Weierstraß-Institut

für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Mesh smoothing: An MMPDE approach

Weizhang Huang[1], Lennard Kamenski[2], Hang Si[2]

submitted: June 17, 2015

[1] The University of Kansas
Department of Mathematics
Lawrence, KS 66045
US
email: whuang@ku.edu

[2] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
email: Lennard.Kamenski@wias-berlin.de
Hang.Si@wias-berlin.de

No. 2130

Berlin 2015

2010 *Mathematics Subject Classification.* 65N50, 65K10.

Key words and phrases. mesh smoothing, moving mesh method, tetrahedral meshes.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

We study a mesh smoothing algorithm based on the moving mesh PDE (MMPDE) method. For the MMPDE itself, we employ a simple and efficient direct geometric discretization of the underlying meshing functional on simplicial meshes. The nodal mesh velocities can be expressed in a simple, analytical matrix form, which makes the implementation of the method relatively easy and simple. Numerical examples are provided.

1 Introduction

Meshes generated with automatic tools often contain poorly shaped elements especially near domain boundaries and their quality needs to be improved before they can be used in the numerical solution of PDEs or other applications. Common means for improving the mesh quality include edge and face swapping, mesh smoothing, centroidal Voronoi tessellations (CVTs), and optimal Delaunay triangulations (ODTs).

Edge and face swapping improves mesh quality by improving connectivity while fixing vertex locations and is the most economic and efficient way to improve the mesh quality [10, 14]. In 3D, it is much more difficult to implement than in 2D and it tends to stuck locally; when it stucks, more complicated operations have to be applied.

CVTs are a special type of Voronoi tessellation where the site of each Voronoi cell is positioned at the centroid of the cell [5]. The major effort in CVT methods is to generate a CVT from a Voronoi tessellation by improving vertex locations. Commonly used methods include Lloyd's method [20] (which in 3D is not as efficient as in 2D) and a quasi-Newton method [19]. Voronoi tessellation leads to general polygonal/polyhedral meshes. Their duals, Delaunay triangulations, can also be used. In 2D, a CVT gives a Delaunay triangulation of high quality; however, in 3D, a Delaunay triangulation resulting from a CVT may contain slivers.

ODT, a more recent concept proposed by Chen and Xu [4], achieves an optimal Delaunay triangulation by improving vertex locations, although it is slow in convergence. An improved algorithm that updates vertex locations simultaneously in each iteration (global updating) is proposed by Alliez et al. [1].

Mesh smoothing improves the mesh quality by improving vertex locations, typically through Laplacian smoothing or some optimization-based algorithms. It is often quite effective in eliminating extremal dihedral angles in the mesh. Most commonly used mesh smoothing methods are Laplacian smoothing and its variants (Field [7] and Lo [21]), where a vertex is moved to the geometric center of its neighboring vertices. While economic, easy to implement, and often effective, Laplacian smoothing does not actually guarantee an improvements of the mesh quality.

Alternatives are optimization-based methods that are effective for a variety of mesh quality measures. For example, Bank [2] uses the ratio of the area to the sum of the squared edge lengths for triangular meshes, Shephard and Georges [22] employ the ratio of the volume to a power of the sum of the squared face areas for tetrahedral meshes, Freitag and Ollivier-Gooch [10] use various angle-based measures, and Freitag and Knupp [9] utilize the condition number of the Jacobian matrix of the affine mapping between the reference element and physical elements. Knupp [15, 17] proposes several shape quality measures based on the Jacobian matrix and Canann et al. [3] propose a distortion metric for triangles and quadrilaterals. The mentioned optimization-based methods are local and sequential (by nature), with Gauss-Seidel-type iterations being combined with location optimization problems (over each patch). A parallel algorithm that solves a sequence of independent subproblems is proposed by Freitag et al. [8] for a class of local mesh-smoothing techniques. These methods are based on optimization of some shape regularity measure. They are all local (Gauss-Seidel-type iterative) smoothing methods, sequential in nature, and, thus, not amenable to parallel computation.

The objective of this paper is to study a mesh smoothing algorithm based on the moving mesh PDE (MMPDE), defined as the gradient flow equation of a meshing functiona to move mesh continuously in time (i.e., an objective functional in the context of optimization). Typically, such a functional is based on error estimation, physical and/or geometric considerations. We consider here two functionals, Winslow's functional based on variable diffusion and Huang's functional based on mesh uniformity in the metric specified by a tensor $\mathbb{M} = \mathbb{M}(\boldsymbol{x})$. The uniformity requirement is equivalent to the equidistribution and alignment conditions combined. We shall use a simple and efficient approach recently developed by Huang and Kamenski [12] for the implementation of variational mesh generation and adaptation. The approach is based on a direct geometric discretization of the underlying meshing functional on simplicial meshes. Most importantly, the nodal mesh velocities can be expressed in a simple, analytical matrix form, which makes the implementation of the method relatively easy and simple. It also makes the method amenable to development of efficient solvers and parallel computation. The approach is incorporated into the implementation of the MMPDE method.

Compared to existing optimization-based (local) mesh smoothing methods, the current method has several advantages:

- it is based on a continuous functional for which the existence of minimizers is known,
- the functional has a clear geometric meaning for controlling mesh shape and size quality,
- mesh velocities have a simple, analytical matrix form, which makes programming relatively easy,
- the resulting ODE systems can be solved locally (e.g., by a Gauss-Seidel-type iteration) or globally,
- it is amenable to parallel computation.

2 The MMPDE method

In the moving mesh context, it is common to describe a moving mesh in terms of coordinate transformations. Assume that we are given a computational domain Ω_c, which can be the same as the physical domain $\Omega \subset \mathbb{R}^d$, $d \geq 1$. A moving mesh on Ω is the image of a computational mesh on Ω_c under a time dependent coordinate transformation $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)\colon \Omega_c \to \Omega$. The coordinate transformation in the MMPDE method is determined as the solution of the gradient flow equation of a meshing functional (i.e., an objective functional to optimize). Many existing functionals can be written in the form

$$I[\boldsymbol{\xi}] = \int_\Omega G(\mathbb{J}, \det(\mathbb{J}), \mathbb{M}, \boldsymbol{x}) dx, \tag{1}$$

where $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)\colon \Omega \to \Omega$ is the inverse coordinate transformation of $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$, G is a sufficiently smooth function in all of its arguments, $\mathbb{J} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}$ is the Jacobian matrix of $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)$, and $\mathbb{M}$ is the metric tensor used for controlling mesh concentration (assumed to be symmetric and uniformly positive definite on Ω). Note that (1) is formulated in terms of the inverse coordinate transformation since functionals formulated this way are known to be less likely to produce singular coordinate transformations [6].

A number of meshing functionals have been developed in the past based on error estimates and physical and geometric considerations, e.g., see [13, 16, 18] and references therein. Here, we consider two functionals: Winslow's functional based on variable diffusion [24] and Huang's functional based on mesh uniformity (or equidistribution and alignment combined) [11].

Example 2.1 (generalized Winslow's functional). The first example is a generalization of Winslow's variable diffusion functional [24],

$$I[\boldsymbol{\xi}] = \int_\Omega \operatorname{tr}(\mathbb{J}\mathbb{M}^{-1}\mathbb{J}^T)\, d\boldsymbol{x}, \tag{2}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix and $\mathbb{M}^{-1}$ serves as the diffusion matrix. The functional is coercive and convex and has a unique minimizer [13, Example 6.2.1].

Since our interest is in the MMPDE method as a mesh smoothing scheme (in the Euclidean metric), we consider the morst simple case $\mathbb{M} = I$, for which the functional becomes

$$I[\boldsymbol{\xi}] = \int_\Omega \operatorname{tr}(\mathbb{J}\mathbb{J}^T)\, d\boldsymbol{x}. \tag{3}$$

For this functional, we have

$$G = \operatorname{tr}(\mathbb{J}\mathbb{J}^T), \qquad \frac{\partial G}{\partial \mathbb{J}} = 2\mathbb{J}^T, \qquad \frac{\partial G}{\partial \det(\mathbb{J})} = 0,$$

which are needed for computing the nodal mesh velocities.

Example 2.2 (Huang's functional). The second functional is

$$I[\boldsymbol{\xi}] = \theta \int_\Omega \sqrt{\det(\mathbb{M})}\big(\operatorname{tr}(\mathbb{J}\mathbb{M}^{-1}\mathbb{J}^T)\big)^{\frac{dp}{2}}\, d\boldsymbol{x} + (1-2\theta) d^{\frac{dp}{2}} \int_\Omega \sqrt{\det(\mathbb{M})}\left(\frac{\det(\mathbb{J})}{\sqrt{\det(\mathbb{M})}}\right)^p d\boldsymbol{x}, \tag{4}$$

where $0 \leq \theta \leq 1$ and $p > 0$ are dimensionless parameters. This functional was proposed by Huang [11] based on $\mathbb{M}$-uniformity (requiring the mesh to be uniform in the metric $\mathbb{M}$). The $\mathbb{M}$-uniformity is mathematically equivalent to the so-called alignment (first term) and equidistribution (second term) conditions combined. For $0 < \theta \leq \frac{1}{2}$, $dp \geq 2$, and $p \geq 1$, the functional is coercive and polyconvex and has a minimizer [13, Example 6.2.2].

For the case $\mathbb{M} = I$, (4) becomes

$$I[\boldsymbol{\xi}] = \theta \int_{\Omega} \left(\mathrm{tr}(\mathbb{J}\mathbb{J}^T)\right)^{\frac{dp}{2}} d\boldsymbol{x} + (1-2\theta) d^{\frac{dp}{2}} \int_{\Omega} \det(\mathbb{J})^p \, d\boldsymbol{x}, \tag{5}$$

and derivatives are

$$\begin{aligned} G &= \theta\left(\mathrm{tr}(\mathbb{J}\mathbb{J}^T)\right)^{\frac{dp}{2}} + (1-2\theta) d^{\frac{dp}{2}} \det(\mathbb{J})^p, \\ \frac{\partial G}{\partial \mathbb{J}} &= dp\theta\left(\mathrm{tr}(\mathbb{J}\mathbb{J}^T)\right)^{\frac{dp}{2}-1} \mathbb{J}^T, \\ \frac{\partial G}{\partial \det(\mathbb{J})} &= p(1-2\theta) d^{\frac{dp}{2}} \det(\mathbb{J})^{p-1}. \end{aligned}$$

We note that the first term in (5) associated with alignment with $\mathbb{M} = I$ represents a shape regularity measure while the second term associated with equidistribution represents a volume uniformity measure (when $p > 1$). If $\theta = \frac{1}{2}$ and $dp = 2$ then the functional reduces to the Winslow's functional (3).

The MMPDE is defined as the gradient flow equation of $I[\boldsymbol{\xi}]$, i.e.,

$$\frac{\partial \boldsymbol{\xi}}{\partial t} = -\frac{P}{\tau} \frac{\delta I}{\delta \boldsymbol{\xi}},$$

where $\frac{\delta I}{\delta \boldsymbol{\xi}}$ is the functional derivative of I, τ is a positive constant used for adjusting the time scale of mesh movement, and $P = P(\boldsymbol{x}, t)$ is a positive function used for preserving certain scaling invariances; in our computation we choose $P = 1$. For the functional (1), this becomes

$$\frac{\partial \boldsymbol{\xi}}{\partial t} = \frac{P}{\tau} \nabla \cdot \left(\frac{\partial G}{\partial \mathbb{J}} + \frac{\partial G}{\partial \det(\mathbb{J})} \det(\mathbb{J}) \mathbb{J}^{-1} \right). \tag{6}$$

Using the identity

$$\frac{\partial \boldsymbol{x}}{\partial t} = -\mathbb{J}^{-1} \frac{\partial \boldsymbol{\xi}}{\partial t},$$

we can rewrite the MMPDE into

$$\frac{\partial \boldsymbol{x}}{\partial t} = -\frac{P\mathbb{J}^{-1}}{\tau} \nabla \cdot \left(\frac{\partial G}{\partial \mathbb{J}} + \frac{\partial G}{\partial \det(\mathbb{J})} \det(\mathbb{J}) \mathbb{J}^{-1} \right). \tag{7}$$

After exchanging the roles of dependent and independent variables $\boldsymbol{\xi}$ and $\boldsymbol{x}$ on the right-hand side, the equation can be discretized on a computational mesh of Ω_c and a set of mesh equations for the nodal mesh velocities can be obtained. Then, the mesh equations can be integrated with proper boundary conditions for the vertex locations of the underlying moving mesh.

3 The MMPDE mesh smoothing scheme

A simple approach was proposed in [12] for the implementation of variational mesh generation and adaptation. This approach can also be used for implementing the MMPDE method described in the previous section. More specifically, we denote by $\mathcal{T}_h$ and $\mathcal{T}_{h,c}$ simplcial meshes on Ω and Ω_c, respectively, and assume that they have the same numbers of elements and vertices and the same connectivity. We also assume that Ω_c has been chosen to be almost uniform in the sense that all of its elements have almost the same size and are almost equilateral. We will see below that Ω_c and $\mathcal{T}_{h,c}$ are used only as an intermediate step and do not appear in the final formulation.

With the approach, the functional (1) is first approximated on $\mathcal{T}_h$ using a midpoint quadrature rule with $\mathbb{J}$ being approximated by the Jacobian matrix of the affine mapping between elements in $\mathcal{T}_h$ and their counterparts in $\mathcal{T}_{h,c}$. Notice that the Jacobian matrix can be computed using the edge matrices. The discretized functional is a function of the locations of the vertices of $\mathcal{T}_h$. By assumption, $\mathcal{T}_{h,c}$ is known and so are the locations of its vertices and thus the discretized functional is a function of the locations of the vertices of $\mathcal{T}_h$ only. The mesh equation for the locations of the vertices of $\mathcal{T}_h$ is then obtained as the gradient equation of the discretized functional with respect to the locations. The derivatives of the discretized functional with respect to the locations can be expressed in a simple, analytical matrix form; the interested reader is referred to [12] for details on the derivation. It is worth mentioning a significant simplification in the derivation. Notice that the functionals (2) and (4) are invariant under translations and rotations of the computational coordinate $\boldsymbol{\xi}$. This implies that the elements in $\mathcal{T}_{h,c}$, which are assumed to be almost equilateral, can be made to be similar to the master element $\hat{K}$ by translations and rotations. Moreover, they can be made almost identical to $\frac{1}{N}\hat{K}$, where N is the number of the elements in the mesh, since they are also assumed to have almost the same size. Thus, as long as the functional (1) is invariant under translations and rotation of $\boldsymbol{\xi}$, the computational elements appearing in the final formulation of the mesh equation can be replaced by $\frac{1}{N}\hat{K}$ where $\hat{K}$ is the master element assumed to be chosen as a regular simplex with the unitary volume.

Denote the locations of the vertices of $\mathcal{T}_h$ by $\boldsymbol{x}_i$, $i = 1, \ldots, N_v$, the element patch associated with vertex $\boldsymbol{x}_i$ by ω_i, the generic element in $\mathcal{T}_h$ by K, and the volume of K by $|K|$. Then the mesh equation for these locations is given by

$$\frac{d\boldsymbol{x}_i}{dt} = \frac{1}{\tau}\sum_{K\in\omega_i} |K|\boldsymbol{v}_{i_K}^K, \quad i = 1, \ldots, N_v \tag{8}$$

where i_K is the local index of vertex $\boldsymbol{x}_i$ on K and the local mesh velocities are given by

$$\begin{bmatrix} (\boldsymbol{v}_1^K)^T \\ \vdots \\ (\boldsymbol{v}_d^K)^T \end{bmatrix} = -GE_K^{-1} + E_K^{-1}\frac{\partial G}{\partial \mathbb{J}}\hat{E}E_K^{-1} + \frac{\partial G}{\partial \det(\mathbb{J})}\frac{\det(\hat{E})}{\det(E_K)}E_K^{-1}, \tag{9}$$

$$(\boldsymbol{v}_0^K)^T = -\sum_{j=1}^{d}(\boldsymbol{v}_j^K)^T. \tag{10}$$

In the above equation, $\hat{E}$ and E_K are the edge matrices of $\hat{K}$ and K and G, $\frac{\partial G}{\partial \mathbb{J}}$, and $\frac{\partial G}{\partial \det(\mathbb{J})}$ are evaluated at $\mathbb{J} = \hat{E}E_K^{-1}$ and $\det(\mathbb{J}) = \det(\hat{E})/\det(E_K)$.

The mesh velocities need to be modified for boundary vertices. For example, if $\boldsymbol{x}_i$ is a fixed boundary vertex, we can replace the corresponding equation by

$$\frac{\partial \boldsymbol{x}_i}{\partial t} = 0.$$

When $\boldsymbol{x}_i$ is allowed to move on a boundary curve (in 2D) or surface (in 3D) represented by

$$\phi(\boldsymbol{x}) = 0, \tag{11}$$

then the mesh velocity $\frac{\partial \boldsymbol{x}_i}{\partial t}$ needs to be modified such that its normal component along the curve or surface is zero, i.e.,

$$\nabla \phi(\boldsymbol{x}_i) \cdot \frac{\partial \boldsymbol{x}_i}{\partial t} = 0. \tag{12}$$

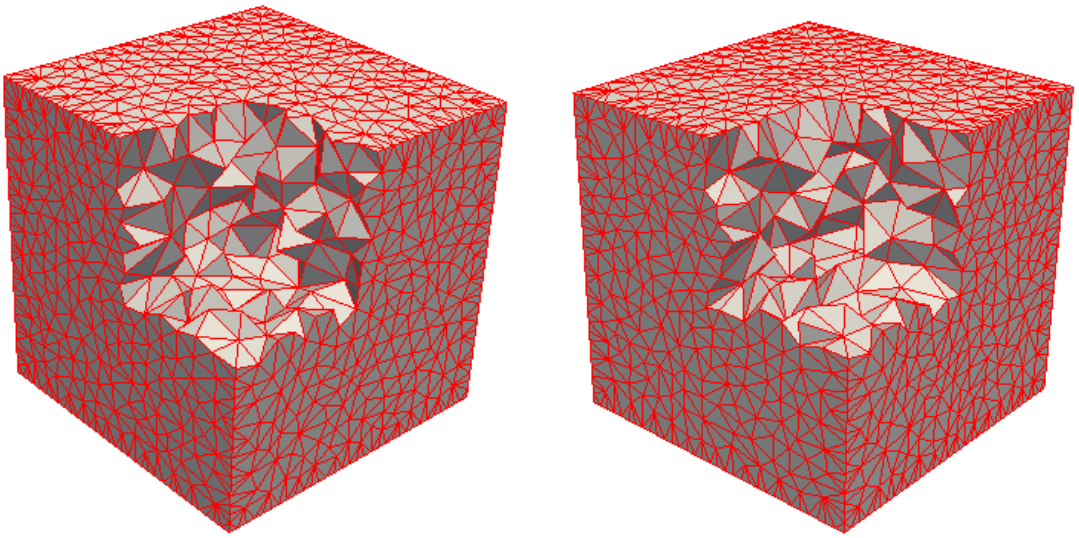
The mesh equation (8) can be integrated using explicit or implicit ODE solvers. It can also be first discretized in time and the linear system is solved globally (such as a Krylov-subspace method with preconditioning) or locally (such as Gauss-Seidel or Jacobi iteration). We can also just solve the balance equations (i.e., set velocities be zeros) using global or local iterative methods.

4 Numerical examples

In the following we provide examples for MMPDE-based smoothing to improve the mesh quality of the tetrahedral meses obtained by *TetGen* [23]: a simple cube mesh (Fig. 1), an L-shape domain (Fig. 2) and *cami1a* part (Fig. 3). For our computation, we use the Huang's functional (Example 2.2 with $\theta = 1/3$ and $p = 2$), which provides a better control of the mesh element sizing (equidistribution) than the Winslow's functional.

For each of the examples we compare the dihedral angle statistics of the original *TetGen* mesh with the dihedral angle statistics after a mesh smoothing iteration. For the first example we also provide both cases: using fixed boundary (11) and allowing the boundary nodes to move along the corresponding surfaces (12) (denoted as BM in Fig. 1b); in the other two examples we always allow the boundary nodes to move along the corresponding boundary surfaces.

In all three examples, the proposed smoothing step significantly reduces the number of small (0°–20°) as well as large (150°–180°) dihedral angles. These first results look quite promising. However, more work needs to be done to investigate the potential of the MMPDE approach for the mesh smoothing. Especially the combination of mesh smoothing with a reconnection step can provide a further improvement to the mesh quality.

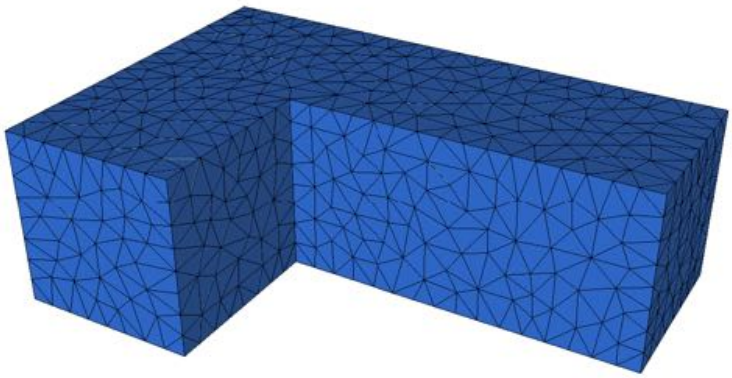


(a) meshes before (left) and after (right) smoothing

(b) statistics of dihedral angles before and after smoothing

angle	before	after	after (BM)	angle	before	after	after (BM)
0 – 5	0	0	0	80 – 110	15 289	14 870	15 065
5 – 10	43	2	0	110 – 120	2 211	2 712	2 811
10 – 20	937	193	50	120 – 130	1 369	2 132	2 219
20 – 30	2 246	1 964	1 731	130 – 140	921	1 279	1 137
30 – 40	4 862	6 606	6 622	140 – 150	567	294	101
40 – 50	8 277	9 333	9 512	150 – 160	346	33	1
50 – 60	10 480	9 905	1168	160 – 170	123	3	0
60 – 70	9 974	9 052	8 991	170 – 175	0	0	0
70 – 80	8 349	7 616	7 586	175 – 180	0	0	0

Figure 1: Cube example

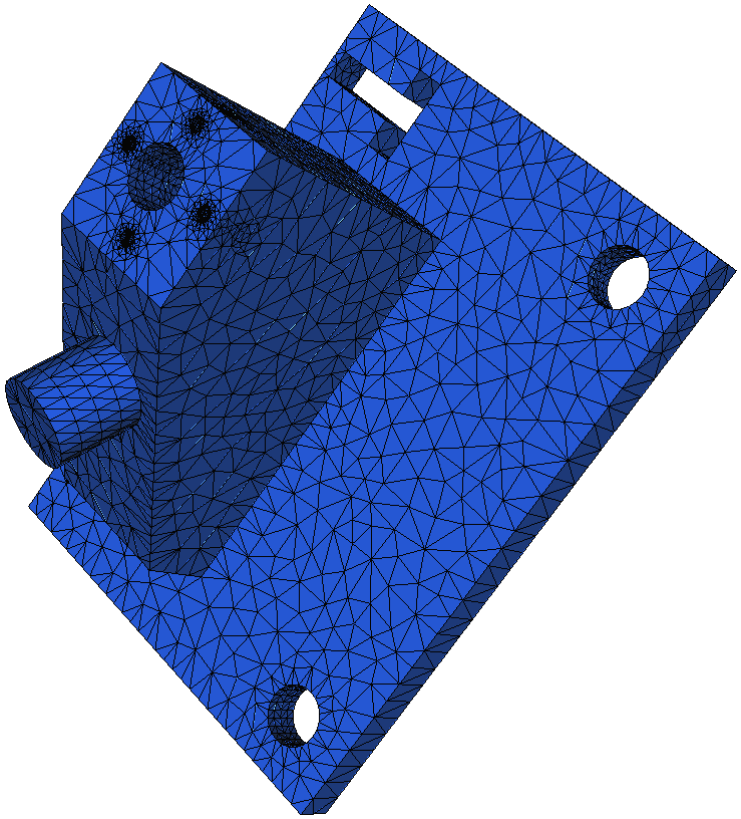


(a) mesh example

(b) statistics of dihedral angles before and after smoothing

angle	before	after	angle	before	after
0 – 5	0	0	80 – 110	5 470	5 421
5 – 10	22	0	110 – 120	807	823
10 – 20	398	51	120 – 130	444	608
20 – 30	773	903	130 – 140	273	476
30 – 40	1 652	2 017	140 – 150	192	152
40 – 50	2 995	3 143	150 – 160	163	3
50 – 60	3 649	3 547	160 – 170	44	0
60 – 70	3 404	3 291	170 – 175	0	0
70 – 80	3 006	2 857	175 – 180	0	0

Figure 2: L-shape example, 3 882 elements



(a) mesh example

(b) statistics of dihedral angles before and after smoothing

angle	before	after	angle	before	after
0 – 5	1	0	80 – 110	30 060	30 087
5 – 10	335	15	110 – 120	5 417	5 607
10 – 20	4 086	3 617	120 – 130	4 329	5 026
20 – 30	8 083	8 701	130 – 140	2 895	3 830
30 – 40	12 172	13 704	140 – 150	1 676	1 739
40 – 50	15 338	15 389	150 – 160	1 033	458
50 – 60	17 175	16 740	160 – 170	307	3
60 – 70	15 208	14 851	170 – 175	0	0
70 – 80	15 703	14 051	175 – 180	0	0

Figure 3: *cami1a* example, 22 303 elements

References

[1] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Trans. Graphics*, 24:617–625, 2005.

[2] R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations. Users' Guide 7.0*. Frontiers Appl. Math. 15. SIAM, 1994.

[3] S. A. Canann, J. R. Tristano, and M. L. Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. In *Proceedings, 7th International Meshing Roundtable*, Sandia National Laboratories, Albuquerque, NM, 1998.

[4] L. Chen and J.-c. Xu. Optimal Delaunay triangulations. *J. Comput. Math.*, 22:299–308, 2004. Special issue dedicated to the 70th birthday of Professor Zhong-Ci Shi.

[5] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41:637–676, 1999.

[6] A. S. Dvinsky. Adaptive grid generation from harmonic maps on Riemannian manifolds. *J. Comput. Phys.*, 95:450–476, 1991.

[7] D. A. Field. Laplacian smoothing and Delaunay triangulations. *Comm. Appl. Num. Meth.*, 4:709–712, 1988.

[8] L. Freitag, M. Jones, and P. Plassmann. A parallel algorithm for mesh smoothing. *SIAM J. Sci. Comput.*, 20, 1999.

[9] L. A. Freitag and P. M. Knupp. Tetrahedral mesh improvement via optimization of the element condition number. *Internat. J. Numer. Methods Engrg.*, 53:1377–1391, 2002.

[10] L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, 1997.

[11] W. Huang. Variational mesh adaptation: isotropy and equidistribution. *J. Comput. Phys.*, 174:903–924, 2001.

[12] W. Huang and L. Kamenski. A geometric discretization and a simple implementation for variational mesh generation and adaptation. *(submitted)*, 2014. (arXiv:1410.7872).

[13] W. Huang and R. D. Russell. *Adaptive Moving Mesh Methods*. Springer, New York, 2011. Applied Mathematical Sciences Series, Vol. 174.

[14] B. M. Klinger and J. R. Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, Seattle, Waschington, USA, October 2007. Sandia National Laboratories.

[15] P. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I – A framework for surface mesh optimization. *Int. J. Numer. Meth. Engng.*, 48:401–420, 2000.

[16] P. Knupp and S. Steinberg. *Fundamentals of Grid Generation*. CRC Press, Boca Raton, 1994.

[17] P. M. Knupp. Applications of mesh smoothing: copy, morph, and sweep on unstructured quadrilateral meshes. *Internat. J. Numer. Methods Engrg.*, 45:37–45, 1999.

[18] V. D. Liseikin. *Grid Generation Methods*. Springer, Berlin, 1999.

[19] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal Voronoi tessellation – energy smoothness and fast computation. *ACM Trans. Graphics*, 28:101:1–101:17, 2009.

[20] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28:129–137, 1982.

[21] S. H. Lo. A new mesh generation scheme for arbitaar planar domains. *Int. J. Numer. Meth. Engng.*, 21:1403–1426, 1985.

[22] M. Shephard and M. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Int. J. Numer. Meth. Engrg.*, 32:709–749, 1991.

[23] H. Si. `http://tetgen.org`.

[24] A. M. Winslow. Adaptive mesh zoning by the equipotential method. Technical Report UCID-19062, Lawrence Livemore Laboratory, 1981 (unpublished).