Parallel Anisotropic Adaptation for Metric and Direct Error Control

Michael A. Park[a], David L. Darmofal[b]

[a] *Computational AeroSciences Branch, NASA Langley Research Center, Hampton, VA 23681*
[b] *Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 77 Massachusetts Ave., 37-401, Cambridge, MA 02139*

Abstract

Parallel, adaptive algorithms are presented that produce high aspect ratio tetrahedra utilizing local grid operations. In the first algorithm, the properties of the mesh such as element size and shape are explicitly specified by a metric field. Alternatively, a new method is proposed in which the local grid operators are used to directly control an error indicator on the adapted grid and the properties of the mesh evolve implicitly through this error control. The methods are then applied to control interpolation error of a function in one, two, and three dimensions. The direct error control method is shown to be more efficient than the metric-based approach requiring a smaller number of elements to achieve the same interpolation error.

Key words: grid, mesh, interpolation, output

1. Introduction

Obtaining a suitable grid remains one of the most difficult parts of the entire Computational Fluid Dynamics (CFD) simulation process. Grids must resolve flow features to provide adequate control of discretization error and remain small enough to permit reasonable computation times. The AIAA Drag Prediction Workshops[1] are well-documented examples of the difficulty of suitable grid generation even under the watchful eyes of experts. Mavriplis[2] shows that discretization error still dominates other error sources (e.g., turbulence model distance function calculation, thin-layer viscous approximation, level of artificial dissipation) even with extremely fine grids that are computable with today's supercomputers. Chaffin and Pirzadeh[3] 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[4] and design[5] are possible with an automated grid generation and adaptation process. Accurate final adapted solutions can be produced that are independent of coarse initial grids.[6, 7, 8]

Common flow features include discontinuities, shear layers, and boundary layers that require a strongly anisotropic grid for efficient resolution. Mavriplis[9] and Baker[10] provide surveys of grid generation and adaptation techniques. Methods that apply isotropic techniques in a metric mapped space have been successful in two dimensions (2D).[11, 12, 13, 14, 15] 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[11, 16, 10, 17] and can include the effect of local errors on an output quantity.[18, 6]

The use of anisotropic grid metrics has become a standard way to specify a resolution request, but it does have limitations. To date, metric-based adaptation has been applied to linear finite element methods, or equivalent second-order accurate finite volume or finite difference methods. The common approach is to determine the metric by estimating the unresolved second derivatives (Hessian) of a scalar (e.g. Mach number is often used in compressible flows), and then specifying the metric to minimize the error in a linear interpolant of this scalar. Difficulties arise in the extension of metric-based approaches to systems of equations, higher-order methods, and output-based error control:

- For systems of equations, such as the Euler or Navier-Stokes equations, the choice of a scalar such as Mach number to determine the mesh metric is arbitrary. For example, the intersection of metrics from multiple states[13] may be superior.

- For higher-order discretizations, the metric should be based on the lowest-order unresolved derivative (e.g. for a quadratic element, the third derivatives are unresolved). One approach to apply metric methods to higher-order elements is to search for the direction of the largest unresolved derivative[8]. Additional problems arise in that an *a priori* estimate of the regularity of the solution is also required to specify the desired mesh size (this is also true for linear elements).
- For output-based error control, the error estimate does not contain directionality, but is rather a scalar quantity. Venditti and Darmofal[6] developed an output-based anisotropic adaptive method by combining the output-based error estimate to control the volume of the element with the Hessian of the Mach number to control the anisotropy.

In this work, we propose an alternative to metric-based adaptation in which mesh adaptation is performed to directly control an arbitrary measure of the solution error. This framework for direct error control adaptation is presented and compared with metric-based adaptation on a series of test problems.

To facilitate the description of the parallel grid adaptation algorithm, the elemental grid operators and their parallel implementation are detailed. An adaptive method that robustly produces high aspect ratio tetrahedra satisfying a general 3D metric is presented. The method is intended for planar geometries, where more complex domains can be simulated with a cut-cell approach.[19] 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. To alleviate the concessions of the metric-based approach, an adaptive method is introduced that directly controls a generic expression for error without an intermediate element size request. In particular, this proposed approach removes the limitations of previous metric-based methods because the output error can be expressed in the form of this generic error expression.[19] This new method is introduced through application to the control of interpolation error of a known analytic function in 1D, 2D, and 3D. Linear and higher-order solution representations are employed to show the utility of the direct method.

2. Grid Operators

The grid adaptation scheme is constructed from a sequence of nearest-neighbor grid modifications: node movement, tetrahedra swap, tetrahedra split, and tetrahedra collapse. The term tetrahedra is used to emphasize the 3D algorithm, but a subset of these operators also form 2D and 1D operators. Each grid in this sequence is represented as the current set of nodes $x_i \in \mathcal{X}_n$ and tetrahedra $\kappa_j \in \mathcal{T}_n$,

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

where x_i is the position of node i and κ_j are the nodes that element j connects. The edge segments, segments connecting element nodes, are indexed by l. Each of the grid operators G modify the current grid to create a new grid,

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \mathrm{G}(\mathcal{X}_n, \mathcal{T}_n, i, j, l, c(), \bar{c}, m), \tag{2}$$

acting on the grid constitutes by index and improving an objective function $c()$ to a tolerance $\bar{c}$. Each specific grid operator may use a subset of these arguments. The tetrahedra removed and added in an operation are the sets,

$$\hat{\mathcal{T}}_n = \mathcal{T}_n \setminus \mathcal{T}_{n+1}, \tag{3}$$

$$\check{\mathcal{T}}_n = \mathcal{T}_{n+1} \setminus \mathcal{T}_n. \tag{4}$$

2.1. Node Movement

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

$$(\mathcal{X}_{n+1}, \mathcal{T}_n) = \mathrm{Mo}(\mathcal{X}_n, \mathcal{T}_n, i, c(), m), \tag{5}$$

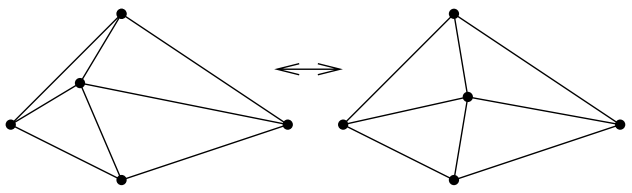


Figure 1: 2D node movement grid operator.

see Fig. 1 for a 2D example with triangles. This is also a popular technique for structured grid methods[10] because of the fixed connectivity. A single node location is adjusted with the neighboring node locations fixed. This method allows explicit constraints on the positivity of tetrahedral volume, i.e.,

$$V(\mathcal{X}_{n+1}, \mathcal{T}_n) > V_{\text{tol}}, \tag{6}$$

where V_{tol} prevents the ambiguity of small element validity in floating-point arithmetic. Global relaxation is not used because explicit constraints cannot be enforced as readily. An objective function $c(\kappa)$ is defined for each tetrahedron incident to a node, $\kappa \ni x_i$. An optimization procedure is invoked to minimize a norm of the incident tetrahedron objective functions,

$$c(x_i) = \left(\sum_{\kappa \ni x_i} c(\kappa)^m \right)^{1/m}, \tag{7}$$

$$x_i|_{n+1} = \operatorname{argmin} c(x_i). \tag{8}$$

If the ∞-norm is selected, smart-Laplacian and quadratic programming optimization schemes[20] are employed. A gradient-free simplex search is used for other norms. Nodes on boundaries are moved in their respective lower dimensional parametrized spaces.

2.2. *Tetrahedra Swapping*

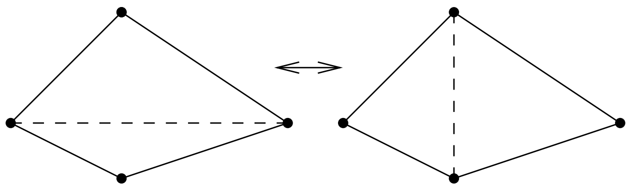


Figure 2: 2D element swap grid operator.

The 3D operations of face and edge swapping[20] are performed,

$$(\mathcal{X}_n, \mathcal{T}_{n+1}) = \text{Sw}(\mathcal{X}_n, \mathcal{T}_n, j, c(), m). \tag{9}$$

A 2D example with triangles is shown in Fig. 2, where this is a simple replacement of two triangles for two triangles. In 3D, this operator is much more complicated. The element κ_j and its neighbors are examined, and the configuration that reduces a norm of the involved tetrahedron objective functions to the greatest degree is chosen,

$$\check{\mathcal{T}}_n = \operatorname{argmin} \left(\sum_{\kappa \in \check{\mathcal{T}}_n} c(\kappa)^m \right)^{1/m}, \tag{10}$$

if it has positive volume tetrahedra,

$$V(\mathcal{X}_n, \mathcal{T}_{n+1}) > V_{\text{tol}}. \tag{11}$$

If there is no swapped configuration with a lower norm, the current configuration is retained, which ensures

$$\left(\sum_{\kappa\in\check{\mathcal{T}}_n} c(\kappa)^m\right)^{1/m} \leq \left(\sum_{\kappa\in\hat{\mathcal{T}}_n} c(\kappa)^m\right)^{1/m}. \tag{12}$$

The 3D configurations that are evaluated by the argmin in Eq. (10) are face swapping[20] and edge swapping.[20] Face swapping is replacing two tetrahedra that share three nodes with three tetrahedra. Edge swapping is replacing all of the tetrahedra that share two nodes with a new set of tetrahedra that fill the same volume. Edge swaps are implemented for three to seven tetrahedra surrounding an edge. The number of possible configurations that can result from an edge swap grows rapidly with the number of tetrahedra surrounding an edge. Fortunately, there is a smaller set of canonical configurations for each of these swap operations. Freitag and Ollivier-Gooch[20] describe these canonical configurations and how to reduce the cost of evaluating the norm of the new configurations. The boundary triangles are also swapped as a result of tetrahedra swapping.

2.3. Tetrahedra Split and Collapse

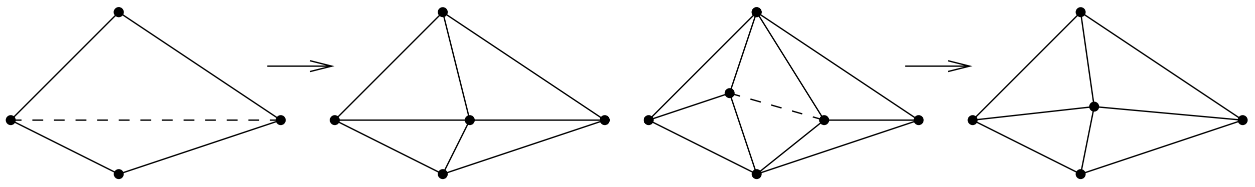

(a) Triangle split. (b) Triangle collapse.

Figure 3: 2D split and collapse operators.

The split and collapse operations modify the density of the grid tetrahedra as well as contribute to obtaining the desired anisotropy. There are many possible splitting stencils.[21, 22] In this work, the split operator inserts a single new node at the center of the segment l connecting two nodes. All the tetrahedra that share l are split into two tetrahedra to include this new node. Segments that are on a boundary also result in split boundary triangles. The tetrahedra split operator is used in two forms,

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \text{Sp}(\mathcal{X}_n, \mathcal{T}_n, l, L()), \tag{13}$$

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \text{Sp}(\mathcal{X}_n, \mathcal{T}_n, j, c(), m). \tag{14}$$

Equation (13) splits an edge if its length function $L(l) > 1$. Equation (14) splits the edge $l \in \kappa_j$ of element j that produces the largest decrease in $c()$,

$$l = \underset{l\in\kappa_j}{\text{argmin}} \frac{\left(\sum_{\kappa\in\hat{\mathcal{T}}_n} c(\kappa)^m\right)^{1/m}}{\left(\sum_{\kappa\in\check{\mathcal{T}}_n} c(\kappa)^m\right)^{1/m}}. \tag{15}$$

In absolute precision arithmetic, 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. In floating-point arithmetic, this may not be true due to floating-point errors. An additional check is placed on the computed volume after the split and the split is not permitted unless,

$$V(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) > V_{\text{tol}}. \tag{16}$$

For a collapse, all the tetrahedra that share l 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 remaining node is placed at the center of the removed segment. The collapse operator also has two forms,

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \text{Co}(\mathcal{X}_n, \mathcal{T}_n, l, L()), \tag{17}$$

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \mathrm{Co}(\mathcal{X}_n, \mathcal{T}_n, j, c(), \bar{c}, m). \tag{18}$$

The collapse operation is only permitted if the new configuration results in positive volume tetrahedra,

$$V(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) > V_{\mathrm{tol}}. \tag{19}$$

Equation (17) collapses an edge if $L(l) < 1.0 \; \forall \; l \in \check{\mathcal{T}}_n$. Equation (18) collapses an edge of a tetrahedra if $c(\kappa_j) < \bar{c} \; \forall \; \kappa_j \in \check{\mathcal{T}}_n$.

3. 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.

3.1. 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 algorithm. The domain decomposition scheme is identical to the method utilized by FUN3D[23].

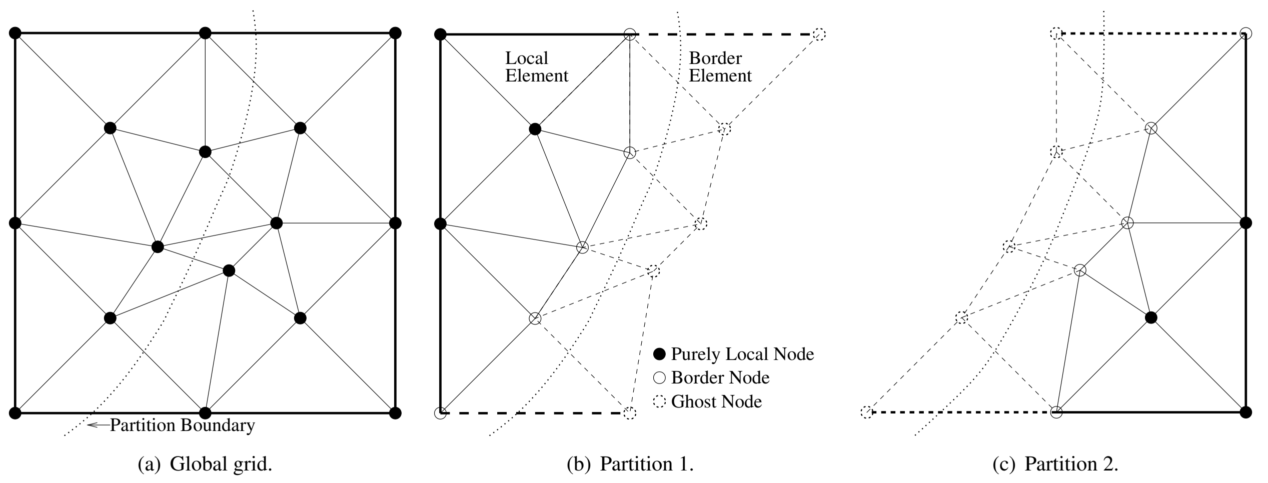


Figure 4: Two partition domain decomposition 2D example.

A two partition example is illustrated in Fig. 4 for 2D. A node is denoted local 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 to which 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.

3.2. *Partition Coloring*

To increase parallel efficiency, groups of unconnected partitions are allowed to simultaneously modify border element connectivities and node positions. Only allowing execution on one partition at a time results in a sequential algorithm, because the other partitions are idle until they have an opportunity to process their border region. The amount of time that processors remain idle during adaptation can be reduced by allowing concurrent execution through partition coloring.

The sets of unconnected partition groups are calculated with a greedy coloring scheme[24] where no two adjoining partitions share a color. All the partitions in a single color can then modify border regions concurrently with current ghost information. With coloring, parallel communication is only required between the processing of different colors, which are less than or equal to the number of partitions.

3.3. *Connectivity Changes*

The connectivity of the tetrahedra change as a result of swapping Sw, splitting Sp, and collapsing Co. In other parallel adaptation implementations,[22, 25, 26, 27] 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 collapse operator is currently only allowed 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 and an increase in code complexity.

Initially, connectivity changes are only performed on local tetrahedra (solid lines, Fig. 4(b) and (c)). This is an efficient parallel operation because all partitions perform connectivity changes simultaneously. Updates of the border tetrahedra region (dashed lines, Fig. 4(b) and (c)) are performed one partition color 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 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 costs 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 removed.

3.4. *Node Movement*

The nearest-neighbor information must be current when smoothing nodes Mo to prevent the creation of invalid tetrahedra. Freitag[28] and Löhner[29] address the issues for maintaining current nearest-neighbors while node smoothing on a shared-memory architecture. In other distributed memory implementations,[25, 27, 26] 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 color at a time. The ghost nodes are updated after each color to ensure that nearest-neighbor information is current.

Ghost node parameters (position, face parameters, and edge 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 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 translation is performed in $O(\log_2 n)$ time with a binary search of the global indexes of local nodes stored in a sorted list. The translated 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.

3.5. 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 if it is available.

When this list of unused global indexes is exhausted, the partition creates a new global index by independently incrementing its copy of the total number of global indexes. This allows two or more partitions to obtain the same global index. These duplicate global indexes are made unique by shifting them. After shifting the newly created global node indexes, the true count of global indexes is shared amongst partitions.

```
              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 5: An example of creating new global indexes on four partitions with 100 original global indexes, before and after shift.

Figure 5 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.

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 and unique global indexes.

3.6. Load Balancing

Load balancing is employed to maintain parallel efficiency. The parallel, graph-partitioning tool ParMETIS[30] 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.

4. Metric-Based Adaptation

A metric tensor 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.[16, 17, 13, 31] The symmetric positive definite matrix M in 3D has the diagonal decomposition

$$M = X \begin{bmatrix} \Lambda_1 & & \\ & \Lambda_2 & \\ & & \Lambda_3 \end{bmatrix} X^T = X \Lambda X^T. \tag{20}$$

The eigenvectors X define an orthonormal basis with length specifications h_i in this basis,

$$\Lambda = \begin{bmatrix} \left(\frac{1}{h_1}\right)^2 & & \\ & \left(\frac{1}{h_2}\right)^2 & \\ & & \left(\frac{1}{h_3}\right)^2 \end{bmatrix}. \tag{21}$$

This metric can be interpreted as an ellipsoid with major and minor axes of direction X_i and length h_i.[16] The linear mapping J is employed to map a vector in physical space $\vec{x} = [x\ y\ z]^T$ to a vector in transformed space $\vec{x}' = [x'\ y'\ z']^T$ where a triangle or tetrahedron becomes equilateral with unit length edges; that is,

$$\vec{x}' = J\vec{x}. \tag{22}$$

The metric tensor M is related to J by

$$M = J^T J, \tag{23}$$

$$J = \Lambda^{\frac{1}{2}} X^T. \tag{24}$$

If $\vec{x}$ describes an edge in physical space, the length L in mapped space is

$$L = \sqrt{\vec{x}'^T \vec{x}'}. \tag{25}$$

Employing Eq. (22), this expression of length becomes

$$L = \sqrt{(J\vec{x})^T (J\vec{x})}, \tag{26}$$

$$L = \sqrt{\vec{x}^T M \vec{x}}. \tag{27}$$

Equation (27) is employed to directly compute edge lengths in the specified metric. Equation (22) is applied to the physical coordinates of tetrahedron nodes to map them to the transformed space before evaluating a quality of the tetrahedron's shape.

The resolution request is stored as M during adaptation instead of storing J directly due to these benefits:

- It is compact; only 6 entries of the 3D symmetric M are stored.
- Multiple M can be readily interpolated and intersected.[13, 31]
- Lengths in the specified metric can be computed efficiently with Eq. (27).

A slight computational cost is incurred computing J from M when J is needed to obtain a metric transformed tetrahedron. This cost is a 3×3 symmetric positive definite matrix diagonalization and Eq. (24). This cost is minimized by tridiagonalizing M with a single rotation and applying an iterative method to find the eigendecomposition of the tridiagonal matrix.

4.1. Shape Measure

In 3D, considering only edge length as a shape measure can result in degenerate tetrahedra, because a nearly-zero-volume ‘sliver’ tetrahedra can be constructed without a short edge. Shape measures [32, 33] 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.[34] A norm of the mesh conformity to a specified multidimensional grid resolution[35] 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[32] is

$$\eta(\kappa) = \frac{L_{01}^2 + L_{02}^2 + L_{03}^2 + L_{12}^2 + L_{13}^2 + L_{23}^2}{12(3V)^{2/3}}, \tag{28}$$

where V is the volume of the tetrahedron and L are the six lengths of vectors defined between nodes 0–3 of the tetrahedron. It is in the range of $[1, \infty]$, where an isotropic tetrahedra is 1 and a degenerate tetrahedra is ∞. The mean ratio is efficient to evaluate and a well-behaved continuously differentiable function,[34] which makes it very suitable for gradient-based optimization. It also has a connection to metric conformity without size specification.[35] 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[34] for examples.

4.2. Metric Adaptation Algorithm

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

- 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 than 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.

The metric conformity multi-dimensional (`MetricConformityMD`) algorithm is:

- Sort the tetrahedra by decreasing η in the specified metric. Consider the elements one-by-one, swap them with their neighbors to minimize the maximum η of the involved tetrahedra,

  $$(\mathcal{X}_n, \mathcal{T}_{n+1}) = \mathrm{Sw}(\mathcal{X}_n, \mathcal{T}_n, j, \eta(), m = \infty)\ \forall\ \kappa_j \in \mathcal{T}_n \text{ from largest } \eta(\kappa_j) \text{ to smallest } \eta(\kappa_j). \tag{29}$$

  If an element is changed remove it from the queue.

- Perform the node movement operation on all nodes from largest $\eta(x_i)$ over incident tetrahedra to smallest,

  $$(\mathcal{X}_{n+1}, \mathcal{T}_n) = \mathrm{Mo}(\mathcal{X}_n, \mathcal{T}_n, i, \eta(), m = \infty)\ \forall\ x_i \in \mathcal{T}_n \text{ from largest } \eta(x_i) \text{ to smallest } \eta(x_i). \tag{30}$$

  Smart-Laplacian and quadratic programming optimization schemes[20] 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 $L(l) < 0.3$.

  $$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \mathrm{Co}(\mathcal{X}_n, \mathcal{T}_n, l, L())\ \forall\ l \in \mathcal{T}_n : l < 0.3 \text{ from smallest } L \text{ to largest } L. \tag{31}$$

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

$$(\mathcal{X}_{n+1}, \mathcal{T}_{n+1}) = \mathrm{Sp}(\mathcal{X}_n, \mathcal{T}_n, l, L()) \; \forall \; l \in \mathcal{T}_n : l > 1.0 \text{ from largest } L \text{ to smallest } L. \tag{32}$$

  A element newly created from a Sp operation is not added into the split queue.

- Repeat these steps until $L(l) < 1.0 \; \forall \; l \in \mathcal{T}_n$.

4.3. Analytic Metric Adaptation

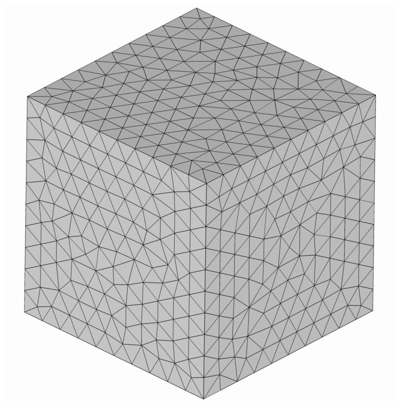
The volume grid of a unit cube domain $[0, 1]\times[0, 1]\times[0, 1]$ is adapted to an analytic metric field. The original grid, Fig. 6(a), is created by the FELISA grid generator[36] through the GridEx framework[37] with an isotropic spacing of 0.1. An anisotropic metric

$$[0.1\hat{\imath}, h + (0.1 - h)\frac{|y - 0.5|}{0.5}\hat{\jmath}, 0.1\hat{k}] \tag{33}$$

is used for adaptation. To reach this high anisotropy, the metric in the $\hat{\jmath}$-direction is reduced in a number of steps

$$h = 0.1^s, \tag{34}$$

where s is the step $s = 1, \ldots, 4$. The grid is adapted to this intermediate metric at each step. The volume grid is clustered near the $y = 0.5$ plane. This clustering due to the anisotropic metric is evident on the $x = 0$ and $z = 0$ visible surfaces of the cube, Fig. 6(b). The adaptation history of edge length is shown as a series of histograms for the step when $h = 0.001$ is reduced to $h = 0.0001$, Fig. 7. The x-axis of each histogram is $\log_{10}(L)$. At the completion of each adaptation step, all edges are shorter than one in the metric and very few are shorter than $\log_{10}(0.3) \approx -0.52$.


(a) FELISA grid.

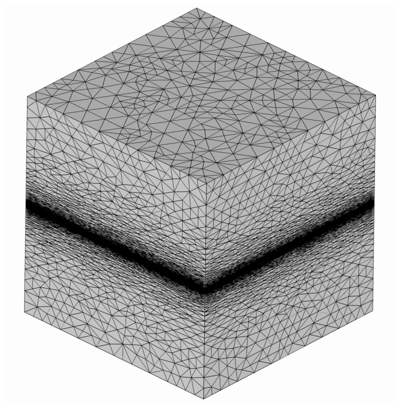
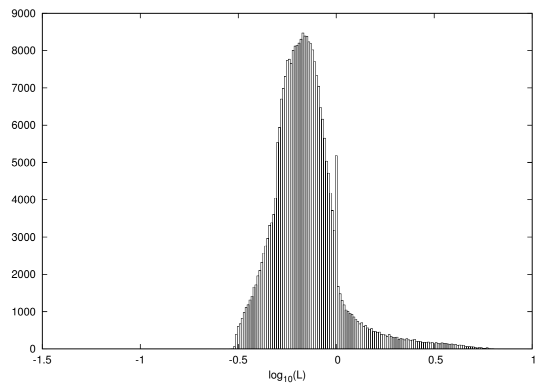
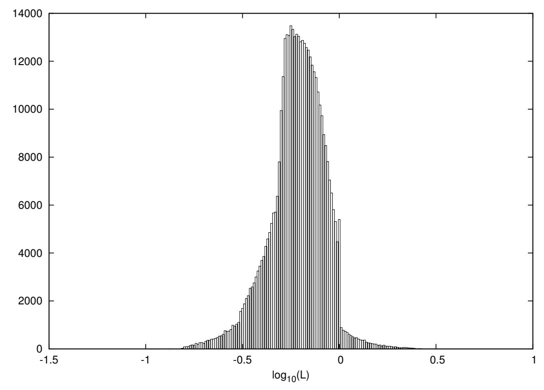

(b) All edges smaller than Eq. (33) metric, $h = 0.0001$.

Figure 6: Isometric views of a cube.

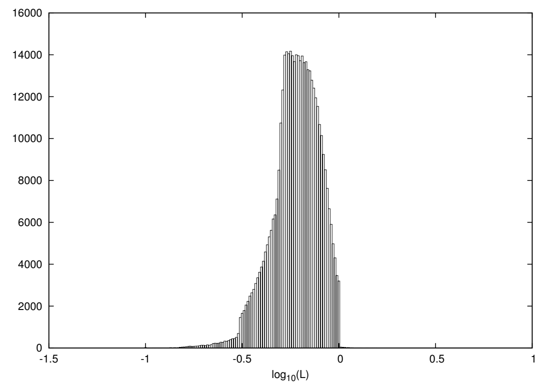
Anisotropic scaling of the figures aids illustration of the final grid corresponding to this simple metric field. The $z = 0$ face of the Fig. 6(b) cubic grid is shown in Fig. 8. The y-axis is progressively scaled in the series of subfigures, but the grid itself remains fixed. Figure 8(a) shows the anisotropically 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 8(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.



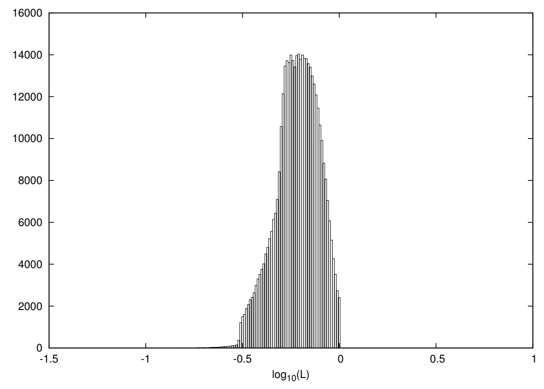
(a) Initial edge lengths.



(b) After first pass of `MetricConformityMD`.

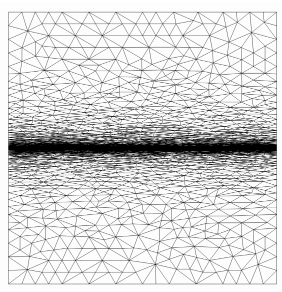


(c) After second pass of `MetricConformityMD`.

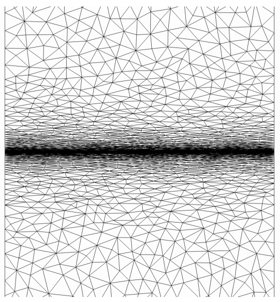


(d) After third pass of `MetricConformityMD`.

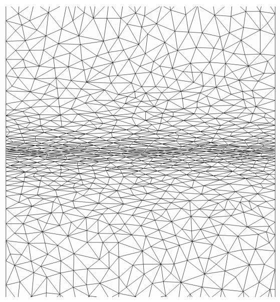
Figure 7: Histogram of the log of edge length in metric $\log_{10}(L)$ for a step of the adapted cube when $h = 0.001$ is reduced to $h = 0.0001$.



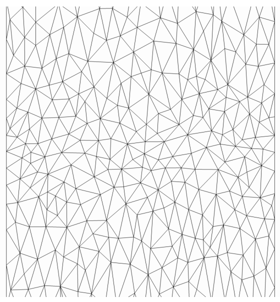
(a) 1–1 view.



(b) 1–10 view.



(c) 1–100 view.



(d) 1–1000 view.

Figure 8: Face of a cube ($z = 0$) adapted to Eq. (33) metric, $h = 0.0001$.

5. Direct Adaptive Control of Error

The total error in the domain Ω is defined as the norm,

$$e(\Omega) = \left(\int_{\Omega} |\epsilon|^m \, \mathrm{d}\Omega \right)^{1/m}, \tag{35}$$

where ϵ is an expression for error to be controlled. Output error can be cast in this form[19] as well as interpolation error, which is used as an illustrative example. The error for each element κ, which completely discretizes Ω is

$$e(\kappa) = \left(\int_{\kappa} |\epsilon|^m \, \mathrm{d}\kappa \right)^{1/m}. \tag{36}$$

The total interpolation error for the domain is the norm of the elemental errors,

$$e(\Omega) = \left(\sum_{\kappa \in \Omega} e(\kappa)^m \right)^{1/m}. \tag{37}$$

Following Zienkiewicz and Zhu,[38] the goal is to equidistribute this error estimate. An equal amount of error for each element is

$$\mathrm{tol}_\kappa = \left(\frac{(\mathrm{tol}_\Omega)^m}{N} \right)^{1/m}, \tag{38}$$

where tol_Ω is the requested total interpolation error and N is the number of elements. The grid operators are applied in sequence to redistribute the error to bring the elemental errors as close as possible to this tol_κ.

5.1. 1D Interpolation Error

The interpolation error is defined as,

$$\epsilon = (\bar{u} - u), \tag{39}$$

where the exact analytical function u and interpolant $\bar{u}$ are defined in the domain Ω. The $\bar{u}$ is defined as a p-order polynomial in each element κ, which completely discretizes Ω. The u is sampled at each node of $\bar{u}$ to form $\bar{u}$ in each element. The elemental interpolation error is

$$e(\kappa) = \left(\int_{\kappa} |(\bar{u} - u)|^2 \, \mathrm{d}\kappa \right)^{1/2}, \tag{40}$$

for $m = 2$. he direct error control in 1D (`ErrorControl1D`(e)) algorithm is defined as:

- Repeat until $e(\kappa) \leq \mathrm{tol}_\kappa \ \forall \ \kappa \in \Omega$:
  - Evaluate Eq. (40) for all elements in the grid; split the element in half with the largest value of $e(\kappa)$ if $e(\kappa) > \mathrm{tol}_\kappa$,

$$\mathrm{Sp}(\mathcal{X}_n, \mathcal{T}_n, j, e(), m = 2) \ \forall \ j \in \mathcal{T}_n : e(\kappa_j) > \mathrm{tol}_\kappa \text{ from largest } e(\kappa_j) \text{ to smallest } e(\kappa_j). \tag{41}$$

  - Evaluate Eq. (40) for all elements in the grid; merge the element with the smallest value of e_κ with a neighbor,

$$\mathrm{Co}(\mathcal{X}_n, \mathcal{T}_n, j, e(), \mathrm{tol}_\kappa, m = 2) \ \forall \ l \in \mathcal{T}_n : e(\kappa_j) < \mathrm{tol}_\kappa \text{ from smallest } e(\kappa_j) \text{ to largest } e(\kappa_j). \tag{42}$$

- Perform two sweeps of node position optimization; each sweep progresses from the node of the element with the largest $e(x_i)$ to the nodes of element with the smallest $e(x_i)$,

$$\mathrm{Mo}(\mathcal{X}_n, \mathcal{T}_n, i, e(), m = 2) \ \forall \ i \in \mathcal{X}_n \text{ from largest } e(x_i) \text{ to smallest } e(x_i). \tag{43}$$

- Repeat these steps until $e(\Omega) \leq \mathrm{tol}_\Omega$.

An exact function u is defined in the 1D region $[0, 1]$,

$$u = \tanh(50(x - 0.4)). \tag{44}$$

For a metric-based approach, a requested mesh size is established from a scaled interpolation estimate[11] commonly used for anisotropic adaptation,

$$h = \min\left(0.5, \frac{1}{\sqrt{100|u_{xx}|}}\right). \tag{45}$$

The min function is used to set a maximum element size of 0.5 when u_{xx} approaches zero. The scaling factor 100 is chosen arbitrarily to produce a reasonably-sized mesh of 32 elements. A metric-based grid is generated that satisfies this h-request with a curve discretization method.[16] This metric-based grid is shown in Fig. 9(a). Element boundaries are marked with a +-symbol. The metric-based grid has a total interpolation error of $e(\Omega) = 1.12 \times 10^{-3}$ with 32 $p = 1$ elements.

The direct adaptive method is invoked with a error tolerance equal to the computed interpolation error of the metric-based grid. The final directly adapted grid is shown in Fig. 9(b). It has 27 $p = 1$ elements and a total error of $e(\Omega) = 1.06 \times 10^{-3}$, which is slightly lower than the metric-based grid, 1.12×10^{-3}, which has more elements. The metric-based, Fig. 9(a), and direct error control, Fig. 9(b), grids are very similar for this 1D example with $p = 1$ elements. The error estimate[11] is sharp for this simple 1D case. The clustering of elements in the high-curvature regions of the tanh function is evident for both methods. The direct adaption method produced a grid with slightly lower interpolation error and a reduction of the number of elements.

The metric-based approach is repeated for a sequence of α with the specified

$$h = \min\left(0.5, \frac{1}{\sqrt{\alpha|u_{xx}|}}\right). \tag{46}$$

The 2-norm of domain interpolation error for this sequence grids is shown as the squares in Fig. 10 as a function of effective $h = 1/N$. The direct interpolation adaption method is invoked multiple times with a tol_Ω equal to the $e(\Omega)$ of the metric-based approach. The $e(\Omega)$ for these directly adapted grids is shown as circles in Fig. 10. Both methods show second-order convergence of the interpolation norm. The direct approach is more efficient in terms of error for an effective h over the entire sequence of grids. The direct method has 0.7 the error of the metric approach for the same number of elements. The algorithm was also studied without the Mo operator[19] to determine its contribution to the overall efficiency of the method.

An advantage of the direct adaption method is that the order of the polynomial representation can be increased without modifying the adaptation algorithm. The h specification based on interpolation estimates has been formulated for $p = 1$ elements. In 1D, an h can be specified for higher degree polynomials, but this task becomes more complicated for multiple dimensions.[8] The 2-norm of domain interpolation error for a sequence grids is shown in Fig. 11 as a function of effective $h = 1/N$ for various p. Triangles are provided with slopes listed on the left of each triangle. These indicate that the error is converging at an optimal rate of $p + 1$.

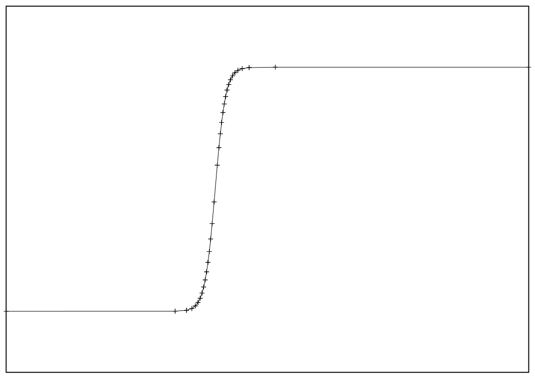
In the previous results, knowledge of the exact solution u was utilized to calculate the error. In practice, the exact solution is not known and the error must be estimated. To simulate this situation, an approximate solution $\tilde{u}$ is constructed of p-order polynomials on a set of discrete elements κ_0. This approximate solution on each element is constructed by minimizing

$$\left(\int_{\kappa_0} |\tilde{u} - u|^2 \, \mathrm{d}\kappa_0\right)^{1/2} \tag{47}$$

independently in each element. The left column of Fig. 12 shows this discontinuous fit for a series of adapted 1D grids.

To estimate the leading order terms of the exact interpolation error, a $p + 1$-order solution $\hat{u}$ is reconstructed from $\tilde{u}$ on κ_0. A C^0 $p + 1$-order function $\hat{u}$ is formed from the discontinuous $\tilde{u}$ by minimizing

$$\left(\int_{\Omega} |\hat{u} - \tilde{u}|^2 \, \mathrm{d}\Omega\right)^{1/2} \tag{48}$$


(a) Metric-based, 32 $p = 1$ elements, $e(\Omega) = 1.12 \times 10^{-3}$.

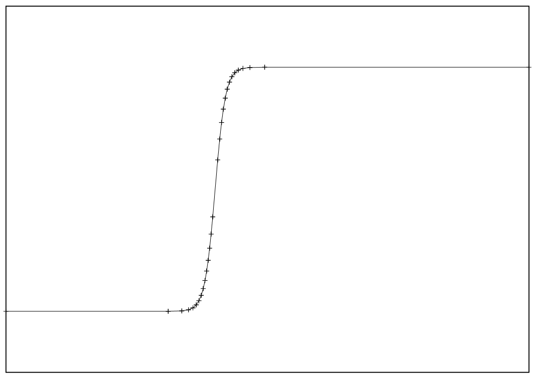

(b) Direct error control, 27 $p = 1$ elements, $\mathrm{tol}_\Omega = 1.12 \times 10^{-3}$, $e(\Omega) = 1.06 \times 10^{-3}$.

Figure 9: Grids to control the interpolation error of tanh with $p = 1$ basis.

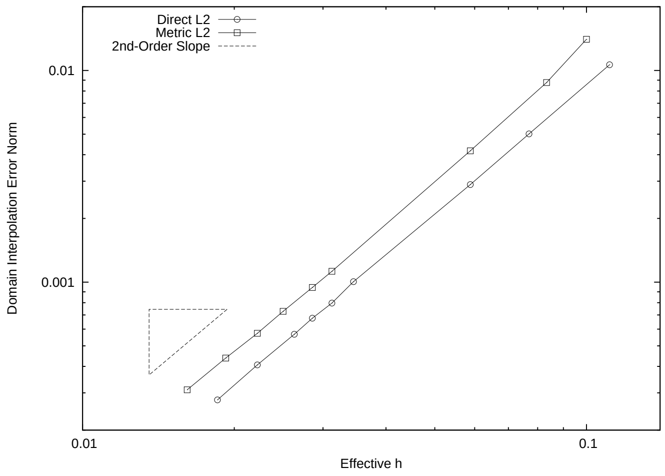


Figure 10: Convergence of the interpolation error in the 2-norm for the metric and direct adaptation method on the tanh function with $p = 1$ elements.

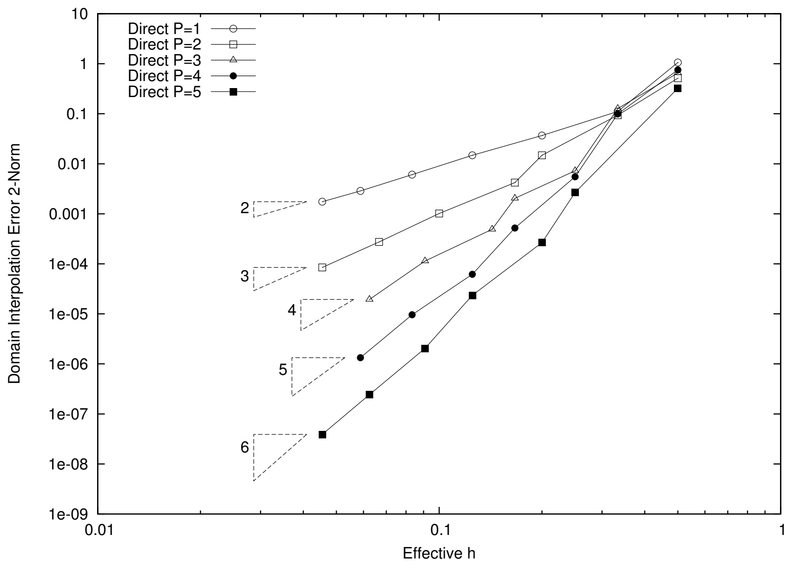


Figure 11: Convergence of the interpolation error in the 2-norm for the direct adaptation method on the tanh function with $p = 1, 2, 3, 4, 5$ elements.

over the entire domain. The formation of $\hat{u}$ is not local; an inexpensive mass matrix inversion is computed. A C^0 $\hat{u}$ is required to prevent the adaptation process from introducing zero width elements at discontinuous function jumps. This $\hat{u}$ is shown in the right column of Fig. 12. Over- and under-shoots are present in $\hat{u}$. The domain discretization is adapted into elements κ. An elemental interpolation error of the form,

$$e'(\kappa) = \left(\int_\kappa |(\bar{u} - \hat{u})|^2 \, \mathrm{d}\kappa \right)^{1/2}, \tag{49}$$

is utilized, where $\bar{u}$ is a p-th order polynomial representation of the function $\hat{u}$ in each element. The function $\bar{u}$ is directly interpolated from $\hat{u}$. A step of the adaptive procedure follows;

- Given the current grid, construct a discontinuous p basis $\tilde{u}$ with Eq. (47) to simulate the solution to a PDE.
- Reconstruct a continuous $p + 1$ basis $\hat{u}$ with a 2-norm fit of $\tilde{u}$ over the domain, Eq. (48).
- Exit adaptive procedure if $e'(\Omega) \leq \mathrm{tol}_\Omega$.
- Apply `ErrorControl1D`(e')

These steps are repeated to form the adaptive procedure.

Each step of the adaptive procedure is shown as a row in Fig. 12. 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 $p = 5$ basis on the initial grid. The upper right subfigure shows the first adapted grid interpolating the $p = 6$ continuous basis $\hat{u}$ on κ_0. 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. The final grid is coarsened away from $x = 0.4$ after the Gibbs phenomenon is reduced.

5.2. *2D Interpolation Error*

In this section, the direct error control approach is extended to 2D by considering adaptation to minimize the 2-norm of interpolation error of a known function. The direct adaptation consists of four modes. These are element swapping, node movement, element collapse, and element split. The direct error control multi-dimensional (`ErrorControlMD`) algorithm is:

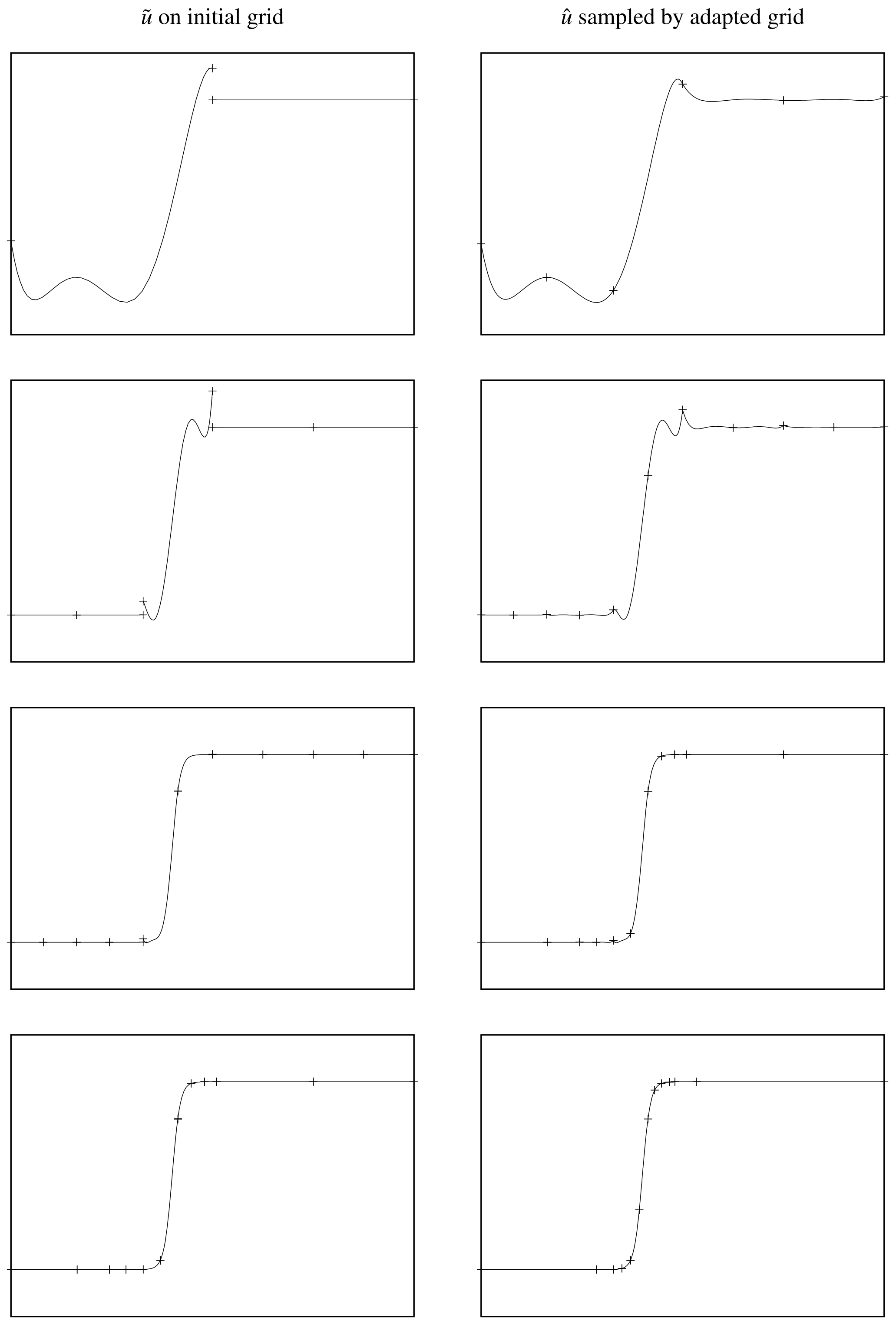


Figure 12: Four 1D adaptation cycles to the interpolation error of tanh with $p = 5$ basis (one cycle per row, approximate solution in left column, reconstructed solution in right column)

- Sort the elements from largest e_κ to smallest. Perform the swap operation on all element with $e(\kappa) > \mathrm{tol}_\kappa$,

$$\mathrm{Sw}(\mathcal{X}_n, \mathcal{T}_n, j, e(), m = 2)\ \forall\ j \in \mathcal{T}_n : e(\kappa_j) > \mathrm{tol}_\kappa \text{ from largest } e(\kappa_j) \text{ to smallest } e(\kappa_j). \tag{50}$$

- Perform the node movement operation on all nodes from largest,

$$e_V(x_i) = \left(\sum_{\kappa \ni x_i} \frac{e(\kappa)^m}{V(\kappa)} \right)^{1/m}, \tag{51}$$

  over incident elements to smallest,

$$\mathrm{Mo}(\mathcal{X}_n, \mathcal{T}_n, i, e_V, m = 2)\ \forall\ i \in \mathcal{X}_n \text{ from largest } e_V(x_i) \text{ to smallest } e_V(x_i). \tag{52}$$

  The volume (area in 2D) of the elements $V(\kappa)$ is included in the denominator to penalize nearly degenerate elements. A simplex search is performed on the prioritized node list to optimize the inverse volume weighted norm.

- The elements with low error are examined for potential collapse,

$$\mathrm{Co}(\mathcal{X}_n, \mathcal{T}_n, j, e(), \mathrm{tol}_\kappa, m = 2)\ \forall\ j \in \mathcal{T}_n : e(\kappa_j) < \mathrm{tol}_\kappa \text{ from smallest } e(\kappa_j) \text{ to largest } e(\kappa_j). \tag{53}$$

- Sort the elements from largest $e(\kappa)$ to smallest,

$$\mathrm{Sp}(\mathcal{X}_n, \mathcal{T}_n, j, e(), m = 2)\ \forall\ j \in \mathcal{T}_n : e(\kappa_j) > \mathrm{tol}_\kappa \text{ from largest } e(\kappa_j) \text{ to smallest } e(\kappa_j). \tag{54}$$

- Repeat these steps until $e(\Omega) < \mathrm{tol}_\Omega$.

An analytic function is defined,

$$u = (2.0 + \sin(10.0x)) \exp(-10.0(y - 0.5)^2), \tag{55}$$

and shown in Fig. 13 for a domain sized $[-1.5, 1.5] \times [0, 1]$. Direct interpolation error control and metric-based algorithms are employed with an initial grid shown in Fig. 14. The target metric M is set to $10|H|$ for the entire

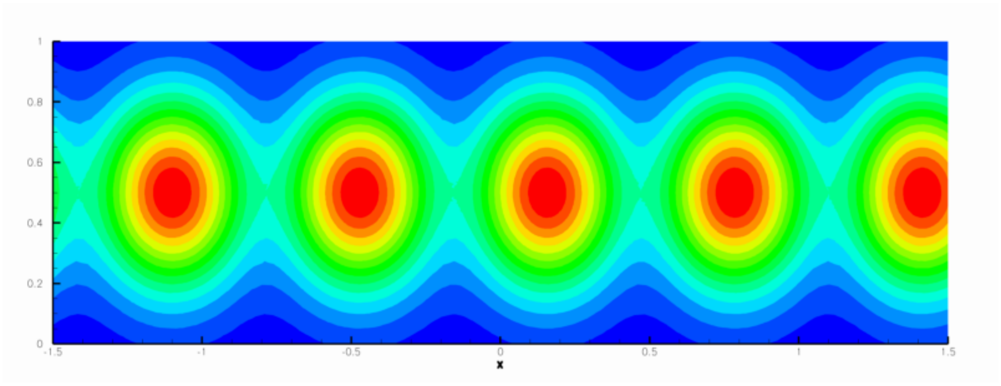


Figure 13: Analytic function, Eq. (55).

adaptation process, where the Hessian,

$$H = \begin{bmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{bmatrix}, \tag{56}$$

is computed analytically. The metric-based approach employs the `MetricConformityMD` algorithm. The metric M is interpolated from a grid with quadratic elements that is frozen at the start of each adaptation cycle. The direct

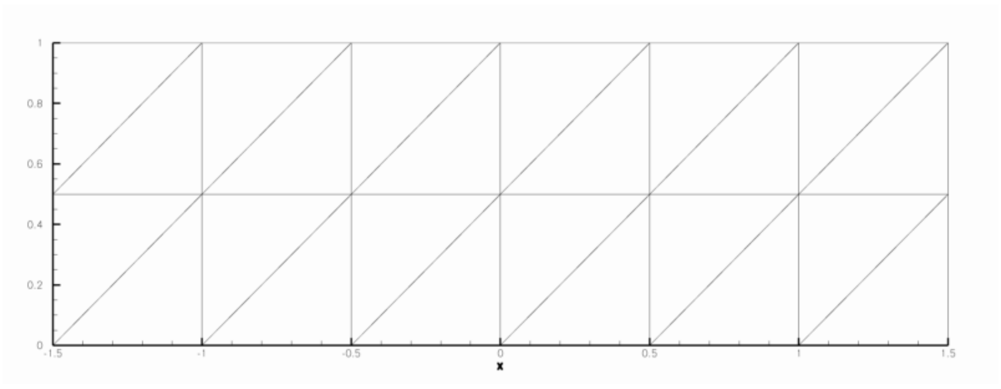


Figure 14: Initial grid of the domain $[-1.5, 1.5] \times [0, 1]$.

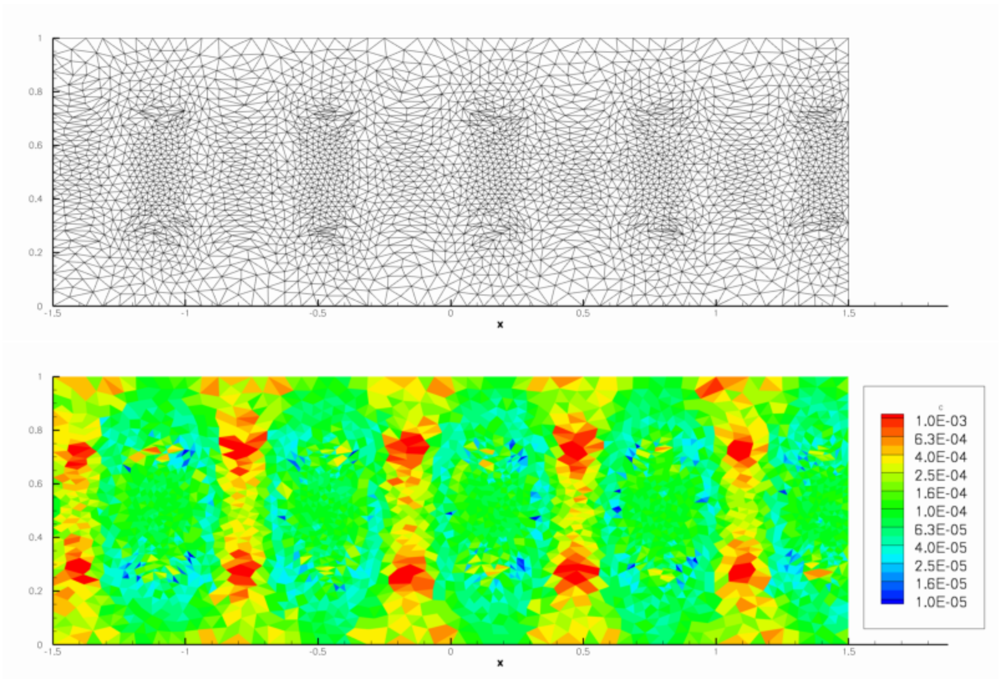


Figure 15: Metric-adapted grid and element interpolation error in 2-norm (color scale is logarithmic).

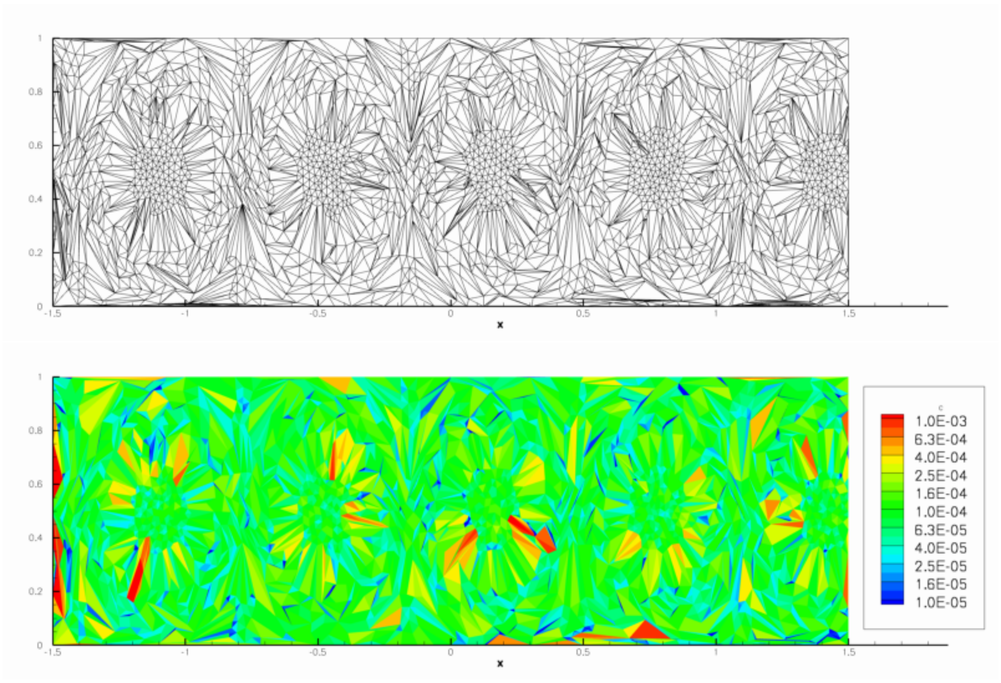


Figure 16: Direct interpolation error adapted grid and element interpolation error in 2-norm (color scale is logarithmic).

approach is terminated when it reached the error level of the metric-adapted grid. The direct approach utilizes the reconstructed solution $\hat{u}$ described in Section 5.1 to define the interpolation error $e(\kappa)$.

The final metric-adapted grid is shown in Fig. 15 with the 2-norm interpolation error for each element. The direct-adapted grid is shown in Fig. 16 with the 2-norm interpolation error for each element. The direct-adapted element error is scattered with the largest values on the border of the domain and radially around each peak in the function. The metric-adapted element error is clustered in the saddle-shaped troughs of the function. The metric-adapted grid is much more regular than the direct-adapted grid. Both methods produce similar grids at the peaks of the functions, where the grid is isotropic. The direct-adapted grid has greater anisotropy radiating from the peaks and along the troughs of the function. The direct approach uses approximately 70% the triangles of the metric-based approach for the same error norm.

The metric-based approach is intended to equidistribute an *a priori* estimate of the interpolation error in the 2-norm.[16] The direct approach may be more efficient in terms of number of elements because it is employing an actual calculation of the error. The direct-adapted grids may be less regular because computed error is a more difficult function to optimize than metric conformity.

The metric-based approach is repeated for a sequence of α with the specified

$$M = \alpha|H|. \tag{57}$$

The 2-norm of domain interpolation error for this sequence grids is shown as the squares in Fig. 17 as a function of the effective $h = 1/\sqrt{N}$. The $e(\Omega)$ for the directly adapted grids is shown as circles in Fig. 17, which are produced by halving the initial computed error to set tol_Ω for each step. Both methods show second-order convergence of the interpolation norm in terms of an effective h. The direct approach is more efficient in terms of error for a given number of elements for the entire sequence of grids. The improvement in efficiency is approximately a constant factor for this entire sequence.

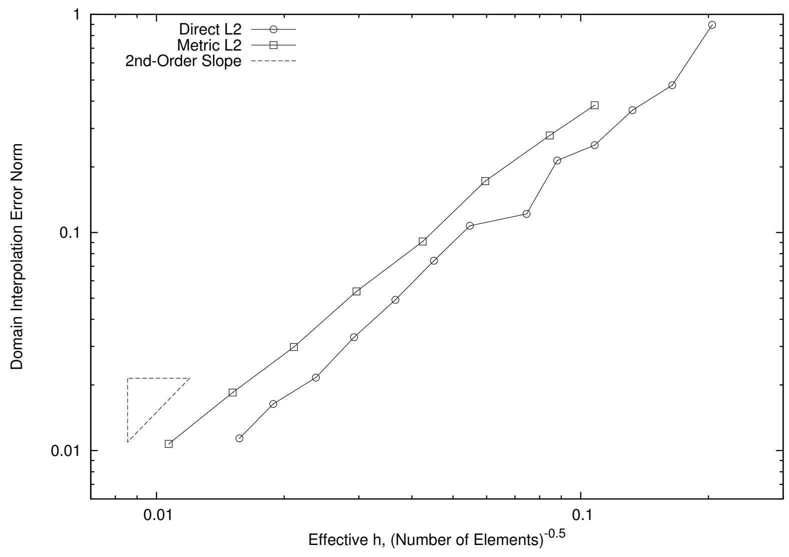


Figure 17: Convergence of the interpolation error in the 2-norm for the 2D metric and direct adaptation method.

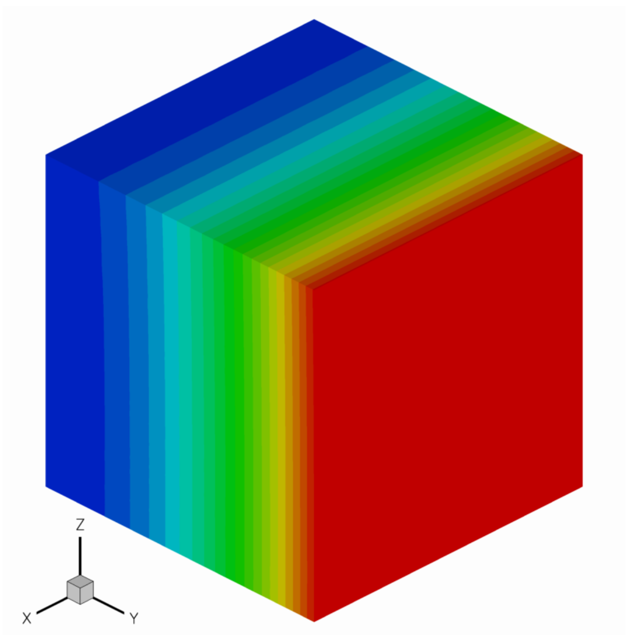


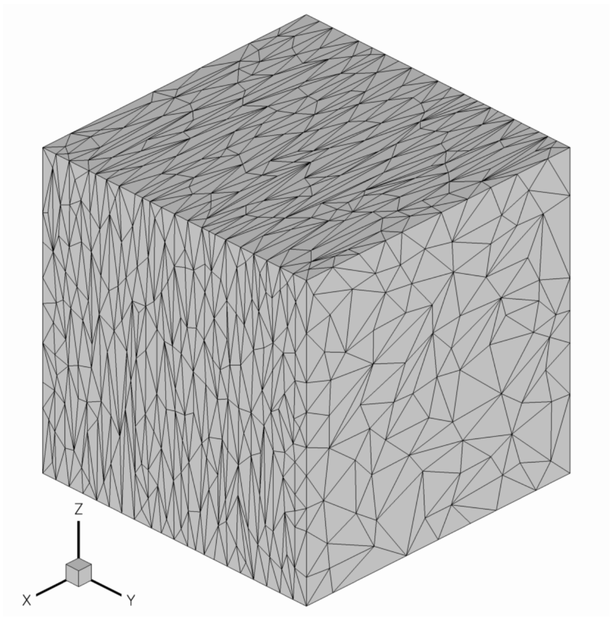
Figure 18: Exact $x^2 + 1000y^2 + 10z^3$ function in a cube.

5.3. *3D Interpolation Error*

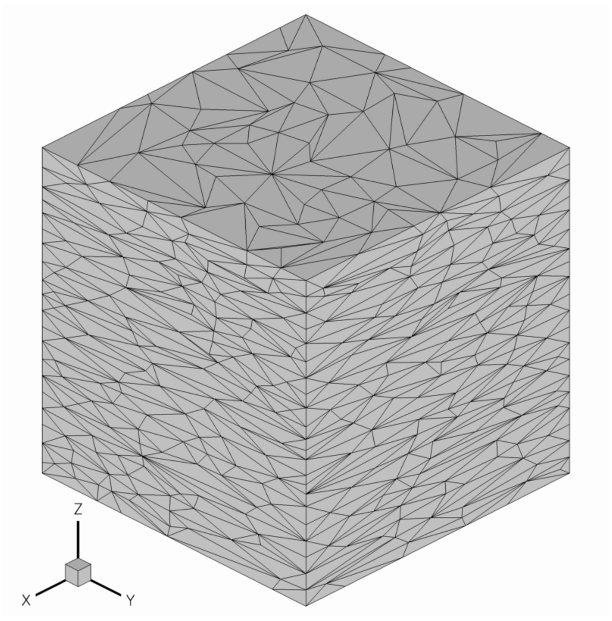
A cubic domain is employed in an 3D adaptation example to directly control interpolation error. The exact function for this example is

$$u = x^2 + 1000y^2 + 10z^3, \tag{58}$$

Fig. 18, used to define the interpolation error $e(\kappa)$. Approximate and reconstructed solutions are not used for this 3D case; the exact function is evaluated. The function is dominated by the $1000y^2$ term, as seen in the nearly planar contours. The direct adaptation algorithm `ErrorControlMD` is utilized. The initial grid is shown in Fig. 6(a). The grid adapted to $\mathrm{tol}_\Omega = 5.0 \times 10^{-1}$ for $p = 1$ tetrahedra is shown in Fig. 19(a). The grid adapted to $\mathrm{tol}_\Omega = 5.0 \times 10^{-5}$ for $p = 2$ tetrahedra is shown in Fig. 19(b). For both cases the final interpolation error norm is approximately half the requested error norm. The anisotropy seen in Fig. 19 is a result of efficiently resolving the anisotropic function. The $p = 1$ basis adapts to anisotropically resolve the $1000y^2$ term. The $p = 2$ basis resolves the quadratic term, $1000y^2$, exactly, so it anisotropically resolves the $10z^3$ term. The shift in the primary anisotropic direction is clearly evident as the interpolation error is directly controlled.



(a) Adapted $p = 1$ to $\mathrm{tol}_\Omega = 5.0 \times 10^{-1}$, actual $e_\Omega = 2.5 \times 10^{-1}$.



(b) Adapted $p = 2$ to $\mathrm{tol}_\Omega = 5.0 \times 10^{-5}$, actual $e_\Omega = 2.0 \times 10^{-5}$.

Figure 19: Isometric views of an interpolation error adapted cube.

6. Conclusions

Elemental grid operators are described with a corresponding parallel execution scheme. These elemental operators are combined into a higher-level algorithm to produce strongly anisotropic tetrahedral grids. This metric-based approach produces tetrahedral elements that satisfy an anisotropic metric field with aspect ratio of 1000 to 1. Although a simple cubic domain was illistrated in this work, the method is applicable to complex domains whe combined with a tetrahedral cut-cell appraoch.[19, 39] A novel direct adaptation approach is developed that evaluates a general expression for error. This avoids the assumptions and approximations required to form a metric to specify adaptation. To illustrate this new approach, the grid adaptation is performed in 1D, 2D, and 3D to directly reduce and equidistribute interpolation error. This new method is uniformly applied to linear and higher-order elements. An example is also provided for a case when the exact interpolant is not available and a surrogate approximation is employed. The direct method is shown to be more efficient than the established metric based approach by having a smaller number of elements for a given total interpolation error. The direct method was also applied to scalar convection-diffusion[19] and found to more efficient than existing metric-based approaches. The direct method produced grids that are less regular then the metric-based method, which may indicate that there is the potential for additional improvement.

References

[1] D. J. Mavriplis, J. C. Vassberg, E. N. Tinoco, M. Mani, O. P. Brodersen, B. Eisfeld, R. A. Wahls, J. H. Morrison, T. Zickuhr, D. Levy, M. Murayama, Grid quality and resolution issues from the drag prediction workshop series, AIAA Paper 2008–930 (2008).

[2] D. J. Mavriplis, Grid resolution study of a drag prediction workshop configuration using the nsu3d unstructured mesh solver, AIAA Paper 2005–4729 (2005).

[3] M. S. Chaffin, S. Pirzadeh, Unstructured navier-stokes high-lift computations on a trapezoidal wing, AIAA Paper 2005–5084 (2005).

[4] S. M. Murman, M. J. Aftosmis, M. Nemec, Automated parameter studies using a cartesian method, AIAA Paper 2004-5076 (2004).

[5] M. Nemec, M. J. Aftosmis, Aerodynamic shape optimization using a cartesian adjoint method and cad geometry, AIAA Paper 2006-3456 (2006).

[6] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, Journal Computational Physics 187 (2003) 22–46.

[7] J. Dompierre, M.-G. Vallet, Y. Bourgault, M. Fortin, W. G. Habashi, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent cfd. part iii. unstructured grids, International Journal for Numerical Methods in Fluids 39 (8) (2002) 675–702.

[8] K. J. Fidkowski, D. L. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible navier–stokes equations, Journal Computational Physics 225 (2) (2007) 1653–1672. doi:http://dx.doi.org/10.1016/j.jcp.2007.02.007.

[9] D. J. Mavriplis, Unstructured mesh generation and adaptivity, Tech. Rep. ICASE 95-26, NASA Langley, Hampton VA, see also VKI Lecture Series 1995-2 and NASA-CR-195069. (Apr. 1995).

[10] T. J. Baker, Mesh adaptation strategies for problems in fluid dynamics, Finite Elements in Analysis and Design 25 (3–4) (1997) 243–273.

[11] J. Peraire, M. Vahdati, K. Morgan, O. C. Zienkiewicz, Adaptive remeshing for compressible flow computations, Journal of Computational Physics 72 (2) (1987) 449–466.

[12] D. J. Mavriplis, Turbulent flow calculations using unstructured and adaptive meshes, International Journal for Numerical Methods in Fluids 13 (1991) 1131–1152.

[13] M. J. Castro-Diáz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaptation for flow simulations, International Journal for Numerical Methods in Fluids 25 (4) (1997) 475–491.

[14] H. Borouchaki, P. L. George, F. Hecht, P. Laug, E. Saltel, Delaunay mesh generation governed by metric specifications. part i. algorithms, Finite Elements in Analysis and Design 25 (1–2) (1997) 61–83.

[15] H. Borouchaki, P. L. George, F. Hecht, P. Laug, E. Saltel, Delaunay mesh generation governed by metric specifications. part ii. applications, Finite Elements in Analysis and Design 25 (1–2) (1997) 85–109.

[16] J. Peraire, J. Peirò, K. Morgan, Adaptive remeshing for three-dimensional compressible flow computations, Journal of Computational Physics 103 (2) (1992) 269–285.

[17] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, M.-G. Vallet, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent cfd. part i: general principles, International Journal for Numerical Methods in Fluids 32 (6) (2000) 725–744.

[18] R. Rannacher, Adaptive galerkin finite element methods for partial differential equations, Journal of Computational and Applied Mathematics 128 (2001) 205–233.

[19] M. A. Park, Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows, Ph.D. thesis, Massachusetts Institute of Technology (Sep. 2008).

[20] L. A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, International Journal for Numerical Methods in Engineering 40 (1997) 3979–4002.

[21] D. J. Mavriplis, Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes, International Journal for Numerical Methods in Fluids 34 (2) (2000) 93–111, see also AIAA Paper 97-0857.

[22] H. L. De Cougny, M. S. Shephard, Parallel refinement and coarsening of tetrahedral meshes, International Journal for Numerical Methods in Engineering 46 (7) (1999) 1101–1125.

[23] E. J. Nielsen, W. K. Anderson, Recent improvements in aerodynamic design optimization on unstructured meshes, AIAA Journal 40 (6) (2002) 1155–1163, see also AIAA Paper 2001–596.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms (Second Edition), MIT Press, Cambridge, Massachusetts, 2003.

[25] C. Y. Lepage, W. G. Habashi, Parallel unstructured mesh adaptation on distributed-memory systems, AIAA Paper 2004-2532 (2004).

[26] F. Alauzet, X. Li, E. S. Seol, M. S. Shephard, Parallel anisotropic 3d mesh adaptation by mesh modification, Engineering with Computers 21 (3) (2006) 247–258.

[27] P. A. Cavallo, N. Sinha, G. M. Feldman, Parallel unstructured mesh adaptation method for moving body applications, AIAA Journal 43 (9) (2005) 1937–1845.

[28] L. A. Freitag, M. T. Jones, P. E. Plassmann, A parallel algorithm for mesh smoothing, SIAM Journal on Scientific Computing 20 (6) (1999) 2023–2040.

[29] R. Löhner, A parallel advancing front grid generation scheme, International Journal for Numerical Methods in Engineering 51 (6) (2001) 647–661, see also AIAA Paper 2000-1005.

[30] K. Schloegel, G. Karypis, V. Kumar, Parallel static and dynamic multi-constraint graph partitioning, Concurrency and Computation: Practice and Experience 14 (3) (2002) 219–240.

[31] P. J. Frey, F. Alauzet, Anisotropic mesh adaptation for cfd computations, Computer Methods in Applied Mechanics and Engineering (194) (2005) 5068–5082.
[32] A. Liu, B. Joe, Relationship between tetrahedron shape measures, BIT Numerical Mathematics 34 (2) (1994) 268–287.
[33] J. Dompierre, M.-G. Vallet, P. L. F. Guibault, An analysis of simplex shape measures for anisotropic meshes, Computer Methods in Applied Mechanics and Engineering 194 (48-49) (2005) 4895–4914. doi:http://dx.doi.org/10.1016/j.cma.2004.11.018.
[34] J. R. Shewchuk, What is a good linear element? interpolation, conditioning, and quality measures, in: 11th International Meshing Roundtable, 2002.
[35] P. Labbé, J. Dompierre, , M.-G. Vallet, F. Guilault, J.-Y. Trépanier, A universal measure of the conformity of a mesh with respect to an anisotropic metric field, International Journal for Numerical Methods in Engineering 61 (15) (2004) 2675–2695.
[36] J. Peiro, J. Peraire, K. Morgan, Felisa reference manual and user's guide, volume i, University of Wales/Swansea Report CR/821/94 (1994).
[37] W. T. Jones, Gridex – an integrated grid generation package for cfd, AIAA Paper 2003–4129 (2003).
[38] O. C. Zienkiewicz, J. Z. Zhu, Adaptivity and mesh generation, International Journal for Numerical Methods in Engineering 35 (4) (1991) 783–810.
[39] M. A. Park, D. Darmofal, Output-adaptive tetrahedral cut-cell validation for sonic boom prediction, AIAA Paper 2008–6594 (2008).