Size functions and mesh generation for high-quality adaptive remeshing

Per-Olof Persson*

Massachusetts Institute of Technology, Department of Mathematics, Cambridge, MA 02139, USA

Abstract

We present a new method for remeshing of triangular and tetrahedral meshes. Relative element sizes are computed from an error estimation. Then their gradient is limited in an optimal way by solving a Hamilton–Jacobi equation numerically. The new mesh is generated using smoothing-based iterations with connectivity updates (changes in topology of the mesh). The boundary nodes are projected using an implicit geometry representation based on distance functions. Our algorithm is simple and efficient, and it produces high-quality meshes.

Keywords: Adaptation; Mesh generation; Mesh size functions; Gradient limiting

1. Introduction

An adaptive finite element solver starts from an initial mesh, solves the physical problem, and estimates the error in each mesh element. From this, a new mesh size function can be derived, for example by equidistributing the error across the domain. The challenge for the mesh generator is to create a new high-quality mesh, conforming to the size, function, and other geometrical constraints.

One approach is to refine the existing mesh, by splitting elements that are too large and possibly also coarsening small elements. These local refinement techniques are efficient and robust and provide simple solution transfer between the meshes. The refinement can be made in a way that completely avoids bad elements, but the average qualities usually drop during the process.

An alternative is to remesh the domain by generating a new mesh from scratch based on the desired size function. This technique has been considered expensive, but it can produce meshes of very high quality if the size function is well-behaved. However, the size functions arising from adaptive solvers may have large gradients, and they have to be modified before being used with a mesh generator that relies on good size functions. These include the advancing front method [1], the paving method for quadrilateral meshing [2], and the smoothing-based algorithm [3]. The Delaunay refinement method [4,5] generates meshes from any size function, but the element qualities are usually higher with good size functions.

In this work, we describe a method that starts by limiting the gradient of the size function, by solving a nonlinear partial differential equation. We then refine the existing mesh using a simple density control, without worrying about the qualities, and apply an iterative procedure to improve the mesh [3]. Assuming a piecewise linear force–displacement relationship in the mesh edges, we find an equilibrium position for the nodes. Mesh points that leave the domain during an update are projected back using an implicit geometry representation. Many mesh generators use simple Laplacian smoothing as a postprocessing step, but our method can start from arbitrarily bad elements and generate good meshes, since it also modifies the mesh connectivity and the distribution of the boundary nodes. Also, limiting the gradient produces high-quality meshes by inserting a minimum of new nodes.

2. Element size functions

The mesh size function $h(\boldsymbol{x})$ is important for generation of high-quality meshes. It should satisfy the size constraints specified by the adaptive solver as well as the geometrical constraints such as curvature and feature

* Tel.: +1 781 856 2608; Fax: +1 617 253 4358;
E-mail: persson@mit.edu

Computational Fluid and Solid Mechanics 2005
K.J. Bathe (Editor)

size. In addition, the element size should not differ too much between neighboring elements, which corresponds to a limit on the gradient $|\nabla h(\boldsymbol{x})|$.

2.1. Numerical adaptation

An adaptive solver provides an error estimate in each element of the mesh, from which a new size function $h(\boldsymbol{x})$ can be derived. This size function specifies smaller element sizes in some regions and larger sizes in others. It typically does not take specific account of features of the geometry. Its gradient may not be limited. If we were to generate a mesh according to this function, then it would likely produce elements of poor quality.

2.2. Geometric adaptation

In addition to the numerical constraints, the size function should also be adapted to the geometry of the computational domain. At boundaries with high curvature, small elements are required. In thin regions, with small distance between boundaries, the elements have to be small in order to have high quality. Our method accepts any geometric size function, and in our examples we compute it directly from our implicit geometry representation. The curvature is given by the Laplacian of the distance function, and we compute the feature size from the medial axis, which we detect as shocks in the distance function. The curvature adaptation specifies sizes only at the boundaries, and we want to extend $h(\boldsymbol{x})$ to the interior.

2.3. Gradient limiting

We can form a total size function $h_0(\boldsymbol{x})$ as the minimum of the numerical and geometrical requirements, as well as any user-specified size constraints. As a final step, we now limit the gradients $|\nabla h(\boldsymbol{x})| \le g$ to bound the size ratio of neighboring elements in the new mesh.

Recently, we presented a new technique for gradient limiting [6]. This is based on the steady-state solution to a Hamilton–Jacobi equation. For convex domains, we showed that we obtain an optimal result:

Theorem 2.1: Let $\Omega \subset \mathbb{R}^n$ be a bounded convex domain and $I = (0, T)$ a given time interval. The steady-state solution $h(x) = \lim_{T\to\infty} h(x, T)$ to

$$\begin{cases} \frac{\partial h}{\partial t} + |\nabla h| = \min(|\nabla h|, g) & (x,t) \in \Omega \times I \\ h(x,t)|_{t=0} = h_0(x) & x \in \Omega \end{cases} \tag{1}$$

is

$$h(x) = \min_y (h_0(y) + g|x-y|) \tag{2}$$

Proof: Use the Hopf–Lax theorem [7] (see Persson [6] for details).

Note how the solution in Eq. (2) is a minimum of infinitely many point-source solutions $h_0(y) + g|x - y|$. Then, $h(\boldsymbol{x})$ is optimal in the sense of minimum deviation from the original size function. We could, in principle, define an algorithm based on Eq. (2) for computing h from a given h_0 (both discretized). Such an algorithm would be trivial to implement, but its computational complexity would be proportional to the square of the number of node points. Instead, we solve Eq. (1) using efficient numerical Hamilton–Jacobi solvers [8,9].

3. Mesh generation

We now turn to the generation of an unstructured mesh for our size function $h(\boldsymbol{x})$. As a simple model example, we solve Poisson's equation with a delta source and estimate the error in the energy norm [10]. The initial mesh and the gradient limited size function are shown in Fig. 1(a).

Our meshing algorithm needs an initial guess for the new locations of the mesh points. Previously [3], we used a random technique based on the rejection method to obtain a point density according to $h(\boldsymbol{x})$. However, for an adaptive solver, we can obtain a good initial guess with correct connectivity faster by density control; that is, splitting edges and merging neighboring nodes (Fig. 1b). Note that this mesh does not have to be of high quality or have good connectivity, so any simple scheme can be used.

To improve this initial mesh, we assign forces in the mesh edges and solve for force equilibrium at the nodes. The force in an edge depends on the length ℓ of the edge and on its unstretched length ℓ_0 (which we set proportional to the desired mesh size $h(\boldsymbol{x})$ evaluated at the edge midpoint). We use a linear spring model to push nodes outward:

$$f(\ell, \ell_0) = \begin{cases} k(\ell_0 - \ell) & \text{if } \ell < \ell_0, \\ 0 & \text{if } \ell \ge \ell_0. \end{cases} \tag{3}$$

By summing the forces at all mesh positions $\boldsymbol{p}$ (for each coordinate direction), we obtain a nonlinear system of equations $\boldsymbol{F}(\boldsymbol{p}) = 0$. We find the positions as a steady state of

$$\frac{d\boldsymbol{p}}{dt} = \boldsymbol{F}(\boldsymbol{p}), \qquad t \ge 0 \tag{4}$$

using forward Euler. Note that this artificial time dependence is unrelated to the (real) time evolution of the geometry as given by $\phi(\boldsymbol{x})$. After each Euler step, we apply normal boundary forces by projecting any

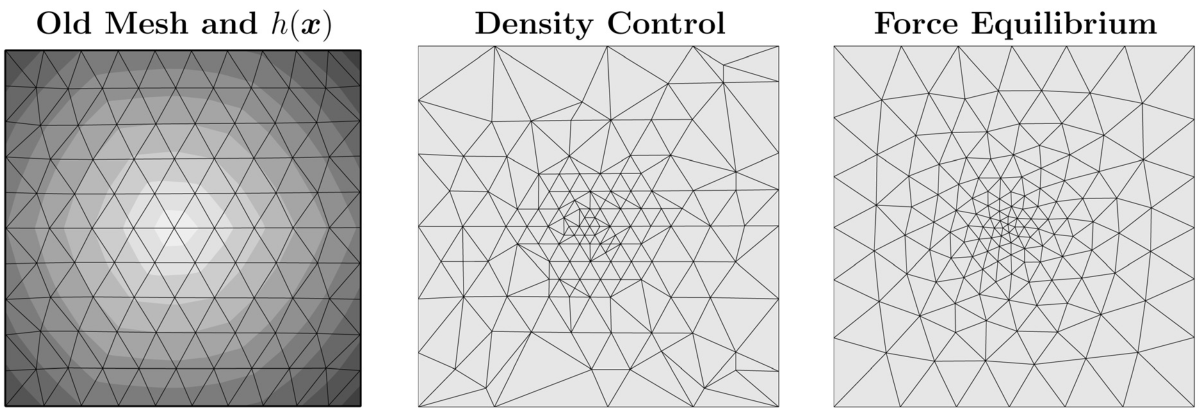

Fig. 1. The steps of the remeshing algorithm. First, a gradient limited size function $h(\boldsymbol{x})$ is generated by solving Eq. (1) on the old mesh. Next, the node density is controlled by edge splitting and merging. Finally, we solve for a force equilibrium in the edges using forward Euler iterations.

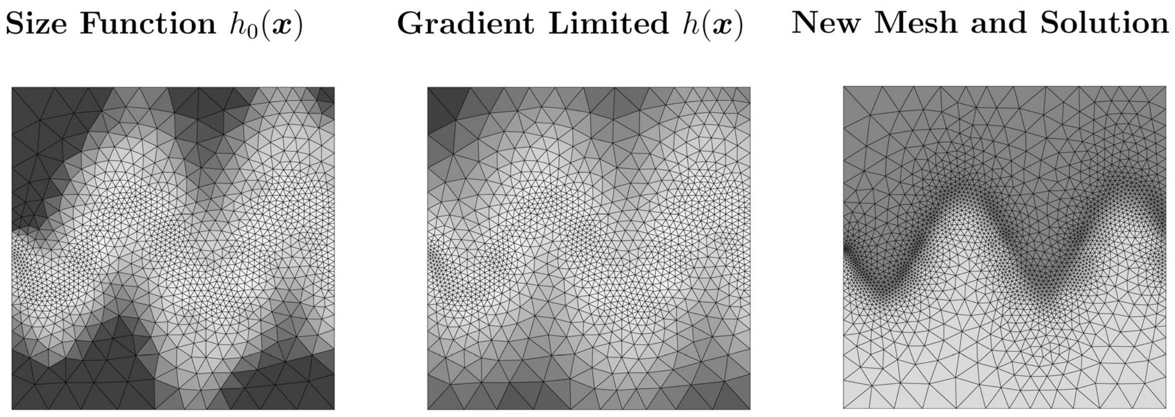
Size Function $h_0(\boldsymbol{x})$
Gradient Limited $h(\boldsymbol{x})$
New Mesh and Solution

Fig. 2. An example of numerical adaptation for solution of Eq. (6). Note the large gradients in the original size function $h_0(\boldsymbol{x})$ and how the gradient limiting improves it.

external nodes back orthogonally to the boundary using the distance function:

$$\boldsymbol{p} \leftarrow \boldsymbol{p} - \phi(\boldsymbol{p})\nabla\phi(\boldsymbol{p}) \tag{5}$$

These normal forces may be seen as Lagrange multipliers that keep nodes exactly along the boundary. This expression can be modified to allow general implicit functions instead of distance functions, either by solving nonlinear equations [3] or by approximate projections.

During the iterations, we always maintain a good connectivity by updating the triangulation. In the previous simple code [3], this was done by recomputing the Delaunay triangulation. Now we have implemented more efficient and robust versions based on local topology updates (such as edge flips). When the mesh quality is sufficiently high, we terminate (Fig. 1c).

4. Results

We show three examples of numerical adaptation and remeshing using our methods.

4.1. Convection with discontinuity

Our first example solves a simple convective model problem on a square geometry:

$$\boldsymbol{v} \cdot \nabla u(x,y) = 0 \text{ with } \boldsymbol{v} = [1, -2\pi A \cos 2\pi x], \quad (x,y) \in (-1,1) \times (-1,1) \tag{6}$$

with a jump in the left boundary condition, $v(0, y)$ = Heaviside (y). We discretize using linear finite elements with streamline-diffusion stabilization. To obtain an accurate numerical solution, the discontinuity along $y = A \sin 2\pi x$ has to be resolved. We do this using numerical

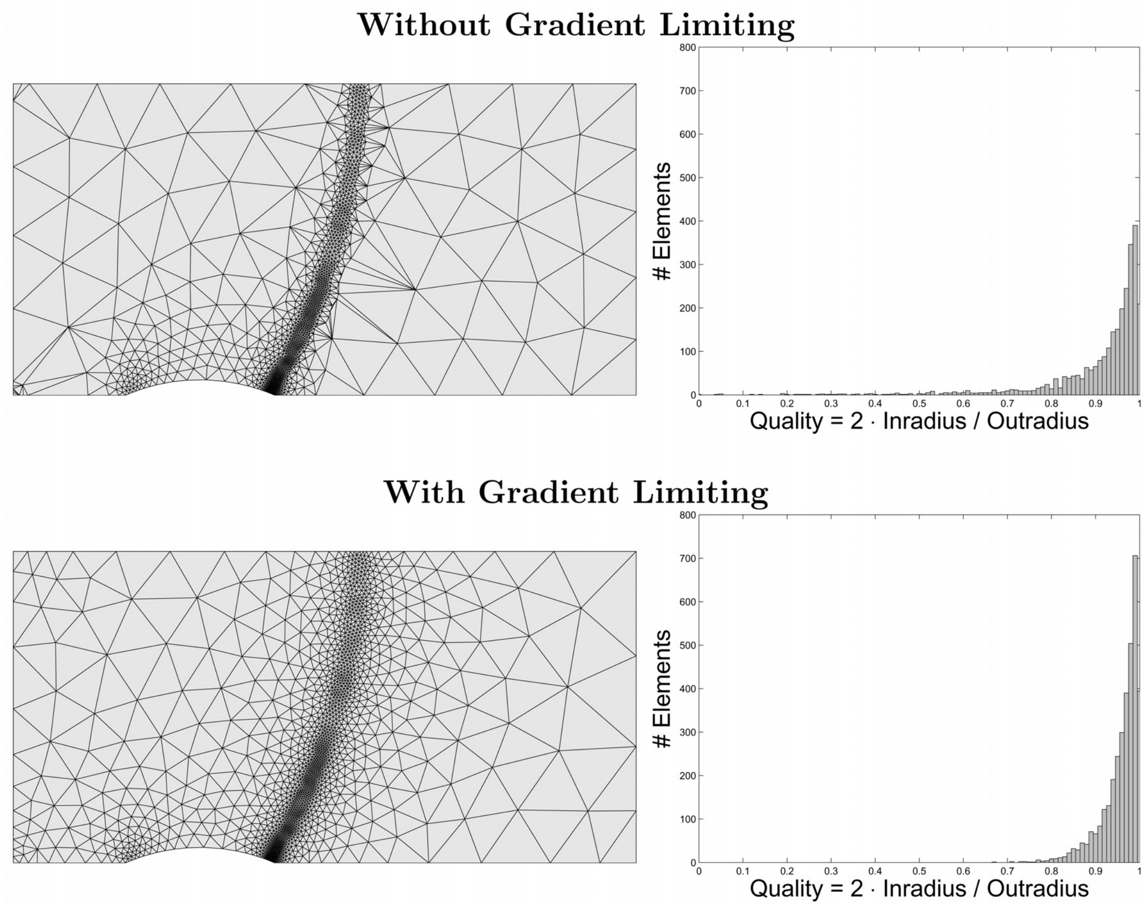

Fig. 3. Numerical adaptation for compressible flow over a bump at Mach 0.95. The second-derivative-based error estimator resolves the shock accurately, but gradient limiting is required to generate a new mesh of high quality.

adaptation in the L_2-norm [10]. The size function from the adaptive scheme is highly irregular, with large variations in element sizes, which would give low-quality triangles (Fig. 2a). After gradient limiting, the mesh size function is well behaved (Fig. 2b) and a high-quality mesh can be generated (Fig. 2c).

4.2. Compressible flow over bump

Our second example simulates compressible flow over a bump at Mach 0.95. A simple adaptive scheme based on second-derivatives of the density [1] resolves the shock accurately but increases the sizes sharply outside the shock. With gradient limiting, a high-quality mesh is generated (Fig. 3).

4.3. Linear elasticity

A final example shows a three-dimensional mechanical component, with forces applied on the circular holes. We use adaptation in the energy norm as well as a geometric mesh size function. The gradient limiting equation extends naturally to three dimensions, and we create tetrahedral meshes with the methods described in [3] and [12]. Bad elements are removed by face-swapping and edge-flipping [11] and the resulting mesh has high elements qualities (Fig. 4).

5. Conclusions

Our new technique remeshes geometries starting with a size function from a previous mesh. This size function is gradient limited by numerical solution of Eq. (1), and a new mesh is generated by solving for a force equilibrium in the mesh edges. The iterations are well-suited for adaptive meshing and moving meshes [12], since the old mesh provides a good initial configuration.

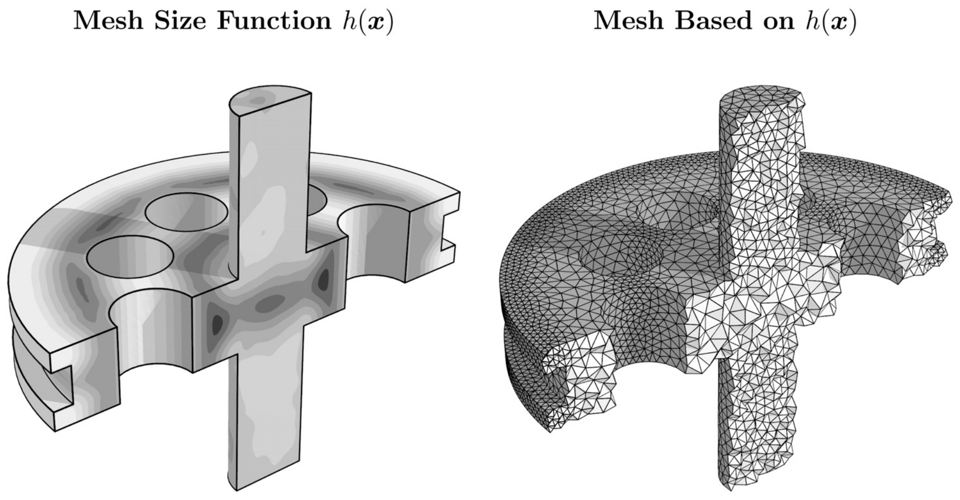
Mesh Size Function $h(\boldsymbol{x})$
Mesh Based on $h(\boldsymbol{x})$

Fig. 4. Adaptation in the energy norm for a linear elasticity problem in three dimensions. The size function is created using the error estimator and the geometrical features.

References

[1] Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J Computat Phys 1987;72:449–466.

[2] Blacker TD, Stephenson MB. Paving: a new approach to automated quadrilateral mesh generation. Int J Numer Meth Engng 1991;32:811–847.

[3] Persson P-O, Strang G. A simple mesh generator in matlab. SIAM Rev 2004;46:329–345.

[4] Ruppert J. A delaunay refinement algorithm for quality 2-dimensional mesh generation. J Algorithms 1995;18:548–585.

[5] Shewchuk JR. Delaunay refinement algorithms for triangular mesh generation. Computat Geometry Theory Appl 2002;22:21–74.

[6] Persson P-O. PDE-based gradient limiting for mesh size functions. In: Proc of 13th International Meshing Roundtable. Sandia Nat Lab, Albuquerque, NM, September 2004, pp. 377–387.

[7] Hopf E. Generalized solutions of non-linear equations of first order. J Math Mech 1965;14:951–973.

[8] Barth TJ, Sethian JA. Numerical schemes for the Hamilton–Jacobi and level set equations on triangulated domains. J Computat Phys 1998;145:1–40.

[9] Kimmel R, Sethian JA. Fast marching methods on triangulated domains. Proc Nat Acad Sc USA 1998;95:8341–8435.

[10] Eriksson K, Estep D, Hansbo P, Johnson C. Computational Differential Equations. Press Syndicate of the University of Cambridge, Cambridge University Press, Cambridge 1996.

[11] Freitag L, Ollivier-Gooch C. Tetrahedral mesh improvement using swapping and smoothing. Int J Numer Meth Engng 1997;40:3979–4002.

[12] Persson P-O. Mesh generation for implicit geometries, PhD Thesis, Massachusetts Institute of Technology, 2005.