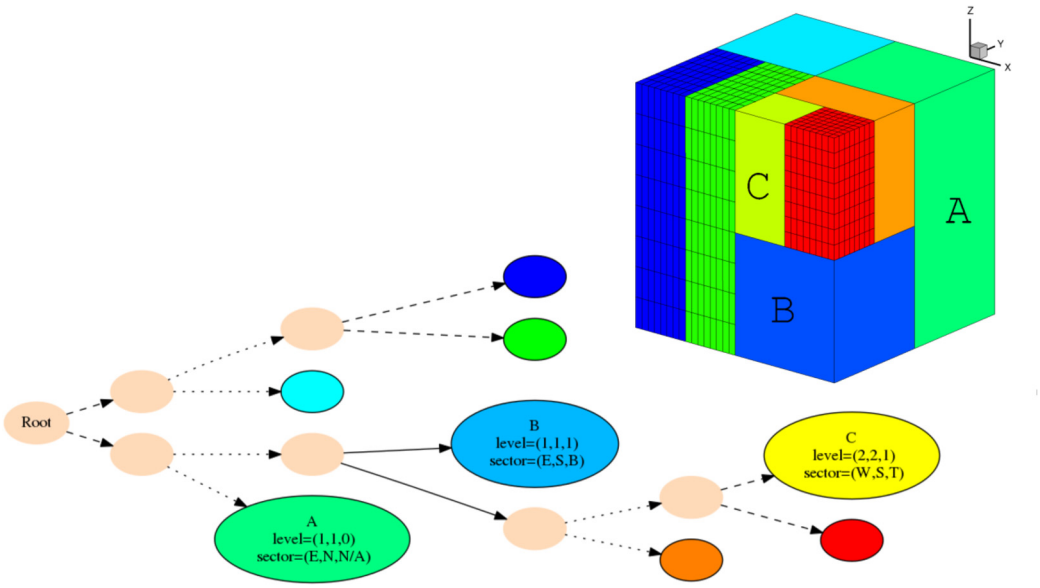


Fig. 4. Example of a 3D multi-block Cartesian mesh obtained after several levels anisotropic AMR of the mesh and the corresponding binary tree used to track both the connectivity and refinement of the non-uniform grid blocks making up the mesh. The dashed arrows represent a division of the grid blocks in the x-coordinate direction, dotted arrows correspond to a division or split in the y-coordinate direction, and solid arrows are associated with a split in z-coordinate direction. Each filled ellipsoid represents a node in the tree and the actual grid blocks are represented by the last node or leaves of the binary tree. The peach colored nodes are not computational blocks (they are used to track the connectivity between blocks) and the other colored nodes represent computational blocks with each color associated with a different grid block.

Algorithm 1 Anisotropic Block-Based AMR Scheme for 3D Body-Fitted Hexahedral Meshes.

1: *Flagging of Grid Blocks.* Based on directional solution-dependent refinement criteria, blocks are flagged for refinement, coarsening, or to maintain their current resolution;

2: *Conflict Checking.* The refinement flags are subsequently modified to remove AMR conflicts and ensure the rules for refinement and coarsening of grid blocks are not violated;

3: *Determination of Newly Created Grid Blocks.* All blocks flagged to refine, are refined in the flagged direction(s), all blocks flagged to coarsen, are coarsened in the flagged direction(s) and the binary tree is updated to reflect the refinements;

4: *Connectivity Rearrangement.* Binary tree is modified to ensure parents of all blocks flagged coarsen are split in the appropriate direction;

5: *Determination of Grid-Block Connectivity.* Solution block connectivity or neighbor information is re-computed and stored for each block;

6: *Message Passing of Information Between Adjacent Grid Blocks.* Solution and geometry information are shared between adjacent blocks through MPI procedures.

neighbors to the block. Additional block sector information for each coordinate direction is also stored for each tree node, indicating which logical half of the parent block the child block occupies.

Fig. 4 depicts an example of a 3D multi-block hexahedral AMR mesh consisting of solution blocks at various levels of refinement and the corresponding binary tree data structure. In the particular case shown, the grid is a regular Cartesian mesh, but more general multi-block body-fitted grids can be treated as well. The last nodes or leaves of the binary tree correspond to the actual computational grid blocks. In this example, block A is obtained after refinement of the initial mesh consisting of a single root block in the x-coordinate direction followed by a refinement in the y-coordinate direction. Hence, the refinement level is given by the triplet (1, 1, 0). Its sector information (E, N, N/A) indicates that block A occupies a East, North section at level (1, 1, 0), with N/A meaning undefined. Other active computational blocks, including blocks B and C, result from further anisotropic refinement of the mesh.

A skeletal summary of the proposed anisotropic AMR scheme is provided in Algorithm 1, which can be seen to consist of 6 sequential stages or steps. The anisotropic AMR scheme here is similar in many respects to the previous approach of Zhang and Groth [63,64] for 2D flows; however, the use of non-uniform representation of the grid blocks as described above and the additional complexities associated with the extension of the algorithm from 2D to 3D result in significant deviations from this previous 2D anisotropic scheme. In what follows, each stage or component of the proposed anisotropic AMR for 3D multi-block body-fitted hexahedral meshes is described, with emphasis on those components that have been extended or modified to deal with 3D computational domains.

Algorithm 2 Blocks flagging in ξ direction.

1: $\mathcal{F}_\xi = \mathrm{N} = \mathrm{N}_\xi$
2: **if** $\max_{i \in N} \epsilon_{i,\xi} < \mathrm{threshold}_{coarse}$ **then**
3: $\quad \mathcal{F}_\xi = \mathrm{C} = \mathrm{C}_\xi$
4: **else if** $\max_{i \in N} \epsilon_{i,\xi} > \mathrm{threshold}_{refine}$ **then**
5: $\quad \mathcal{F}_\xi = \mathrm{R} = \mathrm{R}_\xi$
6: **end if**

4.1. Flagging of grid blocks

The first step of the anisotropic AMR procedure is to assign three separate refinement flags, $\mathcal{F}_\xi$, $\mathcal{F}_\eta$, and $\mathcal{F}_\zeta$, for each direction for each block, indicating whether or not the block should be refined, R, coarsened, C, or remain unchanged N, in each coordinate direction. To determine these flags, one or more refinement criteria can be used. The most effective refinement criteria are based on solution error measures tied to solution outputs or functionals of interest which not only provide direct indications to whether the solution features in local regions are sufficiently resolved, but also the coordinate direction in which the solution is under-resolved as proposed by Becker et al. [90,91], Venditti and Darmofal [92–94], Narechania et al. [95] and references therein. Somewhat less sophisticated refinement criteria based on flow physics and features and structure of the flow are considered in this work. The latter are attractive due to their relative simplicity and efficiency and have been shown to effective in treating a range of multi-scale flow problems having disparate spatial scales [46,47,9,10,48].

Physics-based criteria are generally based on heuristic measures of the solution, which can depend on either primitive or conserved solution quantities, may involve expressions derived from these variables. The heuristic refinement measures are usually scaled by the volume of the cell and normalized by the magnitude of the solution variables involved. For an anisotropic AMR, separate refinement indicators and criteria are required for each logical coordinate direction of the grid. In this study, gradients of density, pressure, and velocity magnitude are used to direct the anisotropic AMR. For cell, i, expressions for the directional refinement criteria based on the gradient are as follows:

$$\epsilon_{i,\xi} = \frac{1}{u_i}\left(\vec{\nabla} u_i \cdot \Delta\vec{x}_\xi\right), \quad \epsilon_{i,\eta} = \frac{1}{u_i}\left(\vec{\nabla} u_i \cdot \Delta\vec{x}_\eta\right), \quad \epsilon_{i,\zeta} = \frac{1}{u_i}\left(\vec{\nabla} u_i \cdot \Delta\vec{x}_\zeta\right), \tag{9}$$

where $\Delta\vec{x}_\xi$, $\Delta\vec{x}_\eta$ and $\Delta\vec{x}_\zeta$ are the vector difference between the mid points of the faces of cell i in the ξ, η, and ζ directions, respectively, and u_i represents the quantity of interest (i.e., density, pressure, or velocity magnitude).

The refinement criteria defined in (9) are evaluated for every cell, i, in each block and the maximum value over all of the cells for each direction in a given block is taken to be the value of the refinement criteria for the block. Using this maximum value and user specified thresholds for both refinement or coarsening, a given block is then flagged to be refined (R), coarsened (C), or to undergo no change (N) as summarized in Algorithm 2 above.

4.2. Conflict checking

The conflict checking procedure essentially revises the list of refinement flags as defined in Section 4.1 to eliminate any cases of refinement or coarsening that are non-permissible. Refinement or coarsening is deemed to be not permissible for the following four reasons: (*i*) the AMR will introduce resolution changes of more than a factor of two between adjacent blocks; (*ii*) the refinement level of a block will exceed user-defined maximum or minimum refinement levels; (*iii*) a block flagged for coarsening does not have a sibling that is also flagged for coarsening or after refinement the sibling/neighbor is not at the same refinement level; and (*iv*) an invalid binary tree connectivity results if the refinement or coarsening is allowed to proceed. In what follows, the treatment of these four non-admissible cases are now reviewed in detail.

First of all, maintaining a maximum difference in the refinement levels of adjacent blocks is done to ensure a smooth transition from refined areas to coarser areas of the mesh, both to make the implementation of the algorithm less complex as well as to avoid potential issues such as numerical accuracy and instabilities. Known as the 2:1 balance, this ratio is commonly enforced in virtually all AMR schemes [96]. Inadmissible AMR associated with this 2:1 balance are the most straightforward conflicts to resolve and involve only the determination of the difference in the refinement levels of adjacent blocks. The three proposed refinement flags of a block $\mathcal{F}_\xi$, $\mathcal{F}_\eta$, $\mathcal{F}_\zeta$ have to be compared to those of its nearest neighbors. If a conflict is found, a preference for refinement is always given over coarsening. Thus, a block is only coarsened if it does not impact the ability of other blocks to be refined.

Secondly, the conflict checking procedure ensures that the new refinement level, l_d, for the block does not exceed user-defined maximum and minimum values for refinement, $l_{\max}$ and $l_{\min}$, respectively, in each direction. This step in the conflict checking is also relatively simple to enforce by merely setting $l_d = \min(\max(l_d, l_{\min}), l_{\max})$ where d corresponds to the logical coordinate directions ξ, η or ζ.

The third step in the conflict checking procedure ensures that a block flagged to coarsen in a particular coordinate direction has an adjacent neighbor that is also flagged for coarsening in that direction. This is referred to as a first sibling

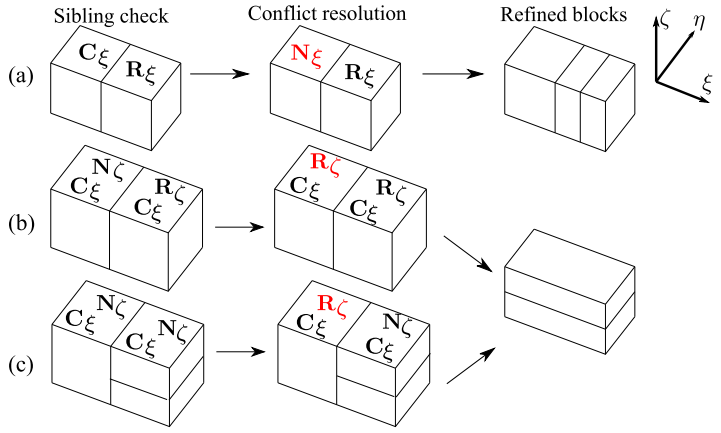


Fig. 5. Example of conflicts resolved through (a) the first sibling check and (b) and (c) via second sibling checks.

check and Fig. 5(a) shows a situation where a block is prohibited from undergoing coarsening since it has no neighbor flagged to coarsen in the same direction. Once it has been established that both blocks are flagged to coarsen in the same direction, one also has to check that both blocks will be at the same refinement level in the directions other than the coarsening direction, before the blocks can be coarsened. An example of this so-called second sibling check is given in Fig. 5(b) and Fig. 5(c). In Fig. 5(b), two neighboring blocks are at the same level of refinement in the ξ and ζ directions. Both are flagged to coarsen in the ξ direction with $\mathcal{F}_\xi = \mathrm{C}_\xi$. Since they have the same level of refinement in that direction, coarsening is allowed. In the ζ direction, one is flagged to refined with $\mathcal{F}_\zeta = \mathrm{R}_\zeta$ and its neighbor is flagged to remain unchanged with $\mathcal{F}_\zeta = \mathrm{N}_\zeta$. This leads to a conflict which is solved by re-setting the value of the refinement flag from N_ζ to R_ζ. Fig. 5(c) is an example of a conflict that arises even if two neighboring blocks have the same flags, C_ξ and N_ζ for the ξ and ζ directions, respectively. Since one block is finer in the ζ direction, coarsening is not permitted. Since refinement is always preferred, the flag of the coarser block in ζ is modified to be R_ζ such that, after the current refinement in ζ, the neighboring blocks will then be at the same level in the ζ direction.

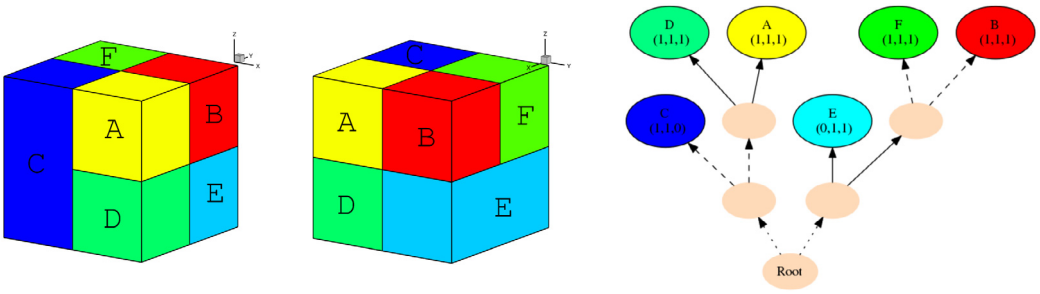
The fourth and final conflict to be resolved is the possibility that the AMR block configuration cannot be represented by a binary tree. This conflict arises only in 3D configurations. The invalid binary trees can result even if all other conflicts have been resolved and are typically the most complex conflict scenarios to resolve. These cases are detected by first ascending the binary tree until a common ancestor node or bridge is reached that is split in the direction of coarsening. Next, all blocks located on the path from the bridge to the searching block are recorded. If any two descendants are at a lower refinement level in both computational directions other than the coarsening direction, the block is prevented from being coarsened.

The overall procedure outlined above is repeated several times until no new conflicts are created by the modifications of the refinement flags. Even though there is no formal proof of convergence for the proposed conflict-checking algorithm, heuristic experiments have shown that no more than three or four repeated passes of the algorithm are required in practice for termination of the process. At the end of the procedure, the grid blocks are updated according to the modified flags.

An example of a sequence of block refinements and coarsenings that results in a block configuration that is not able to be represented by a binary tree data structure is depicted in Fig. 6. In this example, blocks A and B are flagged to be coarsened in the η logical coordinate direction. This check consists in looking for the common node ancestor which is the root. Descending from the root to block B, it is apparent that block E is at a lower level in the ξ direction and from the root to block A, block C is at a lower level in the ζ direction. Thus, blocks A and B are prevented from coarsening in the η direction and a new flag $\mathcal{F}_\eta = \mathrm{N}_\eta$ is set for both grid blocks.

4.3. Determination of newly created grid blocks

Once all conflict checkings have been resolved and a revised list of flags has been obtained, the third step of the proposed AMR scheme is to actually proceed with the coarsening and refinement of all of the flagged grid blocks. By definition, the binary tree data structure supports refinement of grid blocks in just a single coordinate direction at a time, e.g., each block has only two children and, therefore, each instance anisotropic AMR is performed, a scheme could be developed to select a single refinement or coarsening direction for simplicity. However, such a procedure has been identified to be ill-suited for application to unsteady flow problems as discussed by Zhang and Groth [63,64], producing “deadlock” situations where blocks cannot be coarsened even after the flow features driving the refinement are no longer present. Therefore, it is necessary to adopt a more flexible approach allowing grid blocks to both refine and coarsen in all flagged directions in a single step. Furthermore, as the second sibling check requires neighboring blocks flagged to coarsen to have the same level of refinement in the directions other than the coarsening direction, the refinement process is done before the coarsening procedure.



(a) Two views of a multi-block mesh resulting from two applications of anisotropic AMR scheme.

(b) Corresponding binary tree with triplets representing levels of refinement in the ξ, η, and ζ logical coordinate directions.

Fig. 6. Illustration of an anisotropic multi-block mesh where blocks A and B are flagged for coarsening in the η direction. If the coarsening occurs, the block topology could not be represented by a binary tree anymore. The dashed arrows represent a split in the ξ direction, dotted arrows correspond to a split in η direction, and solid arrows are for a split in the ζ direction. Each color represents a block. The grid blocks make up the leaves of the binary tree and the peach colored ovals are nodes of the binary tree.

For refinement of the grid, there are seven possible combinations of grid block refinement that can result during a single step refinement: refinement in one direction only (R_ξ, R_η, or R_ζ), refinement in two directions, (R_ξ and R_η; R_η and R_ζ; or R_ξ and R_ζ), and refinement in all directions (R_ξ, R_η, and R_ζ). For each of these combinations of refinement flags, the computational cells of the new grid blocks are first determined and then the binary tree is updated to reflect the modified grid topology. To compute the physical locations of the nodes representing the computational cells of the newly allocated refined grid blocks, either a simple mid-point or grid metrics-based approach can be applied. In the mid-point method, new nodes are inserted into the grid based on the average of the node locations defining each edge, face, or volume of the cell. For refinement in a single direction, each grid cell is divided in half in the direction of refinement, doubling the number of cells. For refinement in two (resp. three) directions, each grid cell is divided into four (resp. eight) cells in the refinement directions.

Mesh refinement based on the mid-point strategy is simple and accurate enough for Cartesian mesh configurations as described in Section 5.2.1 of the numerical results. However, such an approach cannot accurately preserve grid-line curvature or clustering as considered in the simulations shown in Sections 5.2.3 and 5.2.4. For such meshes, grid metrics based approach is used similar to that proposed by Gao and Groth [97,9] for 3D isotropic AMR and Zhang and Groth [63,64] for 2D anisotropic AMR. In this case, the refined node locations, $\vec{x}(\xi+\delta_\xi, \eta+\delta_\eta, \zeta+\delta_\zeta)$, are computed in terms of the coarse nodes using the second order Taylor expansion given by

$$\vec{x}(\xi+\delta_\xi, \eta+\delta_\eta, \zeta+\delta_\zeta) = \vec{x}(\xi,\eta,\zeta) + \delta_\xi \partial_\xi \vec{x} + \delta_\eta \partial_\eta \vec{x} + \delta\zeta \partial_\zeta \vec{x} + \frac{1}{2}(\delta_\xi^2 \partial_\xi^2 \vec{x} + 2\delta_\xi \delta_\eta \partial_{\xi\eta}^2 \vec{x} + \delta_\eta^2 \partial_\eta^2 \vec{x} + 2\delta_\xi \delta_\zeta \partial_{\xi\zeta}^2 \vec{x} + \delta_\zeta^2 \partial_\zeta^2 \vec{x} + 2\delta_\eta \delta_\zeta \partial_{\eta\zeta}^2 \vec{x}) + \mathcal{O}\left(\delta_\xi^3, \delta_\eta^3, \delta_\zeta^3\right), \tag{10}$$

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. 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.

The actual solution information from a parent block to a child block in the refinement process is transferred via prolongation. In the current study, a lower-accuracy solution prolongation procedure is adopted in which the cell-averaged conserved solution of the parent coarse cells, $\mathbf{U}_c$, is assigned directly to the cell-averaged values of the overlapping sibling fine cells, $\mathbf{U}_f$, i.e.,

$$\mathbf{U}_f = \mathbf{U}_c\,. \tag{11}$$

Following the creation of the new grid blocks, the binary tree is updated to reflect the grid block refinements.

Coarsening of the AMR mesh is handled a similar single step procedure, allowing grid blocks to coarsen in multiple coordinate directions in just one step. The value of the conserved solution vector in the new coarse cell, $\mathbf{U}_c$, is then evaluated via a restriction procedure in terms of the area-weighted average of the overlapping fine cell solutions given by

$$\mathbf{U}_c = \frac{1}{V_c}\sum_{n=1}^{N_{f\in c}} \mathbf{U}_n V_n\,, \tag{12}$$

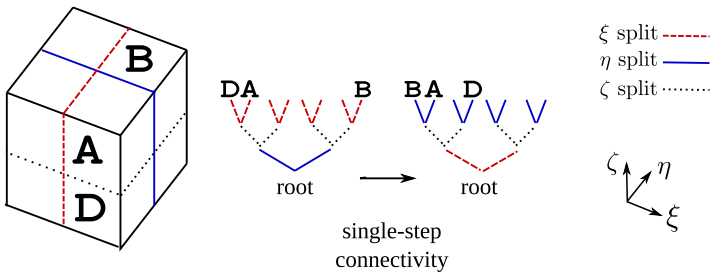


Fig. 7. Illustration of one-step connectivity rearrangement: (left) multi-block domain with blocks A and B flagged for coarsening in the ξ direction which do not share a common parent, (right) binary tree showing the connectivity rearrangement required to allow A and B to share the same parent.

where $\mathbf{U}_n$ and V_n are the fine cell average solution and volume, V_c is the coarse cell volume, and $N_{f\in c}$ is the number of fine cells overlapping the coarse cell. Note that the computation of the node locations of the coarse grid mesh is straightforward and is simply achieved by merely removing every second grid node in the coordinate direction(s) flagged for coarsening.

4.4. Connectivity rearrangement

As pointed out by Ham [20], use of a hierarchical data structure leads to a disadvantage for anisotropic AMR. Coarsening is not always permitted unless it is the exact reverse of the last mesh refinement procedure, implying that two blocks flagged to coarsen must have a common parent. Not allowing this more general type of coarsening greatly reduces the flexibility and efficiency of anisotropic AMR, especially for unsteady simulations. To overcome this limitation, a connectivity rearrangement algorithm has also been developed herein and is now described.

Zhang and Groth [63,64] observed that while the refined mesh is unique, the corresponding binary tree is not. Connectivity rearrangement can therefore be used to re-form the binary tree refinement history in order to allow blocks that are flagged for coarsening to have the same parent. This connectivity rearrangement works by ascending the binary tree and identifying a common ancestor node refined in the coarsening direction. This node is defined as the bridge. If the grandparent is the bridge, then the refinement direction of the bridge is swapped with the refinement direction 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 for the parents of the blocks to be coarsened until the bridge is found. The split direction is then propagated down the tree to the desired location, allowing grid coarsening. An example of this proposed single-step connectivity rearrangement is shown in Fig. 7.

Unique to 3D anisotropic AMR and therefore not treated in the 2D anisotropic approach of Zhang and Groth [63,64], a more complex situation can arise in which blocks flagged for coarsening do not share a common parent and additionally, the parents are at different levels of refinement. To resolve this more complex connectivity problem, a two-step connectivity rearrangement procedure is then proposed here. This two-step procedure consists of a first stage which re-forms the binary tree in order to force the parents to have the same refinement levels. This step is referred as an intermediate rearrangement. In a second stage, the single-step connectivity rearrangement described above is then applied to arrive at a re-order tree permitting coarsening. The intermediate rearrangement process can be generalized and summarized as follows for two blocks, A and B, flagged for coarsening with parents at different refinement levels:

1. ascend the tree from block A until a bridge is found with the same split direction as the parent of block B;
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 block descended to has no children and is at a lower refinement level than A in any direction, start from step 1 with block B switched with 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.

An example of the two-step connectivity rearrangement is provided in Fig. 8. First, a single-step connectivity rearrangement is attempted to allow blocks A and B to coarsen. This rearrangement fails because the refinement levels of the parents are different. Next, an intermediate connectivity rearrangement is completed to force the parents of blocks A and B to share the same refinement levels. Following the intermediate rearrangement, a single-step connectivity rearrangement is performed to create a common parent for blocks A and B. While the intermediate connectivity rearrangement adds additional logic to the coarsening process, the impact on computational cost is minimal as connectivity rearrangement only occurs during coarsening and only for the blocks requiring coarsening. Once the conflict checking procedure has been performed and the binary tree updated, the coarsened grid blocks with updated solution information can be determined based on the refinement/coarsening flags.

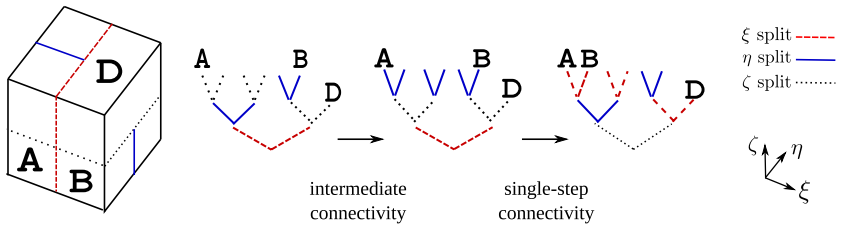


Fig. 8. Illustration of two-step connectivity rearrangement: blocks A and B arise from parent blocks that are at different levels, causing the one-step rearrangement to fail. Following the intermediate rearrangement, the connectivity of blocks A and B can be rearranged via the single-step rearrangement procedure.

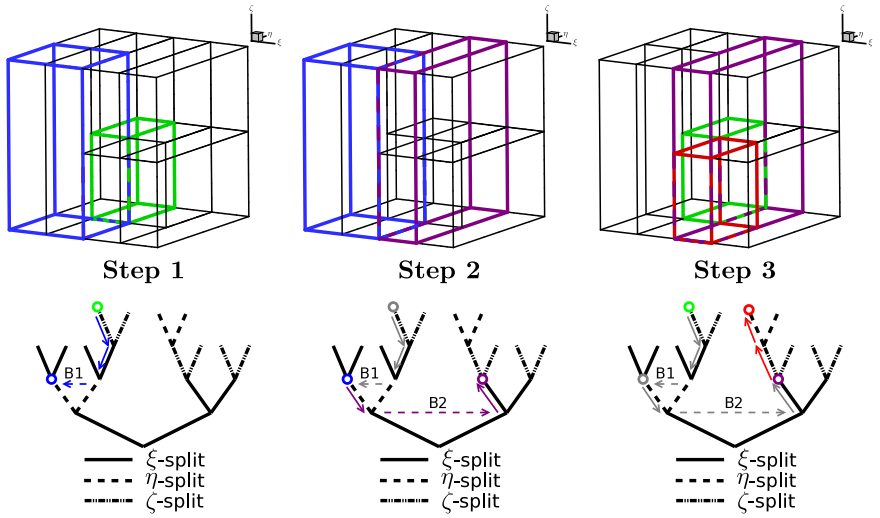


Fig. 9. Example of a neighbor search in the direction $(\xi, -\eta, 0)$ for the grid block shown in green to the neighboring block shown in red. Paths B_1 and B_2 represent the bridges.

4.5. Determination of grid-block connectivity

Following each step in the adaptation of the mesh, the connectivity between the grid blocks is updated and stored so that solution information can be transferred readily between adjacent/neighboring blocks in the mesh. An efficient and fast computation of the connectivity information is carried out here by taking advantage of the binary tree structure. In particular, nearest neighbors may be found by following the stored refinement history. The grid-block connectivity computation algorithm adopted and further developed here is based on the efficient strategy of Zhang and Groth [63,64] proposed for 2D anisotropic AMR scheme. The latter follows the tree-based neighbor search algorithms proposed by De Zeeuw [13] and Coirier [18].

A neighbor search begins by ascending the binary tree toward the root node until a parent is found that is refined in the search direction. This parent block is defined as the bridge, which is a common ancestor of the searching block and the neighbor block. If the root node is reached before finding a bridge, the neighbor block either does not exist or belongs to a different root of the binary tree. In the case of the latter, the precomputed connectivity of the multiple root blocks is used to find the appropriate root block of the neighbor. Once the bridge is identified, the tree descension process is initiated.

First, a descending direction is found based on the block's sector information and split direction. For example, a block that is an east sector of its parent and is searching for an east neighbor will have a western descending direction in the ξ direction 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 process is repeated recursively 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 recursively again for a corner neighbor search in the remaining direction starting from the intermediate node of the edge neighbor.

An example of a neighbor block search for the direction $(\xi, -\eta, 0)$ is illustrated in Fig. 9. Starting from the block highlighted in green, the first search looks for a intermediate neighbor in $(0, -\eta, 0)$. From the green node, the binary tree is ascended until a bridge B_1 in the η direction is found. The bridge is crossed, arriving at the intermediate node shown in blue, corresponding to the block highlighted in blue. This is referred as step 1 in Fig. 9. A next search follows from the

intermediate blue node in the ξ direction and the tree is ascended until the bridge, B_2, is found. The tree is then descended towards the intermediate node and stops when a node or block is reached that is at an equal level to the intermediate node in η or ζ. This corresponds to step 2 in Fig. 9. This new node represented in purple in Fig. 9 is called the second intermediate node. In step 3, this node is then descended towards the original search block until the edge neighbor shown in red is found.

Note that the descending steps may require extra logic when the intermediate node/block 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 sectors for nodes at the same refinement level. This may require ascending the tree until a node is found at the same level, before comparing sector information and determining the descending direction.

4.6. Message passing of information between adjacent grid blocks

At interfaces between blocks, solution information must be exchanged between adjacent blocks and is stored in a layer of overlapping cells, called ghost cells. Message passing of the ghost-cell solution content and geometry is performed in an asynchronous fashion with gathered wait states and message consolidation. As mentioned in Section 3, due to the introduction of non-uniform grid blocks, restriction and prolongation of the solution as well as correction of the numerical fluxes are not needed at block interfaces in the present anisotropic AMR formulation, no matter the mesh resolution between the neighboring grid blocks. Moreover, ordering the message passing so as to send first the messages for coarse blocks and followed next by the messages for the fine grid blocks is also not required in the current implementation. These are significant simplifications and make the overall message passing of solution information simpler and much more straightforward.

4.7. Dynamic load-balancing

One of the key issues related to the parallel implementation of AMR schemes on high-performance computing architectures consisting of multiple processors and cores with distributed memory is the dynamic load balancing (DLB) of the simulation, which allows large-scale adaptive applications to be run efficiently. DLB can be classified as either *scratch-and-remap* or *diffusion-based* schemes [98]. In scratch-and-remap schemes, the workload is repartitioned from scratch and then remapped to the original partition. Diffusion-based schemes employ the neighboring information to redistribute the load between adjacent processors such that global balance is achieved by successive migration of workload from overloaded to underloaded processors.

Dynamic load balancing of the proposed anisotropic AMR scheme is achieved here via an approach that belongs to the class of diffusion-based schemes. The proposed DLB procedure is readily scalable since it only requires local information. In particular, it makes use of the multi-block grid structure and a domain decomposition procedure that distributes the grid blocks, which are of equal size and therefore associated with essentially equal computational work, as equally as possible among the processors. During the adaptive mesh process, the load balance is modified during both the refinement and coarsening procedures. During the refinement process, when newly child blocks are created, blocks are taken from a pool of available resources and distributed among the processors. When blocks are coarsened, one of the blocks being coarsened is assigned to the new coarse grid block and the remaining blocks are returned to the pool of available resources for subsequent re-use. The returned blocks are placed at the top of the pool such that they have higher priority and are used first in the next set refinements of the mesh. This helps to maintain an even sharing of the computational work across processors and reduces load-imbalance. In general, increasing the number of grid blocks via mesh refinement tends to improve the distribution of the computational load across the processors whereas mesh coarsening tends to produce greater load imbalance. It would therefore be preferable to start first with the coarsening process such that the load imbalance may be rectified by the refinement. Unfortunately as outlined in Section 4.3, the refinement process must occur prior to the coarsening procedure and any improvements in load balancing offered by refinement is not realized until the next round of AMR. In some cases, the proposed diffusion-based algorithm can take many steps to rectify a large load imbalance in the computational work. For this reason, the proposed AMR scheme has also been implemented using a so-called scratch-and-remap strategy for situations in which the load imbalance exceeds a given threshold. The latter consists of a space-filling curve or Morton-like-ordering strategy [99] which simplifies the partition problem by mapping the n-dimensional space of blocks to a one dimensional space at a relatively low computational cost.

The first phase of the proposed Morton-ordering algorithm is to generate the space filling curve. As also considered by Gao et al. [10] and following the ideas of Aftosmis et al. [19], a recursive post-order traversal of the tree data structure is used to generate a representation of the space filling curve. The proposed approach for generating the space filling curve can be divided into three steps as follows: (1) traversal of first child's tree; (2) traversal of second child's tree; and (3) visit root nodes.

The second phase of the proposed Morton-ordering DLB procedure is to divide the space filling curve into sub intervals, $\mathcal{I}$, each containing the same weighting or number of objects (i.e., grid blocks) so as to achieve an ideal load balancing. As the space filling curve is constructed to map physically close elements or adjacent grid blocks near one another on the curve, the algorithm will assign physically adjacent or physically close grid block to the same processor to reduce interblock communication costs. To illustrate this procedure, Fig. 10 shows the impact of the space filling curve (black line) passing through each of the multi-block grid blocks (colored blocks) on the partitioning of the computational mesh associated with

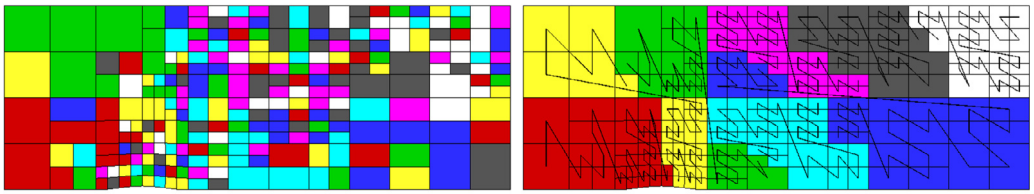


Fig. 10. Partitioning of grid blocks of a multi-block hexahedral mesh (a) before and (b) after the application of the Morton-ordering space-filling curve yielding a more efficient load balancing of blocks on 8 processors. Grid blocks associated with each processor are represented by a unique color and the black line represents the Morton-ordering curve passing through each of the solution blocks of the multi-block mesh.

channel flow over a non-smoothed bump as will be described in Section 5.2.2 to follow. In the figure, each color represents a different processor number. Initially the mesh is composed of just eight root blocks and the simulation is carried out on eight processors with virtually ideal load balancing. Following seven AMR procedures, the partitioning of the mesh across the eight processors is given in Fig. 10(a) using the proposed diffusion-based DLB algorithm. Fig. 10(b) shows the subsequent repartitioning of the grid blocks that is achieved using the proposed Morton-ordering procedure, applied for each root. The latter generally places the neighboring blocks of a given root on the same processor. Once all of the grid blocks are visited, the next root block is considered. The overall resulting distribution of the blocks over the eight processors is well-balanced and as much as possible the neighboring blocks are located on the same processor.

4.8. From anisotropic to isotropic AMR scheme

With minimal effort, the proposed anisotropic AMR scheme as defined in Algorithm 1 above can be made to degenerate to a fully isotropic AMR algorithm. Just two modifications are required. First, stage 1 is modified such that the refinement criteria are no longer directionally based and are such that

$$\epsilon_i = \frac{1}{u_i} ||\vec{\nabla} u_i||_{L^2},$$

where u_i represents any quantity of interest (density, velocity, vorticity) of cell i. A single unique refinement flag, $\mathcal{F}$, is introduced for directing the refinement with $\mathcal{F}_\xi = \mathcal{F}_\eta = \mathcal{F}_\zeta = \mathcal{F}$ where $\mathcal{F}$ take values of R, C, or N corresponding to the grid block being refined in all three logical coordinate directions, coarsened in all three directions, or remaining unchanged.

The second modification needed to yield an isotropic AMR algorithm concerns the conflict checking procedure (stage 2 above) which is reduced to just three cases: (*i*) the AMR will introduce resolution changes of more than a factor of two between adjacent blocks; (*ii*) the refinement level of a block will exceed user-defined maximum or minimum refinement levels; and (*iii*) a block flagged for coarsening does not have all of its seven siblings also flagged for coarsening. The first two conflicts are identical to those of the anisotropic refinement case and the resolution of these conflicts are identical to the procedures outlined in Section 4.2. The third conflict check requires that grid blocks are allowed to coarsen only if the grid block and all of its siblings blocks are flagged for coarsening. If at least one of the sibling is flagged for refine or to remain unchanged, coarsening is prevented and the refinement flags for the blocks are modified such that $\mathcal{F} = \text{N}$. The conflict procedure is finally re-applied to check that these new flags have not introduced new conflicts. Once all conflicts have been resolved, stages 3 to 6 of Algorithm 1 are performed.

It is worth mentioning that using a binary-tree structure for isotropic AMR can be somewhat inefficient in terms of data storage and memory requirements. In particular, the octree hierarchical tree structure Gao and Groth [9] is a more efficient structure when considering just isotropic AMR alone. Indeed, after several isotropic refinements, while the number of leaves (active or computational blocks) of the binary tree and the octree data structures is the same, the total number of nodes in the tree structure is not. Each isotropic refinement of a single grid block involves 17 nodes for a binary tree and just 9 nodes for an octree structure. Nevertheless, the isotropic AMR simulations presented in Section 5 using the binary tree are considered here only for comparison and validation with the simulations results obtained with newly-proposed anisotropic AMR scheme.

5. Numerical results and discussion

In order to evaluate the proposed anisotropic non-uniform block-based AMR approach presented in Sections 3 and 4, predictions of the anisotropic AMR scheme are compared to numerical solutions obtained using both uniform and isotropically refined AMR meshes. Both inviscid and viscous compressible gaseous flows are considered here. Where possible, the numerical predictions are also assessed via comparison to available analytical solutions.

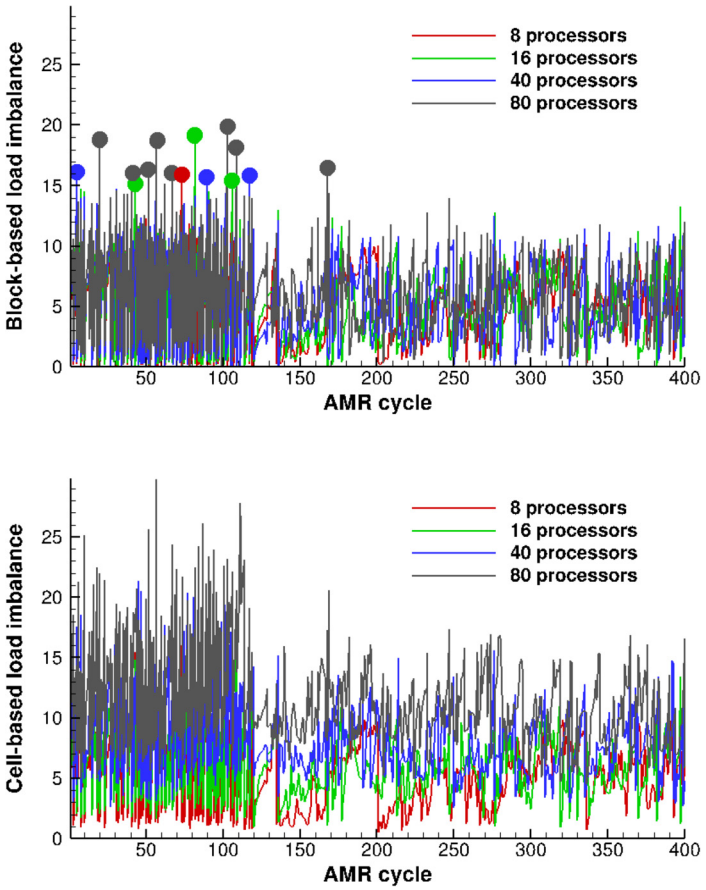


Fig. 11. Comparison of the block-based load imbalance (top), and cell-based load imbalance (bottom) evaluated after each AMR cycle for the unsteady shock-cube problem described in Section 5.2.1 for 8, 16, 40 and 80 processors.

5.1. Performance of the anisotropic AMR scheme

Prior to considering the numerical result for specific flow problems, some measures of the efficiency of proposed anisotropic AMR scheme are first discussed. A common metric for characterizing computational load balance in parallel simulations is the percent imbalance metric, λ, given by [100]:

$$\lambda = \left(\frac{L_{\max}}{\overline{L}} - 1 \right) \times 100, \tag{13}$$

where $L_{\max}$ is the maximum load on any process and $\overline{L}$ is the mean load over all computational processors. The percent imbalance measures the severity of load imbalance with a lower value of the imbalance indicating a better distribution of the computational workload across the available processors. As described in Section 4.7, the proposed diffusion-based and Morton-ordering DLB schemes for the anisotropic AMR scheme attempt to distribute as equally as possible the grid blocks amongst the available processors. The imbalance of the DLB scheme is evaluated within two variables: λ_B and λ_C which are obtained from (13) with the load L representing, the number of grid blocks and the number of computational cells, respectively. Because of the non-uniform nature of the blocks, the block-based measure of imbalance, λ_B, is a somewhat less accurate or coarser evaluation of the DLB scheme while the cell-based measure of imbalance, λ_C, is more accurate. The value of λ_C is expected to be higher than λ_B but since we can not act on the amount of ghost cells per block, only λ_B is used in the DLB scheme for controlling the switch from the diffusion-based DLB approach to the scratch-and-remap strategy of the Morton-ordering method.

The unsteady shock-cube simulation described in detail in Section 5.2.1 is considered here in evaluating the proposed DLB strategy. Imbalances λ_B and λ_C, evaluated after each stage of the anisotropic AMR are depicted in Fig. 11(top) and Fig. 11(bottom) respectively. Imbalances were evaluated using 8, 16, 40 and 80 processors and circles highlight the imbalance obtained before the space filling curve which is applied when the imbalance metrics exceeds 15% (i.e., $\lambda_B > 15\%$). Note that, as for the isotropic block-based AMR schemes of Groth and co-researchers [46,47,9,10,48], the ratio of communication costs to computational work is relatively low for the proposed scheme and thus the anisotropic AMR scheme

affords efficient and scalable implementations for large-scale flow problems involving the use of tens of thousands of processors, provided that the problem size is sufficiently large and there are on the order of 10–50 local grid blocks per processor.

As can be seen in Fig. 11, the imbalance of the proposed AMR is generally less than 15% for 8, 16 and 40 processors and as expected, $\lambda_B < \lambda_C$ throughout the unsteady simulation. The space-filling curve algorithm is applied only once when using 8 processors and only three times for 16 and 40 processors. When 80 processors are used, the space-filling curve is applied 9 times. The imbalance ratios increase as the number of processors increases. This is reasonable because it is more likely that there are fewer blocks per processor and a slight deviation from the mean, $\overline{L}$, represents a higher imbalance percent. At the beginning of the simulation, the total number of blocks is around 2,000 which means that the number of blocks per processor is around 250, 125, 50 and 25 for 8, 16, 40 and 80 processors, respectively. The total number of blocks increases as the simulation progresses in time and after 200 AMR cycles, the number of blocks is around 4,000, which means that there are approximately 500, 250, 100 and 50 local blocks per processor for 8, 16, 40 and 80 processors, respectively. After the 200 AMR cycles, the space-filling curve is applied only once in the case of 80 processors. These findings show that to obtain a good load-balance from the diffusion-based approach, around 10–50 blocks are required on each processor. Furthermore, the load imbalance can be further controlled and maintained by periodic application of the space-filling curve procedure. Note that, after each application of the space-filling curve scheme, the imbalance depicted in Fig. 11 drops below 2–3% with on the order of 50 blocks per processor.

To complete the DLB performance study, it is interesting to evaluate the computational time associated with the anisotropic AMR scheme itself compared to the computational time for solving the governing equations via the proposed finite-volume scheme. This comparison will of course depend on the problem and nature of the simulation. For the unsteady shock-cube simulation described in Section 5.2.1, the AMR procedure is applied every 15 time steps to allow accurate tracking of the shock wave. In this case, the computational time for the anisotropic AMR represents just 6%, 8%, and 12% of the overall total simulation time when using 8, 16, and 64 processors, respectively. For the steady channel-flow simulation of Section 5.2.2, 256 processors were used and the AMR procedure was applied 6 consecutive times after fully converged solutions were obtained on each refined mesh. The computational time for the AMR in this steady case then represents just 0.3% of the total computational effort.

5.2. Numerical results for the Euler equations

Numerical results for several inviscid flow problems governed by the Euler equations will now be discussed. Results are described for an unsteady three-dimensional shock-cube problem and steady-state supersonic channel flow with a bump. Results obtained using a cubed-sphere mesh [48,50] are also presented for a steady supersonic radial outflow problem and steady supersonic flow past a sphere. The numerical results for all four of these inviscid flow cases were obtained using the explicit time-marching schemes of Section 2.3 and, in each case, the anisotropic refinement criteria was based on the density gradient. Additionally, an accuracy assessment is performed for this 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 to demonstrate the validity of the algorithm for solving both steady-state and time-varying problems. Comparing the results with 3D isotropic AMR illustrates the potential benefits of the anisotropic approach in terms of computational cost savings.

5.2.1. Unsteady shock-cube flow

A so-called shock-cube flow problem is first examined and provides an excellent evaluation of the proposed anisotropic block-based AMR algorithm for unsteady inviscid 3D flows, requiring high mesh resolution to capture accurately non-linear wave interactions and effective mesh adaptivity to the deal with the unsteady nature of the solution. Due to the highly anisotropic nature of the proposed shock-cube problem considered here, it also provides a very good evaluation of the capabilities of the anisotropic AMR scheme. The initial conditions for the shock-cube problem examined here are taken to be standard atmospheric conditions $\rho = 1.225$ kg/m3, $p = 101325$ Pa, $\vec{V} = 0$ m/s, and $\gamma = 1.4$ for $x < 0$, $y < 0$, and $z < 0$ and eight times standard atmospheric density and ten times standard atmospheric pressure elsewhere within cube of unit size with $-1/2 \leq x \leq 1/2$, $-1/2 \leq y \leq 1/2$, and $-1/2 \leq z \leq 1/2$. Reflection boundary conditions are imposed on all six boundaries of the computational domain and the second-order, two-stage, Runge-Kutta, time-marching scheme was used to integrate the solution forward in time.

Prior to the initiation of the simulation, an initial mesh containing a single grid block consisting $8 \times 8 \times 8 = 512$ computational cells was refined based on the initial solution six times, with the initial solution re-imposed each time, in order to arrive at an initial mesh that accurately resolved the discontinuity between the two initial flow states. A maximum refinement level of seven was assigned for the simulation. Following this initial refinement of the mesh, the anisotropic AMR scheme was applied every 15 time-steps until a maximum simulation time of $t = 0.75$ ms was reached. The corresponding anisotropic adapted grid blocks and predicted density contours for simulated times $t = [0.05, 0.25, 0.75]$ ms are shown in Fig. 12.

The numerical predictions of the anisotropic AMR scheme are compared in Fig. 13 to those obtained using both a uniform mesh and the related isotropic AMR scheme detailed in Section 4.8 using the same initial conditions and parameters. Throughout the simulation, it can be seen that anisotropic AMR tracks the complex evolving solution well. At $t = 0.05$

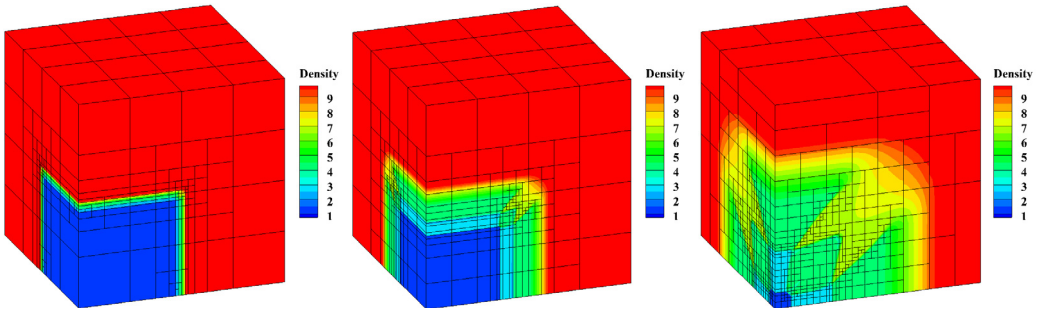


Fig. 12. Predicted distribution of the gas density (kg/m^3) at three different simulated times $t = [0.05, 0.25, 0.75]$ ms obtained using an anisotropic AMR scheme.

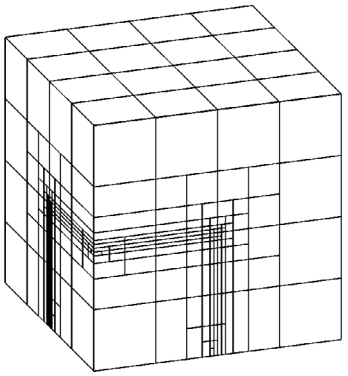
ms, there are large anisotropic features in the solution which are efficiently represented by the anisotropic AMR using blocks with high aspect ratio cells, resulting in a reduction of 85% in the size of the mesh in terms of the number of cells compared to that of the equivalent isotropic AMR solution (same level of refinement, virtually equivalent accuracy). As time evolves, the predicted unsteady wave solutions become less aligned with the grid line orientation of the computational blocks and the anisotropic AMR method results in a more isotropically refined mesh in some regions of the flow. Even though the regions for anisotropic refinement are reduced, the anisotropic procedure still offers a mesh size reduction of 53% compared to the isotropic approach at $t = 0.25$ ms. Finally, at $t = 0.75$ ms, the solution has moved further from being aligned with the mesh. Nevertheless, the proposed anisotropic refinement scheme still achieves a 45% reduction in the overall size of the mesh. Note additionally that the anisotropic and isotropic AMR predictions of the shock-cube problem are also both compared in Fig. 13 to the numerical solution obtained using a fixed uniform mesh based on seven uniform refinements of the original initial mesh. The latter was composed of a total number of 262,144 grid blocks and 134 million cells. In comparison, the anisotropic mesh contains just 1,260 blocks and 0.645 million cells at $t = 0.05$ ms and 5,079 blocks and 2.6 million cells at $t = 0.75$ ms. Additionally, the predicted density profiles along the diagonal of the domain, $[(-1/2, -1/2, -1/2), (1/2, 1/2, 1/2)]$, at time $t = [0.05, 0.25, 0.75]$ ms are also compared in the figure and there is very good agreement between the anisotropic and uniform mesh predictions, illustrating the rather large computational savings offered by the anisotropic AMR approach for this unsteady flow problem.

5.2.2. Steady supersonic channel flow past a bump

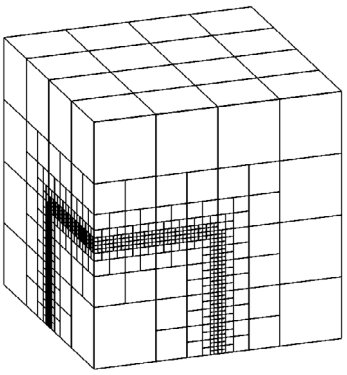
In order to demonstrate the applicability of the proposed anisotropic AMR to the solution of steady inviscid flow problems, supersonic flow over a smooth bump in a fully 3D channel is considered next. The supersonic channel flow problem consists of a rectangular cross-section channel of dimension [5.5 m $\times$ 2 m $\times$ 1 m] with a smoothed circular bump along the bottom wall of the channel section. Air with a flow Mach number of M = 1.4 and at standard atmospheric density and pressure conditions of $\rho = 1.225$ kg/m^3 and $p = 101.325$ kPa, respectively, enters the channel through the left boundary and exits through the right side of the domain. The ratio of specific heats for the air of $\gamma = 1.4$ is assumed.

As the channel flow is two-dimensional in nature, the initial mesh consists of just 8 grid blocks (4 grid blocks along the length of the channel, 2 blocks along the height of the channel, and a single block along the out-of-plane direction) with each block containing $8 \times 8 \times 2$ cells. Reflection boundary conditions are imposed on the lower boundary containing the smooth bump and the top and sides of the channel. The supersonic inflow is fixed at the inlet and constant extrapolation boundary conditions are imposed at the outlet boundary of the channel. Numerical solutions were obtained using both the proposed anisotropic AMR procedure and the reference isotropic approach. The AMR was limited to a maximum of six refinement levels for each of the coordinate directions. Steady-state solutions were obtained by advancing the solution in time until a converged time-invariant solution is achieved on each AMR mesh. The multi-stage optimally smoothing scheme with local time-stepping and a CFL number of 0.5 was used for the temporal integration.

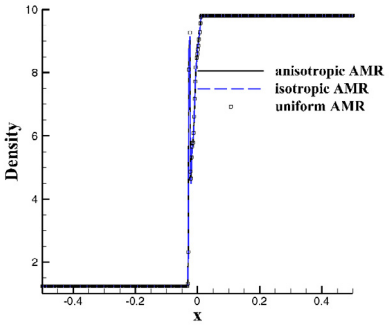
Fig. 14 provides a summary of the results for the steady supersonic channel flow problem. The final anisotropically refined multi-block AMR mesh after 6 levels of mesh refinement, showing the grid blocks (top left panel) is compared to the equivalent isotropically refined mesh (top right panel). The predicted flow field density distribution is also shown in Fig. 14 (bottom left panel) along with the predicted density profile along $y = 1/4$ m (bottom right panel of Fig. 14). The latter compares the predicted profiles of the anisotropic and isotropic AMR schemes. While excellent agreement can be observed between the solutions of the anisotropic and isotropic approaches, the 3D anisotropic AMR procedure does not introduce additional resolution and un-needed grid blocks in the out-of-plane direction (the refinement is two dimensional in nature and the number of block in this direction remains equal to one) as should be expected for an essentially two-dimensional flow, whereas the less efficient 3D isotropic approach introduces a large number of grid blocks and cells in the out-of-plane direction in order to achieve a similar spatial resolution in the two-dimensional plane of the channel. The



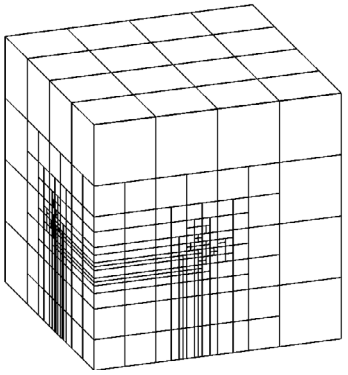
(b)
Anisotropic AMR
(1,260 blocks)



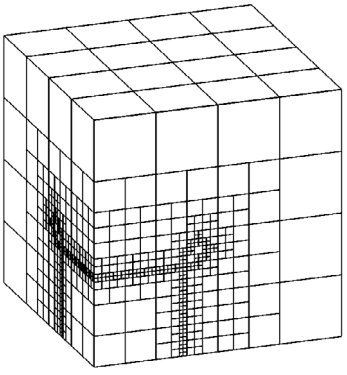
(c)
Isotropic AMR
(14,057 blocks)



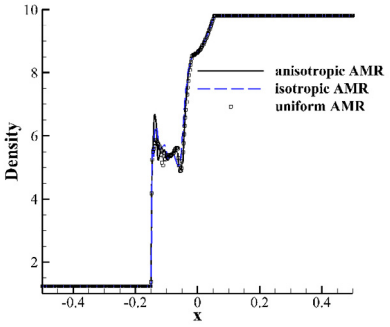
(d)
Diagonal density, $t = 0.05$ s



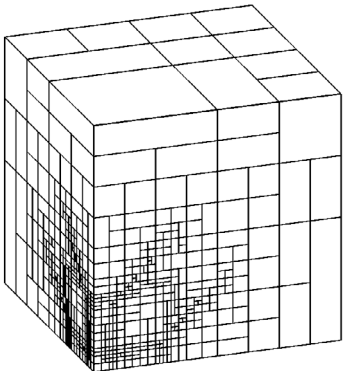
(b)
Anisotropic AMR
(2,034 blocks)



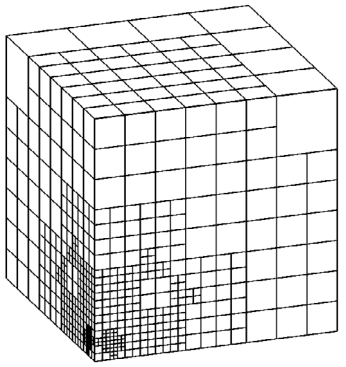
(c)
Isotropic AMR
(10,116 blocks)



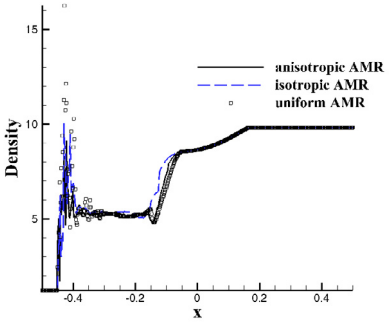
(d)
Diagonal Density profile,
$t = 0.25$ s



(b)
Anisotropic AMR
(5,079 blocks)



(c)
Isotropic AMR
(7,554 blocks)



(d)
Diagonal density profile,
$t = 0.75$ s

Fig. 13. Anisotropic (left) and isotropic (middle) grid blocks making up multi-block computational mesh at three different simulated times $t = [0.05, 0.25, 0.75]$ ms along with comparisons of the predicted density profile along the diagonal of the cube, $[(-1/2, -1/2, -1/2), (1/2, 1/2, 1/2)]$, obtained using a fixed uniform mesh and the isotropic and anisotropic AMR schemes (right).

3D isotropic AMR schemes results in a computational mesh consisting of 47,223 blocks and more than 6 million cells. In contrast, the anisotropic AMR scheme results in a multi-block grid consisting of only 1,255 blocks and 160,640 cells (i.e., the mesh is reduced by more than 97%). In terms of computational time, 256 processors were used with a total computational time of 170 minutes to obtain the anisotropic AMR simulation results while 2,048 processors were needed with more than 12 hours of computational time to obtain the isotropic AMR predictions.

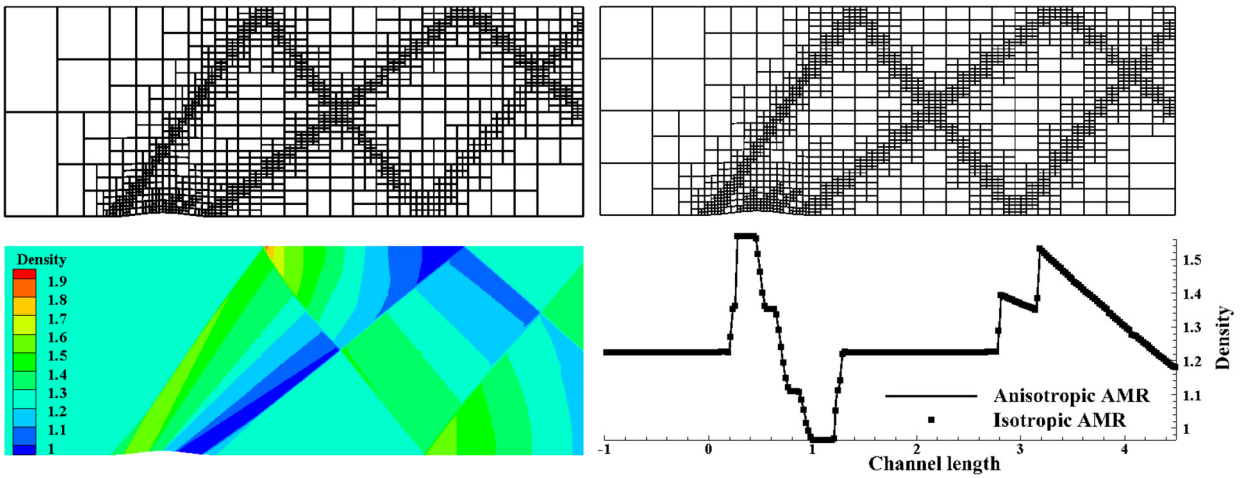


Fig. 14. Predictions of M = 1.4 steady supersonic channel flow past a bump showing the final anisotropic AMR mesh (top left, 1,255 grid blocks, 160,640 cells) after 6 levels of AMR compared to the equivalent isotropic AMR mesh (top right, 47,223 grid blocks, 6 million cells) as well as the predicted density distribution of the anisotropic AMR scheme (bottom left) and a comparison of the predicted density profiles along the line $y = 1/4$ m provided by the anisotropic and isotropic AMR schemes (bottom right).

5.2.3. Steady supersonic spherical outflow

The application the proposed anisotropic AMR to the solution of steady inviscid flow problems is further examined here by considering steady supersonic spherical outflow on a 3D cubed-sphere mesh for which there exists an analytical solution. The spherical computational domain for the problem is represented by the cubed-sphere multi-block mesh originally proposed by Ronchi et al. [101] and more recently implemented within an isotropic block-based AMR approach by Ivan et al. [48,50]. The initial or baseline cubed-sphere multi-block mesh consists of six sectors or root grid blocks connected with edges degeneracies, forming an inner hollow sphere and an outer spherical shell. The anisotropic AMR procedure proposed herein is readily applicable to cubed-sphere meshes and is therefore examined here.

In this first investigation of the proposed enhanced block-based anisotropic AMR for cubed-sphere meshes, supersonic spherical flow of air with $\gamma = 1.4$ enters a spherical shell domain at an inner radius of $R_i = 1$ m and subsequently exits the domain at an outer radius of $R_o = 4$ m. The velocity of the inflowing air is purely radial with a radial component, V_r, subsequently undergoes a supersonic expansion process, and then exits through the outflow sphere surface. The inflow flow properties are fixed with a flow density, $\rho_i = 10$ kg/m^3, radial component of velocity, $V_{r,i} = 4.5$ m/s, and pressure, $p_i = 26$ Pa. The analytical solution to this spherical outflow problem is described by Ivan et al. [48] and the solution at any radial location, r, satisfies the following equation

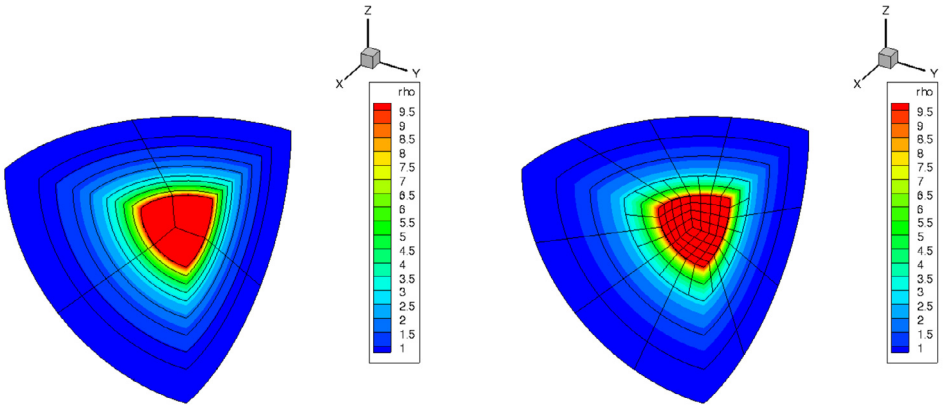
$$C_3 - \frac{1}{r^2 V_r \left[\left(C_2 - V_r^2 \right)^{\frac{1}{\gamma - 1}} \right]} = 0 , \tag{14}$$

where the constants C_2 and C_3 depend on the inflow conditions and are given by

$$C_2 = \frac{2\gamma}{\gamma - 1} \frac{p_i}{\rho_i} + V_{r,i}^2 , \quad C_3 = \frac{1}{\left(\frac{2\gamma}{\gamma - 1} \frac{p_i}{\rho_i}^{\frac{1}{\gamma - 1}} \right) R_i^2 V_{r,i}} . \tag{15}$$

The initial multi-block hexahedral mesh for this case consisted of a uniformly refined cubed-sphere mesh with 24 blocks, each containing $32 \times 32 \times 16$ cells. Constant extrapolation boundary conditions were applied at the outflow boundary at $R_o = 4$ m. Six cycles or levels of adaptive mesh refinement were then applied for which converged steady solutions were obtained on each consecutively adapted mesh using the multi-stage optimally smoothing scheme with local time-stepping. Numerical results were obtained for both the newly-proposed anisotropic scheme and the reference isotropic approach.

The predicted flow density and associated cubed-sphere meshes obtained using both the anisotropic and isotropic AMR methods on the finest adapted meshes are depicted in the quarter slices of the spherical solution domain shown in Figs. 15(a) and 15(b), respectively. It can be observed that the solution varies only in the radial direction for this case and that the anisotropic AMR scheme is able to exploit this feature, refining only in the radial direction, while isotropic AMR approach tends to refine the mesh in a more uniform fashion. As a result, there is a large reduction of 95% in the total cell count for the anisotropically refined cubed-sphere mesh compared to that of the isotropic method. To assess whether this large reduction in the total cell count also corresponds to a significant reduction in solution error, the L_1 and L_2 norms of the solution error in density were also determined for each refined mesh, based on the exact analytical solution given



(a) Anisotropic AMR (54 blocks) (b) Isotropic AMR (1,056 blocks)

Fig. 15. Solutions of the steady supersonic spherical outflow problem showing the predicted distributions of the flow density and grid blocks of the cubed-sphere meshes on quarter sectors of the spherical domain obtained with the anisotropically and isotropically refined multi-block AMR meshes.

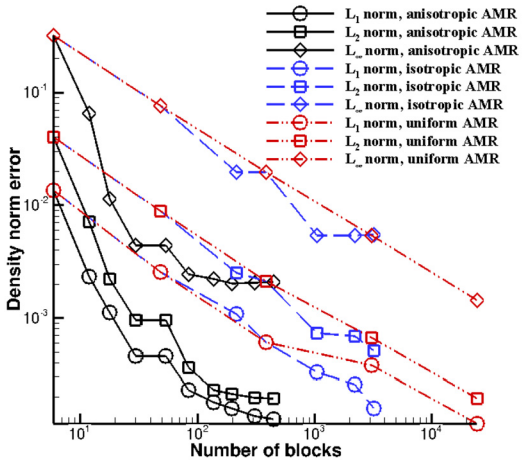


Fig. 16. Solutions of the steady supersonic spherical outflow problem showing the predicted L_1 and L_2 norms of the solution error in density as a function of the number of grid blocks in the cubed-sphere mesh comparing the result for the anisotropically-refined AMR mesh (black solid lines) to those of the isotropically-refined AMR mesh (blue dashed lines) as well as uniformly refined mesh (red dashed dotted lines).

above. The resulting error norms for density as a function of the number of grid blocks in the cubed-sphere mesh is provided in Fig. 16. Results for a uniformly refined mesh are also shown. From this convergence plot it is evident that the third anisotropically-refined mesh provides a solution accuracy better than that of the fifth isotropically-refined mesh and has a total number of grid blocks that is less than that of first isotropically-refined mesh. This again corresponds to a mesh size savings of about 97%. Furthermore, the results of the isotropically-refined cubed-sphere mesh based on the gradient-based refinement criteria can be seen to be similar to those of simple uniform refinement.

It should however be noted here that the gradient-based or physics-based methods used to drive the mesh refinement in the proposed anisotropic AMR scheme do not necessarily directly respond to errors in the numerical solution and this can hamper grid convergence and error control as the mesh refinement proceeds. This is a limitation of the proposed scheme and is apparent in the error convergence results of Fig. 16, which indicate a gradual stalling of the anisotropic AMR scheme as the refinement proceeds. In particular, the high gradients are purely radial in nature for the outflow problem but are not necessarily associated with high solution error (in fact, refinement in the azimuthal direction is required to reduce solution errors), eventually resulting in ineffective mesh refinement. While various additional heuristic strategies can be considered to remedy this issue, the application of output-based or adjoint-based error estimation techniques based on variational principles would enable the use of formal error estimates for directing mesh adaptation [90,91,102,92,93,103,94] and will be the subject of future follow-on studies.

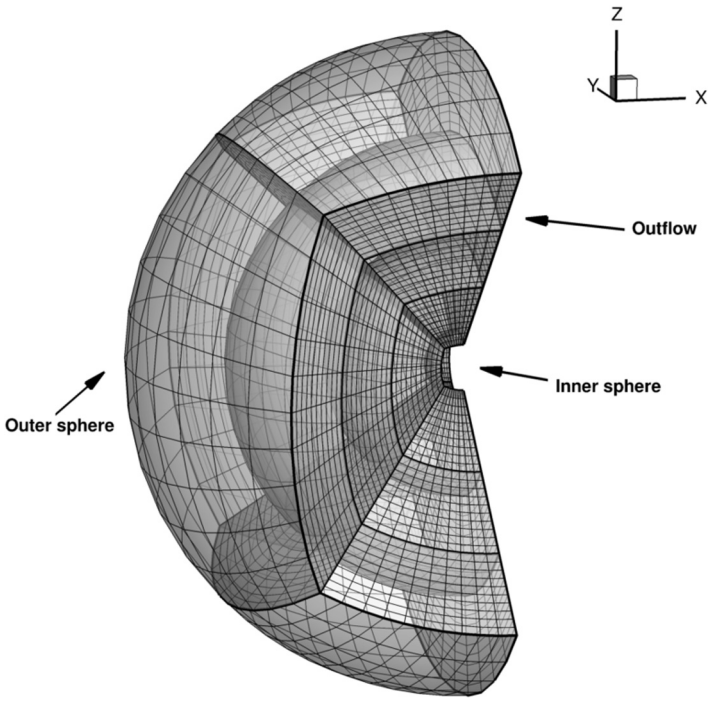


Fig. 17. Initial 5-sector, 15-block, cubed-sphere mesh for M=2 steady supersonic inviscid flow past a sphere.

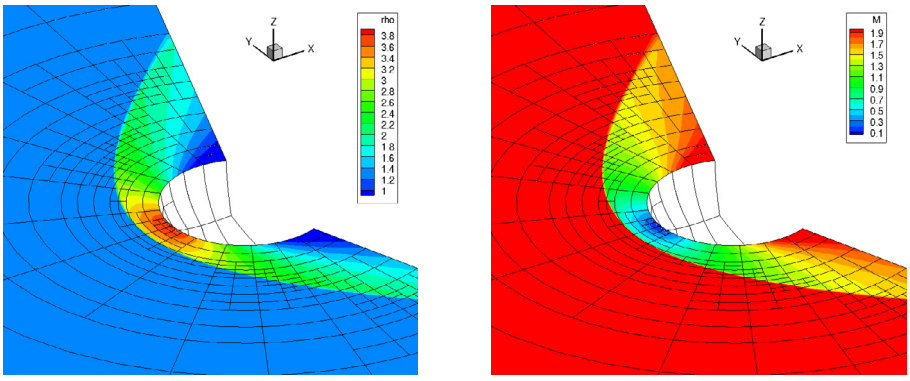
5.2.4. Steady supersonic flow past a sphere

Steady supersonic inviscid flow past a sphere is considered next, again using cubed-sphere based multi-block meshes. The supersonic flow of interest consists of a half sphere of radius of $r = 1$ m with a far-field boundary at $r = 10$ m as shown in Fig. 17. The cubed-sphere meshes for this test problem were constructed to have only 5 sectors so as to reduce the problem size, with 3 blocks stacked in the radial direction for each sector. This resulted in an initial multi-block hexahedral mesh consisting of 15 grid blocks, each containing $10 \times 10 \times 10 = 1,000$ cells, and a total of 15,000 computational cells. The supersonic free-stream air with $\gamma = 1.4$ enters the domain through the upstream far-field boundary (i.e., in the positive x direction) with $p = 101.325$ kPa, $\rho = 1.225$ kg/m^3, and M=2 and exits through the downstream far-field outflow boundary. Fixed solution values were applied at the upstream far-field inflow boundaries and constant extrapolation boundary conditions were imposed at the downstream outflow boundaries, as the outflow remains supersonic. Standard reflection boundary condition were imposed on the surface of the inner sphere. The multi-stage optimal smoothing scheme with local time-stepping was again used to obtain steady solutions for this problem and the HLLE approximate Riemann solver was used in the numerical evaluation of the solution fluxes.

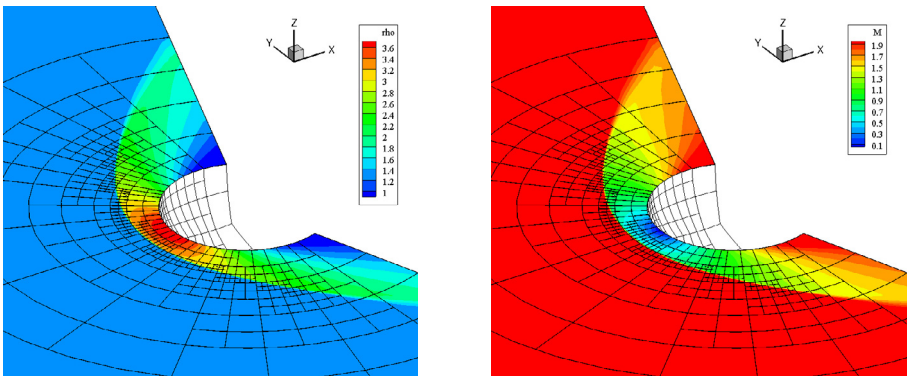
Fig. 18 shows the predicted solutions of the density and Mach number obtained after 5 cycles/levels of refinement for the proposed anisotropic block-based AMR scheme compared to the results of the equivalent isotropic approach. In both cases, the adapted mesh is aligned well with the resulting bow shock near the stagnation streamline but loses some of this alignment moving away from the center line towards the outer regions of the flow where the bow shock is weaker in strength. Nevertheless, the anisotropic AMR scheme again provides a well resolved bow shock yet results in a significantly reduced computational mesh for this case (1,625 grid blocks and 1.625 million cells compared to 12,062 grid blocks and 12.06 million cells for the isotropic AMR method for an 86% reduction in mesh size) due to far fewer refined blocks introduced in the angular directions (the resolutions of the two meshes is similar radially). Moreover, the comparisons of the predicted center-line density and Mach number distributions corresponding to the stagnation streamline for the flow of Fig. 19, which compares the anisotropically- and isotropically-refined results, indicate that the two sets of results are virtually indistinguishable, including the predicted location of the bow shock and post-shock stagnation flow conditions.

5.3. Numerical results for the Navier-Stokes equations

Numerical results for solutions of the Navier-Stokes equations obtained using the implicit time-marching scheme outlined in Section 2.3 are now considered. Four different problems are examined: steady laminar boundary-layer flow past a flat plate; steady subsonic laminar flow over an airfoil; steady subsonic laminar flow past a circular cylinder; as well as steady supersonic laminar flow over an airfoil. Again, the performance of the proposed anisotropic AMR scheme is assessed via comparisons to either available analytical solutions or predicted results obtained with the equivalent isotropic AMR scheme where helpful. For all Navier-Stokes solutions, a combination of two refinement criteria were used based on the gradients of



(a) Predicted density (left) and Mach number (right) distributions, anisotropic AMR (1,625 grid blocks, 1.625 million cells)



(b) Predicted density (left) and Mach number (right) distributions, isotropic AMR (12,062 grid blocks, 12.06 million cells)

Fig. 18. Predictions of M=2 steady supersonic inviscid flow past a sphere showing comparisons of the numerical solutions and adapted cubed-sphere grids obtained using both the anisotropically-refined and equivalent isotropically-refined AMR strategies (grid blocks are shown outlined in black).

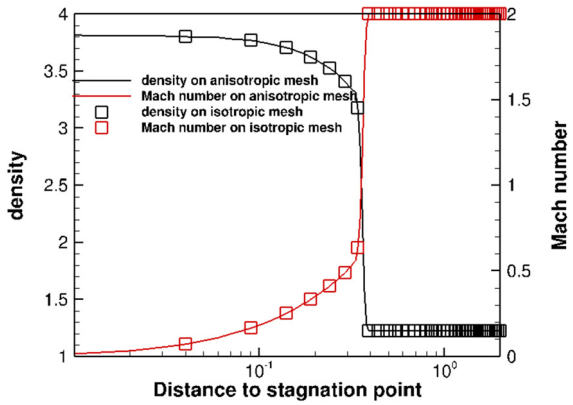


Fig. 19. Comparisons of predicted center-line distributions of the flow density and Mach number associated with the stagnation flow streamline M=2 steady supersonic inviscid flow past a sphere obtained using both the anisotropically-refined (straight lines) and equivalent isotropically-refined (symbols) AMR strategies.

density and velocity magnitude. When the multiple criteria are used, mesh refinement occurs if one or more criteria meet the user-defined thresholds and grid block coarsening takes place only if all criteria satisfy the coarsening thresholds.

5.3.1. Steady subsonic laminar boundary-layer flow past a flat plate

Numerical results for the steady laminar flow past a flat plate are first considered and compared to the classical incompressible solution of Blasius [104]. For the case of interest, the free-stream Mach number and Reynolds num-

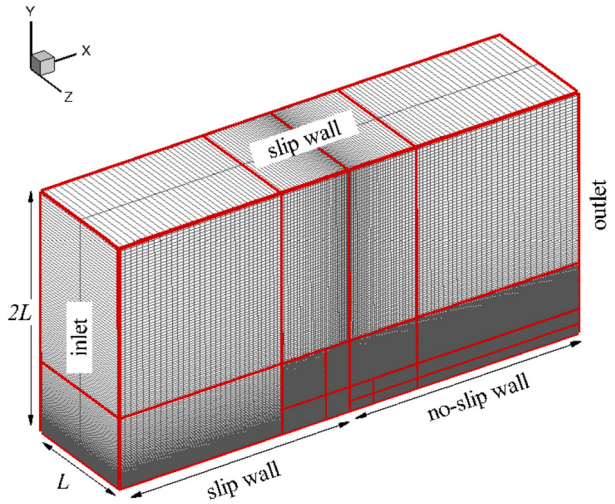


Fig. 20. Final anisotropically-refined multi-block computational mesh used in the simulation of M=0.2 and Re=9,318.92 steady subsonic laminar flow over a flat plate. The final mesh consists of 17 grid blocks, each containing $32 \times 64 \times 2$ cells, for a total of 69,632 cells.

ber based on plate length ($L = 0.002$ m) are M_∞=0.2 and Re_L=9,318.92, respectively. A rectangular prism domain of $[-2L, 2L] \times [0, 2L] \times [0, L]$ was used for obtaining solutions to this problem. The corresponding domain and boundary conditions for the flat plate flow are illustrated in Fig. 20. The far-field, sides, and upstream of the plate are treated using conventional reflection/slip-flow boundary condition, while at the plate, for $x = [0, 2L]$ and $y = 0$, standard no-slip boundary conditions are enforced. A uniform fixed velocity profile is specified on the inlet plane ($x = -2L$) and zero-gradient boundary conditions are applied for the fluid velocity on the outlet plane ($x = 2L$). The pressure is held fixed and equal to the freestream value at the outlet. The initial mesh consisted of two blocks of $32 \times 64 \times 2$ cells for a total of 8,192 computational cells and a uniform flow of air having the free-stream conditions was assumed for the initial conditions. Two levels of AMR refinements were performed and the solution was fully converged on each level of the mesh, including the initial mesh. The steady flow solutions were obtained using the inexact Newton's method described previously for which the Newton iterations were stopped when the non-linear residual was reduced by 10 orders of magnitude (i.e., $\epsilon = 10^{-10}$).

The final computational mesh and mesh block topology obtained using the proposed anisotropic block-based AMR scheme are depicted in Fig. 20, which shows the resulting 17 grid blocks (69,632 cells) clustered near the leading edge of the plate and plate surface. As for the supersonic inviscid channel flow with a bump, no refinement is performed by the anisotropic AMR in the cross-flow direction (z-direction) due to the two-dimensional nature of the flow. The equivalent isotropic AMR approach (not shown) results in a similar grid; however, with additional grid blocks in the z-direction and a total of 44 blocks (180,224 cells). Even with just two levels of refinement, the anisotropic AMR procedure yields a more than 60% reduction in mesh size compared to the isotropic approach.

The predicted distribution of the skin friction coefficient, C_f, along the flat plate and the boundary-layer profiles of the velocity components, u and v, tangential and normal to the plate, respectively, at $x = 0.004$ m (Re_x = 8,900) are all compared to the corresponding results following from the Blasius solution in Fig. 21. In the case of the boundary-layer velocity profiles, the non-dimensional boundary-layer velocity components, u/U_∞ and $\theta = v\sqrt{Re_x/U_\infty}$, where U_∞ is the free-stream velocity are shown as a function of the non-dimensional distance from the plate, $\eta = y\sqrt{Re_x/x}$. For this relatively low-speed flow, the results of Fig. 21 show excellent agreement between the compressible flow predictions of the fine mesh anisotropic AMR solutions and the incompressible-flow Blasius solution for the skin friction coefficient over the entire length of the plate and the boundary-layer profiles of both velocity components at the end of the plate.

5.3.2. Steady subsonic laminar flow over an airfoil

Steady subsonic laminar flow over an airfoil is now examined. In particular, a symmetric NACA0012 airfoil at an angle of attack of $\alpha = 0°$ is considered for which the free-stream Mach and Reynolds number are $M_\infty = 0.5$ and Re=5,000, respectively. The chord length is taken to be unity and the far-field boundary is located at about 50 chord lengths away from the airfoil surface. No-slip boundary conditions were imposed on the airfoil boundary. The main features of this case is the creation of a viscous boundary layer on the airfoil surface and the occurrence of flow separation near the trailing edge of the airfoil, resulting in the formation of small re-circulation bubbles on the upper and lower surfaces that extend into the near-wake region and present numerical challenges for the accurate numerical prediction of the airfoil drag. This airfoil flow was examined in several previous studies [105–108,67,95] and the drag coefficient, C_d, is about 0.055 based on the previous simulations of Ivan and Groth [67]. Note that, at zero angle of attack, the lift is expected to be zero for this symmetric airfoil.

The initial C-type coarse mesh for the subsonic airfoil flow consisted of two grid blocks along the upper and lower surfaces of the airfoil (1 block for each side) of two blocks downstream the airfoil for a total of four grid blocks. Each

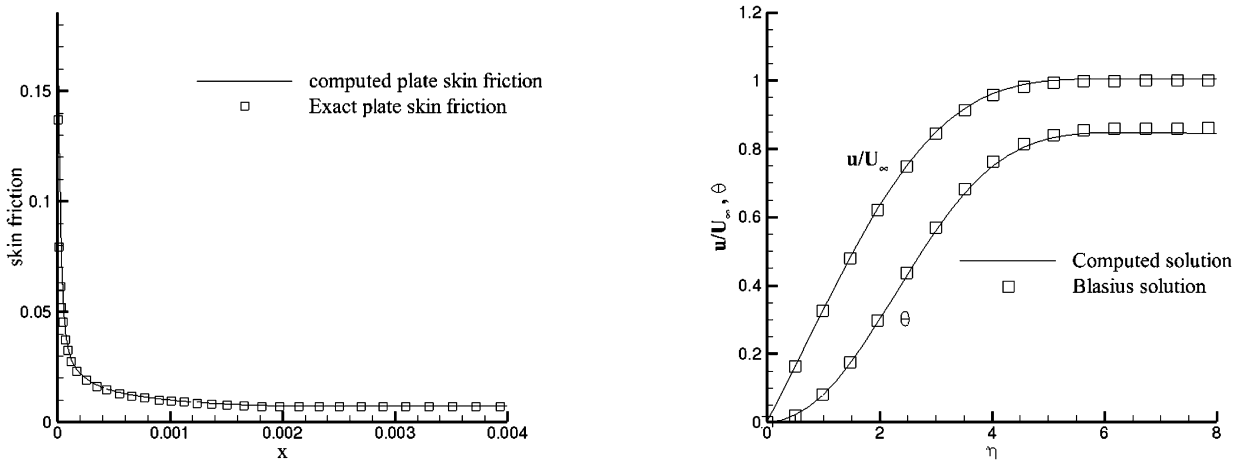


Fig. 21. Numerical prediction of M=0.2 and Re=9,318.92 steady subsonic laminar flow over a 0.0004 m long flat plate showing comparisons of the computed plate skin friction, C_f, along the length of the plate (left) and the boundary-layer profiles of the non-dimensional velocity components, u/U_∞ and $\theta = v\sqrt{\mathrm{Re}_x/U_\infty}$, at x=0.0004 m (right) obtained using the anisotropic AMR scheme to the Blasius solution.

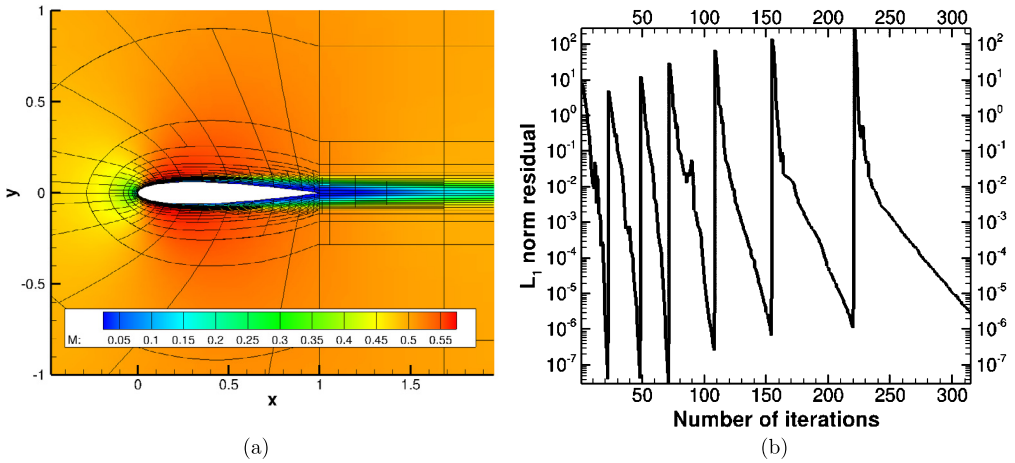


Fig. 22. Solution of $M_\infty = 0.5$ and Re=5,000 steady subsonic laminar flow over a NACA0012 airfoil showing: (a) the predicted distribution of flow Mach number obtained on the final anisotropic AMR mesh having $N_8 = 1,009,664$ computational cells and 986 grid blocks (mesh blocks are outlined in black); and (b) the reduction of the L_2 norm of the solution residual for the total energy as a function of the number of Newton iterations on each of the successively refined AMR meshes.

block contained $16 \times 16 \times 4 = 1,024$ cells and thus the initial mesh contained 4,096 computational cells. The grid lines in the tangential direction to the airfoil surface were clustered such that cell aspect ratio near the airfoil surface is about five. After obtaining the flow solution on this initial mesh, a sequence of eight successively higher-resolution meshes were then generated by applying the proposed anisotropic AMR procedure of Section 4 for which a steady solution was obtained on each consecutive mesh using the inexact Newton solution method. The refinement criteria for the AMR was based on the gradient of the velocity magnitude. The resulting sequence of nine 3D multi-block AMR meshes involved 4, 12, 26, 44, 78, 146, 303, 693, and 986 grid blocks and contained total numbers of computational cells of $N_0 = 4,096$, $N_1 = 12,288$, $N_2 = 26,624$, $N_3 = 45,056$, $N_4 = 79,872$, $N_5 = 149,504$, $N_6 = 310,272$, $N_7 = 709,632$, $N_8 = 1,009,664$, respectively.

The anisotropic block-based AMR results for the $M_\infty = 0.5$ and Re=5,000 NACA0012 airfoil flow are summarized in Figs. 22 and 23(a), where the predicted distribution of flow Mach number obtained on the final anisotropically-refined AMR mesh having $N_8 = 1,009,664$ computational cells and mesh topology are both shown along with the reduction of the L_2 norm of the solution residual for the total energy as a function of the number of Newton iterations illustrating solution convergence on each of the successively refined AMR meshes. Fig. 23(a) shows the convergence of the predicted drag coefficient, C_d, as a function of the mesh size. From the results shown in the figures, it is evident that anisotropic AMR method performs well, providing additional refinement of the mesh in the boundary layer normal to the airfoil surface, with emphasis near the leading and trailing edges of the airfoil and without introducing excessive refinement in the tangential direction (note that no refinement of the 3D mesh is introduced by the method in the out-of-plane or cross stream direction as would be the case of isotropic AMR strategies). In particular, the trailing-edge separation bubbles on the lower and upper

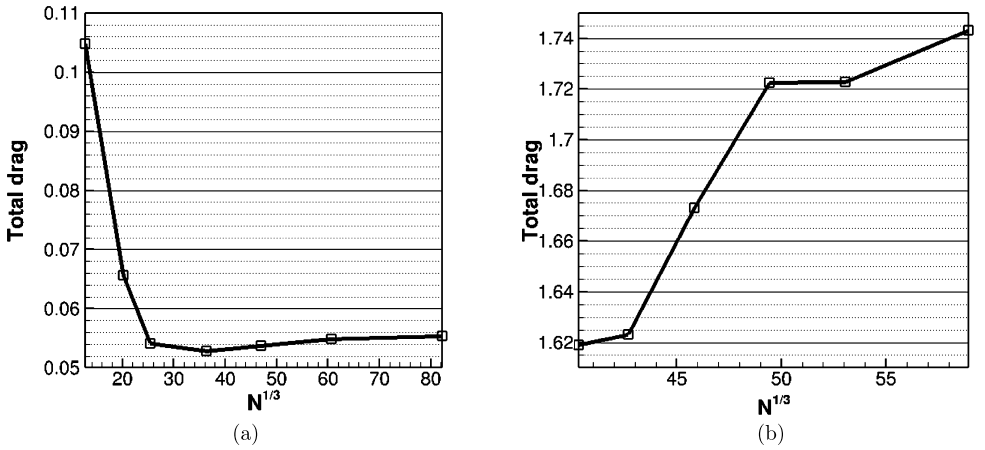


Fig. 23. Predicted drag coefficient, C_d, as a function of total number of computational cells for: (a) $M_\infty = 0.5$ and Re=5,000 steady subsonic laminar flow over a NACA0012 airfoil; and (b) $M_\infty = 0.1$ and Re=30 steady subsonic laminar flow past a circular cylinder.

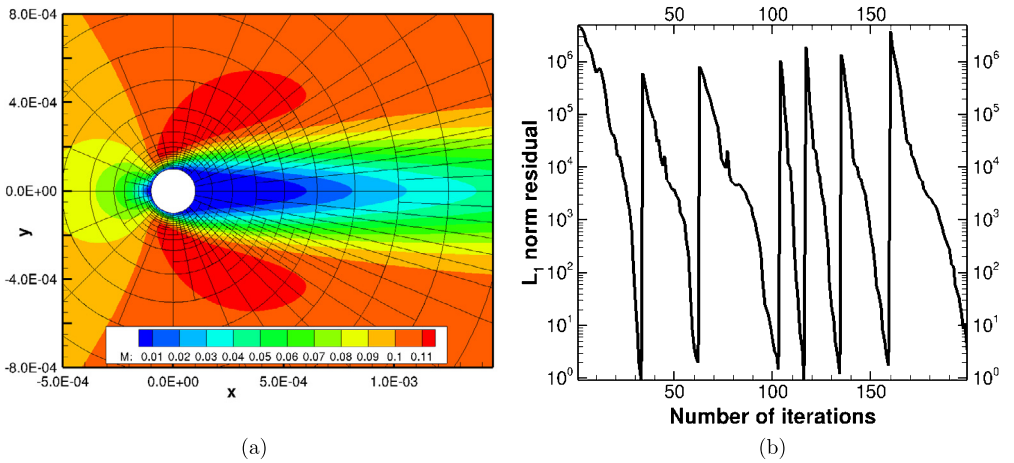


Fig. 24. Solution of $M_\infty = 0.1$ and Re=30 steady subsonic laminar flow past a circular cylinder showing: (a) the predicted distribution of flow Mach number obtained on the final anisotropic AMR mesh having $N_5 = 1,648,640$ computational cells (mesh blocks are outlined in black); and (b) the reduction of the L_2 norm of the solution residual for the total energy as a function of the number of Newton iterations on each of the successively refined AMR meshes.

surfaces are well resolved. As a result, the predicted drag converges rapidly to a value that approximates well previously reported values [67]. It is also evident from Fig. 22(b) that the proposed Newton method is very effective in ensuring rapid and fully converged steady solutions of the Navier-Stokes equations on the multi-block anisotropic AMR meshes.

5.3.3. Steady subsonic laminar flow past a circular cylinder

The third viscous flow problem examined herein is that of steady subsonic laminar flow past a circular cylinder. In this case, the free-stream Mach and Reynolds number are $M_\infty = 0.1$ and Re=30 and numerical solutions were obtained between two concentric cylinders for which the inner and outer cylinder diameters are $D_i = 0.0001$ m and $D_o = 0.004$ m, respectively. No-slip boundary conditions for the flow velocity and adiabatic boundary conditions for temperature were imposed on the inner cylinder wall. This laminar viscous flow was also studied previously by Ivan and Groth [67] and the coefficient of drag as given by the curve fits of Henderson [109] is $C_d = 1.737$. An O-type initial grid was applied for the cylinder flow with clustering of the grid lines toward the surface of the cylinder. The initial grid contained 256 blocks of $16 \times 16 \times 4$ cells that is 262,144 computational cells and, similar to the subsonic airfoil case, a sequence of five successively higher-resolution meshes were obtained by applying the proposed anisotropic AMR procedure. The resulting sequence of six multi-block grids contained $N_0 = 262,144$, $N_1 = 344,064$, $N_2 = 452,608$, $N_3 = 559,104$, $N_4 = 802,816$, $N_5 = 1,648,640$ computational cells, respectively.

The anisotropic AMR cylinder flow results for the six successively refined meshes are presented in Figs. 23(b) and 24. Fig. 23(b) shows the convergence of the predicted drag coefficient, C_d, as a function of the mesh size. Fig. 24 provides the predicted Mach number distribution and mesh topology for the finest mesh with $N_5 = 1,648,640$ computational cells along with the reduction of the solution residual, illustrating the rather good convergence of the Newton method, on each mesh.

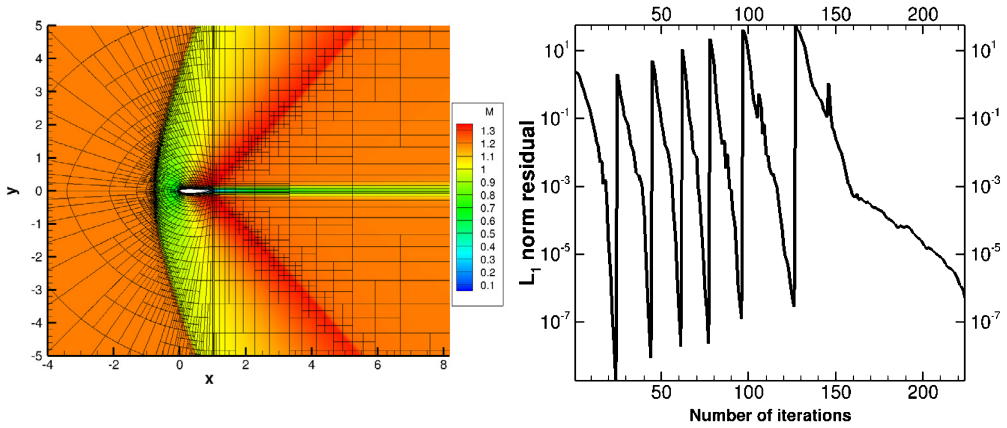


Fig. 25. Solution of M_∞=1.2 and Re=1,000 steady supersonic laminar flow over a NACA0012 airfoil showing: (a) the predicted distribution of flow Mach number obtained on the final anisotropic AMR mesh having $N_8 = 6,307,840$ computational cells and 6,160 grid blocks (mesh blocks are outlined in black); and (b) the reduction of the L_2 norm of the solution residual for the total energy as a function of the number of Newton iterations on each of the successively refined AMR meshes.

The proposed anisotropic AMR scheme efficiently provides refinement of the computational mesh in the normal direction to regions of high shear and flow separation while not introducing un-needed resolution in the transverse directions and, as a consequence, the predicted drag coefficient, C_d, appears to rapidly converge to a value close to that expected for this case.

5.3.4. Steady supersonic laminar flow over an airfoil

As a last viscous case, steady supersonic laminar flow over a NACA0012 symmetric airfoil at zero angle of attack ($\alpha = 0°$) is studied. The free-stream conditions, computational domain, mesh, and boundary conditions are similar to the subsonic case described in Section 5.3.2 except that the freestream Mach number and Reynolds number are M_∞=1.2 and Re=1,000, respectively. This airfoil flow was examined previously by other researchers, including Hartmann and Houston [110] and Narechania et al. [95]. As the flow is supersonic, a bow shock is formed upstream of the leading edge of the airfoil and two weaker shocks emanate downstream from the trailing edge. As for the subsonic airfoil flow, the initial multi-block mesh again contained four grid blocks and consisted of 4,096 computational cells. The proposed anisotropic block-based AMR scheme was then used in combination with the finite-volume scheme and inexact Newton solution method to obtain a sequence of nine successively refined containing $N_0 = 4,096$, $N_1 = 16,384$, $N_2 = 63,488$, $N_3 = 182,272$, $N_4 = 415,744$, $N_5 = 604,160$, $N_6 = 1,354,752$, $N_7 = 2,856,960$, and $N_8 = 6,307,840$ computational cells corresponding to 4, 16, 62, 178, 406, 590, 1,323, 2,790, and 6,160 grid blocks, respectively.

Fig. 25(a) shows the predicted Mach number distribution and mesh block topology for the supersonic NACA0012 airfoil flow obtained on the final anisotropically-refined AMR mesh with $N_8 = 6,307,840$ computational cells. Additionally, Fig. 25(b) depicts the convergence of the Newton method on the sequence of AMR meshes. The results for this case again reveal the effectiveness and efficiency of the proposed anisotropic AMR scheme. The anisotropic method captures very well both the leading-edge bow shock and trailing-edge shocks, as well as the boundary layers near the airfoil surface without introducing un-necessary mesh resolution in directions aligned with the shock waves and shear layers (as would be the case for an isotropic approach). Additionally, the proposed Newton method again proves to be effective in ensuring fully converged steady solutions of the spatially-discretized governing equations are achieved on the anisotropic AMR grids, even for cases with shocks.

6. Conclusions

A new anisotropic block-based AMR scheme based on a novel non-uniform treatment for the grid blocks has been proposed and described for the prediction of 3D compressible gaseous flows. The approach avoids the usual prolongation and restriction of solution content and conservative flux corrections commonly adopted in other block-based AMR approaches, as are the potential complexities for high-order solution methods [32]. A description of the enhanced heterogeneous non-conformal treatment for the grid blocks and the related modifications and extensions to the corresponding second-order finite-volume spatial discretization scheme have been described in detail. A detailed description of the proposed anisotropic block-based AMR scheme has also been provided. Furthermore, the performance and accuracy of the proposed anisotropic AMR method has been assessed for a range of 3D inviscid and viscous flow problems described by the Euler and Navier-Stokes equations of a compressible gas via comparisons to the predictions of the equivalent isotropic AMR approach as well as to available analytical results. The proposed 3D anisotropic block-based AMR scheme has been shown to provide significantly lower computational complexity while also affording significant reductions in mesh size compared to standard

isotropic AMR techniques with comparable accuracy. For flows with highly disparate and anisotropic spatial scales (e.g., shocks/discontinuities and shear layers), mesh reductions of 80–95% were realized for cases in which the anisotropy was aligned with the coordinate axis and reductions of 50–75% were still achieved for more isotropic situations. Future research will involve the further development and application of the proposed 3D anisotropic AMR scheme for use with high-order spatial discretization schemes [70,71] in efforts to simulate more efficiently a broader range of multi-scale, physically-complex, flow phenomena. Additionally, output-based or adjoint-based error estimation techniques [90,91,102,92,93,103,94] will be considered for directing the anisotropic mesh adaptation.

CRediT authorship contribution statement

Lucie Freret: methodology, software, validation, investigation, visualization, writing.
Mickael Williamschen: methodology, software, validation, writing.
Clinton P.T. Groth: conceptualization, methodology, resources, writing, project administration, funding acquisition, supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank the two reviewers for their helpful comments and suggestions for improving the manuscript. This research was supported by the Canadian Space Agency (CSA) through a grant No. 14SUGOAMSM under the GO Canada, Science and Applications Program. Additionally, the computational resources for performing the numerical simulations reported herein were provided by the SOSCIP program as well as the SciNet High Performance Computing Consortium at the University of Toronto and Compute/Calcul Canada (the latter are funded by the Canada Foundation for Innovation (CFI) grant No. 1761 and Province of Ontario, Canada).

References

[1] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[2] M.J. Berger, Data structures for adaptive grid generation, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 904–916.
[3] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[4] M.J. Aftosmis, M.J. Berger, J.E. Melton, Robust and efficient Cartesian mesh generation for component-based geometry, AIAA J. 36 (6) (1998) 952–960.
[5] C.P.T. Groth, D.L. De Zeeuw, K.G. Powell, T.I. Gombosi, Q.F. Stout, A Parallel Solution-Adaptive Scheme for Ideal Magnetohydrodynamics, Paper 99-3273, AIAA, June 1999.
[6] C.P.T. Groth, D.L. De Zeeuw, T.I. Gombosi, K.G. Powell, Global three-dimensional MHD simulation of a space weather event: CME formation, interplanetary propagation, and interaction with the magnetosphere, J. Geophys. Res. 105 (A11) (2000) 25,053–25,078.
[7] M.J. Aftosmis, M.J. Berger, G. Adomavicius, A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries, Paper 2000-0808, AIAA, January 2000.
[8] D.L. De Zeeuw, T.I. Gombosi, C.P.T. Groth, K.G. Powell, Q.F. Stout, An adaptive MHD method for global space weather simulations, IEEE Trans. Plasma Sci. 28 (6) (2000) 1956–1965.
[9] X. Gao, C.P.T. Groth, A parallel solution-adaptive method for three-dimensional turbulent non-premixed combusting flows, J. Comput. Phys. 229 (5) (2010) 3250–3275.
[10] X. Gao, S.A. Northrup, C.P.T. Groth, Parallel solution-adaptive method for two-dimensional non-premixed combusting flows, Prog. Comput. Fluid Dyn. 11 (2) (2011) 76–95.
[11] M.J. Berger, R.J. LeVeque, An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries, Paper 89-1930, AIAA, June 1989.
[12] D. De Zeeuw, K.G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, J. Comput. Phys. 104 (1993) 56–68.
[13] D.L. De Zeeuw, A quadtree-based adaptively-refined Cartesian-grid algorithm for solution of the Euler equations, Ph.D. thesis, University of Michigan, September 1993.
[14] K.G. Powell, P.L. Roe, J. Quirk, Adaptive-mesh algorithms for computational fluid dynamics, in: M.Y. Hussaini, A. Kumar, M.D. Salas (Eds.), Algorithmic Trends in Computational Fluid Dynamics, Springer-Verlag, New York, 1993, pp. 303–337.
[15] W.J. Coirier, K.G. Powell, An accuracy assessment of Cartesian-mesh approaches for the Euler equations, J. Comput. Phys. 117 (1995) 121–131.
[16] W.J. Coirier, K.G. Powell, Solution-adaptive Cartesian cell approach for viscous and inviscid flows, AIAA J. 34 (5) (1996) 938–945.
[17] S.M. Murman, M.J. Aftosmis, M.J. Berger, Numerical Simulation of Rolling-Airframes Using a Multi-Level Cartesian Method, Paper 02–2798, AIAA, June 2002.
[18] W.J. Coirier, An adaptively-refined, Cartesian, cell-based scheme for the Euler and Navier-Stokes equations, Ph.D. thesis, University of Michigan, 1994.
[19] M.J. Aftosmis, M.J. Berger, S.M. Murman, Applications of Space-Filling Curves to Cartesian Methods for CFD, Paper 2004-1232, AIAA, January 2004.
[20] F.E. Ham, F.S. Lien, A.B. Strong, A Cartesian grid method with transient anisotropic adaptation, J. Comput. Phys. 179 (2002) 469–494.
[21] G. Iaccarino, F. Ham, Automatic mesh generation for LES in complex geometries, in: Annual Research Briefs, Center for Turbulence Research, 2005, pp. 460–490.
[22] G. Iaccarino, F. Ham, LES on cartesian grids with anisotropic refinement, in: S.C. Kassinos, C.A. Langer, G. Iaccarino, P. Moin (Eds.), Complex Effects in Large Eddy Simulations, in: Lecture Notes in Computational Science and Engineering, vol. 56, Springer, Berlin, 2007, pp. 219–233.
[23] T. Leicht, R. Hartmann, Error estimation and anisotropic mesh refinement for 3D laminar aerodynamic flow simulations, J. Comput. Phys. 229 (2010) 7344–7360.
[24] M.J. Berger, Adaptive mesh refinement for hyperbolic partial differential equations, Ph.D. thesis, Stanford University, January 1982.

[25] J. Christopher, X. Gao, S.M.J. Guzik, Anisotropic Patch-Based Adaptive Mesh Refinement for Finite-Volume Method, Paper 2016-0605, AIAA, January 2016.
[26] M. Adams, P. Colella, D. Graves, N.D.K.H. Johnson, T.J. Ligocki, D. Matrin, P. Corquodale, D. Modiano, P.O. Schwartz, T.D. Sternberg, B. Van Straalen, Chombo software package for AMR applications — design document, Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, December 2015.
[27] A.M. Wissink, R.D. Horning, S.R. Kohn, S.S. Smith, N. Elliott, Large scale parallel structured AMR calculations using the SAMRAI framework, in: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, ACM, New York, 2001, pp. 1–6.
[28] B.W. O'shea, G. Bryan, J. Bordner, M.L. Norman, T. Abel, R. Harkness, A. Kritsuk, Introducing Enzo, an AMR cosmology application, in: T. Plewa, T.L. T, V. Gregory Weirs (Eds.), Adaptive Mesh Refinement — Theory and Applications, in: Lecture Notes in Computational Science and Engineering, vol. 41, Springer, Berlin, 2005, pp. 341–349.
[29] A. Mignonge, G. Bodo, S. Massaglia, T. Matsakos, O. Tesoleanu, C. Zanni, A. Ferrari, PLUTO: a numerical code for computational astrophysics, Astrophys. J. Suppl. Ser. 170 (1) (2007) 228–242.
[30] U. Ziegler, The NIRVANA code: parallel computational MHD with adaptive mesh refinement, Comput. Phys. Commun. 179 (4) (2008) 227–244.
[31] A.J. Cunningham, A. Frank, P. Varnière, S. Mitran, T.W. Jones, Simulating magnetohydrodynamical flow with constrained-transport and adaptive mesh refinement: algorithms and tests of the AstroBEAR code, Astrophys. J. Suppl. Ser. 182 (2) (2009) 519–542.
[32] A. Dubey, A. Almgren, J. Bell, M. Berzins, S. Brandt, G. Bryan, P. Colella, D. Graves, M. Lijewski, F. Löffler, B. O'Shea, E. Schnetter, B. Van Straalen, K. Weide, A survey of high level frameworks in block-structured adaptive mesh refinement packages, J. Parallel Distrib. Comput. 74 (12) (2014) 3217–3227.
[33] S.M. Guzik, X. Gao, L.D. Owen, P. McCorquodale, P. Colella, A freestream-preserving fourth-order finite-volume method in mapped coordinates with adaptive-mesh refinement, Comput. Fluids 123 (2015) 202–217.
[34] X. Gao, L.D. Owen, S.M.J. Guzik, A parallel adaptive numerical method with generalized curvilinear coordinate transformation for compressible Navier–Stokes equations, Int. J. Numer. Methods Fluids 82 (10) (2016) 664–688.
[35] L.D. Owen, S.M. Guzik, X. Gao, A high-order adaptive algorithm for multispecies gaseous flows on mapped domains, Comput. Fluids 170 (2018) 249–260.
[36] J.J. Quirk, U.R. Hanebutte, A parallel adaptive mesh refinement algorithm, Report 93-63, ICASE, August 1993.
[37] M.J. Berger, J.S. Saltzman, AMR on the CM-2, Appl. Numer. Math. 14 (1994) 239–253.
[38] P. MacNeice, K.M. Olson, C. Mobarry, R. de Fainchtein, C. Packer, PARAMESH: a parallel adaptive mesh refinement community toolkit, Comput. Phys. Commun. 126 (3) (2000) 330–354.
[39] C. Burstedde, L.C. Wilcox, O. Ghattas, p4est: scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133.
[40] R. Keppens, Z. Meliani, A.J. van Marle, P. Delmont, A. Vlasis, B. van der Holst, Parallel, grid-adaptive approaches for relativistic hydro and magnetohydrodynamics, J. Comput. Phys. 231 (2012) 718–744.
[41] O. Porth, C. Xia, T. Hendrix, S.P. Moschou, R. Keppens, MPI-AMRVAC for solar and astrophysics, Astrophys. J. Suppl. Ser. 214 (1) (2014) 1–26.
[42] O. Porth, H. Olivares, Y. Mizuno, Z. Younsi, L. Rezzolla, M. Moscibrodzka, H. Falcke, M. Kramer, The black hole accretion code, Comput. Astrophys. Cosmol. 4 (1) (2017) 1–42.
[43] A. Dubey, K. Antypas, A. Calder, B. Fryxell, D. Lamb, P. Ricker, L. Reid, K. Riley, R. Rosner, A. Siegel, F. Timmes, N. Vladimirova, K. Weide, The software development process of FLASH, a multiphysics simulation code, in: Proceedings of the 5th International Workshop on Software Engineering for Computational Science and Engineering, IEEE Press, Piscataway, NJ, 2013, pp. 1–8.
[44] G. Tóth, B. van der Holst, I.V. Sokolov, D.L. De Zeeuw, T.I. Gombosi, F. Fang, W.B. Manchester, X. Meng, D. Najib, K.G. Powell, Q.F. Stout, A. Glocer, Y.-J. Ma, M. Opher, Adaptive numerical algorithms in space weather modeling, J. Comput. Phys. 231 (2012) 870–903.
[45] D.A. Calhoun, C. Burstedde, ForestCLAW: A Parallel Algorithm for Patch-Based Adaptive Mesh Refinement on a Forest of Quadtrees, arXiv Online Article arXiv:1703.03116v1, Cornell University Library, 2017.
[46] J.S. Sachdev, C.P.T. Groth, J.J. Gottlieb, A parallel solution-adaptive scheme for predicting multi-phase core flows in solid propellant rocket motors, Int. J. Comput. Fluid Dyn. 19 (2) (2005) 159–177.
[47] X. Gao, C.P.T. Groth, A parallel adaptive mesh refinement algorithm for predicting turbulent non-premixed combusting flows, Int. J. Comput. Fluid Dyn. 20 (5) (2006) 349–357.
[48] L. Ivan, H. De Sterck, S.A. Northrup, C.P.T. Groth, Hyperbolic conservation laws on three-dimensional cubed-sphere grids: a parallel solution-adaptive simulation framework, J. Comput. Phys. 255 (2013) 205–227.
[49] J.S. Sachdev, C.P.T. Groth, A mesh adjustment scheme for embedded boundaries, Commun. Comput. Phys. 2 (6) (2007) 1095–1124.
[50] L. Ivan, H. De Sterck, A. Susanto, C.P.T. Groth, High-order central eno finite-volume scheme for hyperbolic conservation laws on three-dimensional cubed-sphere grids, J. Comput. Phys. 282 (2015) 157–182.
[51] J.S. Sachdev, C.P.T. Groth, J.J. Gottlieb, Parallel Solution-Adaptive Scheme for Multi-Phase Core Flows in Rocket Motors, Paper 2003–4106, AIAA, June 2003.
[52] S.A. Northrup, C.P.T. Groth, Solution of Laminar Diffusion Flames Using a Parallel Adaptive Mesh Refinement Algorithm, Paper 2005–0547, AIAA, January 2005.
[53] M.R.J. Charest, C.P.T. Groth, Ö.L. Gülder, A computational framework for predicting laminar reactive flows with soot formation, Combust. Theory Model. 14 (6) (2010) 793–825.
[54] S.A. Northrup, C.P.T. Groth, Parallel Implicit Adaptive Mesh Refinement Scheme for Unsteady Fully-Compressible Reactive Flows, Paper 2013-2433, AIAA, June 2013.
[55] J.G. McDonald, C.P.T. Groth, Numerical Modeling of Micron-Scale Flows Using the Gaussian Moment Closure, Paper 2005-5035, AIAA, June 2005.
[56] J.G. McDonald, J.S. Sachdev, C.P.T. Groth, Application of Gaussian moment closure to micro-scale flows with moving and embedded boundaries, AIAA J. 51 (9) (2014) 1839–1857.
[57] M.R.J. Charest, C.P.T. Groth, Ö.L. Gülder, Solution of the equation of radiative transfer using a Newton-Krylov approach and adaptive mesh refinement, J. Comput. Phys. 231 (2012) 3023–3040.
[58] E.P. Boden, E.F. Toro, A combined Chimera-AMR technique for computing hyperbolic PDEs, in: 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.
[59] G. Chesshire, W.D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, J. Comput. Phys. 90 (1990) 1–64.
[60] W.D. Henshaw, A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids, J. Comput. Phys. 113 (1994) 13–25.
[61] W.D. Henshaw, D.W. Schwendeman, An adaptive numerical scheme for high-speed reacting flow on overlapping grids, J. Comput. Phys. 191 (2003) 420–447.
[62] B. van der Holst, R. Keppens, Hybrid block-AMR in Cartesian and curvilinear coordinates: MHD applications, J. Comput. Phys. 226 (2007) 925–946.
[63] Z.J. Zhang, Parallel anisotropic block-based adaptive mesh refinement finite-volume scheme, Master's thesis, University of Toronto, 2011.
[64] Z.J. Zhang, C.P.T. Groth, Parallel High-Order Anisotropic Block-Based Adaptive Mesh Refinement Finite-Volume Scheme, Paper 2011-3695, AIAA, June 2011.

[65] M.J. Williamschen, C.P.T. Groth, Parallel Anisotropic Block-Based Adaptive Mesh Refinement Algorithm for Three-Dimensional Flows, Paper 2013-2442, AIAA, June 2013.
[66] L. Freret, C.P.T. Groth, Anisotropic Non-uniform Block-Based Adaptive Mesh Refinement for Three-Dimensional Inviscid and Viscous Flows, Paper 2015-2613, AIAA, June 2015.
[67] L. Ivan, C.P.T. Groth, High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows, J. Comput. Phys. 257 (2014) 830–862.
[68] A. Susanto, L. Ivan, H.D. Sterck, C.P.T. Groth, High-order central ENO finite-volume scheme for ideal MHD, J. Comput. Phys. 250 (2013) 141–164.
[69] M.R.J. Charest, C.P.T. Groth, P.Q. Gauthier, A high-order central ENO finite-volume scheme for three-dimensional low-speed viscous flows on unstructured mesh, Commun. Comput. Phys. 17 (3) (2015) 615–656.
[70] L. Freret, L. Ivan, H. De Sterck, C.P.T. Groth, A High-Order Finite-Volume Method with Anisotropic AMR for Ideal MHD Flows, Paper 2017-0845, AIAA, January 2017.
[71] L. Freret, L. Ivan, H. De Sterck, C.P.T. Groth, High-order finite-volume method with block-based AMR for magnetohydrodynamics flows, J. Sci. Comput. 79 (1) (2019) 176–208.
[72] H. Schlichting, K. Gersten, Boundary-Layer Theory, 8th edition, Springer, New York, 2000.
[73] C. Hirsch, Numerical Computation of Internal and External Flows, Volume 1, Fundamentals of Numerical Discretization, John Wiley & Sons, Toronto, 1989.
[74] T.J. Barth, Recent Developments in High Order k-Exact Reconstruction on Unstructured Meshes, Paper 93-0668, AIAA, January 1993.
[75] T.J. Barth, D.C. Jespersen, The Design and Application of Upwind Schemes on Unstructured Meshes, Paper 89-0366, AIAA, January 1989.
[76] V. Venkatakrishnan, On the Accuracy of Limiters and Convergence to Steady State Solutions, Paper 93-0880, AIAA, January 1993.
[77] A. Harten, P.D. Lax, B. van Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, SIAM Rev. 25 (1) (1983) 35–61.
[78] B. Einfeldt, On Godunov-type methods for gas dynamics, SIAM J. Numer. Anal. 25 (1988) 294–318.
[79] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[80] S.A. Northrup, A parallel implicit adaptive mesh refinement algorithm for predicting unsteady fully-compressible reactive flows, Ph.D. thesis, University of Toronto, December 2013.
[81] H. Lomax, T.H. Pulliam, D.W. Zingg, Fundamentals of Computational Fluid Dynamics, Springer-Verlag, New York, 2001.
[82] B. van Leer, C.H. Tai, K.G. Powell, Design of Optimally-Smoothing Multi-Stage Schemes for the Euler Equations, Paper 89-1933-CP, AIAA, June 1989.
[83] B. van Leer, W.-T. Lee, P.L. Roe, K.G. Powell, C.-H. Tai, Design of optimally smoothing multistage schemes for the Euler equations, Commun. Appl. Numer. Methods 8 (10) (1992) 761–769.
[84] C.P.T. Groth, S.A. Northrup, Parallel Implicit Adaptive Mesh Refinement Scheme for Body-Fitted Multi-Block Mesh, Paper 2005-5333, AIAA, June 2005.
[85] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear equations, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[86] Y. Saad, Krylov subspace methods on supercomputers, SIAM J. Sci. Stat. Comput. 10 (6) (1989) 1200–1232.
[87] P.N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Stat. Comput. 11 (3) (1990) 450–481.
[88] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, Boston, 1996.
[89] W.A. Mulder, B. van Leer, Experiments with implicit upwind methods for the Euler equations, J. Comput. Phys. 59 (1985) 232–246.
[90] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, Acta Numer. 10 (2001) 1–102.
[91] R. Becker, V. Heuveline, R. Rannacher, An optimal control approach to adaptivity in computational fluid dynamics, Int. J. Numer. Methods Fluids 40 (2002) 105–120.
[92] D.A. Venditti, D.L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow, J. Comput. Phys. 164 (2000) 204–227.
[93] D.A. Venditti, D.L. Darmofal, Grid adaptation for functional outputs: application to two-dimensional inviscid flows, J. Comput. Phys. 176 (2002) 40–69.
[94] D.A. Venditti, D.L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, J. Comput. Phys. 187 (2003) 22–46.
[95] N. Narechania, L. Freret, C.P.T. Groth, Block-Based Anisotropic AMR with a Posteriori Adjoint-Based Error Estimation for Three-Dimensional Inviscid and Viscous Flows, Paper 2017-4113, AIAA, June 2017.
[96] M.A. Kopera, F.X. Giraldo, Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solution of the compressible Euler equations with application to atmospheric simulations, J. Comput. Phys. 275 (2014) 92–117.
[97] X. Gao, A parallel solution-adaptive method for turbulent non-premixed combusting flows, Ph.D. thesis, University of Toronto, August 2008.
[98] Z. Lan, V.E. Taylor, G. Bryan, A novel dynamic load balancing scheme for parallel systems, J. Parallel Distrib. Comput. 62 (12) (2002) 1763–1781.
[99] H. Sagan, Space-Filling Curves, Springer-Verlag, New York, 1994.
[100] O. Pearce, T. Gamblin, B.R. de Supinski, M. Schulz, N.M. Amato, Quantifying the effectiveness of load balance algorithms, in: Proceedings of the 26th ACM International Conference on Supercomputing, ACM, New York, 2012, pp. 185–194.
[101] C. Ronchi, R. Iacono, P.S. Paolucci, The “Cubed Sphere”: a new method for the solution of partial differential equations in spherical geometry, J. Comput. Phys. 124 (1996) 93–114.
[102] R. Becker, M. Braack, B. Vexler, Numerical parameter estimation for chemical models in multidimensional reactive flows, Combust. Theory Model. 8 (4) (2004) 661–682.
[103] D.A. Venditti, D.L. Darmofal, Anisotropic Adaptation for Functional Outputs of Viscous Flow Simulations, Paper 2003-3845, AIAA, June 2003.
[104] H. Schlichting, Boundary-Layer Theory, 7th edition, McGraw-Hill, Toronto, 1979.
[105] D.J. Mavriplis, A. Jameson, J. Martinelli, Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes, Paper 89-0120, AIAA, January 1989.
[106] Y. Sun, Z.J. Wang, Y. Liu, Efficient implicit non-linear LU-SGS approach for compressible flow computation using high-order spectral difference method, Commun. Comput. Phys. 5 (2–4) (2009) 760–778.
[107] R. Kannan, Z.J. Wang, A study of viscous flux formulations for a p-multigrid spectral volume Navier Stokes solver, J. Sci. Comput. 41 (2) (2009) 165–199.
[108] R. Kannan, Z.J. Wang, A variant of the LDG flux formulation for the spectral volume method, J. Sci. Comput. 46 (2) (2011) 314–328.
[109] R.D. Henderson, Details of the drag curve near the onset of vortex shedding, Phys. Fluids 7 (9) (1995) 2102–2104.
[110] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2002) 508–532.