Space-Time Meshes for Non-linear Hyperbolic Problems Satisfying a Non-uniform Cone Constraint

Alper Üngör
Alla Sheffer
Robert Haber

University of Illinois at Urbana-Champaign,
Urbana, IL, 61801
e-mail : ungor@cs.uiuc.edu, sheffa@uiuc.edu, r-haber@cse.uiuc.edu

Abstract

Space-time Discontinuous Galerkin (DG) methods provide a solution for a wide variety of numerical problems. For an element-by-element procedure to be used to solve a DG system, the space-time mesh has to satisfy a *cone constraint*, which dictates that the mesh faces cannot be steeper in the time direction than a specified angle function $\alpha()$. This paper discusses algorithms that generate space-time meshes satisfying a *non-uniform cone constraint* (NUCC), i.e. $\alpha()$ varies through out the domain. We consider the extension of existing *uniform cone constraint* (UCC) meshing algorithms and present a new 1D$\times$TIME meshing algorithm for handling NUCC.

Introduction

Space-time Discontinuous Galerkin (DG) methods are becoming a popular tool in solving numerical problems like inviscid Berger's equation and elastodynamic analysis [1, 4, 7]. These methods utilize the notion of a domain 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 method of Yin *et al.* [7] supports unstructured, fully discontinuous space-time discretizations, enforces element-wise conservation and is compatible with direct element-by-element solution procedures. Hence, there is no need to solve a large global system of equations.

Direct element-by-element solutions using this technique require construction of a special mesh that satisfies the cone constraint. 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 $\alpha()$. The value of $\alpha()$ is determined by the boundaries of the domain of influence. If the characteristic equations, which yield the wave speed in the material, are linear and the material properties are homogeneous, $\alpha()$ is a *uniform cone constraint* (UCC), i.e. it has the same value through out the domain. However, the cone constraint can be *non-uniform* (NUCC) with the size of $\alpha()$ varying based both on the mesh face location in space and its orientation.

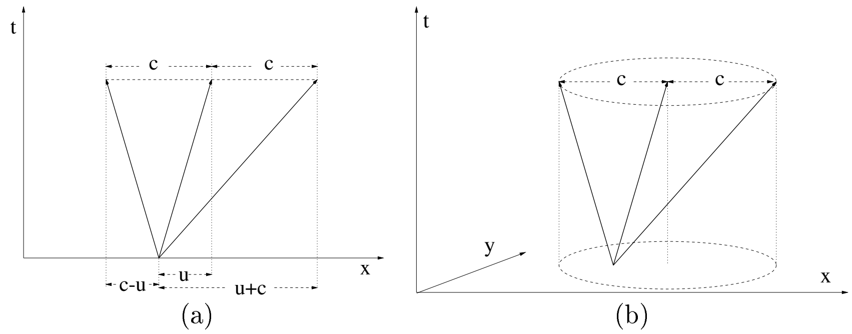


Figure 1: The cones imposed by the characteristic equations yielding the wave speed in the material: (a) 1D$\times$TIME; (b) 2D$\times$TIME.

When solving a DG system, the cone constraint function is often unknown *a priori* and is computed as part of the solution. In this case, the mesh has to be generated as the numerical problem is being solved. An advancing front procedure feeds the meshing module with the cone constraint function on the moving front which is a result of the numerical solution on the previously generated elements. Furthermore, it is important for the mesh to closely follow the cone constraint [2], i.e. have as many faces as possible whose dihedral angle with respect to space is equal to $\alpha()$ (at the face location). Finally, DG analysis not only imposes no restriction on the mesh element type, but allows mixed element meshes and non-conformal one-to-many adjacency relationships between mesh elements as well.

Previous Work - UCC Meshing

Several methods [1, 3, 5] were recently proposed to generate 2D$\times$TIME meshes satisfying the uniform cone constraint. Lowrie [1] presents a pyramid based meshing scheme. However, the scheme is applicable only when the initial quadrilateral mesh of the space domain is structured and has uniformly sized elements.

A hexahedral meshing algorithm is presented in [3]. It starts with an initial 2D quadrilateral space mesh and then tilts the projections of the edges onto the next layer, so that the face angles between the layers satisfy the angle constraint. This single mesh layer is then flipped over to fill the domain. However, this algorithm may generate a sweep mesh topology that can be inadequate when large changes of the mesh boundary occur over time.

Üngör *et al.* [5] present a solution using all tetrahedra. It starts with a non-obtuse triangular mesh of the initial space domain. A single time slab mesh is constructed in three steps. First, face-based tetrahedra are built by computing the in-centers of the base triangles. Then, edge-based tetrahedra are constructed by connecting the apices 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 is flipped over to construct the next. This algorithm guarantees satisfaction of the angle constraint. A drawback of the algorithm is that many mesh faces do not follow the cone constraint closely.

Contribution

In this paper, we address the *non-uniform cone constraint* (NUCC) meshing problem. We formally define the problem and consider the extension of existing UCC meshing algorithms to handle it. We discuss the drawbacks of using such extensions. We then present a new 1D$\times$TIME meshing algorithm for handling the NUCC problem. In the UCC case the 1D$\times$TIME meshing problem is fairly easy to solve. Unfortunately, in the NUCC case even for domains with simple geometry this problem is no longer trivial. Here we discuss the difficulties that arise when NUCC mesh is required and provide a solution that finds application in solving Euler equations. We verify the correctness of the 1D$\times$TIME NUCC meshing algorithm. Our experiments also verify that the resulting mesh closely follows the cone constraint. This is crucial for achieving accurate analysis results [2].

NUCC Meshing Problem Definition

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 $\alpha()$.*

In order to guarantee the algorithm validity, we assume that the mesh faces are never required to be horizontal, i.e. $\alpha() \geq \alpha_0 > 0$, nor vertical, i.e. $\alpha() \leq \alpha_1 < \frac{\pi}{2}$.

Definition 2 (NUCC Meshing Problem) *Given a space-time domain D, a mesh of the initial space sub-domain and a cone constraint function $\alpha()$, build a mesh of D that satisfies $\alpha()$.*

NUCC Extension of the UCC Meshing Algorithms

One approach for extending the UCC meshing algorithms mentioned above to handle the non-uniform cone constraint is using the minimum of $\alpha() \in D$ as a uniform cone constraint. For layer based methods such as [3, 5] the most limiting $\alpha()$ within each layer can be used to establish the time slab height. The limit on the height of the slab is determined by a combination of $\alpha()$ and the mesh element size. However, even using this approach many of the mesh faces can be significantly below the cone constraint function. Therefore, to generate NUCC meshes that closely follow the cone constraint function, an element-by-element construction technique has to be adopted. The next section describes such an algorithm for the 1D$\times$TIME domain.

New Algorithm for 1D$\times$TIME NUCC Meshing

The algorithm maintains an advancing front which is initialized by the input space domain mesh. At each step it adds a group of elements to the mesh updating the front. The algorithm terminates when the entire domain is covered. Below we introduce the necessary definitions, describe the advancing front and the element generation procedures.

Definitions

For 1D$\times$TIME meshing, the *front* is a sequence of edges (nodes) with only one element attached to each. In this domain, a cone associated with a point $x \in D$ is uniquely defined by $\alpha(x, l)$ and $\alpha(x, r)$ which are the left and the right sides of the cone respectively (Figure 1 (a)). A *left (right) cone constraint ray* at a front node p starts from p and makes an angle of $\alpha(p, l)$ ($\alpha(p, r)$) with the space axis. A left (right) cone constraint of a node is *handled* if an edge based on it has already been added to the mesh, and is *unhandled* otherwise. Figure 2 illustrates the terminology used.

We define $h(p, q)$ as the *horizontal distance* between two front points p and q, i.e. the distance between the lines that go through p and q parallel to the time axis.

Definition 3 (Matching & Closest Matching) *A pair of nodes (p, q) (p is left to q) on the front is* matching *if both the right cone constraint of p and the left cone constraint of q are unhandled. A pair of matching nodes which do not have a matching pair in between form a* closest matching.

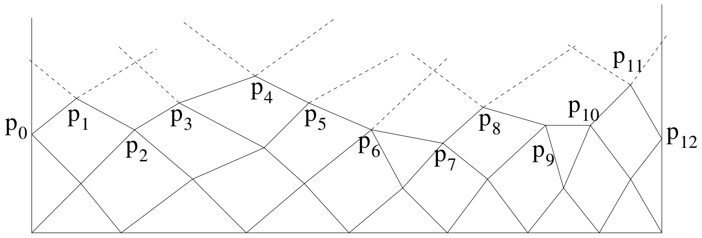


Figure 2: The nodes on the front might have both cone constraints (e.g. p_2), only the left one (e.g. p_5), only the right one (e.g. p_3), or neither of them (e.g. p_4) handled. (p_5, p_8) is a matching pair but not closest, whereas (p_6, p_8) is a closest matching pair.

Advancing Front Procedure

The advancing front algorithm manages the interaction between the numerical simulation code and the mesh generation algorithm (Figure 3). It starts by meshing the space domain. The advancing front is initialized to this space mesh. In the advancing step, the mesh generator constructs (a) 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. The ray-intersection algorithm described below provides the advancing step of this main algorithm.

```
ALGORITHM Advancing Front
  construct the space mesh at the initial time
  let the front be this initial mesh
  while the 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 3: Interface routine between the simulation and mesh generation.

Ray-Intersection Algorithm

At each step of the advancing front, the ray-intersection algorithm chooses an arbitrary closest matching pair of nodes, p and q. The matching cone constraints of the two nodes are handled based on the intersection point of the two cone constraint rays at p and q (Figure 4).

If pq is below both rays, then they intersect at a point r. In this case the

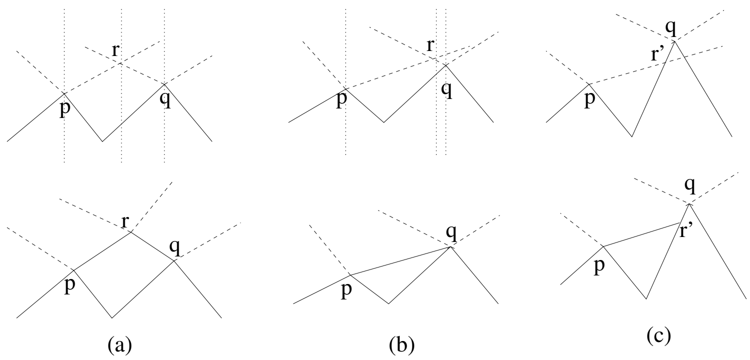


Figure 4: Handling closest matching pairs in the ray-intersection algorithm.

horizontal distances between r and each of p and q are considered. In order to guarantee termination with the desired output, the algorithm keeps at least a distance $T > 0$ between any two adjacent front nodes. If both $h(p, r)$ and $h(q, r)$ are greater than or equal to T, then the edges pr and qr are built and the region enclosed by them is added to the mesh (Figure 4 (a)). In this case both cone constraints are tagged as handled. The new node r is inserted into the front with its left and right cone constraints tagged as unhandled. If $h(p, r)$ or $h(q, r)$ is less than T, then the edge pq is built and the cone constraint of the lower node in the time axis is tagged as handled (Figure 4 (b)). Specifically, the right cone constraint of p is tagged as handled if p is below q and the left cone constraint of q is tagged as handled, otherwise.

ALGORITHM Ray-Intersection

while the domain D is not covered
  find a closest matching pair on the front
  compute the intersection of the cone constraint rays
  handle the two node cone constraints as shown in Figure 4

Figure 5: Intersection based meshing algorithm.

There can be a case when the intersection does not exist (Figure 4 (c)). This happens only if the ray from the lower node, e.g. p, passes below q. In this case, there exists a front edge e which the ray intersects first at a point r'. In this case an edge pr' is added to the mesh, and the right cone constraint of p is tagged as handled (Figure 4 (c)). The new node r' is not considered a front node.

The regions generated by the method can, then, be divided into quadrilateral or triangular elements. This is achieved by drawing diagonals if those diagonals satisfy the cone constraint or by trapezoidal decomposition with horizontal edges.

Boundary Treatment A node on the left or right boundary of the domain, has only a right or left cone constraint respectively, which is treated similarly to the constraints at the other front nodes. When a node on the front has an unhandled left (right) cone constraint, and no other node to its left (right) has an unhandled right (left) cone constraint, the left (right) boundary itself is viewed as an unhandled right (left) constraint ray starting from the front boundary node (Figure 7).

Algorithm Verification

This section proves that the presented algorithm solves the NUCC meshing problem. First, we prove that the mesh edges satisfy the cone constraint. Then, the generated mesh is shown to be valid, i.e. the mesh edges intersect only at common nodes. We also show that the algorithm can always proceed, i.e. a closest matching pair of nodes always exists. Finally, the algorithm is shown to terminate discretizing the entire domain.

Lemma 1 (Cone Constraint) *The algorithm generates a mesh that satisfies the given cone constraint function* $\alpha()$.

PROOF. By construction region boundary edges satisfy the cone constraint. The edges subdividing each region are either diagonals or horizontal lines. Diagonals are used only when they satisfy the cone constraint and horizontal lines satisfy it by default. □

Observation 1 (Regular Node Property) *A left (right) cone constraint at a front node is tagged as handled only when its left (right) adjacent front node is above it (or at the same height) in the time axis.*

If a left (right) cone constraint at a front node is tagged as handled and its left (right) adjacent front node is at the same height then the right (left) cone constraint at this adjacent node is tagged as unhandled (it can happen only in the case shown in Figure 4(b)).

Lemma 2 (Legality) *The algorithm generates a valid mesh, i.e. the mesh edges intersect only at common nodes.*

PROOF. We prove that the newly generated edges do not intersect existing edges. It is sufficient to show that for a closest matching pair (p, q) (p to the left of q), pq does not intersect another front edge. Without loss of generality let p be not above q. For intersection to occur a front node between p and q has to be above pq. Let s be the highest-left node above pq (note that s has to be above p). Since s is above its adjacent left node, based on Observation 1, it has an unhandled left cone constraint, contradictory to p and q being a closest matching. □

Lemma 3 (Advance) *The algorithm can always proceed.*

Proof. It is sufficient to show that a closest matching pair of nodes always exists which is equivalent to showing that a matching pair exists. The highest node in the mesh is a front node by construction. Hence, it has either one or both cone constraints unhandled. As described in the boundary handling section, it can be matched with the right (left) boundary node. By Observation 1, in the degenerate case where several highest nodes exist, at least one has an unhandled right or left cone constraint. $\square$

Lemma 4 (Termination) *The algorithm terminates, filling the domain.*

Proof. We give a lower bound on the the size of the region created when a new node r is placed at an intersection of two cone constraint rays (Figure 4(a)). The two other cases reduce the number of cone constraints on the front and hence can occur only a finite number of times. Without loss of generality, let p be not above q in the time axis. Let c be the intersection of the horizontal line that passes through q and vertical line that passes through r (Figure 6). The length of the line segment qc, $|qc|$, is at least T, and $\alpha() > \alpha_0$, hence $|rc| \geq \tan(\alpha_0)T$. Both triangles $\triangle qrc$ and $\triangle prc$ have a heigth (with respect to rc) of at least T implying each has area at least $\frac{1}{2}\tan(\alpha_0)T^2$. Since p is below both q and c in the time axis, $A(pqr) \geq A(qrc) + A(prc) \geq \tan(\alpha_0)T^2$. $\square$

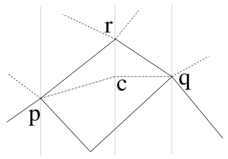


Figure 6: Size of an element.

The main result of this paper follows from the Lemmas above.

Theorem 1 *The algorithm generates a mesh M of a given domain D, such that the faces of M satisfy a given cone constraint function $\alpha()$.*

Results

Figure 7 illustrates the application of the ray-intersection algorithm with various cone constraint functions $\alpha()$. The implementation is capable of generating both mixed (quad dominant) meshes (Figure 7 (a)-(c)) and pure triangular meshes (Figure 7 (d)).

Conclusions

We have proposed a provably correct algorithm to generate 1D×TIME meshes that satisfy a non-uniform cone constraint. These meshes are critical for efficient, element-by-element DG solutions of problems with nonlinear conservation laws, such as the inviscid Bergers' equation, and of nonlinear second-order problems, such as elastodynamics in nonlinear materials. To our knowledge, the generation of such meshes has never been studied before.

There is no obvious extension of the ray-intersection method to 2D×TIME as intersection of planes in 3D becomes complex. An alternative algorithm is considered in [6] which extends to higher dimensions by relaxing the NUCC meshing problem requirements.

References

[1] 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.

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

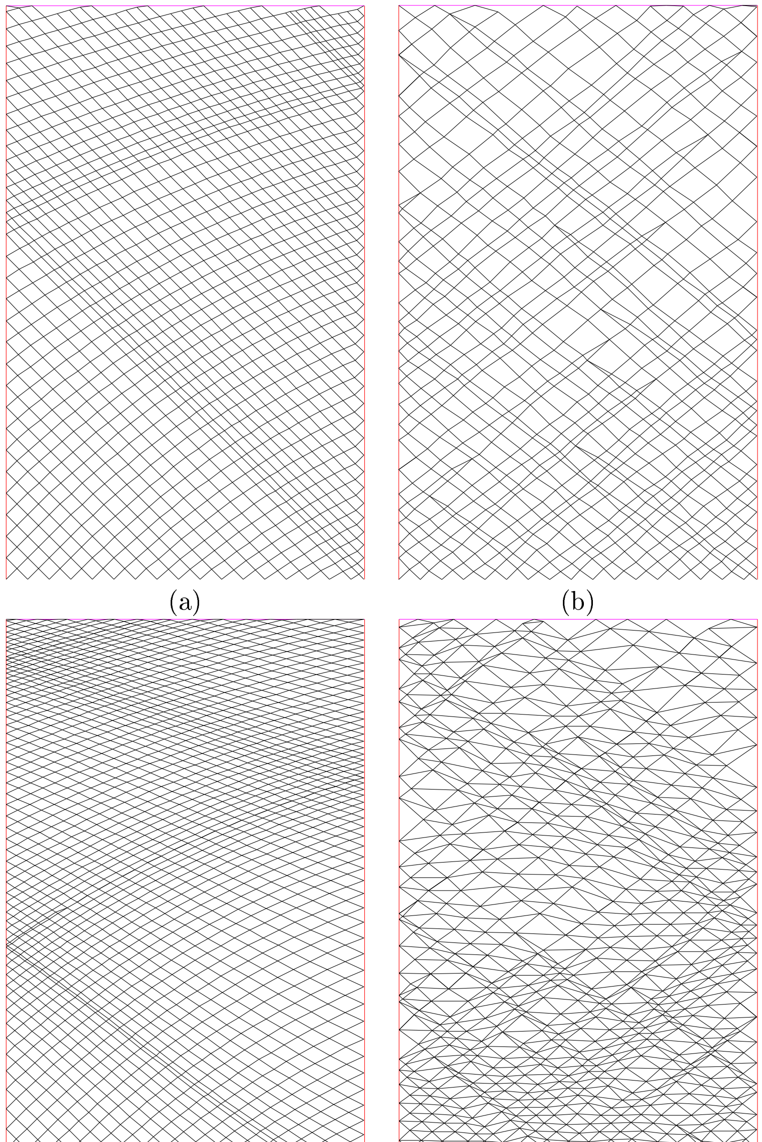
[3] A. Sheffer, A.Üngör, S.-H. Teng, and R. B. Haber. *Generation of 2D Space-Time Meshes Obeying the Cone Constraint,* Proc. of Int. Conference on Computational Eng. Sci., Los Angeles, CA, August 2000.

[4] 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.

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

[6] A.Üngör, and A. Sheffer. *Tent-Pitcher: A Meshing Algorithm for Space-Time Discontinuous Galerkin Methods,* submitted to the 9th Int. Meshing Roundtable New Orleans, LO, October 2000.

[7] L. Yin, A. Acharya, N. Sobh, R.B. Haber and D.A. Tortorelli. *A Space-Time Discontinuous Galerkin Method for Elastodynamic Analysis,* Proc. of Int. Symp. on Discontinuous Galerkin Methods, May 1999.



(a) (b) (c) (d)

Figure 7: **Meshes of a rectangular domain with various cone constraints ((a)-(c) quad dominant, (d) triangular): (a) uniform $\alpha()$; (b) random $\alpha()$; (c) quadratic $\alpha()$; (d) random $\alpha()$.**