Four-dimensional anisotropic mesh adaptation

Philip Claude Caplan[a,*], Robert Haimes[b], David L. Darmofal[b]

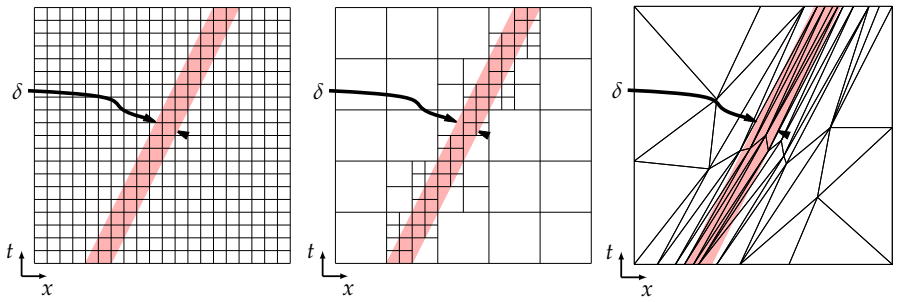
[a]*Middlebury College, Department of Computer Science*
[b]*Massachusetts Institute of Technology, Department of Aeronautics & Astronautics*

Abstract

Anisotropic mesh adaptation is important for accurately simulating physical phenomena at reasonable computational costs. Previous work in anisotropic mesh adaptation has been restricted to studies in two- or three-dimensional computational domains. However, in order to accurately simulate time-dependent physical phenomena in three dimensions, a four-dimensional mesh adaptation tool is needed. This work develops a four-dimensional anisotropic mesh adaptation tool to support time-dependent three-dimensional numerical simulations. Anisotropy is achieved through the use of a background metric field and the mesh is adapted using a dimension-independent cavity framework. Metric-conformity – in the sense of edge lengths, element quality and element counts – is effectively demonstrated on four-dimensional benchmark cases within a unit tesseract in which the background metric is prescribed analytically. Next, the metric field is optimized to minimize the approximation error of a scalar function with discontinuous Galerkin discretizations on four-dimensional domains. We demonstrate that this four-dimensional mesh adaptation algorithm achieves optimal element sizes and orientations. To our knowledge, this is the first presentation of anisotropic four-dimensional meshes.

Keywords: mesh adaptation, metric-conforming, four-dimensional, function approximation, high-order finite elements

*Corresponding author
Email addresses: pcaplan@middlebury.edu (Philip Claude Caplan), haimes@mit.edu (Robert Haimes), darmofal@mit.edu (David L. Darmofal)



(a) Uniform refinement (b) Tensor-product refinement (c) Unstructured approach

Figure 1: Uniform, tensor-product and unstructured adaptation approaches for capturing the propagation of a one-dimensional feature.

1. Introduction

The increase in computational power in the last several decades has given rise to efficient algorithms for predicting engineering quantities of interest with numerical simulations. In particular, high-order finite element methods combined with mesh adaptation techniques for numerically solving partial differential equations have demonstrated their potential for *accurately* predicting these quantities. Yano and Darmofal [1] remark upon the importance of mesh adaptation when using high-order discretizations for estimating the drag and lift with the Reynolds-Averaged Navier-Stokes equations. This finding is further supported by the CFD Vision 2030 study [2] in which the authors emphasize the development of (1) mesh adaptation and (2) high-order discretizations to achieve an autonomous and reliable CFD simulation.

For the calculation of time-varying quantities of interest, first consider a one-dimensional ($1d$) problem with a propagating feature of characteristic size δ (moving from left to right in Fig. 1). A purely spatial mesh would require elements of width $\Delta x < \delta$ near the feature to accurately resolve its position. Now, to accurately track the position in time, a traditional time-marching approach would require a small time step (Δt).

In a fully coupled spatiotemporal approach, elements can be constructed

from the tensor product of a spatial element with a time interval – see the pioneering work of Oden [3], Argyris and Scharpf [4] and Fried [5]. That is, a two-dimensional ($2d$) mesh is used for the coupled $1d + t$ problem and a three-dimensional ($3d$) mesh is used for the coupled $2d + t$ problem. This coupling method enables the use of a wide range of adaptation techniques, ranging from timeslab to fully unstructured approaches.

First, a timeslab approach constructs spacetime elements as the tensor-product of a (possibly unstructured) spatial element with a time interval. These are simpler than fully unstructured spatiotemporal approaches because they essentially require spatially-adapted meshes. The coupled spatiotemporal mesh can be constructed in several ways. Behr extrudes an initial spatial mesh to form prismatic elements which are then subdivided to form simplices [6]. Temporal refinement is achieved by inserting vertices along edges formed during this extrusion process. The Tent Pitcher algorithm of Ungor [7] and Erickson [8] is essentially an advancing front method, starting from an initial spatial mesh and inserts points in the temporal direction to satisfy a *cone constraint* imposed by the characteristics of the governing partial differential equation. This method was extended to the $3d + t$ case by Mont [9]. Thite [10] improved the Tent Pitcher algorithm to also allow for coarsening of $2d + t$ spatiotemporal meshes.

Fidkowski also employs a tensor-product approach for the solution of the compressible Navier-Stokes equations [11, 12] in which the spatial mesh is fixed and the temporal discretization is refined by bisecting the time intervals. In other words, each spatial element at a particular time takes the same time step.

Bangerth and Rannacher [13] employ a hierarchical refinement approach to subdivide quadrilateral spatiotemporal elements in a tensor-product manner (Figure 1b). Their adaptive method is driven by the dual-weighted residual error estimator. They demonstrate the approach on linear second-order hyperbolic problems in $1d + t$ and later to $2d + t$ [14]. They conclude that their refinement approach is superior to uniform refinement approaches (Figure 1a)

in achieving a lower output error at a lower degree-of-freedom (DOF) count. The advantage of this tensor-product approach was further demonstrated for Burger's equation by Hartmann [15].

Yano demonstrates that these tensor-product approaches are effectively isotropic and, whereas they offer a substantial DOF savings over uniform refinement, they are dramatically outperformed by a fully unstructured anisotropic approach. In the context of the one-dimensional problem of Figure 1 with a propagating feature of characteristic size δ, Yano experimentally observes that uniform (Figure 1a), tensor-product (Figure 1b) and anisotropic refinement approaches (Figure 1c) respectively require $\mathcal{O}(\delta^{-2})$, $\mathcal{O}(\delta^{-1})$ and $\mathcal{O}(1)$ DOF. Yano further demonstrates his approach for nonlinear $2d + t$ problems which is later extended to oil reservoir simulations by Jayasinghe [16]. In fact, Jayasinghe [17] is the first to demonstrate that a spatiotemporal approach also scales very well with the number of parallel processors when compared to time-marching approaches. To enable fully unstructured adaptive numerical simulations for the resolution of unsteady $3d$ physical phenomena, unstructured anisotropic four-dimensional ($4d$) meshes are needed for the coupled $3d + t$ spatiotemporal domain.

Foteinos et al. employ a Delaunay refinement algorithm to construct $4d$ meshes for $3d + t$ image data in which the topology and geometry of the evolving object can change significantly in time and provide bounds an the aspect ratios of the pentatopes [18]. Belda-Ferrín et al. present a conformal bisection procedure for local isotropic refinement of $4d$ meshes [19] and ensure the quality of the pentatopes does not degenerate upon successive refinement stages. Previous efforts attempt to generate geometry-conforming $4d$ meshes from the restricted Voronoi diagram [20].

Local mesh modification operators, such as *edge splits*, *edge collapses*, *edge swaps*, *face swaps*, *vertex smoothing*, are popular for constructing adapted meshes. Software packages such as `feflo.a` [21], `EPIC` [22], `refine` [23], `gamanic3d` [24], `pragmatic` [25], `omega_h` [26] and `MMG` [27] successfully produce adapted meshes for either $2d$ or $3d$ computational domains. Inspired by the work of Coupez [28]

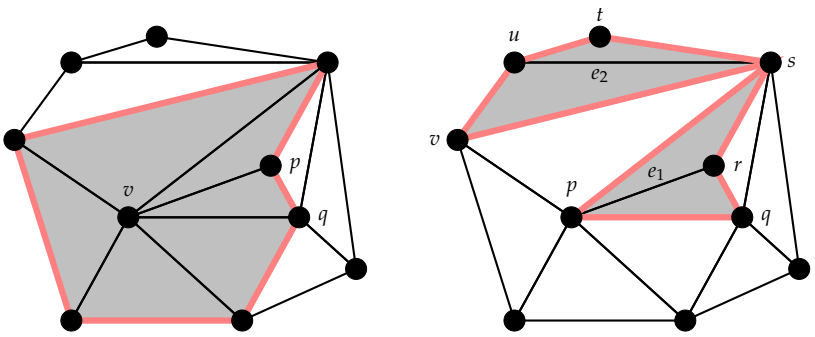
and Gruau [29], we pursue a dimension-independent approach. The aforementioned authors focused on $3d$ applications; in $4d$, they only demonstrated their mesh adaptation capability for the special case of a uniform Euclidean metric. Furthermore, Tremblay also attempted to use local operators to adapt $4d$ meshes [30]. He used edge splits and collapses but avoids edge swapping by employing a combination of point insertions and collapses to *simulate* an edge swap. His algorithm does not seem capable of producing metric-conforming meshes since the resulting edge lengths are short and the element quality is very poor, even for an analytic metric with a maximum aspect ratio of 10:1. Tremblay further attempted to demonstrate his algorithm for the solution of the unsteady heat equation in $3d + t$ but the resulting mesh is isotropic.

At the time of this writing, a truly anisotropic metric-conforming mesh adaptation capability has yet to be demonstrated in $4d$. Our dimension-independent approach, described in Section 2, is a fusion of the *star operator* of Coupez [28] with the *cavity operator* of Loseille [21]. We then apply our algorithm to problems within the four-dimensional domain of a unit tesseract. We first verify that our algorithm produces metric-conforming meshes in Section 3 in which a background metric field is prescribed analytically. Next, we consider the optimal approximant of a scalar function defined on the unit tesseract with discontinuous Galerkin discretizations. In Section 4 we verify that the optimal mesh size and aspect ratio distributions are achieved.

2. Mesh adaptation via local operators

First, we review some notation that is used in our dimension-independent mesh adaptation algorithm. Let $\mathcal{V}$ be a set of vertices in $\mathbb{R}^n$. A *mesh*, $\mathcal{M}$ of some domain Ω, is a pair $(\mathcal{V},\ \mathcal{T})$ where $\mathcal{T}$ represents the *topology* that references the set of vertices $\mathcal{V}$. $\mathcal{T}$ is a collection of topological elements, $\mathcal{T} = \kappa_0 \cup \kappa_1 \cdots \cup \kappa_m$, where the element indices correspond to the vertices $\mathcal{V}$.

The *boundary* of the mesh, denoted by $\partial\mathcal{M} = (\mathcal{V},\ \partial\mathcal{T})$, is a mesh itself,



(a) Cavity about vertex v.

(b) Cavities about edges e_1 and e_2.

Figure 2: Identification of a set of cavity elements. Elements in gray (shaded) form the cavities $\mathcal{C}(v)$, $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$. The boundaries of each cavity are outlined in red (thick).

and is the set of $(n-1)$-facets that appear as a facet of only *one* simplex of the mesh. Any $(n-1)$-facet (also a simplex) that is shared by two simplices is not on the boundary, but is *interior*. The extraction of the boundary is purely topological (i.e., we only need to determine $\partial\mathcal{T}$).

We study n-simplicial meshes embedded in $\mathbb{R}^n$. Thus a physical element $\boldsymbol{\kappa}$ should be understood as the convex hull of the vertices indexed by the topological element κ, and is denoted by $\mathcal{V}(\kappa)$.

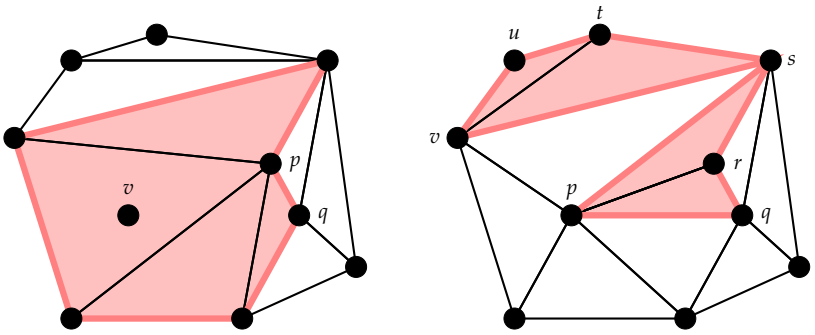
2.1. *Dimension-independent local operators*

Given an initial n-simplicial mesh $\mathcal{M} = (\mathcal{V}, \mathcal{T})$ of some domain $\Omega \subset \mathbb{R}^n$, a local operation on $\mathcal{M}$ consists of the transformation of its topology $\mathcal{T}$ [28, 29, 21]:

$$\mathcal{T}^{k+1} = \mathcal{T}^k \setminus \mathcal{C}^k(f) \cup \mathcal{B}^k(p, \partial\mathcal{C}^k) \tag{1}$$

where the superscripts represent the sequence of meshes at each application of the cavity removal, $\mathcal{C}^k(f)$, and the insertion, $\mathcal{B}^k(p, \partial\mathcal{C}^k)$. $\mathcal{C}^k(f)$ denotes the set of *cavity* elements about a j-dimensional facet $f \subset \mathcal{T}^k$ which might be enlarged to ensure topological and geometric validity. This set of cavity elements can be written as

$$\mathcal{C}^k(f) = \left\{ \kappa \,\middle|\, f \subset \kappa, \forall \kappa \in \mathcal{T}^k \right\}. \tag{2}$$



(a) Valid insertion for the cavity of Fig. 2a.

(b) Valid insertions for the cavities of Fig. 2b.

Figure 3: Selection of the re-insertion vertex to produce valid meshes. The set of insertion elements is shown in red which is obtained by selecting a vertex and connecting it to the boundary of the set of cavity elements, thickly outlined in red. After extracting the set of cavity elements $\mathcal{C}(e_1)$, possible re-insertion candidates include vertices p, q, r and s. However, candidates q and s create invalid meshes. Also observe that selecting either p or r as the re-insertion vertex maintains the original configuration of Fig. 2a. In comparison, any of the re-insertion vertices s, t, u or v produce valid meshes when connected to the boundary of the cavity $\mathcal{C}(e_2)$.

Often f is chosen from the set vertices (as integers) or edges of $\mathcal{T}^k$, as in the work of Gruau and Coupez [31]. For a vertex v and edges e_1 and e_2, the corresponding cavities $\mathcal{C}(v)$, $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$ are shown in the gray (shaded) triangles of Fig. 2. The boundary of the cavity, $\partial\mathcal{C}^k(f)$, is outlined in red (thick). Note that two possible cavities are shown in Fig. 2b about edges e_1 and e_2.

The insertion operator, $\mathcal{B}^k(p,\ \partial\mathcal{C}^k)$, is obtained by connecting some (possibly new) vertex p with the set of $(n-1)$-dimensional facets in $\partial\mathcal{C}^k$ which do not contain p [28, 29]. Mathematically,

$$\mathcal{B}^k(p, \partial\mathcal{C}^k) = \left\{ \kappa \,\middle|\, \kappa = \{p\} \cup g,\ g \in \partial\mathcal{C}^k,\ p \notin g \right\}. \tag{3}$$

where g is a $(n-1)$-dimensional facet on the boundary $\partial\mathcal{C}^k$. The *re-insertion vertex* p is often chosen from the set of vertices in the cavity, though Loseille allows it to be outside the cavity.

Superscripts will now be dropped for brevity. Following the example of Fig. 2, possible insertions are shown in red of Fig. 3. Note that in Fig. 3a,

the vertex p was selected as the re-insertion vertex because the re-insertion vertex q would create an invalid mesh. To see this, observe that not every facet in the boundary is *visible* to q. As a result, Loseille proposes to iteratively enlarge the cavity until q *is* visible to the boundary of the cavity [21]. The set of re-insertion elements should satisfy a *volume* criterion:

$$v(\underbrace{\text{conv}(\overbrace{\mathcal{V}(\{p\} \cup g)}^{\kappa})}_{\kappa}) > 0, \forall g \in \partial\mathcal{C},\ p \notin g. \tag{4}$$

This volume can be computed with exact geometric predicates, which we develop for the case of pentatopes using the Predicate Construction Kit of Lévy [32], thus providing an `orient4d` function similar to that of the `orient2d` and `orient3d` [33] capabilities.

Similarly, Coupez enlarges an initial set of cavity elements $\mathcal{C}(f)$ to include those simplices in $\mathcal{T}$ which are in the *closure* of the vertices of the original cavity $\mathcal{N}(\mathcal{C}(f))$, where $\mathcal{N}(\cdot)$ retrieves the set of vertices (as integers) in the provided set of elements. Combined with his *minimum volume principle*, Coupez initially demonstrated this guarantees the topological validity of the mesh [28]. In contrast to the works of Loseille and Coupez, we do not allow cavities to enlarge, which simplifies the implementation of our method.

2.2. *Maintaining a valid mesh*

The first thing we do when checking for the validity of a local operator is to check whether the proposed re-insertion vertex is visible to the boundary of the removed cavity using Eq. 4 as in the work of Loseille [21]. This allows us to geometrically filter out operators that would create invalid meshes.

Furthermore, we *close* the mesh such that it is without boundary, similar to the work of Coupez and Gruau [28, 29]. To achieve a mesh without boundary, a ghost (fictitious) vertex is created for each connected boundary. This vertex is then connected to the boundary facets of the original mesh and the resulting (ghost) simplices are appended to this incoming mesh. Without loss of generality, consider a mesh with a single connected boundary, such as that

of a n-cube with no interior holes. Denote the ghost vertex as v_0 and the incoming mesh as $\mathcal{T}_0$. The closed mesh is obtained by defining a set of ghost simplices: $\mathcal{T}_g = \{\{v_0\} \cup g \,|\, \forall g \in \partial\mathcal{T}_0\}$. The full topology of the mesh now becomes the union of the simplices in the incoming mesh with the ghost simplices: $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_g$. By working with a closed mesh, we ensure that $\partial\mathcal{T} = \emptyset$ and should remain so upon application of each local operation, similar to the suggestion of Coupez [28, 34].

Upon application of the cavity operator, the faces on the boundary of the cavity, as seen from the remaining mesh ($\mathcal{T} \setminus \mathcal{C}$), are first cached. Next, the boundary of the insertion operator is computed and the neighbor-relations are updated by progressively removing the facet in the cache that exists in the boundary of the insertion operator. After the operator terminates, the cache should be empty, which we strictly enforce. We note that the use of a closed mesh, motivated by the work of Coupez & Gruau [28, 29], made our implementation simpler, though other implementations are certainly possible.

Table 1: Choice of re-insertion vertices (with associated coordinates) for local operators. The cavities are obtained using Eq. 2 about the j-dimensional facet f listed in the second column.

	j	facet, f	vertex, $p(\mathbf{x})$
collapse	1	vertex v_0	$v_1(\mathbf{x}_1)$
split	1	edge $e = (v_0, v_1)$	$v_s(\mathbf{x}_s)$
edge swap	1	edge $e = (v_0, v_1)$	$p(\mathbf{x}_p) \in \mathcal{N}(\mathcal{C})$
smooth	0	vertex p	$p(\widetilde{\mathbf{x}}_p)$

2.3. Recovery of common mesh modification operators

The advantage of employing the cavity-insertion framework is that, with the appropriate selection of the cavities and re-insertion vertices, all other mesh operators can be recovered. In fact, Figs. 2a and Figs. 3a are the sequence of steps taken to perform an edge collapse (or vertex removal). Similarly, the sequence of steps in Figs. 2b and Figs. 3b show two possible edge swaps. Table 1 gives the cavities and re-insertion vertices which recover common mesh modification operators.

The coordinates for the re-insertion vertices can also be modified during the local operation. For collapses and swaps, the vertex coordinates are not modified and simply fixed at the coordinates of the selected re-insertion vertex. For edge splits, the re-insertion vertex inherits the coordinates of the midpoint of the edge upon which the split occurs ($\mathbf{x}_s$ in Table 1). The modification to the coordinates obtained with vertex smoothing is discussed in a later section.

Now, let us study the computational cost of the local mesh operator. In particular, we will measure the cost per operation for each operator when modifying a mesh in $2d$, $3d$ and $4d$. In each dimension, we vary the size of a Kuhn-Freudenthal triangulation [35] with m^d vertices (modifying m). Next, we sample the time to perform 1000 instances of each local operation on this mesh and average the result. These timing results were obtained with a 2.8GHz Intel Core i7-8569U processor.

The collapse operator is the most expensive, the cost of which appears to increase linearly with the size of the mesh (see the gray lines in Fig. 4). This is likely due to the design of our data structures, which must update both the topology indices and inverse topology (an element pointed to by a particular vertex) when vertices and elements are removed. Although further work could seek to improve this, we generally start our adaptation problems with coarser meshes – the collapse operator is scarcely employed as the adaptation converges and few edge collapses are needed. The cost of the insertion operator also increases linearly with the size of the mesh, though at a lower rate than the collapse operator (yellow lines in Fig. 4). The swap operator (blue lines) is the most efficient and does not increase in cost with the size of the mesh. Finally, the timing results for vertex smoothing (dashed red lines) are also independent of the size of the mesh (they almost overlap with the cost of the swap operator). Though the nonlinear nature of the smoothing problem can be costly, our vertex smoothing procedure is a *localized* one – we do not minimize a global objective function on the vertices, but consider a local smoothing scheme for every vertex (see Section 2.6). Note that the same

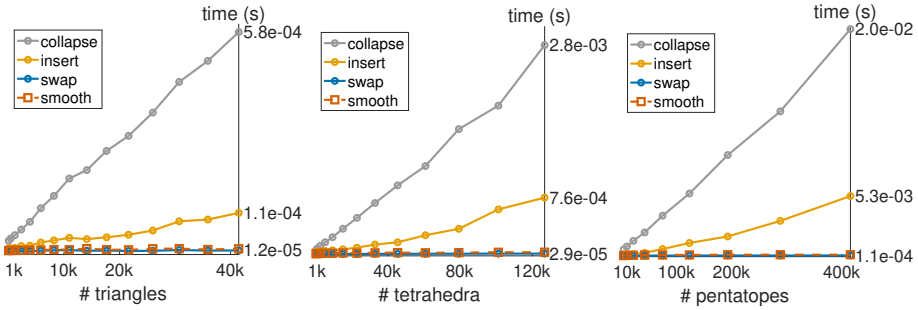


Figure 4: Timing breakdown of the topology-modifying operators (collapse, insert, swap) in $2d$, $3d$ and $4d$ acting on Kuhn-Freudenthal triangulations of various sizes.

trends in computational costs are observed in $2d$ (triangles), $3d$ (tetrahedra) and $4d$ (pentatopes).

2.4. The importance of the geometry metadata

We restrict our attention to the four-dimensional computational domain of a unit tesseract. The geometry representation is constructed by assigning the appropriate hierarchical relations between the geometric entities: Nodes, Edges, Faces (Squares) and Volumes (Cubes) [36]. Please note the distinction between a *mesh vertex* and a *geometry Node*, as well as a *mesh edge* and a *geometry Edge*. Capital letters are used to reference entities of the geometry topology.

It is critical to check whether a proposed mesh modification operator violates the discretization of the geometry. To perform this check, we strictly enforce that all vertices are tagged with geometry metadata. This *must* be the *lowest-dimensional* geometry entity. For example, a vertex on a geometry Edge is also on Faces (and Volumes for a tesseract geometry) but the associated geometry for this vertex must be the Edge. For interior vertices, this geometry entity is empty ($\emptyset$).

A common operation in the mesh adaptation algorithm is to determine which geometry entity a facet lies on. For a j-dimensional facet $f \subset \mathcal{T}$, denote the geometric entities of the vertices of f as $\{g_v\}_{v \in f}$. Now, define the set of parents of a geometry entity g as the set of all geometry entities higher in the

geometry hierarchy $\mathcal{G}$:

$$\mathcal{P}(g) = \left\{ h \;\middle|\; g \preceq h,\ \forall h \in \mathcal{G} \right\}. \tag{5}$$

For each vertex of f, we have the set of parents $\{\mathcal{P}(g_v)\}_{v \in f}$. The geometry entity of this facet g_f is computed from the lowest-dimensional member of the intersection of all these parents. Denote all common parents as G_f:

$$G_f = \bigcap_{v \in f} \mathcal{P}(g_v). \tag{6}$$

The geometry entity on which this facet lies is then

$$g_f = \arg\min_{g \in G_f} \ \dim(g). \tag{7}$$

Of course, g_f can be empty even if $g_v \neq \emptyset, \forall v \in f$. It is possible that g_f is nonempty despite f being an interior facet of the mesh. The *closed* mesh is handy in treating this scenario. Every facet on a geometry entity must be adjacent to a ghost simplex. That is,

$$\mathcal{C}(f) \in \mathcal{T}_g \quad \text{where } \mathcal{C}(f) \text{ is from Eq. 2} \tag{8}$$

for facets on geometry discretizations.

We are now equipped to check whether a mesh modification operator violates the geometry discretization. In the following, assume the full geometry hierarchy is denoted, as a partially ordered set, by $\mathcal{G}$. For an edge collapse with edge $e = (v_0, v_1)$, if $g_e \neq \emptyset$, the removed vertex v_0 must be higher in $\mathcal{G}$ than v_1. That is $g_{v_1} \preceq g_{v_0}$. When swapping an edge $e = (v_0, v_1)$ with a re-insertion vertex p, the geometry of the re-insertion vertex g_p must be lower than (or equal to) g_e: $g_p \preceq g_e$. Note that if $\dim(g_e) = 1$, then the swap is rejected because we do not want to swap along a geometry Edge. For straight-sided domains, a split along an edge e always creates a valid insertion, both

geometrically and topologically. However, to avoid mistakes in subsequent mesh operations, the created vertex must be tagged with g_e, the geometry entity of the mesh edge being split.

2.5. *Achieving anisotropy using a background metric field*

The use of a background metric field to produce anisotropic meshes has been well demonstrated [37]. This metric field, prescribed discretely at the vertices of the input mesh (to be adapted), is used to modify the calculations of edge length and element volume, which ultimately drive the local mesh operator schedule. We follow the conventions proposed by the Unstructured Grid Adaptation Working Group (UGAWG) [38] and calculate the length between vertices $\mathbf{p}$ and $\mathbf{q}$ as

$$\ell_{\mathbf{m}}(\mathbf{p},\mathbf{q}) \approx \ell_{\mathbf{m}(\mathbf{p})} \frac{r-1}{r \log r} \quad \text{with } r \equiv \frac{\ell_{\mathbf{m}(\mathbf{p})}}{\ell_{\mathbf{m}(\mathbf{q})}}, \tag{9}$$

where the length of an edge $\mathbf{e}$ under a constant metric $\mathbf{m}$ is $\ell_{\mathbf{m}}(\mathbf{e}) = \sqrt{\mathbf{e}^t \mathbf{m} \mathbf{e}}$. The volume of an element $\boldsymbol{\kappa}$ is [38]

$$v_{\mathbf{m}}(\boldsymbol{\kappa}) \approx \sqrt{\det \mathbf{m}_\nu}\, v(\boldsymbol{\kappa}), \quad \text{with } \nu = \arg\max_{\nu \in \kappa} \det \mathbf{m}_\nu, \tag{10}$$

where $v(\boldsymbol{\kappa})$ is the Euclidean volume of the element.

The goal of the anisotropic mesh adaptation algorithm is for all edges of the metric-conforming mesh to be of unit length under the metric field, and for the mapped volumes of all elements to be that of the unit simplex. Instead of optimizing element volumes, we optimize element quality, similar to the conventions of the UGAWG [38]. Similar to what is done in the literature [21, 38, 39], we relax the edge length condition such that edges satisfy a quasi-unit length condition: $\ell_{\mathbf{m}}(\mathbf{p},\mathbf{q}) \in [\sqrt{2}/2, \sqrt{2}]$. Furthermore, we target element quality to be above 0.8, similar to Loseille's proposition [40]. Similar to the

UGAWG [38], element quality is computed as

$$q_{\mathbf{m}}(\kappa) = \beta_n \frac{v_{\mathbf{m}}(\kappa)^{2/n}}{\sum_{e \in \mathcal{E}(\kappa)} \ell^2_{\mathbf{m}}(e)} \tag{11}$$

where β_n is a constant that normalizes the quality of an n-simplex to be within $[0, 1]$. Note that $\beta_4 = 10/\sqrt{v_\Delta}$ such that the quality of a unit equilateral pentatope (under the metric field) is unit: $v_n = \sqrt{n+1}/(n!\sqrt{2^n})$ which, for $n = 4$, gives $v_\Delta = \sqrt{5}/96$. Finally, we expect the number of elements to be close to that expected by the background (target) metric field. Ideally, the volume of the domain in the metric space should be meshed with equilateral pentatopes. The expected number of pentatopes is then the volume of the domain in the metric space, normalized by the volume of an equilateral pentatope v_Δ. This mapped volume can be computed by integrating $\sqrt{\det \mathbf{m}(\mathbf{x})}$ over the domain [39]. Therefore, the number of pentatopes is

$$n_s \approx \frac{1}{v_\Delta} \int_\Omega \sqrt{\det \mathbf{m}(\mathbf{x})} \mathrm{d}\mathbf{x}. \tag{12}$$

All three quantities: edge length, element (pentatope) quality and total element counts are indications of how well the mesh conforms to the metric field. We report the percentage of edges within the quasi-unit range, the average quality of the elements and the total number of generated pentatopes.

A note about computing the implied metric of the mesh. Finally, the mesh itself defines a Riemannian metric, which we use to limit the change from one mesh to the next in the adaptation sequence. The implied metric at each vertex, $\mathbf{m}_{I,v}$, can be computed in several ways, such as averaging the implied metrics of the surrounding elements, using the length distribution tensor of Coupez [41], or by solving an optimization problem. Here, we solve an optimization problem that penalizes (1) the deviation between the number of elements and the complexity of the metric (Eq. 12), and (2) the deviation of the edge lengths from the quasi-unit range, as measured under the implied metric [36] – for brevity, we

refer the interested reader to the aforementioned reference for further details.

2.6. Scheduling the local operators

The local operator schedule is inspired by the work of Loseille [21, 40] whereby collapses and splits are first performed to create a unit mesh. Next, swaps and smoothing are used to optimize the quality of the mesh elements.

Motivated by the suggestion of Loseille, we ensure that no short edges are created during any edge split operation as this would require another pass of the collapse operator [21]. Furthermore, edge splits are restricted if the number of produced pentatopes grows too large. In $4d$, we have experimentally observed the number of pentatopes attached to an edge can be on the order of 15-20, which means 15-20 new pentatopes are created upon the insertion of a single vertex. As a result, a density control factor ρ is introduced to control the metric volume of the inserted simplices. For the inserted cavity $\mathcal{B}$, we require $|\mathcal{B}| \approx v_{\mathbf{m}}(\mathcal{B})/v_{\Delta}$. That is, the number of simplices expected by the metric volume should be approximately equal to the number of inserted pentatopes. In practice, this condition is relaxed by allowing $|\mathcal{B}| < \rho\ v_{\mathbf{m}}(\mathcal{B})/v_{\Delta}$ so as to not be too restrictive with insertions. Here, we empirically set $\rho = \sqrt{2}$ [36]. The impact of this decision on metric conformity will be discussed using some results in Section 3.

Collapses always target edges which are shorter than $\sqrt{2}/2$, however, splits more generally target edges that are longer than some target length ℓ_t. We observed that the ability to parametrize splits in terms of a target length is desirable in order to control the number of simplices generated by the adaptation algorithm [36]. As a result of this observation, the operator schedule is composed of two main stages. The first stage consists of targeting edges longer than $\ell_t = 2$ in the metric space whereas the second stage targets edge lengths longer than $\ell_t = \sqrt{2}$. Each stage visits the edge collapse and split routines twice, which we observed to be helpful in weaving out of restrictive topological configurations [36].

After every visit to the global collapse and split routines, edge swaps are

attempted in order to improve the mesh quality defined by Eq. 11. Before a swap is accepted, the inserted topology is checked to ensure the current minimum and maximum edge lengths do not worsen with the application of the swap.

After edge swaps, a local vertex smoothing procedure is performed. When smoothing a vertex p, the set of neighboring vertices lower in the geometry hierarchy than p are determined and used to compute the new coordinates of p using Eq. 13 (below). This tends to drive the edge lengths (measured under the metric field) surrounding p to unity. For vertex smoothing, the new coordinates are computed from the local edge lengths surrounding the vertex:

$$\widetilde{\mathbf{x}}_p = \mathbf{x}_p + \omega \sum_{e \in \bar{\mathcal{E}}(p)} (1 - \ell_{\mathbf{m}}(e)) \exp(-\ell_{\mathbf{m}(e)}) \mathbf{e} \tag{13}$$

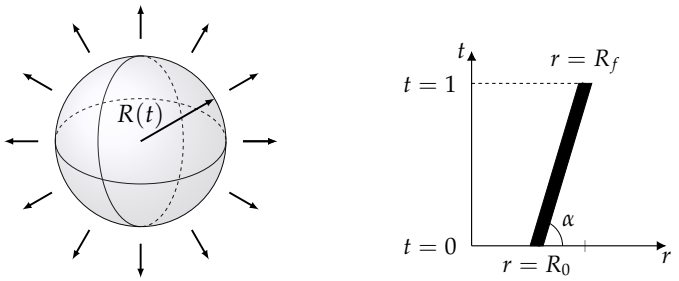
where $\bar{\mathcal{E}}(p)$ is a selected set of the edges that surround the vertex p and $\mathbf{e}$ is the unit vector along that edge. The relaxation factor is selected as $\omega = 0.2$, similar to the inspiring work of Bossen and Heckbert [42]. For interior vertices, $\bar{\mathcal{E}}$ is the full set of edges connected to a vertex. However, for vertices on geometric entities, $\bar{\mathcal{E}}$ is a subset of these edges. This subset is taken as the set of all edges such that the geometry attached to the opposite vertex q is lower than (or equal to) the geometry of vertex p: $g_q \preceq g_p$. Thus vertices on straight-sided geometry entities (such as those in the examples we consider) are smoothed along the entity.

3. Applications I: metric-conforming mesh generation

Let us now study some four-dimensional metric-conforming problems to demonstrate our anisotropic mesh adaptation capability.

Tesseract Linear (TL). The first metric, inspired by the UGAWG Linear metric [38], is represented analytically by

$$\mathbf{m}(\mathbf{x}) = \text{diag}\left(h_x^{-2}, h_y^{-2}, h_z^{-2}, h_t^{-2}\right), \tag{14}$$



(a) Sphere expanding at constant velocity.

(b) Sphere expanding at constant velocity.

Figure 5: Illustration of the Tesseract Wave case and expected refinement.

where $h_x = h_y = h_z = h_{\max}$ and $h_t = h_0 + 2(h_{\max} - h_0)|t - 0.5|$, with $h_0 = 0.01 h_{\max}$. Meshes within a unit tesseract are generated for the two cases where $h_{\max} = [0.25, 0.125]$ so as to assess the performance at both low and moderate mesh sizes. Here, we expect to see six of the eight bounding hypercubes (non-constant t hyperplanes) to show refinement in the t direction. The constant t hyperplanes should exhibit uniform meshes. This case will be referred to as the Tesseract Linear (TL) case and will be further classified as the Tesseract Linear 1 ($h_{\max} = 0.25$) and Tesseract Linear 2 ($h_{\max} = 0.125$). Applying Eq. 12 suggests the Linear 1 case expects 51k pentatopes whereas the Linear 2 case expects 818k pentatopes.

Tesseract Wave (TW). The second metric field is modeled after an expanding spherical wave in $3d$ (see Fig. 5a). Consider a spherical wave of radius $R_0 = 0.4$ centered about the origin at time $t = 0$. If the wave expands at a constant velocity v_p to a radius $R_f = 0.8$ at time $t = 1$, then the expanding sphere traces the geometry of a hypercone in $4d$.

Fig. 5b exhibits the behavior of the expanding $(d-1)$-sphere in a spherical-temporal coordinate system. Note that a slice of the $(d+1)$-dimensional cone with a hyperplane with non-constant temporal component yields a d-cone. Here, this appears as a line but rotational symmetry implies the hypercone sliced by a hyperplane with non-constant temporal component yields a three-dimensional cone. Hence, when extracting the eight cubes bounding the unit

tesseract, we expect to see three-dimensional cones along hyperplanes with a varying temporal component.

The metric used to capture the propagation of this wave is

$$\mathbf{m}(\mathbf{x}) = \mathbf{Q}\,\mathrm{diag}\left(h_r^{-2}, h_\theta^{-2}, h_\phi^{-2}, h_t^{-2}\right)\,\mathbf{Q}^t. \tag{15}$$

The eigenvectors $\mathbf{Q}$ are readily derived by rotating the spherical coordinate unit vectors by an angle α corresponding to the angle made by the velocity vector with the temporal axis. Only the radial unit vector is affected by the rotation which results in a basis given by

$$\mathbf{Q} = \begin{bmatrix} \sin\alpha\cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi & \cos\alpha\cos\phi\sin\theta \\ \sin\alpha\sin\phi\sin\theta & \cos\theta\sin\phi & \cos\phi & \cos\alpha\sin\phi\sin\theta \\ \sin\alpha\cos\theta & -\sin\theta & 0 & \cos\alpha\cos\theta \\ -\cos\alpha & 0 & 0 & \sin\alpha \end{bmatrix}. \tag{16}$$

The spacings in the tangential directions are

$$h_\theta, h_\phi = \begin{cases} h_1, & |r - R(t)| > \delta, \\ (h_1 - h_2)|r - R(t)|/\delta + h_2, & |r - R(t)| \leq \delta, \end{cases} \tag{17}$$

with $R(t) = R_0 + (R_f - R_0)t$ is the position of the spherical wave with time. The spacing in the radial direction is $h_r = h_0 + 2(h_1 - h_0)|r - R(t)|$ and the spacing in the temporal direction is $h_t = 0.5$. Note the use of spherical coordinates, $r = \sqrt{x^2 + y^2 + z^2}$, $\theta = \arccos(z/r)$ and $\phi = \arctan(y, x)$. The rotation angle is equal to $\alpha = \arctan(t_f - t_0, R_f - R_0)$. The remaining parameters are $h_0 = 0.0025$, $h_1 = 0.125$, $h_2 = 0.05$ and $\delta = 0.1$. We were unable to determine the analytic number of pentatopes for this case (i.e. integrating Eq. 12), however, using numerical quadrature on the resulting meshes provides an estimate of 275k pentatopes.

Results. We produce metric-conforming meshes using the procedure of Algorithm 1, starting with a Kuhn-Freudenthal triangulation containing three

generateMetricConformingMesh

input: $\mathcal{M} = (\mathcal{V}, \mathcal{T})$, $\mathbf{m}_t$
output: $\mathcal{M}$
1 **for** $i = 1$ **to** n_{iter}
2 $\quad \mathbf{m}_I \leftarrow$ **impliedMetric**$(\mathcal{M})$ ▷ calculate implied metric of mesh [36]
3 $\quad \mathbf{m} \leftarrow \mathbf{m}_t(\mathcal{V})$ ▷ evaluate target metric at vertices
4 $\quad$ **for** $v \in \mathcal{V}$
5 $\quad\quad \mathbf{s}_v \leftarrow \log\left(\mathbf{m}_{I,v}^{-1/2}\mathbf{m}_v\mathbf{m}_{I,v}^{-1/2}\right)$
6 $\quad\quad \mathbf{s}_v \leftarrow$ **limit**$(\mathbf{s}_v)$ by directly limiting entries to be $\leq 2\log 2$
7 $\quad\quad \mathbf{m}_v \leftarrow \exp\left(\mathbf{m}_{I,v}^{-1/2}\mathbf{s}_v\mathbf{m}_{I,v}^{-1/2}\right)$
8 $\quad \mathcal{M} \leftarrow$ **adaptMesh**$(\mathcal{M}, \mathbf{m})$ ▷ call mesh adaptation tool

Algorithm 1: Target metric assessment procedure. The analytic metric $\mathbf{m}_t$ is first evaluated at the current mesh vertices (Line 3) and then limited in Line 6 according to the step from the implied metric of the mesh to the target (Line 5). A new mesh is then generated on Line 8 from which metric conformity under the analytic metric can be assessed.

vertices in each of the four coordinate directions. We perform twenty iterations of (1) computing the implied metric field of the current mesh and (2) prescribing the discrete metric (from the analytic descriptions in Eqs. 14 and 15) so that the change in edge lengths from the current mesh is limited by a factor of two – see Line 6. Specifically, Line 5 computes the *step matrix* from the implied metric of the mesh at a vertex v to the target metric evaluated at the coordinates of that vertex, using the log-Euclidean framework [43]. The entries of the step matrix are directly limited on Line 6 so that their magnitude is not larger than $2\log 2$, as suggested by Yano [44]. The metric at each vertex that is passed to our mesh adaptation algorithm is then computed as the tensor that is a *step* $\mathbf{s}_v$ from the implied metric at that vertex (Line 7), in the log-Euclidean framework – we elaborate upon this in Section 4.1.

Meshes at the $x = 0$ and $x = 1$ hyperplanes (cubes) for the Tesseract Linear cases are provided in Fig. 7 (Linear 1) and Fig. 8 (Linear 2). For clarity, all eight meshes bounding the tesseract for the Tesseract Wave case are shown in Fig. 9 (in later sections, only a subset of these eight bounding meshes will shown

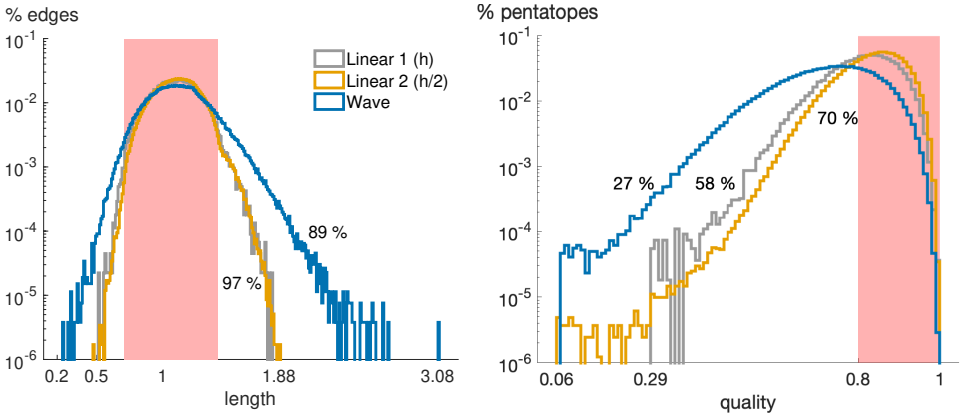


Figure 6: Metric conformity statistics obtained from Algorithm 1 for the four-dimensional benchmark cases. Shaded regions outline the target regions for the edge lengths and element qualities.

Table 2: Metric-conformity statistics for the $4d$ benchmark cases with (ρ) and without ($\not{\rho}$) the density control factor.

	ℓ_{min}	ℓ_{max}	ℓ_{avg}	% ℓ_{unit}	q_{min}	q_{avg}	% q_{unit}	$n_{\text{pentatope}}$
Linear 1 (ρ)	0.52	1.82	1.10	96.8%	0.30	0.81	57.8%	55k
($\not{\rho}$)	0.47	1.81	1.08	97.4%	0.17	0.80	55.2%	60k
Linear 2 (ρ)	0.48	1.88	1.11	97.3%	0.07	0.83	69.9%	816k
($\not{\rho}$)	0.38	1.85	1.08	98.0%	0.02	0.83	67.3%	914k
Wave (ρ)	0.24	3.08	1.12	88.9%	0.07	0.72	26.5%	347k
($\not{\rho}$)	0.28	2.58	1.08	92.5%	0.03	0.72	28.7%	394k

for brevity). Observe that the expected refinement of the cones are observed along hyperplanes with non-constant temporal component. At $t = 0$ and $t = 1$, the sphere at the initial and final radii are respectively observed. The edge length and quality histograms of Fig. 6 demonstrate an acceptable level of metric conformity for all cases. Specifically, the number of edges within the quasi-unit range is very good: approximately 97% for the Linear cases and 89% for the Wave case. Element quality is best for the Linear 2 case and poorest for the Wave case.

These results compare well with metric conformity statistics of existing $3d$ mesh adaptation technologies. In fact, we apply our same algorithm to the benchmarks proposed by the UGAWG [38] and achieve comparable results to those presented in the paper. We summarize the results of these benchmarks –

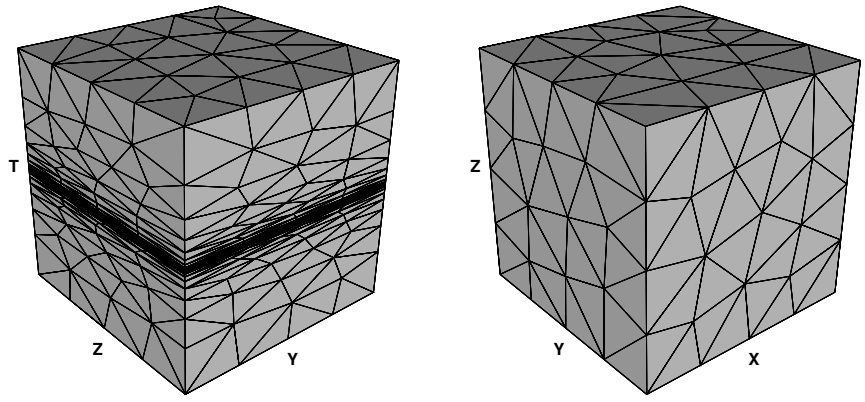
Table 3: Metric-conformity statistics for the 3*d* UGAWG benchmark cases.

	ℓ_{min}	ℓ_{max}	ℓ_{avg}	% ℓ_{unit}	q_{min}	q_{avg}	% q_{unit}	$n_{\text{tetrahedra}}$
CL	0.57	1.50	1.06	99.9%	0.33	0.90	94.2%	39k
CCL	0.64	1.60	1.06	99.8%	0.32	0.90	93.9%	31k
CCP1	0.13	5.92	1.10	92.1%	0.00	0.72	44.6%	24k
CCP2	0.56	1.75	1.08	98.1%	0.28	0.87	83.9%	35k

Cube-Linear (CL), Cube-Cylinder-Linear (CCL), Cube-Cylinder-Polar1 (CCP1) and Cube-Cylinder-Polar2 (CCP2) – in Table 3. Applying Eq. 12 reveals that the CL case expects 39k tetrahedra, the CCL case expects 31k, the CCP1 case expects 21k and the CCP2 case expects 36k tetrahedra. Our algorithm produces meshes with a minimum of 92% of the edges in the quasi-unit range and an average element quality of at least 0.72, though the results for the CL, CCL and CCP2 cases are much better.

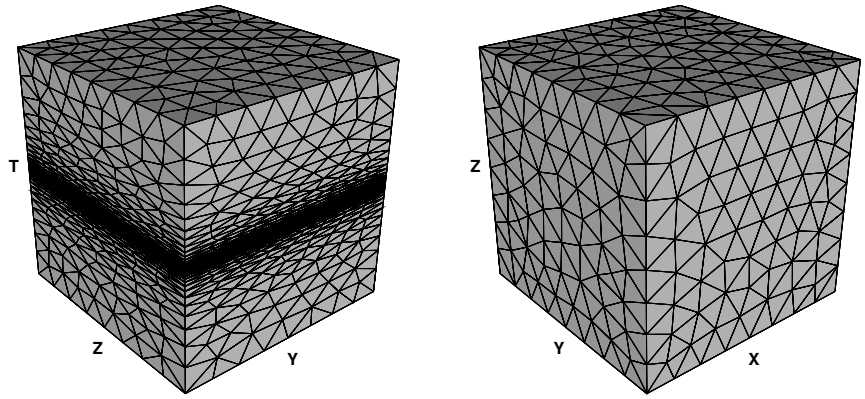
When the density control factor (ρ) is employed, the number of pentatopes overshoots the expected number of 51k by 8% for the Linear 1 case and undershoots the expected number of 818k by 0.2% for the Linear 2 case (Table 2). The overshoot in number of pentatopes is roughly 26% for the difficult Tesseract Wave case. We also provide the metric conformity results when the density control factor is not employed ($\not{\rho}$) in which the overshoot is much larger, and metric conformity (in terms of % of edges in the quasi-unit range, and average element quality) is slightly improved. We note that the use of this factor is empirical, however, the ability of the mesh generator to closely match the target element counts is important for our adaptation scheme in the following sections, which optimizes the mesh for a target DOF count. We, therefore, employ it in for the remaining results in this paper.

We also studied the valency statistics of the produced meshes. The average number of pentatopes surrounding an interior vertex is 111, 116 and 118 for the Linear 1, Linear 2 and Wave cases, respectively. The average number of pentatopes surrounding an edge is 12 (Linear 1), 14 (Linear 2) and 13 (Wave).



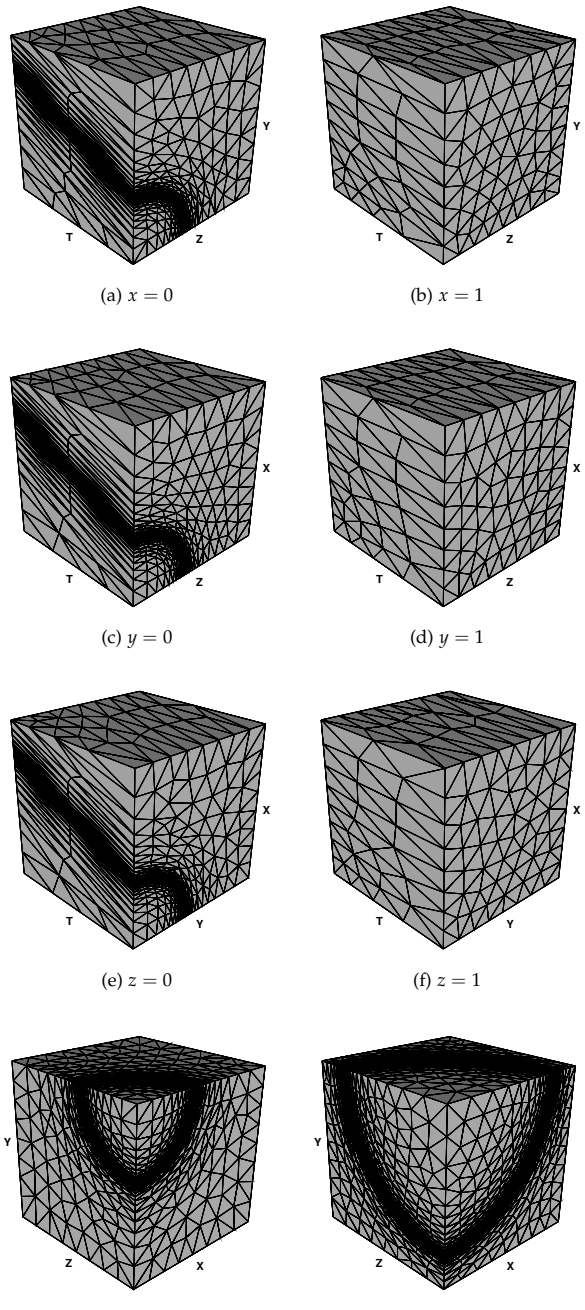
(a) $x = 0$ (b) $t = 0$

Figure 7: Meshes generated by Algorithm 1 for the Tesseract Linear 1 case show the expected refinement on a hyperplane of non-constant t (left) but a uniform mesh along a hyperplane of constant t (right).



(a) $x = 0$ (b) $t = 0$

Figure 8: Meshes generated by Algorithm 1 for the Tesseract Linear 2 case show the expected refinement on a hyperplane of non-constant t (left) but a uniform mesh along a hyperplane of constant t (right).



(a) $x = 0$ (b) $x = 1$

(c) $y = 0$ (d) $y = 1$

(e) $z = 0$ (f) $z = 1$

(g) $t = 0$ (h) $t = 1$

Figure 9: Meshes of the eight bounding cubes generated by Algorithm 1 for the Tesseract Wave case show the projection of the hypercone onto hyperplanes with non-constant temporal component (a-f) and spheres on hyperplanes with constant temporal component (g and h).

4. Applications II: optimal approximant of a scalar function

We now turn our attention to finding the optimal mesh to represent a function of four variables using high-order discontinuous Galerkin finite element discretizations. Instead of directly optimizing the mesh, we optimize the metric field and pass this optimal metric to our mesh adaptation tool. That is, we consider a continuous relaxation of the discrete mesh optimization problem, as in the work of Yano [44, 45] and Loseille [39, 46, 47, 48]. A review of this approach is provided by Alauzet and Loseille [37]. Previous methods have sought to optimize the mesh by controlling local bounds on the interpolation error for linear finite elements [49, 50, 51]. Other work includes that of Coupez, in which the length distribution tensor is used to control an edge-based interpolation error estimate [41, 52]. Coulaud and Loseille extend the continuous mesh framework to high-order interpolations [53]. Park et al. recently compare the use of the multiscale metric [54, 55, 56] which controls the ℓ^p norm of the interpolation error of a scalar solution field with output-based methods methods [44, 45, 57] that employ the continuous mesh framework of Loseille and Alauzet [39, 48].

In the continuous mesh framework, the goal of the adaptation algorithm is to find the optimal metric ($\mathbf{m}$) to minimize the ℓ^2 approximation error ($\mathcal{E}$) subject to a computational cost constraint c_t:

$$\mathbf{m}^* = \arg\min_{\mathbf{m}} \ \mathcal{E}(\mathbf{m}), \quad \text{such that } c(\mathbf{m}) \leq c_t \tag{18}$$

where $\mathcal{E}$ is the sum of the error over each element of the computational domain Ω:

$$\mathcal{E} = \sqrt{\int_\Omega \left(u - u_{h,p}\right)^2 \mathrm{d}\mathbf{x}} = \sqrt{\sum_{\boldsymbol{\kappa}\in\mathcal{M}} \eta(\boldsymbol{\kappa})} = \sqrt{\sum_{\boldsymbol{\kappa}\in\mathcal{M}} \int_{\boldsymbol{\kappa}} \left(u - u_{h,p}\right)^2 \mathrm{d}\mathbf{x}} \tag{19}$$

where u is a prescribed analytic function and $u_{h,p} \in V_{h,p}$, a discontinuous

finite element space

$$V_{h,p} = \left\{ v \in \ell^2(\Omega) : v_{h,p} \circ \varphi_q(\boldsymbol{\kappa}) \in \mathcal{P}^p(\boldsymbol{\kappa}_0), \forall\, \boldsymbol{\kappa} \in \mathcal{M} \right\}, \tag{20}$$

where $\varphi_q(\boldsymbol{\kappa})$ is the q-th order diffeomorphic mapping from physical element $\boldsymbol{\kappa}$ to master element $\boldsymbol{\kappa}_0$ and $\mathcal{P}^p(\boldsymbol{\kappa}_0)$ denotes the complete p-th order polynomial space on the reference element $\boldsymbol{\kappa}_0$. We study straight-sided pentatopes in this work, so $q = 1$.

The discrete solution $u_{h,p}$ is obtained by minimizing the square of the ℓ^2 projection error, i.e.

$$u_{h,p} = \underset{v_{h,p} \in V_{h,p}}{\arg\inf} \int_\Omega (u - v_{h,p})^2 \, \mathrm{d}\mathbf{x}. \tag{21}$$

For discontinuous Galerkin discretizations of order p, the computational cost can be computed by accumulating the volume mapped by the metric field over each element – recall Eq. 10:

$$c(\mathbf{m}) = c_p \sum_{\boldsymbol{\kappa} \in \mathcal{M}} v_{\mathbf{m}}(\boldsymbol{\kappa}) = c_p \sum_{\boldsymbol{\kappa} \in \mathcal{M}} \int_{\boldsymbol{\kappa}} \sqrt{\det \mathbf{m}(\mathbf{x})} \, \mathrm{d}\mathbf{x}, \tag{22}$$

where $c_p = (p+1)(p+2)(p+3)(p+4)/24$ for pentatopal meshes, which represents the number of basis functions, and hence DOF, associated with each discontinuous element.

4.1. Mesh optimization via error sampling and synthesis

Our mesh adaptation algorithm is an extension of the Mesh Optimization via Error Sampling and Synthesis (MOESS) algorithm of Yano [44] to four dimensions combined with the metric field optimization of Kudo [58]. The localization of the elemental error available using the discontinuous Galerkin finite element discretization enables the construction of local (elemental) error models which relate the error to the implied metric of the sampled configurations.

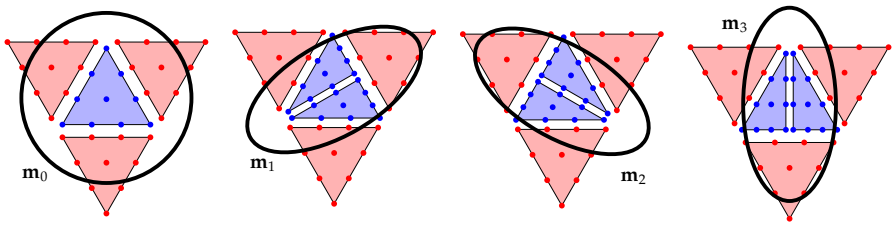


Figure 10: Sketch of the error-metric sampling procedure used in MOESS (in $2d$). For each edge split, the solution (and error) is recalculated in the blue triangles but frozen in the red triangles. The black line represents the implied metric of the split configuration.

Each element, $\boldsymbol{\kappa}_0$, can then be split and the error over each child element can be recalculated. Our local sampling method for discontinuous Galerkin discretizations consists of extracting the parent simplex along with its immediate neighbors (opposite each facet). One by one, each edge of the parent simplex is then divided at its midpoint and the solution is recalculated within the local split mesh. During this solution process, Dirichlet boundary conditions are applied to the DOF of the neighboring simplices such that only the solution of the split elements of the original parent are recomputed. The edge split configurations used for $n = 2d$ are shown in Fig. 10. In $4d$, ten edge splits are used (for each edge of a pentatope) which is needed to fully determine the error model in terms of the ten unique metric tensor entries.

The implied metric associated with each split configuration is taken as the log-Euclidean mean [43] of the implied metrics of the split elements and the error introduced by the split elements is summed to complete the set of n_s metric-error pairs: $\{\mathbf{m}_i, \eta_{\boldsymbol{\kappa}_i}\}$.

The metric-error pairs are then used to construct a model of the error as a function of the metric field. This model is constructed in terms of the *step matrix*, $\mathbf{s}$, which represents the difference between two metric tensors. In the context of the discrete set of sampled data, these tensors are computed from $\mathbf{m}_{\boldsymbol{\kappa}_i}$ and $\mathbf{m}_{\boldsymbol{\kappa}_0}$ as [43]

$$\mathbf{s}_{\boldsymbol{\kappa}_i} = \log\left(\mathbf{m}_{\boldsymbol{\kappa}_0}^{-1/2}\mathbf{m}_{\boldsymbol{\kappa}_i}\mathbf{m}_{\boldsymbol{\kappa}_0}^{-1/2}\right), \quad i = 1, \ldots, n_s. \tag{23}$$

A linear model is then constructed from the n_s metric-error samples,

$$f_{\boldsymbol{\kappa}}(\mathbf{s}) \equiv \log\left(\eta(\mathbf{s})/\eta_0\right) = \mathbf{r}_{\boldsymbol{\kappa}} : \mathbf{s}, \tag{24}$$

where $\mathbf{r}_{\boldsymbol{\kappa}}$ is referred to as the *rate matrix*. A linear regression of the sampled data is used to determine $\mathbf{r}_{\boldsymbol{\kappa}}$.

Over an element $\boldsymbol{\kappa}$, the resulting local error model can finally be written in terms of the average element step matrix $\mathbf{s}_{\boldsymbol{\kappa}}$ as

$$\eta_{\boldsymbol{\kappa}}(\mathbf{s}_{\boldsymbol{\kappa}}) = \eta_{\boldsymbol{\kappa},0} \exp(\mathrm{tr}(\mathbf{r}_{\boldsymbol{\kappa}}\mathbf{s}_{\boldsymbol{\kappa}})) + \frac{p}{n}\left(||\widetilde{\mathbf{s}}_{\boldsymbol{\kappa}}||_F^2 + \sum_{e\in\mathcal{E}(\kappa)} ||\widetilde{\mathbf{s}}_{e_1} - \widetilde{\mathbf{s}}_{e_2}||_F^2\right) \tag{25}$$

where $\widetilde{\mathbf{s}}_{(\cdot)}$ denotes the trace-free portion of the step matrix and $||\cdot||_F$ is the Frobenius norm. The Frobenius norm penalties in Eq. 25 are introduced to control the trace-free portion of the step matrix.

In general, we optimize the vertex-valued step matrices, therefore, the average step matrix over the element $\mathbf{s}_\kappa$ is computed from

$$\mathbf{s}_\kappa = \frac{1}{|\mathcal{V}(\kappa)|} \sum_{\nu\in\mathcal{V}(\kappa)} \mathbf{s}_\nu,$$

where $\mathcal{V}(\kappa) = n+1$ ($n = 4$ in $4d$) is the number of vertices in a simplicial element κ. The error in the mesh is then the sum of the local element contributions. Eq. 18 is then solved using the elemental error models of Eq. 25 with the optimization procedure of Kudo [58].

The optimized metric is then passed to our metric-conforming mesh adaptation tool. An outline of the adaptation procedure is given in Algorithm 2. The initial mesh is a Kuhn-Freudenthal triangulation with three vertices in each of the four coordinate directions. One hundred adaptation iterations are used to ensure the optimal mesh is achieved, though only twenty iterations are usually needed to reach this mesh (for the cases studied here).

adaptError

input: $\mathcal{M}_0$, c_t, p, u, `maxIter`
output: $\mathcal{M}^*$

$\mathcal{M} \leftarrow \mathcal{M}_0$
for `iter` = 1,...,`maxIter`
  $u_{h,p} \leftarrow$ project u onto solution space (order p) on current mesh $\mathcal{M}$
  $\mathbf{m} \leftarrow$ **moess**$(\mathcal{M}, u_{h,p}, c_t, p)$
  $\mathcal{M} \leftarrow$ **adaptMesh**$(\mathcal{M}, \mathbf{m})$ ▷ call mesh adaptation tool
$\mathcal{M}^* \leftarrow \mathcal{M}$

Algorithm 2: Adaptation algorithm to compute the optimal ℓ^2 approximant of a $4d$ function u with a target computational cost c_t, polynomial order p of the discrete solution. The algorithm starts from an initial mesh $\mathcal{M}_0$ and performs `maxIter` adaptation iterations to produce the optimal mesh $\mathcal{M}^*$.

4.2. Boundary layer

The first function we consider is an extension of the regularized boundary layer studied by Yano [44] to four dimensions:

$$u(x,y,z,t) = \exp(-x/\epsilon) + \frac{\beta_y}{(p+1)!}y^{p+1} + \frac{\beta_z}{(p+1)!}z^{p+1} + \frac{\beta_t}{(p+1)!}t^{p+1}. \quad (26)$$

The first term causes a strong gradation in the x direction whereas the remaining three regularization terms ensure the aspect ratios in the y, z and t directions remain bounded.

The mesh which best approximates the function in Eq. 26 has an optimal mesh grading (h_x) perpendicular to the wall ($x = 0$) with [44]

$$h_x = h_{x,0}\exp(k_{h_x}x),\ k_{h_x} = \frac{2p+5}{\epsilon(p+1)(2p+6)}, \quad (27)$$

where $h_{x,0}$ is a constant determined by the computational cost constraint. The aspect ratio distributions, a_i, in the three remaining directions are

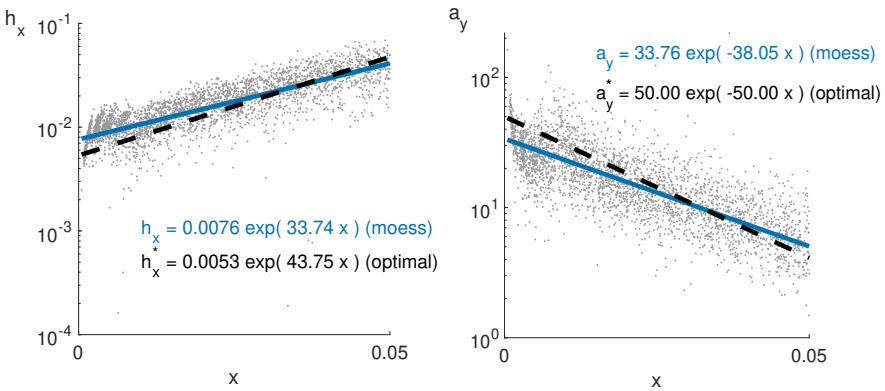
$$a_i = a_{i,0}\exp(k_{a_i}x),\ a_{i,0} = \frac{1}{\epsilon\beta_i^{\frac{1}{p+1}}},\ k_{a_i} = -\frac{1}{\epsilon(p+1)},\ i = y,z,t. \quad (28)$$

Here, $\epsilon = 0.01$, $\beta_y = 2^{p+1}$, $\beta_z = 4^{p+1}$ and $\beta_t = 6^{p+1}$.

We restrict our attention to the discontinuous Galerkin (dG) solution spaces and study linear ($p = 1$) and quadratic ($p = 2$) polynomial orders. To verify the optimality of the produced meshes, consider the mesh size and aspect ratios obtained near the $x = 0$ wall. Eqs. 27 and 28 suggest the mesh size and aspect ratios should be linear in x versus $\log(h_x)$ and x versus $\log(a_i)$ ($i = y, z, t$). Near the wall, the mesh sizes are computed directly from the diagonal entries of the implied metric of each pentatope. That is, $h_i(\kappa) \approx (\mathbf{m}_\kappa)_{i,i}^{-1/2}$ ($i = 1, 2, 3, 4$, representing x, y, z and t directions, respectively). The aspect ratios are then $a_i = h_i/h_x$ ($i = y, z, t$). For both linear and quadratic polynomial bases, the distributions of the mesh size and aspect ratios for the 512k-optimized meshes are shown in Fig. 11. A linear regression of the size and aspect ratios in $x - \log(h_x)$ and $x - \log(a_i)$ ($i = y, z, t$) spaces shows the distributions are well-aligned with the analytic mesh distributions (shown in dashed). Table 4 tabulates the full set of regression coefficients for all target DOF with both $p = 1$ and $p = 2$ discretizations. For $p = 1$, better alignment with the analytic values for both the wall aspect ratios and gradings away from the wall is obtained as the number of DOF is increased. For $p = 2$, the wall aspect ratios are all fairly good and an improvement is seen in the gradings as the number of target DOF is increased.

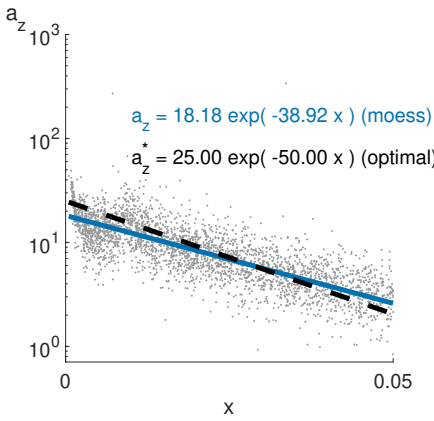
The meshes at $t = 0$ obtained at 512k for both $p = 1$ and $p = 2$ polynomial orders are shown in Fig. 12. The meshes at constant x-hyperplanes are omitted from this text since they exhibit the least anisotropy – there is little gradation in the solution in the y, z or t directions. All other bounding cubes with a variation in x effectively resolve the boundary layer with anisotropic simplices near the $x = 0$ boundary.

The ℓ^2 error in the solution and approximate mesh size, $h \approx \sqrt[4]{\text{DOF}}$, from the last five adaptation iterations are plotted in the circles of Fig. 13 and these values from the last two target DOF requests are fit in $\log h - \log \mathcal{E}$ ($\mathcal{E}$ here being the exact ℓ^2 error) space to estimate the rate of convergence. The boundary layer function of Eq. 26 is four-dimensional and we expect an asymptotic

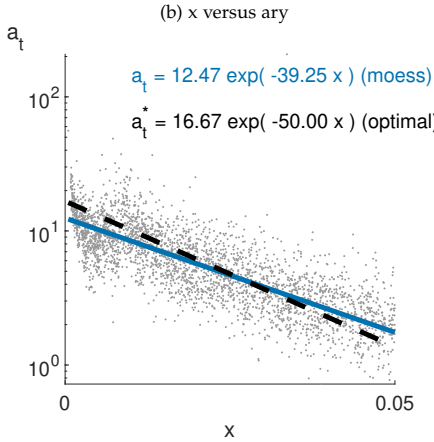


(a) x versus h

(b) x versus ary



(c) x versus arz



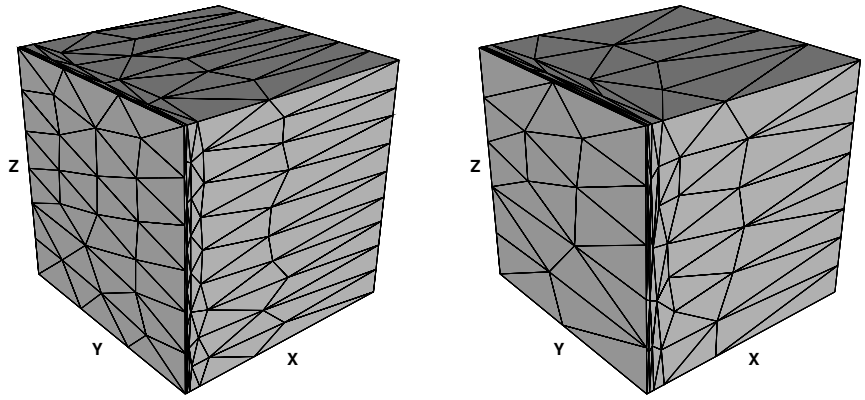
(d) x versus art

Figure 11: Mesh size and aspect ratio distributions for the boundary layer error control case ($p = 1$ and $p = 2$) are in agreement with analytic distributions. Only the distributions for the 512k-optimized meshes with $\delta = 0.01$ are shown. The remaining distribution statistics are tabulated in Table 4.

Table 4: Mesh size and aspect ratio distributions for the boundary layer error control case ($p = 1$ and $p = 2$) are in agreement with analytic distributions (here, $\delta = 0.01$).

$p = 1$	$h_{x,0}$	$h^*_{x,0}$	k_{h_x}	$a_{y,0}$	k_{a_y}	$a_{z,0}$	k_{a_z}	$a_{t,0}$	k_{a_t}
Analytic	$h^*_{x,0}$	-	43.75	50.00	-50.00	25.00	-50.00	16.67	-50.00
64k	0.0155	0.0090	25.14	25.52	-26.57	14.32	-26.84	9.59	-26.11
128k	0.0121	0.0075	28.39	29.46	-31.14	16.02	-31.79	10.75	-31.44
256k	0.0095	0.0063	31.29	32.15	-35.22	17.09	-35.92	11.76	-35.58
512k	0.0076	0.0053	33.74	33.76	-38.05	18.18	-38.92	12.47	-39.25

$p = 2$	$h_{x,0}$	$h^*_{x,0}$	k_{h_x}	$a_{y,0}$	k_{a_y}	$a_{z,0}$	k_{a_z}	$a_{t,0}$	k_{a_t}
Analytic	$h^*_{x,0}$	-	30.00	50.00	-33.33	25.00	-33.33	16.67	-33.33
64k	0.0146	0.0125	25.46	38.06	-24.69	22.41	-25.96	15.39	-26.60
128k	0.0119	0.0105	24.56	42.92	-28.12	21.51	-27.69	14.55	-28.50
256k	0.0100	0.0088	27.36	40.45	-29.80	22.11	-30.55	14.64	-30.14
512k	0.0081	0.0074	28.00	44.13	-31.04	22.06	-30.68	15.22	-31.74



(a) $p = 1$ (b) $p = 2$

Figure 12: Meshes along the $t = 0$ hyperplane generated by Algorithm 2 for the boundary layer case optimized for $p = 1$ and $p = 2$ 512k-DOF solutions ($\delta = 0.01$).

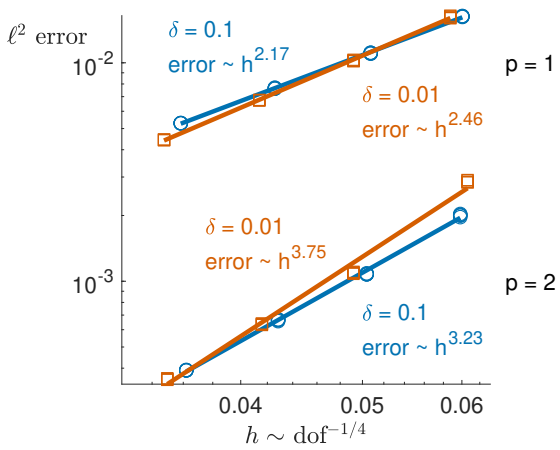


Figure 13: Convergence of the ℓ^2 error in the boundary layer solution for both $p = 1$ and $p = 2$ discretizations as the mesh is refined with two boundary layer thicknesses: $\delta = 0.1$ (blue, circles) and $\delta = 0.01$ (red, squares).

convergence rate of approximately h^{p+1} [44, 59, 60]. This is certainly observed in the rates obtained for both $p = 1$ and $p = 2$ as seen in Fig. 13 though the rates are slightly higher than expected. This may be due to the fact that the meshes are still in the pre-asymptotic range and would likely approach rates of $\mathcal{E} \sim h^{p+1}$ should finer mesh resolutions be studied. With $\delta = 0.1$, the convergence rates are closer to the theoretical asymptotic ones (see the blue lines in Fig. 13).

4.3. *Expanding spherical wave*

Now, consider a function modeling the expansion of a spherical wave, similar to the Tesseract Wave case of the previous section:

$$u(\mathbf{x}, t) = k_0 \exp(-\alpha t) \exp\left(-k_1 (r(t) - ||\mathbf{x}||)^2\right), \quad \mathbf{x} \in \mathbb{R}^3,\ t \in [0, 1] \tag{29}$$

with $r(t) = r_0 + v_s t$, $\alpha = 1$, $k_0 = 1$, $k_1 = 200$, $v_s = 0.7$, $r_0 = 0.4$. The strength of the wave, initially k_0, decays exponentially in time at a rate of α. The parameter k_1 controls the width of wave strength δ about the increasing radius $r(t)$. In particular, since Eq. 29 is a normal distribution; roughly 99.7% of u will lie within three standard deviations of the nominal wave radius $r(t)$.

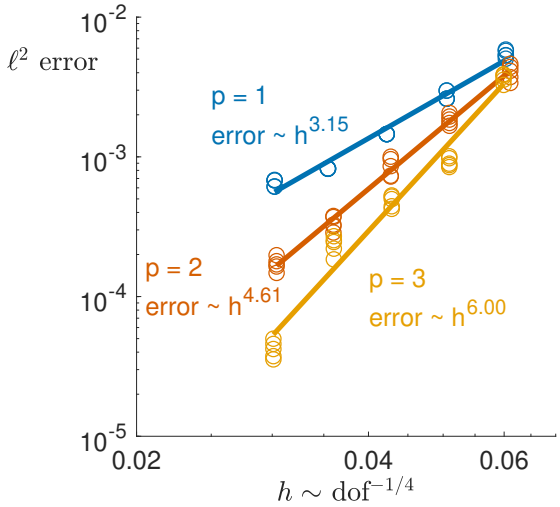


Figure 14: Convergence of the ℓ^2 error in the spherical wave solution for $p = 1$, $p = 2$ and $p = 3$ discretizations as the mesh is refined.

Table 5: Metric conformity at final adaptation iteration for 1024k meshes adapted to the spherical wave solution.

	$\ell_{\min}$	$\ell_{\max}$	ℓ_{avg}	% ℓ_{unit}	$q_{\min}$	q_{avg}	% q_{unit}	$n_{\text{pentatope}}$	n_{dof}
$p = 1$	0.43	2.21	1.11	94.0%	0.11	0.75	35.0%	237k	1.186M
$p = 2$	0.46	1.96	1.10	95.7%	0.23	0.76	38.5%	78k	1.170M
$p = 3$	0.47	1.98	1.10	95.1%	0.20	0.75	84.8%	35k	1.239M

This thickness can be approximated as

$$\delta \approx 3\sigma = \frac{3}{\sqrt{2k_1}}. \tag{30}$$

For this problem, $\delta \approx 0.15$ and a total of 2δ should be visibly refined about the expanding sphere by the adaptation algorithm.

The $p = 1$, $p = 2$ and $p = 3$ optimized meshes along $t = 0$, $t = 1$, and $x = 0$ and $x = 1$ hyperplanes (cubes) obtained for the target DOF request of 1024k are shown in Fig. 15. The expected three-dimensional cones are seen along hyperplanes with non-constant temporal component whereas the initial sphere (with radius $r_0 = 0.4$) and final sphere (with radius $r_f = 1.1$) are correctly obtained at constant t hyperplanes. Note that the expected width ($2\delta \approx 0.3$) of the solution around the expanding wave is seen since the width

	64k	128k	256k	512k	1024k
$p = 1$	1.10e+03	5.30e+02	2.25e+03	5.68e+03	3.67e+03
$p = 2$	5.02e+01	1.71e+02	2.50e+02	3.11e+02	9.54e+02
$p = 3$	4.65e+01	5.26e+01	1.57e+02	1.80e+02	4.11e+02

Table 6: Maximum aspect ratios for the optimized meshes (at various DOF) with $p = 1$, $p = 2$ and $p = 3$ discretizations for the spherical wave error control case.

of the mesh resolution is approximately one third in each spatial direction. Furthermore, the stretching in the direction of the wave is evident; on average, two elements are needed in the temporal direction.

The metric conformity statistics for the meshes optimized at 1024k DOF are provided in Table 5. At least 94% of the edges are within the quasi-unit range, which is very good. The number of pentatopes of quality greater than 0.8 is low for the cases with larger meshes and higher anisotropy (see the $p = 1$ and $p = 2$ results).

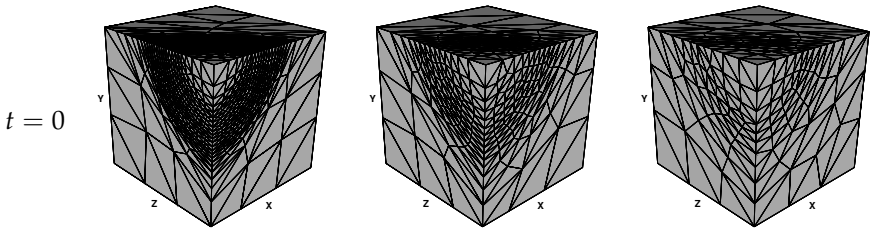
For the spherical wave case, the fitted convergence rates are even more accelerated than h^{p+1} which may be due to the fact that the function is effectively two-dimensional in an $r - t$ coordinate system. Thus the convergence rate should approach $h^{2(p+1)}$, though, again the results obtained from the simulations exhibit pre-asymptotic behavior.

The maximum aspect ratios for each mesh (64k-1024k) optimized for $p = 1$, $p = 2$ and $p = 3$ discretizations are tabulated in Table 6 which effectively demonstrate a maximum aspect ratio of over $10^3 : 1$ with our algorithm.

5. Perspectives

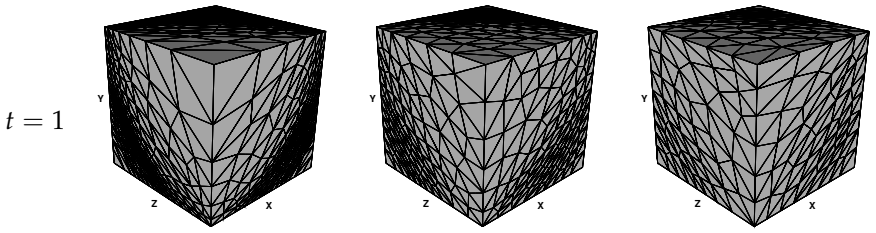
This paper presented the first four-dimensional anisotropic mesh adaptation capability. Our mesh adaptation algorithm builds upon a dimension-independent cavity framework for performing local mesh modifications. Anisotropy was achieved through the use of a background metric field.

First, anisotropic four-dimensional meshes were obtained by prescribing an analytic metric field on a background mesh. We introduced analytic metric fields consisting of the Tesseract Linear (coarse and fine) and Tesseract Wave



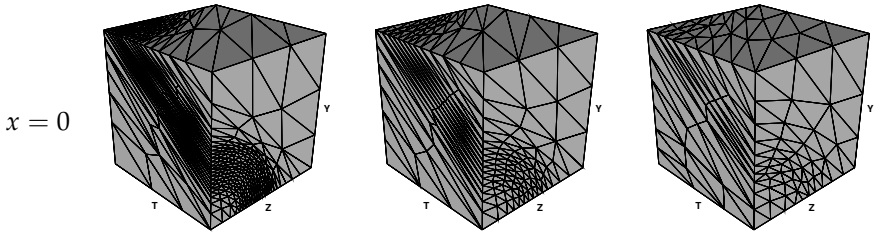
(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

$t = 1$



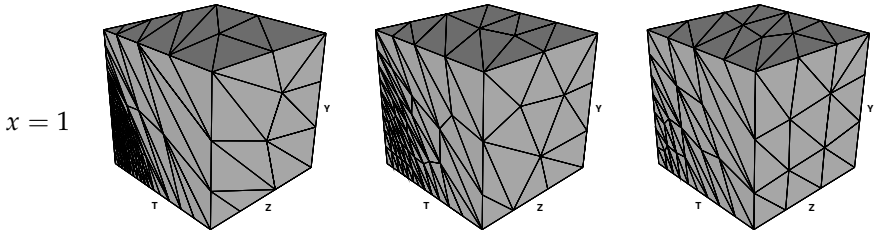
(d) $p = 1$ (e) $p = 2$ (f) $p = 3$

$x = 0$



(g) $p = 1$ (h) $p = 2$ (i) $p = 3$

$x = 1$



(j) $p = 1$ (k) $p = 2$ (l) $p = 3$

Figure 15: Meshes generated by Algorithm 2 for the spherical wave case optimized for $p = 1$, $p = 2$ and $p = 3$ 1024k-DOF solutions. The initial and final spheres are observed at $t = 0$ and $t = 1$ respectively. The expected projection of a hypercone onto a plane with non-constant t is also observed at $x = 0$ and $x = 1$.

cases. The resulting edge lengths, pentatope qualities and total pentatope counts indicate that good metric conformity is achieved with our algorithm: roughly 90% of the edges are in the quasi-unit range, average element quality as at least 0.7 and the pentatope counts are within 10% of the expected numbers. Furthermore, our algorithm performs well on $3d$ benchmark cases proposed by the Unstructured Grid Adaptation Working Group.

Next, we produced meshes that adapt to the exact ℓ^2 error between a prescribed function and its discrete representation in a polynomial basis. The correct mesh size and aspect ratio distributions were obtained for a function with a rapid variation near one of the boundaries of the domain. Similarly, the ℓ^2 error control of a four-dimensional function simulating the expansion of a spherical wave was effective in resolving the strength of the wave as it propagated in time. In particular, the meshes exhibited a significant amount of clustering within 99.7% of the wave radius and metric conformity was very good for this case. The convergence of the ℓ^2 error in the solution with decreasing mesh size approached the expected theoretical asymptotic rate.

The adaptation algorithm is currently restricted to a serial implementation, the performance of which is acceptable for the problem sizes studied in this work. One iteration of the mesh adaptation algorithm takes approximately $15 - 20$ minutes for the larger four-dimensional meshes with $800 - 900$k pentatopes. Future work consists of parallelizing the mesh adaptation components using both shared- and distributed-memory approaches.

Furthermore, we only studied discontinuous Galerkin finite element discretizations, which suffers from a high per-element DOF cost. It would be worthwhile to study alternative finite element discretizations, such as continuous ones to reduce the computational cost of the numerical simulation. This is particularly true when moving towards the solution of partial differential equations, such as the Navier-Stokes equations.

Acknowledgements

This work was funded by the CAPS project: AFRL Contract FA8050-14-C-2472: *CAPS: Computational Aircraft Prototype Syntheses* with Dean Bryson as technical monitor. The authors would also like to thank Dr. Marshall Galbraith for his help developing the software in this work.

References

[1] M. Yano, J. M. Modisette, D. L. Darmofal, The Importance of Mesh Adaptation for Higher-Order Discretizations of Aerodynamic Flows, in: 20th AIAA Computational Fluid Dynamics Conference, 3852, 2011.

[2] J. Slotnick, A. Khodadoust, J. Alonso, D. L. Darmofal, W. Gropp, E. Lurie, D. J. Mavriplis, CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences, Technical Report NASA/CR-2014-218178, 2014.

[3] J. T. Oden, A General Theory of Finite Elements II. Applications, International Journal for Numerical Methods in Engineering 1 (1969) 247–259.

[4] J. Argyris, D. Scharpf, Finite Elements in Time and Space, Nuclear Engineering and Design 10 (1969) 456 – 464.

[5] I. Fried, Finite-Element Analysis of Time-Dependent Phenomena, AIAA Journal 7 (1969) 1170 – 1173.

[6] M. Behr, Simplex Space-Time Meshes in Finite Element Simulations, International Journal for Numerical Methods in Fluids 57 (2008) 1421–1434.

[7] A. Üngör, A. Sheffer, Tent-Pitcher: A Meshing Algorithm for Space-Time Discontinuous Galerkin Methods, in: Proceedings of the 9th International Meshing Roundtable, 2000, pp. 111–122.

[8] J. Erickson, D. Guoy, J. Sullivan, A. Üngör, Building Space-Time Meshes over Arbitrary Spatial Domains, Engineering with Computers 20 (2005) 342–353.

[9] A. D. Mont, Adaptive Unstructured Spacetime Meshing for Four-Dimensional Spacetime Discontinuous Galerkin Finite Element Methods, Master's thesis, University of Illinois at Urbana-Champaign, 2011.

[10] S. Thite, Adaptive Spacetime Meshing for Discontinuous Galerkin Methods, Computational Geometry 42 (2007) 20 – 44.

[11] K. J. Fidkowski, Y. Luo, Output-Based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations, Journal of Computational Physics 230 (2011) 5753 – 5773.

[12] K. J. Fidkowski, Output-Based Space-Time Mesh Optimization for Unsteady Flows Using Continuous-in-Time Adjoints, Journal of Computational Physics 341 (2017) 258–277.

[13] W. Bangerth, R. Rannacher, Finite Element Approximation of the Acoustic Wave Equation: Error Control and Mesh Adaptation, East-West Journal of Numerical Mathematics 7 (1999) 263–282.

[14] W. Bangerth, M. Geiger, R. Rannacher, Adaptive Galerkin Finite Element Methods for the Wave Equation, Computational Methods in Applied Mathematics 10 (2010) 3–48.

[15] R. Hartmann, Adaptive FE Methods for Conservation Equations, in: H. Freistühler, G. Warnecke (Eds.), Hyperbolic Problems: Theory, Numerics, Applications, volume 141 of *International Series of Numerical Mathematics*, 2001, pp. 495–503.

[16] S. Jayasinghe, An Adaptive Space-Time Discontinuous Galerkin Method for Reservoir Flows, PhD thesis, Massachusetts Institute of Technology, 2018.

[17] S. Jayasinghe, D. L. Darmofal, N. K. Burgess, M. C. Galbraith, S. R. Allmaras, A Space-Time Adaptive Method for Reservoir Flows: Formulation and One-Dimensional Application, Computational Geosciences 22 (2018) 107–123.

[18] P. Foteinos, N. Chrisochoides, 4D Space-Time Delaunay Meshing for Medical Images, in: Proceedings of the 22nd International Meshing Roundtable, 2014.

[19] G. Belda-Ferrín, A. Gargallo-Peiró, X. Roca, Local Bisection for Conformal Refinement of Unstructured 4D Simplicial Meshes, in: X. Roca, A. Loseille (Eds.), Proceedings of the 27th International Meshing Roundtable, volume 127 of *Lecture Notes in Computational Science and Engineering*, Springer, 2019, pp. 229–247.

[20] P. C. Caplan, R. Haimes, D. L. Darmofal, M. C. Galbraith, Anisotropic Geometry-Conforming *d*-Simplicial Meshing via Isometric Embeddings, Procedia Engineering 203 (2017) 141–153. 26th International Meshing Roundtable.

[21] A. Loseille, F. Alauzet, V. Menier, Unique Cavity-Based Operator and Hierarchical Domain Partitioning for Fast Parallel Generation of Anisotropic Meshes, Computer-Aided Design 85 (2017) 53 – 67.

[22] T. Michal, J. Krakos, Anisotropic Mesh Adaptation through Edge Primitive Operations, in: 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, 159, 2012.

[23] M. A. Park, D. L. Darmofal, Parallel Anisotropic Tetrahedral Adaptation, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, 917, 2008.

[24] P. George, H. Borouchaki, P. Laug, An Efficient Algorithm for 3D Adaptive Meshing, Advances in Engineering Software 33 (2002) 377 – 387.

[25] G. Rokos, G. J. Gorman, J. Southern, P. H. J. Kelly, A Thread-Parallel Algorithm for Anisotropic Mesh Adaptation, Technical Report, 2013. `arXiv:arXiv:1308.2480`.

[26] D. A. Ibanez, Conformal Mesh Adaptation on Heterogeneous Supercomputers, PhD thesis, Rensselaer Polytechnic Institute, 2016.

[27] C. Dobrzynski, P. Frey, Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations, in: Proceedings of the 17th International Meshing Roundtable, 2008, pp. 177–194.

[28] T. Coupez, Génération de Maillage et Adaptation de Maillage par Optimisation Locale, Revue Européenne des Éléments Finis 9 (2000) 403–423.

[29] C. Gruau, Metric Generation for Anisotropic Mesh Adaptation with Numerical Applications to Material Forming Simulation, PhD thesis, École Nationale Supérieure des Mines de Paris, 2005.

[30] P. Tremblay, 2-D, 3-D and 4-D Anisotropic Mesh Adaptation for the Time-Continuous Space-Time Finite Element Method with Applications to the Incompressible Navier-Stokes Equations, PhD thesis, University of Ottawa, 2007.

[31] C. Gruau, T. Coupez, 3D Tetrahedral, Unstructured and Anisotropic Mesh Generation with Adaptation to Natural and Multidomain Metric, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4951 − 4976.

[32] B. Lévy, Robustness and Efficiency of Geometric Programs: The Predicate Construction Kit, Computer-Aided Design 72 (2016) 3–12.

[33] J. R. Shewchuk, Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates, Discrete & Computational Geometry 18 (1996) 305–363.

[34] T. Coupez, H. Digonnet, R. Ducloux, Parallel Meshing and Remeshing, Applied Mathematical Modelling 25 (2000) 153–175.

[35] H. W. Kuhn, Simplicial Approximation of Fixed Points, Proceedings of the National Academy of Science 61 (1968) 1238–1242.

[36] P. C. Caplan, Four-dimensional Anisotropic Mesh Adaptation for Spacetime Numerical Simulations, PhD thesis, Massachusetts Institute of Technology, 2019.

[37] F. Alauzet, A. Loseille, A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics, Computer-Aided Design 72 (2016) 13–39.

[38] D. Ibanez, N. Barral, J. Krakos, A. Loseille, T. Michal, M. Park, First Benchmark of the Unstructured Grid Adaptation Working Group, Procedia Engineering 203 (2017) 154 – 166. 26th International Meshing Roundtable.

[39] A. Loseille, F. Alauzet, Continuous Mesh Framework Part I: Well-Posed Continuous Interpolation Error, SIAM Journal on Numerical Analysis 49 (2011) 38–60.

[40] A. Loseille, Metric-Orthogonal Anisotropic Mesh Generation, Procedia Engineering 82 (2014) 403 – 415. 22nd International Meshing Roundtable.

[41] T. Coupez, Metric Construction by Length Distribution Tensor and Edge Based Error for Anisotropic Adaptive Meshing, Journal of Computational Physics 230 (2011) 2391–2405.

[42] F. J. Bossen, P. S. Heckbert, A Pliant Method for Anisotropic Mesh Generation, in: Proceedings of the 5th International Meshing Roundtable, 1996, pp. 63–74.

[43] V. Arsigny, P. Fillard, X. Pennec, N. Ayache, Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors, Magnetic Resonance in Medicine 56 (2006) 411–421.

[44] M. Yano, An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes, PhD thesis, Massachusetts Institute of Technology, 2012.

[45] M. Yano, D. L. Darmofal, An Optimization-Based Framework for Anisotropic Simplex Mesh Adaptation, Journal of Computational Physics 231 (2012) 7626–7649.

[46] A. Loseille, F. Alauzet, Optimal 3D Highly Anisotropic Mesh Adaptation Based on the Continuous Mesh Framework, in: Proceedings of the 18th International Meshing Roundtable, Springer Berlin Heidelberg, 2009, pp. 575–594.

[47] A. Loseille, A. Dervieux, F. Alauzet, On 3D Goal-Oriented Anisotropic Mesh Adaptation Applied to Inviscid Flows in Aeronautics, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 1067, 2010.

[48] A. Loseille, F. Alauzet, Continuous Mesh Framework Part II: Validations and Applications, SIAM Journal on Numerical Analysis 49 (2011) 61–86.

[49] L. Formaggia, S. Perotto, New Anisotropic A Priori Error Estimates, Numerische Mathematik 89 (2001) 641–667.

[50] W. Huang, Metric Tensors for Anisotropic Mesh Generation, Journal of Computational Physics 204 (2005) 633 – 665.

[51] P. Frey, F. Alauzet, Anisotropic Mesh Adaptation for CFD Computations, Computer Methods in Applied Mechanics and Engineering 194 (2005) 5068 – 5082.

[52] T. Coupez, G. Jannoun, J. Veysset, E. Hachem, Edge-Based Anisotropic Mesh Adaptation for CFD Applications, in: X. Jiao, J.-C. Weill (Eds.), Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 567–583.

[53] Very High Order Anisotropic Metric-Based Mesh Adaptation in 3D, Procedia Engineering 163 (2016) 353–365.

[54] A. Loseille, A. Dervieux, P. Frey, F. Alauzet, Achievement of Global Second Order Mesh Convergence for Discontinuous Flows with Adapted Unstructured Meshes, AIAA 4186, 2007.

[55] F. Alauzet, A. Loseille, High-Order Sonic Boom Modeling Based on Adaptive Methods, Journal of Computational Physics 229 (2010) 561 – 593.

[56] F. Alauzet, Size Gradation Control of Anisotropic Meshes, Finite Elements in Analysis and Design 46 (2010) 181–202.

[57] A. Belme, F. Alauzet, A. Dervieux, An A Priori Anisotropic Goal-Oriented Error Estimate for Viscous Compressible Flow and Application to Mesh Adaptation, Journal of Computational Physics 376 (2019) 1051 – 1088.

[58] J. Kudo, Robust Adaptive High-Order RANS Methods, Master's thesis, Massachusetts Institute of Technology, Computation for Design and Optimization, 2014.

[59] P. Houston, E. H. Georgoulis, E. Hall, Adaptivity and A Posteriori Error Estimation for DG Methods on Anisotropic Meshes, in: Proceedings of the International Conference on Boundary and Interior Layers, 2006.

[60] W. Cao, An Interpolation Error Estimate on Anisotropic Meshes in R^n and Optimal Metrics for Mesh Refinement, SIAM Journal on Numerical Analysis 45 (2007) 2368–2391.