COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **2**, No. 6, pp. 1095-1124

Commun. Comput. Phys.
December 2007

A Mesh Adjustment Scheme for Embedded Boundaries

J. S. Sachdev and C. P. T. Groth*

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

Received 20 December 2006; Accepted (in revised version) 20 March 2007

Communicated by Kun Xu

Available online 15 June 2007

Abstract. An adaptive meshing technique and solution method is proposed in which a two-dimensional body-fitted multi-block mesh is locally adjusted to arbitrarily embedded boundaries that are not necessarily aligned with the mesh. Not only does this scheme allow for rapid and robust mesh generation involving complex embedded boundaries, it also enables the solution of unsteady flow problems involving bodies and interfaces moving relative to the flow domain. This scheme has been implemented within a block-based adaptive mesh refinement (AMR) numerical framework which can ease computational expense while maintaining a detailed representation of the embedded boundary and providing an accurate resolution of the spatial characteristics of the fluid flow. Rigid body motion and evolving motion due to physical processes are considered. A block-based AMR level set method is used to deal with evolving embedded boundaries. Numerical results for various test problems are presented to verify the validity of the scheme as well as demonstrate the capabilities of the approach for predicting complex two-dimensional inviscid and laminar fluid flows.

AMS subject classifications: 65M50, 68W10, 74S10, 76M12

PACS (2006): 47.11.-j, 47.11.Df, 47.55.N-, 47.40.-x

Key words: Dynamic and stationary embedded boundaries, mesh adjustment algorithm, block-based adaptive mesh refinement.

1 Introduction and motivation

Fluid flows involving moving boundaries are relevant to many aerospace engineering applications, such as: aerodynamic control surfaces, helicopter rotor blades, compressor and turbine blades, stage separation in launch vehicles, and the combustion interface in solid propellant rocket motors. Numerical solution of these unsteady flows must account for the motion the boundaries through the domain of interest. This can be accomplished

*Corresponding author. *Email addresses:* j.sachdev@utoronto.ca (J. S. Sachdev), groth@utias.utoronto.ca (C. P. T. Groth)

by regenerating or adjusting the computational mesh according to the object's location [6,25], by using overlapping meshes [7], or by representing the boundary as a force-term which influences the flow solution in a manner consistent with fluid dynamics [40,41]. These approaches are now briefly summarized.

The Arbitrary Lagrangian-Eulerian (ALE) method pioneered by Hirt et al. [25] is a popular method for dealing with moving boundaries [3, 19, 20, 54]. This scheme involves the use of a grid, typically unstructured, that moves according to the motion of the boundaries. Conservation is strictly satisfied in ALE methods, however, the quality of the mesh can be severely degraded when large scale motions are involved requiring re-meshing and/or untangling procedures [50].

An alternative approach is the use of overlapping meshes, also known as overset mesh or the Chimera method, where a separate body-fitted mesh is constructed around each object in the computational domain and is overlapped with each other and a simpler, in many cases Cartesian, mesh encompassing the computational domain of interest. This method was originated by Benek et al. [7] and a review of recent work on this method was given by Noack and Slotnick [35]. Interpolation is required between the meshes during the solution of the governing equations. These interpolation schemes are typically non-conservative and non-monotone [36], which may be problematic for many applications.

The immersed boundary method was devised by Peskin [40,41] in order to predict fluid-structure interactions in biological fluid dynamics. In this scheme the boundary is represented as a force term in the governing equations which influences the flow solution in a manner consistent with fluid dynamics. The immersed boundary method has been successfully applied to complex laminar and turbulent flows [14,29,49].

Although the methods described above have been shown to be effective for treating embedded boundaries, in this paper, a new approach is proposed in which a body-fitted multi-block mesh is locally adjusted to arbitrarily embedded boundaries that are not necessarily aligned with the mesh. Not only does this scheme allow for rapid and robust mesh generation involving complex embedded boundaries, it also enables the solution of unsteady flow problems involving bodies and interfaces moving relative to the flow domain. The mesh adjustment algorithm described here has similarities with the Cartesian cut-cell techniques developed by De Zeeuw and Powell [15] and Aftosmis et al. [2] except that the underlying mesh is no longer restricted to a Cartesian mesh. The readjustment of the mesh for moving embedded boundaries follows the approach used in the Cartesian cut cell methods developed by Bayyuk et al. [6] and Murman et al. [34]. Unlike the Cartesian cut-cell method, the mesh adjustment algorithm presented here does not result in the generation of tiny cut-cells which can be restrictive on the time-step and do not permit the construction of consistent and accurate operators for viscous (elliptic) fluxes [12]. This scheme has been implemented within a block-based adaptive mesh refinement (AMR) numerical framework [43]. It has been well established that AMR techniques, which automatically adapt the computational grid to the solution of the governing partial differential equations, are very effective in treating fluid problems with disparate length scales while minimizing computational expense [8,9,15,23,43].

The scheme outlined above is described in the following six sections: (i) finite volume formulation for inviscid and laminar compressible fluid flows; (ii) parallel block-based AMR scheme; (iii) the mesh adjustment scheme; (iv) spatial discretization methods; (v) the mesh adjustment scheme for moving embedded boundaries; and (vi) the parallel block-based AMR level set method used for modeling evolving boundaries. Numerical results for various test problems are also presented to verify the validity of the scheme as well as demonstrate the capabilities of the approach for predicting complex two-dimensional inviscid and laminar fluid flows.

2 Finite volume formulation for compressible fluids flows

Although the proposed mesh adjustment scheme would seem applicable to a wide range of engineering problems, the application of current interest to the authors are related to compressible gaseous flows. For this reason, we shall restrict our attention to two-dimensional inviscid and laminar flows of compressible gases. A higher-order finite-volume scheme is used here to solve the equations governing the motion of compressible gases in two space dimensions. The governing equations are integrated to obtain area-averaged solution quantities within computational cells. For a time-dependent domain, the semi-discrete form of the governing equations for an individual elemental volume are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{U}A)=-\sum_k\left[\vec{\mathbf{F}}_k-\vec{w}_k\mathbf{U}_k-\vec{\mathbf{G}}_k\right]\cdot\hat{n}_k\Delta\ell_k, \tag{2.1}$$

where A is the area of the computational cell and $\Delta\ell$ and $\hat{n}$ are the length and unit outward normal of the cell faces. The vector, $\mathbf{U}$, contains the cell-averaged conserved solution quantities given by

$$\mathbf{U}=[\rho,\rho u,\rho v,E]^T, \tag{2.2}$$

where u and v are the components of the gas velocity, $\vec{v}$, and ρ and E are the density and total energy per unit volume of the gas. The gas is taken to be calorically perfect and the total energy per unit volume, E, is

$$E=\frac{p}{(\gamma-1)}+\frac{\rho}{2}(\vec{v}\cdot\vec{v})=\rho c_vT+\frac{\rho}{2}(\vec{v}\cdot\vec{v}),$$

where $\gamma=c_p/c_v$ is the ratio of the specific heats for the gas and p is the pressure. The ideal gas law, $p=\rho\mathrm{R}T=\rho a^2/\gamma$, provides a relationship between the pressure, p, density, ρ, and temperature, T, where $a=\sqrt{\gamma\mathrm{R}T}$ is the sound speed and R is the gas constant.

The flux dyads, $\vec{\mathbf{F}}(\mathbf{U})=(\mathbf{F}_x,\mathbf{F}_y)$ and $\vec{\mathbf{G}}(\mathbf{U},\vec{\nabla}\mathbf{U})=(\mathbf{G}_x,\mathbf{G}_y)$, correspond to the inviscid flux dyad, and viscous flux dyad, respectively,

$$\mathbf{F}_x=\left[\rho u,\rho u^2+p,\rho uv,u(E+p)\right]^T,\quad \mathbf{F}_y=\left[\rho v,\rho uv,\rho v^2+p,v(E+p)\right]^T,$$
$$\mathbf{G}_x=\left[0,\tau_{xx},\tau_{xy},u\tau_{xx}+v\tau_{xy}-q_x\right]^T,\quad \mathbf{G}_y=\left[0,\tau_{xy},\tau_{yy},u\tau_{xy}+v\tau_{yy}-q_y\right]^T,$$

where the components of the laminar stress tensor, $\vec{\vec{\tau}}$, and the heat transfer vector, $\vec{q}$, are given by $\vec{\vec{\tau}} = 2\mu\left(\vec{\nabla}\vec{v} - \frac{1}{3}\vec{\nabla}\cdot\vec{v}\vec{\vec{I}}\right)$ and $\vec{q} = -\kappa\vec{\nabla}T$, respectively.

If the area of the computational cell varies with time then the flux contribution due to the motion of the cell interfaces must be accounted for, as given by the $\vec{w}\mathbf{U}$ term in the above equation where $\vec{w}$ is the velocity of the cell interface. The time-derivative term can be expanded to give the following form for the semi-discrete ordinary differential equations:

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = -\frac{1}{A}\sum_k \left[\vec{\mathbf{F}}_k - \vec{w}_k\mathbf{U}_k - \vec{\mathbf{G}}_k\right]\cdot\hat{n}_k\Delta\ell_k - \frac{\mathbf{U}}{A}\frac{\mathrm{d}A}{\mathrm{d}t}. \tag{2.3}$$

The last term on the right-hand side of this equation corresponds to the rate of change of the cell area which can be approximated by the geometric conservation law, written in semi-discrete form, as

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \sum_k \vec{w}_k\cdot\hat{n}_k\Delta\ell_k. \tag{2.4}$$

The geometric conservation law states that the change in cell area is equal to the area swept by the moving surfaces [47]. The semi-discrete system of ordinary differential equations given by equation (2.3) is used here for simulations involving dynamic boundaries. Note that the use of this formulation dictates that the geometry is only updated at the end of each time step and not during or as part of the time step.

Spatial discretization of the hyperbolic and elliptic fluxes used in the proposed scheme are discussed in Section 5. For time-accurate calculations, predictor-corrector and fourth order Runge-Kutta time-marching methods are used to integrate the set of ordinary differential equations that result from the spatial discretization of the governing equations. The optimally-smoothing multi-stage schemes developed by van Leer et al. [51] are adopted for steady-state calculations.

3 Parallel block-based adaptive mesh refinement

The finite-volume formulation described above is applied to quadrilateral computational cells of a multi-block body-fitted mesh. Examples of the block structure are shown in Figs. 1 and 2 (mesh not shown). Solution data associated with each block are stored in indexed array data structures and it is therefore straightforward to obtain solution information from neighboring cells within blocks. In order that the solution algorithm can be applied to all blocks in a more independent manner, some solution information is shared between adjacent blocks having common interfaces. This information is stored in an additional two layers of overlapping "ghost" cells associated with each block. At interfaces between blocks of equal resolution, these ghost cells are simply assigned the solution values associated with the appropriate interior cells of the adjacent blocks.

In this work, each of the structured blocks consist of $N_x \times N_y$ quadrilateral cells, where N_x and N_y are even but not necessarily equal integers. The self-similar nature of the solution blocks provides a natural domain decomposition for parallel implementation of the algorithm. Domain decomposition is carried out by merely farming the solution blocks out to the separate processors, with more than one block permitted on each processor. A simple stack is used to keep track of available (open) processors. For homogeneous architectures with multiple processors all of equal speed, an effective load balancing is achieved by exploiting the self-similar nature of the solution blocks and simply distributing the blocks equally among the processors. In doing so, all blocks are treated equally and, currently, no use is made of the hierarchical data structure nor grid partitioning techniques to preferentially place neighboring blocks on the same processors. A detailed explanation of the parallel implementation and an assessment of its performance was provided in Ref. [43].

Adaptive mesh refinement (AMR) techniques which automatically adapt the computational grid to the solution of the governing partial differential equations can be very effective in treating problems with disparate length scales. Here, a block-based AMR method is employed as described in detail in Ref. [43]. This method follows the approach developed by Groth et al. for computational magnetohydrodynamics [23]. The use of curvilinear (arbitrary quadrilateral) mesh makes the application of the block-based AMR more amenable to flows with thin boundary layers and permits anisotropic refinement as dictated by the initial mesh stretching.

In the AMR scheme, mesh adaptation is accomplished by the dividing and coarsening of appropriate solution blocks. In regions requiring increased cell resolution, a "parent" block is refined by dividing itself into four "children" or "offspring." Each of the four quadrants or sectors of a parent block becomes a new block having the same number of cells as the parent and thereby doubling the cell resolution in the region of interest. This process can be reversed in regions that are deemed over-resolved and four children are coarsened into a single parent block. Although several approaches are possible, for this study, the coarsening and division of blocks are directed using multiple physics-based refinement criteria [39]. The gradient of the density field and the divergence and curl of the velocity field are used herein. These quantities correspond to local measures of the density gradient, compressibility, and vorticity of the gas-phase and enable the detection of contact surfaces, shocks, and shear layers. The mesh refinement is constrained such that the grid resolution changes by only a factor of two between adjacent blocks and the minimum resolution is not less than that of the initial mesh. Standard multigrid-type restriction and prolongation operators are used to evaluate the solution on all blocks created by the coarsening and division processes, respectively. Similar restriction and prolongation operators are employed to evaluate the ghost cell solution values for neighboring blocks at different refinement levels. Additional inter-block communication is also required at interfaces with resolution changes to strictly enforce the flux conservation properties of the finite-volume scheme [8, 9]. In particular, the interface fluxes computed on more refined blocks are used to correct the interface fluxes computed on

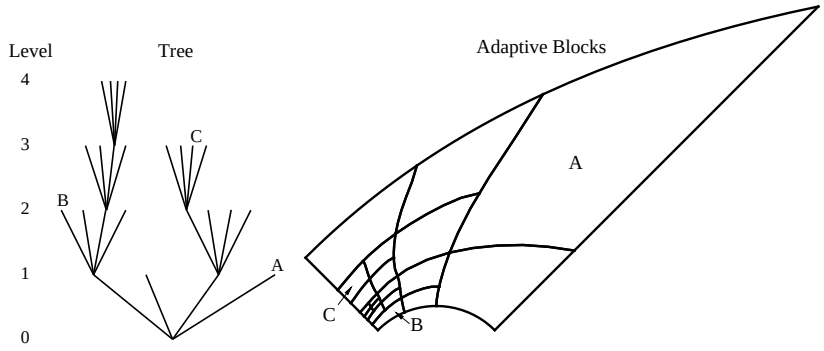


Figure 1: Solution blocks of a computational mesh with four refinement levels originating from one initial block and the associated hierarchical quadtree data structure. Interconnects to neighbors are not shown.

coarser neighboring blocks and ensure that the fluxes are conserved at block interfaces.

A hierarchical tree-like data structure with multiple "roots", multiple "trees", and additional interconnects between the "leaves" of the trees is used to keep track of mesh refinement and the connectivity between solution blocks. This interconnected "forest" data structure is depicted in Fig. 1. The blocks of the initial mesh are the roots of the forest which are stored in an indexed array data structure. Associated with each root is a separate "quadtree" data structure that contains all of the blocks making up the leaves of the tree created from the original parent blocks during mesh refinement. One of the advantages of the hierarchical quadtree data structure is that it readily permits local mesh refinement at any point in a calculation. Local modifications to the multi-block mesh can be performed without re-gridding the entire mesh and re-calculating solution block connectivity.

In order to carry out mesh refinement and inter-block communication, a complete copy of the hierarchical quadtree data structure is stored on each processor. This is possible because, unlike cell-based unstructured meshing techniques, the block-based tree data structure is not overly large. The structure need only retain the connectivity between the solution blocks as opposed to a complete map of the cell connectivity required by general unstructured mesh procedures. Inter-processor communication is mainly associated with block interfaces and involves the exchange of ghost-cell solution values and conservative flux corrections at every stage of the multi-stage time integration procedure. Message passing of the ghost-cell values flux corrections is performed in an asynchronous fashion with gathered wait states and message consolidation, and as such, typically amounts to less than 5-10% of the total processor time.

An example illustrating the adaptation of a two-dimensional multi-block quadrilateral mesh for an inviscid transonic bump channel flow at Mach 0.85 is shown in Fig. 2. The pressure contours and mesh block structure are shown for the initial mesh consisting of 8 blocks and 8,192 cells, $(N_x,N_y) = (32,32)$ and after five refinements with 125 blocks

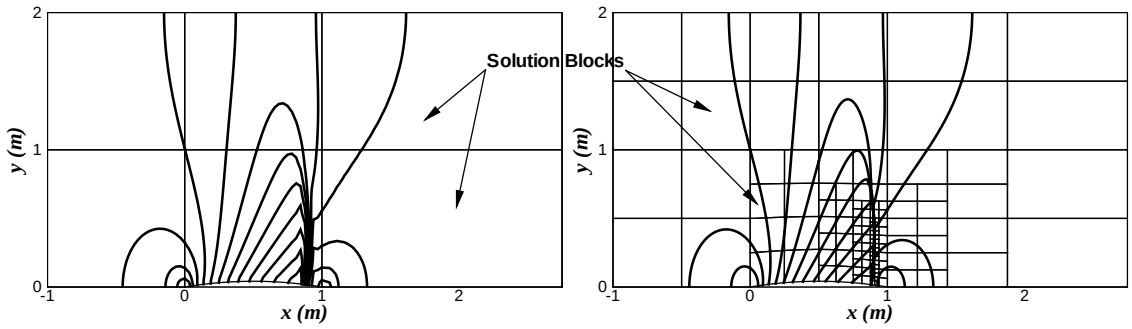


Figure 2: Pressure contours and body-fitted block structure (cells not shown) for an inviscid transonic bump channel flow at Mach 0.85 on the initial mesh of 8 blocks and 8,192 cells, $(N_x,N_y)=(32,32)$ (left panel, entire domain not shown) and after five refinements with 125 blocks, 128,000 cells, and $\eta=0.985$ (right panel).

and 128,00 cells in left and right panels, respectively. It can be seen that the algorithm has successfully refined the mesh at the shock located near the trailing edge of the bump. A measure of the efficiency of the block-based AMR scheme can be defined by

$$\eta = 1 - N_{\text{cells}} / N_{\text{uniform}}, \tag{3.1}$$

where N_{cells} is the total number of cells and N_{uniform} is the total number of cells that would have been used on a uniform mesh composed of cells of the finest size on the current mesh. After five levels of refinement, the mesh has an efficiency of $\eta = 0.985$, indicating the ability of the approach to deal with flows having disparate spatial scales, providing reduced numbers of cells while maintaining cell resolution in areas of interest.

4 Mesh adjustment scheme for embedded boundaries

The motivation for developing a mesh adjustment scheme for arbitrarily embedded boundaries stems from the desire to numerically predict flow involving complex bodies which can be moving relative to the computational domain. The scheme proposed here requires only local alterations of the mesh allowing for its implementation within the context of the block-based AMR scheme described previously. Many of the techniques used by Cartesian cut-cell methods, such as bounding box filters and ray-casting [1], are used here to improve the efficiency of the mesh adjustment algorithm.

An illustration of the mesh adjustment algorithm is given in Figs. 3 to 11. In Fig. 3 the mesh of the initial domain is shown along with two embedded NACA0012 aerofoils. These objects are stored as n-sided polygons or n-element polylines with additional data stored at each vertex of the shape to specify local characteristics of the embedded interfaces (such as boundary condition type, velocity, and temperature). Although not required for this scheme, for illustrative purpose, the domain is discretized by a uniform Cartesian mesh (the proposed algorithm is more generally applicable to any body-fitted

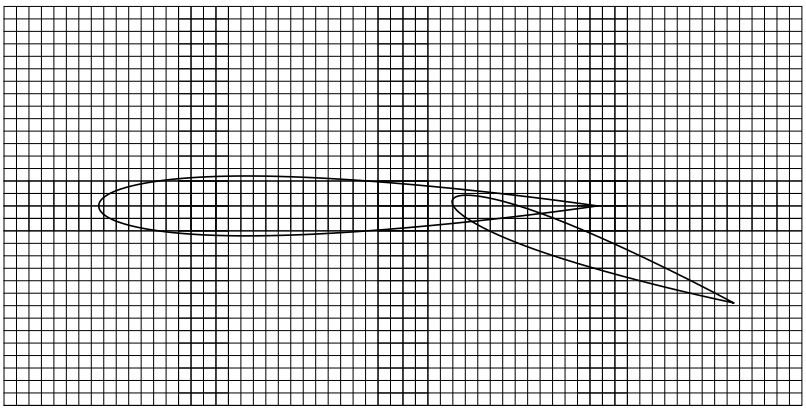


Figure 3: Example discretized domain with 4,096 cells, $(N_x,N_y)=(32,128)$ and two embedded boundaries for the illustration of the mesh adjustment algorithm.

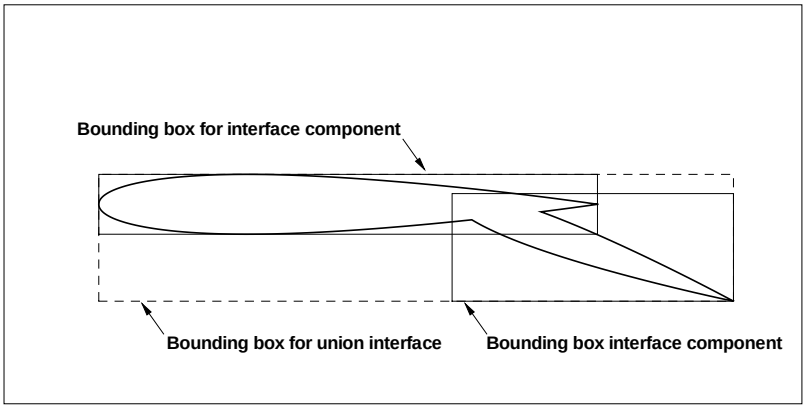


Figure 4: Interface of union of the two embedded NACA0012 aerofoils and their associated bounding boxes.

multi-block mesh). As for Cartesian cut-cell methods, the union of the embedded boundary components is found before the mesh is adjusted to the boundary [1]. This allows for a more robust and efficient implementation of the algorithm. The Weiler-Atherton algorithm is an efficient and robust method for finding the union of two intersecting polygons [53] and is adopted here. The union of the two embedded NACA0012 aerofoils is shown in Fig. 4. Included in this figure are the bounding boxes for the individual interface components as well as the union interface. A bounding box is defined by the maximum and minimum Cartesian coordinates of the object of interest and, therefore, contains that object entirely. The bounding boxes of the interface components are used as filters to quickly determine which computational cells are potentially intersected by or are contained by the associated union interface.

The first stage of the mesh adjustment procedure is to tag/paint each cell as an *active* cell, *inactive* cell, or an *unknown* cell and every node of each cell as *known* or *unknown*.

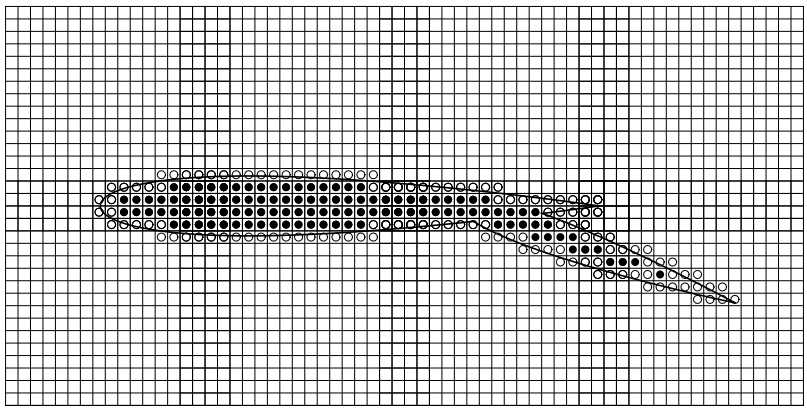


Figure 5: Tagging of the computational cells: cells with filled circles are tagged as inactive (internal to the embedded boundary), cells with hollow circles are tagged as unknown and are candidates for adjustment, and all other cells are active (external to the embedded boundary).

Note that all cells and nodes are initialized as active and known, respectively. The unknown nodes (of an unknown cell) are candidates for adjustment. Most of the active cells (and known nodes) are quickly identified by comparing each of the nodes of a cell with bounding boxes of each of the interface components. If all of the nodes are deemed outside the bounding box then the cell is deemed active. Otherwise, each edge of the cell with contrasting tags are tested for potential intersection points with each edge of the interface union polygons. If an intersection exists between the cell edge and an interface edge then the nodes of the cell edge and the cell itself are tagged as unknown. If no edge/interface intersections exist, ray-tracing is performed to determine if the cell is deemed active or inactive. The ray-tracing algorithm is performed by counting the number of intersections between the line composed of the cell centroid and a reference point within the embedded boundary (typically the centroid) and each edge of the embedded boundary. An odd number of intersections indicates that the cell is outside the interface (active) and an even number of intersections indicates that the cell is inside the interface (inactive). Note that the actual point of intersection is not required during this initial tagging procedure and the point of intersection must be contained on both line segments. The result of this procedure is shown in Fig. 5.

The NACA0012 aerofoil includes a sharp trailing edge. To provide an accurate representation of the embedded boundary, the next step of the adjustment algorithm is to identify the unknown cell that contains these sharp points and move the cell's node that is closest to the sharp point onto that point. Note that the points of union between the two NACA0012 aerofoils components are also defined as sharp points. The adjustment of the mesh to these four sharp points is high-lighted by the arrows in Fig. 6. All mesh nodes that are adjusted to the embedded boundary are tagged as *aligned*.

The mesh adjustment is comprised of two main stages. The primary adjustment involves merely relocating the unknown mesh nodes that are closest to the intersection

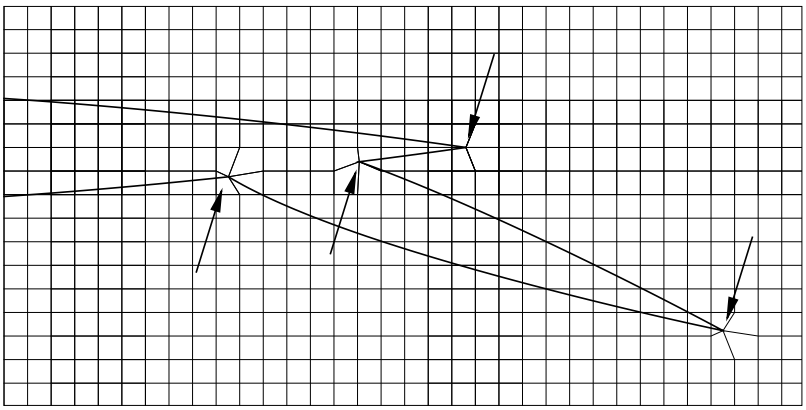


Figure 6: Adjustment of the mesh to sharp corners defined in the embedded boundary: two at the trailing edge of the aerofoils and two at points of union between the two components.

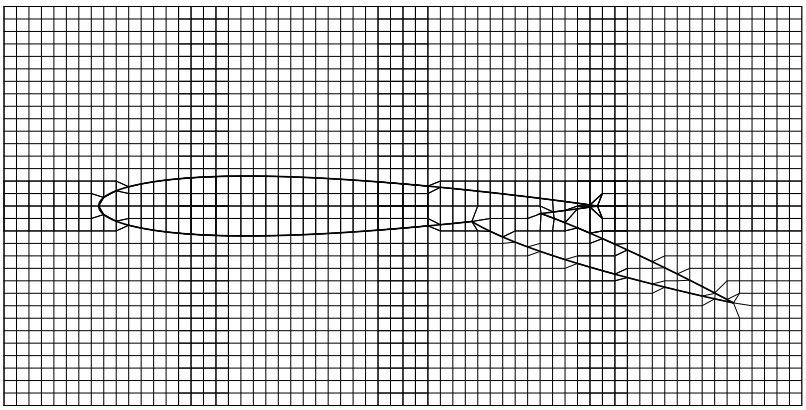


Figure 7: Result of the primary mesh adjustment.

point between the spline defining the embedded boundary and the mesh lines. The resulting adjusted grid is shown in Fig. 7. All potential adjustment directions (along north, south, east, or west mesh lines) and intersection points are tabulated before the actual adjustment occurs. This allows for the parsing/reduction of the adjustments near zones of interest such as near the trailing edge of an aerofoil. Here the nodes internal to the aerofoil are potentially an equal distance from the embedded body in opposite directions, e.g., north and south. These nodes are marked for adjustment in both directions. To ensure a symmetric adjustment, these choices are eliminated and the neighbor nodes are marked for adjustment instead, e.g., the $j+1$ and $j-1$ nodes to the north and south are marked for adjustment. As done previously, all mesh nodes that are adjusted to the embedded boundary are tagged as *aligned*. It was found that the robustness and accuracy of the adjustment procedure was greatly improved if this primary adjustment procedure was performed twice in succession. In particular, this is essential on coarse meshes in

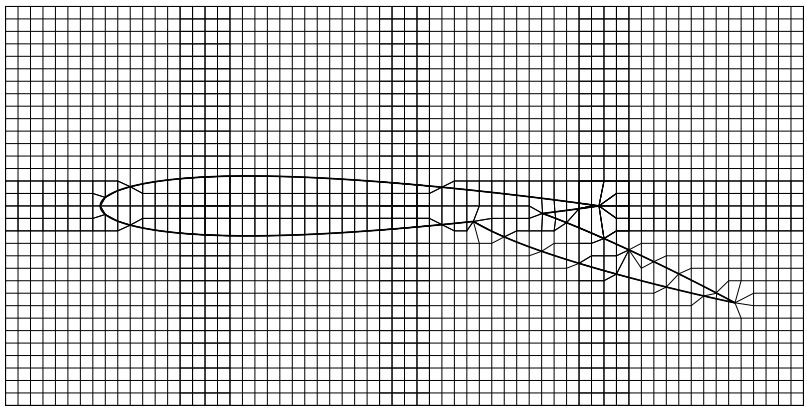


Figure 8: Result of the second mesh adjustment step.

which an embedded boundary may pierce the edge of a cell more than once. Note that aligned nodes are not permitted to be readjusted.

It can be seen in Fig. 7 that the primary mesh adjustment creates computational cells that are bisected diagonally by the embedded boundary. The second mesh adjustment stage is required to account for these cells. This is accomplished by choosing the closest not-aligned node to the embedded boundary and relocating it to that boundary point, as shown in Fig. 8. This adjustment can potentially create a cell with zero area – either the cell under consideration or one of its neighbors. If this occurs, the adjusted node is reset to its initial location and the other not-aligned node is adjusted. If no valid choice is available then the mesh is under-resolved and the mesh cannot be adjusted without refinement. Some triangular cells are generated as a result of this second adjustment step. These cells are treated as degenerate quadrilateral cells with two coincident nodes. Note that the degenerate edges are only generated on the embedded boundary.

The final stage of the mesh adjustment algorithm is to re-tag all of the cells previously marked as unknown. The ray-tracing algorithm discussed above is used to determine if the adjusted cell is inside (active) or outside (inactive) the embedded boundary. In situations with multiple embedded boundaries which could potentially have unique boundary conditions and motion characteristics, all inactive cells are tagged using the number of the interface of union that it is associated with (possible values range from one to the number of union interfaces). All union interfaces are stored in an array and, therefore, this tagging method allows for quick identification of the associated embedded boundary. All active cells are tagged as zero. The final adjusted mesh is given in Fig. 9.

One of the primary advantages of the proposed mesh adjustment scheme is that it will generate a piece-wise linear representation of the embedded boundary while still maintaining the (i,j) data structure of the original body-fitted mesh, as shown by Fig. 10 for a section of the embedded boundary. Note that the inactive nodes are retained to maintain the mesh data structure and may be reactivated in computations involving moving boundaries. The resulting mesh allows for accurate calculation of cell areas and straight-

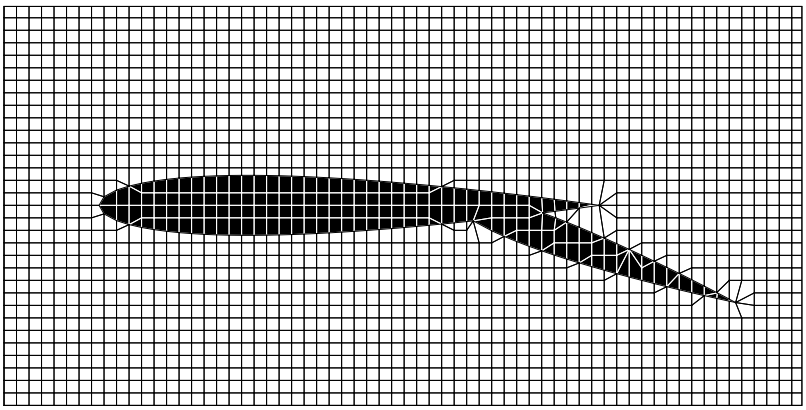


Figure 9: Final tagging of the computational cells: shaded cells are tagged as inactive (internal to the embedded boundary) and all other cells are active (external to the embedded boundary).

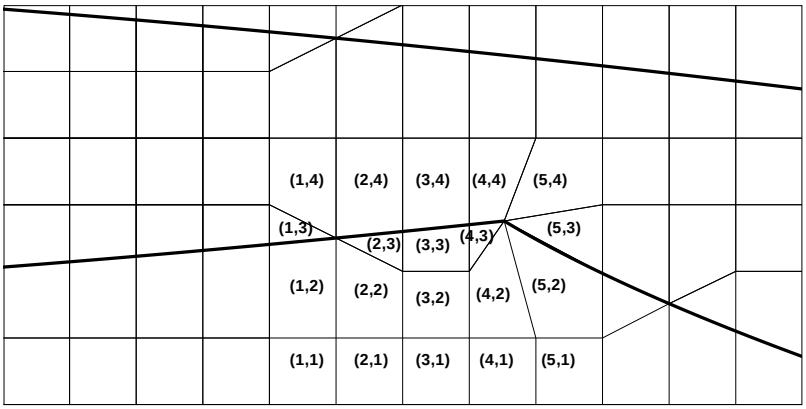


Figure 10: Example of the (i,j)-indexing on an adjusted mesh.

forward application of boundary conditions. Another advantage of this mesh adjustment algorithm is that very small cut-cells are not introduced and cell merging techniques are not required. The ratio of the smallest to largest neighbor cell areas produced by the proposed embedded boundary treatment has been found to be not less than about 0.2–0.25. Since only local alterations are made to the mesh, the need for inter-solution-block communication is not required and is, therefore, transparent to the parallel block-based AMR scheme. Moreover, application of the block-based AMR allows for an improved representation of the embedded boundary.

Use of the AMR scheme for the two embedded aerofoils is presented in Fig. 11. The block structure of the initial adjusted mesh is shown in the left panel and includes 8 blocks and 2,048 cells. After four refinements, 497 blocks and 127,232 cells are used. To avoid excessive tangling/skewing of the mesh, the mesh is unadjusted (restored to its original form) before the mesh is refined and is readjusted after refinement. Note that

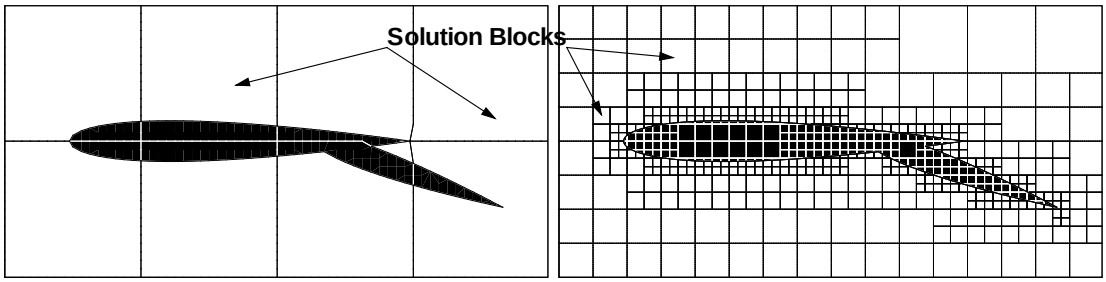


Figure 11: Application of the mesh adjustment scheme with the block-based AMR procedure. The initial mesh (left panel) contains 8 blocks and 2,048 cells and after four refinements (right panel) with 497 blocks and 127,232 cells. The mesh block structure is shown (cells not shown).

to perform the restriction or prolongation of the solution information from the old mesh to the new mesh, the adjusted mesh before the refinement must be saved. Three copies of the grid are required: the unadjusted mesh, the adjusted mesh, and the previously adjusted mesh (before AMR is performed). Prolongation of the solution information from a parent solution block into the child solution block is performed by simple injection. Restriction of the solution information is determined using an area-weighted average of the fine cell solution information of the cells that intersect with the coarse cell. The Weiler-Atherton algorithm can also be used to determine the polygon of intersection between the two cells (triangular or quadrilateral polygons) [53].

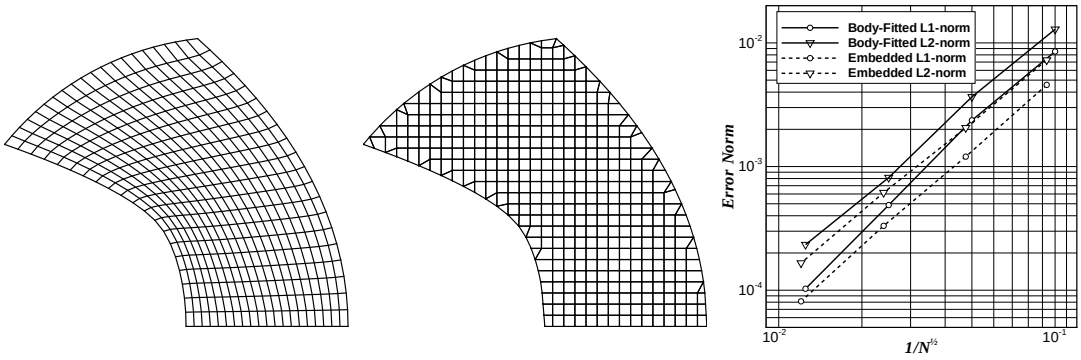
5 Spatial discretization

5.1 Hyperbolic flux evaluation

The hyperbolic flux that must be evaluated at the face of each finite volume includes the inviscid flux and the flux contribution due to the moving boundaries and is determined using a higher-order Godunov scheme. Upwind finite-volume schemes for the gasdynamic equations were originally introduced by Godunov in 1959 [21]. Application and development of these schemes for the gas-dynamic equations has been well documented in literature (see the textbooks by Hirsch [24], Laney [31], and Toro [48] for a detailed review of these schemes). The hyperbolic numerical fluxes at the faces of each cell are determined from the solution of a Riemann problem. Given the left and right initial state vectors, $\mathbf{U}_l$ and $\mathbf{U}_r$, and velocity $\vec{w}$ of the cell interface, the numerical flux is given by

$$\left(\vec{\mathbf{F}}-\vec{w}\mathbf{U}\right)\cdot\hat{n}=\mathcal{F}(\mathbf{U}_l,\mathbf{U}_r,\vec{w},\hat{n}), \tag{5.1}$$

where $\mathcal{F}$ is evaluated by solving the Riemann problem in a direction defined by the normal to the face, $\hat{n}$. The left and right initial conditions are determined using limited piecewise linear solution reconstruction [5, 52]. Roe's approximate Riemann solver [42] can be directly applied to Riemann problems with moving cell interfaces if the wavespeeds are corrected into a stationary frame of reference as given for the x-component flux by



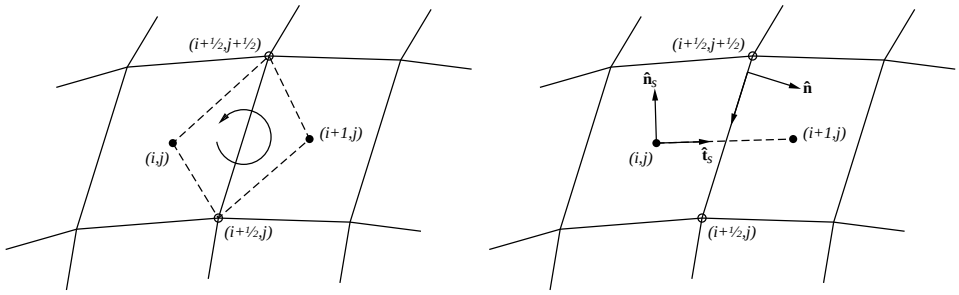
(a) (b) (c)

Figure 12: Ringleb's flow: (a) Body-fitted mesh with 400 cells, (b) Adjusted mesh with 429 cells, and (c) Computed norms of the solution error.

$u-w-a$, $u-w$, $u-w$, $u-w+a$. Note that for stationary (non-evolving) embedded boundaries, the preceding spatial discretization scheme is strictly conservative and monotonicity of the solution is enforced via the slope limiting procedure.

5.2 Ringleb's flow

In order to demonstrate the accuracy of the inviscid spatial discretization procedure for problems involving embedded boundary treatment, the predictions of the proposed algorithm are considered for a test problem for which an exact analytic solution exists. Ringleb's flow is a hodograph solution to the Euler equations that is widely used in validation studies [4,11]. The flow pattern involves an isentropic, irrotational flow contained between two streamlines. The accuracy of the inviscid spatial discretization was assessed by comparing the computed solution on a series of uniformly refined meshes to the analytic solution. See Ref. [43] for a brief description of the analytic solution and the configuration used here. The inflow boundary is defined by a subsonic iso-tach contour and a mixed supersonic and subsonic outflow occurs at the lower boundary. The L_1- and L_2-norms of the difference in the solution densities were used as the measure of solution accuracy. The error-norms were computed for a body-fitted mesh and for a Cartesian mesh with an embedded boundary. Sample body-fitted and adjusted meshes are shown in Figs. 12(a) and (b) with 400 and 429 cells, respectively. The left and right states for the evaluation of the Riemann problem on the inflow and outflow boundaries are determined from a characteristic boundary condition with a specified static pressure based on the exact solution at the Gauss point. The L_1- and L_2-norms of the solution error are plotted in Fig. 12(c). The slopes of the L_1- and L_2-norms are 2.11 and 1.94 for the body-fitted mesh, respectively. For the Cartesian mesh, the slopes of the L_1- and L_2-norms are 1.93 and 1.81, respectively. However, considering only the two finest meshes, the L_1- and the and L_2-norms are 2.03 and 1.89, respectively. This would indicate that the scheme is



(a) (b)

Figure 13: Definition of (a) the diamond-path used for the gradient reconstruction of the east-face elliptic (viscous) flux and (b) the local coordinate systems.

indeed second order accurate, even with the embedded boundary treatment.

5.3 Elliptic flux calculation

The calculation of the viscous flux of the compressible Navier-Stokes equations requires the knowledge of the solution as well as the gradient of the solution at cell faces. The numerical flux can be stated by

$$\vec{\mathbf{G}}\cdot\hat{n} = \mathcal{G}(\mathbf{U},\vec{\nabla}\mathbf{U},\hat{n}), \tag{5.2}$$

where the dependence on the gradient of the solution quantities dictates that the flux is elliptic by nature. In particular, the gradients of the velocity and temperature fields are required. These gradients can be found by performing a Green-Gauss integration around the diamond-path [12] as depicted in Fig. 13(a). The linearity-preserving weighting scheme of Holmes and Connell is used to determine the solution data at the nodes [26]. Use of this scheme ensures that the node solution data is linearly constructed from the solution data stored at the centroid of all neighboring cells. The gradient is determined by evaluating

$$\vec{\nabla}\phi = \frac{1}{A}\sum_k \phi_k \hat{n}_k \Delta\ell_k, \tag{5.3}$$

where ϕ is the solution quantity of interest, A is the area of the quadrilateral specified by the contour path, $\Delta\ell_k$ is the length of face k, $\hat{n}_k$ is the unit outward normal of face k, and ϕ_k is an approximation of the solution variable at the centre of the face. An average of the vertex solution data is used to determine the solution at the centre of the face. For the integration at the east face of cell (i,j) given in Fig. 13(a), Eq. (5.3) can be expanded to give

$$\vec{\nabla}\phi = \frac{1}{\hat{n}\cdot\hat{t}_s}\left(\frac{\phi_{i+1,j}-\phi_{ij}}{\Delta s}\hat{n} + \frac{\phi_{i+\frac{1}{2},j+\frac{1}{2}}-\phi_{i+\frac{1}{2},j-\frac{1}{2}}}{\Delta\ell}\hat{n}_s\right) \tag{5.4}$$

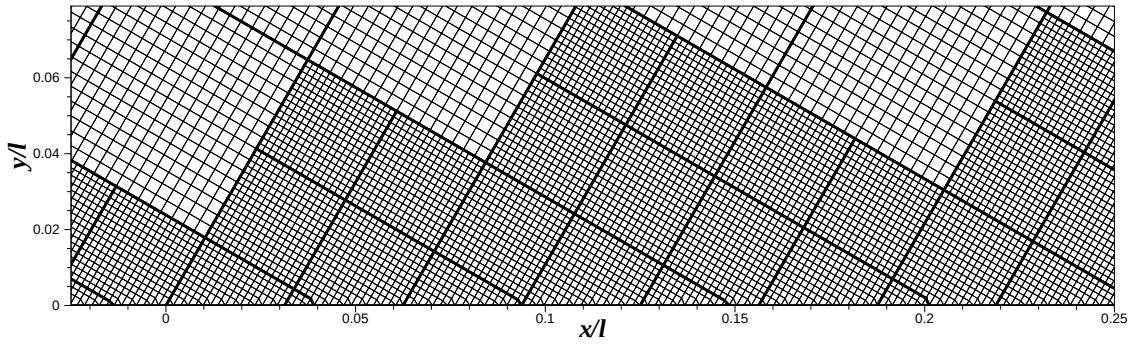


Figure 14: Adjusted mesh at the leading edge of an embedded flat plate for a refined multi-block grid rotated at 30° (6 levels of refinement with 360 active blocks and 80,023 active cells). Block boundaries are indicated by the thick black lines. Each block contains 256 cells, $(N_x,N_y)=(16,16)$.

for the geometry specified in Fig. 13(b) where $\Delta\ell$ and Δs are the length of the cell face and the distance between cell centres, respectively. At a solid wall boundaries (embedded or otherwise), the diamond path integration can be readily simplified to enforce specific conditions such as the no-slip condition for the velocity and either adiabatic or isothermal conditions for the temperature.

5.4 Laminar flat plate boundary layer

The computation of laminar flow over a flat plate is now considered to explore the accuracy and capability of the proposed scheme for predicting moderate Reynolds number viscous flows. Coirier and Powell [12] considered this case to investigate the use of the Cartesian cut-cell approach for computing viscous flows and were able to show accurate prediction of the mean-flow quantities. However, the resulting skin-friction coefficient proved to be quite oscillatory. This was a direct result of the extremely small cut-cell generated by Cartesian cut-cell approach which makes the creation of a consistent and accurate viscous discretization virtually impossible.

In this work, a non-axis aligned flat plate is embedded at 30° to the mesh. The freestream Mach number is 0.2 and the Reynolds number (based on the length of the plate) is 10,000. The mesh in the vicinity of the leading edge of the plate is shown in Fig. 14. The initial mesh consists of 360 active blocks and 80,023 active cells after six refinements. The predicted skin-friction coefficient for a Reynolds number of 10,000 is shown in Fig. 15(a) for the initial mesh and after two additional refinements. The initial mesh does not provide the desired resolution in the boundary layer and the skin-friction coefficient is slightly under predicted. Improvements in the drag estimation is provided by the additional mesh refinements. More importantly, the oscillatory nature of the skin-friction predictions produced by the cut-cell method are experienced here to a much smaller degree. The cut-cell method of Ref. [12] resulted in peak-to-peak changes in the skin-friction coefficient of approximately 0.0025 whereas the variations in the current

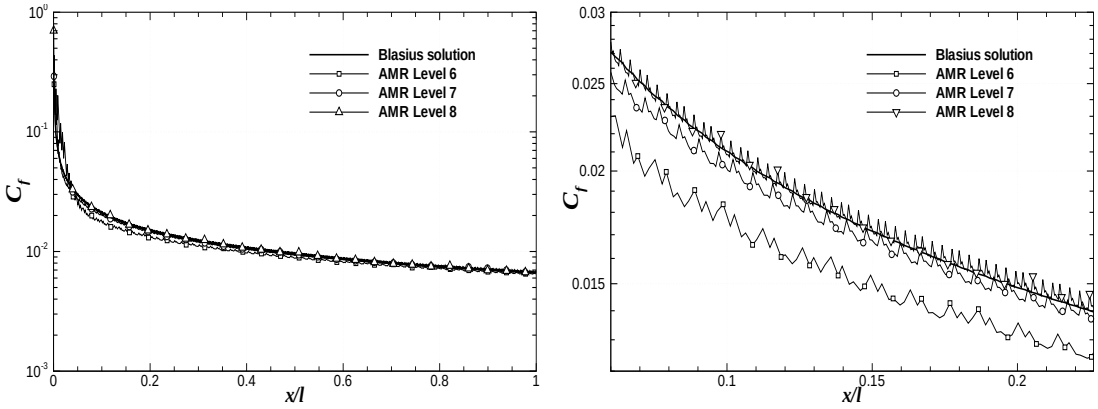


Figure 15: Estimation of the skin-friction coefficient for a non-axis aligned flat plate at $\mathrm{M}=0.2$ and $\mathrm{Re}=10{,}000$ for six, seven, and eight levels of refinement: (a) prediction for the entire plate and (b) a close-up of the prediction near the leading edge of the plate.

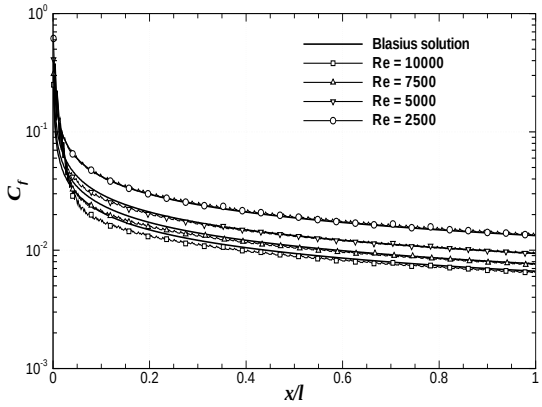


Figure 16: Estimation of the skin-friction coefficient for a non-axis aligned flat plate at $\mathrm{M}=0.2$ for Reynolds numbers of 2,500, 5,000, 7,500, and 10,000 on the mesh shown in Fig. 14.

predictions are at least an order of magnitude less. Moreover, Fig. 16 indicates that, as the flow Reynolds number is decreased, less resolution is required to accurately predict the skin-friction coefficient in a monotonic fashion. This is because the boundary-layer is generally thicker and contains more computational cells. The preceding results would seem to confirm the applicability of meshes created by the adjustment scheme for viscous flows, at least for low-to-moderate Reynolds numbers.

6 Mesh adjustment for dynamic embedded boundaries

By performing only local alterations to the mesh, the motion of the embedded boundaries can be accomplished by first unadjusting the mesh and then re-applying the mesh

adjustment algorithm according to the new location of the embedded boundaries. This is the approach used by the Cartesian cut-cell community [6, 27, 34]. In the present work, embedded boundaries undergoing a prescribed rigid body motion or evolving motion due to a physical process are also considered. Evolving embedded boundaries are modeled using a parallel block-based AMR level set method which is described in Section 7.

The algorithm for moving embedded boundaries is performed in the following sequence of steps:

1. Determine the velocity at all spline points of each interface component.
2. Determine the union of the embedded boundaries.
3. Perform mesh adjustment algorithm.
4. Perform the solution of equation (2.3). The velocity at the Gauss point of cell edges that are aligned with the embedded boundaries are determined from the corresponding interface component.
5. Determine the new location of the embedded boundary components and recompute the velocity at all spline points if necessary.
6. Determine the union of the embedded boundaries.
7. Unadjust the mesh (store previous adjustment).
8. Perform mesh adjustment and redistribute solution information as required.
9. Loop through steps 4-8 until the computation is finished.

Unadjusting the mesh before performing the mesh adjustment at the new locations for the embedded boundaries avoids excessive tangling of the mesh, however, other complications can arise. Due to this step, some of the computational cells that were formerly active may now be inactive. Conversely, some inactive cells may now be active. A reaveraging of the solution data is required to ensure conservation of the solution quantities. The solution content of any cells that have been newly deactivated is area-averaged into neighboring active cells. The solution content of the all of the active cells involved in the mesh adjustment procedure are then determined by taking the area-average of all of the previously active cells that intersect with newly active cells. This is similar to the restriction process required for AMR and has the same grid requirements. Here, the previously adjusted mesh corresponds to that of the previous time-step.

Note that the time-step of the explicit time-marching scheme is restricted to ensure that an entire cell may not be overtaken during a single step. An estimation of the maximum allowable time-step is given by

$$\triangle t = \beta \min\left[\min_k\left(\frac{A}{(a+|\vec{v}-\vec{w}_k|+|\vec{w}_k|)\Delta\ell_k}, \frac{2A}{(\Delta\ell_k)^2\nu}\right)\right], \tag{6.1}$$

where the index, k, refers to the faces of the cell. The first condition corresponds to the stability criterion for hyperbolic equations, corrected to account for the motion of the cell interface. This formulation is similar to that used by Hunt [27]. The second condition is simply the Neumann condition for elliptic equations. The *CFL*-number, β, is used to ensure stability of the time-stepping scheme.

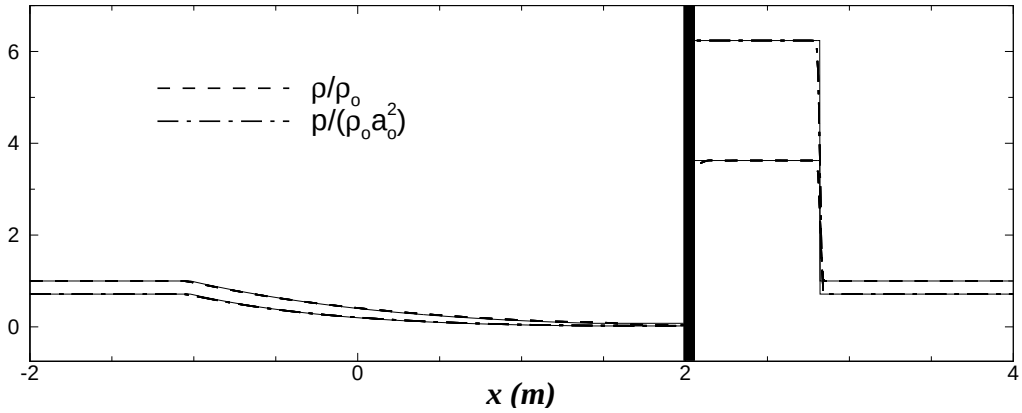


Figure 17: Non-dimensional pressure and density ratios for the 1D moving piston at time 3 ms for a piston moving at Mach 2.

6.1 Piston problem

A one-dimensional piston problem is now considered to assess the conservation properties of the proposed scheme for moving boundaries [34]. Although the moving piston problem is a rather simple inviscid problem, analytic expressions can be determined for the resulting shock wave that forms ahead of the piston and the rarefaction wave that is generated behind the piston and it provides a good test of the scheme's conservation properties. If the method accurately conserves mass, momentum, and energy at the moving interface, the predicted shock strength and speed must match the analytic expression. Poor agreement would indicate solution content may be lost at the moving surface.

The predicted non-dimensional pressure and density fields are shown in Fig. 17 for a piston moving at Mach 2 into a quiescent gas. Comparison with the analytical results reveals that the proposed numerical scheme accurately predicts the shock position and strength on the compression-side of the piston, as well as the rarefaction wave solution behind the piston, indicating that the conservation properties of the finite-volume formulation are maintained when a moving interface is introduced.

6.2 Oscillating aerofoil

Prediction of the fluid flow over an oscillating aerofoil is considered next and compared to the experimental measurements obtained by Landon [30] to provide further validation of the numerical method for moving embedded boundaries. The original experimental work was conducted to examine flow conditions experienced by helicopter blades. In particular, a NACA0012 aerofoil is undergoing an oscillatory motion about the quarter chord defined by $\alpha(t) = \alpha_0 + \alpha_m \sin(2\pi f t)$ where α is the angle of attack. The parameters $\alpha_0 = 0.016^\circ$, $\alpha_m = 2.51^\circ$, and $f = 62.5$ are the initial angle of attack, the amplitude of the oscillation, and the frequency of the oscillation. The freestream Mach number was

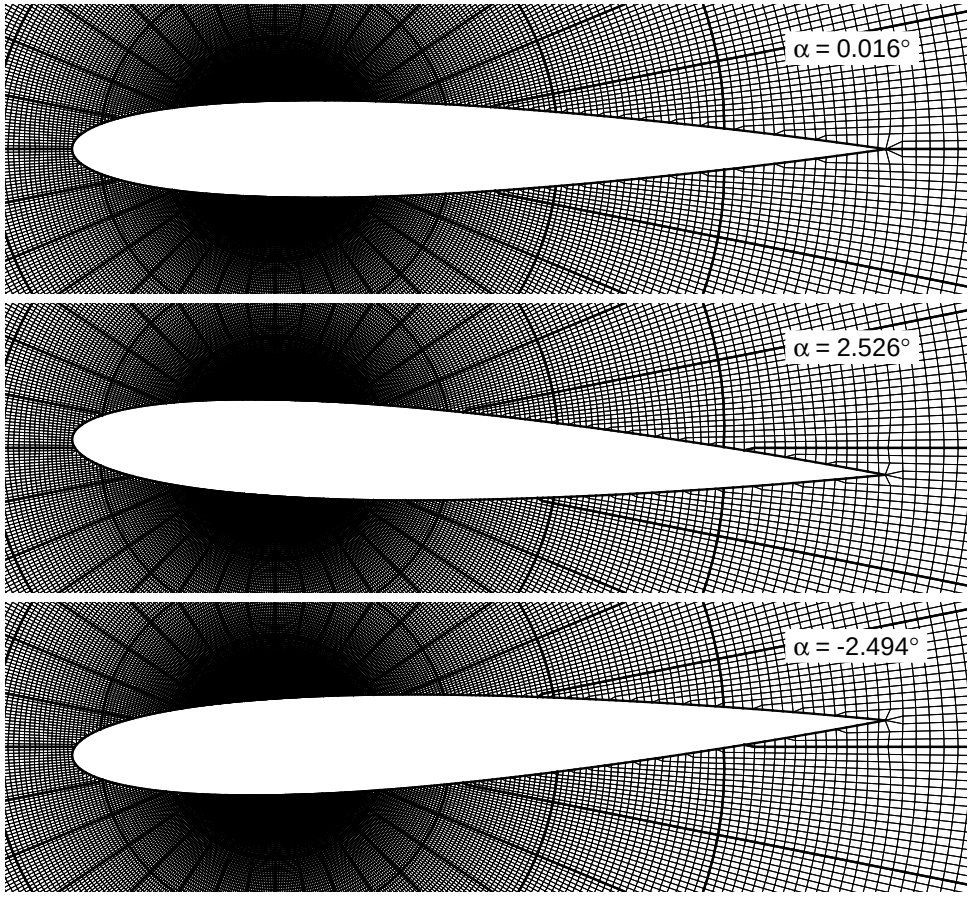


Figure 18: Close-up of the adjusted mesh at the initial angle of attack (top frame), $\alpha = 0.016°$, the extreme pitch-up angle of attack (middle frame), $\alpha = 2.526°$, and the extreme pitch-down angle of attack (bottom frame), $\alpha = -2.494°$. Block boundaries are indicated by the thick black lines. Each block contains 384 cells, $(N_x, N_y) = (16,24)$.

$M = 0.755$ and the flow Reynolds number based on the chord length was $Re = 5.5 \times 10^6$. This particular set of experimental results have been used in previous studies to validate numerical algorithms for inviscid (e.g., Dubuc et al. [17] and Murman et al. [34]) and turbulent (e.g., Chassaing et al. [10]) flow.

In this work, inviscid flow over the oscillating NACA0012 aerofoil is computed where the aerofoil is embedded on a cylindrical grid. The outer radius of the grid was located 32 chord lengths from the quarter chord of the aerofoil. The initial mesh consisted of two solution blocks with 384 cells each, $(N_x, N_y) = (16,24)$. To achieve an accurate representation of the aerofoil, five mesh refinements were performed in an area surrounding the embedded boundary. No further mesh refinement was used during the computation. The final mesh consisted of 248 blocks and 95,232 cells. A close-up of the mesh surrounding the embedded boundary at the initial angle of attack is shown in the top panel of Fig. 18.

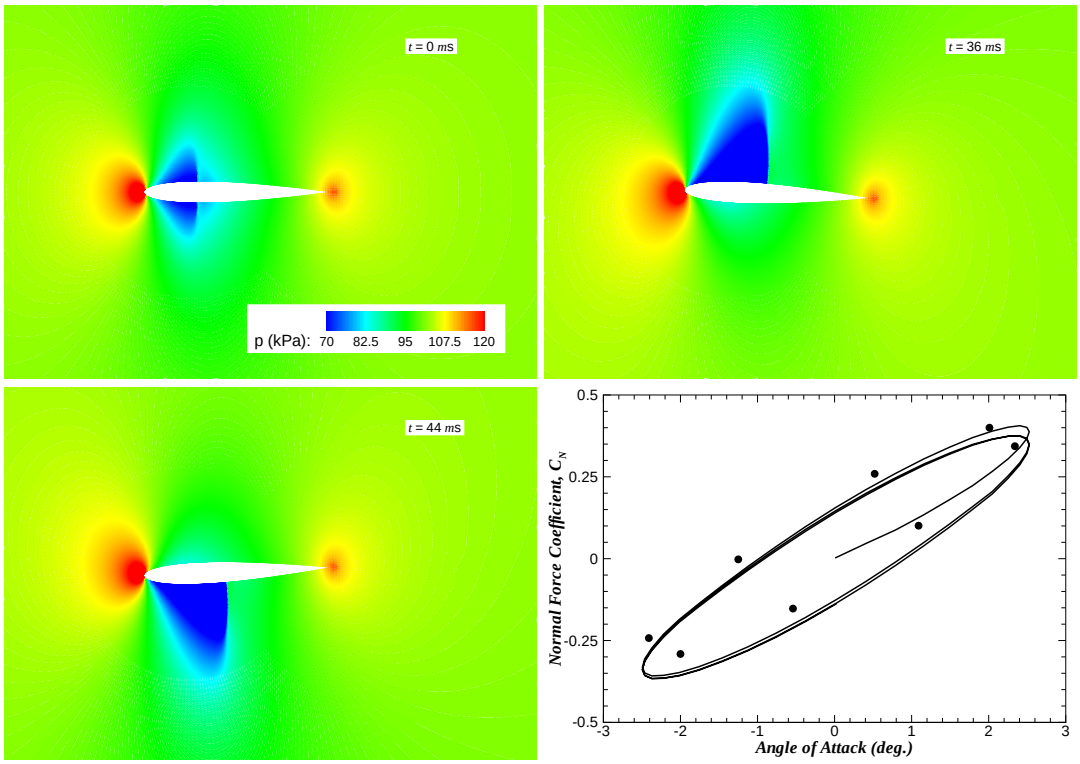


Figure 19: Top two and bottom left panels: pressure contours, 70-120 kPa, for the steady-state solution before the oscillation and at two times during the third period of oscillation, 36 and 44 ms. Bottom right panel: comparison of the computed normal force coefficient for varying angle of attack with the experimental results of Landon [30].

The block boundaries are indicated by the thicker black lines. Note that a steady-state solution is achieved at each mesh level before refinement and before the pitching of the aerofoil is started. The adjusted meshes at angles of attack corresponding to the extreme points in the oscillation are shown in the bottom two panels of Fig. 18.

The pressure contours for the steady-state solution before the oscillation begins are shown in the top right panel of Fig. 19. The contour levels range from 70-120 kPa. Shocks have been formed on the upper and lower surfaces of the aerofoil. The initial angle of attack corresponds to a slight pitch-up and, therefore, the shock on the upper surface is slightly stronger. The oscillating motion of the aerofoil was started after the steady-state solution was achieved. As the aerofoil pitches up the shock on the upper surface strengthens and moves towards the trailing edge of the aerofoil. The shock on the lower surface diminishes in strength. At the peak of the pitch-up the shock on the lower surface has disappeared as can be seen in the upper right panel of Fig. 19. Here, the pressure contours have been plotted at a time corresponding to 36 ms into the unsteady solution (first quarter of the third period). The converse of this situation occurs as the aerofoil pitches down. The shock on the upper surface begins to diminish and moves towards

the leading edge of the aerofoil, whereas a shock on the lower surfaces emerges from the leading edge, strengthens, and moves towards the trailing edge. At the extreme pitch-down point, the pressure contours are opposite to that of the peak pitch-up point as shown in the lower left panel of Fig. 19. The pressure contours are plotted 44 ms into the unsteady solution (third quarter of the third period). It should be noted that during this process, the initial shock strengths are not re-established. For example, the shock on the upper surface is stronger than that on the lower surface at the half period (initial angle of attack, pitching down from maximum pitch-up). The reverse occurs at the end of each period. This leads to the hysteresis in the normal force coefficient as indicated in the lower right panel of Fig. 19. In the figure, the normal force coefficient for varying angle of attack is plotted for the first three periods of the oscillation. After over-coming the initial response to the unsteady motion, the computed results are comparable to other numerical results [17,34] and are in good agreement with Landon's experimental results. The predictions provide additional support for the implementation and use of the mesh adjustment algorithm for dynamic embedded boundaries.

7 Level set method for evolving embedded boundaries

7.1 Level set method with adaptive mesh refinement

The level set method is an Eulerian front tracking scheme in which a scalar field, $\phi(x,y)$, is initialized as a signed-distance function where the zero-contour corresponds to the location of the embedded boundaries [37, 38, 45]. Evolution of the level set function is determined through the use of a scalar equation which can include motion due to a passive advection field, motion defined in the normal direction, and motion driven by the curvature of the interface. The level set method is used in this work to track embedded boundaries that are evolving due to physical processes. In particular, the motion of the combustion interface of a solid propellant rocket motor is of interest.

A standard implementation of the level set method for Cartesian mesh as outlined in the textbooks by Sethian [45] and Osher and Fedkiw [38] is used on a separate Cartesian grid that overlaps the computational domain of interest. The block-based AMR scheme described in Section 3 for the Euler and Navier-Stokes equations is also used during the solution of the level set method. The use of AMR can significantly improve solutions provided by the level set method and Milne [33] (see also Sethian [45]) has applied cell-based AMR methods [9] to improve mesh resolution in areas of high-curvature of the zero-level set. The curvature of the zero level set function is also used here as the refinement criteria. The curvature of the level set function, κ, is computed by

$$\epsilon \propto |\kappa| = \left| \vec{\nabla} \cdot \left(\frac{\vec{\nabla}\phi}{|\vec{\nabla}\phi|} \right) \right|. \tag{7.1}$$

User defined thresholds are used to determine the levels of high and low curvature in which block division and coarsening are to be pursued.

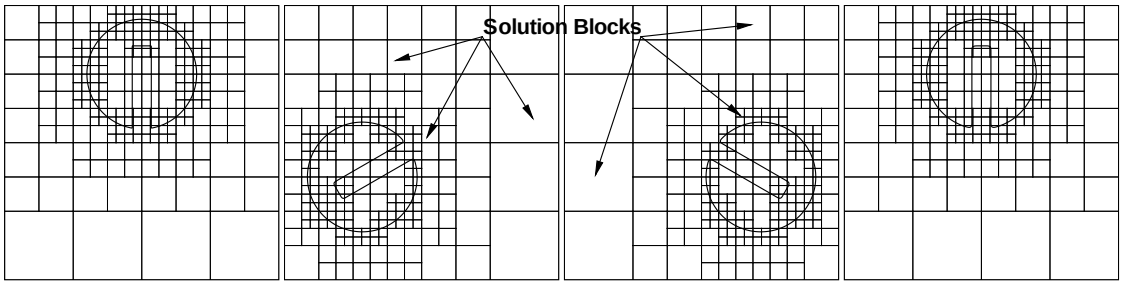


Figure 20: Zalesak's disk, from left to right: initial configuration (250 blocks, 100,000 cells, $(N_x,N_y)=(20,20)$, $\eta=0.756$), after 120° rotation (232 blocks, 92,800 cells, $\eta=0.773$), after 240° rotation (229 blocks, 91,600 cells, $\eta=0.776$), and 360° rotation (250 blocks, 100,000 cells, $\eta=0.756$). The zero contour is shown by the thick black line and the block boundaries are given by the this black lines.

7.2 Zalesak's disk

A popular test case for assessing the performance of front tracking algorithms is the evolution of a rigid slotted disk in a rotating flow, also known as Zalesak's disk [18,46]. The problem configuration used here is the same as that studied by Sussman et al. [46]. The rotating two-dimensional velocity field is taken to be $(u,v)=(\pi/314)(50-y,x-50)$. The initial configuration of the disk is shown in the left-most panel of Fig. 20 where the disk boundary is given by the zero contour shown by the thick black line. Initially, the mesh includes three levels of refinement with 250 blocks and 100,000 cells, $(N_x,N_y)=(20,20)$. The subsequent three panels of Fig. 20 show the computed position of the disk after an incremental rotation of 120° about the origin. These meshes contain 232, 229, and 250 blocks, respectively. The level set equation and the Eikonal equation are both solved using the fifth-order weighted essentially non-oscillatory spatial discretization procedure as outlined by Osher and Fedkiw [38] and a second-order predictor-corrector time-marching method. Solution of the Eikonal equation is required at every time-step to maintain an accurate and valid signed-distance function. It can be seen that the level set method coupled with the block-based AMR scheme is capable of maintaining an accurate definition of the disk throughout the rotation. Additional refinement could be employed to further reduce the diffusion of the disk as it rotates.

7.3 Solid propellant rocket motor

Prediction of the internal flow of a solid propellant rocket motor is now described to demonstrate the viability and capability of the proposed scheme for computing two-dimensional, axisymmetric, laminar flow with an evolving embedded boundary. The combustion of the propellant occurs in a thin, high temperature layer between the propellant grain and the main flow cavity, known as the combustion interface. This topologically complex surface evolves as the propellant burns and the hot gaseous combustion products are injected into the combustion chamber. For this work, the propellant grain

Table 1: Characteristics of the solid propellant and the combustion products.

Propellant		Combustion Products	
Density (ρ_s)	1740 kg/m^3	Specific gas constant (R)	318 J/kgK
Specific heat (c_s)	1510 J/kg K	Ratio of specific heats (γ)	1.21
Flame temperature (T_f)	3060 K	Absolute viscosity (μ)	8.19×10^{-5}kg/ms
Surface temperature (T_s)	1130 K	Thermal conductivity (κ)	0.184 W/mK

consists of a non-aluminized composite propellant composed of 80% oxidizer (ammonium perchlorate, AP) and 20% fuel (hydroxyl terminated polybutadiene, HTPB). Ignition of the propellant is assumed to be instantaneous and the burning rate dependent only on the pressure at the surface of the propellant grain as given by the empirical St. Robert relation [28], $r_{bs} = 16.18[p(\text{kPa})]^{0.50}$ mm/s. The burning of the solid propellant and production of the propellant gases is determined through the solution of a Riemann problem [43]. The characteristics of the solid propellant and the combustion products are summarized in Table 1.

The level set method described above is used to determine the evolution of the propellant grain based on the pressure-dependent burning rate. It is computed on a separate overlapping Cartesian mesh which is adapted independently from the body-fitted mesh. However, the mesh adjustment scheme requires an explicit representation of the embedded boundary at it's current location. Therefore, a contour tracing algorithm is required to provide an approximation of the zero level set. The algorithm proposed by Dobkin et al. [16] is used for this purpose. This method generates a piecewise linear approximation to the zero level set contour. It is worth noting that the computational effort associated with evolving the location of the combustion interface, tracing the location of the zero level set, and subsequently performing the mesh adjustment algorithm according to the new location of the embedded boundary was found to require approximately one-and-a-half to two times the computational work associated with the application of one step of the explicit time-marching scheme for this case.

The predicted results are presented in Figs. 21 to 23 for a cylindrical grain rocket motor with a 208.75 mm chamber length, 31.75 mm internal radius, a nozzle throat radius of 10.15 mm, an initial internal port radius of 20 mm, and an initial distance from the throat to the propellant grain of 65 mm. The initial geometry of the rocket motor can be seen in the top half of Fig. 21. Note that the entire length of the combustion chamber is not shown. The overlapping Cartesian domain used for the level set problem is shown in the lower half of Fig. 21. The location of combustion interface is given by the zero level set contour. Five levels of adaptive mesh refinement are used on the initial level set solution (56 blocks, 16,800 cells, $(N_x,N_y)=(30,10)$, and $\eta=0.985$). The steady state solution on the initial geometry is computed before the propellant grain is allowed to evolve. Six levels of adaptive mesh refinement are used during the steady state solution (168 blocks, 64,512 cells, $(N_x,N_y)=(24,16)$, and $\eta=0.945$). The steady state pressure and Mach number

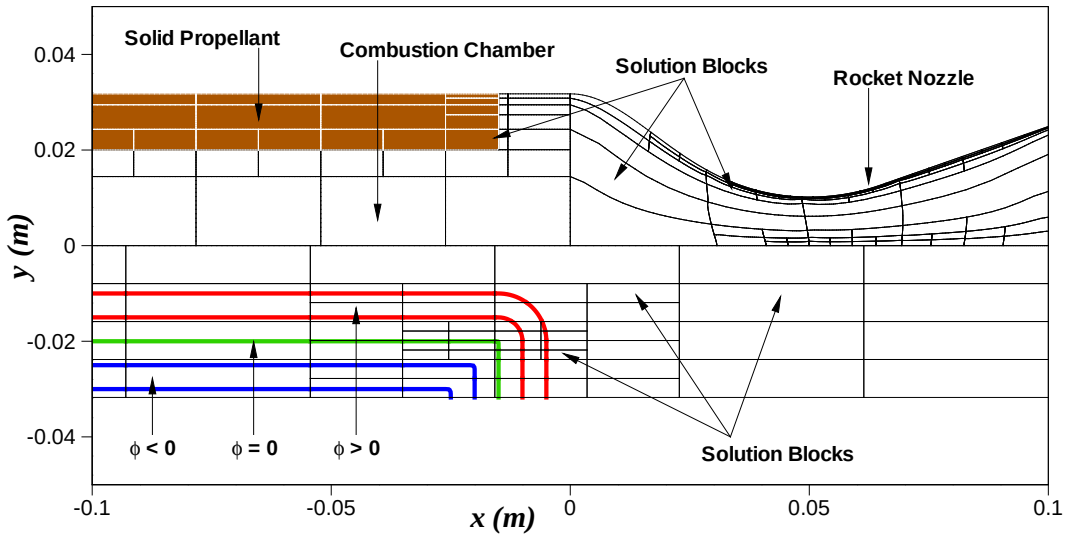


Figure 21: Block structure and configuration for the prediction of an internal rocket motor flow for the fluid (top panel) and level set (lower panel) domains before the evolution of the combustion interface. Five levels of adaptive mesh refinement are used on the initial level set solution (56 blocks, 16,800 cells, $(N_x,N_y)=(30,10)$, and $\eta=0.985$). Six levels of adaptive mesh refinement is used during the steady state solution (168 blocks, 64,512 cells, $(N_x,N_y)=(24,16)$, and $\eta=0.945$).

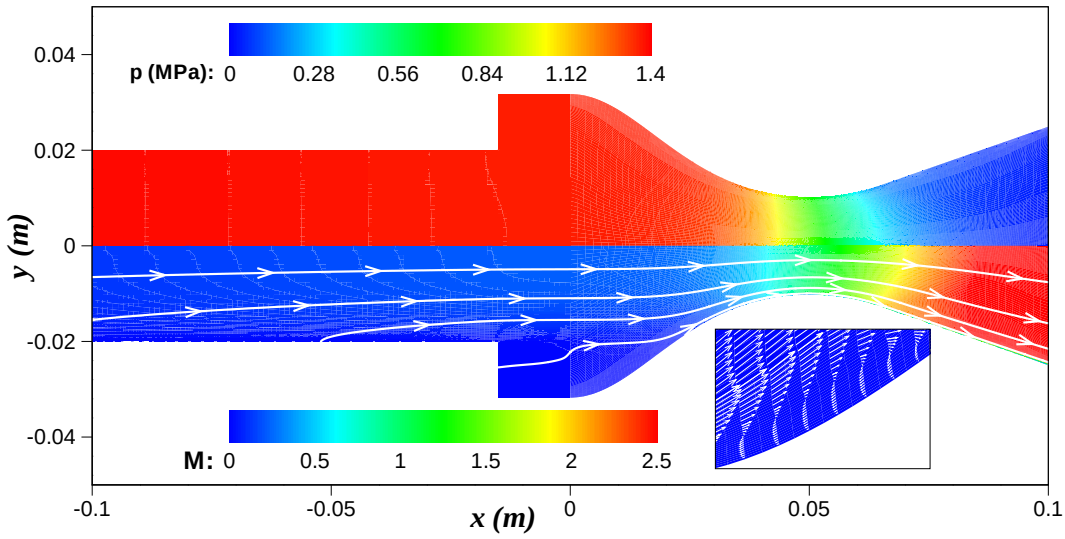


Figure 22: Predicted steady-state pressure (top panel) and Mach number (bottom panel) distribution for the initial configuration of the cylindrical grain rocket motor. The streaklines are also shown in the bottom panel. The inset contains the velocity vectors which indicate a low speed recirculation zone near the beginning of the converging section of the nozzle.

contours are shown in the upper and lower panels of Fig. 22, respectively. The burning of the solid propellant leads to a head end pressure in excess of 1.4 MPa and produces sonic

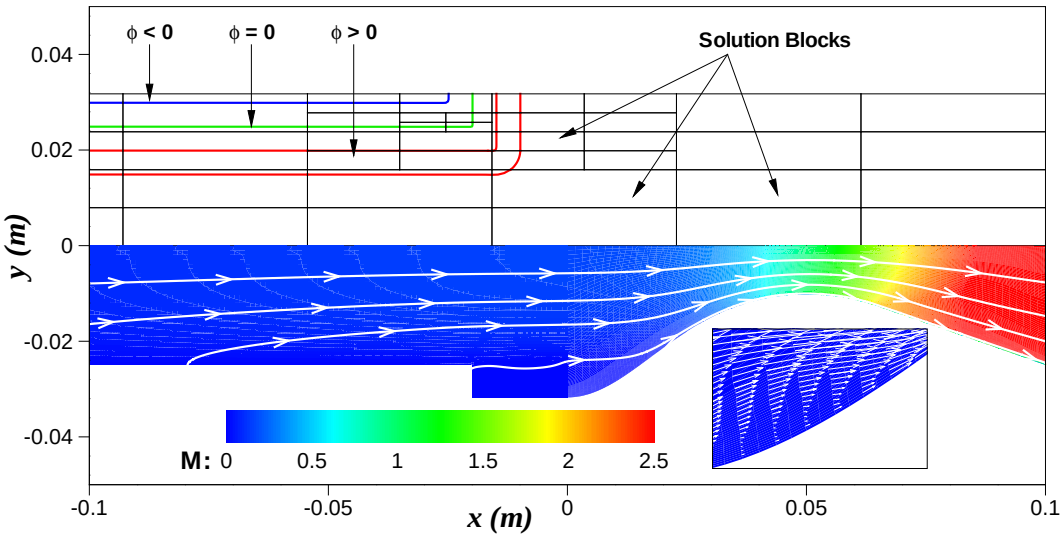


Figure 23: Predicted cylindrical grain rocket motor solution 8 ms after the combustion interface is allowed to evolve. The level set solution and block structure (41 blocks, 12,300 cells, and $\eta = 0.92$) are shown in the top panel. The Mach number distribution and streaklines are shown in the bottom panel. The adjusted body-fitted mesh includes 189 blocks and 72,576 cells ($\eta = 0.938$). The inset contains the velocity vectors which indicates that the low speed recirculation zone near the beginning of the converging section of the nozzle has decreased in size.

flow conditions at the nozzle throat and supersonic outflows in the rocket nozzle with Mach numbers over 2.5. The streaklines are also depicted in the lower half of Fig. 22. The inset contains the velocity vectors which clearly indicate a low speed recirculation zone near the beginning of the converging section of the nozzle.

The predicted solutions 8 ms after the combustion interface is allowed to evolve is shown in Fig. 23. The level set solution and block structure are given in the upper panel of this. At this time, the Cartesian mesh for the level set problem contains 41 blocks and 12,300 cells. The Mach number distribution and streaklines are given in the lower panel. The adjusted body-fitted mesh contains 189 blocks and 72,576 cells (the block structure is not shown in the figure). It can be seen that just over 40% of the propellant has been burnt at this time. The low-speed recirculation region shown in the inset has decreased in size and strength since the sudden expansion after the propellant grain has decreased in height. As a result, the flow from the main cavity is able to interact more directly with the converging section of the nozzle.

8 Discussion and concluding remarks

A mesh adjustment scheme has been proposed and described in which a body-fitted multi-block mesh is locally adjusted to embedded boundaries that are not necessarily aligned with the mesh. This scheme allows for quick and robust mesh generation in-

volving complex embedded boundaries as well as enables the solution of unsteady flow problems involving bodies moving relative to the flow domain. The primary advantages of this mesh adjustment algorithm are that the mesh alteration is local, the structured nature of the mesh blocks is maintained, very small cut-cells are not introduced, and cell merging techniques are not required. The ratio of the smallest to largest neighbor cell areas produced by the proposed embedded boundary treatment is not less than about 0.2–0.25. The viability of this scheme has been demonstrated for stationary and moving embedded boundaries involving inviscid and laminar flow. Coordination of this scheme with a block-based AMR procedure allows for a more detailed representation of the embedded boundary and accurate resolution of flows having multiple scales. A block-based AMR level set method has been used to model evolving embedded boundaries. The computational cost of evolving the location of the combustion interface, tracing the location of the zero level set, and subsequently performing the mesh adjustment algorithm according to the new location of the embedded boundary is equivalent to the application of 1-2 steps of the explicit time-marching scheme.

Due to the resolution required within boundary layers, it is felt that the mesh adjustment scheme as described herein should be restricted to embedded boundary flows with low-to-moderate Reynolds numbers. However, this restriction could be removed by using a hybrid mesh approach where a body-fitted mesh is attached to the embedded boundary. The mesh adjustment algorithm is then performed at the interface between the mesh fitted to the embedded boundary and the domain mesh. This approach has previously been used with the Cartesian cut-cell method [13]. Not only will the use of a hybrid mesh approach allow the computation of high-Reynolds number flows but the oscillations that appear in the skin friction coefficient due to the non-smooth mesh, although greatly reduced by the current treatment, would then completely disappear. In addition, the block-based AMR scheme could be applied independently to the background and body-fitted meshes.

Future work will include the extension to hybrid meshes as well as three-dimensional fluid flows. Application of the scheme will include internal rocket motor flows [44] as well as non-equilibrium micron-scale flows, such as those encountered in the complex micron-sized conduits of micro-electromechanical systems (MEMS) and flows associated with chemical-vapour deposition (CVD) [32].

Acknowledgments

This research was supported by a Premier's Research Excellence Award from the Ontario Ministry of Energy, Science, and Technology and by the Natural Sciences and Engineering Research Council of Canada. Funding for the parallel computing facility used to perform the computations described herein was obtained from the Canadian Foundation for Innovation and Ontario Innovation Trust (CFI Project No. 2169). The authors are very grateful to these funding agencies for this support.

References

[1] M. J. Aftosmis, Solution Adaptive CArtesian Grid Methods For Aerodynamics Flows With Complex Geometries, von Karman Institute for Fluid Dynamics Lecture Series, Belgique, 1997.

[2] M. J. Aftosmis, M. J. Berger and J. E. Melton, Robust and efficient Cartesian mesh generation for component-base geometry, AIAA J., 36 (1998), 952-960.

[3] R. W. Anderson, N. S. Elliot and R. B. Pember, An arbitrary Lagrangian-Eulerian method with adaptive mesh refinement for the solution of the Euler equations, J. Comput. Phys., 199 (2004), 598-617.

[4] T. J. Barth and P. O. Fredrickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, AIAA Paper 1990-0013.

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

[6] S. A. Bayyuk, K. G. Powell and B. van Leer, A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry, AIAA Paper 1993-3391.

[7] J. A. Benek, J. Steger and F. Dougherty, A flexible grid embedding technique with applications to the Euler equations, AIAA Paper 1983-1944.

[8] M. J. Berger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys., 53 (1984), 484-512.

[9] M. J. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys., 82 (1989), 67-84.

[10] J. C. Chassaing, G. A. Gerolymos and I. Vallet, Reynolds-stress model dual-time-stepping computation of unsteady three-dimensional flows, AIAA J., 41 (2003), 1882-1894.

[11] W. J. Coirier and K. G. Powell, An accuracy assessment of Cartesian-mesh approaches for the Euler equations, J. Comput. Phys., 117 (1995), 121-131.

[12] W. J. Coirier and K. G. Powell, Solution-adaptive Cartesian cell approach for viscous and inviscid flows, AIAA J., 34 (1996), 938-945.

[13] M. Delanaye, M. J. Aftosmis, M. J. Berger, Y. Liu and T. H. Pulliam, Automatic hybrid-Cartesian grid generation for high-Reynolds number flows around complex geometries, AIAA Paper 1999-0777.

[14] P. De Palma, M. D. de Tullio, G. Pascazio and M. Napolitano, An immersed-boundary method for 3D compressible viscous flows, AIAA Paper 2005-4715.

[15] D. De Zeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, J. Comput. Phys., 104 (1993), 56-68.

[16] D. P. Dobkin, S. V. F. Levy, W. P. Thurston and A. R. Wilks, Contour tracing by piecewise linear approximations, ACM T. Graphic., 9 (1990), 389-423.

[17] L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben, K. J. Badcock and B. E. Richards, Solution of the unsteady Euler equations using an implicit dual-time method, AIAA J., 36 (1998), 1417-1424.

[18] D. Enright, R. Fedkiw, J. Ferziger and I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys., 183 (2002), 83–116.

[19] C. Farhat, P. Geuzaine and C. Grandmont, The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids, J. Comput. Phys., 174 (2001), 669-694.

[20] P. Geuzaine, C. Grandmont and C. Farhat, Design and analysis of ALE schemes with prov-

able second-order time-accuracy for inviscid and viscous flow simulations, J. Comput. Phys., 191 (2003), 206-227.
[21] S. K. Godunov, Finite-difference method for numerical computations of discontinuous solutions of the equations of fluid dynamics, Mat. Sb., 47 (1959), 271-306.
[22] J. J. Gottlieb and C. P. T. Groth, Collection of boundary conditions for one- and some multi-dimensional unsteady flows of polytropic gases. Can. Aeronaut. Space J., 45 (1999), 161-182.
[23] C. P. T. Groth, D. L. De Zeeuw, K. G. Powell, T. I. Gombosi and Q. F. Stout, A parallel solution-adaptive scheme for ideal magnetohydrodynamics, AIAA Paper 1999-3273.
[24] C. Hirsch, Numerical Computation of Internal and External Flows, Volume 2, Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, Toronto, 1990.
[25] C. W. Hirt, A. A. Amsden and J. L. Cook, An Arbitrary Lagrangian-Eulerian computing method for all flow speeds, J. Comput. Phys., 14 (1974), 227-253.
[26] D. G. Holmes and S. D. Connell, Solution of the 2D Navier-Stokes equations on unstructured adaptive grids, AIAA Paper 1989-1932.
[27] J. D. Hunt, An adaptive 3D Cartesian approach for the parallel computation of inviscid flow about static and dynamic configurations, Ph.D. Thesis, University of Michigan, 2005.
[28] K. K. Kuo and M. Summerfield, Fundamentals of Solid Propellant Combustion, Progress in Astronautics and Aeronautics V. 90, American Institute of Aeronautics and Astronautics, 1986.
[29] G. Iaccarino and R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech. Rev., 56 (2003), 331-347.
[30] R. H. Landon, Compendium of unsteady aerodynamic measurements, NATO AGARD Advisory Report 702, 1982.
[31] C. B. Laney, Computational Gasdynamics, Cambridge University Press, Cambridge, 1998.
[32] J. G. M^{c}Donald, J. S. Sachdev and C. P. T. Groth, Gaussian moment closure for the modelling of continuum and micron-scale flows with moving boundaries, in: Proceedings of the 4th International Conference on Computational Fluid Dynamics, Ghent, Belgium, July 10-14, 2006.
[33] R. B. Milne, An Adaptive Level Set Method, PhD Thesis, University of California-Berkeley, 1995.
[34] S. M. Murman, M. J. Aftosmis and M. J. Berger, Implicit approaches for moving boundaries in a 3-D Cartesian method, AIAA Paper 2003-1119.
[35] R. W. Noack and J. P. Slotnick, A summary of the 2004 overset symposium on composite grids and solution technology, AIAA Paper 2005-0921.
[36] R. W. Noack, DiRTlib: A library to add overset capability to your flow solver, AIAA Paper 2006-5116.
[37] S. Osher and J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys., 79 (1988), 12-49.
[38] S. Osher and R. P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys., 169 (2001), 463-502.
[39] H. Paillère, K. G. Powell and D. L. De Zeeuw, A wave-model-based refinement criterion for adaptive-grid computation of compressible flows, AIAA Paper 1992-0322.
[40] C. S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys., 25 (1977), 220-252.
[41] C. S. Peskin, The immersed boundary method, Acta Numer., (2002), 1-39.
[42] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys., 43 (1981), 357-372.

[43] J. S. Sachdev, C. P. T. Groth and 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 (2005), 157-175.
[44] J. S. Sachdev, C. P. T. Groth and J. J. Gottlieb, Parallel AMR scheme for turbulent multi-phase rocket motor core flows, AIAA Paper 2005-5035.
[45] J. A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge Monographs on Applied and Computational Mathematics, 2nd ed., Cambridge University Press, 1999.
[46] M. Sussman, E. Fatemi, P. Smereka and S. Osher, An improved level set method for incompressible two-phase flow, Comput. Fluids, 27 (1998), 663-680.
[47] P. D. Thomas and C. K. Lombard, Geometric conservation laws and its application to flow computations on moving grids, AIAA J., 17 (1979), 1030-1037.
[48] E. F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag, Berlin, 1999.
[49] Y.-H. Tseng and J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys., 192 (2003), 593-623.
[50] P. Vachal, R. V. Garimella and M. J. Sashkov, Untangling of 2D meshes in ALE simulations, J. Comput. Phys., 196 (2004), 627-644.
[51] B. van Leer, C. H. Tai and K. G. Powell, Design of optimally-smoothing multi-stage schemes for the Euler equations, AIAA Paper 1989-1933-CP.
[52] V. Venkatakrishnan, On the accuracy of limiters and convergence to steady state solutions, AIAA Paper 1993-0880.
[53] K. Weiler and P. Atherton, Hidden surface removal using polygon area sorting, in: Proceedings of SIGGRAPH '77, July 1977, pp. 214-222.
[54] P. J. Zwart, G. D. Raithby and M. J. Raw, The integrated space-time finite volume method and its application to moving boundary problems, J. Comput. Phys., 154 (1999), 497-519.