46th AIAA Aerospace Sciences Meeting and Exhibit
7 - 10 January 2008, Reno, Nevada

AIAA 2008-917

Parallel Anisotropic Tetrahedral Adaptation

Michael A. Park*
NASA Langley Research Center, Hampton, VA 23681
David L. Darmofal †
Massachusetts Insitute of Technology, Cambridge, MA 02139

An adaptive method that robustly produces high aspect ratio tetrahedra to a general 3D metric specification without introducing hybrid semi-structured regions is presented. The grid operators and higher-level logic is described with their respective domain-decomposed parallelizations. An tetrahedral adaptation scheme is demonstrated for 1000–1 anisotropy in a simple cube geometry. This form of adaptation is applicable to more complex domain boundaries via a cut-cell approach as demonstrated by a parallel 3D supersonic simulation of a complex fighter aircraft. To avoid the assumptions and approximations required to form a metric to specify adaptation, an approach is introduced that directly evaluates interpolation error. The grid is adapted to reduce and equidistribute this interpolation error calculation without the use of an intervening anisotropic metric. Direct interpolation error adaptation is illustrated for fifth-order elements in 1D and linear and quadratic tetrahedra in 3D.

Nomenclature

$\bar{u}$	Interpolated Solution
$\hat{u}$	Reconstructed Solution
$\mathbf{J}$	Linear Mapping
$\mathbf{M}$	Grid Metric
$\mathbf{R}$	Grid Metric Eigenvectors
$\tilde{u}$	Approximate Solution
$\mathcal{T}_i$	Grid Tetrahedra
$\mathcal{X}_i$	Grid Nodes
A	Integration Surface Area
$d\hat{x}, d\hat{y}, d\hat{z}$	Cartesian Unit Vectors
e	Interpolation Error
h	Length Specification
l	Vector Length Along an Edge
m	Norm Power
N	Number of Elements
p	Pressure
u	Exact Solution
V	Volume
w	Interpolation Error Weight

Symbols

η	Inverse of Tetrahedron Mean Ratio[1]
κ	Element
Λ	Grid Metric Eigenvalues
Ω	Domain

Superscripts

$'$	Transformed Space

I. Introduction

Computational Fluid Dynamics (CFD) has become a powerful tool for the analysis and design of aerospace vehicles. Obtaining a suitable grid remains one of the most difficult parts of the entire CFD simulation process. Grids must provide adequate resolution of flow features to control discretization error and remain small enough to permit reasonable computation times. The AIAA Drag Prediction Workshops[2–6] are well-documented examples of this difficulty even under the watchful eyes of experts. Mavriplis[7] shows that discretization error still dominates other error sources with the use of extremely fine grids that are

*Research Scientist, Computational AeroSciences Branch, NASA Langley Research Center, Hampton, VA 23681, AIAA Member.
†Professor, Department of Aeronautics and Astronautics, 77 Massachusetts Ave., 37-401, Cambridge, MA 02139, Senior AIAA Member.

computable with today's supercomputers. Chaffin and Pirzadeh[8] detail the effort required to manually specify grid resolution for accurate three dimensional (3D) high-lift computations. The high degree of manual intervention required in these cases prohibits the use of automated design tools. The manual interaction required to create a suitable grid is reduced with adaptive grid methods. Automated parameter studies[9] and design[10] are possible with an automated grid generation and adaptation process. Accurate final adapted solutions can be produced that are independent of coarse initial grids.[11–13]

Aerospace flow features include discontinuities, shear layers, and boundary layers that require a strongly anisotropic grid to resolve efficiently. Mavriplis,[14] Owen,[15] and Baker[16] provide surveys of grid generation and adaptation techniques. Hybrid methods[17] treat the highly anisotropic regions of the domain as separate semi-structured zones lacking the generality to treat changes in the topology of these anisotropic regions off body. Global[18] and local[19] regeneration can suffer from the same robustness issues as initial grid generation. Methods that apply isotropic techniques in a metric mapped space have been successful in two dimensions (2D).[18,20–23] The metric-based adaptation process has two principle components: determining an improved resolution request and creating an improved grid that satisfies that request. The improved resolution request is commonly based on local error estimates[16,18,24,25] and can include the effect of local errors on a global output quantity.[11,26,27] The use of anisotropic grid metrics has become a very standard way to specify a resolution request, but it does have limitations. Concessions must be made to fit output-based[11] and higher-order[13] methods into the metric framework.

In 3D, general anisotropic adaptation conforming to curved domain boundaries is a much more difficult task. Highly sophisticated methods that directly account for curved domain boundaries[28] still require local regeneration[29] when invalid tetrahedra can not be repaired. A different approach removes the constraint of generating a body-fitted grid and the complexities of adaptation directly on curved domain boundaries. This cut-cell approach allows the grid to arbitrarily intersect the boundary and the CFD code is modified to account for this arbitrarily intersection. Cartesian cut-cell methods have been very successful for Euler simulations[30–33] and some viscous simulations.[34,35] This cut cell method has also been applied to simplex meshes.[13,36,37] When the constraint of providing a body-fitted grid is removed, the grid generation and adaptation task becomes much simpler. Local incremental modification of an existing grid is then able to produce grids with high anisotropy without curved boundary related robustness issues.

An adaptive method that robustly produces high aspect ratio tetrahedra satisfying a general 3D metric specification without introducing hybrid semi-structured regions is presented. The method is intended for planar geometries, where more complex domains can be simulated with a cut-cell approach. A simple example with 1000–1 stretching is provided for illustration. A supersonic cut-cell simulation of a complex body is also presented to illustrate the utility of the cut-cell approach with an adapted tetrahedral background grid.

Domain decomposition is employed to fully exploit the distributed memory of a cluster of computers to increase simulation problem size. A secondary goal is to exploit parallelism to reduce the execution time of the adaptation process. An added benefit of a parallel implementation is that parallel applications, i.e., a CFD solver, may interface directly with the adaptation library without creating a global grid image. There are a number of parallel simplex adaptation examples in the literature.[38–47] Chrisochoides[48] has compiled a survey on the related problem of parallel grid generation.

An alternative approach to the standard formulation of adapting a grid to metric is introduced. The grid is to adapted to improve an objective function such as interpolation error. This allows direct control of actual computed error without the intervening metric specification. Bank and Smith[49] have investigated a similar approach for controlling estimated gradient errors of linear elements with 2D node movement. It is more natural to express an error integral for output-based and higher-order adaptation schemes. This procedure is demonstrated for an analytical function in 1D with fifth-order elements and in 3D with linear and quadratic elements.

II. Grid Operators

The grid adaptation scheme is constructed from a sequence of 3D nearest-neighbor grid modifications. Each grid in this sequence is represented as the current set of nodes $x_j \in \mathcal{X}_i$ and tetrahedra $\kappa_j \in \mathcal{T}_i$,

$$(\mathcal{X}_i, \mathcal{T}_i), \tag{1}$$

where x_j is the position of a node and κ_j are the nodes that this element connects. Each of the grid operators modify the current grid to create a new grid

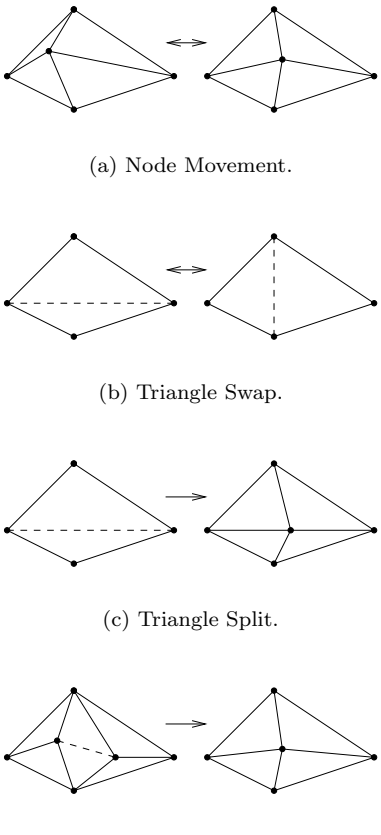
$$(\mathcal{X}_{i+1}, \mathcal{T}_{i+1}) = f(\mathcal{X}_i, \mathcal{T}_i). \tag{2}$$

The tetrahedra that change in a operation are in the sets

$$\breve{\mathcal{T}}_i = \mathcal{T}_i \setminus \mathcal{T}_{i+1} \text{ and} \tag{3}$$

$$\breve{\mathcal{T}}_{i+1} = \mathcal{T}_{i+1} \setminus \mathcal{T}_i. \tag{4}$$

The removed tetrahedra are $\breve{\mathcal{T}}_i$ and the newly introduced are $\breve{\mathcal{T}}_{i+1}$. The 2D analogue of node movement, tetrahedra swap, tetrahedra split, and tetrahedra collapse are shown in Fig. 1.



(a) Node Movement.

(b) Triangle Swap.

(c) Triangle Split.

(d) Triangle Collapse.

Figure 1. 2D Nearest-Neighbor Grid Modifications.

II.A. Node Movement

Node movement changes the locations of the nodes while keeping the tetrahedra connectivity fixed,

$$(\mathcal{X}_{i+1}, \mathcal{T}_i) = M(\mathcal{X}_i, \mathcal{T}_i). \tag{5}$$

This has also been a popular technique for structured grid methods,[16] because of the fixed connectivity. A node location is adjusted with the neighboring node locations fixed to produce a Gauss-Seidel style relaxation scheme. This method allows explicit constraints on the positivity of tetrahedral volume, i.e.,

$$V(\mathcal{X}_{i+1}, \mathcal{T}_i) > 0. \tag{6}$$

Global relaxation is not used, because explicit constraints can not be enforced directly. An objective function $|\kappa|$ is defined for each tetrahedron incident to a node, $\kappa \ni x$. An optimization procedure is invoked to minimize a norm of the incident tetrahedron objective functions,

$$x_j = \text{argmin} \left(\sum_{\kappa \ni x_j} |\kappa|^m \right)^{1/m}. \tag{7}$$

If the L_2-norm is selected a gradient-free simplex search is used when gradients are not available. If the L_∞-norm is selected, smart-Laplacian and quadratic programming optimization schemes[50] are implemented. Nodes on boundaries are moved in their respective parameterized spaces.

II.B. Tetrahedra Swapping

The 3D operations of face and edge swapping[50] are performed,

$$(\mathcal{X}_i, \mathcal{T}_{i+1}) = Sw(\mathcal{X}_i, \mathcal{T}_i). \tag{8}$$

The configuration that reduces a norm of the involved tetrahedron objective functions is chosen if it has positive volume tetrahedra,

$$\left(\sum_{\kappa \in \breve{\mathcal{T}}_{i+1}} |\kappa|^m \right)^{1/m} \leq \left(\sum_{\kappa \in \breve{\mathcal{T}}_i} |\kappa|^m \right)^{1/m} \text{ and} \tag{9}$$

$$V(\mathcal{X}_i, \mathcal{T}_{i+1}) > 0. \tag{10}$$

If there is no swapped configuration with lower norm, the current configuration is retained. Face swapping[50] is replacing two tetrahedra that share a three nodes with three tetrahedra. Edge swapping[50] is replacing 3 to 7 tetrahedra that all share two nodes with a new set of tetrahedra that fill the same region. The boundary triangles are also swapped as a result of tetrahedra swapping.

II.C. Tetrahedra Split and Collapse

The tetrahedra split

$$(\mathcal{X}_{i+i}, \mathcal{T}_{i+1}) = Sp(\mathcal{X}_i, \mathcal{T}_i) \tag{11}$$

and collapse

$$(\mathcal{X}_{i+i}, \mathcal{T}_{i+1}) = C(\mathcal{X}_i, \mathcal{T}_i) \tag{12}$$

operations modify the density of the grid tetrahedra as well as contribute to obtaining the desired anisotropy. There are many possible splitting stencils.[51,52] In this work the simplest 2 for 1 splitting is employed. The split operator inserts a single new node on the segment connecting two nodes. All the the tetrahedra that share this segment are split to include this new node. These split tetrahedra have positive volume if the original tetrahedra have positive volume and the new node is placed on the segment connecting two nodes. Segments that are on a boundary also result in split boundary triangles.

For a collapse, All the the tetrahedra that share a segment connecting two nodes are removed. One of these two nodes is removed and the remaining tetrahedra incident to the removed node are reconnected to the remaining node. The collapse operation is only permitted if the new configuration results in positive volume tetrahedra,

$$V(\mathcal{X}_{i+i}, \mathcal{T}_{i+1}) > 0. \tag{13}$$

III. Parallelization

The grid operators are executed in parallel with a domain decomposed scheme. The resulting partitions allow the independent application of operators to the partition interiors. Most of the complexity of this parallel approach arises when these operators are applied in the partition-border regions.

III.A. Domain Decomposition

To fit large problems on a cluster of distributed-memory machines, the domain or grid is decomposed into partitions. A node-based, domain-decomposition scheme is utilized, where each node is uniquely assigned to a single partition. Tetrahedra that span the partitions are duplicated to complete the border regions of the partitions. This is the Single Program Multiple Data (SPMD) paradigm; each partition data structure is processed by the same program. The domain decomposition scheme is identical to the method utilized by FUN3D.[53]

A 2D, two partition example is illustrated in Fig. 2. A node is denoted local and added to a partition if it has been assigned to that partition (denoted filled circles and open circles). The geometry edges, boundary triangles, and tetrahedra that contain one or more local nodes are also added to the partition. These include tetrahedra that are entirely local to a partition (solid lines) or border tetrahedra that span partition boundaries (dashed lines). Local nodes that are only used to construct local tetrahedra are denoted as purely-local nodes (filled circles). Local nodes that are used to construct border tetrahedra are denoted as border nodes (open circles).

The local nodes of other partitions are added as ghost nodes (dashed circles) to the current partition as required to complete the border tetrahedra. The partition that these ghost nodes are assigned is also stored to reduce the parallel communication required to update partition-border regions. Purely-local nodes are only connected by tetrahedra to other local nodes. A partition's border nodes are local nodes that are connected to a ghost node by a border tetrahedron. Therefore, nodes are either purely-local, border, or ghost. Tetrahedra are either local or border. Nodes have a unique global index and a local index on each partition.

The parallel aspects of node movement will be discussed first. Then the parallel aspects of connectivity changes, swap, split, or collapse, will be detailed. These will be followed by the implementation issues of maintaining unique global node and tetrahedra indexes and equally sized partitions for parallel efficiency.

III.B. Ghost Node Update

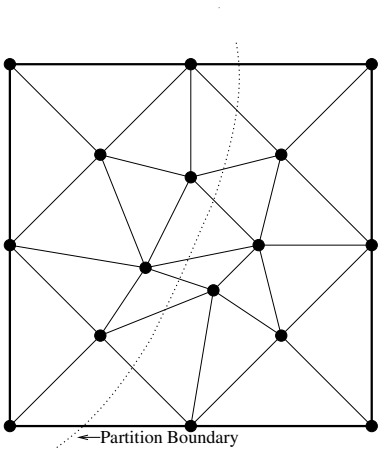
Ghost node parameters must be updated when the corresponding border node on another partition is modified. This operation is complicated by tetrahedron connectivity changes during adaptation. These connectivity changes in partition borders alter the inter-partition communication pattern to update ghost nodes.

To recompute the communication pattern, every partition sends a list of ghost node global indexes to the partition that is assigned those nodes. On the assigned partitions, these requests for updated nodal information are translated into local node indexes. The global indexes of local nodes are stored in a sorted list. The translation is performed in $\mathcal{O}(\log_2 n)$ time with a binary search algorithm. These indexes allow access to the stored local values of the requested nodal position, face parameters, and edge parameter. The requested information is sent back to the partitions to update their ghost nodes.

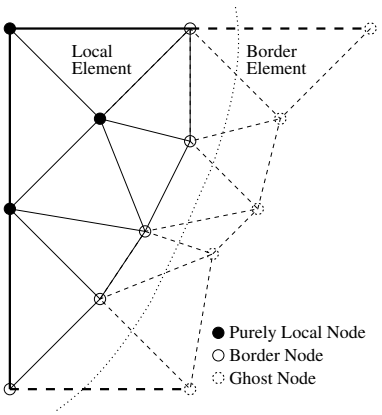
III.C. Purely-Local and Border Node Movement

The nearest-neighbor information must be current when smoothing nodes to prevent the creation of invalid tetrahedra. Freitag[41] and Löhner[39] address the issues for maintaining current nearest-neighbors while node smoothing on a shared-memory architecture. In other distributed memory implementations,[45,46] border nodes are frozen during adaptation. Then the domain is repartitioned and migration is used to change these frozen border nodes into local nodes. A number of repartitioning steps ultimately allow the processing of all nodes as local nodes. In this implementation, the nearest-neighbor node movement operators are applied separately to local and border regions without repartitioning.

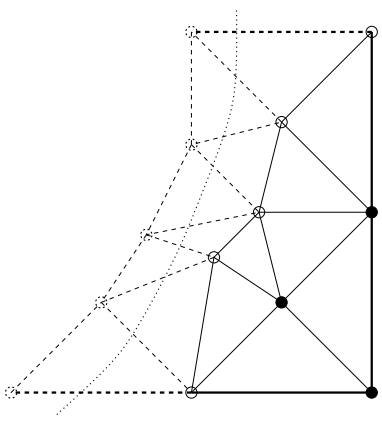
An initial pass is made through all the purely-local nodes on all partitions. These nodes do not require any information from other partitions. Therefore, the partitions can modify purely-local nodes concurrently without parallel communication. After the purely-local regions of the partition have been smoothed, the border nodes are smoothed one partition at a time. The ghost nodes are updated after each partition takes its turn to ensure that nearest-neighbor information is current. Unfortunately, only allowing execution on one partition results in suboptimal scaling, because the other partitions are idle until they have an opportunity to process their border nodes. The amount of time that processors remain idle during adaptation can be reduced by allowing concurrent execution through partition coloring.



(a) Global Grid.



(b) Partition 1.



(c) Partition 2.

Figure 2. Two Partition Domain Decomposition 2D Example.

III.D. Partition Coloring

To increase parallel efficiency, groups of unconnected partitions are allowed to simultaneously smooth border nodes in between ghost node updates. The sets of unconnected partition groups are calculated with a coloring scheme where no two adjoining partitions share a color. All the partitions in a single color can then move nodes concurrently with current ghost node information. Parallel communication is only required between the processing of different colors, which are less than or equal to the number of partitions. The partition coloring scheme is also applied to increase the parallel efficiency of border tetrahedron connectivity changes.

III.E. Local and Border Connectivity Changes

The connectivity of the tetrahedra change as a result of swapping, splitting, and collapsing. In other parallel adaptation implementations,[45–47,52] connectivity changes are performed in the interior of the partition and migration is used to make border regions interior. In this work, the tetrahedron connectivity changes are performed on purely-local and border tetrahedra in separate operations without migration. These connectivity changes may be due to edge swapping, face swapping, or the insertion of new nodes. The removal of nodes is currently allowed only in

purely-local partition regions, because the geometry topology information required to maintain a topologically valid discrete surface triangulation is not stored with ghost nodes. This limitation may be removed with a small amount of additional storage.

Initially, connectivity changes are only performed on local tetrahedra. This in an efficient parallel operation because all partitions perform connectivity changes simultaneously. Updates of the border tetrahedra region are performed one partition at a time. To correctly mirror connectivity changes across partitions, transcripts are recorded of the nodes and tetrahedra that are added or removed as a result of this border tetrahedra adaptation. These transcripts are serialized and broadcast to other partitions. The receiving partitions apply the transcripts to maintain consistency between partitions. The parallel inefficiency of this inherently sequential operation is alleviated by allowing unconnected partitions to change border connectivities concurrently via the partition coloring scheme described in Section III.D.

The transcript is composed of all the data required to update connectivities (tetrahedra, geometry triangles, and geometry edge segments) and create new ghost nodes (position, face parameters, and edge parameters). The size of these transcripts is reduced by utilizing the stored ghost node partition assignments to restrict the transcript to the minimum information required to maintain consistency. Minimizing the contents of the transcripts reduces parallel communication cost and the time required to perform the specified transcript operations on the other partitions. Parallel efficiency is maintained while applying the transcripts to other partitions, because transcripts are processed concurrently. Ghost nodes that are no longer connected to a border tetrahedron after connectivity changes are also removed.

III.F. Global Indexes

The global node indexes are required to reconstruct the parallel communication pattern after connectivity changes. Global cell indexes are not required for parallel adaptation, but are maintained as a convenience to the application requesting adaptation. Unique global cell indexes are maintained with exactly the same algorithm as the global node indexes, so only the global node index scheme is described.

The algorithm to assign and maintain global node indexes is simple. A partition takes ownership of a global node index during the initialization of its local nodes. Once a partition is assigned a global index, it never relinquishes it until the completion of adaptation. If a local node is deleted during adaptation, the partition stores this unused global index in a list. If the partition needs a unique global index to insert a node into the discretization, it will extract an unused global index from the list.

When this list of unused global indexes is exhausted, the partition creates a new global index by incrementing its count of global indexes. Two or more partitions may obtain the same global index; these repeated global indexes are made unique by shifting the newly created global indexes. After shifting the newly created global node indexes, the true count of global indexes is shared amongst partitions.

Figure 3 provides a four processor example that begins with 100 unique global nodes. Three of four partitions need new global nodes to insert nodes while concurrently performing edge split operations. Partition C does not insert any nodes. After the edge split operation, these newly created nodes are shifted to make them unique. The correct total number of nodes is also computed by partition D and communicated to all partitions. The parallel communication required to maintain unique global indexes is reduced by allowing the temporary creation of repeated global indexes that are later corrected.

	Partition	Total	Indexes of New Nodes
Before Shift	A	103	101 102 103
	B	102	101 102
	C	100	none
	D	104	101 102 103 104
After Shift	A	109	101 102 103
	B	109	104 105
	C	109	
	D	109	106 107 108 109

Figure 3. An Example of Creating New Global Indexes On Four Partitions With 100 Original Global Indexes, Before and After Shift.

The unused global indexes are removed at the completion of the adaptation by a reverse, global-index shifting procedure. This ensures that the calling application will be returned a grid with continuous global indexes.

III.G. Load Balancing

Load balancing is employed to maintain parallel efficiency. The parallel, graph-partitioning tool ParMETIS[54] is utilized to create well-balanced partitions with a minimum number of connections. During adaptation, the number of nodes in each partition can change significantly due to the addition and removal of nodes. Also, the communication cost can increase due to connectivity changes. At the completion of the adaptation process, ParMETIS is called and portions of the grid are migrated to regain an optimal partitioning.

IV. Adaptation to Anisotropic Metric

A metric tensor $\mathbf{M}$ is commonly employed to define the desired multidimensional grid resolution because it is a natural way to express local interpolation error estimates of linear functions.[21,24,25,55] The symmetric positive definite matrix $\mathbf{M}$ in 3D has the diagonal decomposition

$$\mathbf{M} = \mathbf{R} \begin{bmatrix} \Lambda_1 & & \\ & \Lambda_2 & \\ & & \Lambda_3 \end{bmatrix} \mathbf{R}^t = \mathbf{R}\Lambda\mathbf{R}^t. \tag{14}$$

The eigenvectors $\mathbf{R}$ define an orthonormal basis with length specifications h_i in this basis,

$$\Lambda = \begin{bmatrix} \left(\frac{1}{h_1}\right)^2 & & \\ & \left(\frac{1}{h_2}\right)^2 & \\ & & \left(\frac{1}{h_3}\right)^2 \end{bmatrix}. \tag{15}$$

This metric can be interpreted as an ellipsoid with major and minor axes of direction $\mathbf{R}_i$ and length h_i.[24] The linear mapping $\mathbf{J}$ is employed to map a vector in physical space $\mathbf{x} = [x\ y\ z]^t$ to a vector in transformed space $\mathbf{x}' = [x'\ y'\ z']^t$ where a triangle or tetrahedron becomes equilateral with unit length edges; that is,

$$\mathbf{x}' = \mathbf{J}\mathbf{x}. \tag{16}$$

The metric tensor $\mathbf{M}$ is related to $\mathbf{J}$ by

$$\mathbf{M} = \mathbf{J}^t\mathbf{J} \text{ and} \tag{17}$$

$$\mathbf{J} = \Lambda^{\frac{1}{2}}\mathbf{R}^t. \tag{18}$$

If a vector $\mathbf{x}$ describes an edge in physical space, the length l in mapped space is

$$l = \sqrt{\mathbf{x}'^t\mathbf{x}'}. \tag{19}$$

Employing Eq. (16), this expression of length becomes

$$l = \sqrt{(\mathbf{J}\mathbf{x})^t\,(\mathbf{J}\mathbf{x})} \text{ and} \tag{20}$$

$$l = \sqrt{\mathbf{x}^t\mathbf{M}\mathbf{x}}. \tag{21}$$

Equation (21) is employed to directly compute edge lengths in the specified metric. Equation (16) is applied to the physical coordinates of tetrahedron nodes to map them to the transformed space before evaluating a tetrahedron's quality.

The resolution request is stored as $\mathbf{M}$ during adaptation instead of storing $\mathbf{J}$ directly due to these benefits:

- It is compact; only 6 entries of the 3D symmetric $\mathbf{M}$ are stored.
- Multiple $\mathbf{M}$ can be readily interpolated and intersected.[21,55]
- Lengths in the specified metric can be computed efficiently with Eq. (21).

A slight computational cost is incurred computing $\mathbf{J}$ from $\mathbf{M}$ when $\mathbf{J}$ is needed to obtain metric transformed tetrahedron. This cost is a 3×3 symmetric positive definite matrix diagonalization and Eq. (18). This cost is minimized by tridiagonalizing $\mathbf{M}$ with a single rotation and applying an iterative method to find the eigendecomposition of the tridiagonal matrix.

Grid adaptation is essentially an optimization procedure where a current mesh is iteratively modified to improve a figure of merit. These modifications can be continuous (node movement) or discontinuous (connectivity change). The optimization process requires an objective function. In 3D, considering only edge length as a quality measure can result in degenerate tetrahedra, because a nearly-zero-volume 'sliver' tetrahedra can be constructed without a short edge. Shape measures[1,56] are employed to penalize near-degeneracies and provide a smooth function to optimize. These measures are based on interpolation error estimates, departure from a right or isotropic tetrahedron, or transformation matrix conditioning.[57] A norm of the mesh conformity to a specified multidimensional grid resolution[58] can also be stated directly.

Shape measures are typically defined in the range $[0, 1]$, with 0 denoting a degenerate tetrahedron and 1 denoting an ideal tetrahedron. In this work, the inverse of a shape measure is used so that grid optimization becomes a minimization problem. The inverse of the mean ratio[1]

$$\eta = \frac{l_{01}^2 + l_{02}^2 + l_{03}^2 + l_{12}^2 + l_{13}^2 + l_{23}^2}{12(3V)^{2/3}} \tag{22}$$

is selected for this work, where V is the volume of the tetrahedron and l are the six lengths of vectors defined between the nodes of the tetrahedron. It is in the range of $[1, \infty]$, where an isotropic tetrahedra is 1 and a degenerate tetrahedra is ∞. It is efficient to evaluate and a well-behaved continuously differentiable function,[57] which makes it very suitable for gradient-based optimization. It also has a connection to metric conformity without size specification.[58] Minimizing η in the specified metric produces tetrahedra that are isotropic in the metric space. This tends to produce tetrahedra with large dihedral angles in physical space. Shape measures that penalize large angles can have discontinuous derivatives, making them difficult to optimize with gradient-based methods, see Shewchuk[57] for examples.

IV.A. Metric Adaptation Iteration

Directly stating and adapting to minimize a norm of the conformity of grid tetrahedra to an implied metric provides simple minimization statement. This metric conformity approach has been applied in 2D to a metric with 20–1 anisotropy.[58] This approach is extended to 3D for evaluation in the context of this study. Both L_∞- and L_2-norms of tetrahedral $\mathcal{E}_K$[58] where minimized in a 10–1–1 anisotropic metric field using the grid operators defined in Section II with limited success. This metric conformity approach is extremely attractive because it can be stated as a single minimization objective, however it has not been able to outperform the following ad-hoc method in terms of anisotropy or execution time.

The goal of the ad-hoc adaptive processes is to produce a grid with the following properties in decreasing priority:

- All tetrahedra have positive volume.
- All edges have a length less than or equal to unity in the specified metric field.
- The number of nodes and tetrahedra in the mesh should be reduced by collapsing edges shorter then unity in the metric field.
- The tetrahedra shape quality in the metric should be improved.

These are potentially contradictory goals that do not unify to a single minimization statement. However, grid validity with elements equal or smaller than the metric is possible with tetrahedra splits alone for a domain with planar boundaries, avoiding being trapped in the local minima of an objective function. The last two goals are for efficiency (minimizing degrees of freedom) and regularity of the resulting grid.

A iterative procedure to reach these goals is:

- Sort the tetrahedra from largest η_κ to smallest in the specified metric. Attempt the swap operation on all tetrahedra to a new or existing configuration that minimizes the maximum η of the involved tetrahedra ($m = \infty$) to the greatest extent while maintaining positive volume,

$$Sw(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i \text{ from largest } \eta_{\kappa_j} \text{ to smallest } \eta_{\kappa_j}. \tag{23}$$

- Perform the node movement operation on all nodes from largest $\eta_x = ||\eta||_\infty$ over incident tetrahedra to smallest,

$$M(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; x_j \in \mathcal{T}_i \text{ from largest } \eta_{x_j} \text{ to smallest } \eta_{x_j}. \tag{24}$$

  Smart-Laplacian and quadratic programming optimization schemes[50] are performed on the prioritized node list to minimize η.

- The orbit of tetrahedra that surround segments connecting two nodes are examined for potential collapse if the length of the edge in the metric is $l < 0.3$. This collapse is performed if the new configuration does not introduce any new edges $l > 1.0$ and the resulting tetrahedra have positive volume,

$$C(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; l \in \kappa_j \in \mathcal{T}_i : l < 0.3 \text{ iff } l \in \kappa_j \in \breve{\mathcal{T}}_{i+1} \leq 1.0. \tag{25}$$

- Sort the grid edges from largest l to smallest. The edges with $l > 1.0$ are split,

$$Sp(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; l \in \kappa_j \in \mathcal{T}_i : l > 1.0 \text{ from largest } l \text{ to smallest } l. \tag{26}$$

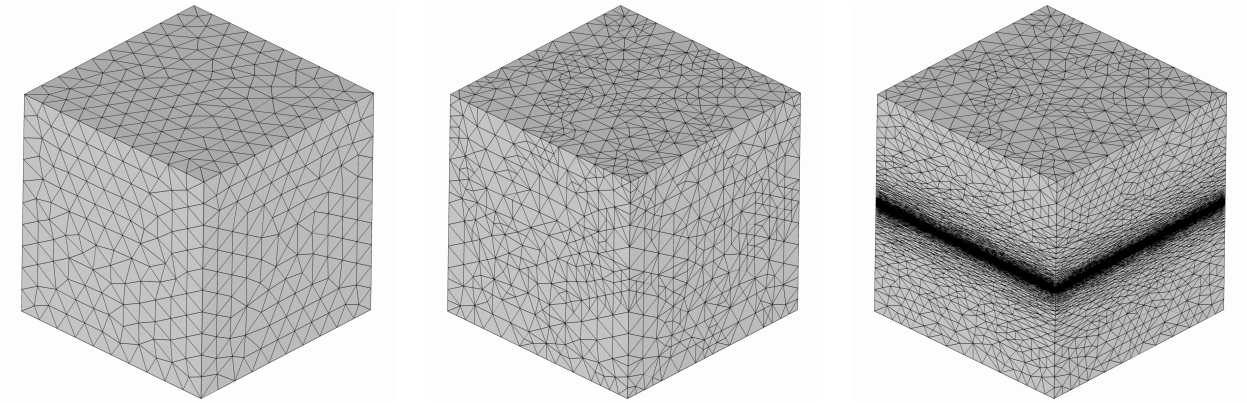
- The adaptation process repeats until $l < 1.0 \forall l \in \mathcal{T}_i$.

IV.B. Analytic Metric Adaptation

The volume grid of a unit cube domain $[0,1] \times [0,1] \times [0,1]$ is adapted to an analytic metric field. The original grid, Fig. 4(a), is created by the FELISA grid generator[59] through the GridEx framework[60] with an isotropic spacing of 0.1. This original grid is adapted to ensure that all edges are less than or equal to 0.1, Fig. 4(b). An anisotropic metric

$$[0.1d\hat{x}, 0.0001 + 0.0999\frac{|y - 0.5|}{0.5}d\hat{y}, 0.1d\hat{z}] \tag{27}$$

is used for adaptation in Fig. 4(c). The volume grid is clustered near the $y = 0.5$ plane. This clustering is evident on the $x = 0$ and $z = 0$ visible surfaces of the cube.



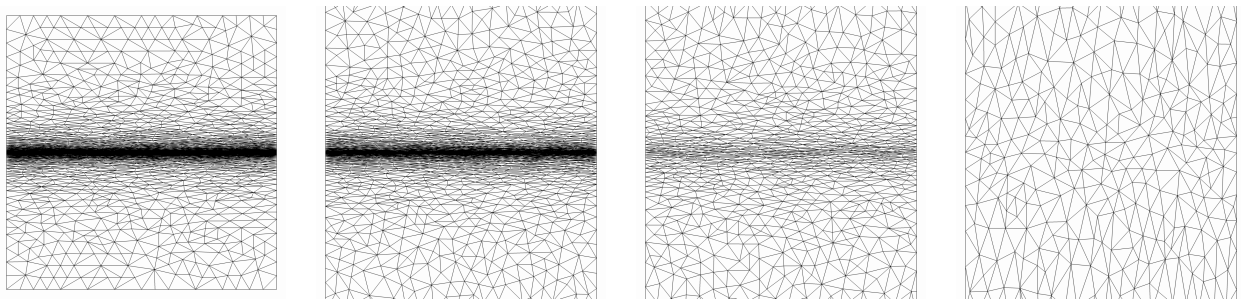
(a) FELISA Grid.

(b) All Edges Smaller than Metric 0.1–0.1–0.1.

(c) All Edges Smaller than Metric 0.1–0.1–0.0001.

Figure 4. Isometric Views of a Cube.

Anisotropic scaling of the figures aids illustration of the the grid corresponding to this simple metric field. The $z = 0$ face of the Fig. 4(c) cubic grid is shown in Fig. 5. The y-axis is progressively scaled in the series of subfigures, but the grid itself remains fixed. Figure 5(a) shows the anisotropically adapted grid with 1–1 scaling. The y-axis scales by a factor of ten in each subsequent subfigure while the x axis is held constant. Figure 5(d) shows the anisotropically adapted grid with 1–1000 scaling, which results in a figure width of 1 and a figure height of 0.001 in physical space. The center of the strongly anisotropic grid appears isotropic in this anisotropically scaled view as a result of this mapped isotropic method.



(a) 1–1 View. (b) 1–10 View. (c) 1–100 View. (d) 1–1000 View.

Figure 5. Face of a Cube ($z = 0$) Adapted to a 0.1–0.1–0.0001 Metric.

IV.C. Flow Solution Metric Adaptation

The adaptation mechanics have been applied to 3D output-based[61–63] and local-error-estimate-based[64] adaptive simulations with body-fitted grids. These previous examples have been limited to mild anisotropy or frozen boundary discretizations to avoid the complexities of high anisotropy adaptation of body-fitted grids on curved boundaries.[28] An alternative is to employ a cut-cell procedure to handle complex geometries and perform adaptation on an uncut background grid with planar geometry (e.g., cube).

An example of a geometry that may suffer from body-fitted adaptation robustness is NASA ship 837, a modified F-15.[65] This geometry contains many narrow regions around the control surfaces and high surface curvatures. The shaded surface of the modified F-15 is shown in Fig. 6 and is composed of 544 thousand triangles. The volume defined interior to this surface is subtracted from the dual of a tetrahedral background grid. The resulting triangle-triangle intersections[32] determine the portion of the background grid that remains inside the computational domain. The FUN3D flow and adjoint solver are modified to handle the complex cut cells[66] that arise at the intersection of the surface and background grids. An anisotropic output-based adaptation scheme[11,61] is modified to account for the cut cells. An implied metric is constructed for the background grid nodes inside the surface geometry, which are outside of the computational domain to limit unnecessary adaptation of these regions. Adaptation iterations are performed to improve the integral of a surface pressure p delta from free stream pressure p_0 below the configuration,[63]

$$\iint (p - p_0)\, \mathrm{d}A. \tag{28}$$

The cylindrical integration surface is trimmed to approximately the width of the wing tip and the length of the beginning and end of the boom signature below the aircraft, see Fig. 7. The cylindrical surface is constructed by triangulating the iso-surface of radius, see the "Volume Slicing" section of the Visual3[67] description. The simulation is performed at Mach 1.4 and 1 deg angle of attack. The symmetry plane of the original and adapted tetrahedral background grid is shown in Fig. 8. A crude outline of the F-15 on the symmetry plane is indicated on the adapted grid where the grid inside the configuration is not modified because the implied metric is satisfied. The background grid has anisotropically aligned to the complex shock structure of the supersonic flow field seen in Fig. 9. The density of the grid is specified by the output-based adaptation intensity and the anisotropy is specified by a Mach Hessian.[11,61]

V. Adaptation to Directly Control Interpolation Error

The adaptation metric is based on local interpolation error estimates, specifically, the departure of a linear and quadratic representation of the solution via a Hessian.[18] Output-based adaptation methods include a similar weighted interpolation error.[11] The Hessian method determines the direction of the largest next higher order derivative for linear functions. For functions with order higher than linear, a search for this direction can be employed.[13] An alternative methodology is to evaluate the (possibly weighted) interpolation error directly to define the adaptation objective function.

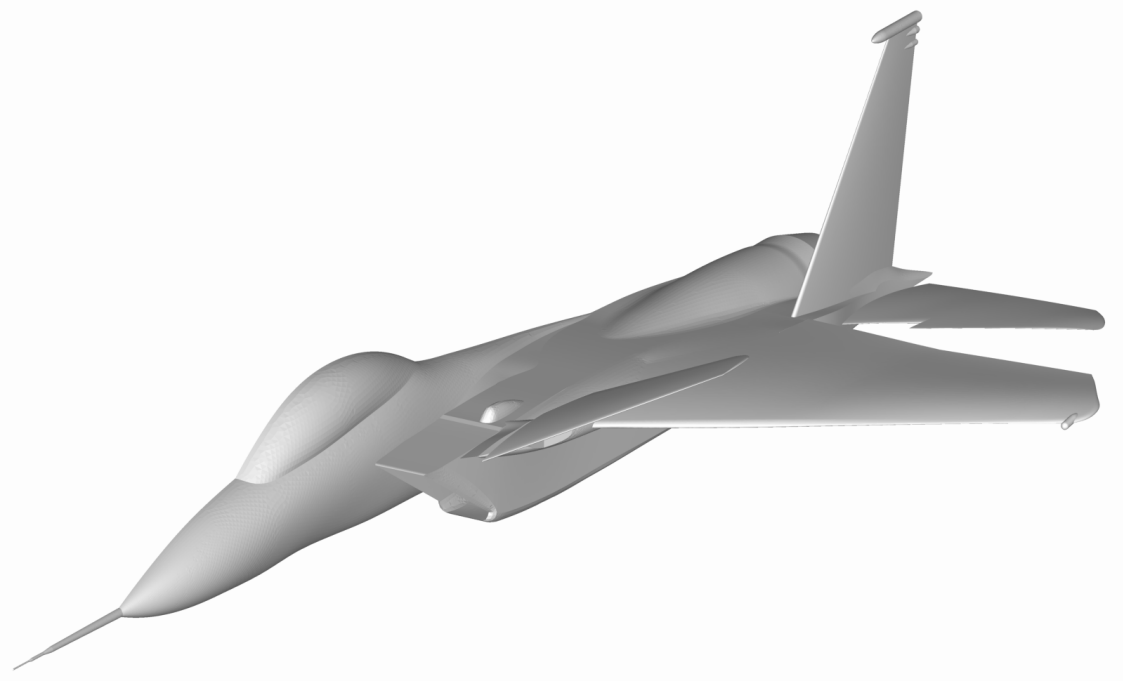


Figure 6. Surface Geometry of the Starboard Half of NASA Ship 837.

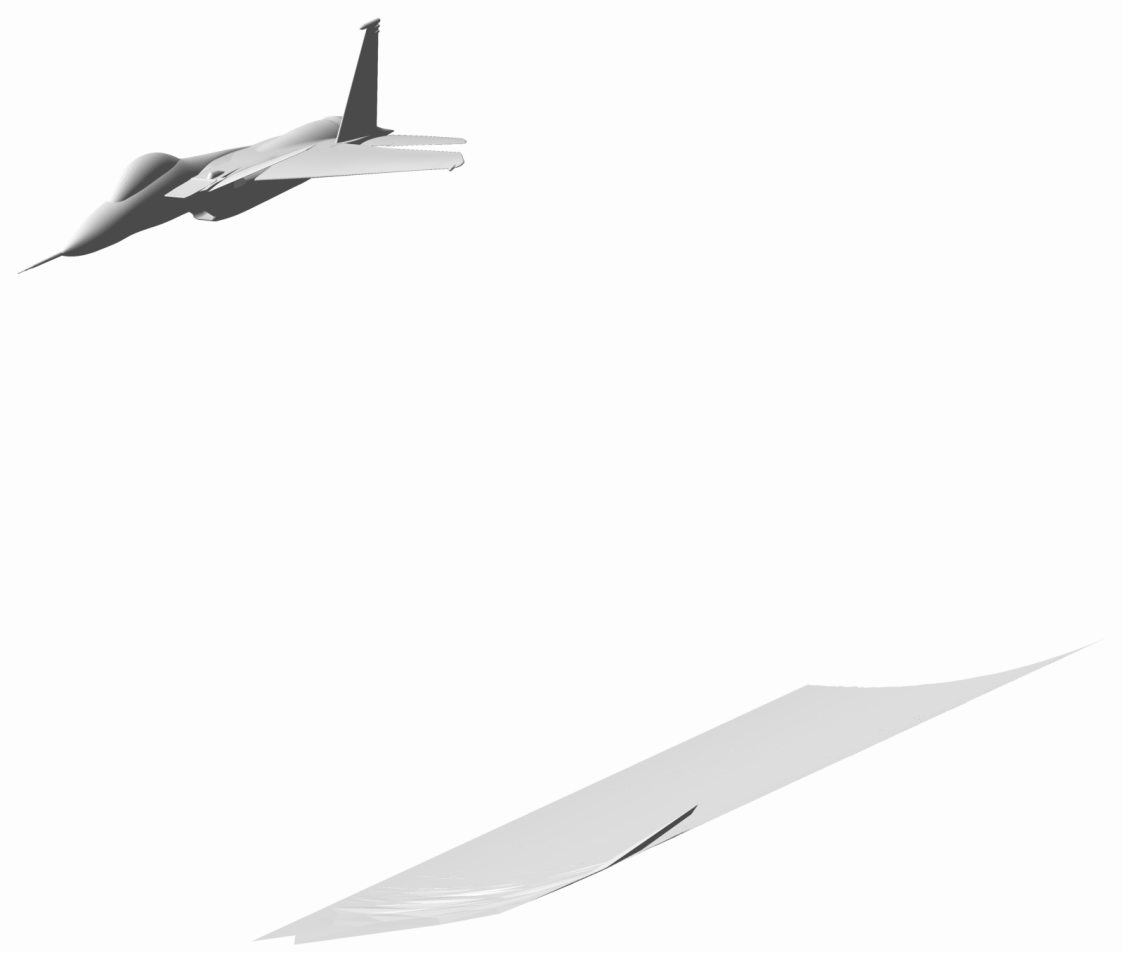


Figure 7. Surface Geometry of the Starboard Half of NASA Ship 837 and Pressure Integration Surface.

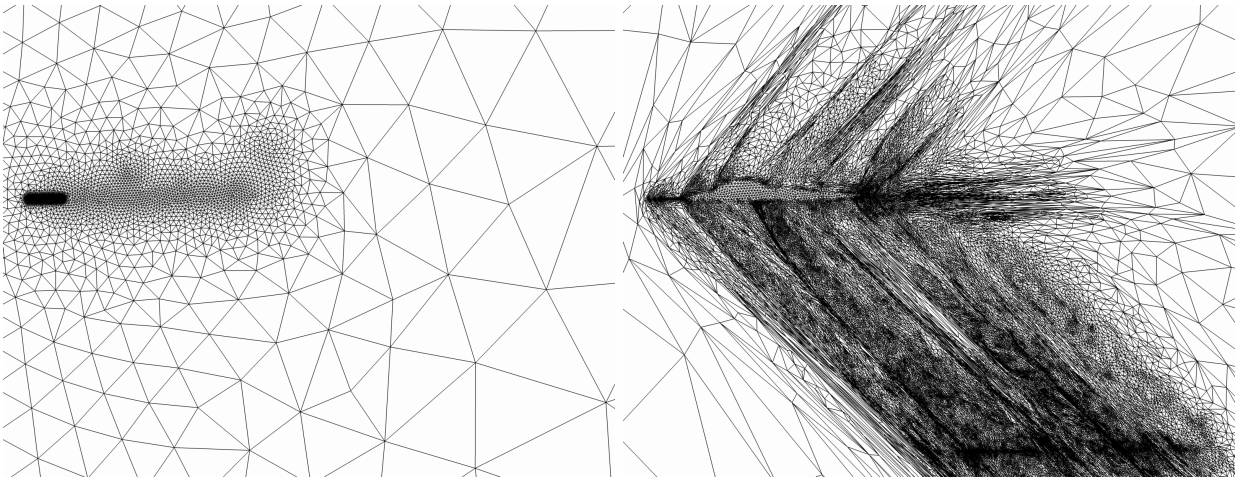


Figure 8. Symmetry Plane Triangulation of the Original and Adapted Tetrahedral Background Grid.

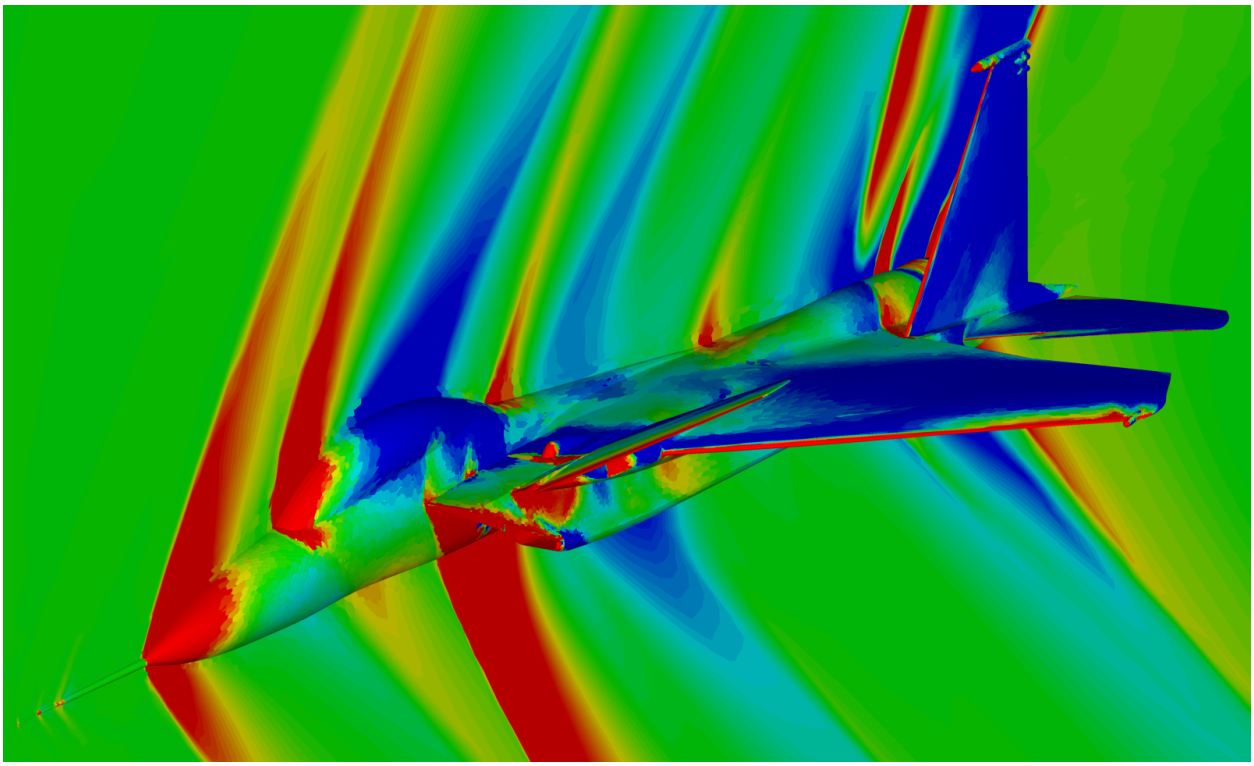


Figure 9. Computed Surface and Symmetry Plane Pressure Contours on the Final Adapted Grid.

Weighted interpolation error is defined as

$$\left(\int_{\Omega} |w(\bar{u}-u)|^m \, \mathrm{d}\Omega\right)^{1/m}, \tag{29}$$

where the exact solution u, interpolant $\bar{u}$, and the weight w are defined in the domain Ω. The $\bar{u}$ is defined as a p-order polynomial in each element κ, which completely discretize Ω. The u is sampled to form $\bar{u}$ in each element.

This u may be an unknown solution (possibly of a partial differential equation), but if an approximate solution $\tilde{u}$ is available on a set of discrete elements κ_0, the exact solution is approximated as follows. The $\tilde{u}$ consists of p-order polynomials individually defined in each κ_0. To estimate the leading order terms of the exact interpolation error, a $p+1$-order solution $\hat{u}$ is reconstructed from $\tilde{u}$.[68] This results in a computable interpolation error

$$\left(\int_{\Omega} |w(\bar{u}-\hat{u})|^m \, \mathrm{d}\Omega\right)^{1/m}. \tag{30}$$

V.A. 1D Interpolation Error Adaptation Example

A 1D interpolation error adaptation example is shown in Fig. 10 for $p = 5$ elements. The tanh exact function u is defined in the 1D region $[0, 1]$,

$$u = \tanh(50(x - 0.4)). \tag{31}$$

Although the exact solution is available for this case, a $\tilde{u}$ will be constructed by fitting the p-order Lagrangian basis that minimizes

$$\left(\int_{\kappa_0} |\tilde{u}-u|^2 \, \mathrm{d}\kappa_0\right)^{1/2} \tag{32}$$

independently in each element. The left column of Fig. 10 shows this discontinuous fit for a series of adapted 1D grids. A $\mathcal{C}^0$ $p+1$-order function $\hat{u}$ is formed from the discontinuous $\tilde{u}$ by minimizing

$$\left(\int_{\Omega} |\hat{u}-\tilde{u}|^2 \, \mathrm{d}\Omega\right)^{1/2} \tag{33}$$

over the entire domain. A $\mathcal{C}^0$ $\tilde{u}$ is required to prevent the adaptation process from introducing zero width elements at the discontinuous function jumps. This $\tilde{u}$ is shown in the right column of Fig. 10. The domain discretization is adapted into elements κ. Given a local element output error estimate of the form

$$e_\kappa = \left(\int_{\kappa} |w(\bar{u}-\hat{u})|^m \, \mathrm{d}\kappa\right)^{1/m}, \tag{34}$$

where $\bar{u}$ is a p-th order polynomial representation of the function $\hat{u}$ in each element and $m = 2$. The function $\bar{u}$ is directly interpolated from $\hat{u}$. The total output error for the domain is the sum of the elemental errors

$$e_\Omega = \left(\sum_{\kappa\in\Omega} (e_\kappa)^m\right)^{1/m}. \tag{35}$$

Following Zienkiewicz and Zhu,[69] the goal is to equidistribute this error estimate. An equal amount of error for each element is

$$\bar{e}_\kappa = \left(\frac{(\bar{e}_\Omega)^m}{N}\right)^{1/m}, \tag{36}$$

where $\bar{e}_\Omega$ is the requested total interpolation error and N is the number of elements. A step of the adaptive procedure is as follows;

- Given the current grid, construct a discontinuous p Lagrange basis $\tilde{u}$ with an L_2-norm fit of the element at Gaussian integration points weighted with the Gaussian integration weights.
- Reconstruct a continuous $p+1$ Lagrange basis $\hat{u}$ with an L2-norm fit of $\tilde{u}$ over the domain.

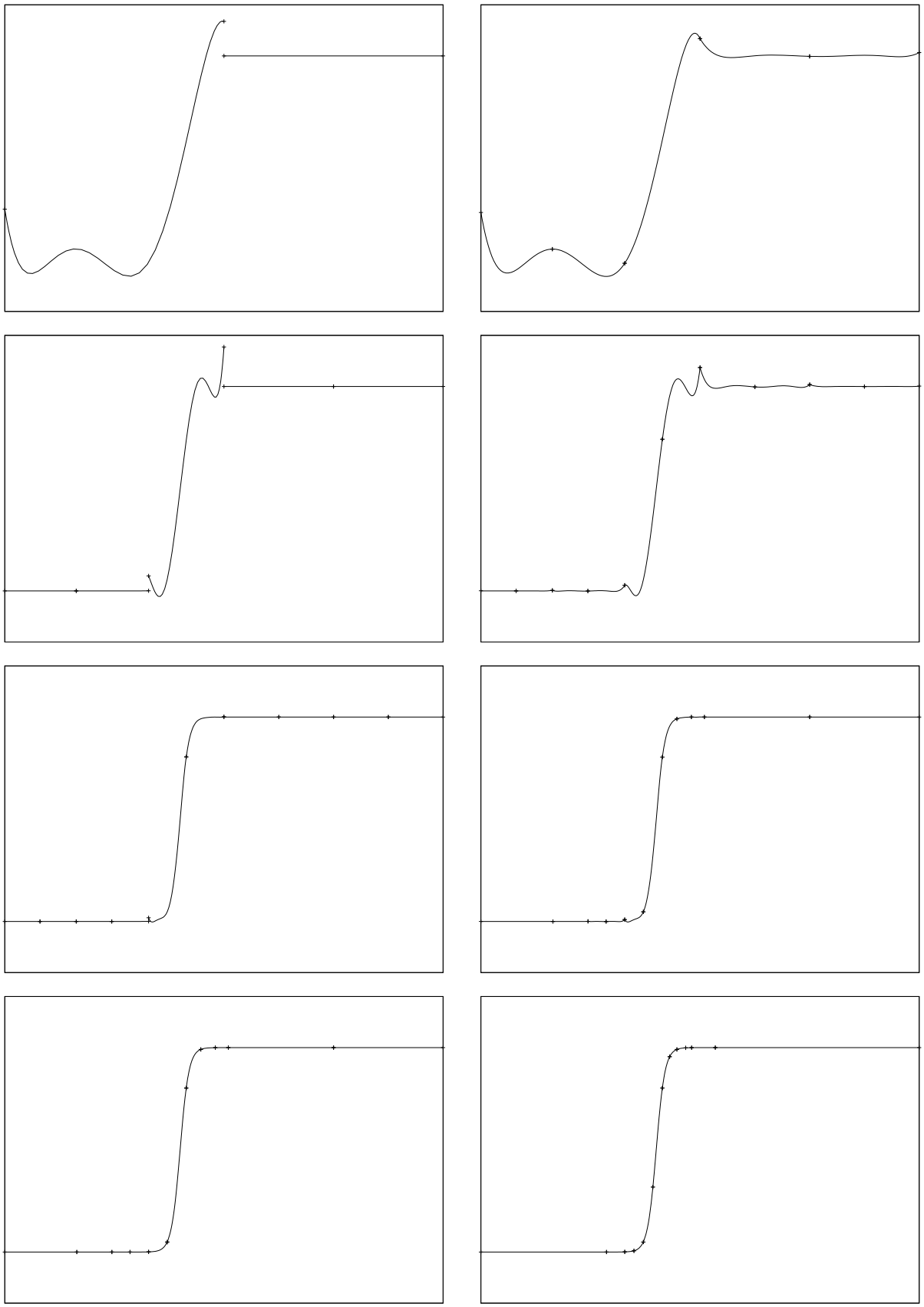


Figure 10. Four 1D Adaptation Cycles to the Interpolation Error of $\tanh$ **with** $p = 5$ **Basis (One Cycle per Row, Approximate Solution in Left Column, Reconstructed Solution in right Column)**

- The function w is assumed to be unity and $m = 2$.
- Exit adaptive procedure if $e_\Omega \leq \bar{e}_\Omega$.
- Repeat until the number of elements stabilize:
  - Evaluate Eq. (34) for all elements in the grid; split the element in half with the largest value of e_κ if $e_\kappa > \bar{e}_\kappa$,

$$Sp(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i : e_{\kappa_j} > \bar{e}_\kappa \text{ from largest } e_{\kappa_j} \text{ to smallest } e_{\kappa_j}. \tag{37}$$

  - Evaluate Eq. (34) for all elements in the grid; merge the element with the smallest value of e_κ with a neighbor if $e_\kappa < \bar{e}_\kappa$ after merge,

$$C(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i : e_{\kappa_j} < \bar{e}_\kappa \text{ iff } e_{\kappa_j} \in \breve{\mathcal{T}}_{i+1} \leq \bar{e}_\kappa \text{ from smallest } e_{\kappa_j} \text{ to largest } e_{\kappa_j}. \tag{38}$$

- Perform 2 Gauss-Seidel sweeps of node position optimization; each sweep progresses from the nodes of the element with the largest e_κ to the nodes of element with the smallest e_κ,

$$M(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; x \in \kappa_j \in \mathcal{T}_i \text{ from largest } e_{\kappa_j} \text{ to smallest } e_{\kappa_j}. \tag{39}$$

These steps are repeated to form the adaptive procedure.

Each step of the adaptive procedure is shown as a row in Fig. 10. The upper left subfigure shows $\tilde{u}$ on the initial two element grid. Element boundaries are marked with a + symbol. The inter-element jump is evident for this discontinuous $p = 5$ Lagrangian basis on the initial grid. The upper right subfigure shows the first adapted grid interpolating the $p = 6$ continuous Lagrangian basis $\hat{u}$. The second row of subfigures exhibits the Gibbs phenomenon near the rise in the tanh function. This phenomenon is damped in the third and fourth rows as adaptation progresses.

V.B. 3D Interpolation Error Adaptation Example

A cubic domain is employed in an 3D adaptation example to directly control interpolation error. The adaptation consists of four modes. These are tetrahedra swapping, node movement, tetrahedra collapse, and tetrahedra split.

- Sort the tetrahedra from largest e_κ to smallest. Perform the swap operation on all tetrahedra with $e_\kappa > \bar{e}_\kappa$ to a new or existing configuration that minimizes $(\sum e_\kappa^m)^{1/m}$ of the involved tetrahedra to the greatest extent while maintaining positive volume,

$$Sw(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i : e_{\kappa_j} > \bar{e}_\kappa \text{ from largest } e_{\kappa_j} \text{ to smallest } e_{\kappa_j}. \tag{40}$$

- Perform the node movement operation on all nodes from largest

$$e_x = \left(\sum_{\kappa \ni x} \frac{(e_\kappa)^m}{V_\kappa} \right)^{1/m} \tag{41}$$

  norm over incident tetrahedra to smallest,

$$M(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; x_j \in \mathcal{X}_i \text{ from largest } \eta_{x_j} \text{ to smallest } \eta_{x_j}. \tag{42}$$

  The volume of the tetrahedra V_κ is included to penalize nearly degenerate tetrahedra. A simplex search is performed on the prioritized node list to optimize the inverse volume weighted norm.

- The orbit of tetrahedra that surround segments connecting two nodes are examined for potential collapse. This collapse is performed if the new configuration has an error norm less than the specified $\bar{e}_\kappa$ and the resulting tetrahedra have positive volume,

$$C(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i : e_{\kappa_j} < \bar{e}_\kappa \text{ iff } e_{\kappa_j} \in \breve{\mathcal{T}}_{i+1} \leq \bar{e}_\kappa \text{ from smallest } e_{\kappa_j} \text{ to largest } e_{\kappa_j}. \tag{43}$$

- Sort the tetrahedra from largest e_κ to smallest. The six edges of the tetrahedra with $e_\kappa > \bar{e}_\kappa$ are examined and a split is performed that lowers e_κ to the greatest amount,

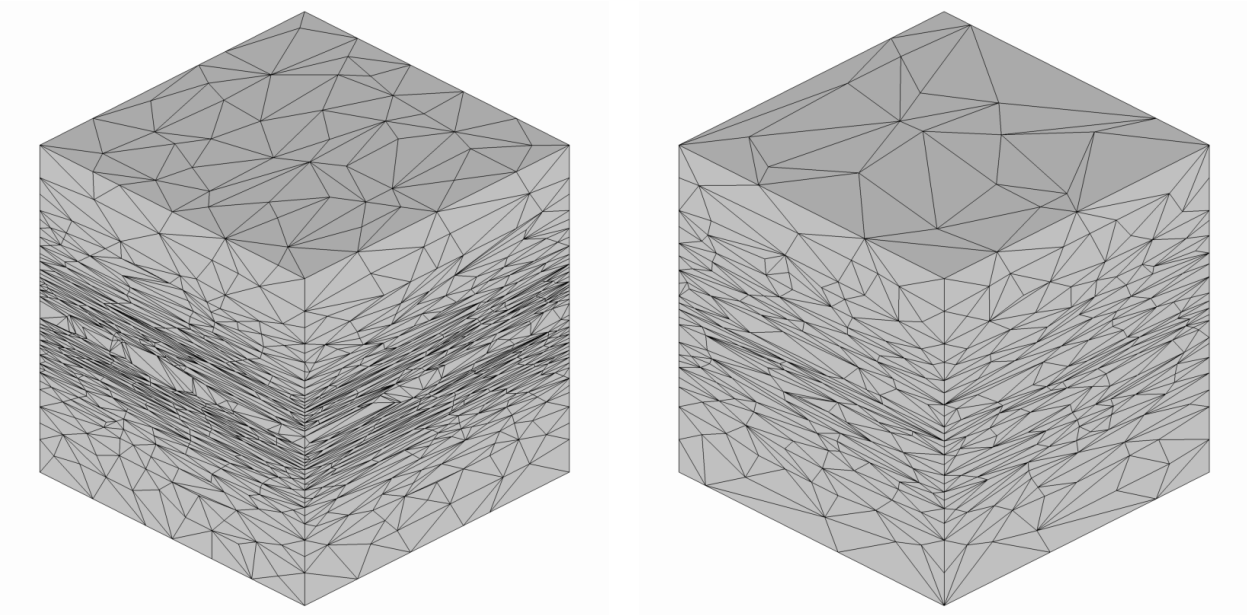
$$Sp(\mathcal{X}_i, \mathcal{T}_i) \; \forall \; \kappa_j \in \mathcal{T}_i : e_{\kappa_j} > \bar{e}_\kappa \text{ from largest } e_{\kappa_j} \text{ to smallest } e_{\kappa_j}. \tag{44}$$

- The adaptation process repeats until $e_\Omega > \bar{e}_\Omega$.

The exact function for this example is

$$u = 8x^2 + 32y^2 + 1000 \tanh(10(z - 0.5)). \tag{45}$$

As in the 1D example, the interpolation weight is unity $w = 1$. Approximate and reconstructed solutions are not used for this 3D case; the exact function is evaluated for interpolation error. The initial grid is shown in Fig. 4(a). The grid adapted to $\bar{e}_\Omega = 1.0$ for $p = 1$ tetrahedra is shown in Fig. 11(a). The grid adapted to $\bar{e}_\Omega = 0.1$ for $p = 2$ tetrahedra is shown in Fig. 11(b). For both cases the final interpolation error norm is slightly less than half the requested error norm. An anisotropic metric is not employed for this case; interpolation error is directly specified and controlled. The anisotropy seen in Fig. 11 is a result of efficiently resolving the anisotropic function.



(a) Adapted $p = 1$ to $\bar{e}_\Omega = 1.0$, Actual $e_\Omega = 0.46$.

(b) Adapted $p = 2$ to $\bar{e}_\Omega = 0.1$, Actual $e_\Omega = 0.041$.

Figure 11. Isometric Views of an Interpolation Error Adapted Cube.

VI. Concluding Remarks

An anisotropic tetrahedral grid adaptation scheme is demonstrated for a 1000–1 metric in a simple cube geometry. A supersonic adaptive cut-cell simulation of a fighter geometry is also presented. The method adapts an existing mesh to a specified metric and does not require semi-structured regions to attain high aspect ratios. The adaptation operators are described with their respective domain-decomposed parallelizations. An alternative approach is introduced that avoids the use of a specified metric and directly evaluates interpolation error to drive the adaptation mechanics. This direct method is applicable to output-based and higher-order adaptation without the concessions required to fit into a metric adaptation framework.

References

[1]Liu, A. and Joe, B., "Relationship between tetrahedron shape measures," *BIT Numerical Mathematics*, Vol. 34, No. 2, 1994, pp. 268–287.

[2]Levy, D. W., Zickuhr, T., Vassberg, J., Agrawal, S., Wahls, R. A., Pirzadeh, S., and Hemsch, M. J., "Data Summary from the First AIAA Computational Fluid Dynamics Drag Prediction Workshop," *AIAA Journal of Aircraft*, Vol. 40, No. 5, 2003, pp. 875–882, See also AIAA Paper 2002–841.

[3]Hemsch, M. J., "Statistical Analysis of Computational Fluid Dynamics Solutions from the Drag Prediction Workshop," *AIAA Journal of Aircraft*, Vol. 41, No. 1, 2004, pp. 95–103, See also AIAA Paper 2002–842.

[4]Laflin, K. R., Klausmeyer, S. M., Zickuhr, T., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Rakowitz, M. E., Tinoco, E. N., and Godard, J.-L., "Data Summary from Second AIAA Computational Fluid Dynamics Drag Prediction Workshop," *AIAA Journal of Aircraft*, Vol. 42, No. 5, 2005, pp. 1165–1178, See also AIAA Paper 2004–555.

[5]Hemsch, M. J. and Morrison, J., "Statistical Analysis of CFD Solutions from 2nd Drag Prediction Workshop," AIAA Paper 2004–556, 2004.

[6]Vassberg, J., Mani, E. T. M., Brodersen, O., Eisfeld, B., Wahls, R., Morrison, J., Zickuhr, T., Laflin, K., and Mavriplis, D., "Summary of the Third AIAA CFD Drag Prediction Workshop," , No. 2007–260, 2007.

[7]Mavriplis, D. J., "Grid Resolution Study of a Drag Prediction Workshop Configuration Using the NSU3D Unstructured Mesh Solver," AIAA Paper 2005–4729, 2005.

[8]Chaffin, M. S. and Pirzadeh, S., "Unstructured Navier-Stokes High-Lift Computations on a Trapezoidal Wing," AIAA Paper 2005–5084, 2005.

[9]Murman, S. M., Aftosmis, M. J., and Nemec, M., "Automated Parameter Studies Using a Cartesian Method," AIAA Paper 2004-5076, 2004.

[10]Nemec, M. and Aftosmis, M. J., "Aerodynamic Shape Optimization Using a Cartesian Adjoint Method and CAD Geometry," AIAA Paper 2006-3456, 2006.

[11]Venditti, D. A. and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows," *Journal Computational Physics*, Vol. 187, 2003, pp. 22–46.

[12]Dompierre, J., Vallet, M.-G., Bourgault, Y., Fortin, M., and Habashi, W. G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured grids," *International Journal for Numerical Methods in Fluids*, Vol. 39, No. 8, 2002, pp. 675–702.

[13]Fidkowski, K. J. and Darmofal, D. L., "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier–Stokes equations," *Journal Computational Physics*, Vol. 225, No. 2, Aug. 2007, pp. 1653–1672.

[14]Mavriplis, D. J., "Unstructured Mesh Generation and Adaptivity," Tech. Rep. ICASE 95-26, NASA Langley, Hampton VA, April 1995, See also VKI Lecture Series 1995-2 and NASA-CR-195069.

[15]Owen, S. J., "A Survey of Unstructured Mesh Generation Technology," *7th International Meshing Roundtable*, Oct. 1998.

[16]Baker, T. J., "Mesh Adaptation Strategies for Problems in Fluid Dynamics," *Finite Elements in Analysis and Design*, Vol. 25, No. 3–4, 1997, pp. 243–273.

[17]Parthasarathy, V. and Kallinderis, Y., "Adaptive prismatic-tetrahedral grid refinement and redistribution for viscous flows," *AIAA Journal*, Vol. 34, No. 4, 1996, pp. 707–716.

[18]Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C., "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, Vol. 72, No. 2, 1987, pp. 449–466.

[19]Pirzadeh, S. Z., "A Solution-Adaptive Unstructured Grid Method by Grid Subdivision and Local Remeshing," *AIAA Journal of Aircraft*, Vol. 37, No. 5, September–October 2000, pp. 818–824, See also AIAA Paper 99–3255.

[20]Mavriplis, D. J., "Turbulent Flow Calculations Using Unstructured and Adaptive Meshes," *International Journal for Numerical Methods in Fluids*, Vol. 13, 1991, pp. 1131–1152.

[21]Castro-Diáz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic Unstructured Mesh Adaptation for Flow Simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, No. 4, 1997, pp. 475–491.

[22]Borouchaki, H., George, P. L., Hecht, F., Laug, P., and Saltel, E., "Delaunay mesh generation governed by metric specifications. Part I. Algorithms," *Finite Elements in Analysis and Design*, Vol. 25, No. 1–2, 1997, pp. 61–83.

[23]Borouchaki, H., George, P. L., Hecht, F., Laug, P., and Saltel, E., "Delaunay mesh generation governed by metric specifications. Part II. Applications," *Finite Elements in Analysis and Design*, Vol. 25, No. 1–2, 1997, pp. 85–109.

[24]Peraire, J., Peirò, J., and Morgan, K., "Adaptive Remeshing for Three-Dimensional Compressible Flow Computations," *Journal of Computational Physics*, Vol. 103, No. 2, 1992, pp. 269–285.

[25]Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, 2000, pp. 725–744.

[26]Rannacher, R., "Adaptive Galerkin Finite Element Methods for Partial Differential Equations," *Journal of Computational and Applied Mathematics*, Vol. 128, 2001, pp. 205–233.

[27]Pierce, N. A. and Giles, M. B., "Adjoint Recovery of Superconvergent Functionals from PDE Approximations," *SIAM Review*, Vol. 42, No. 2, 2000, pp. 247–264.

[28]Li, X., Shephard, M. S., and Beall, M. W., "Accounting for curved domains in mesh adaptation," *International Journal for Numerical Methods in Engineering*, Vol. 58, No. 1, 2000, pp. 247–276.

[29]Karamete, B. K., Beall, M. W., and Shephard, M. S., "Triangulation of arbitrary polyhedra to support automatic mesh generators," *International Journal for Numerical Methods in Engineering*, Vol. 49, No. 12, 2000, pp. 167–191.

[30]Young, D. P., Melvin, R. G., Bieterman, M. B., Johnson, F. T., Samant, S. S., and Bussoletti, J. E., "A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics," *Journal of Computational Physics*, Vol. 92, No. 1, 1991, pp. 1–66.

[31]Charlton, E. F. and Powell, K. G., "An octree solution to conservation laws over arbitrary regions (OSCAR)," AIAA Paper 97-198, 1997.

[32]Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Effcient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.

[33]Domel, N. D. and Karman, Jr., S. L., "Splitflow: Progress in 3D CFD with Cartesian Omni-tree Grids for Complex Geometries," AIAA Paper 2000–1006, 2000.

[34]Coirier, W. J. and Powell, K. G., "Solution-adaptive Cartesian cell approach for viscous and inviscid flows," *AIAA Journal*, Vol. 34, No. 5, 1996, pp. 938–945.

[35]Kamatsuchi, T., "Turbulent Flow Simulation around Complex Geometries with Cartesian Grid Method," AIAA Paper 2007–1459, 2007.

[36]Löhner, R., Baum, J. D., Mestreau, E., Sharov, D., Charman, C., and Pelessone, D., "Adaptive embedded unstructured grid methods," *International Journal for Numerical Methods in Engineering*, Vol. 60, 2004, pp. 641–660.

[37]Löhner, R., Appanaboyina, S., and Cebral, J. R., "Comparison of Body-Fitted, Embedded and Immersed Solutions of Low Reynolds-Number Incompressible Flows," AIAA Paper 2007–1296, 2007.

[38]Norton, C. D., Lou, J. Z., and Cwik, T. A., "Status and Directions for the PYRAMID Parallel Unstructured AMR Library," *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, IEEE Computer Society, Washington, DC, USA, 2001, pp. 1224–1231.

[39]Löhner, R., "A parallel advancing front grid generation scheme," *International Journal for Numerical Methods in Engineering*, Vol. 51, No. 6, 2001, pp. 647–661, See also AIAA Paper 2000-1005.

[40]Vidwans, A. and Kallinderis, Y., "Unified Parallel Algorithm for Grid Adaptation on a Multiple-Instruction Multiple-Data Architecture," *AIAA Journal*, Vol. 32, No. 7, 1994, pp. 1800–1807.

[41]Freitag, L. A., Jones, M. T., and Plassmann, P. E., "A Parallel Algorithm for Mesh Smoothing," *SIAM Journal on Scientific Computing*, Vol. 20, No. 6, 1999, pp. 2023–2040.

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

[43]Park, Y. M. and Kwon, O. J., "A parallel unstructured dynamic mesh adaptation algorithm for 3-D unsteady flows," *International Journal for Numerical Methods in Fluids*, Vol. 48, No. 6, 2005, pp. 671–690, See also AIAA Paper 2001–865.

[44]Waltz, J., "Parallel Adaptive Refinement for 3D Unstructured Grids," AIAA Paper 2003-1115, 2003.

[45]Lepage, C. Y. and Habashi, W. G., "Parallel Unstructured Mesh Adaptation on Distributed-Memory Systems," AIAA Paper 2004-2532, 2004.

[46]Cavallo, P. A., Sinha, N., and Feldman, G. M., "Parallel Unstructured Mesh Adaptation Method for Moving Body Applications," *AIAA Journal*, Vol. 43, No. 9, Sept. 2005, pp. 1937–1845.

[47]Alauzet, F., Li, X., Seol, E. S., and Shephard, M. S., "Parallel anisotropic 3D mesh adaptation by mesh modification," *Engineering with Computers*, Vol. 21, No. 3, 2006, pp. 247–258.

[48]Chrisochoides, N., *Numerical Solution of Partial Differential Equations on Parallel Computers*, chap. Parallel Mesh Generation, Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2006.

[49]Bank, R. E. and Smith, R. K., "Mesh Smoothing Using A Posteriori Error Estimates," *SIAM Journal on Numerical Analysis*, Vol. 34, No. 3, 1997, pp. 979–997.

[50]Freitag, L. A. and Ollivier-Gooch, C., "Tetrahedral Mesh Improvement Using Swapping and Smoothing," *International Journal for Numerical Methods in Engineering*, Vol. 40, 1997, pp. 3979–4002.

[51]Mavriplis, D. J., "Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 34, No. 2, 2000, pp. 93–111, See also AIAA Paper 97-0857.

[52]De Cougny, H. L. and Shephard, M. S., "Parallel Refinement and Coarsening of Tetrahedral Meshes," *International Journal for Numerical Methods in Engineering*, Vol. 46, No. 7, 1999, pp. 1101–1125.

[53]Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163, See also AIAA Paper 2001–596.

[54]Schloegel, K., Karypis, G., and Kumar, V., "A Unified Algorithm for Load-balancing Adaptive Scientific Simulations," Supercomputing 2000, 2000.

[55]Frey, P. J. and Alauzet, F., "Anisotropic mesh adaptation for CFD computations," *Computer Methods in Applied Mechanics and Engineering*, , No. 194, Nov. 2005, pp. 5068–5082.

[56]Dompierre, J., Vallet, M.-G., and Guibault, P. L. F., "An analysis of simplex shape measures for anisotropic meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, 2005, pp. 4895–4914.

[57]Shewchuk, J. R., "What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures," *11th International Meshing Roundtable*, Sept. 2002.

[58]Labbé, P., Dompierre, J., , Vallet, M.-G., Guilault, F., and Trépanier, J.-Y., "A universal measure of the conformity of a mesh with respect to an anisotropic metric field," *International Journal for Numerical Methods in Engineering*, Vol. 61, No. 15, 2004, pp. 2675–2695.

[59]Peiro, J., Peraire, J., and Morgan, K., "FELISA Reference Manual and User's Guide, Volume I," University of Wales/Swansea Report CR/821/94, 1994.

[60]Jones, W. T., "GridEx – An Integrated Grid Generation Package for CFD," AIAA Paper 2003–4129, 2003.

[61]Park, M. A., "Three–Dimensional Turbulent RANS Adjoint–Based Error Correction," AIAA Paper 2003–3849, 2003.

[62]Lee-Rausch, E. M., Park, M. A., Jones, W. T., Hammond, D. P., and Nielsen, E. J., "Application of a Parallel Adjoint-Based Error Estimation and Anisotropic Grid Adaptation for Three-Dimensional Aerospace Configurations," AIAA Paper 2005–4842, 2005.

[63]Jones, W. T., Nielsen, E. J., and Park, M. A., "Validation of 3D Adjoint Based Error Estimation and Mesh Adaptation for Sonic Boom Prediction," AIAA Paper 2006–1150, 2006.

[64]Bibb, K. L., Gnoffo, P. A., Park, M. A., and Jones, W. T., "Application of Parallel Gradient-Based Anisotropic Mesh Adaptation for Re-Entry Vehicle Configuratons," AIAA Paper 2006–3579, 2006.

[65]Orme, J. S., Hathaway, R., and Ferguson, M. D., "Initial Flight Test Evaluation of the F-15 ACTIVE Axisymmetric Vectoring Nozzle Performance," AIAA Paper 98–3871, 1998, See also NASA TM-1998-206558.

[66]Aftosmis, M. J., Berger, M. J., and G., A., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000-0808, 2000.

[67]Haimes, R., Giles, M., and Darmofal, D., "Visual3 – A Software Environment for Flow Visualization," Computer Graphics and Flow Visualization in Computational Fluid Dynamics, VKI Lecture Series 10, 1991.

[68]Zienkiewicz, O. C. and Zhu, J. Z., "The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique," *International Journal for Numerical Methods in Engineering*, Vol. 33, No. 7, 1992, pp. 1331–1364.

[69]Zienkiewicz, O. C. and Zhu, J. Z., "Adaptivity and mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 35, No. 4, 1991, pp. 783–810.