AIAA JOURNAL
Vol. 43, No. 9, September 2005

Parallel Unstructured Mesh Adaptation Method for Moving Body Applications

Peter A. Cavallo,* Neeraj Sinha,† and Gregory M. Feldman‡
Combustion Research and Flow Technology, Inc., Pipersville, Pennsylvania 18947

Adaptive, unstructured grid methods, in which the mesh is allowed to deform, and grid quality is subsequently restored through localized coarsening and refinement, offer the potential of a more rapid, straightforward approach to generalized moving body problems. Automation of this mesh movement and quality correction strategy requires close coupling with the flow solution process. With parallel simulations now common, parallel coarsening and refinement methods for moving meshes are needed. In this work, parallel mesh adaptation strategies are developed to treat deforming, decomposed domains. Distortion of the moving mesh is assessed using a deformation matrix analysis. A two-pass approach is implemented in which cell migration shifts the interprocessor boundary, thereby accommodating coarsening and refinement of the interprocessor faces. The adapted grids are rebalanced among the processors using available techniques. Representative cases are presented to demonstrate the parallel approach and maintenance of mesh quality for practical separation events.

Nomenclature

$[A]$	=	transformation matrix
CF	=	coarsening factor
e	=	error estimate
$[P]$	=	dilatation matrix
RF	=	refinement factor
$[T]$	=	tetrahedron edge matrix in initial state
$[T']$	=	tetrahedron edge matrix in deformed state
t	=	time
U	=	solution variable
U^*	=	reconstructed solution variable
$[U]$	=	rotation matrix
u_i	=	nodal displacements
x, y, z	=	Cartesian coordinates
$\Delta x, \Delta y, \Delta z$	=	distance between nodes of the mesh
δ_{ij}	=	Kronecker delta
ε_{ij}	=	strain rate tensor
$\kappa_\infty(A)$	=	spectral norm of transformation matrix
λ	=	Lamé constant
μ	=	shear modulus
ρ	=	local point spacing
σ_{ij}	=	stress tensor
$\sigma_{\max}$	=	maximum eigenvalue of dilatation matrix
$\sigma_{\min}$	=	minimum eigenvalue of dilatation matrix
τ	=	mesh deformation measure

Introduction

THREE-dimensional numerical simulations on parallel computing platforms have become commonplace in recent years. A number of structured and unstructured grid solvers for fluid dynamics, finite element analysis, and electromagnetics have been extended to operate in parallel environments. Not surprisingly, parallel adaptive mesh refinement methods have also received a fair amount of attention.[1–8] These include treatments for embedded quadrilateral[1] and Cartesian meshes,[2] r-refinement of block structured grids,[3] cell subdivision of unstructured triangular[4] and tetrahedral meshes,[5,6] octree-based tetrahedral grids,[7] and even unstructured hexahedral grids.[8]

In the parallel mesh adaptation research cited, the focus was primarily on mesh refinement for problems with fixed boundaries, and the subsequent load rebalancing. In most cases, the mesh refinement methods employ cell subdivision procedures, and a parallel implementation of these techniques is readily accomplished by exchanging the cell subdivision patterns at interprocessor boundaries, thus ensuring a conforming mesh. Coarsening the original mesh was not a concern; however, mesh coarsening could be accomplished by restoring parent cells after a previous cell subdivision.[6] Each of these methods has proven effective in solving steady state and transient problems for geometries with fixed boundaries. Moving mesh problems, however, require additional considerations not addressed by prior research in parallel adaptation. Most significant among these is the need to coarsen the original grid to accommodate boundary motion.

Moving boundaries appear in numerous problems of interest. These include store separation from aircraft, piston and valve motion in internal combustion engines, aeroelastic deformation of wings and control surfaces, and compliant arteries and biomedical applications, to name a few. Treatments for moving computational meshes have been developed by several groups using overset grid approaches,[9,10] sliding mesh techniques,[11,12] and deforming, unstructured grids.[13,14]

The adaptive unstructured grid approach presents a conceptually simple setup process. A single grid is generated, and the motion of boundary surfaces prescribed. No grid assembly or donor cell search process is needed, as with overset methods. Keys to this approach lie in robust mesh movement and modification schemes for ensuring a good quality mesh throughout the simulation, and automation of the adaptation process by operating directly on the decomposed domains used by the flow solver. Although mesh adaptation has been reported for general moving body flows,[15,16] the mesh modifications are typically performed on a single processor, after which the flowfield calculation is continued on multiple processors once the adapted grid is repartitioned.

This paper describes the extension of previous work[17–19] in moving mesh adaptation for unstructured grids to parallel computing environments, where the mesh is partitioned by domain decomposition. The focus of the adaptive treatments is to recover mesh quality during the course of large-scale motions. A move–coarsen–enrich strategy[19] is employed to treat the deforming mesh, in which the overall procedure is to 1) move the mesh boundaries and interior

Presented as Paper 2004-1057 at the 42nd Aerospace Sciences Meeting, Reno, NV, 5–8 January 2004; received 22 January 2004; revision received 23 November 2004; accepted for publication 19 March 2005.

*Senior Research Scientist; cavallo@craft-tech.com. Senior Member AIAA.
†Vice President and Technical Director. Associate Fellow AIAA.
‡Research Scientist.

nodes; 2) coarsen the grid to accommodate the motion after identifying the distorted cells; and 3) enrich the mesh to restore an appropriate distribution of cells.

A two-pass approach is presented for adapting the evolving grid in parallel, wherein the interprocessor boundaries are shifted using a cell migration technique. This shift allows interprocessor faces to be readily coarsened and refined as interior faces. The parallel mesh movement, deformation detection, and adaptation methods employed in the mesh adaptation package CRISP CFD are demonstrated on selected problems illustrating their utility in practical moving body applications.

Flow Solution Method

The unstructured flow solver used in this work is CRUNCH CFD.[20,21] CRUNCH CFD is a three-dimensional, edge-based, mixed-element unstructured Navier–Stokes solver for chemically reacting turbulent real gases and dynamic domains. The basic numerical framework of the CRUNCH CFD code is a finite-volume higher-order Roe/total variation diminishing (TVD) scheme in which the flow variables are defined at the vertices of the mesh. An edge-based data structure is employed wherein a polyhedral control volume is constructed from the union of all cells incident to a given node, and the control volume faces are associated with each edge. The inviscid flux calculation proceeds by looping over all edges in the mesh, and is grid-transparent, whereas a cell-based method is employed to compute the flowfield gradients at the control volume faces for evaluating the viscous fluxes.[20] Turbulence modeling is provided by a two-equation k–ε model.

For moving body problems, the grid motion in CRUNCH CFD is strongly coupled to the flow solution, as the additional fluxes generated by the moving control volume faces are taken into account.[22] For efficient computation of large three-dimensional problems, a parallel framework for distributed memory systems has been implemented along with an implicit sparse matrix solver, permitting large Courant–Friedrichs–Lewy (CFL) numbers. Additional details on the solver may be found in Refs. 20–22.

Grid Movement Procedure

The simulation of arbitrary dynamic multibody flows is accomplished by means of a generalized node movement scheme. Given a change in the boundary mesh, resulting from a rigid body motion, structural deformation, or design process, the interior nodes of the tetrahedral grid are redistributed. The method models the tetrahedral mesh as a deformable, elastic solid, subject to the equations of elasticity[22]:

$$\frac{\partial \sigma_{ij}}{\partial x_i} = 0 \tag{1}$$

In this model, the stress and strain tensors are related to the node displacements u_i:

$$\sigma_{ij} = \lambda \varepsilon_{kk} \delta_{ij} + 2\mu \varepsilon_{ij} \tag{2}$$

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3}$$

and λ and μ are functions of cell volume. This permits small cells to resist further deformation, whereas larger cells are more elastic. When these equations are solved in parallel on decomposed domains, the displacements of the interprocessor nodes are matched among all incident processors, and a subiteration is performed to ensure adequate propagation of the stress across interprocessor boundaries. Mesh movement with viscous, multielement grids composed of tetrahedra and near-wall prism cells is done by treating the prism cells as rigid, so that they move with the underlying solid boundary, and the tetrahedral region of the domain deforms to accommodate the motion.

This node movement scheme has proven to be much more robust than the commonly used spring analogy or velocity diffusion techniques, because representing the full stress tensor allows mesh deformations to be propagated farther into the field, particularly for shearing motions. It should be noted that researchers have recently remedied this deficiency by including torsional springs.[23,24] Regardless of the method used, a reliable grid movement scheme permits a given mesh to be useful for a greater period of time before corrective measures need to be taken, reducing the frequency with which mesh adaptation is required.

Mesh Deformation Analysis

Mesh movement alone cannot tolerate large-scale boundary displacements, particularly for bodies in close proximity. For large degrees of motion the unstructured mesh will eventually become so distorted that the resulting poor cell quality will affect the computed flowfield, or worse, form inverted cells that terminate the simulation. Although adaptive mesh techniques may be used to alleviate this distortion, a means of detecting and monitoring these mesh deformations is needed to inform the refinement and coarsening modules where and how to modify the grid.

A mesh deformation matrix concept previously introduced by Baker[17,18] is used to detect distorted regions of the mesh in need of corrective coarsening and refinement. The analysis examines the transformation of a tetrahedral cell from its initial state at time t_0 to its current shape at time t_1. This transformation matrix $[A]$ is formed from the edge matrix of the cell vertex coordinates at time t_0 and t_1:

$$[T] = \begin{bmatrix} \Delta x_{21} & \Delta x_{31} & \Delta x_{41} \\ \Delta y_{21} & \Delta y_{31} & \Delta y_{41} \\ \Delta z_{21} & \Delta z_{31} & \Delta z_{41} \end{bmatrix}_{t=t_0}$$

$$[T'] = \begin{bmatrix} \Delta x_{21} & \Delta x_{31} & \Delta x_{41} \\ \Delta y_{21} & \Delta y_{31} & \Delta y_{41} \\ \Delta z_{21} & \Delta z_{31} & \Delta z_{41} \end{bmatrix}_{t=t_1} \tag{4}$$

$$[T'] = [A][T] \tag{5}$$

$$[A] = [T'][T]^{-1} \tag{6}$$

The subscripts in Eq. (4) denote coordinate differences between the four node numbers defining the tetrahedral cell. Note that the deformation represented by the matrix $[A]$ is cumulative over the entire history between times t_0 and t_1. The deformation matrix $[A]$ is further decomposed into two matrices that represent dilatation and rigid body rotation:

$$[A] = [P][U] \tag{7}$$

If the cell rotates and/or translates without deforming, then $[P]$ is the identity matrix. Conversely, if the cell deforms without rotating, $[U]$ is the identity matrix. The eigenvalues of $[P]$ represent the dilatation of the tetrahedron in each of three principal directions and are equivalent to the singular values of the transformation matrix $[A]$. In the limit, eigenvalues of one correspond to an undeformed cell, whereas an eigenvalue of zero indicates a cell that has collapsed to zero volume.

The transformation matrix $[A]$ is invariant under translation of the cell, and any unitarily invariant norm will be invariant under cell rotation.[17] The condition number formed from any such norm is also invariant under scaling. If one chooses the spectral norm of the transformation matrix $[A]$, then the condition number is given by

$$\kappa_\infty(A) = \sigma_{\max}/\sigma_{\min} \tag{8}$$

The reciprocal of the condition number forms a deformation measure that indicates the extent of cell distortion from the reference state at t_0:

$$\tau = 1/\kappa_\infty(A) = \sigma_{\min}/\sigma_{\max} \tag{9}$$

This deformation measure takes on values between 0 and 1 for each tetrahedral cell. Mesh coarsening is then driven by this deformation measure in prescribing a larger point spacing to remove the poor-quality elements. Mesh refinement then restores an adequate point

spacing based on the boundary spacing or current flowfield features. Note that the reference state is reset each time mesh adaptation is performed. Thus the deformation is measured from the last time adaptive coarsening and refinement were invoked.

Mesh Modification Operations

Prescribed Point Spacing

Adaptive refinement and coarsening are triggered by prescribing what the point spacing should be, based on the mesh deformation measure and/or solution error estimates. An initial point spacing ρ_0 is defined for each vertex as the average edge length for all edges incident to the node. A larger grid spacing produces an iterative coarsening of the mesh. Using the mesh deformation measure defined in Eq. (9), the local grid spacing is modified to be

$$\rho = \rho_0/\tau \tag{10}$$

Note that the more deformed the cell, the larger the prescribed spacing, and hence an increased amount of coarsening will be performed. This improves the likelihood of the distorted cell being removed.

Conversely, the enrichment procedures are invoked by specifying a smaller spacing. After the coarsening phase, an appropriate gradation of cell size is restored by solving a Laplace equation for ρ, using the boundary mesh spacings as Dirichlet boundary conditions.[25] An approximate solution is obtained by summing the difference in the point spacing for each of N edges incident to the node by using a relaxation technique:

$$\rho^{n+1} = \rho^n + 0.5\frac{1}{N}\sum_{k=1}^{N}\left(\rho_k^n - \rho^n\right) \tag{11}$$

Note that interprocessor boundary nodes are iterated upon, and only physical boundary nodes are included as boundary conditions.

Prescribing new point spacings also drives solution-based coarsening and refinement. A variation of the solution error estimate developed in two dimensions by Ilinca et al.[26] has been implemented in three dimensions for multielement unstructured meshes. The method is based on forming a higher order approximation of the solution at each mesh point using a least-squares approach. The difference between the higher order reconstruction from incident nodes and the current solution forms the error measurement. If the current mesh is sufficiently fine to support the spatial variation in the solution, the estimated error will be low, allowing coarsening to take place. Conversely, a high degree of error indicates that additional refinement is needed.

At any location (x, y, z), the piecewise quadratic solution is formed at node j from the solution at grid point i by Taylor series expansion using all first- and second-order derivatives. The solution itself is assumed to be piecewise linear. The error at grid point i is the sum of the difference between the actual solution at node j, U_j, and the reconstructed solution at j, U_j^*, for each of N edges incident to node i:

$$e_i = \sum_{j=1}^{N}\left|U_j^* - U_j\right| \tag{12}$$

$$e_i = \sum_{j=1}^{N}$$

$$\left|U_i - U_j + \Delta x\frac{\partial U}{\partial x} + \Delta y\frac{\partial U}{\partial y} + \Delta z\frac{\partial U}{\partial z} + \Delta x^2\frac{\partial^2 U}{\partial x^2} + \Delta y^2\frac{\partial^2 U}{\partial y^2} + \Delta z^2\frac{\partial^2 U}{\partial z^2} + \Delta x\Delta y\frac{\partial^2 U}{\partial x\partial y} + \Delta y\Delta z\frac{\partial^2 U}{\partial y\partial z} + \Delta x\Delta z\frac{\partial^2 U}{\partial x\partial z}\right| \tag{13}$$

The error is scattered to each node by looping over all edges in the mesh and then normalized to take on values between 0 and 1. In regions where the error is low, the new, local point spacing is increased as

$$\rho = \mathrm{CF}\rho_0 \tag{14}$$

whereas in high-error regions, smaller spacings are defined:

$$\rho = \rho_0/[1 + (\mathrm{RF} - 1)e] \tag{15}$$

In Eqs. (14) and (15), CF and RF are user-defined coarsening and refinement factors, respectively. These factors permit varying degrees of coarsening and/or refinement within a given adaptation pass.

Computing the various derivatives in Eq. (13) requires solution of a least-squares problem. Inversion of the system requires a stencil of at least nine points. Currently, all of the edges incident to a node are employed, as well as the cell and face centers for the elements incident to the node.

Mesh Coarsening

Mesh coarsening is accomplished by collapsing edges. Given an edge selected for removal, the ring of tetrahedra incident to the edge is identified. These cells are removed from the mesh, the two nodes of the edge are merged into a single vertex, and the cells adjacent to the ring are redefined. This procedure is applicable to boundary edges as well as interior edges.

The list of edges to be removed is formed by comparing the length of each edge to the prescribed spacing at the two endpoints, as defined by Eq. (11) or (14). If the edge length is less than half the prescribed spacing, it is considered for collapse. The coarsening process is continued until no edges are selected for removal, indicating that the mesh is consistent with the prescribed spacing.

Mesh Refinement

The tetrahedral region of the grid is refined using a constrained Delaunay cavity reconstruction. If the circumradius of a tetrahedron is too large compared to the prescribed point spacing, a new point is inserted at the circumcenter of the cell. Tetrahedra whose circumspheres contain the new point are then deleted, forming a Delaunay cavity. New cells are formed by connecting the faces of the cavity with the new point.[27] This process is repeated until no cell violates the spacing consistency check.

Refinement of the mesh boundary is performed by bisecting edges of the boundary triangles. The list of faces to refine is formed by comparing the boundary point spacing to the prescribed spacing. A face is added to the list if the boundary spacing for all three nodes is 1.5 times greater than the spacing specified. These faces are sorted by the ratio of circumradius to inradius, and a list of edges to split is created. The boundary edges are bisected in sequence, subdividing the incident ring of tetrahedra along the way.[28] The boundary point spacing is recomputed, and the process repeated until no edges are split.

The prismatic region of a viscous, multielement grid may be refined through cell subdivision procedures. As the boundary of the tetrahedral region is refined, the adjacent prism layers are also modified. This is accomplished by splitting edges at the tet/prism interface and propagating this subdivision down to the wall through all of the layers. In addition, a procedure is in place to refine entire layers of prisms if improved boundary layer resolution is desired.[28]

Optimization and Smoothing

After each alteration of the mesh, the grid is optimized to remove any slivers. These are nearly flat tetrahedra that exhibit large dihedral angles. These problematic cells are removed from the grid through edge- and face-swapping procedures.

The tetrahedra are first sorted by circumradius to inradius ratio. For each cell, the largest dihedral angle between adjacent faces is computed. If this angle exceeds 120 deg, various procedures are applied in an attempt to remove or replace the poorly shaped cell. The method described in Ref. 27 involves removing an edge and the incident N cells and replacing them with a set of $N - 2$ new interior faces and $2N - 4$ new cells. Alternately, a face may be removed between two cells and an edge inserted joining the two nodes opposite the common face, creating three tetrahedra incident to the new edge. Mesh optimization is repeated until no more cells are replaced.

An analogous optimization is applied to the boundary mesh by swapping diagonals. A list of candidate boundary edges is made

by sorting the boundary faces according to their maximum interior angles. For each edge where this angle exceeds 120 deg, the adjacent ring of N cells is replaced, and the two incident boundary faces are redefined.

Parallel Implementation

Parallel Strategy: Two-Pass Approach

The simultaneous alteration of the decomposed domains of an unstructured mesh presents a number of challenges. In parallel, each processor operates on its own partition, concurrent with and independent of the others. Previous work in parallel mesh refinement[1–8] demonstrated methods in which adaptation was performed on each processor, and patterns for cell subdivision were exchanged across interprocessor boundaries, ensuring a consistent mesh. Coarsening the interprocessor boundary was not a concern, nor was the possible motion of the mesh boundaries.

With moving meshes, however, there is a need to coarsen the original grid to accommodate the motion of the boundaries. Furthermore, the boundary mesh refinement and coarsening operations used in the current work do not employ pattern formation procedures, and the synchronization of these sequential operations quickly becomes complicated for arbitrary partitions where multiple processors may share a common edge. Therefore the first issue that arises in the parallel adaptation of moving body problems is how to treat the interprocessor boundaries. Rather than modifying these faces, the interprocessor boundaries are shifted by passing cells across processors. The interprocessor faces and adjacent cells then become interior faces and interior cells, which may be readily modified through a second adaptation pass. The second pass does not incur a significant expense because only the former interprocessor boundary region needs to be coarsened, refined, or smoothed. The remainder of the mesh is already consistent with the prescribed point spacing.

Figure 1 illustrates the two-pass approach to solution-based coarsening and refinement of supersonic flow entering a duct. The original mesh partitions, shown in Fig. 1a, are independently coarsened and refined to produce the adapted mesh of Fig. 1b. Note that the interprocessor boundaries are not modified, which leaves a region of the mesh that still requires adaptation. Several layers of cells are migrated from the right processor to the left, as seen in Fig. 1c. The interprocessor boundary is now to the right of its original location. A second coarsening and refinement pass treats the former interprocessor faces and adjacent cells, producing the final adapted grid of Fig. 1d.

Cell Migration Method

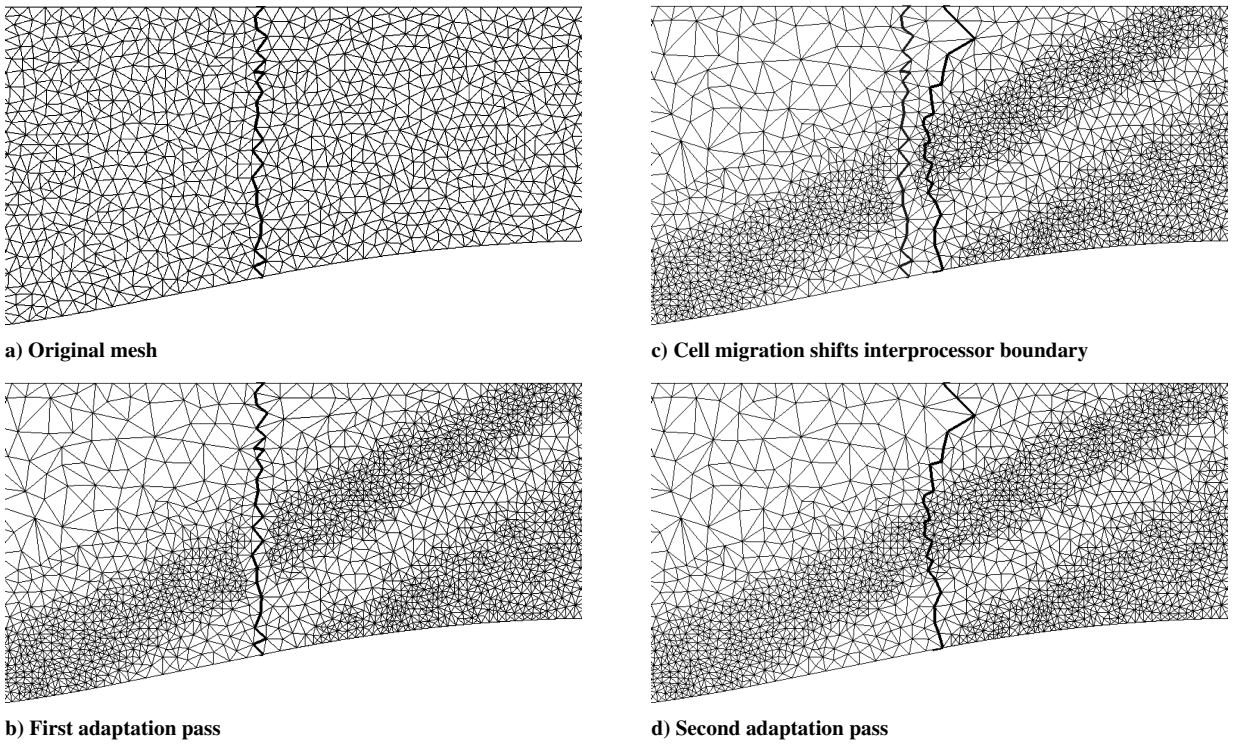
After the first coarsening and refinement pass, cell migration shifts the interprocessor boundaries. A migration is performed for each pair of neighboring domains. After all migrations are complete, the second adaptation pass begins.

In a given communication cycle, each processor forms a set of cells, faces, and vertices to exchange, starting at the interprocessor boundary and agglomerating the cells in successive layers. Two to five layers of cells are sufficient. These cell, face, and node sets are exchanged between adjacent processors. The decision to accept the set received or remove the set sent is based on the current load distribution, in an attempt to maintain load balance. If the grid on the current CPU is larger than that of the adjacent CPU, this domain will give up cells to its neighbor. The cell set received is discarded and the set sent is removed from the grid. Conversely, if the local grid is smaller than the adjacent grid, this domain will gain cells from its neighbor. The set sent is ignored and the set of cells, faces, and nodes received is appended to the grid. After each migration, the interprocessor communication schedule and list of common vertices must be updated.

Implications of Cell Migration

A consequence of the cell migration approach is that the shape and extent of the decomposed domains change. The cell migration process may introduce new pairs of adjacent domains that did not initially communicate. Similarly, pairs of processors that once shared common nodes, edges, and faces may become disconnected as a result of cell migration, because the decomposed domains are dynamically changing shape.

Updating the interprocessor communication schedule proceeds in two stages. First, the current communication lists are updated for each pair of adjacent domains. If no common nodes are found between two domains, the communication is removed from the cycle. The second stage involves checking for any new communication pairs introduced as a result of migration.



a) Original mesh

c) Cell migration shifts interprocessor boundary

b) First adaptation pass

d) Second adaptation pass

Fig. 1 Two-pass approach for parallel coarsening and refinement.

An efficient octree search procedure[29] is used to facilitate the communication update. First, the list of all interprocessor boundary nodes is stored in a message to be sent to the adjacent processor. Each pair of domains exchange node lists, and the list received from the neighboring processor is placed into the octree. For each point in the list sent, the octree is searched to find a matching node in the adjacent domain. This process reestablishes the common node pairs for exchanging data in parallel once we return to the flow solver.

The search for new communication pairs involves additional steps. For each domain, the (x, y, z) extrema are stored in a message to be sent to other processors. A list is formed of processors the local domain does not currently communicate with. For each domain in this list, the (x, y, z) extrema are exchanged, and a check is performed as to whether the extrema of the two domains overlap. The subset of processors that pass this test are then put through the interprocessor node matching procedure to determine whether a new communication pair is present.

In addition, one can no longer refer to the original decomposed grid to obtain data for solution transfer or for establishing point spacing after the coarsening phase. This issue is remedied by recomposing the global grid arrays at the start of the mesh adaptation process, so that the list of global vertex coordinates, solution vectors, and computed point spacings may be readily available to all processors. This allows global grid data to be readily retrieved as needed. Another motivation for reestablishing the global grid is to form the cell adjacency structure required by the load-balancing method employed in this work. The method operates on the global cell numbers to diffuse cells across processors. After all mesh modifications are complete, the individual domains for each processor are concatenated to determine global cell and node numbering.

Load Rebalancing

An adapted grid is inherently unbalanced, with mesh refinement taking place on certain partitions and coarsening taking place on others. Reestablishing load balance is essential for transient simulations, particularly after several adaptations. A number of rebalancing approaches have been explored, and the review of Hendrickson and Devine[30] provides an overview of the various issues and methods.

Load rebalancing is accomplished by using the *ParMETIS* package[31] developed at the University of Minnesota. The global cell numbers and cell adjacencies are first formed for each of the existing partitions. At interprocessor boundaries, the adjacent global cell number on the neighboring processor must also be determined. The adaptive repartitioning routines in *ParMETIS* redistribute the existing partitions of the adapted grid using a diffusion concept.[31] New processor assignments for each cell in the global grid are obtained. From these new assignments, the balanced partitions may be reformed.

Conceptually, it is also possible to rebalance the adapted grid using the cell migration technique. In practice the migration must be performed in several sweeps, on the order of 5–10 passes, to obtain a balanced decomposition. The approach is therefore not as cost-effective as the *ParMETIS* package.

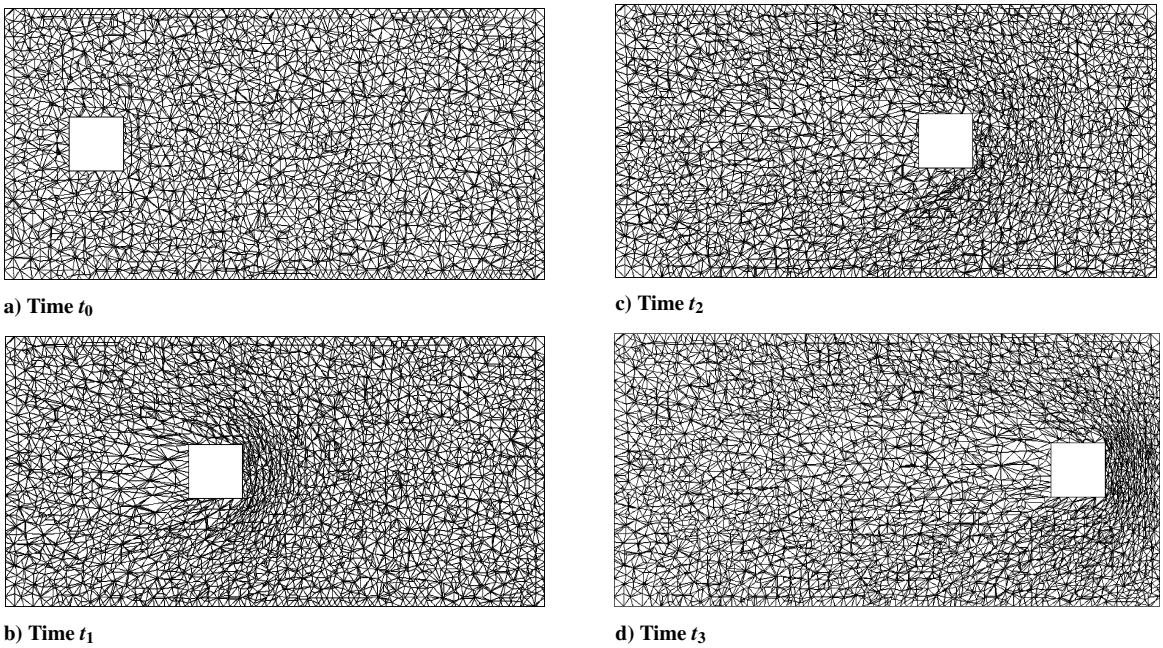
Sample Applications

Cube Translation in a Duct

The first case considered illustrates the parallel adaptation method on a simple test problem. No flow solution is computed. A cube one unit on all sides translates the length of a $5 \times 5 \times 10$ duct at constant velocity. The initial mesh is uniformly spaced, composed of approximately 205,000 cells, and is partitioned onto four processors.

Figure 2 depicts the evolution of the partitioned grid as the cube translates down the length of the domain. A slice of the mesh is shown at four instants in time. Mesh adaptation is performed a total of five times, as the deformation measure approaches a value of 0.1. In the sequence of images shown in Fig. 2, mesh adaptation is seen to limit the amount of distortion experienced by the mesh. Furthermore, it is evident that the coarsening and refinement operators are recovering the grid quality and grid spacing throughout the extent of the motion. Computations were performed on a Linux personal computer cluster using 1-GHz Pentium III processors. The combined mesh adaptation and mesh movement required approximately 900 s. The mesh adaptation and rebalancing represented 20% of this cost. Note that if the flowfield were also computed the fraction of CPU time associated with adaptation would be even lower.

A quantitative assessment of cell quality is presented in Fig. 3. As the mesh deforms, the maximum and mean cell aspect ratio increase, with the maximum aspect ratio rising several orders of magnitude with increased motion and distortion. Each mesh adaptation performed restores the aspect ratio statistics to acceptable levels. Aspect ratio in this analysis is defined as the ratio of cell circumradius to cell inradius, divided by 3, the circumradius to inradius ratio for an ideal, equilateral tetrahedron. The history of the minimum mesh deformation measure for the entire grid is shown in Fig. 4. With each adaptation, the reference state for the deformation analysis is reset. As the mesh movement proceeds, the deformation measure τ decreases accordingly.



a) Time t_0 **c) Time t_2**

b) Time t_1 **d) Time t_3**

Fig. 2 Cube translation in a duct, slice through mesh at selected intervals.

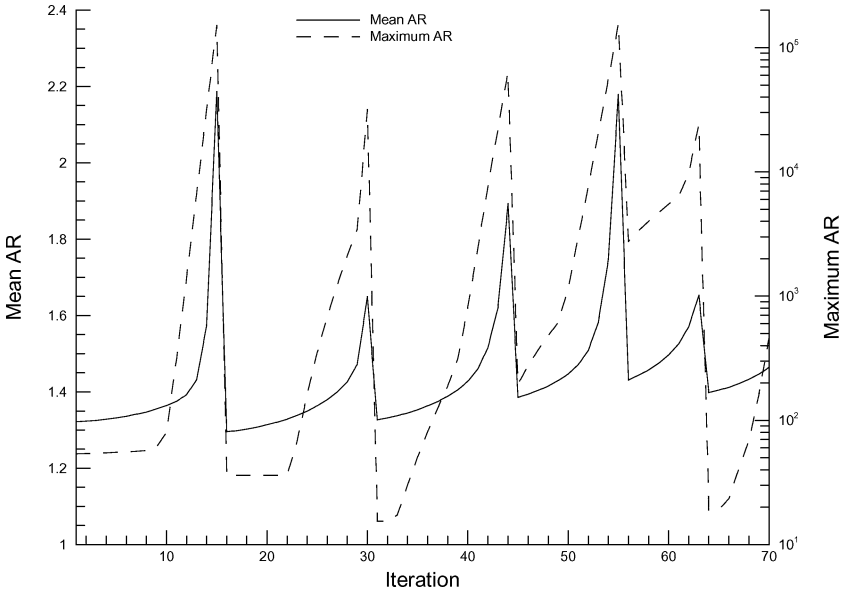


Fig. 3 Cell aspect ratio history for translating cube case.

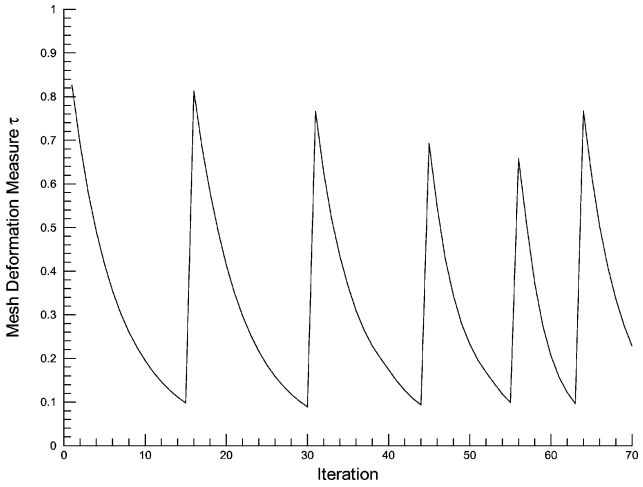


Fig. 4 Mesh deformation measure during cube translation.

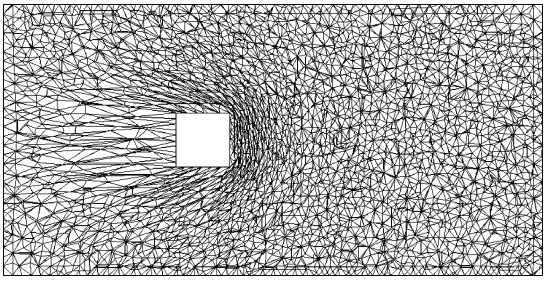


Fig. 5 Failure of mesh movement without adaptation.

The amount of motion present in this test problem cannot be handled by the mesh movement procedure alone, as seen in Fig. 5. At the instant shown, the mesh movement failed, and inverted cells resulted. Localized remeshing or adaptation is required for large-scale motions. The ability to adaptively correct the mesh in parallel facilitates transient analyses in conjunction with a flow solver. The examples that follow expand the concepts presented to moving body flowfields.

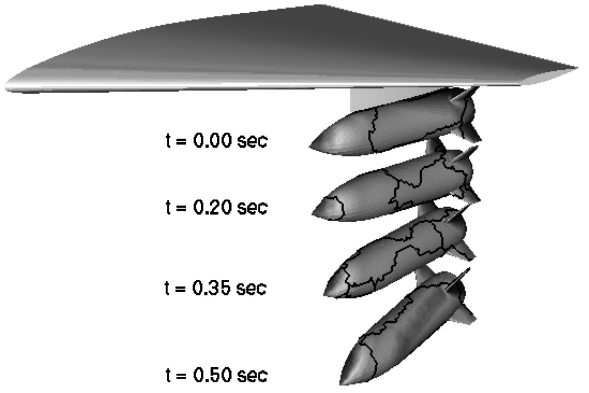


Fig. 6 Store position, orientation, and surface pressures at selected points in trajectory.

Generic Transonic Store Release

The first practical application considered is the separation of a finned store from a wing/pylon configuration at Mach 1.2. Inviscid flow is assumed for this tetrahedral grid. A constant ejection force is applied over the first 0.1 s of the simulated separation. After the initial ejection stroke, the motion of the store is provided by general six-degree-of-freedom (6-DOF) equations of motion using the current integrated surface pressure distribution. Gravitational acceleration is also included. The focus of the study is to assess proper handling of grid deformation.

Figure 6 provides an overview of the simulation. A total of 10 adaptations were performed at regular intervals. The unstructured grid is composed of approximately 2.7 million cells and is decomposed on 16 processors. In this image, the store is shaded according to the current pressure distribution at each of the four instants shown, and the lines on the store surface indicate the changing interprocessor boundaries resulting from cell migration and load rebalancing. As it translates, the store yaws nose away from the symmetry plane and pitches nose down. The surface pressure distribution reflects the changing local angle of attack and sideslip angle of the store.

Figure 7 shows the variation of the minimum mesh deformation measure for the entire grid during the simulation. As the distance between the store and pylon surfaces increases, the mesh distortion becomes less severe. With each successive adaptation, the deformation measure reduces to a minimum value greater than the previous minimum. This indicates that mesh movement may

likely be applied for a longer period of time before adaptation is warranted. However such strategies and tradeoffs are yet to be investigated.

The evolution of the unstructured mesh as the store falls away is depicted in Fig. 8. Interprocessor boundaries are illustrated as well. Through the adaptive coarsening and refinement procedures, overall mesh quality is maintained between the pylon and store surfaces, and an appropriate cell distribution is provided as the distance between the store and pylon increases. Although Fig. 8 illustrates only a slice through the mesh, the migration and rebalancing of the interprocessor boundaries is evident. This simulation was completed in 20 h of CPU time on the Pentium III cluster. For this case the overhead associated with mesh adaptation and rebalancing was 5% of the overall simulation time.

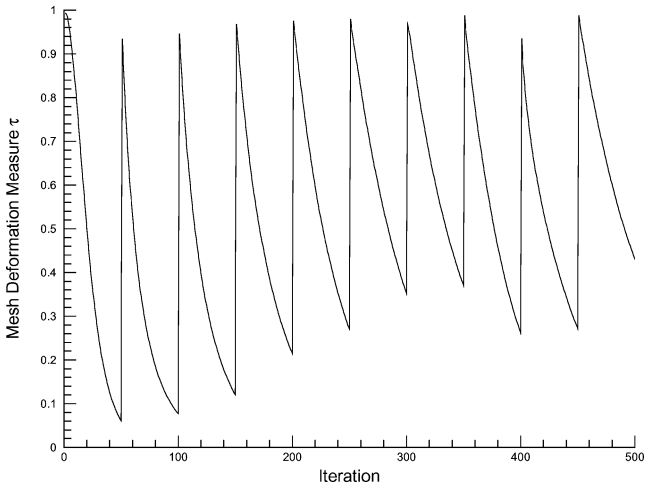


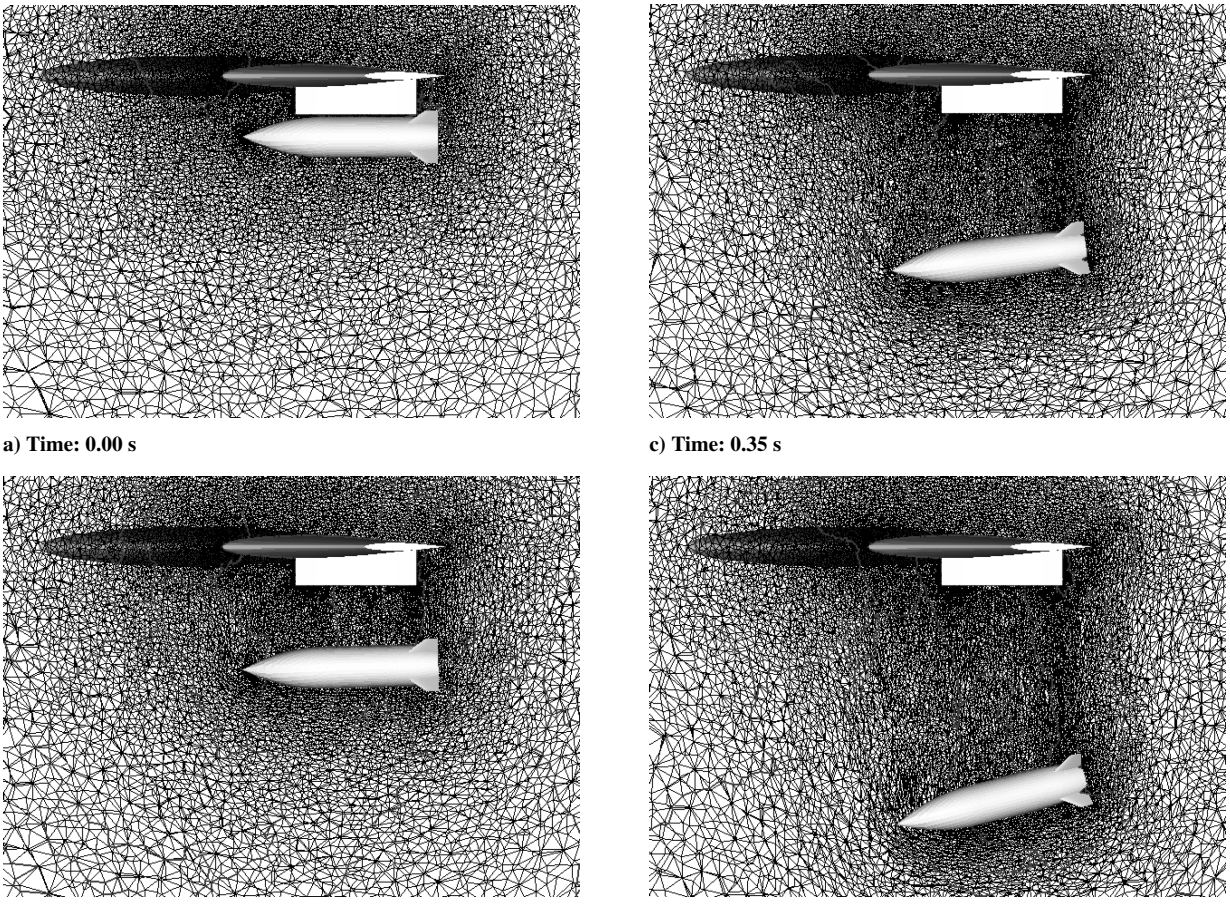
Fig. 7 Mesh deformation measure during store release.

Rocket Stage Separation

This example examines lower stage separation from a rocket due to upper stage motor ignition. Motion of the lower stage is coupled to the instantaneous pressure and shear stress distribution. A 5-deg sector is modeled with the lower stage translating axially. The viscous tetrahedral/prismatic grid is decomposed on 16 processors. Initially, the rocket is in steady flight at Mach 7.5. In addition to the adaptive treatment for handling the mesh motion, solution-based adaptation is also applied. The simulation required 150 h, with mesh adaptation reflecting 4% of the cost.

Figure 9 illustrates contours of exhaust mixture fraction, the local fraction of mass originating in the upper stage exhaust, on one of the symmetry planes at 10-ms intervals. A mirror image of the current decomposed mesh is also provided. Initially, there is no plume and the mesh is relatively coarse away from the rocket. As the plume expands and produces a large forward separation, additional refinement takes place and the interprocessor boundaries are shifted to reestablish load balance. One aspect of the problem that remains an issue is how frequently to perform mesh adaptation, to keep up with the expanding front rather than "chasing" the evolving flowfield. Combined solution adaptation and mesh deformation treatment is a rich area for future work.

A second major challenge is handling the rapid motion of the lower stage. Figure 10 shows a close-up of the interstage connector region during the first 30 ms of the separation event. The large range of motion creates tremendous shear in the mesh, and over 40 adaptations were performed to restore grid quality during this time. The need for corrective adaptation becomes more frequent as the lower stage accelerates; however, the frequency with which the mesh must be modified is very much problem-dependent. Coupling



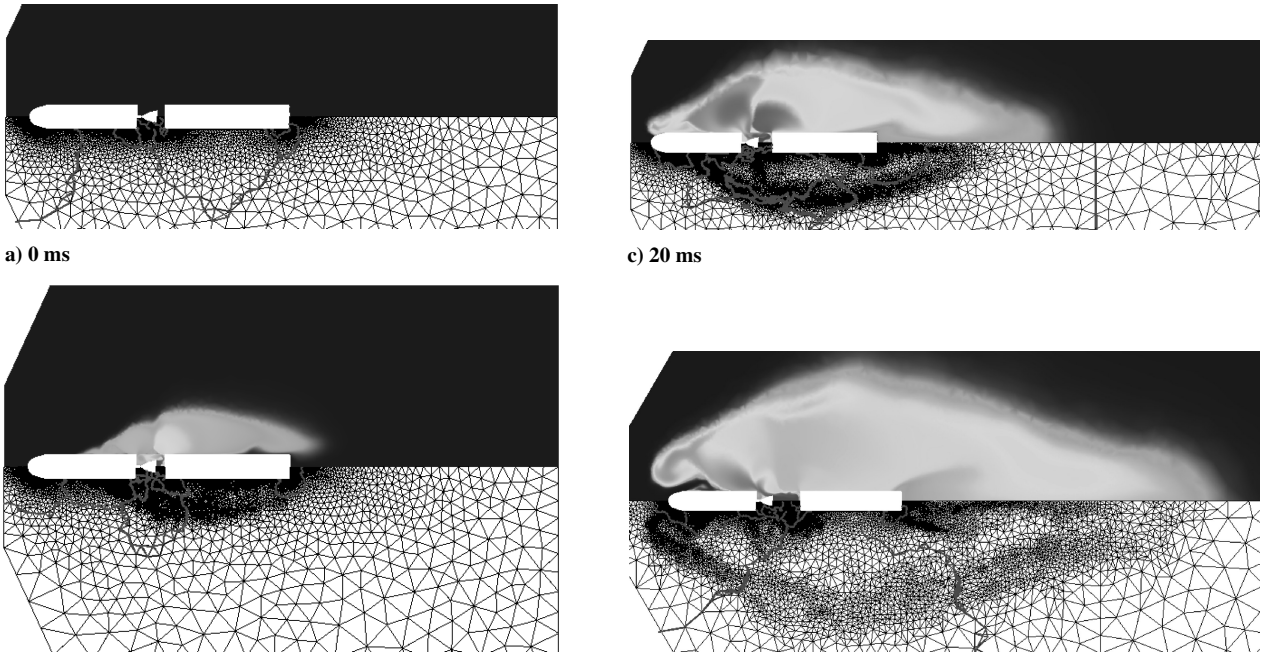
a) Time: 0.00 s

c) Time: 0.35 s

b) Time: 0.20 s

d) Time: 0.50 s

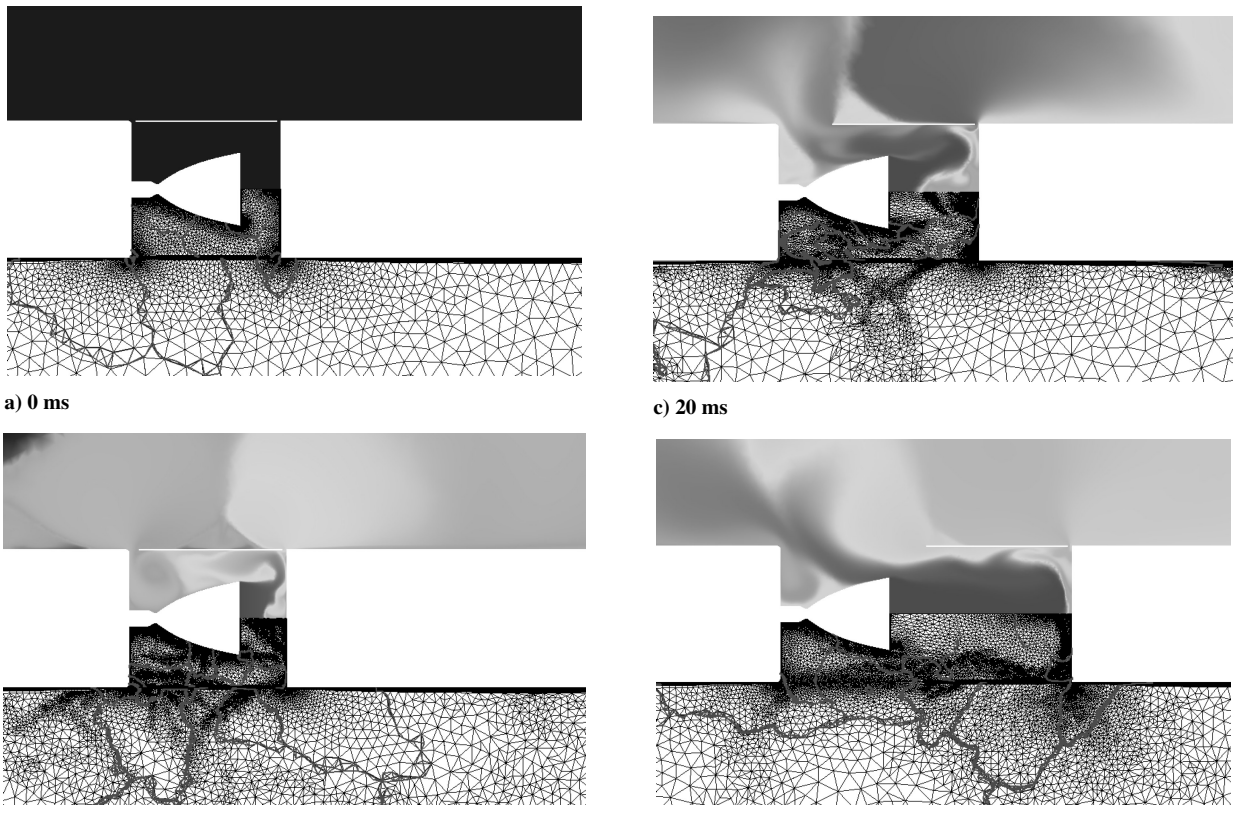
Fig. 8 Adapted mesh during store dispense.



a) 0 ms c) 20 ms

b) 10 ms d) 30 ms

Fig. 9 Overview of rocket stage separation.



a) 0 ms c) 20 ms

b) 10 ms d) 30 ms

Fig. 10 Close-up of interstage connector region during early separation event.

the mesh modifications with the flow solver will likely automate this process; however, this has yet to be achieved.

Conclusions

A novel parallel mesh adaptation strategy has been developed for deforming unstructured viscous grids. A two-pass approach is shown to effectively permit coarsening and refinement of the interprocessor boundary by shifting its location through cell migration. The parallel mesh movement and adaptation methods are seen to be effective for practical moving body problems such as store separation and rocket staging events. A good cell distribution is consistently restored as the unstructured mesh undergoes large-scale motion. Although there is no formal guarantee that the cell migration will permit coarsening and removal of highly distorted cells at the interprocessor boundary for arbitrary decompositions, in the

cases considered to date the approach has proven effective. The parallel mesh adaptation method presented also provides an efficient path for obtaining grid-converged, steady-state solutions; however, that is not the focus of the current effort and may be demonstrated in future work.

The frequency with which adaptation is required to accommodate arbitrary boundary motion is highly problem-dependent, and additional research is expected to yield more practical ways of applying the methods presented here. Acceptable thresholds on the minimum mesh deformation measure are of particular importance, to minimize user interaction. The current work, however, represents an important step toward a coupled flow solution and adaptation process for general moving body applications.

Acknowledgments

Funding for this research was provided by the Air Force Research Laboratory, Wright–Patterson Air Force Base, Ohio. Timothy Baker of Princeton University provided portions of the serial version of the mesh adaptation methods on which the parallel developments presented are based, as well as the routines for mesh deformation assessment. The authors thank him for many productive discussions throughout our collaboration.

References

[1]De Keyser, J., and Roose, D., "Run-Time Load Balancing Techniques for a Parallel Unstructured Multi-Grid Euler Solver with Adaptive Grid Refinement," *Parallel Computing*, Vol. 21, No. 2, 1995, pp. 179–198.

[2]MacNeice, P., Olson, K. M., Mobarry, C., de Fainchtein, R., and Packer, C., "PARAMESH: A Parallel Adaptive Mesh Refinement Community Toolkit," *Computer Physics Communications*, Vol. 126, No. 3, 2000, pp. 330–354.

[3]Soni, B. K., Koomullil, R., Thompson, D. S., and Thornburg, H., "Solution Adaptive Grid Strategies Based on Point Redistribution," *Computer Methods in Applied Mechanics and Engineering*, Vol. 189, No. 4, 2000, pp. 1183–1204.

[4]Jones, M. T., and Plassman, P. E., "Parallel Algorithms for Adaptive Mesh Refinement," *SIAM Journal on Scientific Computing*, Vol. 18, No. 3, 1997, pp. 686–708.

[5]Barry, W. J., Jones, M. T., and Plassman, P. E., "Parallel Adaptive Mesh Refinement Techniques for Plasticity Problems," *Advances in Engineering Software*, Vol. 29, Nos. 3–6, 1998, pp. 217–225.

[6]Oliker, L., Biswas, R., and Gabow, H. N., "Parallel Tetrahedral Mesh Adaptation with Dynamic Load Balancing," *Parallel Computing*, Vol. 26, No. 12, 2000, pp. 1583–1608.

[7]Flaherty, J. E., Loy, R. M., Shephard, M. S., Szymanski, B. K., Teresco, J. D., and Ziantz, L. H., "Adaptive Local Refinement with Octree Load Balancing for the Parallel Solution of Three-Dimensional Conservation Laws," *Journal of Parallel and Distributed Computing*, Vol. 47, No. 2, 1997, pp. 139–152.

[8]Niekamp, R., and Stein, E., "An Object-Oriented Approach for Parallel Two- and Three-Dimensional Adaptive Finite Element Computations," *Computers and Structures*, Vol. 80, No. 3–4, 2002, pp. 317–328.

[9]Lijewski, L. E., and Suhs, N. E., "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," *Journal of Aircraft*, Vol. 31, No. 4, 1994, pp. 886–891.

[10]Prewitt, N. C., Belk, D. M., and Shyy, W., "Parallel Computing of Overset Grids for Aerodynamic Problems with Moving Objects," *Progress in Aerospace Sciences*, Vol. 36, No. 2, 2000, pp. 117–172.

[11]Mathur, S. R., "Unsteady Flow Simulations Using Unstructured Sliding Meshes," AIAA Paper 94-2333, June 1994.

[12]Sinha, N., Cavallo, P. A., Lee, R. A., Hosangadi, A., Kenzakowski, D. C., Dash, S. M., Affes, H., and Chu, D., "Novel CFD Techniques for In-Cylinder Flows on Tetrahedral Grids," SAE Paper 980138, Feb. 1998.

[13]Löhner, R., Yang, C., and Baum, J. D., "Rigid and Flexible Store Separation Simulations Using Dynamic Adaptive Unstructured Grid Technologies," *Proceedings of the 1st AFOSR Conference on Dynamic Motion CFD*, edited by L. Sakell and D. Knight, Rutgers Univ., New Brunswick, NJ, 1996, pp. 3–29.

[14]Singh, K. P., Newman, J. C., and Baysal, O., "Dynamic Unstructured Method for Flows past Multiple Objects in Relative Motion," *AIAA Journal*, Vol. 33, No. 4, 1995, pp. 641–649.

[15]Johnson, A. A., and Tezduyar, T. E., "3D Simulation of Fluid–Particle Interactions with the Number of Particles Reaching 100," *Computer Methods in Applied Mechanics and Engineering*, Vol. 145, No. 3–4, 1997, pp. 301–321.

[16]Cavallo, P. A., and Dash, S. M., "Aerodynamics of Multi-Body Separation Using Adaptive Unstructured Grids," AIAA Paper 2000-4407, Aug. 2000.

[17]Baker, T. J., "Mesh Movement and Metamorphosis," *Engineering with Computers*, Vol. 18, No. 3, 2002, pp. 188–198.

[18]Baker, T. J., "Deformation and Quality Measures for Tetrahedral Meshes," *Proceedings of the European Congress on Computation Methods in Applied Sciences and Engineering (ECCOMAS 2000)*, edited by E. Onate, G. Burgeta, and B. Suarez, International Center for Numerical Methods in Engineering, Barcelona, 2000.

[19]Baker, T. J., and Cavallo, P. A., "Dynamic Adaptation for Deforming Tetrahedral Meshes," AIAA Paper 99-3253, June 1999.

[20]Hosangadi, A., Lee, R. A., Cavallo, P. A., Sinha, N., and York, B. J., "Hybrid, Viscous, Unstructured Mesh Solver for Propulsive Applications," AIAA Paper 98-3153, July 1998.

[21]Hosangadi, A., Lee, R. A., York, B. J., Sinha, N., and Dash, S. M., "Upwind Unstructured Scheme for Three-Dimensional Combusting Flows," *Journal of Propulsion and Power*, Vol. 12, No. 3, 1996, pp. 494–503.

[22]Cavallo, P. A., Hosangadi, A., Lee, R. A., and Dash, S. M., "Dynamic Unstructured Grid Methodology with Application to Aero/Propulsive Flowfields," AIAA Paper 97-2310, June 1997.

[23]Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional Springs for Three-Dimensional Dynamic Unstructured Fluid Meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 163, No. 1–4, 1998, pp. 231–245.

[24]Murayama, M., Nakahashi, K., and Matsushima, K., "Unstructured Dynamic Mesh for Large Movement and Deformation," AIAA Paper 2002-0122, Jan. 2002.

[25]Baker, T. J., "Mesh Deformation and Modification for Time Dependent Problems," *Proceedings of the ECCOMAS Computational Fluid Dynamics Conference (ECCOMAS CFD 2001)*, edited by N. P. Weatherill and K. Morgan, Inst. of Mathematics and Its Applications, Swansea, Wales, U.K., 2001.

[26]Ilinca, C., Zhang, X. D., Trépanier, J.-Y., and Camarero, R., "A Comparison of Three Error Estimation Techniques for Finite-Volume Solutions of Compressible Flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 189, 2000, pp. 1277–1294.

[27]Baker, T. J., and Vassberg, J. C., "Tetrahedral Mesh Generation and Optimization," *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Haeuser, and P. R. Eiseman, International Society of Grid Generation, Birmingham, 1998, pp. 337–349.

[28]Cavallo, P. A., and Baker, T. J., "Efficient Delaunay-Based Solution Adaptation for Three-Dimensional Unstructured Meshes," AIAA Paper 2000-0809, Jan. 2000.

[29]Baker, T. J., "Generation of Tetrahedral Meshes Around Complete Aircraft," *Numerical Grid Generation in Computational Fluid Mechanics'88*, edited by S. Sengupta, J. Haeuser, P. R. Eiseman, and J. F. Thompson, Pineridge Press, Swansea, Wales, U.K., 1988, pp. 675–685.

[30]Hendrickson, B., and Devine, K., "Dynamic Load Balancing in Computational Mechanics," *Computer Methods in Applied Mechanics and Engineering*, Vol. 184, No. 2–4, 2000, pp. 485–500.

[31]Schloegel, K., Karypis, G., and Kumar, V., "A Unified Algorithm for Load-Balancing Adaptive Scientific Simulations," Dept. of Computer Science and Engineering, TR 00-033, Univ. of Minnesota, Minneapolis, MN, May 2000.

G. Candler
Associate Editor