Spacetime Meshing with Adaptive Refinement and Coarsening*

Reza Abedi[†] Shuo-Heng Chung[‡] Jeff Erickson[‡§] Yong Fan[†] Michael Garland[‡]
Damrong Guoy[¶] Robert Haber[†] John M. Sullivan[‖] Shripad Thite[‡] Yuan Zhou[‡]

Center for Process Simulation and Design
University of Illinois at Urbana-Champaign
{rabedi,schung6,jeffe,yongfan,garland,guoy,r-haber,jms,thite,yuanzhou}@uiuc.edu

ABSTRACT

We propose a new algorithm for constructing finite-element meshes suitable for spacetime discontinuous Galerkin solutions of linear hyperbolic PDEs. Given a triangular mesh of some planar domain Ω and a target time value T, our method constructs a tetrahedral mesh of the spacetime domain $\Omega \times [0, T]$ in constant running time per tetrahedron in $\mathbb{R}^3$ using an advancing front method. Elements are added to the evolving mesh in small patches by moving a vertex of the front forward in time. Spacetime discontinuous Galerkin methods allow the numerical solution within each patch to be computed as soon as the patch is created. Our algorithm employs new mechanisms for adaptively coarsening and refining the front in response to *a posteriori* error estimates returned by the numerical code. A change in the front induces a corresponding refinement or coarsening of future elements in the spacetime mesh. Our algorithm adapts the duration of each element to the local quality, feature size, and degree of refinement of the underlying space mesh. We directly exploit the ability of discontinuous Galerkin methods to accommodate discontinuities in the solution fields across element boundaries.

*See http://www.cs.uiuc.edu/~jeffe/pubs/refine.html for the most recent version of this paper. Work on this paper was partially supported by NSF ITR grant DMR-0121695.

[†]Department of Theoretical and Applied Mechanics, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (UIUC).

[‡]Department of Computer Science, UIUC.

[§]Also partially supported by NSF CAREER award CCR-0093348 and NSF ITR grant CCR-0219594.

[¶]Center for Simulation of Advanced Rockets (CSAR), Computational Science and Engineering Program, UIUC. The CSAR research program is supported by the US Department of Energy through the University of California under subcontract B523819.

[‖]Department of Mathematics, UIUC, and Department of Mathematics, Technische Universität Berlin. Also partially supported by NSF grant DMS-00-71520.

SCG'04, June 8–11, 2004, Brooklyn, New York, USA.

Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational geometry and object modeling—*Geometric algorithms, languages, and systems*; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical algorithms and problems—*Geometric problems and computations*; G.1.8 [Numerical Analysis]: Partial differential equations—*Hyperbolic equations; finite element methods*

Keywords: mesh generation, unstructured meshes, tetrahedral meshes, spacetime discontinuous Galerkin, advancing front, adaptivity

General Terms: Algorithms, Performance

1. INTRODUCTION

Scientists and engineers use conservation laws and hyperbolic partial differential equations to model transient, wave-like behavior in bodies or spatial control volumes. Example applications are numerous, ranging from the Euler equations for compressible gas dynamics, to the Schrödinger equation for time-dependent density functional theory in quantum mechanics, to the equations of elastodynamics in seismic analysis. Closed-form solutions are typically unavailable for these problems, so analysts usually resort to numerical approximations. However, the continuum solutions can exhibit strongly nonlinear behavior as well as shocks and other discontinuities that make this class of numerical problems particularly challenging.

Finite element methods are a good option for solving these problems, especially when the geometry of the analysis domain is complicated. In the standard *semi-discrete* approach, a finite element mesh discretizes space to generate a system of ordinary differential equations in time that is then solved by a time-marching integration scheme. Most methods enforce a uniform time step size over the entire spatial domain. This approach can be very costly for strongly graded grids, because the allowable time step size is limited by the global minimum element diameter. However, physical causality only implies a local limit on the step size, so algorithms that use a nonuniform time step size can substantially improve computational efficiency. In addition, reductions in computational complexity and improvements in parallel structure can be realized by incorporating physical causality into the meshing algorithm.

Dynamic adaptive refinement and coarsening is essential for capturing solution features that move with traveling interfaces and shock fronts. Each discrete remeshing operation requires a projection of the solution fields from the old grid onto the newly adapted mesh. These projections can be costly and inconvenient to compute, and they introduce significant error. A more continuous approach to adaptive refinement could eliminate these data projections and the error they produce.

In summary, the following are desirable features for an effective adaptive meshing scheme for hyperbolic problems:

1. local coordination of time step sizes and element diameters;
2. dynamic refinement and coarsening, with refinement restricted to the spacetime trajectories of moving interfaces and shock fronts;
3. continuous mesh adaptation to eliminate projection operations associated with discrete remeshing; and
4. incorporation of causality conditions to support efficient serial and parallel solution algorithms.

Spacetime discontinuous Galerkin methods [31, 6, 15, 36, 35, 21] are a relatively new class of finite element methods that comprise an interesting alternative to semi-discrete methods. (For further background on discontinuous Galerkin methods in general, we refer the reader to Cockburn, Karniadakis, and Shu [7].) Two features distinguish spacetime discontinuous Galerkin methods from conventional finite element models. First, spacetime discontinuous Galerkin methods work with meshes that cover the entire spacetime analysis domain. For example, simulation of evolving behavior on a three-dimensional spatial domain requires a four-dimensional spacetime mesh. The spacetime discontinuous Galerkin algorithm weakly enforces the governing equations over each spacetime element, and this eliminates the need for a separate time integration procedure. The time duration of a spacetime element is the local analogue of the global time step size in semi-discrete methods. The second distinguishing feature is the use of discontinuous basis functions with support on individual elements for representing the physical fields. In contrast to conventional finite element methods that use continuous bases, this approach eliminates artificial coupling between adjacent elements when the mesh satisfies certain causality constraints. It also guarantees exact satisfaction of the relevant balance laws on every spacetime element.

In principle, all of the desirable features listed above can be realized by implementing discontinuous Galerkin models on unstructured spacetime grids. For example, an unstructured spacetime mesh can coordinate the time duration and the spatial diameter of individual elements, thereby eliminating the inefficiencies induced by enforcing a global time step. The refinement can vary continuously in spacetime to track the trajectories of moving solution features and to circumvent discrete remeshing operations. Finally, the combination of a causality-conforming spacetime mesh with discontinuous basis functions supports new solution algorithms with linear complexity in the number of elements and with rich parallel structure. In practice, the need to simultaneously satisfy all of these criteria poses a number of novel problems in numerical mesh generation, some of which are the subject of this paper.

Building on ideas from earlier specialized algorithms [15, 28, 32, 34], Üngör and Sheffer [33] and Erickson *et al.* [8] developed the first algorithm to build graded spacetime meshes over arbitrary simplicially meshed spatial domains, called 'Tent Pitcher'. Unlike most traditional approaches, the Tent Pitcher algorithm does not impose a fixed global time step on the mesh, or even a local time step on small regions of the mesh. Rather, it produces a fully unstructured simplicial spacetime mesh, where the duration of each spacetime element depends on the local feature size and quality of the underlying space mesh. See Figure 1.

Given a triangular mesh of some planar domain Ω, and a target time value T, the Tent Pitcher algorithm meshes the spacetime domain $\Omega \times [0, T]$ in $\mathbb{R}^3$ using an advancing front method. Elements are added to the evolving mesh in small patches by moving a vertex of the front forward in time. The advance in time is limited by causality, to ensure that the solution in the new patch depends only on boundary data and solution data from previously computed patches. The use of discontinuous basis functions avoids artificial coupling with subsequent patches. Thus, the spacetime discontinuous Galerkin solution can be computed locally within each new patch as soon as it is created. Provided the initial ground mesh has constant degree, each patch contains only a constant number of elements and can therefore be solved in constant time. The time to generate the spacetime mesh and compute the numerical solution is, therefore, linear in the number of spacetime elements. Moreover, because patches with no causal relationship can be solved independently, the Tent Pitcher algorithm is uniquely well-suited for parallel implementation. In Section 2, we give a new and more intuitive formulation of the Tent Pitcher algorithm and review its theoretical guarantees.

In this paper, we introduce new mechanisms into the Tent Pitcher algorithm that adaptively refine or coarsen the advancing front in response to *a posteriori* error estimates computed by the numerical code. The front at any stage of the meshing and solution process is a hierarchical subdivision of the original space mesh. A change in the front induces a corresponding refinement or coarsening of future elements in the spacetime mesh. We refine the front using the classical newest vertex bisection method of Sewell [27] and Mitchell [18, 19, 20]; we coarsen the front only by reversing earlier refinements. These modifications generate non-conforming spacetime meshes; two adjacent spacetime elements may not share a common face. However, because discontinuous Galerkin methods directly accommodate discontinuities in the solution field across element boundaries, they do not require conforming meshes or a data projection operation. We describe the remeshing operations in detail in Section 3.

The original Tent Pitcher algorithm of Üngör and Sheffer [33] required every angle in the input mesh to be acute; if the ground mesh contains an obtuse angle, their algorithm can get stuck. Erickson *et al.* [8] remove this limitation by imposing an additional constraint, beyond those due to causality, on the maximum height of any tent. This so-called *progress constraint*, which depends on the shape of the underlying space elements, guarantees a lower bound on the *minimum* height of the tent. Adapting this progress constraint to support the changing geometry of the front is the main technical contribution of this paper; see Section 4.

2. SPACETIME MESHING

The formulation of our spacetime meshing problem relies on the notions of domain of influence and domain of depen-

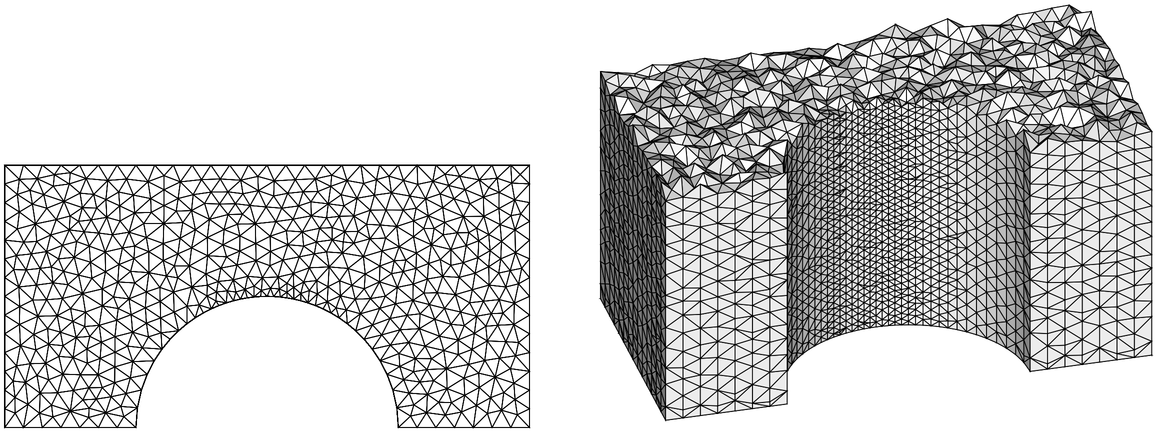


Figure 1. An input space mesh and the resulting spacetime mesh computed by Tent Pitcher [8]

dence for hyperbolic boundary value problems. We say that a point $\hat{p}$ in spacetime *depends on* another point $\hat{q}$ if the values of the salient physical fields (e.g., temperature, velocity, pressure, stress, momentum) at $\hat{p}$ depend on the field values at $\hat{q}$; that is, if changing the conditions at $\hat{q}$ can affect the conditions at $\hat{p}$. The *domain of influence* of $\hat{p}$ is the set of points that depend on $\hat{p}$; symmetrically, the *domain of dependence* is the set of points that $\hat{p}$ depends on.

If the governing equations of the underlying problem are linear and the material properties are homogeneous and isotropic, the domains of influence and dependence are circular cones with a common apex $\hat{p}$. This double cone can be described by a scalar *wave speed* $c(\hat{p}) \in \mathbb{R}$, which specifies how quickly the radius of the cones grows as a function of time. In this paper, we consider only linear problems, where the wave speed is uniform and isotropic over the entire spacetime domain. In this case, we can choose an appropriate time scale, independent of the spatial scale, so that $c(\hat{p}) = 1$ everywhere. For more general problems, the wave speed can vary with direction and as a function of inhomogeneous physical parameters, and might even depend nonlinearly on the solution.

We say that one spacetime element $\triangle$ depends on another spacetime element $\triangle'$ if any point $\hat{p} \in \triangle$ depends on any point $\hat{q} \in \triangle'$. The solution can be computed in each element one at a time, following any linear extension of the dependency partial order. Alternatively, the solutions within any anti-chain of elements can be computed in parallel.

Although discontinuous Galerkin methods impose no *a priori* restrictions on the primitive shapes of individual elements, it is convenient to work with simplicial elements. We say that a facet F of a tetrahedral element is *causal* if it separates the cone of influence from the cone of dependence at every point on F (Figure 3), or equivalently, if $\|\nabla F\| \leq 1$. If a facet is causal, physical information flows in only one direction across that facet.To construct an efficient tetrahedral mesh, we group elements into *patches*, each with a constant number of elements. The Tent Pitcher algorithm ensures that the boundary facets between patches are *causal* by construction; we refer to this requirement as the *causality constraint* (called the *cone constraint* in Erickson *et al.* [8]).

In general, the internal facets between elements within a single patch are not causal. Since information can flow in both directions across these facets, the physical response in adjacent elements is coupled, and all of the elements within a patch must therefore be solved simultaneously. However, if we assume polynomial basis functions of bounded degree, the resulting linear system still has bounded size and can therefore be solved in constant time. Thus, the total time to compute the numerical solution is still linear in the number of elements.

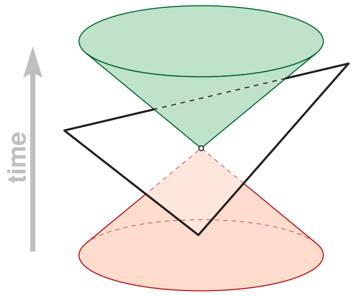


Figure 3. The causality constraint: The facet separates the cone of influence (above) from the cone of dependence (below).

Richter [22] observed that undesirable numerical dissipation increases in certain discontinuous Galerkin methods as the slopes of boundary facets decrease below the local wave speed. Thus, our goal is to construct an efficient mesh comprised of patches, each containing a small number of simplices, such that the boundary facets of each patch are as close as possible to the cone constraint without violating it.

2.1 Advancing-Front Spacetime Meshing

Given a triangulated planar domain Ω and a target time value T, Tent Pitcher constructs a tetrahedral mesh of the spacetime domain $\Omega \times [0, T]$. The algorithm is designed as an advancing front procedure, which alternately constructs a new patch of elements and invokes a spacetime discontinuous Galerkin finite-element method to compute the solution within that patch.

At every iteration, the *front* is the graph of a continuous time function $t : \Omega \to \mathbb{R}$, such that within every triangle, t is linear and $\|\nabla t\| \leq 1$, where ∇t denotes the gradient vector of the time function t. We will assume that the initial time function is constant but more general initial conditions are also permitted. The front is a terrain whose facets corre-

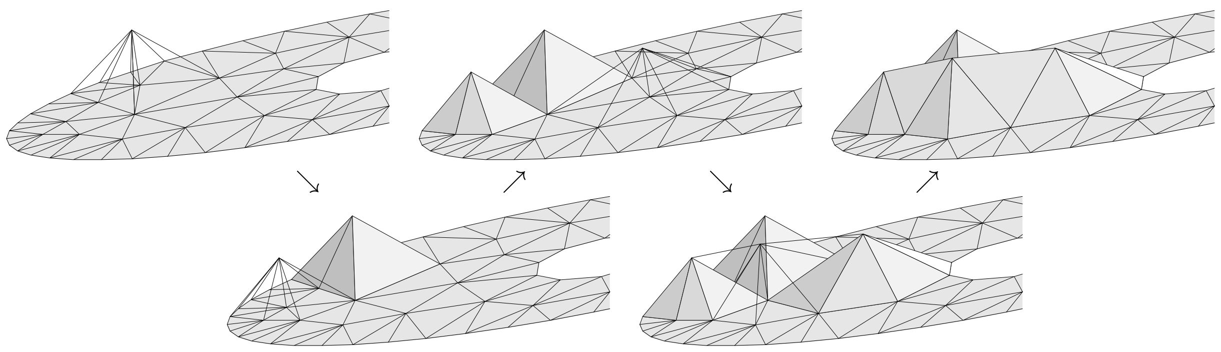


Figure 2. Pitching tents in spacetime.

spond to triangles in the underlying space mesh. To advance the front, the algorithm chooses an arbitrary local minimum vertex $\hat{p} = (p, t(p))$ from the front and moves it forward to a new point $\hat{p}' = (p, t'(p))$ where $t'(p) > t(p)$. The volume between the new and old fronts is called a *tent*. We decompose the tent into simplices sharing the edge $\hat{p}\hat{p}'$ and pass these elements, along with the inflow elements just below the tent, to a DG solver. When the algorithm has several choices for the vertex to advance next, we apply one of several heuristics; some heuristics appear to work better in practice than others. The algorithm is free to advance a vertex that is not a local minimum; however, in that case, a finite amount of progress cannot always be guaranteed.

2.2 Causality and Progress Constraints

To complete the description of Tent Pitcher, all that remains is to describe how to compute the new time values for each vertex to be pitched, or less formally, how high to pitch each tent. Each triangle in the ground mesh imposes constraints on the time values at each of its vertices. When we pitch a tent over a local minimum vertex $\hat{p}$, the new time value $t'(p)$ is simply the largest value that satisfies the constraints for every triangle adjacent to p in the ground mesh.

Thus, it suffices to consider the case where the ground mesh consists of a single triangle $\triangle pqr$. At any stage of the algorithm, the front consists of a single triangle $\triangle\hat{p}\hat{q}\hat{r}$ whose vertices have time values $t(p)$, $t(q)$, and $t(r)$. Suppose without loss of generality that $t(p) \le t(q) \le t(r)$ and we want to advance $\hat{p}$ forward in time. We must choose the new time value to satisfy the causality constraint $\|\nabla t\| \le 1$. Erickson *et al.* [8] show that this constraint can be expressed by the following inequality:

$$\boxed{\begin{aligned} t'(p) \le t(q) + &\frac{t(r)-t(q)}{|qr|^2}\,(\vec{qp}\cdot\vec{qr}) \\ &+ \sqrt{1-\left(\frac{t(r)-t(q)}{|qr|}\right)^2}\,d(p,qr) \end{aligned}} \tag{1}$$

Here, $d(p, qr)$ denotes the distance from p to the line qr.

Üngör and Sheffer [33] proved that if every angle in $\triangle pqr$ is acute, then exactly meeting the causality constraint guarantees that further advancement is always possible. However, if $\angle pqr$ is obtuse, then lifting $\hat{p}$ as far as possible may prevent us from lifting $\hat{q}$ in the next iteration without violating causality. To avoid this problem, Erickson *et al.* [8] impose an additional *progress constraint*, which limits the gradient of the time function restricted to the edges touching the obtuse angle.

The progress constraint can be visualized easily using a *circle diagram*; see Figure 5. Think of the gradient vector ∇t as a point in the plane; the causality constraint restricts this point to the unit disk centered at the origin. We can partition this disk into three wedges by the inward normal vectors of the edges of $\triangle pqr$. Each wedge is associated with a vertex of the triangle; for example, the wedge for p is bounded by vectors normal to the edges pq and pr. The gradient vector ∇t lies inside p's wedge if and only if p has the smallest time value, and pitching p moves the point ∇t along the inward normal vector to the edge qr, or in other words, parallel to the vector that does not bound the current wedge.

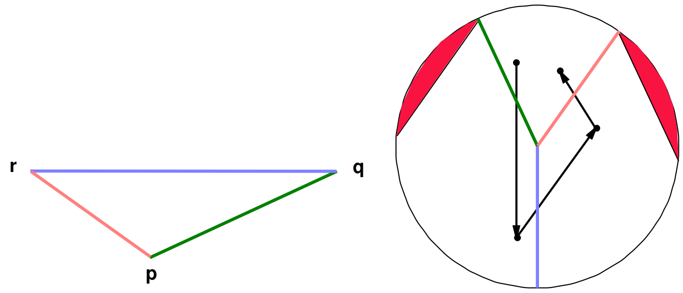


Figure 5. A triangle and its circle diagram. To guarantee causality, the gradient vector ∇t must stay inside the circle; to guarantee progress, ∇t must stay out of the shaded forbidden zones.

To guarantee that the algorithm never gets stuck, we must ensure that we can always move the point ∇t out of its current wedge; otherwise, the lowest vertex will never progress past its neighbor. If $\triangle pqr$ has an obtuse angle, this requirement creates "forbidden zones" inside the two obtuse wedges, bounded by lines normal to the edges adjacent to the obtuse angle. The progress constraint is simply to stay out of these forbidden zones. We can express this constraint algebraically as the inequality $t'(p) \le t(r) + (1-\varepsilon)d(p, qr)$, where $0 < \varepsilon \le 1/2$ is a tunable parameter.

3. HIERARCHICAL MESH REFINEMENT

Our adaptive algorithm uses the *newest-vertex bisection* refinement method, originally developed by Sewell [27], later adapted by Mitchell [18, 19, 20] in the context of multigrid methods, and still later studied and generalized to three

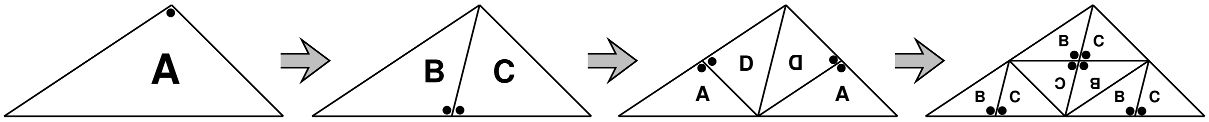


Figure 4. Newest vertex refinement.

dimensions by Bänsch [3]. This method is similar to the more common *longest-edge* refinement methods introduced by Rosenberg and and Stenger [26] and popularized by Rivara [23, 24, 25] and others [12, 1, 29]. For a general discussion of mesh refinement methods, we refer the the reader to surveys by Jones and Plassmann [9] and by Bern and Plassmann [4].

We call the newest vertex of a triangle its *apex* and the opposite edge the *base* of the triangle. Initially, one vertex of each triangle in the mesh is chosen arbitrarily as its apex. Newest-vertex bisection replaces a triangle with two smaller triangles, each with half the area of the original triangle, obtained by bisecting along the line segment through the apex and the midpoint of the base. The new vertex introduced at the midpoint is the newest vertex of both smaller triangles.

The descendants of any marked triangle under newest-vertex bisection fall into only eight homothetic classes—see Figure 4—and there are only four directions along which the triangle or any of its descendants can be further bisected. These four directions are parallel to the three edges of the triangle and the bisecting segment. Any two triangles in the same equivalence class have corresponding newest vertices.

When we refine one triangle in a mesh, we may be forced to refine other nearby triangles in order to maintain a conforming triangulation. Suppose vertex a is the apex of a $\triangle abc$ that is bisected (Figure 6). If bc is not a boundary edge then some neighboring triangle $\triangle cbe$ shares the edge bc. To maintain a triangulation, $\triangle cbe$ must be bisected also. If bc is not the base of $\triangle cbe$ then the child of $\triangle cbe$ sharing edge bc must be bisected as well, and the bisection of $\triangle cbe$ will propagate recursively; see Figure 6. It is easy to prove that this propagation terminates, regardless of which vertex is chosen as the apex of each triangle. The propagation path touches every triangle in the worst case, but in practice, the propagation path usually has small constant length; see Suárez *et al.* [30] for an analysis of a similar refinement algorithm.

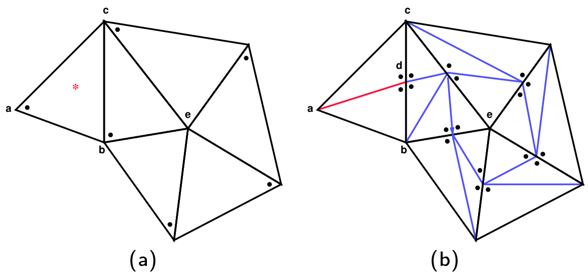


Figure 6. Refinement of one triangle propagating to neighboring triangles: (a) before and (b) after refinement.

The entire refinement sequence can be expressed as a sequence of edge bisection and quad flip operations. The former operation bisects an edge shared by one or two triangles, and the latter replaces one diagonal of a convex quadrilateral with the other diagonal.

Coarsening is the opposite of refinement; we coarsen locally by undoing a single edge bisection. Unlike refinement, coarsening does not require propagation further into the mesh to maintain a conforming triangulation, although one coarsening step may make other coarsenings possible. In particular, if we refine a triangle and then immediately coarsen, we can (but need not) coarsen along the entire refinement path.

Newest-vertex bisection never subdivides any angle of a triangle more than once. Therefore, the degree of any vertex in the mesh at most doubles as a result of refinement. Moreover, given a triangulation with maximum degree d with vertices in general position, if the vertex with largest angle in each triangle is chosen as the apex, then any refinement of the initial triangulation has maximum degree at most $\max\{d+5, 8\}$.

4. MAKING TENT PITCHER ADAPTIVE

To incorporate adaptivity into our meshing algorithm, we make a small change to the main loop. At each iteration, just as before, we choose a local minimum vertex $\hat{p}$ of the front, move it forward in time to create a tent, and pass the tent and its inflow data to the spacetime DG solver. We assume that the solver also computes an *a posteriori* estimate of its own numerical error. If the error within any element of the tent is above some threshold, the solver *rejects* the patch, at which point the meshing algorithm throws away the tent and refines the facets of the front whose elements had high error. (Alternately, we could refine the facets until their diameter is smaller than a target length scale computed by the solver.) Note that this refinement may propagate far outside the neighborhood of $\hat{p}$. We accept the numerical solution and update the front only if the error within every element of the patch is acceptable.

On the other hand, the error estimate within an element may fall below some second threshold, indicating that the mesh is finer than necessary to compute the desired result. In this case, the DG solver marks the outflow face of that element as *coarsenable*. We can coarsen four facets of the front into two only if they are the result of an earlier refinement, they are all marked as coarsenable, and each pair of triangles to be merged is coplanar. To make coarsening possible, our algorithm tries to make coarsenable siblings coplanar, by lowering the top of the tent. However, to avoid very thin elements, we only accept the lower tent if its height is above some threshold. If the lower tent is accepted and its outflow faces are still marked coarsenable, then we coarsen the front.

4.1 New Progress Constraints

The more subtle and important change in our algorithm is the introduction of new progress constraints. In the earlier non-adaptive algorithm [8], the progress constraint was a function of the shape of the underlying space elements. In

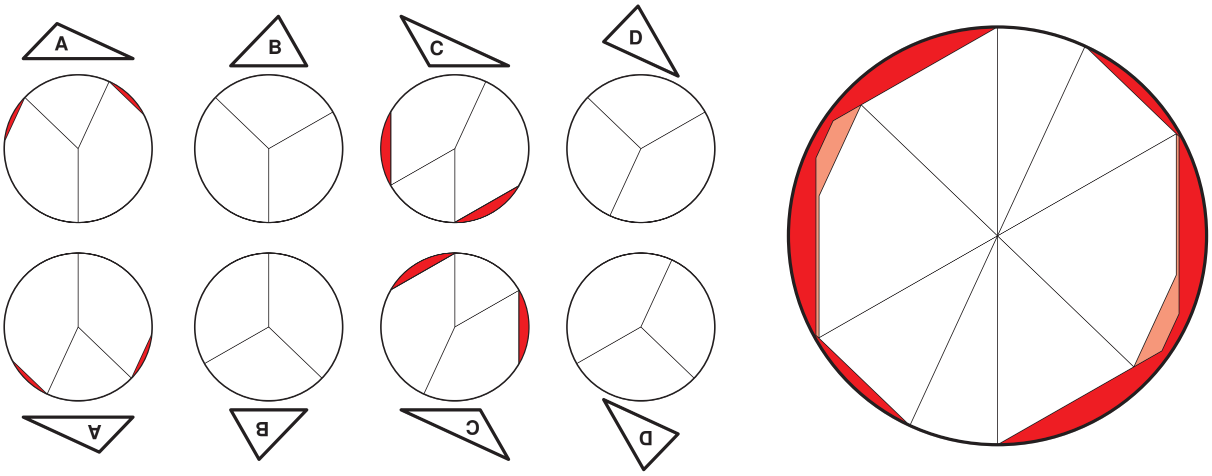


Figure 7. Merging eight progress constraints into one.

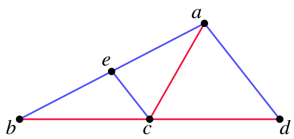
our new algorithm, the shape of the underlying element is subject to change; each triangle in the current front may be the result of any number of coarsenings or refinements since the last time any ancestor or descendant of that triangle was last pitched. Consequently, our progress constraints must take into account the shapes of *all possible* descendants of a triangle simultaneously. This requirement motivates our choice of newest-vertex refinement; because the descendants of any triangle fall into only a finite number of homothety classes, we have only a finite number of simultaneous progress constraints.

We can visualize these constraints by referring back to our circle diagram; see Figure 7. A minimal set of progress constraints can be obtained by simply taking the union of the non-adaptive progress constraints of the eight shapes in the hierarchy. These constraints are not necessarily sufficient, however, since it may be impossible for the gradient to leave a wedge for one triangle without violating a progress constraint for a different triangle. Fortunately, we can avoid these conflicts by strengthening some of the constraints.

Omitting the geometric derivation, we formally describe the resulting constraints as follows. Fix two real numbers $0 < \delta, \varepsilon < 1$. For any triangle $\triangle abd$ with apex a, we define the *diminished width* of $\triangle abd$ as follows:

$$w_\varepsilon(\dot{a}bd) := \min \left\{ \begin{array}{l} (1-\varepsilon)\, d(a, bd), \\ (1-\delta)\, d(b, ad), \\ (1-\delta)\, d(d, ab) \end{array} \right\}$$

The first distance is measured from the apex to the opposite edge and is scaled differently from the other two altitudes. This definition extends recursively to any descendants of $\triangle abd$ obtained by newest-vertex subdivision; in the interest of readability, we will always list the vertices of any triangle with the apex first. Now let c be the midpoint of bd and let e be the midpoint of ab, as illustrated below. We express our new progress constraints algebraically by limiting the difference in time values along each edge in the subdivided triangle, as follows.



$$\boxed{\begin{array}{l} |t(a)-t(b)| \le 2w_\varepsilon(\dot{c}ad) \\ |t(a)-t(c)| \le w_\varepsilon(\dot{a}bd) \\ |t(a)-t(d)| \le 2w_\varepsilon(\dot{c}ab) \\ |t(b)-t(d)| \le 4w_\varepsilon(\dot{e}ca) \end{array}}$$

These constraints apply recursively to all descendants of the triangle $\triangle abd$—for example, we also enforce the constraint $|t(c) - t(e)| \le w_\varepsilon(\dot{c}ab)$—but these recursive constraints are redundant, since they limit the gradient of the time function in exactly the same direction as one of the four constraints above.

Now suppose we are pitching a triangle $\triangle pqr$, where $t(p) \le t(q) \le t(r)$. Let s be the midpoint of qr; let u be the midpoint of pr; and let v be the midpoint of pq; see Figure 8. Depending on which of the three vertices is marked as the apex, the new time value $t'(p)$ is bounded as a result of these progress constraints in the three different ways enumerated below. Notice that when p is not the apex, lifting p also lifts either u or v, so progress constraints along edges qu or rv also indirectly limit $t'(p)$.

If p is the apex:

$$t'(p) \le \min \left\{ \begin{array}{l} t(q) + 2w_\varepsilon(\dot{s}pr), \\ t(s) + w_\varepsilon(\dot{p}qr), \\ t(r) + 2w_\varepsilon(\dot{s}pq) \end{array} \right\} \tag{2}$$

If q is the apex:

$$t'(p) \le \min \left\{ \begin{array}{l} t(q) + 2w_\varepsilon(\dot{u}qr), \\ t(r) + 4w_\varepsilon(\dot{v}qu), \\ 2t(q) - t(r) + 2w_\varepsilon(\dot{q}pr) \end{array} \right\} \tag{3}$$

If r is the apex:

$$t'(p) \le \min \left\{ \begin{array}{l} t(q) + 4w_\varepsilon(\dot{u}rv), \\ t(r) + 2w_\varepsilon(\dot{v}qr), \\ 2t(r) - t(q) + 2w_\varepsilon(\dot{r}pq) \end{array} \right\} \tag{4}$$

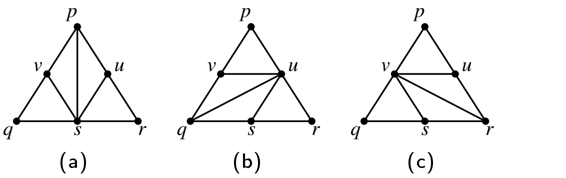


Figure 8. Cases for implementing the combined progress constraints: (a) p is the apex, (b) q is the apex, (c) r is the apex.

4.2 Proof of Positive Progress

In this section, we prove that for any triangle $\triangle pqr$ that satisfies the causality and progress constraints, we can lift $\hat{p}$ at least Δ above $\hat{q}$ without violating either constraint, where Δ is a positive function of the shape of the triangle and the scaling parameters ε and δ. The causality constraint and the progress constraints will each imply a different upper bound on the maximum progress Δ that we can guarantee.

Lemma 1. *If $\triangle\hat{p}\hat{q}\hat{r}$ satisfies the causality and progress constraints, we can set $t'(p) = t(q) + \varepsilon \cdot d(p, qr)$ without violating the causality constraint.*

Proof: First, we show that the progress constraints imply $t(r) - t(q) < (1-\varepsilon)|qr|$ no matter which vertex is the apex. If p is the apex, we have the following series of inequalities:

$$\begin{aligned} t(r) - t(q) &\leq 4w_\varepsilon(\dot{u}ps) \\ &\leq 4(1-\varepsilon)d(u, ps) \\ &= 2(1-\varepsilon)d(r, ps) \\ &\leq 2(1-\varepsilon)|rs| = (1-\varepsilon)|qr| \end{aligned}$$

Here, the first inequality is the actual progress constraint, the second follows from the definition of diminished width, and the rest follow from the geometry of the recursively bisected triangle. Similarly, if q is the apex, we have

$$\begin{aligned} t(r) - t(q) &\leq 2w_\varepsilon(\dot{u}pq) \\ &\leq 2(1-\varepsilon)d(u, pq) \\ &= (1-\varepsilon)d(r, pq) \leq (1-\varepsilon)|qr|, \end{aligned}$$

and if r is the apex, we have

$$\begin{aligned} t(r) - t(q) &\leq 2w_\varepsilon(\dot{v}pr) \\ &\leq 2(1-\varepsilon)d(v, pr) \\ &= (1-\varepsilon)d(q, pr) \leq (1-\varepsilon)|qr|. \end{aligned}$$

Solving for ε gives us the inequality

$$\varepsilon \leq 1 - \frac{t(r) - t(q)}{|qr|} \leq \sqrt{1 - \left(\frac{t(r) - t(q)}{|qr|}\right)^2},$$

which implies that

$$\begin{aligned} t'(p) &= t(q) + \varepsilon \cdot d(p, qr) \\ &\leq t(q) + \sqrt{1 - \left(\frac{t(r) - t(q)}{|qr|}\right)^2}\, d(p, qr) \\ &\leq t(q) + \frac{t(r) - t(q)}{|qr|^2}\ (\vec{qp} \cdot \vec{qr}) \\ &\quad + \sqrt{1 - \left(\frac{t(r) - t(q)}{|qr|}\right)^2}\, d(p, qr). \end{aligned}$$

This is exactly the expression of the causality constraint in equation (1). $\square$

Lemma 2. *If $0 < \varepsilon < \delta < (1+\varepsilon)/2 < 1$ and $\triangle\hat{p}\hat{q}\hat{r}$ satisfies the causality and progress constraints, we can set $t'(p) = t(q) + \Delta$ without violating any progress constraint, where $\Delta > 0$ is a function of the triangle $\triangle pqr$ and the parameters ε and δ.*

Proof: As in the previous proof, we have three cases to consider. In each case, we will derive positive upper bounds on the possible value of Δ; setting Δ to the minimum of these upper bounds will satisfy the conditions of the lemma.

First, suppose p is marked. Since $t(r) \geq t(s) \geq t(q)$, the relevant progress constraint (2) is satisfied if

$$\Delta \leq \min\{2w_\varepsilon(\dot{s}pr),\ w_\varepsilon(\dot{p}qr),\ 2w_\varepsilon(\dot{s}pq)\}.$$

Similarly, suppose r is marked. Since $t(r) \geq t(q)$, we have $2t(r) - 2t(q) \geq 0$, which implies that the relevant progress constraint (4) is satisfied if

$$\Delta \leq \min\{4w_\varepsilon(\dot{u}rv),\ 2w_\varepsilon(\dot{v}qr),\ 2w_\varepsilon(\dot{r}pq)\}.$$

Finally, suppose q is marked. By the inductive hypothesis, edge qr already satisfies its individual progress constraint $t(r) \leq t(q) + 2w_\varepsilon(\dot{u}pq)$, which implies that $t(q) - t(r) \geq -2w_\varepsilon(\dot{u}pq)$. Since $t(r) \geq t(q)$, the relevant progress constraint (3) is satisfied if

$$\Delta \leq \min\{2w_\varepsilon(\dot{u}qr),\ 4w_\varepsilon(\dot{v}qu),\ 2w_\varepsilon(\dot{q}pr) - 2w_\varepsilon(\dot{u}pq)\}.$$

The last term in this expression is the only one that is not obviously positive. To prove that $X = w_\varepsilon(\dot{q}pr) - w_\varepsilon(\dot{u}pq)$ is in fact bounded away from zero, we expand the definition of diminished width. We have $X = \min\{A, B, C\}$, where

$$\begin{aligned} A &= (1-\varepsilon)d(q, pr) - \min\left\{\begin{array}{l}(1-\varepsilon)d(u, pq), \\ (1-\delta)d(p, qu), \\ (1-\delta)d(q, pu)\end{array}\right\} \\ &\geq (1-\varepsilon)d(q, pr) - (1-\delta)d(q, pu) \\ &= (\delta - \varepsilon)d(q, pr), \end{aligned}$$

$$\begin{aligned} B &= (1-\delta)d(r, pq) - \min\left\{\begin{array}{l}(1-\varepsilon)d(u, pq), \\ (1-\delta)d(p, qu), \\ (1-\delta)d(q, pu)\end{array}\right\} \\ &\geq (1-\delta)d(r, pq) - (1-\varepsilon)d(u, pq) \\ &= (1 - 2\delta + \varepsilon)d(r, pq), \end{aligned}$$

$$\begin{aligned} C &= (1-\delta)d(p, qr) - \min\left\{\begin{array}{l}(1-\varepsilon)d(u, pq), \\ (1-\delta)d(p, qu), \\ (1-\delta)d(q, pu)\end{array}\right\} \\ &\geq (1-\delta)\big(d(p, qr) - \min\{d(p, qu), d(q, pu)\}\big). \end{aligned}$$

Since by assumption $\varepsilon < \delta < (1+\varepsilon)/2$, we clearly have $A > 0$ and $B > 0$. We prove that $C > 0$ as follows:

$$\begin{aligned} d(p, qr) &= \frac{2\,\mathrm{Area}(\triangle pqr)}{|qr|} \\ &= \frac{2\,\mathrm{Area}(\triangle pqu)}{|uv|} \\ &> \frac{2\,\mathrm{Area}(\triangle pqu)}{\max\{|qu|, |pu|\}} \\ &= \min\{d(p, qu), d(q, pu)\}. \end{aligned}$$

The key observation is that the bisector segment uv must be shorter than at least one of the two edges pu and qu. $\square$

We have thus proved the following theorem.

Theorem 3. *Given a triangular mesh Ω in the plane and a target time value T, our algorithm builds a mesh of the spacetime domain $\Omega \times [0, T]$ consisting of a finite number of tetrahedral elements, in constant running time per element, provided that each triangle is refined only a finite number of times.*

5. EXPERIMENTAL RESULTS

We have implemented our adaptive spacetime meshing algorithm in 1D×time and 2D×time. We present here an example to demonstrate its application to a scientific problem of practical complexity: the phenomenon of crack-tip wave scattering within an elastic solid subjected to shock loading.

Figure 9 shows a stationary crack embedded in a rectangular plate subjected to a spatially uniform tensile traction on its top and bottom edges. We prescribe traction-free boundary conditions on the left and right edges and plane-strain conditions overall. The load history approximates a step function; as depicted in Figure 9, the load intensity σ rises from 0 to a maximum value $\sigma^* = 10$ over the ramp time $t_{\text{ramp}} = 0.0044$. The material properties of the plate are: Young's modulus $E = 10$, Poisson's ratio $\nu = 0.1$, and mass density $\rho = 1$. We model the upper right quadrant using symmetry boundary conditions. The initial space mesh contains only 16 triangles; we depend on the adaptive meshing algorithm to generate all of the necessary grid refinement. We use a complete quadratic polynomial basis, $(1\ x\ y\ t\ xy\ yt\ tx\ x^2\ y^2\ t^2)$, to represent the displacement field in each tetrahedral spacetime element.

A dissipation-based error indicator drives the adaptive meshing scheme. The energy dissipated by each patch is compared to the total energy flux into the patch (zero dissipation indicates exact energy balance). We set a normalized dissipation target, dissipation/energy influx $= 1.0 \times 10^{-7}$. Elements with dissipation more than 20% above the target are rejected and marked for mandatory refinement. Elements with dissipation more than 20% below the target are accepted and marked as eligible for coarsening. We also specify a minimum dissipation of 2.5×10^{-10}, below which no further refinement is allowed. The final adapted spacetime mesh contains $PPPP$ patches and $EEEE$ tetrahedral elements; the ratio of the largest to smallest element diameters in the adapted grid is 5.0×10^5. The normalized dissipation for the entire spacetime analysis domain is 2.0×10^{-7}.

Figure 10 shows three pairs of visualizations of the computed solution. Each view shows the upper right quadrant of the plate; the color field portrays the strain-energy-density on a logarithmic scale, while the height field indicates the velocity magnitude. Each image was produced by computing the intersection of a constant-time plane with the spacetime data set. The images on the right show the trace of the unstructured spacetime tetrahedral grid on the constant-time planes.

The first pair of images in Figure 10 shows the main shock front as it approaches the crack surface for the first time. The suddenly applied loading generates a plane shock wave

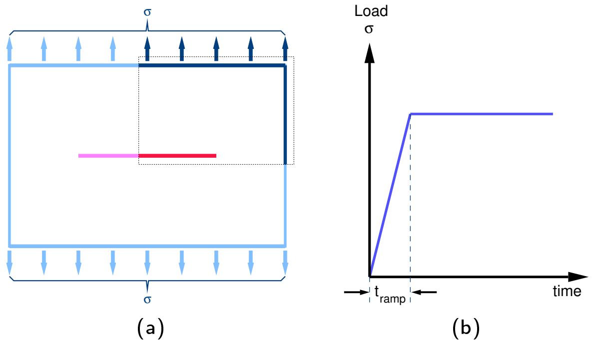


Figure 9. Crack geometry and loading for the crack-tip wave scattering problem.

that moves from the top loaded edge toward the crack surface at the bottom. The pattern of mesh refinement is responsive to the fine details of the traveling waves.

The second pair of images shows the initial crack-tip wave scattering pattern some time after the shock first arrives at the crack. The circular scattering pattern is reflected in the mesh refinement. A singular strain-energy-density field is visible at the crack tip, a stationary feature that generates a persistent region of intense mesh refinement. The third pair of images shows the solution at a later time, when the primary wave has reflected off the top loaded surface and returned to scatter a second time at the crack tip.

Figure 11 shows two views of the spacetime mesh at roughly the same time depicted in Figures 10(b)–(c). The vertical axis is time, and the pattern of refinement on the vertical faces follows closely the spacetime trajectories of the shocks; see Figure 11(a). The cone-shaped region of intense mesh refinement in Figure 11(b) covers the domain of influence of the initial crack-tip scattering event.

6. CURRENT AND FUTURE WORK

We implemented a sequential version of our adaptive Tent Pitcher algorithm for one- and two-dimensional spatial domains which we used to solve linear PDEs with constant wavespeed and nonlinear problems where the maximum wavespeed is bounded. We also prototyped a parallel version for 1D×time that leverages the mutual independence of any two patches over the same front. Since the patches can be solved independently of each other, we were able to solve them simultaneously on different processors. We used the CHARM++ parallel language and runtime system [11] developed by the Parallel Programming Group at the University of Illinois [10] to manage interprocessor communication, load balancing, and the automatic dispatch and scheduling of patches to be solved. In our preliminary experiments, we observed a parallel speedup of about 20. We observed that the speedup was limited mainly by the mesh generation, which was not running in parallel.

We are also investigating spacetime discontinuous Galerkin methods for nonlinear conservation laws. In such problems, the wavespeed is not constant throughout the spacetime domain. Instead, the wavespeed is one of the physical parameters governed by the underlying PDE, and can therefore change in unpredictable ways. We have extended Tent Pitcher to adapt the number and duration of spacetime elements to changing wavespeeds, for problems defined over

Figure 10. Adaptive solution of crack-tip wave scattering problem: (a)–(b) the shock front approaches the crack surface; (b)–(c) initial scattering pattern after first reflection off the crack surface; (d)–(e) second scattering pattern after the main shock front returns to the crack surface. The color field portrays strain-energy-density on a logarithmic scale, while the height field indicates the magnitude of the velocity field. The images on the right show the trace of the spacetime tetrahedral mesh on a current constant-time plane.

one- and two-dimensional spatial domains. As in the current paper, the major theoretical bottleneck lies in developing appropriate progress constraints. We have implemented our algorithm for nonlinear problems in 1D×time, and we are currently implementing the algorithm in 2D×time. We expect to publish these results in a subsequent paper.

We are also working on combining the mesh refinement with adaptivity to local wavespeeds. We plan to implement this combined, completely adaptive algorithm in parallel.

Finally, we would like to extend all the current results to higher dimensions, specifically to 3D×time. For the actual mesh refinement, we can directly use the generalization of newest-vertex bisection to three dimensions by Bänsch [3]; see also Arnold *et al.* [2], Bey [5], Liu and Joe [13, 14], Maubach [16], and Merrouche *et al.* [17]. Again, the main theoretical hurdle in higher dimensions is deriving minimal progress constraints that guarantee that the front can always advance. Another concern is to be able to describe the constraints in such a fashion that they are easy to implement. Up to our current work in 2D×time, we have encountered only a constant number of linear constraints per element. It seems likely that such constraints that can be described analytically should suffice in higher dimensions, but it is not clear that they are the best possible.

Acknowledgments

The authors thank the other members of the CPSD spacetime discontinuous Galerkin group — Morgan Hawker, Mark Hills, Katarina Jegdic, Robert Jerrard, Sanjay Kale, Yangsuk Ko, Jayandran Palaniappan, and Boris Petracovici — as well as Jonathan Booth and David Bunde for several helpful comments and discussions. Shripad Thite thanks Alper Üngör for his encouragement.

References

[1] A. Adler. On the bisection method for triangles. *Math. Comp.* 40(162):571–574, 1983.

[2] D. N. Arnold, A. Mukherjee, and L. Pouly. Locally adaptive tetrahedral meshes using bisection. *SIAM J. Sci. Comput.* 22(2):431–448, 2001.

[3] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *Impact of Computing in Science and Engineering* 3:181–191, 1991.

[4] M. Bern and P. Plassmann. Mesh generation. *Handbook of Computational Geometry*, 291–332, 2000. Elsevier Science Publishers B.V. North-Holland.

[5] J. Bey. Tetrahedral grid refinement. *Computing* 55(4):355–378, 1995.

[6] B. Cockburn and P. A. Gremaud. Error estimates for finite element methods for hyperbolic conservation laws. *SIAM J. Num. Anal.* 33:522–554, 1996.

[7] B. Cockburn, G. Karniadakis, and C. Shu. *Discontinuous Galerkin methods: theory, computation and applications.* Lecture Notes in Computational Science and Engineering 11. Springer, 2000.

[8] J. Erickson, D. Guoy, J. M. Sullivan, and A. Üngör. Building space-time meshes over arbitrary spatial domains. *Proc. 11th Int. Meshing Roundtable*, 391–402, 2002.

[9] M. T. Jones and P. E. Plassmann. Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design* 25:41–60, 1997.

[10] L. V. Kalé et al. Parallel Programming Laboratory, Computer Science Department, Univerity of Illinois. ⟨http://charm.cs.uiuc.edu/⟩.

[11] L. V. Kalé and S. Krishnan. Charm++: Parallel programming with message-driven objects. *Parallel Programming using C++*, 175–213, 1996. MIT Press.

[12] B. Kearfott. A proof of covergence and an error bound for the method of bisection in $\mathbb{R}^n$. *Math. Comp.* 32(144):1147–1153, 1978.

[13] A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Math. Comp.* 63(207):141–154, 1994.

[14] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM J. Sci. Comput.* 16:1269–1291, 1995.

[15] R. B. Lowrie, P. L. Roe, and B. van Leer. Space-time methods for hyperbolic conservation laws. *Barriers and Challenges in Computational Fluid Dynamics*, 79–98, 1998. ICASE/LaRC Interdisciplinary Series in Science and Engineering 6, Kluwer.

[16] J. M. Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *SIAM J. Sci. Comput.* 16:210–227, 1995.

[17] A. Merrouche, A. Selman, and C. Knopf-Lenoir. 3D adaptive mesh refinement. *Communications In Numerical Methods In Engineering* 14:397–407, 1998.

[18] W. F. Mitchell. *Unified multilevel adaptive finite element methods for elliptic problems.* Ph. D. thesis, Computer Science Department, University of Illinois, Urbana, IL, 1988. Tech. Rep. UIUCDCS-R-88-1436.

[19] W. F. Mitchell. A comparison of adaptive refinement techniques for elliptic problems. *ACM Trans. Math. Soft* 15:326–347, 1989.

[20] W. F. Mitchell. Adaptive refinement for arbitrary finite-element spaces with hierarchical bases. *J. Comp. Appl. Math.* 36:65–78, 1991.

[21] J. Palaniappan, R. B. Haber, and R. L. Jerrard. A spacetime discontinuous galerkin method for scalar conservation laws. *Comp. Methods Appl. Mechs. Engng.* , 2004. in press.

[22] G. R. Richter. An explicit finite element method for the wave equation. *Applied Numerical Mathematics* 16:65–80, 1994.

[23] M. C. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *Internat. J. Numer. Methods Eng.* 20:745–756, 1984.

[24] M. C. Rivara. A grid generator based on 4-triangles conforming mesh-refinement algorithms. *Internat. J. Numer. Methods Eng.* 24(7):1343–1354, 1987.

[25] M. C. Rivara. Local modification of meshes for adaptive and/or multigrid finite element methods. *J. Comp. Appl. Math.* 36:79–89, 1991.

[26] I. G. Rosenberg and F. Stenger. A lower bound on the angles of triangles constructed by bisecting the longest side. *Math. Comput.* 29:390–395, 1975.

[27] E. G. Sewell. *Automatic generation of triangulations for piecewise polynomial approximation.* Ph. D. thesis, Department of Mathematics, Purdue University, West Lafayette, IN, 1972.

[28] A. Sheffer, A. Üngör, S.-H. Teng, and R. B. Haber. Generation of 2D space-time meshes obeying the cone constraint. *Advances in Computational Engineering & Sciences*, 1360–1365, 2000. Tech Science Press.

[29] M. Stynes. On faster convergence of the bisection method for all triangles. *Math. Comp.* 35(152):1195–1201, 1980.

[30] J. P. Suárez, A. Plaza, and G. F. Carey. Propagation path properties in iterative longest-edge refinemen. *Proc. 12th Internat. Meshing Roundtable*, 79–90, 2003. ⟨http://www.andrew.cmu.edu/user/sowen/abstracts/Su983.html⟩.

[31] 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.

[32] A. Üngör, C. Heeren, X. Li, A. Sheffer, R. B. Haber, and S.-H. Teng. Constrained 2D space-time meshing with all tetrahedra. *Proc. 16th IMACS World Congress*, 2000.

[33] A. Üngör and A. Sheffer. Pitching tents in space-time: Mesh generation for discontinuous Galerkin method. *Int. J. Foundations of Computer Science* 13(2):201–221, 2002.

[34] A. Üngör, A. Sheffer, R. B. Haber, and S.-H. Teng. Layer based solutions for constrained space-time meshing. *Appl. Numer. Math.* 46:425–443, 2003.

[35] L. Yin. *A Spacetime Discontinuous Galerkin Finite-Element Method for Elastodynamic Analysis*. Ph. D. thesis, Department of Theoretical & Applied Mechanics, University of Illinois, Urbana, IL, 2002.

[36] L. Yin, A. Acharya, N. Sobh, R. Haber, and D. A. Tortorelli. A space-time discontinuous Galerkin method for elastodynamic analysis. *Lecture Notes in Computational Science and Engineering*, 459–464, 2000. vol. 11, Springer.

Figure 11. Spacetime mesh during the solution of the crack–tip wave scattering problem: (a) overview from the traction free edge; (b) detail of the initial crack-tip wave scattering pattern.