TENT-PITCHER: A MESHING ALGORITHM FOR SPACE-TIME DISCONTINUOUS GALERKIN METHODS

Alper Üngör[1], Alla Sheffer[2]

[1] *Dept. of Computer Science, University of Illinois at Urbana-Champaign, ungor@cs.uiuc.edu*
[2] *Computational Science and Engineering, University of Illinois at Urbana-Champaign, sheffa@uiuc.edu*

ABSTRACT

Space-time discontinuous Galerkin (DG) methods provide a solution for a wide variety of numerical problems such as inviscid Berger's equation and elastodynamic analysis. Recent research shows that in order to solve a DG system using an element-by-element procedure, the space-time mesh has to satisfy a cone constraint, i.e. that the faces of the mesh can not be steeper in the time direction than a specified angle function $\alpha()$. Whenever there is a face that violates the cone constraint, the elements at the face must be coupled in the solution. In this paper we consider the problem of generating a simplicial space-time mesh where the size of each group of elements that need to be coupled is bounded by a constant number k. We present an algorithm for generating such meshes which is valid for any nD$\times$TIME domain (n is a natural number). The k in the algorithm is based on a node degree in a n-dimensional space domain mesh.

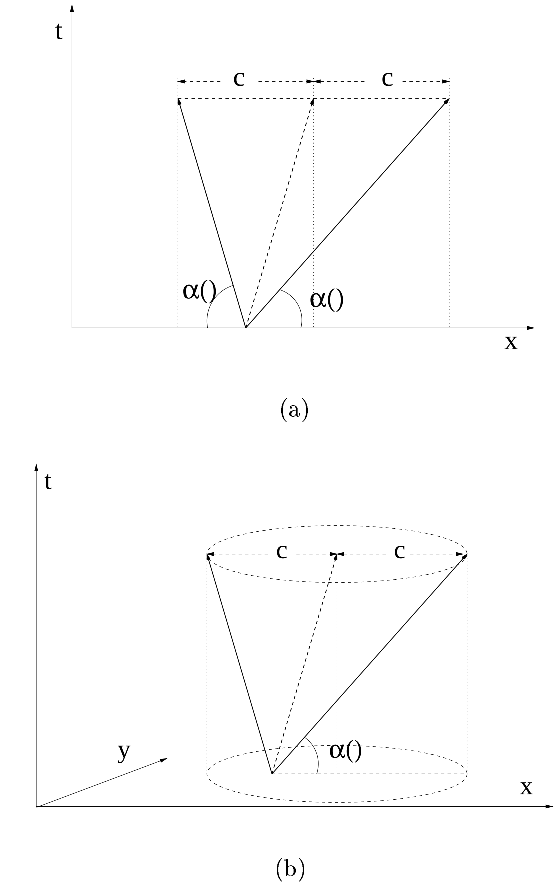
Keywords: space-time meshes, cone constraint, simplicial meshes

1 INTRODUCTION

Space-time finite element methods formulate numerical simulation problems by introducing time as an additional domain dimension for discretization. Space-time discontinuous Galerkin (DG) methods are studied in [5, 6, 3, 9, 12]. These methods depend on the notion of domains of influence for dynamic data. The domain of influence for each point p in the space-time domain is a cone with vertex p whose boundaries are determined by the wave speed in the material (Figure 1). The element-by-element solution strategy proposed by Yin *et al.* [12], and Lowrie *et al.* [3] supports unstructured, fully-discontinuous space-time discretizations and enforces element-wise conservation. Hence, it removes the need to solve a large global system of equations.

A space-time nD$\times$TIME mesh is composed of (n+1)-dimensional elements. The mesh has to satisfy a special constraint to enable a direct element-by-element solution using this technique. The *cone constraint* requires that the dihedral angle of each interior mesh face with respect to the space domain is less than or equal to a specified angle function $\alpha()$. It is a function of both position in the space time domain and mesh face orientation. The value of $\alpha()$ at a point p is determined by the boundary of the domain of influence cone at p (Figure 1). If the characteristic equations for the analysis problem are linear and the material properties are homogeneous, $\alpha()$ is constant through the entire domain. In Figure 2, we show a mesh which satisfies the cone constraint in 1D$\times$TIME domain, and a possible ordering of the elements to be solved using an element-by-element procedure.

If a mesh face violates the cone constraint, the analysis on the elements adjacent to the face can no longer be performed independently and the DG method has to solve the equation system on the elements that share this face simultaneously. This is called *coupling*.



(a)

(b)

Figure 1. The cones imposed by the characteristic equations (a) 1D×TIME (b) 2D×TIME

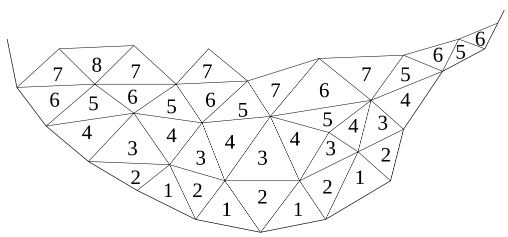


Figure 2. Ordering of elements for an element-by-element procedure. Elements with the same number can be solved independently or in parallel, elements with higher numbers depend for the solution on the output of those with lower ones.

When several faces violate the cone constraint, the entire set of adjacent elements involved has to be coupled.

When solving the DG system, the cone constraint function is often unknown *a priori* and is computed as part of the solution. In this case, the mesh generation needs to be performed in parallel to the numerical solution, using an advancing front procedure where the cone constraint at the front is already known.

Richter [4] shows that to achieve accurate results it is not sufficient to simply satisfy the cone constraint, but it is also important for the mesh to closely follow the cone constraint, i.e. have as many faces as possible whose dihedral angle with respect to space is equal to $\alpha()$ (at the face location).

An additional difference between DG analysis and standard FE methods is that the DG analysis imposes no restriction on the mesh element shape. Tetrahedral, hexahedral and mixed meshes — possibly with other element shapes — are acceptable.

Based on the requirements above, the ideal DG mesh should consist of elements, nearly all of whose faces have a dihedral angle equal to $\alpha()$ with respect to the space domain. In 1D×TIME this can be achieved using triangular elements [11]. However it seems that in 2D×TIME and higher dimensions, it can not be achieved by using basic mesh element (hexahedra/tetrahedra). Hence the requirement has to be relaxed by either building faces which satisfy the cone but can be significantly below it [3, 8, 10] or by allowing some of the faces to violate the cone constraint.

Contribution In this paper we use the second alternative. We redefine the meshing problem by letting some of the mesh faces to violate the cone constraint but request that the size of each group of elements which must be coupled is small and limited by a constant value. This relaxation does not restrict the use of space-time meshes which satisfy this definition for DG methods, but it does require element coupling during the solution procedure. However, if we regard each group as a single element, an element-by-element technique can still be used.

We provide a provably correct algorithm to solving this problem for any nD×TIME domain. This is the first method which handles non-uniform cone constraints in 2D×TIME and higher dimensions. Since the algorithm follows the advancing front approach it can be used when $\alpha()$ is unknown *a priori*.

Outline The rest of the paper is organized as follows. The next section reviews the previous approaches to solve the cone constrained space-time meshing problem. In Section 3 we formally define a new meshing problem. Section 4 presents the new algorithm. The verification of the algorithm is given in Section 5. In Section 6 the algorithm is demonstrated on 1D×TIME and 2D×TIME examples. Finally, Section 7 summarizes the paper and outlines aspects of future work.

2 PREVIOUS WORK

Several methods were recently proposed to generate space-time meshes for the 2D×TIME domain, which satisfy the cone constraint for analysis problems with linear characteristic equations, where the angle constraint is assumed to be uniform (i.e. $\alpha()$ is constant with regard to both location and orientation).

The method suggested by Lowrie *et al.* [3] uses pyramid elements to generate 2D×TIME meshes satisfying the cone constraint. The main disadvantage of the method is that it requires a fully structured quadrilateral mesh of the 2D space sub-domain as input.

Two other methods were recently presented by the authors of this paper [8, 10]. A short description of the algorithms is given below.

Tetrahedra-based algorithm[10] (Figure 3) A solution based on simplicial elements starts with an initial (time t_0) non-obtuse triangular mesh for the space domain [1]. Then the algorithm constructs time slabs filled with tetrahedra in three steps. The first step constructs the face-based tetrahedra by computing the center of the largest inscribed circle of the base triangle. The second step involves building edge-based tetrahedra by connecting the apexes of the face-based ones. Finally, each vertex cavity (polygon based pyramid) is decomposed into tetrahedra by triangulating its base. Once a layer is constructed, it can be flipped over to construct the next. Üngör *et al.* [10] prove that this algorithm guarantees satisfaction of the angle constraint. A drawback of the algorithm is that in order to satisfy the cone constraint for the entire mesh many of the mesh faces will have to be significantly below the cone constraint. To avoid the generation of the sub-optimal faces the paper suggests generating pyramid elements or groups of simplicial elements.

Hexahedra-based algorithm [8] (Figure 4) This approach starts with a 2D quadrilateral space mesh at the initial time t_0. It generates an element layer by sweeping the 2D mesh in the time direction. The procedure used for computing the new layer node locations guarantees that the sweep faces satisfy the angle constraint. Similar to the tet-based method, a single mesh layer can be repeatedly flipped over to fill the space-time domain once it has been constructed. The main disadvantage of the algorithm is that it generates a sweep mesh topology which can be inadequate when large changes of the mesh boundary occur over time.

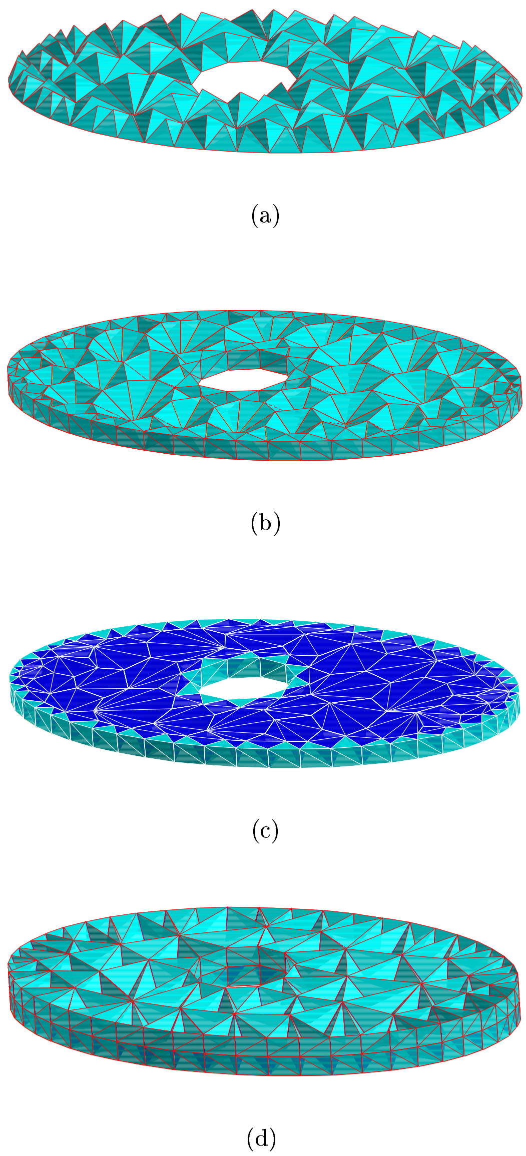
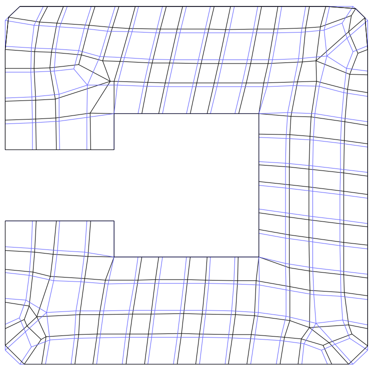
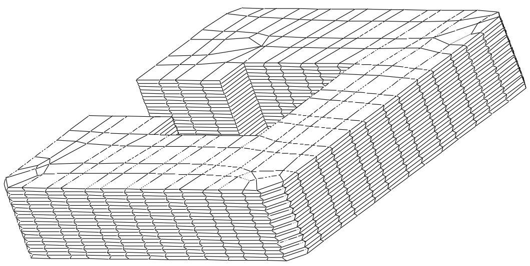


Figure 3. Tetrahedra based algorithm. (a) Face based tetrahedra; (b) edge based tetrahedra; (c) vertex based tetrahedra; (d) flipping over of vertex based tetrahedra.

A later study considers algorithms which handle the more general case of nonlinear hyperbolic problems that require meshes that satisfy a non-uniform cone constraint [11]. The algorithms above can be extended to handle non-uniform cone constraint by choosing at each layer the harshest cone constraint as the one to be satisfied by the entire layer. While this gives us a



(a)



(b)

Figure 4. Hexahedra based algorithm. (a) Location in space of the original and new layer nodes (original mesh shown in grey, new layer in black). (b) final space-time mesh with hex elements.

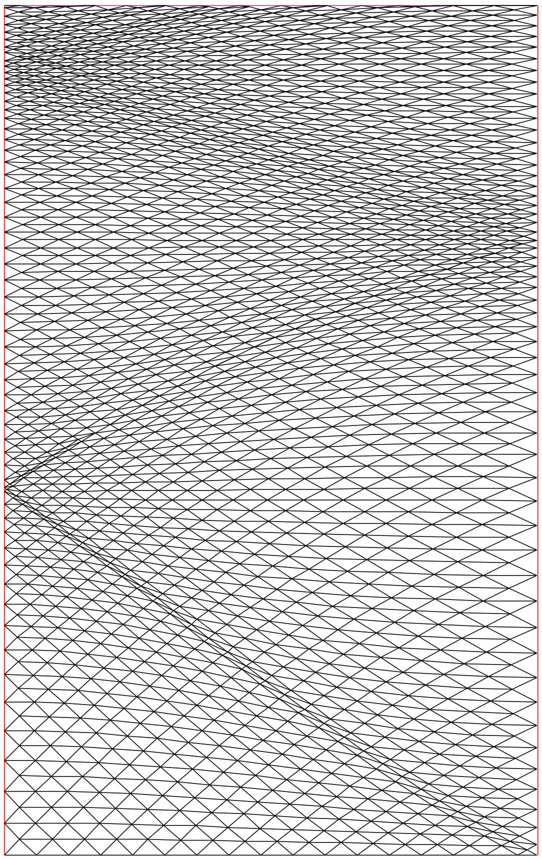


Figure 5. Output of the intersection based 1D×TIME algorithm. The size of the mesh is strongly driven by the cone constraint.

mesh satisfying the cone constraint it creates a large number of faces which are much lower than their local cone constraint. This reduces the analysis accuracy [4]. It also significantly slows the meshing process and generates unnecessarily small elements.

To satisfy the cone constraint locally, an element-by-element advancing front procedure has to be used rather then the layer based approaches above. Üngör *et al.* [11] consider this approach for the 1D×TIME domain, and show that even in this simple domain the problem is rather difficult. The algorithm is based on computing the intersections of the cones that are imposed on two neighboring nodes. The method then cuts the generated regions to obtain a simplicial mesh. The output of the algorithm is given in Figure 5.

3 PROBLEM DEFINITION

To formalize the new cone constrained meshing problem, the following definitions are used.

Definition 1 (Cone Constraint Function) *The* cone constraint function $\alpha(x,\theta)$ *defines the required dihedral angle between a mesh face and the space domain where x is a point in space-time domain D and $0 \leq \theta < 2\pi$ denotes the orientation of the mesh face around the time axis. A mesh face is said to* satisfy *the cone constraint function if the face angle is less than or equal to $\alpha()$.*

The following assumptions are used to guarantee the algorithm validity.

- $\alpha(x,\theta)$ is Lipschitz (Section 5).
- $\alpha(x,\theta) \geq \alpha_0 > 0$, i.e. to satisfy the cone constraint faces are never required to be horizontal.
- $\alpha(x,\theta) \leq \alpha_1 < \frac{\pi}{2}$. This means that mesh faces that satisfy the cone constraint are never vertical with respect to space.

For many DG problems $\alpha()$ is not known *a priori* and is part of the solution.

Definition 2 (k-satisfaction) *A mesh M of a domain D k-satisfies a cone constraint function*

$\alpha()$ if the mesh elements can be subdivided into groups of k or less elements such that the outside faces of each group satisfy the cone constraint.

This condition implies that when solving the analysis problem within each group all elements have to be addressed simultaneously, but between the groups there exists an order allowing us to apply the solution procedure to one group at a time.

Note that a simplicial mesh which k-satisfies a cone constraint can in fact be viewed as a mesh which satisfies the cone constraint for each element if each coupled group of simplices is viewed as a mesh element.

Problem In this paper we formulate a new space-time meshing problem as follows.

Definition 3 *Given a space-time domain D, a mesh of the space sub-domain at time t_0 and a cone constraint function $\alpha()$, build a mesh of D that k-satisfies $\alpha()$.*

A polynomial-time algorithm is proposed to solve it. The algorithm does not depend on the rank of the space domain, i.e. it can be applied to any nD$\times$TIME domain. In this paper the algorithm is demonstrated in 1D$\times$TIME and 2D$\times$TIME but both the algorithm and the proofs of its validity are directly applicable to any n. We show that for 1D$\times$TIME the algorithm produces a mesh which k-satisfies $\alpha()$ with $k = 2$ and for 2D$\times$TIME it k-satisfies $\alpha()$ with an upper bound of $k = 12$ (on average k is at most 6).

4 ALGORITHM

4.1 Advancing Front Procedure

As mentioned above, the cone constraint function is often unknown *a priori* and is in fact part of the numerical simulation solution. Hence the mesh cannot be computed *a priori* and has to be computed during the simulation.

The advancing front algorithm manages the interaction between the numerical simulation code and the mesh generation algorithm (see Figure 6). It starts by meshing the space domain. We use an acute triangulation for the space domain. The reason for this choice will be further discussed in Section 5. The advancing front is initialized to this initial space mesh. In the advancing step, the mesh generator constructs new element(s). Every time a (set of) new element(s) is constructed, a numerical problem is solved on the new element(s). Based on this solution an estimate of the cone constraint function is fed back to the mesh generator. Note that the cone constraint at the new mesh node(s) is unknown *a priori*, hence element adjustment might be required to satisfy the cone constraint as required.

```
ALGORITHM Advancing Front
  construct the space mesh at initial time
  let the front be this initial mesh
  while domain D is not covered
    advance the front
        computing the new element(s)
    solve the numerical problem
        on the new element(s)
    adjust the element(s) if necessary
    update the cone constraint on the front
```

Figure 6. Interface routine between the numerical simulation and the mesh generation

The tent-pitcher algorithm provides the advancing step of this main algorithm. It is intuitive to think of the advancing front surface as a terrain. The tent-pitcher adds a group of elements to the mesh by projecting an advancing front node p and connecting the new node to the neighbors of p. The set of newly introduced elements can be seen as a *tent* built on the terrain. The outer faces of each tent satisfy the cone constraint function. The tents consist of at most k tetrahedra, resulting in a space-time mesh that k-satisfies the cone constraint, where k is the degree of the node p in the space mesh.

4.2 Tent-Pitcher Definitions

The following definitions are used in the algorithm description. Let p be a node on the advancing front. $N(p)$ denotes the set of nodes that are on the current advancing front and connected to p via a single edge. For each node p on the advancing front, $pole(p)$ is a ray emanating from p into the uncovered domain. It is used to project p. In the current implementation, each pole is parallel to the time axis. To guarantee the validity of the algorithm, the distance between adjacent poles has to be bounded from below and above. Note that using cone axises as poles can easily violate this constraint. To simplify the description for a general $(n+1)$D (nD$\times$TIME) domain we use the following 3D terminology: a *tetrahedron* is an $(n+1)$D simplex, a *face* is an nD simplex, an *edge* is an $(n-1)$D simplex, and a *plane* refers to an nD hyper-plane. Let $E(p)$ denotes the set of edges all of whose endpoints are in $N(p)$. Each point on an edge $e \in E(p)$ defines a cone constraint. There are two maximal (largest dihedral angle) half planes one on each side of the edge that are below all the cones based on e. We call these half planes that satisfy the cone constraints on e, the *cone planes*. Let $P_p(e)$ denote the cone plane emanating from edge e towards $pole(p)$. Let $\alpha_p(e)$ denote the dihedral angle between the cone plane $P_p(e)$ and the space. The DG method keeps a record of the cone constraint at the mesh nodes. The cone constraint on an edge is computed as a linear interpolation of the cone constraint at its endpoints. Hence, both the shallowest and the steepest cones are defined at the endpoints. Using these definitions a $tent(p)$ is defined as follows.

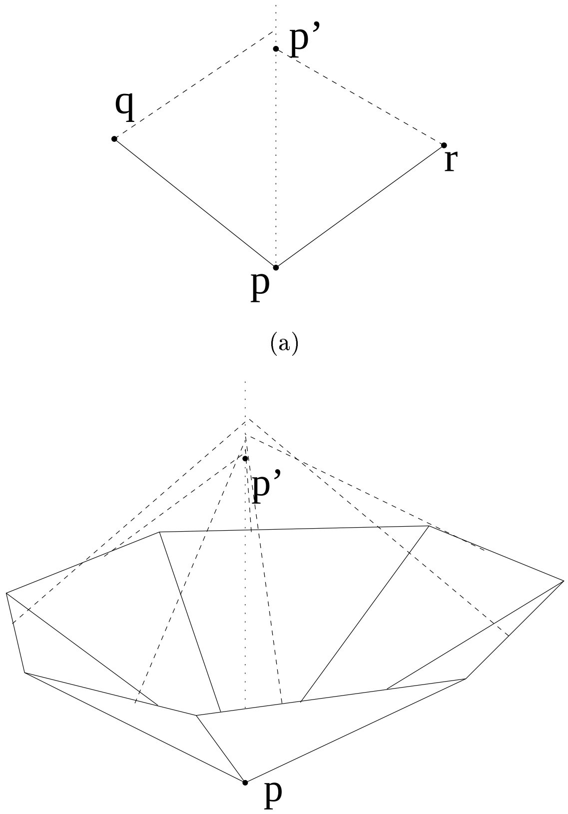


Figure 7. Generating a tent at node p based on the cone planes from its neighbors. The apex p' is chosen as the lowest of the set of points where the cone planes intersect $pole(p)$ (dotted line).

Definition 4 (Tent) *For each node p, $tent(p)$ is a volume defined by p, $N(p)$, and a projection node p' on $pole(p)$. It is formed by connecting p' to all the nodes in $N(p)$. The volume is a union of tetrahedra defined by p, p' and the end nodes of $e \in E(p)$. The node p is called the* base node *and the new node p' is called the* apex *of the tent.*

The apex of a tent based on node p is determined by the cone planes of $E(p)$ edges with respect to p. By definition each such plane intersects $pole(p)$. p' is chosen as the lowest intersection point in the time direction. Let (e, p') denote a tent face constructed by edge $e \in E(p)$ and p'. Figure 7 demonstrates the tent construction in 1D$\times$TIME and 2D$\times$TIME.

Based on the tent definition, the number of tetrahedra which form $tent(p)$ is at most $|N(p)|$.

To prevent the construction of tents with zero or near zero volume we identify some of the nodes as illegal to pitch a tent on.

Definition 5 (Illegal base node) *A mesh node p is called an* illegal base node *for building a tent if there exists a mesh node $q \in N(p)$ where q is below p in the time direction.*

Using this definition it is illegal to pitch a tent on the peak or the slope of the mountain (Figure 8). One can however put a tent on a valley, cavity or a flat area.

This definition is in fact over-restrictive and can be relaxed taking into account the $\alpha()$ values at $N(p)$ and the space mesh size. We use this over-restrictive version to simplify the algorithm description and verification.

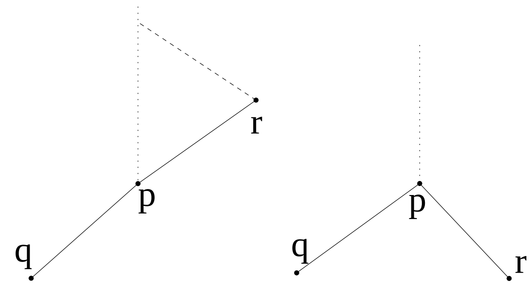


Figure 8. Mesh nodes on the front where it is illegal (forbidden by the Park Ranger!) to pitch a tent on: (a) slope; (b) peak.

4.3 Tent-Pitcher Algorithm

At each step of the advancing front, the tent-pitcher chooses a legal base node to pitch a tent on. Such a node always exists, since the lowest node on the front is always legal. The order of choosing the base node affects the mesh quality. Since the global information on $\alpha()$ is not available, a local optimization strategy has to be used. To accommodate this, each base node is assigned a value based on the *quality* of the potential tent on it. An illegal base node is assigned a negative value. At each advancing step the base node with the highest quality value is chosen. After a tent is built, the quality values of the neighbor nodes are recomputed. This involves recomputing the tents of those nodes (Figure 9 (b)). For efficient computation we use a priority queue to store the nodes. The quality value is used as the key to the priority queue. The priority queue is initialized from the original space mesh nodes. The algorithm is formalized in Figure 9 (a).

```
ALGORITHM Tent-Pitcher
  Pull the best tent from the priority queue
  Construct the tent by connecting the apex
    of the tent to the neighbor nodes
  Recompute the tents of the neighbor nodes
    and relocate them in the priority queue
    based on the quality of the new tents
  Insert the tent of the new apex node
    into the priority queue
    (with negative quality)
```

(a)

```
ALGORITHM Tent(p)
  If pitching a tent at p is illegal
      return -1 as quality
  Compute the intersection of the cone plane
      of each e ∈ E(p) with pole(p)
  Choose the lowest intersection as p'
  Compute and return the quality of tent(p)
```

(b)

Figure 9. Pitching a tent

When choosing a tent base, the quality metric is chosen to locally optimize some mesh criteria such as minimizing the deviation from the desired cone constraint, or maximizing the tent volume. Other heuristics can be used as well. Here we discuss some alternatives.

- *Lowest apex:* This metric suggests advancing generating the tent with the lowest apex. While it often sacrifices the local element quality, we had found that in practice it tends to improve the overall mesh quality.
- *Optimal face angle:* To achieve accurate results, it is important for the mesh faces to be as close as possible to the cone constraint function. To capture the deviation of the tent faces from the cone function, we measure the angular distance between the cone planes of E(p) and the actual tent faces corresponding to them. Each of those tent faces is defined by $e \in E(p)$ and the tent apex p'. Let $\beta_p(e)$ denote the angle between a face (e, p') and the space domain. The tent quality $V(p)$ is measured as

$$V(p) = \sum_{e \in E(p)} [\alpha_p(e) - \beta_p(e)]$$

  Note that the summation term is always non-negative because the tent faces satisfy the cone constraint function. The tent with the smallest $V(p)$ is chosen as optimal.
- *Maximal tent volume:* The location of the apex is determined as before. The volume of each tetrahedron is computed and summed to obtain the volume of the tent. The volume can be normalized by the local mesh size to disable the obvious priority of larger elements.
- *The lowest node:* This heuristic suggests advancing using the lowest node on the front ignoring the tent quality. Note that the lowest node is never an illegal base node.

5 VERIFICATION OF THE ALGORITHM

Below we prove that the presented algorithm creates a valid mesh of the domain which k-satisfies the cone constraint function $\alpha()$. First, we show that the generated simplices are valid and that the tent faces satisfy the cone constraint. Then, an upper bound is given on the number of simplices in each tent. Finally, we show that the algorithm terminates.

Definition 6 (Lipschitz) *A function $f()$ is said to be Lipschitz on a domain D if there exists a constant c such that for any two points $x_1, x_2 \in D$ the following inequality is true,*

$$|f(x_1) - f(x_2)| \leq c\|x_1 - x_2\|$$

We can also refer to $f()$ as c-Lipschitz.

In the next proofs the following definitions are used. Let s be the smallest mesh feature in the original space mesh and let $|s|$ denote its size. Let S be the largest distance between two adjacent nodes in the space mesh. Since the projections are vertical, the smallest feature on the front is always at least $|s|$ and the horizontal (space sub-domain) distance between two adjacent front nodes is always less or equal to S.

Lemma 1 (Validity) *Each mesh element has a positive volume larger than $\epsilon > 0$.*

PROOF. To ensure mesh validity we need to show that the generated simplices have positive volume. To show this it is enough to prove that for each tent the apex p' is above p in the time axis. It is sufficient to show that for each $e \in E(p)$, p_e (the intersection of the cone plane $P_p(e)$ with $pole(p)$) is above p in the time axis. The point p_e can be represented by $\sum_i a_i e_i + v$ where e_i are the space coordinates of the end-nodes of e, v is a vector orthogonal to e and a_i are the appropriate coefficients. Since the triangulation is acute, there exists a choice of coefficients a_i such that $a_i > 0$ for all i and $\sum_i a_i = 1$. Given this choice of coefficients, since e and p_e are in the plane $P_p(e)$, v is a vector in the plane $P_p(e)$ (Figure 10). Based on the plane construction, the time component of v is non-negative. Without loss of generality, let e_1 be the lowest end node of e. The vectors $e_i - e_1,\ i = 2 \ldots n-1$ and v span the plane $P_p(e)$. Hence the plane dihedral angle $\alpha_p(e)$ is a monotonously increasing function of the angles of those vectors, and $\alpha_p(e)$ is zero only when the angles of all the vectors are zero. All of $e_i - e_1$ have non-negative time components. Since $\alpha_p(e) > \alpha_0 > 0$, there exist $\epsilon_1 > 0$ and $\epsilon_2 > 0$ such that either the angle between v and the space is bigger than ϵ_1 or the angle of some $e_j - e_1$ with respect to space is bigger than ϵ_2. Hence the angle of

$$p_e - e_1 = \sum_i a_i e_i + v - e_1 = \sum_i a_i(e_i - e_1) + v$$

with respect to space is bigger than $\epsilon_3 > 0$. Since this angle is positive, p_e is higher than e_1, hence since p is a valid base node, p_e is higher than p.

As defined $|s|$ is the size of the smallest mesh feature. Using $|s|$ and ϵ_3 we can put a lower bound on the tent height $h(p) = \|p - p'\|$:

$$h(p) > |s| \tan(\epsilon_3)$$

Hence, there is a lower bound $\epsilon > 0$ on the tent volume. Since at every iteration the mesh nodes advance at least this amount in the time axis, the tent-pitcher algorithm eventually fills the space. □

Lemma 2 ($\alpha()$ satisfaction) *If the cone constraint function $\alpha()$ is Lipschitz and S is smaller than a given constant C, then the tent faces satisfy the cone constraint.*

PROOF. To prove this we need to show that each tent face (e, p') defined by $e \in E(p)$ and p' satisfies the cone constraint along its edges. By construction, p' is above p and not above p_e, hence the dihedral angle of (e, p') with respect to space is bounded by the angles of the faces (e, p_e) and (e, p). Since both

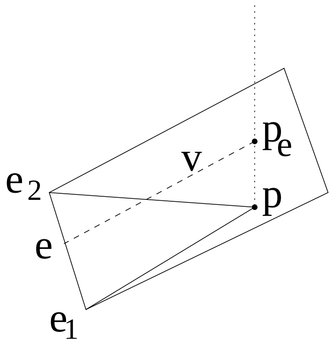


Figure 10. The intersection of the plane $P_p(e)$ with $pole(p)$.

(e, p_e) and (e, p) satisfy the cone constraint at e by construction, the face (e, p') does it as well. All edges of (e, p') besides e, have as their end-nodes p' and a subset of the nodes e_i. Let's consider an edge d with the end-nodes p' and $d_j = \{e_i\} \backslash e_j$. The faces (e, p_e) and (e, p) satisfy the cone constraint along the edges defined by d_j and p_e or p respectively. Those edges have the same space orientation as d, hence if (e, p_e) and (e, p) satisfy the cone constraint at d_j so does (e, p'). As descried above, the cone constraint along an edge is bounded by the cone constraint at the edge end-nodes. Since the constraint at d_j is satisfied only the cone constraint at p' (with respect to the orientation of the edge d) needs to be considered.

If $\alpha()$ is *Lipschitz* with a constant l on a closed domain D, the tangent of $\alpha()$ is *Lipschitz* as well with a different constant c (c-*Lipschitz*). This implies that

$$|\tan(\alpha(x_1, \theta_0)) - \tan(\alpha(x_2, \theta_0))| < c\|x_1 - x_2\|$$

for any $x_1, x_2 \in D$ and any orientation θ_0.

Let C be $\frac{1}{c}$, i.e. $S < \frac{1}{c}$. Given this constraint, we show that the face (e, p') satisfies the cone constraint at p'.

If the face (e, p') violates $\alpha_{p'}(e)$, it has to be either above the upward cone at p' (Figure 11(a)) or below the downward cone (Figure 11(b)). In the first case the upward cone at p' intersect the face (e, p), in the second the downward cone intersect the face (e, p_e) (Figure 11). Lets consider the first case. The face (e, p) satisfies $\alpha_p(e)$, by previous construction. Hence, the upward cone of p' intersects the upward cone of p.

For the cones to intersect, the situation in Figure 12 has to occur, i.e. for two points p and p' with the same space coordinates, the cones intersect at a distance $d < S$.

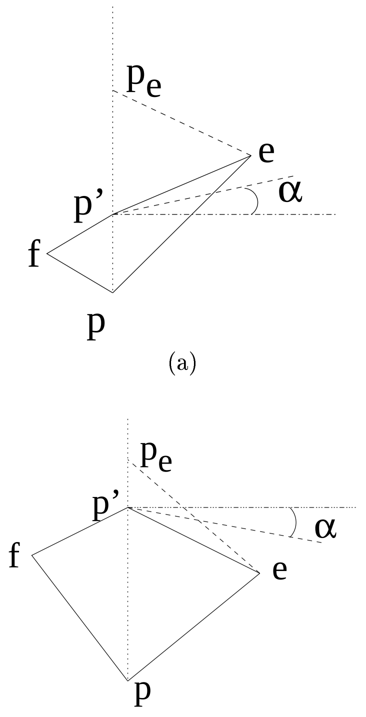


Figure 11. Two possible cases of cone constraint violation. (a) violation by a face of the upward constraint at p'. (b) violation of the downward constraint at p'.

Using the notations in Figure 12 and the *c-Lipschitz* property (Eq. 5), this means that for $h_1, h_2, h_3 > 0$

$$\begin{aligned} ch_1 &= c\|p - p'\| \\ &> \tan(\alpha_p(e)) - \tan(\alpha_{p'}(e)) \\ &= \frac{h_1 + h_2 + h_3}{S} - \frac{h_2}{S} \\ &= \frac{h_1 + h_3}{S} \\ &> \frac{h_1}{S} \end{aligned}$$

which implies (divide by h_1)

$$c > \frac{1}{S}$$

contradicting the mesh size constraint. The case of intersecting (e, p_e) can be addressed similarly. □

Lemma 3 (k-satisfaction) *The tent-pitcher algorithm generates a mesh that*

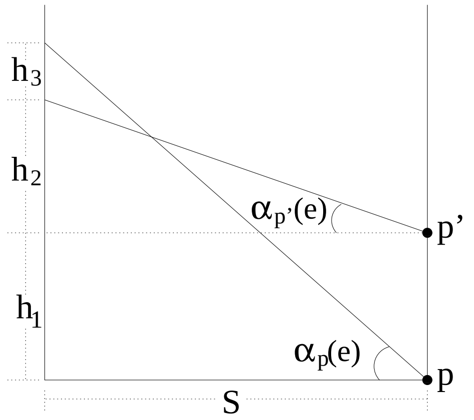


Figure 12. Intersection of two cones starting from two nodes on one pole.

k-satisfies $\alpha()$ where k is the highest node degree in the input space sub-domain mesh.

Proof. The number of simplicial elements introduced in the space-time mesh is exactly $|N(p)|$ if the p is an interior node and $|N(p)| - 1$ if p is on the boundary. $|N(p')|$ is by construction equal to $|N(p)|$. By definition $|N(p)|$ is equal to the degree of p in the advancing front mesh. Also all the outer faces of a tent satisfy the cone constraint function $\alpha()$ (Lemma 2). Hence, the number of mesh elements that need to be coupled is bounded by the maximum degree node in the space mesh. □

The following lemma states an upper bound on the maximum degree of a node in 1D and 2D simplicial meshes.

Lemma 4 (k in 1D and 2D) *It is possible to construct 1D and 2D simplicial meshes that have degree at most k.*

Proof. In 1D k is equal to 2. We provide both the average value and maximum value for k in 2D. Euler's formula [7] suggests that $n - e + f = 2$ for planar graphs, where n is the number of nodes, e is the number of edges and f is the number of faces. Simple counting argument for edges and faces gives $3f \leq 2e$ (there are 3 edges for each face and 2 faces for each edge except the boundary edges). Factoring and substitution leads $3n - e \geq 6$, which implies $e \leq 3n - 6$ Since the total node degree, d, in a graph is twice the number of edges, $d \leq 6n - 12$. Hence the average degree in a triangulation is at most 6.

To prove a bound from above in 2D we use a result of Chew [2]. Chew proposed a Delaunay based algorithm that triangulates a given polygonal domain into a mesh with a minimum angle of 30°. The generated mesh has almost uniform spacing, however, is still within a constant factor of the optimal size. This result gives a bound of 12 on the degree of a mesh node. □

The main theorem of this paper directly follows from the mesh validity and the lower bound on element size, $\alpha()$ satisfaction, and k-satisfaction lemmas (Lemma 1, 2 and 3 respectively).

Theorem 1 (Verification)
The tent-pitcher algorithm terminates generating a mesh that k-satisfies the cone constraint.

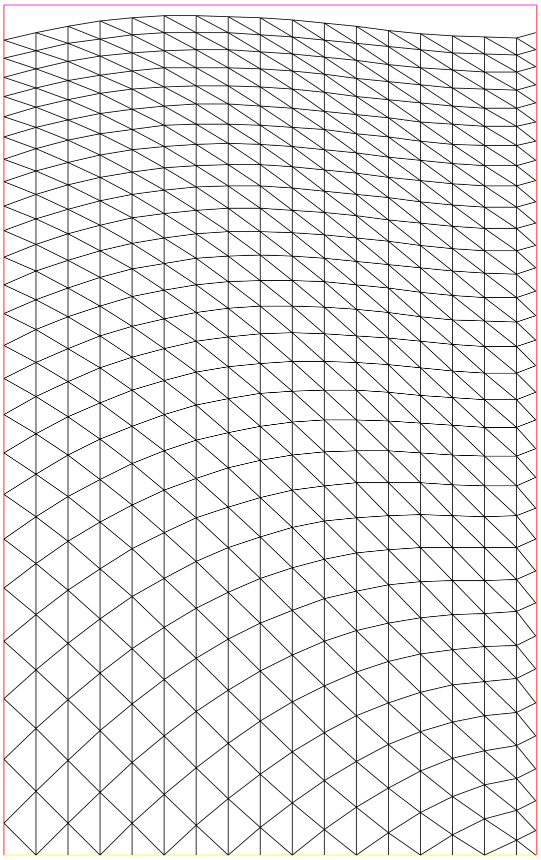


Figure 13. 1D×TIME mesh generated by the tent-pitcher algorithm. The size of the mesh is weakly driven by the cone constraint.

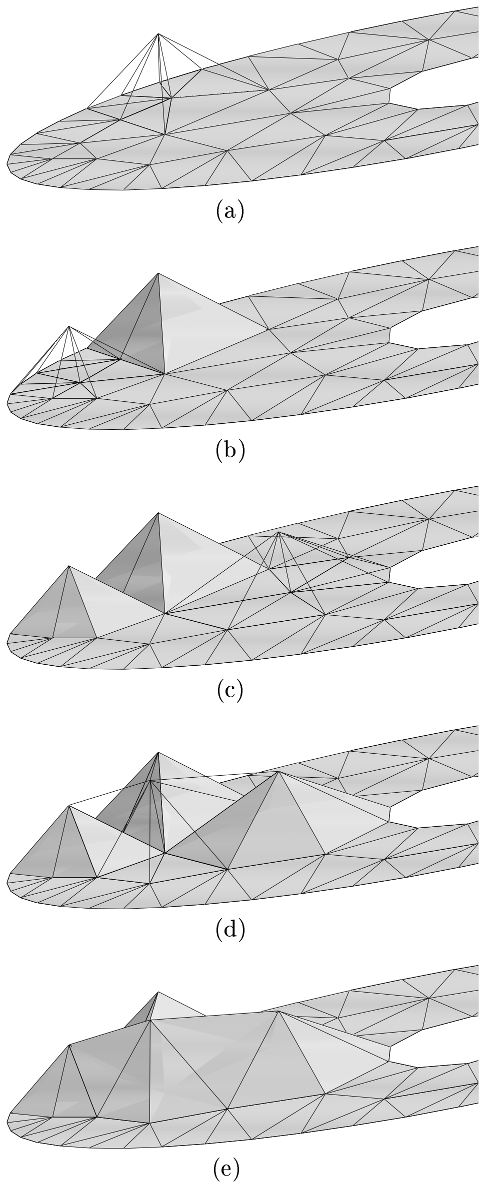


Figure 14. Illustration of the Tent-Pitcher algorithm in 2D×TIME. The existing mesh is shaded and the new element is shown as wire-frame.

6 EXPERIMENTS

The Tent-Pitcher algorithm is implemented in 1D×TIME (Figure 13) and 2D×TIME (Figure 14). A comparison of the 1D×TIME mesh with the output of the previous intersection based algorithm (Figure 5) suggests that the size of the mesh is less related to the cone constraint function, since the tent-pitcher algorithm does not modify the space sub-domain mesh. As can be seen from the figure, to reduce k in 1D×TIME from 2 to 1, i.e. make all the mesh faces obey the cone constraint, all that is required is to remove the vertical diagonal in each element cluster, replacing it by the nearly-horizontal one. Hence, the only disadvantage of the tent based algorithm is that the number of faces that exactly follow the cone constraint in the output mesh of the Tent-Pitcher algorithm is smaller than in the output mesh of intersection based algorithm. Indeed almost all of the edges of 1D×TIME mesh in Figure 5 satisfy the cone constraint exactly. Although this states that the Tent-Pitcher algorithm performs weaker than the previous algorithms in 1D×TIME, our main motivation is space-time meshing in higher dimensions. It is not possible to extend the intersection based method to higher dimensions simply because of the increasing complexity of hyper-planes. In the 1D×TIME algorithm we used the *lowest base* as the criteria for choosing the tent base (Section 4).

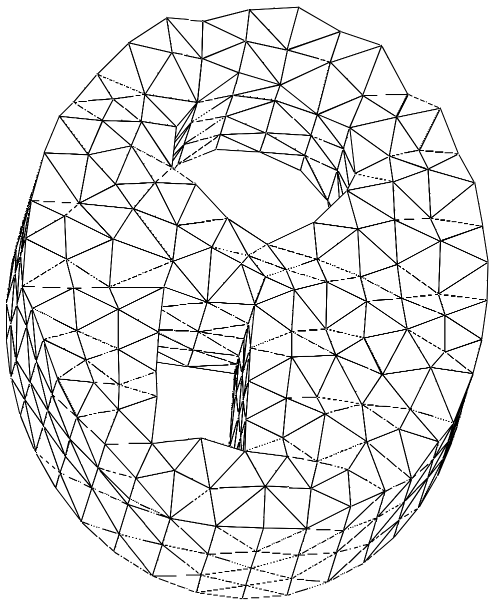


Figure 15. 2D×TIME mesh generated by the tent-pitcher algorithm.

Figures 14 and 15 show the implementation of the algorithm in 2D×TIME. The extension from 1D×TIME to 2D×TIME is mostly straight forward. The only point requiring special attention is the computation of the cone planes $P_p(e)$ based on an edge $e = (r, s)$ and a dihedral angle $\alpha_p(e)$. To compute the plane $P_p(e)$ defined by $ax + by + cz + d = 0$ we do the following. First, normalize the plane equation by setting $a^2 + b^2 + c^2 = 1$. Using the formula of an angle between two planes $a_1x + b_1y + c_1z + d_1 = 0$ and $a_2x + b_2y + c_2z + d_2 = 0$ ($P_p(e)$ and $z = 0$ respectively) we get

$$\cos(\alpha_p(e)) = \frac{a_1a_2 + b_1b_2 + c_1c_2}{\sqrt{a_1^2 + b_1^2 + c_1^2}\sqrt{a_2^2 + b_2^2 + c_2^2}} = c.$$

This gives us the following system of equations

$$a^2 + b^2 + \cos(\alpha_p(e))^2 - 1 = 0 \qquad (1)$$
$$ar_x + br_y + \cos(\alpha_p(e))r_z + d = 0 \qquad (2)$$
$$as_x + bs_y + \cos(\alpha_p(e))s_z + d = 0 \qquad (3)$$

We obtain the plane by solving the system, and choosing the solution (of the quadratic equation) which gives the plane which goes upward on p's side of (r, s).

In the 2D×TIME algorithm we used the *lowest apex* as the criteria for choosing the tent base (Section 4). While it gives less optimal local results, it seems to improve the overall mesh quality.

7 CONCLUSION

The paper presents a new algorithm for generating space-time meshes that satisfy a non-uniform cone constraint. To achieve this, a notion of k-satisfaction is introduced and a new meshing problem is formulated. The presented algorithm is provably correct and can be applied directly to any nD×TIME domain.

Among the properties of the new algorithm are: it generates meshes which closely follow the cone constraint; it can handle non-uniform cone constraints and changes in the model geometry; it is formulated as an advancing front method and can be used to generate the mesh on-line during the analysis where $\alpha()$ is not known *a priori*; it is easy and efficient to implement.

Future areas of research include: integration with real space-time DG analysis applications; implementation of the method for 3D×TIME; providing a numerical bound for k for higher dimensions; providing a parallel implementation of the algorithm; analysis of different ordering strategies for choice of base nodes for the tents; and developing mesh adaptivity techniques.

8 ACKNOWLEDGMENTS

The work reported here was supported, in part, by the Center for Process Simulation and Design (NSF DMS 98-73945) and the Center for Simulation of Advanced Rockets (DOE LLNL B341494). We would like to thank Jeff Erickson and Robert Haber for helpful discussions.

REFERENCES

[1] M. Bern, S. Mitchell and J. Ruppert *Linear-Size Non-obtuse Triangulation of Polygons,* Proc. of 10^{th} Symp. on Computational Geometry, New York, 221-230, (1994).

[2] P.L. Chew, *Constrained Delaunay Triangulation*, Algorithmica, (4), 97-108, 1994.

[3] R. B. Lowrie, P. L. Roe and B. van Leer, *Space-Time Methods for Hyperbolic Conservation Laws*, Barriers and Challenges in Computational Fluid Dynamics, Kluwer, (6), 79-98, 1998.

[4] G. R. Richter, *An explicit finite element method for the wave equation*, Applied Numerical Mathematics, (16), 65-80, 1994.

[5] T.J.R. Hughes and G.M. Hulbert, *Space-Time Finite Element Methods for Elastodynamics: Formulations and Error Estimates*, Computational Methods Applications in Meachanical Engineering. (66), 339-363, 1998.

[6] C. Johnson, *Discontinuous Galerkin Finite Element Methods for Second Order Hyperbolic Problems*, Computational Methods Applications in Meachanical Engineering. (107), 145-157, (153).

[7] K. H. Rosen *Discrete Mathematics and Its Applications* McGraw-Hill, 1995

[8] A. Sheffer, A. Üngör, S.-H. Teng, and R. B. Haber, *Generation of 2D Space-Time Meshes Obeying the Cone Constraint,* Proc. of Int. Conference on Computational Engineering Science, Los Angeles, CA, August 2000.

[9] L. L. Thompson, *Design and Analysis of Space-Time and Galerkin Least-Squares Finite Element Methods for Fluid-Structure Interaction in Exterior Domains,* Ph.D. thesis, Stanford University, 1994.

[10] A.Üngör, C. Heeren, X.-Y. Li, A. Sheffer, R. B. Haber, and S.-H. Teng, *Constrained 2D Space-Time Meshing with All Tetrahedra,* Proc. of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland, August 2000.

[11] A.Üngör, A. Sheffer, and R. B. Haber, *Space-Time Meshes for Nonlinear Hyperbolic Problems Satisfying a Nonuniform Angle Constraint*, to appear in Proc. of 7th International Conference on Numerical Grid Generation in Computational Field Simulations, Whistler, British Columbia, Canada, September 2000.

[12] L. Yin, A. Acharya, N. Sobh, R. B. Haber, and D. A. Tortorelli, *A Space-Time Discontinuous Galerkin Method for Elastodynamic Analysis,* Proc. of International Symp. on Discontinuous Galerkin Methods, Salve Regina University, Newport, RI, May 1999.