AIAA SciTech Forum
7-11 January 2019, San Diego, California
AIAA Scitech 2019 Forum

10.2514/6.2019-1992

# Extension of local cavity operators to $3d + t$ spacetime mesh adaptation

Philip Claude Caplan [*], Robert Haimes [†], David L. Darmofal [‡] and Marshall C. Galbraith [§]
*Massachusetts Institute of Technology, Cambridge, MA, 02139, USA*

**This work extends local cavity operators to generate anisotropic four-dimensional meshes, consisting of pentatopes, for adaptive numerical simulations. The implementation details needed to address the robustness requirements in both the *geometry* and *topology* of the developed mesh generation tool are first discussed. Strict requirements on the metadata needed to track the boundary representation are imposed and the geometry hierarchy of a four-dimensional tesseract is presented. Furthermore, the schedule of the local operators is discussed in detail along with a metric limiting procedure to emulate the small changes requested of metric fields during an iteration of an adaptive numerical simulation. The $3d$ benchmark cases of the Unstructured Grid Adaptation Working Group are first used to demonstrate the capability of the developed software. Finally, anisotropic meshes are generated from two analytic metric fields in $4d$; the corresponding edge length and quality histograms are discussed.**

## I. Introduction

The CFD Vision 2030 study suggests (1) high-order discretizations and (2) mesh adaptation are two technologies needed to obtain accurate solutions to complex physical flows such that engineers can design next generation aircraft [1]. Robust mesh adaptation frameworks which target the accurate prediction of a prescribed quantity of interest (such as the drag on an aircraft) have been successfully demonstrated on a variety of spatial problems in $2d$ and $3d$.

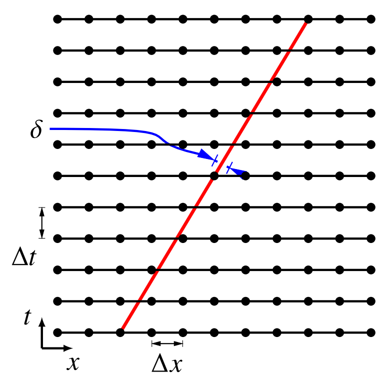
In the quest for accurate predictions of unsteady outputs, a variety of approaches exist. First, uniform refinement on a fixed spatial mesh and a fixed time step can be used to accurately compute the output of interest – see Fig. 1a. Alternatively, time can be treated as an additional dimension and the entire spatiotemporal domain can be solved to estimate the output of interest. It has been shown in $1d$ [2] that, for a feature with characteristic size $\delta$, uniform refinement in this spatiotemporal domain would require an order of $O(\delta^{-2})$ degrees of freedom (DOF) to achieve a certain level of error in the output. Furthermore, a time-slab tensor-product adaptive method, as in Fig. 1b, [3, 4] permits isotropic refinement around the characteristic feature and has been shown to reduce the number of DOF required to $O(\delta^{-1})$. The advantage of the latter method is that unstructured meshes can be used in space while a simpler time-slab approach can be used in the temporal direction. However, to truly obtain the most accurate prediction of the unsteady output of interest with the minimal DOF count, Yano demonstrates that a fully unstructured approach – see Fig. 1c – can reduce the DOF to $O(1)$ [2].

To date, mesh adaptation has been applied to either spatial problems in $2d$ or $3d$ [5] or to spatiotemporal problems in $1d$ or $2d$ [2, 6, 7]. The goal of this paper is to provide the mesh generation tool to achieve fully unstructured spacetime mesh adaptation for $3d$ spatial problems, thus providing a $4d$ anisotropic metric-conforming mesh generator.

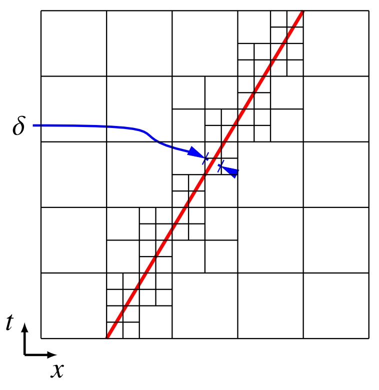
Our method of choice for generating $4d$ metric-conforming meshes is inspired by the local cavity approach of Loseille [8–10] and Coupez [11–13]. In fact, it appears Gruau is the first to discuss $4d$ meshes in the appendices of his thesis [14], however, it should be noted that the results are for Euclidean metrics and were not demonstrated in the fully anisotropic setting. The use of local cavity operators has been nonetheless very successful when applied to $3d$ mesh adaptation and is further attractive because of recent demonstrations of its use in the parallel setting [8, 15].

This paper discusses the extension of local cavity operators to produce pentatopal (4-simplices) $4d$ metric-conforming meshes. The developed approach is demonstrated on the Unstructured Grid Adaptation Working Group's (UGAWG) benchmark cases in $3d$ and finally on two anisotropic metric fields in $4d$.

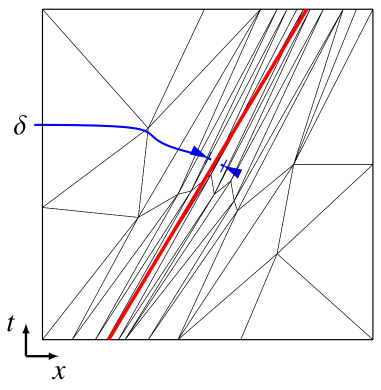
[*]Graduate Student, Department of Aeronautics & Astronautics, AIAA Student Member
[†]Principal Research Engineer, Department of Aeronautics & Astronautics, AIAA Member
[‡]Professor, Department of Aeronautics & Astronautics, AIAA Member
[§]Research Engineer, Department of Aeronautics & Astronautics, AIAA Member



(a) **Time-stepping approach**



(b) **Tensor-product approach**



(c) **Unstructured approach**

**Fig. 1 Time-stepping, tensor product and unstructured approaches to capture the propagation of an unsteady feature moving from left to right in $1d + t$.**

## A. Review of local cavity operators for anisotropic mesh adaptation

In contrast to global remeshing methods in which a new mesh is generated at each adaptation iteration, local methods start with and operate on a *valid* geometry-conforming mesh upon the application of each mesh operator. The term *geometry-conforming* is meant to imply that the tessellation of the geometric entities in the boundary representation are homeomorphic to an input description of the geometry as in, say, a BRep geometry. Given an initial $d$-simplicial mesh $\mathcal{T}_h \subset \mathbb{R}^d$, a local operation on $\mathcal{T}_h$ consists of the transformation:

$$\mathcal{T}_h^{k+1} = \mathcal{T}_h^k - \overline{C}^k(f) + \mathcal{B}^k(p) \tag{1}$$

where the superscripts represent the sequence of meshes at each application of operators $\overline{C}^k(f)$ and $\mathcal{B}^k(p)$. $\overline{C}^k(f)$ denotes the set of cavity elements about a $j$-dimensional facet $f \subset \mathcal{T}_h^k$ which is further enlarged to ensure topological and geometric validity and $\mathcal{B}^k(p)$ denotes the ball of point $p$. In particular, $\mathcal{B}^k(p) \equiv \mathcal{S}(\partial\overline{C}^k(f), p)$ where $\mathcal{S}(\cdot,\cdot)$ is the *star* operator of Coupez [11, 12]. Often $f$ is chosen from the vertices or edges of $\mathcal{T}_h^k$ [13]. Superscripts will now be dropped for brevity.

Coupez enlarges an initial set of cavity elements $C(f)$ to include those simplices in $\mathcal{T}_h$ which are in the *closure* of the vertices of the original cavity $\mathcal{N}(C(f))$. Loseille enlarges the initial cavity elements to enforce a *visibility* criterion: the simplex formed by $p \cup g$ for $g \in \partial\overline{C}$ ($p \notin g$) should exhibit a strictly positive volume. Here, we prefer the visibility check of Loseille but do not allow cavities to enlarge. Instead, edge swaps are used upon the rejection of the operator to weave out of a restrictive topology causing the rejection; these are discussed in a later section. Table 1 lists the choice of cavity $C$ and re-insertion vertex $p$ for each mesh operator. For the coordinates of the split operator, $\mathbf{x}_s = \mathbf{x}_0 + s(\mathbf{x}_1 - \mathbf{x}_0)$, $s \in [0,1]$ which may need to be projected to the geometry. Likewise, the coordinates resulting from the smoothing operator, $\widetilde{\mathbf{x}}$, may also need to be projected to the geometry. In this work, smoothing is achieved using a procedure similar to that of Bossen & Heckbert [16].

The foundational material for metric-based mesh adaptation has been extensively discussed in the literature and is omitted from the current text [17, 18] – we simply remark upon how length and quality are computed in the metric space. The length of an edge under the Riemannian metric field $\mathcal{M}$, denoted by $\ell_{\mathcal{M}}(e)$, is computed by assuming a Log-Euclidean interpolation of the metrics stored at the edge ($e$) endpoint vertices: $\mathbf{m}(e_0)$, $\mathbf{m}(e_1)$. Note in the following that the length of edge $e$ in a constant metric $\mathbf{m}_p$ is denoted by $\ell_{\mathbf{m}(p)}(e) = (\mathbf{x}_e^T \mathbf{m}_p \mathbf{x}_e)^{1/2}$ for $\mathbf{x}_e \equiv \mathbf{x}(e_1) - \mathbf{x}(e_0)$. The quality of a $d$-simplex $\kappa$ under the metric field $\mathcal{M}$ is computed to be consistent with the quality measure used by the Unstructured Grid Adaptation Working Group (UGAWG) [19]:

$$\ell_{\mathcal{M}}(e) = \ell_{\mathbf{m}(e_0)}(e)\frac{r-1}{r\log r},\ r \equiv \frac{\ell_{\mathbf{m}(e_0)}(e)}{\ell_{\mathbf{m}(e_1)}(e)}, \qquad q_{\mathcal{M}}(\kappa) = c_d \frac{\left(\sqrt{\mathbf{m}_{\max}}|\kappa|\right)^{2/d}}{\sum\limits_{e\in\kappa} \ell_{\mathbf{m}_{\max}}(e)}, \quad \mathbf{m}_{\max} = \arg\min_{v\in\mathcal{N}(\kappa)} \mathbf{m}(v),$$

where $c_d$ is a constant which normalizes the quality such that the unit equilateral simplex has unit quality. Note that $c_4 = 40\sqrt{6}/5^{\frac{1}{4}}$. The volume of the physical simplex is represented by $|\kappa|$. The goal of the mesh adaptation procedure is to create a quasi-unit mesh. That is, all edge lengths in the output mesh should ideally be in the range $[\sqrt{2}/2, \sqrt{2}]$ and the quality of all simplices should be greater than $\beta$ (often $\beta = 0.8$).

**Table 1 Choice of cavities and re-insertion vertices (with associated coordinates) for local cavity operators.**

| **operator** | $j$ | $j$-**facet** $f$ | **cavity** $C$ | **vertex** $p(\mathbf{x})$ |
|---|---|---|---|---|
| collapse | 1 | edge $e = (v_0, v_1)$ | $\mathcal{T}_h \cap v_0$ | $v_1(\mathbf{x}_1)$ |
| split | 1 | edge $e = (v_0, v_1)$ | $\mathcal{T}_h \cap e$ | $v_s(\mathbf{x}_s)$ |
| edge swap | 1 | edge $e = (v_0, v_1)$ | $\mathcal{T}_h \cap e$ | $p(\mathbf{x}_p) \in \mathcal{N}(C)$ |
| facet swap | $d-1$ | simplex $f$ | $\mathcal{T}_h \cap f$ | $p(\mathbf{x}_p) \in \mathcal{N}(C)$ |
| smooth | 0 | vertex $v$ | $\mathcal{T}_h \cap v$ | $v(\widetilde{\mathbf{x}})$ |

## II. Dimension-independent implementation

The dimension-independent implementation of the local cavity operators is successfully achieved by treating mesh operations in set notation with the `c++` standard library. As mesh generation combines the mathematical disciplines of geometry and topology, the subsequent subsections discuss the tools needed to develop a *robust* four-dimensional mesh generator in terms of these two disciplines. In addition, the boundary representation metadata and local operator scheduling are discussed.

### A. Topological robustness

In the terminology of Coupez [11, 12], the mesh needs to be a valid *mesh topology*, in which every $(d-1)$-facet of a $d$-simplex is counted either once (boundary facet) or twice (interior facet). To ensure its topology is always valid, the mesh is closed such that it is *without boundary*. Each application of a local operator should maintain a null boundary of the mesh. Practically, this is checked by storing a cache (initially empty) of those $(d-1)$-facets which occur only once and is modified when updating the simplex-to-simplex neighbour relations needed to 1) efficiently enlarge cavities and 2) compute the boundary of a set of cavity elements.

In order to close a mesh, Coupez suggests connecting every boundary facet to some fictitious vertex (call this a *ghost* vertex). However, this is not guaranteed to create a valid mesh topology. In fact, there needs to be a fictitious vertex for every closed boundary of the input mesh. Since this work is only concerned with tesseract geometries (mesh topologies with genus 0) it suffices to connect every boundary facet with a single ghost vertex. The topology of the mesh thus consists of all the original simplices combined with the set of ghost simplices on the boundary. A simplex is said to be a ghost if it contains the ghost vertex – this is important for performing geometry checks with local operators.

### B. Geometric robustness

Critical to any mesh generator is a set of geometric predicates which eliminate the possibility of producing invalid meshes during the mesh generation decision-making process. For local mesh operations (and often many other applications), the computation of simplex volume needs to be *exact*. `Triangle` [20] and `TetGen` [21] respectively use the `orient2d` and `orient3d` predicates [22]. Here, the `orient4d` predicate is implemented to compute the sign of the volume of an input pentatope. The Predicate Construction Kit (PCK) [23] is used to alleviate the expertise needed in tracking numerical roundoff errors [22]. The code used to compute volumes of pentatopes for performing visibility checks upon application of each operator is listed in Alg. 1. The procedure computes nothing more than the determinant of the Jacobian of a linear 4-simplex, however, we emphasize how simple the predicate is to implement thanks to Lévy's versatile framework.

**Algorithm 1** Exact volume calculation of a pentatope (in the language of PCK [23]) for performing visibility checks. A filter is used to obtain a quicker solution when possible.

```
double orient4d( const double* p0 , const double* p1 , const double* p2 ,
                 const double* p3 , const double* p4 )
{
  // use the filter to determine if we can use the fast volume calculation
  int s = orient4d_filter(p0,p1,p2,p3,p4);
  if (s!=FPG_UNCERTAIN_VALUE)
    return orient4dfast(p0,p1,p2,p3,p4);

  const expansion& a11 = expansion_diff( p1[0] , p0[0] );
  const expansion& a12 = expansion_diff( p2[0] , p0[0] );
  const expansion& a13 = expansion_diff( p3[0] , p0[0] );
  const expansion& a14 = expansion_diff( p4[0] , p0[0] );

  const expansion& a21 = expansion_diff( p1[1] , p0[1] );
  const expansion& a22 = expansion_diff( p2[1] , p0[1] );
  const expansion& a23 = expansion_diff( p3[1] , p0[1] );
  const expansion& a24 = expansion_diff( p4[1] , p0[1] );

  const expansion& a31 = expansion_diff( p1[2] , p0[2] );
  const expansion& a32 = expansion_diff( p2[2] , p0[2] );
  const expansion& a33 = expansion_diff( p3[2] , p0[2] );
  const expansion& a34 = expansion_diff( p4[2] , p0[2] );

  const expansion& a41 = expansion_diff( p1[3] , p0[3] );
  const expansion& a42 = expansion_diff( p2[3] , p0[3] );
  const expansion& a43 = expansion_diff( p3[3] , p0[3] );
  const expansion& a44 = expansion_diff( p4[3] , p0[3] );

  // compute the determinant with exact arithmetic
  const expansion& Delta = expansion_det4x4( a11 , a12 , a13 , a14 ,
                                             a21 , a22 , a23 , a24 ,
                                             a31 , a32 , a33 , a34 ,
                                             a41 , a42 , a43 , a44 );
  if (Delta.sign()==ZERO) return 0.0;
  return Delta.value();
}
```

## C. Boundary representation

Adaptive numerical simulations require the set of facets discretizing the boundary representation of the domain. In the terminology of EGADS [24], the output tessellation of the boundary Faces, Edges and Nodes (of some BRep geometry, for example) must be homeomorphic to their associated geometric entities.

To retrieve the tessellations of the boundary entities, the vertices of the input mesh are strictly enforced to contain information as to which geometry entity they lie on. In particular, the lowest topological dimensional entity is needed and tracked during the application of a local operator. Given a $j$-dimensional simplicial facet $f \subset \mathcal{T}_h$ with vertices $f_i$ $(0 \le i \le j)$ lying on geometry entities $e_i$ $(0 \le i \le j)$, $f$ is a facet of the geometry entity

$$e_f = \min(\mathcal{E}), \quad \text{with } \mathcal{E} \equiv \mathcal{P}_0 \cap \mathcal{P}_1 \cap \cdots \cap \mathcal{P}_j, \tag{2}$$

where $\mathcal{P}_i$ denotes the set of all geometric entities higher in the topology hierarchy (the parents) than their associated $e_i$. The ordering $<$ on the geometry entities sorts the entities by topological number. This ensures $e_f$ is the lowest topological dimensional member of $\mathcal{E}$. If $\mathcal{E} = \emptyset$, then $f$ is an interior facet of the mesh. Furthermore, if $\mathcal{T}_h \cap e_f$ does not contain a ghost simplex, then $e_f$ is not on the geometry. This is an important check when meshing a domain with curved geometries.

**Tesseract geometry** The geometry of the $4d$ domain studied in this work is the unit tesseract. Geometric entities have been specialized for this simple geometry, but the hierarchy of these entities needs to be constructed. Within the `EGADS` framework [24], this is done by specializing the `ego` structure for the bounding geometric entities of the tesseract: Nodes, Edges, Squares and Cubes. The full hierarchy of the geometry is constructed by first labelling the eight bounding hyperplanes onto the 16 Nodes and traversing, dimension-by-dimension, the facets of the corresponding Cube until the Nodes are reached. Each facet is either constructed upon first visit, or referenced using a hash table, thus ensuring uniqueness of the 16 Nodes, 32 Edges, 24 Squares and 8 Cubes bounding the tesseract. The algorithm for building this hierarchy is inspired by our previous method for computing restricted Voronoi simplices [25]. In short, the Node-bisector information is first used to visit each bounding Cube and extract the bounding Squares using the $\mathcal{V}$Rep and $\mathcal{H}$Rep [25, 26]. Similarly, the bounding Edges of each Square are then extracted using the same Node-bisector information as well as the $\mathcal{V}$Rep and $\mathcal{H}$Rep. The final geometry topology is checked to ensure the required incidence relations are respected. The final tesseract Body references the eight bounding Cubes which are reproduced for the interested reader in Appendix IV.

## D. Validity checks

It is important to check both the visibility and the topological validity of each proposed local operator. If the re-insertion vertex $p$ with coordinates $\mathbf{x}$ is not visible to the boundary of the proposed cavity, then the operator is rejected. Furthermore, collapses are rejected if a vertex $v_0$ on geometry entity $g_0$ is collapsed onto a vertex $v_1$ on geometry entity $g_1$ for $g_0 < g_1$. That is, $g_0$ is lower in the geometry hierarchy than $g_1$. Edge splits are always accepted unless the proposed insertion is on a geometry entity. In this case, the coordinates $\mathbf{x}_s$ of the insertion vertex $v_s$ should be projected onto the common geometry entity of the edge $(v_0, v_1)$. Though cavity enlargement is disallowed for all other operators, enlargement is allowed for insertions on curved geometry entities provided the enlargement does not cause any vertex to be deleted in the process. Swaps are rejected if the re-inserted vertex $p$ on geometry entity $g_p$ is higher in the geometry hierarchy than the common entity of the edge being swapped. Smoothing is only rejected based on the visibility criterion.

## E. Operator scheduling

The local operator schedule is inspired by the work of Loseille [8, 9]. However, some very important changes were made to ensure the best possible metric conformity is obtained in the final mesh.

For reference, Loseille proposes to first perform collapses and splits to create a unit mesh, followed by swaps and smoothing to optimize the mesh. An important feature of Loseille's work is to ensure that no short edges are created during any edge split operation as this would require another pass of the collapse operator. Building upon this idea, we find it important to interleave swaps within the collapse and split operators to weave out of restrictive (geometry- or visibility-related) topological configurations.

The operator schedule, listed in Alg. 2, is composed of two main stages; each stage is divided into two sub-stages. The first stage consists of targeting edges longer than 2 in the metric space whereas the second stage targets edge lengths longer than $\sqrt{2}$. This was necessary to ensure the number of elements is not overshot in the adaptation process. The division of the stages into sub-stages is motivated by the fact that restrictive topological configurations might cause an operator to be rejected, therefore, it is important to allow a global pass of the swap operator to weave out of these configurations before attempting another sub-stage of collapses and splits. In addition, the swap operator has been directly interleaved within the split and collapse operators to immediately weave out of these restrictive configurations. Before a swap is accepted, the inserted topology is checked to ensure the current lengths do not get worse with the application of the swap. This ensures the edge length bounds can only improve during the mesh adaptation procedure.

The collapse portion of Stage 1a does not limit the maximum length created by the collapse operator, however, subsequent stages *do* limit this maximum length so as to ensure an improvement in the length bounds. Passes on the swap operator are divided into two target qualities (0.4 and 0.8) simply for performance reasons since there will be much fewer simplices with low quality as the mesh converges. All sub-stages are performed recursively, i.e. until no further operator in the sub-stage can be performed, except the pass on quality when `qt` $> 0.8$ since this may slow down performance when few swaps are being performed anyway.

**Algorithm 2** Local operator schedule.

| | sub-stage | comments |
|---|---|---|
| Stage 1a | `collapse(lmax_lim = false)` | do not limit max length produced by collapses |
| | `insert(lt = 2)` | edge splits with target length > 2 |
| | `swap_edges(qt = 0.4)` | swap edges touching any simplex with quality < 0.4 |
| | `swap_edges(qt = 0.8)` | swap edges touching any simplex with quality < 0.8 |
| | `smooth(max_iter = 10)` | perform 10 iterations of smoothing |
| Stage 1b | `collapse(lmax_lim = true)` | limit the max length produced by collapses |
| | `insert(lt = 2)` | edge splits with target length > 2 |
| | `swap_edges(qt = 0.4)` | swap edges touching any simplex with quality < 0.4 |
| | `swap_edges(qt = 0.8)` | swap edges touching any simplex with quality < 0.8 |
| | `smooth(max_iter = 10)` | perform 10 iterations of smoothing |
| Stage 2a | `collapse(lmax_lim = true)` | limit the max length produced by collapses |
| | `insert(lt = sqrt(2))` | edge splits with target length > sqrt(2) |
| | `swap_edges(qt = 0.4)` | swap edges touching any simplex with quality < 0.4 |
| | `swap_edges(qt = 0.8)` | swap edges touching any simplex with quality < 0.8 |
| | `smooth(max_iter = 10)` | perform 10 iterations of smoothing |
| Stage 2b | `collapse(lmax_lim = true)` | limit the max length produced by collapses |
| | `insert(lt = sqrt(2))` | edge splits with target length > sqrt(2) |
| | `swap_edges(qt = 0.4)` | swap edges touching any simplex with quality < 0.4 |
| | `swap_edges(qt = 0.8)` | swap edges touching any simplex with quality < 0.8 |

### F. Metric limiting

We assume the mesh adaptation algorithm will supply a metric field requesting *small* changes with respect to the input mesh [2]. As such, the analytic metric fields of the following sections are first limited with respect to the implied metric of the input mesh. This is achieved in two main stages. First, a continuous vertex-based implied metric of the mesh, $\mathcal{M}_v$, is calculated from the discontinuous simplex-based one, $\mathcal{M}_\kappa$, by solving an optimization problem of the vertex metrics:

$$\mathcal{M}_v^* = \arg\min_{\mathcal{M}_v} \left(C(\mathcal{M}_v) - C(\mathcal{T}_h)\right)^2 + \left(\sum_e \delta(e)^d\right)^2, \text{ where } \delta(e) = \begin{cases} \ell_{\mathcal{M}_v}(e) - \sqrt{2}, & \ell_{\mathcal{M}_v}(e) > \sqrt{2}, \\ \sqrt{2}/2 - \ell_{\mathcal{M}_v}(e), & \ell_{\mathcal{M}_v}(e) < \sqrt{2}/2, \\ 0, & \text{else,} \end{cases}$$

where $d$ is the topological dimension of the mesh. This objective function drives the complexity of the implied metric $C(\mathcal{M}_v)$ to the actual complexity $C(\mathcal{T}_h)$ [17] with a penalty on the lengths of the edges ($e$) since all edge lengths under the implied metric should be close to 1. Next, the analytic target metric field is evaluated at each vertex of the mesh, yielding a discrete vertex-based target metric $\mathcal{M}_{t,v}$. Each $\mathbf{m}_{t,v} \in \mathcal{M}_{t,v}$ is limited as described in Alg. 3.

**Algorithm 3** Metric-limiting procedure

```
1) compute the *step* from the implied metric to the target: $\mathbf{s}_v = (\mathbf{m}_v)^{-1/2}\mathbf{m}_{t,v}(\mathbf{m}_v)^{-1/2}$ ($\mathbf{m}_v \in \mathcal{M}_v$),
2) directly limit the entries of the step matrix $\mathbf{s}_v$ such that $|s_{v,i,j}| \leq 2\log\alpha$, $\forall i, j \in 1 \ldots d$,
3) compute the limited target metric $\widetilde{\mathbf{m}}_{t,v} = (\mathbf{m}_v)^{1/2}\exp(\mathbf{s}_v)(\mathbf{m}_v)^{1/2}$
```

Since the developed mesh generator targets metric fields provided by the work of Yano [2], in which the metric is limited by a factor of 2 due to the approximability of the local error model, we set $\alpha = 2$. Upon adaptation of the mesh, the target metric is re-evaluated at the vertices of the new mesh and the above procedure is repeated until the number of limited vertex metrics is small. Usually, this occurs within 10-20 iterations of the above algorithm, whereby less than 1% of the step matrices have been limited.

## III. Numerical results

Here we examine the ability of the dimension-independent metric-conforming mesh generator to produce both $3d$ and $4d$ meshes composed of tetrahedra and pentatopes, respectively. Metric-conformity will be assessed by examining the edge lengths and cell qualities of the produced meshes. We differentiate between the two descriptions of the metric field: analytic and discrete. The *analytic* metric field is used to analyze the properties of the mesh as measured under the analytic description of the metric field. The *discrete* metric field is a continuous representation of the analytic metric field on the background mesh obtained from the previous iteration of Alg.3. The former is used to demonstrate that the sequence of meshes ultimately converges to a mesh conforming to the desired metric field. The latter is used to assess the iteration-to-iteration metric-conformity properties of the mesh generator which is important for adaptive numerical simulations. Both the analytic and discrete metric-conformity statistics are tabulated. However, we simply provide the length and quality histograms as measured under the *analytic* metric field since the trends in the histograms when evaluating the quantities with the discrete metric are very similar to their analytic counterpart. We additionally discuss the number of tetrahedra and pentatopes produced for each case which is important when imposing the degree-of-freedom constraint used by the adaptive algorithm of Yano [2].

### A. Verification on UGAWG benchmark cases

First let us demonstrate the ability of the developed mesh generator to conform to the benchmark cases of the Unstructured Grid Adaptation Working Group (UGAWG). The geometries and metric fields are described in the group's first benchmark paper [19]. The meshes generated for each of the four cases are shown in Fig. 2. The meshes appear similar to those produced by the other codes in the UGAWG which is a good verification of our software implementation. However, we remark upon one very important difference. Though the UGAWG participants report the number of tetrahedra produced by each software implementation, no remark is made as to how many tetrahedra are expected for each case. Integrating the mapped volume under metric field and dividing by the volume of the ideal unit tetrahedron yields the expected number of tetrahedra. Table 2 tabulates the final number of tetrahedra our method produces in each case and compares this with the expected number of tetrahedra alongside the number of tetrahedra of the participating UGAWG codes which is closest to the expected number. We note that our software achieves the expected number of tetrahedra much closer than the other codes which can be due to either (1) the proposed operator schedule or (2) the use of metric limiting from the current mesh. The only case in which the target number of tetrahedra was overshot (though still less than the other UGAWG codes) was in the Cube-Cylinder Polar 1 case which was deemed a difficult case due to the high gradation requested by the metric [19]. As such, we decided to target a smaller length in the tangential direction to eliminate issues that might be caused by the competing metric field and curved geometry. Indeed, when setting this tangential spacing to $h_t = 0.01$ instead of its original $h_t = 0.1$, we agree much better with the expected number of tetrahedra. The ability to match the target complexity is a promising result because of the computational cost constraint used by the adaptive algorithm of Yano [2].

The edge length and quality histograms, as recomputed under the analytic metric, for these $3d$ benchmark cases are provided in Fig. 3 and metric-conformity statistics measured under both the analytic and discrete representations of the metric field are given in Tables 3 and 4, respectively. All meshes exhibit very good metric conformity; roughly 99% of the edges are within the quasi-unit bounds and a large number of tetrahedra have a quality greater than 0.8. The only case for which metric-conformity was low was unsurprisingly the Cube Cylinder Polar 1 due to the high gradation requested by the metric field near the curved surface. When this gradation is relaxed ($h_t = 0.01$), metric conformity improves significantly.

**Table 2 The number of tetrahedra produced by our method for each $3d$ UGAWG case is much closer to the expected number than other mesh generators participating in the UGAWG [19].**

| Case | # tetrahedra | expected | UGAWG # tetrahedra (code) |
|---|---|---|---|
| Cube Linear | 40,212 | 39.4k | 45,158 (feflo.a) |
| Cube-Cylinder Linear | 32,434 | 31.7k | 38,302 (EPIC-ICSM) |
| Cube-Cylinder Polar 1 ($h_t = 0.1$) | 25,388 | 20.2k | 30,378 (EPIC-ICSM) |
| Cube-Cylinder Polar 1 ($h_t = 0.01$) | 203,550 | 202k | n/a |
| Cube-Cylinder Polar 2 | 36,445 | 36.4k | 44,280 (EPIC-ICSM) |

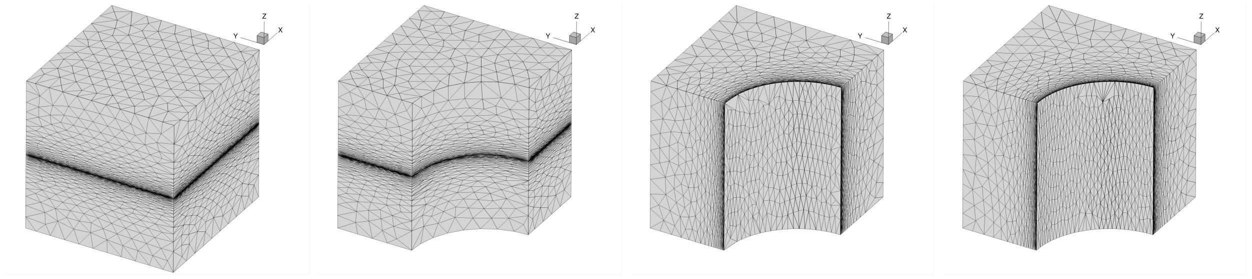


**Fig. 2 Meshes generated from the UGAWG benchmark cases appear similar to those produced by existing mesh generators [19].**

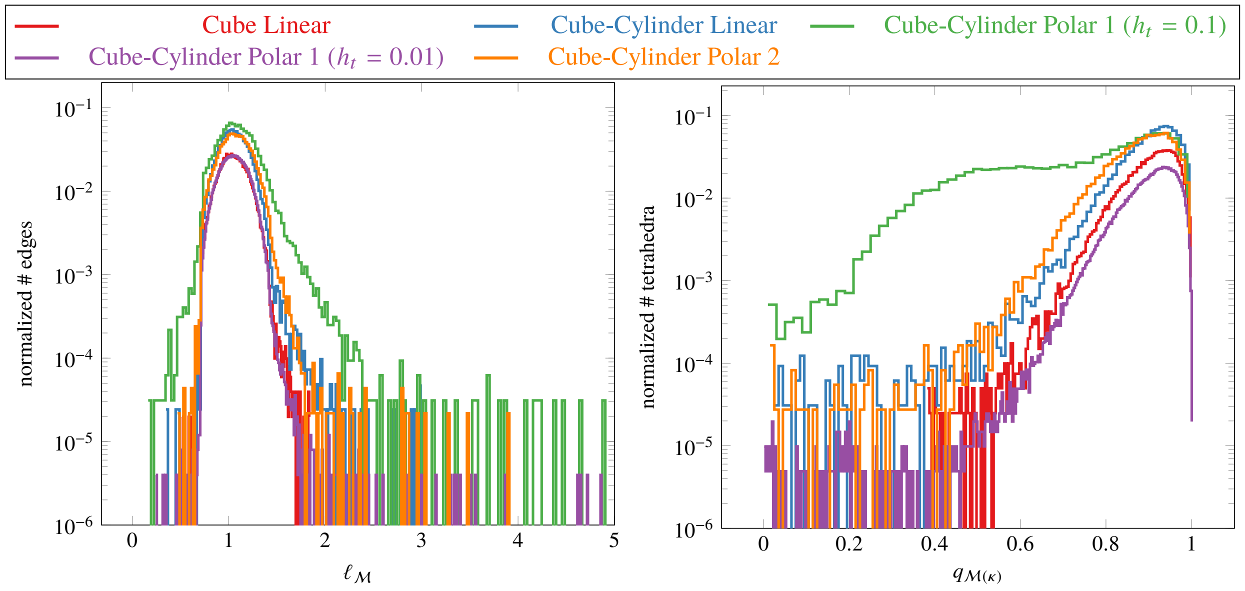


**Fig. 3 Analytic metric edge length and cell quality for the $3d$ UGAWG benchmark cases demonstrate that our method conforms very well to the metric fields used in the UGAWG benchmark cases. The highly-graded Cube-Cylinder Polar 1 metric with $h_t = 0.1$ exhibits the poorest metric conformity (as with the other mesh generators) but is improved with $h_t = 0.01$.**

**Table 3 Analytic mesh statistics for the $3d$ UGAWG benchmark cases. All quantities are measured under the analytic description of the metric field. The percentage of quasi-unit edge lengths (% $\ell_{\text{unit}}$) refers to the percentage of edges with lengths within $[\sqrt{2}/2, \sqrt{2}]$. The percentage of quasi-unit tetrahedra (% $q_{\text{unit}}$) refers to the percentage of tetrahedra with quality greater than $0.8$.**

| Case | $\ell_{\min}$ | $\ell_{\max}$ | % $\ell_{\text{unit}}$ | $\mathbf{q}_{\min}$ | $\mathbf{q}_{\text{avg}}$ | % $\mathbf{q}_{\text{unit}}$ |
|---|---|---|---|---|---|---|
| Cube Linear | 0.513 | 1.832 | 99.5 | 0.3820 | 0.8983 | 92.6 |
| Cube-Cylinder Linear | 0.342 | 2.996 | 99.2 | 0.0282 | 0.8937 | 91.3 |
| Cube-Cylinder Polar 1 ($h_t = 0.1$) | 0.177 | 16.419 | 93.8 | 0.0011 | 0.7220 | 45.7 |
| Cube-Cylinder Polar 1 ($h_t = 0.01$) | 0.242 | 9.167 | 99.6 | 0.0019 | 0.9010 | 93.3 |
| Cube-Cylinder Polar 2 | 0.465 | 6.009 | 98.5 | 0.0156 | 0.8719 | 83.8 |

**Table 4 Discrete mesh statistics for the $3d$ UGAWG benchmark cases. All quantities are measured under the discrete description of the metric field at the 20th iteration of Alg. 3. The percentage of quasi-unit edge lengths (% $\ell_{\text{unit}}$) refers to the percentage of edges with lengths within $[\sqrt{2}/2, \sqrt{2}]$. The percentage of quasi-unit tetrahedra (% $q_{\text{unit}}$) refers to the percentage of tetrahedra with quality greater than $0.8$.**

| **Case** | $\ell_{\text{min}}$ | $\ell_{\text{max}}$ | % $\ell_{\text{unit}}$ | $\mathbf{q}_{\text{min}}$ | $\mathbf{q}_{\text{avg}}$ | % $\mathbf{q}_{\text{unit}}$ |
|---|---|---|---|---|---|---|
| Cube Linear | 0.579 | 1.902 | 99.5 | 0.3557 | 0.8984 | 92.7 |
| Cube-Cylinder Linear | 0.363 | 2.837 | 99.2 | 0.0312 | 0.8939 | 91.3 |
| Cube-Cylinder Polar 1 ($h_t = 0.1$) | 0.012 | 10.439 | 94.6 | 2.9e-06 | 0.7323 | 46.7 |
| Cube-Cylinder Polar 1 ($h_t = 0.01$) | 0.349 | 3.383 | 99.6 | 0.0070 | 0.9011 | 93.4 |
| Cube-Cylinder Polar 2 | 0.595 | 4.107 | 98.5 | 0.0156 | 0.8723 | 84.3 |

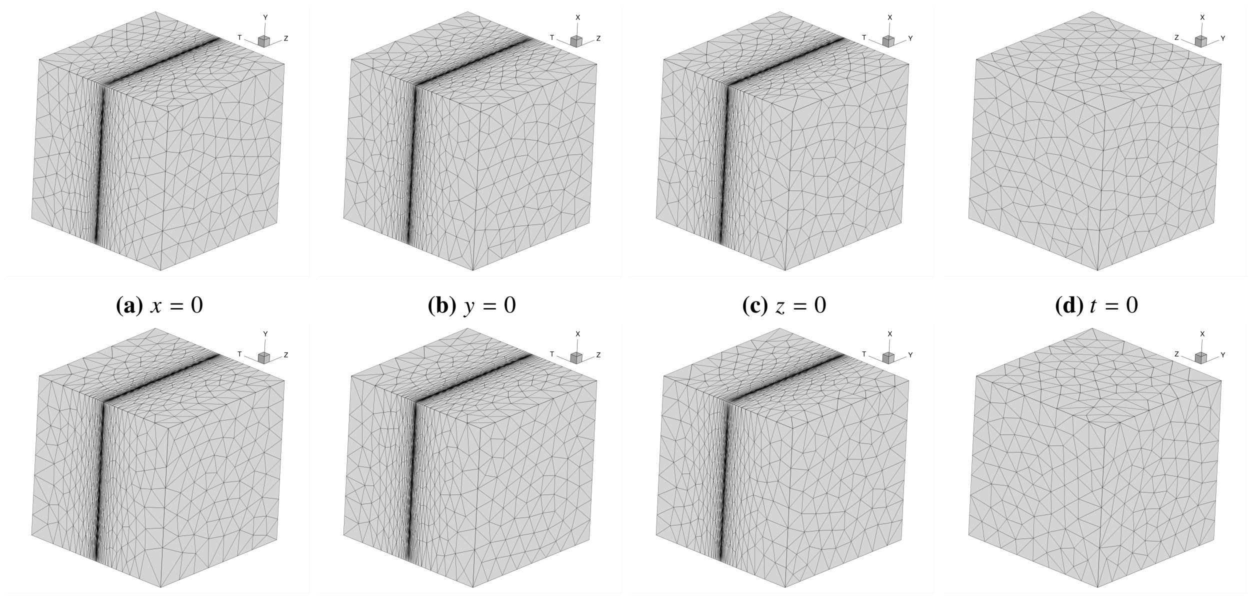
## B. Anisotropic metrics in $4d$

Having verified that the mesh generator works well in $3d$, let us now study two $4d$ problems on the unit tesseract $\mathbf{x} = [x, y, z, t]^T \in [0, 1]^4$.

**Tesseract Linear** The first problem, inspired by the UGAWG Linear metric, is represented analytically by

$$\mathbf{m}(\mathbf{x}) = \text{diag}\left(h_x^{-2}, h_y^{-2}, h_z^{-2}, h_t^{-2}\right), \tag{3}$$

where $h_x = h_y = h_z = h_{\text{max}} = 0.125$ and $h_t = h_0 + 2(h_{\text{max}} - h_0)|t - 0.5|$. Meshes were generated for the two cases where $h_t = [0.25, 0.125]$. 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 case.



**(a)** $x = 0$ **(b)** $y = 0$ **(c)** $z = 0$ **(d)** $t = 0$

**(e)** $x = 1$ **(f)** $y = 1$ **(g)** $z = 1$ **(h)** $t = 1$

**Fig. 4 Meshes of the eight bounding cubes generated from the Tesseract Linear case with $h_t = 0.125$ demonstrate the expected behaviour: the six bounding Cubes corresponding to the hyperplanes with non-constant $t$ show a refinement in the $t$ direction. The coordinate axes have been labelled according the associated hyperplane.**

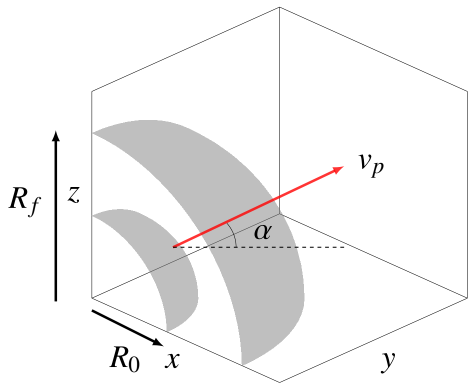
**Tesseract Wave** The second metric field is modeled after a propagating spherical wave in $3d$ (see Fig. 5a). Consider a spherical wave of radius $R_0 = 0.4$ centered about the origin at time $t = 0$. If the wave expands at a constant velocity to a radius $R_f = 0.8$ at time $t = 1$, then the expanding sphere traces the geometry of a hypercone in $4d$. To visualize this, consider an expanding circle moving in a direction orthogonal to the circle – the expanding circle traces the geometry of a cone (Fig. 5b). The metric used to capture the propagation of this wave is

$$\mathbf{m}(\mathbf{x}) = \mathbf{Q}\,\mathrm{diag}\left(h_r^{-2}, h_\theta^{-2}, h_\phi^{-2}, h_t^{-2}\right)\,\mathbf{Q}^T, \quad \mathbf{Q} = \begin{bmatrix} \sin\alpha\cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi & \cos\alpha\cos\phi\sin\theta \\ \sin\alpha\sin\phi\sin\theta & \cos\theta\sin\phi & \cos\phi & \cos\alpha\sin\phi\sin\theta \\ \sin\alpha\cos\theta & -\sin\theta & 0 & \cos\alpha\cos\theta \\ -\cos\alpha & 0 & 0 & \sin\alpha \end{bmatrix}. \quad (4)$$

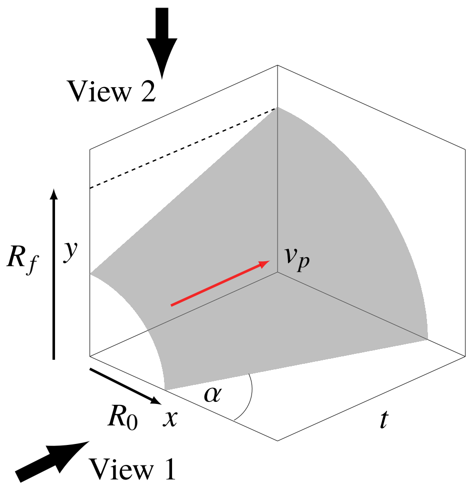
The spacings in the tangential directions are

$$h_\theta, h_\phi = \begin{cases} h_1, & |r - r_0| > \delta, \\ (h_1 - h_2)|r - r_0|/\delta + h_2, & |r - r_0| \le \delta, \end{cases} \quad (5)$$

with $r_0 = R_0 + (R_f - R_0)t$ is the position of the spherical wave with time. The spacing in the radial direction is $h_r = h_0 + 2(h_1 - h_0)|r - r_0|$ 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 $\phi = \arctan(y, x)$. Also observe that the eigenvectors $\mathbf{Q}$ are simply the unit vectors for spherical coordinates with the radial unit vector rotated by an angle $\alpha = \arctan(t_f - t_0, R_f - R_0)$. The remaining parameters are $h_0 = 0.0025$, $h_1 = 0.125$, $h_2 = 0.05$ and $\delta = 0.1$.



**(a) Propagating spherical wave problem description.**

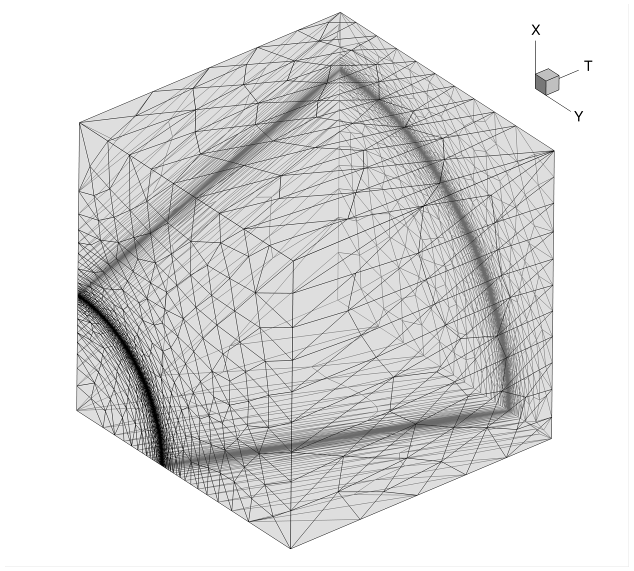


**(b) Analogous problem in $2d + t$.**

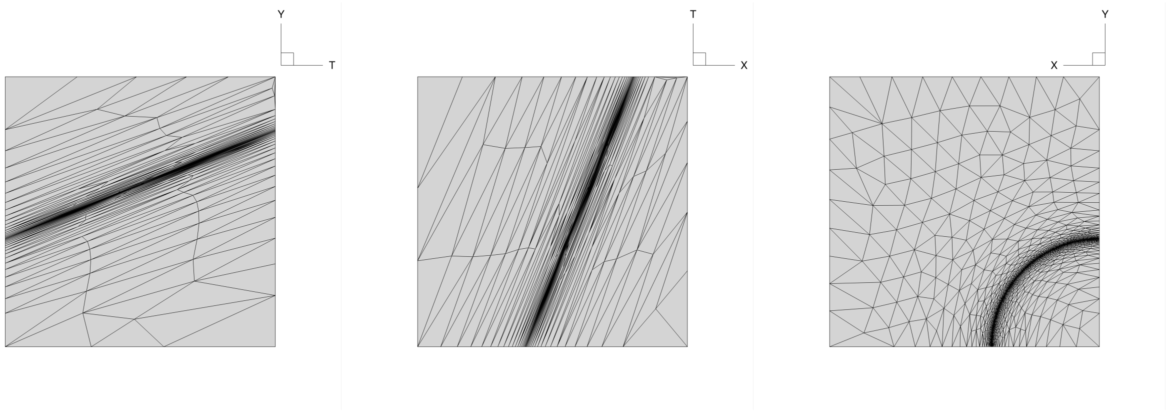
**Fig. 5 Problem description for the Tesseract Wave case and analogous problem description in $2d + t$. View 1 corresponds to visualizing constant $t$ hyperplanes whereas View 2 corresponds to visualizing constant $x$ or constant $y$ (or constant $z$ in $3d + t$) hyperplanes. In $2d + t$, we expect to see lines with slope $\tan\alpha$, however, these should appear as cones in $3d + t$ due to symmetry about the time axis.**

Since we are essentially conforming to the geometry of a hypercone in $4d$, we can infer what the meshes along the eight bounding hyperplanes of the final tesseract mesh should look like. At $t = 0$ and $t = 1$, we can clearly expect to see a mesh of the spherical wave at its initial and final radii, respectively. This corresponds to $3d$ projections of the hypercone at constant $t$ (equivalent to View 1 in Fig. 5b). Along the constant $x$, $y$, or $z$ hyperplanes, we expect to see $3d$ cones since these correspond to the projection of the hypercone along these spatial dimensions (non-constant $t$) – this is analogous to View 2 in Fig. 5b. This case will be referred to as the Tesseract Wave case.

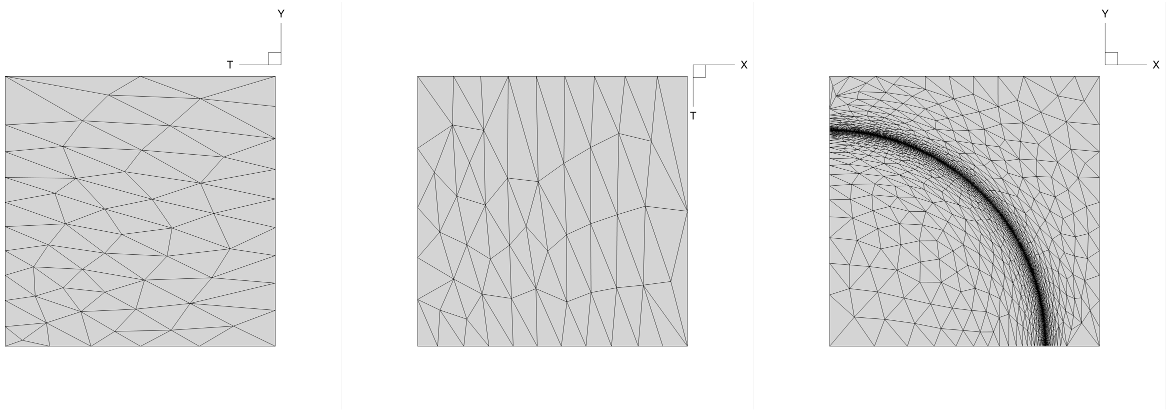
To better visualize the meshes of the eight bounding hyperplanes, we have provided the meshes of the six bounding planes of the analogous $2d + t$ case (described in Fig. 5b) in Fig. 6). The projection of the $3d$ cone onto the $x = 0$ and $y = 0$ planes is indeed a straight line with slope $\tan\alpha$ but we see no refinement at the $x = 1$ or $y = 1$ planes. We also see the initial and final radii of the expanding circle at $t = 0$ and $t = 1$ planes.



**(a) Boundary of full spacetime mesh**



**(b)** $x = 0$ **(c)** $y = 0$ **(d)** $t = 0$



**(e)** $x = 1$ **(f)** $y = 1$ **(g)** $t = 1$

**Fig. 6 Meshes of the six bounding Faces generated from the $3d$ case (analogous to the Tesseract Wave case) along with full spacetime mesh. The coordinate axes have been labelled according the associated hyperplane.**

**Discussion** Indeed, Figs. 4 and 8 show the expected behaviour in the meshes of the bounding hypercubes. The mesh generator succeeds in refining in the $t$ direction for the expected six hyperplanes of non-constant $t$ in the Tesseract Linear case. For the Tesseract Wave case, we also observe the expected mesh distributions. At time $t = 0$, we see a mesh of the initial sphere (Fig. 8d) and at $t = 1$ we see the final sphere (Fig. 8h). Additionally, we do see the expected cone shape in Figs. 8a,8b and 8c. Unfortunately, we have no current way of visualizing the actual pentatopes in the full tesseract mesh but will investigate this in a future work.

The analytic edge length and quality histograms for the Tesseract Linear (with $h_t = 0.25$ and $0.125$) and Tesseract Wave cases are shown in Figs. 7. Both analytic and discrete metric conformity statistics are reported in Table 6 and 7 with the pentatope counts given in Table 7. Despite metric conformity being very good (about 99% in edge lengths and at least 80% in simplex quality) for the $3d$ UGAWG benchmark cases, the $4d$ metrics seem slightly more difficult to obtain the same level of metric conformity conformity (about 93 – 98% in edge lengths) – observe the wider distribution in the edge lengths and lower quality elements of Fig. 7. Furthermore, the generated element counts is slightly higher than the expected number which is attributable to the poorer metric conformity in these cases – see Table 5. In particular, the Tesseract Wave case exhibits the poorest quality elements which, in turn, influences the ability of the edge lengths to conform to the metric field. Perhaps more swaps are needed to weave out of restrictive topological configurations for this difficult metric.

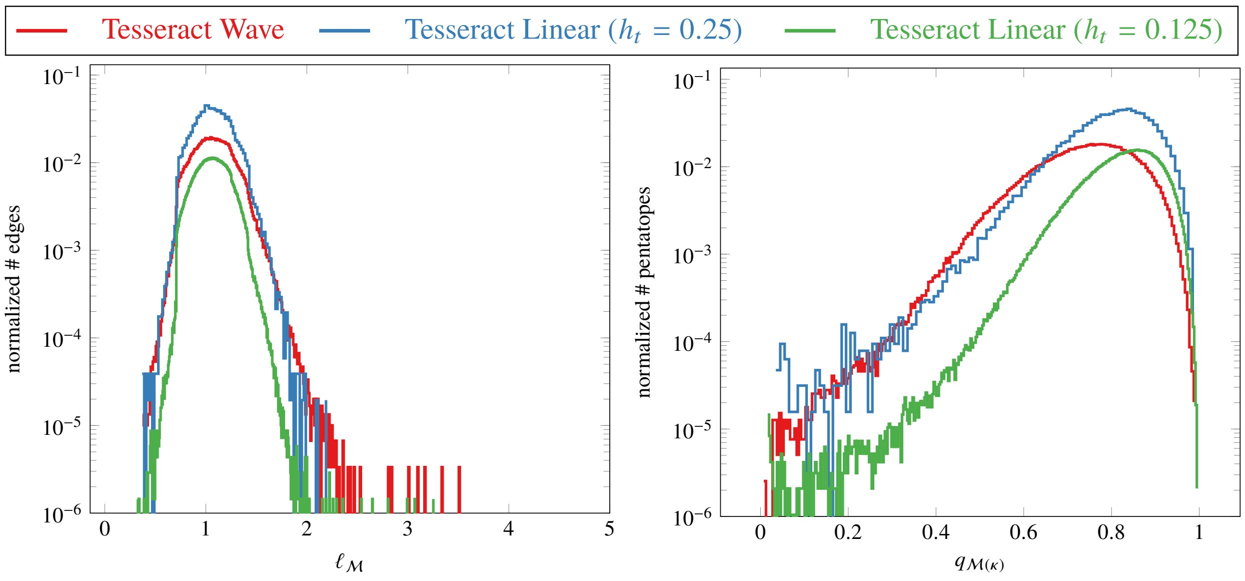


**Fig. 7 Analytic metric edge length and quality for the $4d$ anisotropic cases demonstrate that our method conforms fairly well to the metric fields in our $4d$ cases. The best metric conformity is obtained with the more resolved Tesseract Linear case ($h_t = 0.125$) whereas the Tesseract Wave case shows the poorest metric conformity, perhaps due to the difficulty in conforming to this metric field.**

**Table 5 The number of pentatopes produced for each $4d$ case demonstrates that our method overshoots the expected number of pentatopes but is still reasonable.**

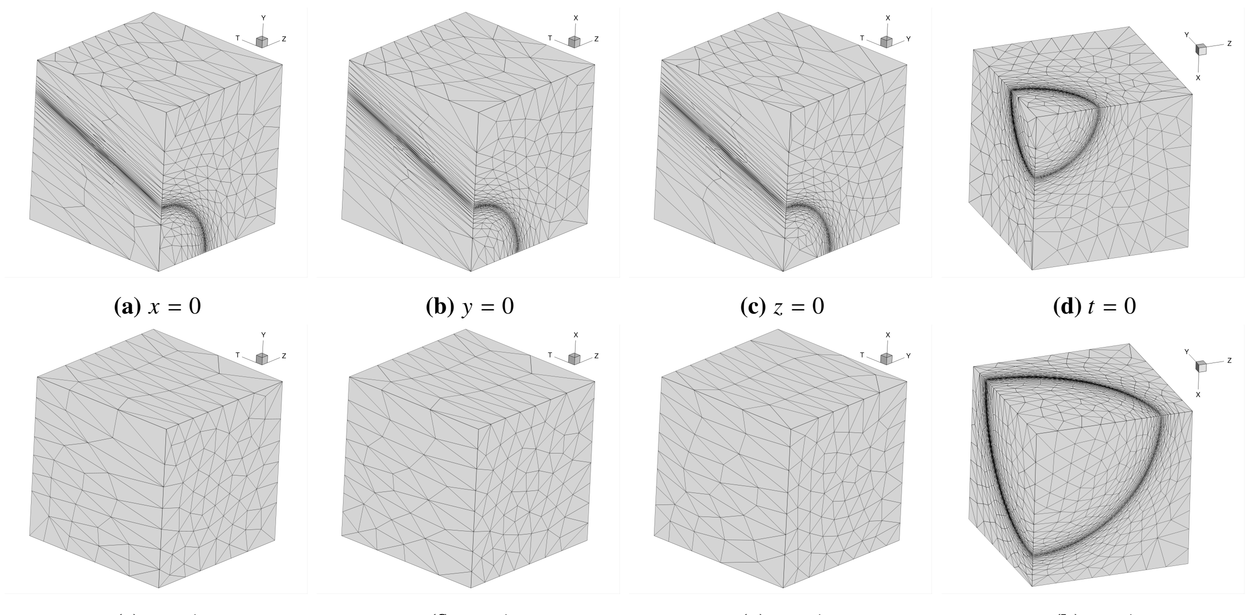
| Case | # pentatopes | expected |
|---|---|---|
| Tesseract Linear ($h_t = 0.25$) | 63,686 | 51k |
| Tesseract Linear ($h_t = 0.125$) | 957,453 | 818k |
| Tesseract Wave | 393,907 | n/a |

**Table 6 Analytic mesh statistics for the $4d$ tesseract cases. All quantities are measured under the analytic description of the metric field. The percentage of quasi-unit edge lengths (% $\ell_{\text{unit}}$) refers to the percentage of edges with lengths within $[\sqrt{2}/2, \sqrt{2}]$. The percentage of quasi-unit tetrahedra (% $q_{\text{unit}}$) refers to the percentage of tetrahedra with quality greater than $0.8$.**

| Case | $\ell_{\min}$ | $\ell_{\max}$ | % $\ell_{\text{unit}}$ | $\mathbf{q}_{\min}$ | $\mathbf{q}_{\text{avg}}$ | % $\mathbf{q}_{\text{unit}}$ |
|---|---|---|---|---|---|---|
| Tesseract Linear ($h_t = 0.25$) | 0.371 | 2.191 | 96.1 | 0.0360 | 0.7856 | 50.0 |
| Tesseract Linear ($h_t = 0.125$) | 0.323 | 3.252 | 98.0 | 0.0162 | 0.8187 | 64.1 |
| Tesseract Wave | 0.371 | 5.035 | 93.8 | 0.0068 | 0.7298 | 29.5 |

**Table 7 Discrete mesh statistics for the $4d$ tesseract cases. All quantities are measured under the discrete description of the metric field at the 20th iteration of Alg. 3. The percentage of quasi-unit edge lengths (% $\ell_{\text{unit}}$) refers to the percentage of edges with lengths within $[\sqrt{2}/2, \sqrt{2}]$. The percentage of quasi-unit tetrahedra (% $q_{\text{unit}}$) refers to the percentage of tetrahedra with quality greater than $0.8$.**

| Case | $\ell_{\min}$ | $\ell_{\max}$ | % $\ell_{\text{unit}}$ | $\mathbf{q}_{\min}$ | $\mathbf{q}_{\text{avg}}$ | % $\mathbf{q}_{\text{unit}}$ |
|---|---|---|---|---|---|---|
| Tesseract Linear ($h_t = 0.25$) | 0.299 | 1.953 | 96.1 | 0.0360 | 0.7832 | 48.8 |
| Tesseract Linear ($h_t = 0.125$) | 0.359 | 1.977 | 98.1 | 0.0163 | 0.8188 | 64.2 |
| Tesseract Wave | 0.286 | 2.857 | 94.3 | 0.0239 | 0.7330 | 30.2 |



**(a)** $x = 0$ **(b)** $y = 0$ **(c)** $z = 0$ **(d)** $t = 0$

**(e)** $x = 1$ **(f)** $y = 1$ **(g)** $z = 1$ **(h)** $t = 1$

**Fig. 8 Meshes of the eight bounding cubes generated from the $3d + t$ Tesseract Wave case demonstrate the expected behaviour: the meshes at $t = 0$ and $t = 1$ show the initial and final spheres whereas the meshes of the bounding Cubes with non-constant $t$ show a cone – the projection of a $4d$ hypercone onto $3d$. The coordinate axes have been labelled according to the associated hyperplane.**

## IV. Conclusions & future work

We have investigated and successfully implemented an extension of local cavity operators to generate metric-conforming meshes consisting of pentatopes in $4d$. Important choices in the design of the software consisted of (1) closing the mesh with a ghost vertex such that it is always without boundary, (2) enforcing that a description of the vertex-to-geometry associations be provided by the input mesh, (3) the development of a predicate for *exactly* computing the volume of a pentatope, (4) a simple and efficient operator schedule to achieve good metric conformity and (5) a metric limiting procedure to emulate the behaviour of a metric request from an adaptive numerical simulation. The software was first demonstrated on benchmark $3d$ cases; the results show excellent metric conformity and a very good ability to match the expected number of simplices. Furthermore, we demonstrated our method on two anisotropic metrics to generate $4d$ meshes. Metric conformity was good but can likely be improved with additional swapping. A visualization of the Cubes at the eight hyperplanes bounding the tesseract were given and future work consists of investigating visualization techniques of the full $4d$ mesh. We will also use the developed software to perform fully unstructured adaptive numerical simulations of unsteady partial differential equations in $3d + t$. The software, henceforth referred to as `avro`, will be made available with an open-source license in the Spring of 2019.

## Acknowledgments

We wish to thank Dr. Adrien Loseille for his help in clarifying questions about local cavity operators. 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.

## References

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

[2] Yano, M., "An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Jun. 2012.

[3] Hartmann, R., "Adaptive FE methods for conservation equations," *Hyperbolic Problems: Theory, Numerics, Applications: Eighth International Conference in Magdeburg, February, March 2000*, International series of numerical mathematics, Vol. 141, edited by H. Freistühler and G. Warnecke, Birkhäuser, Basel, 2001, pp. 495–503.

[4] Bangerth, W., Geiger, M., and Rannacher, R., "Adaptive Galerkin finite element methods for the wave equation," *Comput. Methods Appl. Math.*, Vol. 10, No. 1, 2010, pp. 3–48.

[5] Yano, M., and Darmofal, D. L., "An optimization-based framework for anisotropic simplex mesh adaptation," *J. Comput. Phys.*, Vol. 231, No. 22, 2012, pp. 7626–7649.

[6] Jayasinghe, S., Darmofal, D. L., Burgess, N. K., Galbraith, M. C., and Allmaras, S. R., "A space-time adaptive method for reservoir flows: formulation and one-dimensional application," *Computational Geosciences*, Vol. 22, No. 1, 2018, pp. 107–123.

[7] Jayasinghe, S., "An adaptive space-time discontinuous Galerkin method for reservoir flows," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2018.

[8] Loseille, A., Alauzet, F., and Menier, V., "Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes," *Computer-Aided Design*, Vol. 85, 2017, pp. 53 – 67. doi:https://doi.org/10.1016/j.cad.2016.09.008, URL `http://www.sciencedirect.com/science/article/pii/S0010448516301142`, 24th International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[9] Loseille, A., "Metric-orthogonal Anisotropic Mesh Generation," *Procedia Engineering*, Vol. 82, 2014, pp. 403 – 415. doi:http://dx.doi.org/10.1016/j.proeng.2014.10.400, URL `http://www.sciencedirect.com/science/article/pii/S1877705814016798`.

[10] Loseille, A., and Löhner, R., "Cavity-Based Operators for Mesh Adaptation," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.*, edited by A. I. of Aeronautics and Astronautics, American Institute of Aeronautics and Astronautics, 2013. doi:10.2514/6.2013-152, URL `https://hal.inria.fr/hal-00935363`.

[11] Coupez, T., Digonnet, H., and Ducloux, R., "Parallel meshing and remeshing," *Applied Mathematical Modelling*, Vol. 25, 2000, pp. 153–175.

[12] Coupez, T., "Génération de maillage et adaptation de maillage par optimisation locale," *Revue européenne des éléments finis*, Vol. 9, 2000, pp. 403–423.

[13] Gruau, C., and Coupez, T., "3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 4849, 2005, pp. 4951 – 4976. doi:http://dx.doi.org/10.1016/j.cma.2004.11.020, URL `//www.sciencedirect.com/science/article/pii/S0045782505000745`, unstructured Mesh Generation.

[14] Gruau, C., "Metric generation for anisotropic mesh adaptation, with numerical applications to material forming simulation," Ph.D. thesis, École Nationale Supérieure des Mines de Paris, 2005.

[15] Digonnet, H., Coupez, T., Laure, P., and Silva, L., "Massively parallel anisotropic mesh adaptation," *The International Journal of High Performance Computing Applications*, 2017. doi:10.1177/1094342017693906.

[16] Bossen, F. J., and Heckbert, P. S., "A Pliant Method for Anisotropic Mesh Generation," *5th Intl. Meshing Roundtable*, 1996, pp. 63–74.

[17] Loseille, A., and Alauzet, F., "Continuous mesh framework part I: Well-posed continuous interpolation error," *SIAM J. Numer. Anal.*, Vol. 49, No. 1, 2011, pp. 38–60.

[18] Loseille, A., and Alauzet, F., "Continuous mesh framework part II: Validations and applications," *SIAM J. Numer. Anal.*, Vol. 49, No. 1, 2011, pp. 61–86.

[19] Ibanez, D., Barral, N., Krakos, J., Loseille, A., Michal, T., and Park, M., "First benchmark of the Unstructured Grid Adaptation Working Group," *Procedia Engineering*, Vol. 203, 2017, pp. 154 – 166. doi:https://doi.org/10.1016/j.proeng.2017.09.800, URL `http://www.sciencedirect.com/science/article/pii/S1877705817343618`, 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

[20] Shewchuk, J. R., "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator," *Applied Computational Geometry: Towards Geometric Engineering*, edited by M. C. Lin and D. Manocha, Springer-Verlag, 1996, pp. 203–222.

[21] Si, H., "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator," Weierstrass Institute for Applied Analysis and Stochastics, 2005. `http://tetgen.berlios.de`.

[22] Shewchuk, J. R., "Adaptive precision floating-point arithmetic and fast robust geometric predicates," Tech. rep., Carnegie Mellon University, 1996.

[23] Lévy, B., "Robustness and efficiency of geometric programs: The Predicate Construction Kit," *Computer-Aided Design*, Vol. 72, 2016, pp. 3–12.

[24] Haimes, R., and Dannenhoffer, J., *The Engineering Sketch Pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry*, 2013.

[25] Caplan, P. C., Haimes, R., Darmofal, D. L., and Galbraith, M. C., "Anisotropic geometry-conforming *d*-simplicial meshing via isometric embeddings," *Proceedings of the 26th International Meshing Roundtable*, Springer Berlin Heidelberg, 2017.

[26] Henk, M., Richter-Gebert, J., and Ziegler, G. M., "Basic Properties of Convex Polytopes," *Handbook of Discrete and Computational Geometry, 2nd Ed.*, 2004.

## Tesseract geometry

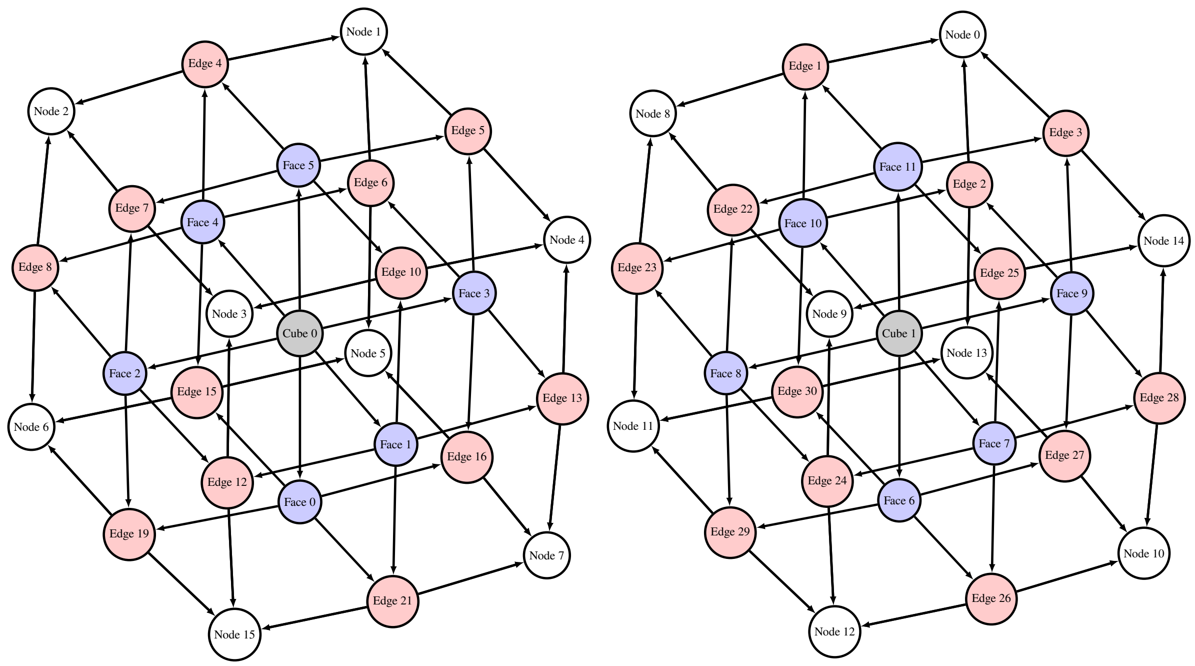
In case the interested reader wishes to reproduce the tesseract geometry hierarchy (this is a fun exercise), the coordinates and hyperplane incidence relations are provided in Table 8. From this data, the entire geometry hierarchy can be constructed by traversing the hyperplanes and extracting the $j$-dimensional polytope along each hyperplane by starting with the Cubes ($j = 3$) and ending with the Edges ($j = 1$). Specifically, this can be done by extracting the $\mathcal{H}$Rep of the current polytope, from which the $\mathcal{V}$Rep of the children polytopes can be computed [26]. For reference, this hierarchy for each bounding cube is reproduced as graphs in Figs. 9,10,11 and 12.

**Table 8 Node coordinates for the tesseract geometry with the set of bisectors $\mathcal{B}$ (corresponding to the set of bounding Cubes – or hyperplanes) touching each Node.**

| Node | **x** | $\mathcal{B}$ |
|---|---|---|
| 1 | (1,1,1,1) | { 1,2,3,4 } |
| 2 | (-1,1,1,1) | { -1,2,3,4 } |
| 3 | (-1,-1,1,1) | { -1,-2,3,4 } |
| 4 | (-1,-1,-1,1) | { -1,-2,-3,4 } |
| 5 | (-1,1,-1,1) | { -1,2,-3,4 } |
| 6 | (-1,1,1,-1) | { -1,2,3,-4 } |
| 7 | (-1,-1,1,-1) | { -1,-2,3,-4 } |
| 8 | (-1,1,-1,-1) | { -1,2,-3,-4 } |
| 9 | (1,-1,-1,1) | { 1,-2,3,4 } |
| 10 | (1,-1,-1,1) | { 1,-2,-3,4 } |
| 11 | (1,1,-1,-1) | { 1,2,-3,-4 } |
| 12 | (1,-1,1,-1) | { 1,-2,3,-4 } |
| 13 | (1,-1,-1,-1) | { 1,-2,-3,-4 } |
| 14 | (1,1,1,-1) | { 1,2,3,-4 } |
| 15 | (1,1,-1,1) | { 1,2,-3,4 } |
| 16 | (-1,-1,-1,-1) | { -1,-2,-3,-4 } |

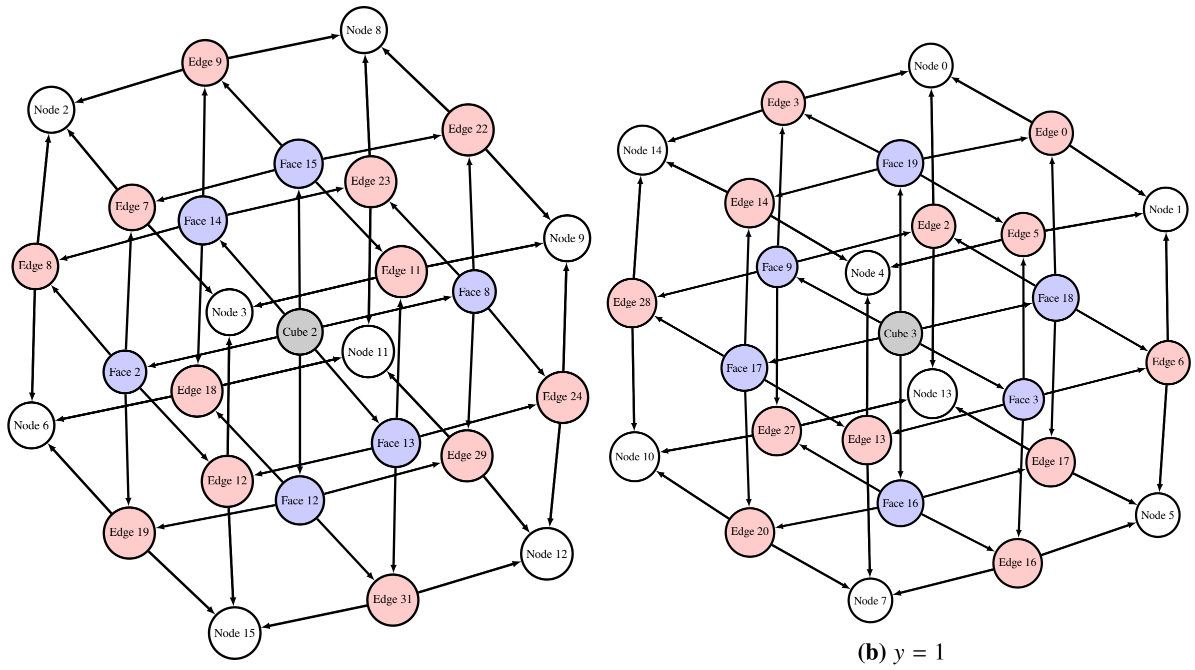
**Table 9 Hyperplane-bisector associations.**

| bisector | hyperplane |
|---|---|
| -1 | $x = -1$ |
| 1 | $x = 1$ |
| -2 | $y = -1$ |
| 2 | $y = 1$ |
| -3 | $z = -1$ |
| 3 | $z = 1$ |
| -4 | $t = -1$ |
| 4 | $t = 1$ |



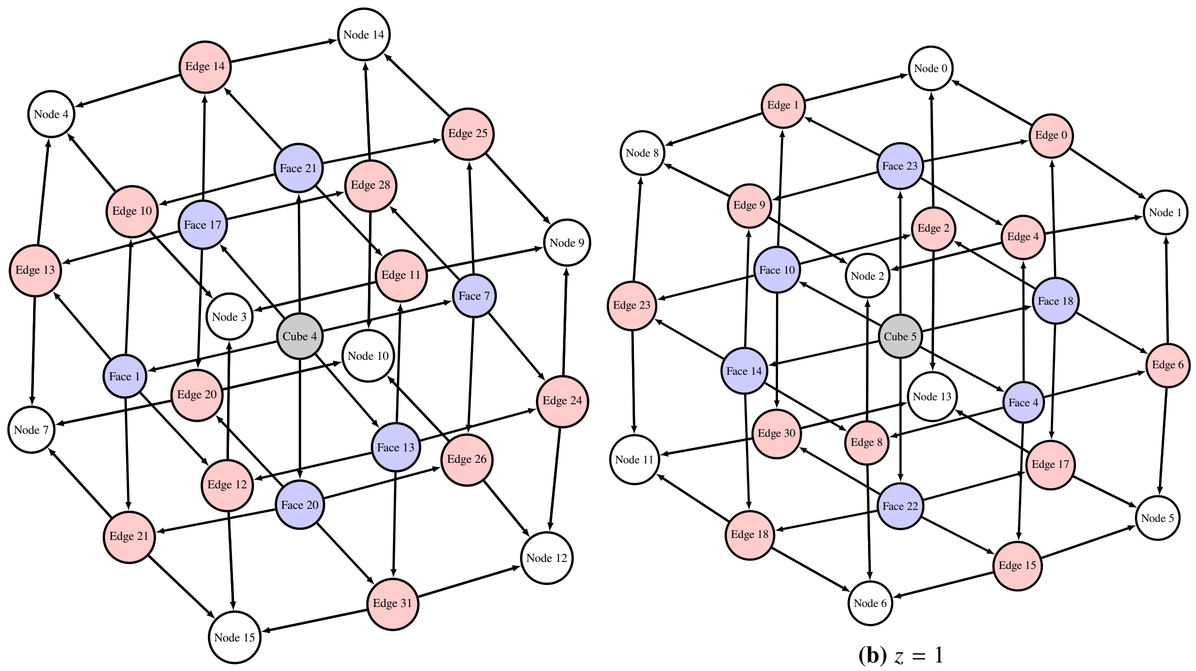
**(a)** $x = 0$ **(b)** $x = 1$

**Fig. 9 Bounding cube geometry hierarchy at constant $x$ hyperplanes.**



**(a)** $y = 0$ **(b)** $y = 1$

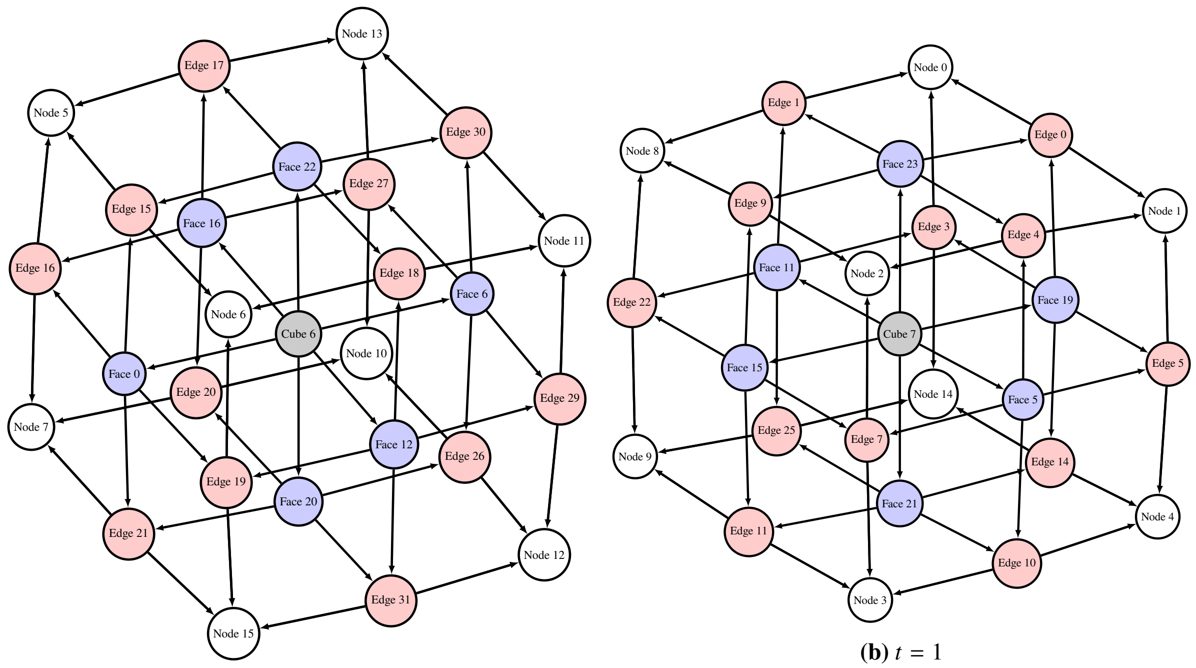
**Fig. 10 Bounding cube geometry hierarchy at constant $y$ hyperplanes.**



**(a)** $z = 0$

**(b)** $z = 1$

**Fig. 11 Bounding cube geometry hierarchy at constant $z$ hyperplanes.**



**(a)** $t = 0$

**(b)** $t = 1$

**Fig. 12 Bounding cube geometry hierarchy at constant $t$ hyperplanes.**