Hessian-free metric-based mesh adaptation via geometry of interpolation error

Abdellatif Agouzal[1], Konstantin Lipnikov[2], and Yuri Vassilevski[3]

[1] Universite de Lyon 1, Laboratoire d'Analyse Numerique, agouzal@univ-lyon1.fr
[2] Los Alamos National Laboratory, Theoretical Division, lipnikov@lanl.gov
[3] Institute of Numerical Mathematics, vasilevs@dodo.inm.ras.ru

1 Introduction

Generation of meshes adapted to a given function u requires a specially designed metric. For metric derived from the Hessian of u, optimal error estimates for the interpolation error on simplicial meshes have been proved in [2, 5, 8, 10, 11]. The Hessian-based metric has been successfully applied to adaptive solution of PDEs [4, 7, 9]. However, theoretical estimates have required to make an additional assumption that the discrete Hessian approximates the continuous one in the maximum norm. Despite the fact that this assumption is frequently violated in many Hessian recovery methods, the generated adaptive meshes still result in optimal error reduction.

In this article we continue the rigorous analysis [1, 3] of an alternative way for generating a space tensor metric using the error estimates prescribed to *mesh edges*. The new methodology produces meshes resulting in the optimal reduction of the P_1-interpolation error or its gradient. We define a tensor metric $\mathfrak{M}$ such that the volume and the perimeter of a simplex measured in this metric control the norm of error or its gradient. The equidistribution principle, which can be traced back to D'Azevedo [6], suggests to balance $\mathfrak{M}$-volumes and $\mathfrak{M}$-perimeters. This leads to meshes that are quasi-uniform in the metric $\mathfrak{M}$.

The paper outline is as follows. In Section 2, we derive appropriate metrics from analysis of the interpolation errors. In Section 3, we present the algorithm for generating adaptive meshes and its application to a model problem.

Acknowledgments. Research of the third author has been supported partially by the RFBR project 08-01-00159-a.

2 Metric derivation from local error analysis

Let $\Omega \subset \Re^d$ be a bounded polyhedral domain and Ω_h be a conformal simplicial mesh with N_h simplexes. Let $\mathfrak{M}$ be a piecewise constant tensor metric on Ω_h.

The volume of simplex Δ and the total length of its edges in this metric are denoted by $|\Delta|_{\mathfrak{M}}$ and $|\partial\Delta|_{\mathfrak{M}}$, respectively [2].

Let $\mathcal{I}_1 u$ be the piecewise linear interpolant of u, and $\mathcal{I}_{1,\Delta} u$ be its restriction to Δ. Our goal is to generate meshes that minimize the L_p-norm, $p \in (0, \infty]$, of the interpolation error $e = u - \mathcal{I}_1 u$ or its gradient ∇e.

Let us consider a particular d-simplex Δ with vertices $\mathbf{v}_i$, $i = 1, \ldots, d+1$, edge vectors $\mathbf{e}_k = \mathbf{v}_i - \mathbf{v}_j$, $1 \le i < j \le d+1$, and mid-edge points $\mathbf{c}_k$, $k = 1, \ldots, n_d$, where $n_d = d(d+1)/2$. Let λ_i, $i = 1, \ldots, d+1$, be the linear functions on Δ such that $\lambda_i(\mathbf{v}_j) = \delta_{ij}$ where δ_{ij} is the Kronecker symbol. For every edge $\mathbf{e}_k$, we define the quadratic bubble function $b_k = \lambda_i \lambda_j$.

Step 1. We begin with the derivation of a tensor metric from edge data.

Lemma 1 (Metric existence [1]). *Let α_k, $k = 1, \ldots, n_d$, be values prescribed to edges of a d-simplex Δ such that $\alpha_k \ge 0$ and $\sum_{k=1}^{n_d} \alpha_k > 0$. Then, there exists a constant tensor metric $\mathfrak{M}_\Delta$ such that*

$$\left(\frac{d!}{(d+1)(d+2)}\right)^{1/d} |\Delta|_{\mathfrak{M}_\Delta}^{2/d} \le \sum_{k=1}^{n_d} \alpha_k \le |\partial\Delta|_{\mathfrak{M}_\Delta}^2. \tag{1}$$

Step 2. Let u_2 be a continuous piecewise quadratic function and $e_2 = u_2 - \mathcal{I}_{1,\Delta} u_2$ be the linear interpolation error. We have

$$e_2 = 4 \sum_{k=1}^{n_d} (u_2(\mathbf{c}_k) - \mathcal{I}_{1,\Delta} u_2(\mathbf{c}_k))\, b_k = \sum_{k=1}^{n_d} \gamma_k\, b_k,$$

where $\gamma_k = 4(u_2(\mathbf{c}_k) - \mathcal{I}_1 u_2(\mathbf{c}_k))$. The L_2-norm of the error e_2 is given by

$$\|e_2\|_{L_2(\Delta)}^2 = |\Delta|\, (\mathbb{B}\boldsymbol{\gamma}, \boldsymbol{\gamma}), \tag{2}$$

where $\boldsymbol{\gamma}$ is a vector with n_d components γ_k and $\mathbb{B}$ is the $n_d \times n_d$ symmetric positive definite Gramm matrix with positive entries

$$\mathbb{B}_{k,l} = \frac{1}{|\Delta|} \int_\Delta b_k b_l \, \mathrm{d}V.$$

Note that $\mathbb{B}$ is spectrally equivalent to the identity matrix. Thus,

$$c_1\, |\Delta| \left(\sum_{k=1}^{n_d} |\gamma_k|\right)^2 \le \|e_2\|_{L_2(\Delta)}^2 \le c_2\, |\Delta| \left(\sum_{k=1}^{n_d} |\gamma_k|\right)^2, \tag{3}$$

where the constants $c_2 \ge c_1 > 0$ depend only on the space dimension d.

Analysis of the L_2-norm of ∇e_2 in [1] uses the Cholesky decomposition of $\widetilde{\mathbb{B}} = \mathbb{L}\mathbb{L}^T$, the Gramm matrix for vector-functions ∇b_k, to get:

$$\|\nabla e_2\|_{L_2(\Delta)}^2 = |\Delta| \sum_{k=1}^{n_d} \beta_k^2, \tag{4}$$

where $\boldsymbol{\beta} = \mathbb{L}^T\boldsymbol{\gamma}$. Thus, the L_2-norms of e_2 and ∇e_2 are controlled by a sum of non-negative numbers associated with the edges of simplex Δ times $|\Delta|$.

Using Lemma 1 we build the metric $\mathfrak{M}_\Delta$ for e_2 by setting $\alpha_k = |\gamma_k|$. Similarly, to build the metric $\widetilde{\mathfrak{M}}_\Delta$ for ∇e_2, we set $\alpha_k = \beta_k^2$. In the next step we convert these metrics to optimal metrics for the L_p-norm.

Step 3. The extension of error estimates to general L_p-norms follows the path described in [1]. With a slight modification of the argument used there, we may show that the *optimal metrics* for the L_p-norm of e_2 and ∇e_2 are:

$$\mathfrak{M}_{\Delta,p} = (\det(\mathfrak{M}_\Delta))^{-1/(d+2p)}\,\mathfrak{M}_\Delta \quad \text{and} \quad \widetilde{\mathfrak{M}}_{\Delta,p} = (\det(\widetilde{\mathfrak{M}}_\Delta))^{-1/(d+p)}\widetilde{\mathfrak{M}}_\Delta.$$

For simplicity, we confine ourselves to the case $p = \infty$. In this case, the metrics generated by Lemma 1 are optimal, i.e. $\mathfrak{M}_{\Delta,\infty} = \mathfrak{M}_\Delta$ and $\widetilde{\mathfrak{M}}_{\Delta,\infty} = \widetilde{\mathfrak{M}}_\Delta$.

Step 4. For a given continuous function u, we define a computable error e_2 which will be used to estimate the true error e_Δ:

$$e_2 = \mathcal{I}_{2,\Delta}u - \mathcal{I}_{1,\Delta}u \qquad \text{and} \qquad e_\Delta = u - \mathcal{I}_{1,\Delta}u,$$

where $\mathcal{I}_{2,\Delta}u$ be the piecewise quadratic Lagrange interpolant of u on Δ.

Let $\mathcal{F}$ be the space of symmetric $d \times d$ matrices and $|\mathbb{H}|$ be the spectral module of $\mathbb{H} \in \mathcal{F}$. We introduce the following notations:

$$|||\mathbf{e}_k|||^2_{|\mathbb{H}|} = \max_{\mathbf{x}\in\Delta}\,(|\mathbb{H}(\mathbf{x})|\mathbf{e}_k, \mathbf{e}_k) \qquad \text{and} \qquad |||\partial\Delta|||^2_{|\mathbb{H}|} = \sum_{k=1}^{n_d} |||\mathbf{e}_k|||^2_{|\mathbb{H}|}.$$

Lemma 2 (L_2 error). *Let $u \in C^2(\bar{\Delta})$. Then*

$$\frac{d+1}{2d}\|e_2\|_{L_\infty} \le \|e_\Delta\|_{L_\infty} \le \|e_2\|_{L_\infty} + \frac{1}{4}\inf_{\mathbb{F}\in\mathcal{F}} |||\partial\Delta|||^2_{|\mathbb{H}-\mathbb{F}|}.$$

Lemma 3 (Gradient error [1]). *Let $u \in C^2(\bar{\Delta})$. Then, there exist positive constants c_s, C_s, and $c_4 = c_4(d)$ such that*

$$c_s\|\nabla e_2\|_{L_\infty} - \operatorname{osc}(\mathbb{H},\Delta) \le \|\nabla e_\Delta\|_{L_\infty} \le C_s\|\nabla e_2\|_{L_\infty} + \operatorname{osc}(\mathbb{H},\Delta), \tag{5}$$

where the oscillation term is

$$\operatorname{osc}(\mathbb{H},\Delta) = c_4\frac{|\partial\Delta|^{d-1}}{|\Delta|}\inf_{\mathbb{F}\in\mathcal{F}} |||\partial\Delta|||^2_{|\mathbb{H}-\mathbb{F}|}$$

The oscillation terms are conventional in contemporary error analysis. Their value depend on the simplex and particular features of the function. For instance, if $u \in C^2(\bar{\Delta})$, and Δ is shape regular, then $\operatorname{osc}(\mathbb{H},\Delta) \sim |\partial\Delta|^2$.

3 Metric-based mesh adaptation

We use the Algorithm 1 to build an adaptive mesh minimizing the L_p-norm of error or its gradient. The algorithm is more robust for continuous tensor

Algorithm 1 Adaptive mesh generation

```
Generate an initial mesh Ω_h and compute the metric 𝔐_p.
loop
    Generate a 𝔐_p-quasi-uniform mesh Ω_h.
    Recompute the metric 𝔐_p.
    If Ω_h is 𝔐_p-quasi-uniform, then exit the loop .
end loop
```

metrics that provide faster convergence and result in smoother meshes. We suggest two methods for recovering of a continuous nodal-based metric from the discontinuous piecewise-constant metric $\mathfrak{M}_p$.

For every node $\mathbf{a}_i$ in Ω_h, we define the superelement σ_i as the union of all d-simplices sharing $\mathbf{a}_i$. The first method is based on simple shifting: to every node $\mathbf{a}_i$, we assign the metric with the largest determinant from all metrics available in superelement σ_i. The second method is generalization of the ZZ-recovery method [12]. On every superelement σ_i, we search for a polynomial u_3 containing only cubic and quadratic terms. Let $\mathbb{H}_3$ be the Hessian of u_3. The free parameters are chosen to minimize the functional

$$\sum_{1\le j\le k\le d} \sum_{\Delta\in\sigma_i} \left(\mathbb{H}_{3,jk}(\mathbf{b}_\Delta) - (\mathfrak{M}_\Delta)_{jk}\right)^2,$$

where $\mathbf{b}_\Delta$ is the barycenter of simplex Δ. We set $\mathfrak{M}(\mathbf{a}_i) = \mathbb{H}_3(\mathbf{a}_i)$. To generate a $\mathfrak{M}$-quasi-uniform mesh, we use local mesh modifications described in [2, 4, 10] and implemented in package `Ani2D` (sourceforge.net/projects/ani2D).

In $\Omega = [0,\, 1]^2$ we consider the analytical function proposed in [6]:

$$u(x,\, y) = \frac{(x-0.5)^2 - (\sqrt{10}y + 0.2)^2}{((x-0.5)^2 + (\sqrt{10}y + 0.2)^2)^2}.$$

The function has an anisotropic singularity at point $(0.5,\, -0.2/\sqrt{10})$ located outside the computational domain but close to its boundary. Table 1 shows that the L_∞-norm of the interpolation error is proportional to N_h^{-1}, while the L_∞-norm of its gradient is proportional to $N_h^{-0.5}$. Note that the meshes minimizing the interpolation error and its gradient are different (see Fig. 1).

References

1. Agouzal A, Vassilevski Y (2008) Minimization of gradient errors of piecewise linear interpolation on simplicial meshes. Submitted to SIAM J. Numer. Anal.
2. Agouzal A, Lipnikov K, Vassilevski Y (1999) Adaptive generation of quasi-optimal tetrahedral meshes. *East-West J. Numer. Math.* **7**, 223–244.
3. Agouzal A, Lipnikov K, Vassilevski Y (2007) *Generation of quasi-optimal meshes based on a posteriori error estimates.* Proceedings of 16th International Meshing Roundtable (M.Brewer, D.Marcum; Editors), Springer, 139–148.

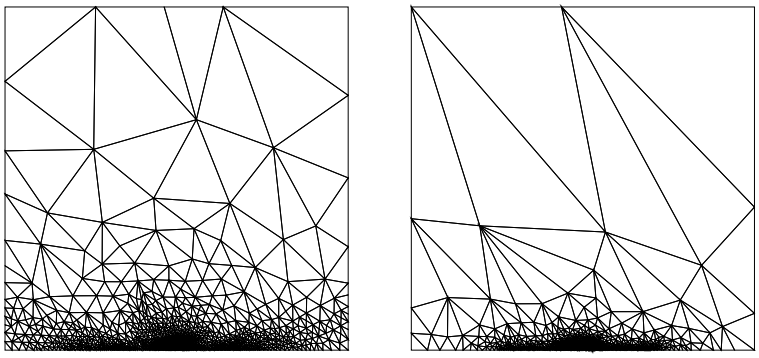


Fig. 1. The adaptive meshes with 2000 triangles minimizing the maximum norm of the interpolation error (left) and its gradient (right).

	Method of shifts		ZZ-type method	
N_h	$\|e\|_{L_\infty(\Omega)}$	$\|\nabla e\|_{L_\infty(\Omega)}$	$\|e\|_{L_\infty(\Omega)}$	$\|\nabla e\|_{L_\infty(\Omega)}$
1000	1.55e-1	6.55e+1	3.74e-1	9.49e+1
4000	4.64e-2	3.16e+1	6.83e-2	4.91e+1
16000	1.14e-2	1.71e+1	2.00e-2	2.79e+1
64000	3.33e-3	8.39e+0	8.14e-3	1.34e+1
rate	0.93	0.49	0.92	0.47

Table 1. Convergence of the interpolation error and its gradient.

4. Buscaglia GC, Dari EA (1997) Anisotropic mesh optimization and its application in adaptivity. *Inter. J. Numer. Meth. Engrg.* **40**, 4119–4136.
5. Chen L, Sun P, Xu J (2007) Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm. *Mathematics of Computation* **76**, 179-204.
6. D'Azevedo E (1991) Optimal triangular mesh generation by coordinate transformation. *SIAM J. Sci. Stat. Comput.* **12**, 755–786.
7. Frey PJ, Alauzet F (2005) Anisotropic mesh adaptation for CFD computations. *Comput. Meth. Appl. Mech. Eng.* **194**, 5068–5082.
8. Huang W (2005) Metric tensors for anisotropic mesh generation. *J. Comp. Physics.* **204**, 633–665.
9. Lipnikov K, Vassilevski Y (2003) Parallel adaptive solution of 3D boundary value problems by Hessian recovery. *Comput. Meth. Appl. Mech. Eng.* **192**, 1495–1513.
10. Vassilevski Y, Lipnikov K (1999) Adaptive algorithm for generation of quasi-optimal meshes. *Comp. Math. Math. Phys.* **39**, 1532–1551.
11. Vassilevski Y, Agouzal A (2005) An unified asymptotic analysis of interpolation errors for optimal meshes. *Doklady Mathematics* **72**, 879–882.
12. Zhu JZ, Zienkiewicz OC (1990) Superconvergence recovery technique and a posteriori error estimators. *Inter. J. Numer. Meth. Engrg.* **30**, 1321–1339.

Quadrilateral Mesh Improvement

Bret D. Anderson[1], Jason F. Shepherd[2], Joel Daniels[3], and Steven E. Benzley[1]

[1] Brigham Young University bret.d.anderson@gmail.com,seb@et.byu.edu
[2] Sandia National Laboratories jfsheph@sandia.gov
[3] Unversity of Utah jdaniels@cs.utah.edu

1 Introduction

In this paper, we build on the mesh improvement methods intended for parallel execution with the paving method [1] and introduce topological changes in an existing quadrilateral-only mesh to improve element shape and nodal valence. This discussion is intended to demonstrate the effectiveness of utilizing mesh cleanup operations as a post-processing step to other mesh generation algorithms. The technique is applied to meshes produced via triangle-to-quadrilateral subdivision and octree-based surface generation methods to exemplify dramatric improvement in mesh quality independent of the original mesh structure and generating method.

2 Mesh Improvement Algorithm

2.1 Mesh Improvement

The purpose of this research is to identify poor topological conditions in quadrilateral meshes and replace specific templates with alternate templates to improve quality and topology and improve mesh structure by targeting high vertex valence. The templates are illustrated in Figure 1. Although most *Doublets* are resolved with a face close operation, doublets that are constrained by curves must be treated with a doublet insertion and edge swap (case 2) or a face open operation (case 3). *Triangular Quadrilaterals* are cleaned up by a doublet insertion and then are treated by case 2 of the *Doublet Removal* operation. Similarly, *Flat Quadrilaterals* are modified with an edge swap and then treated as *Triangular Quadrilaterals*. The *Diamond Collapse* and *Quadrilateral Edge Swap* operations are respectively a face close operation or an edge swap operation performed on specific topologies, always resulting in a more structured mesh. *High Valence Nodes* are resolved with either a face open

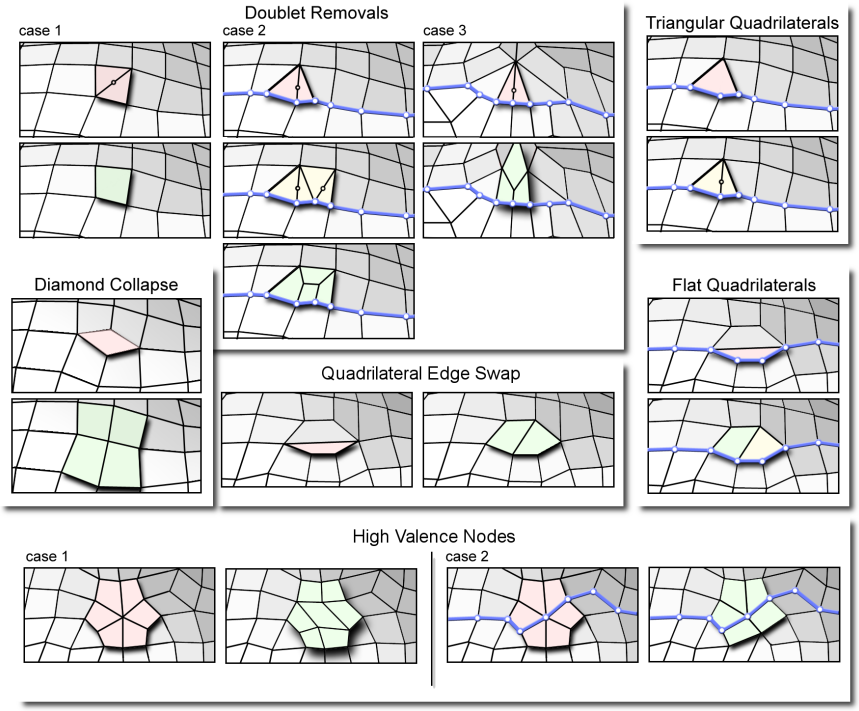


Fig. 1. Quadrilateral improvement operations.

operation (case 1) or a face close operation (case 2). To avoid creating additional problematic topologies, the more simple operations are performed first and the more difficult ones later, in this order: *Diamond Removal, Doublet Removal, Quadrilateral Edge Swap, Triangle Quads, and finally High Valence Nodes.*

2.2 Geometric Constraints and Smoothing

When performing local topological changes to a surface mesh, care must be taken not to alter the geometry of the model. Geometric features are identified a priori and template replacement that results in the removal of geometric features is disallowed. Additionally, to preserve geometric characteristics that are not explicitly defined by curves or vertices, the normal of each pair of adjacent quadrilaterals is calculated. Large differences in the angles of the normal vectors of adjacent quadrilaterals indicate the presence of a geometric feature. If this is the case, the edge and nodes common to the quadrilaterals are marked to be treated as a curve to preserve the feature.

Element quality is considered in addition to local topology, and smoothing is performed after each iteration of the algorithm. An iterative centroid area

scheme is used because it is very straight-forward to implement and requires much less computational effort than many other smoothers [2].

3 Examples

3.1 Plane Mesh Improvement

The first example, illustrated in Figure 2, is a triangle mesh that has been subdivided into quadrilaterals using a Catmull-Clark subdivision and then cleaned up with the method presented in Section 2. As can be seen in Table 1 the number of elements was reduced and the mesh quality was improved significantly, with dramatic improvement in the mesh topology as shown in Figure 2.

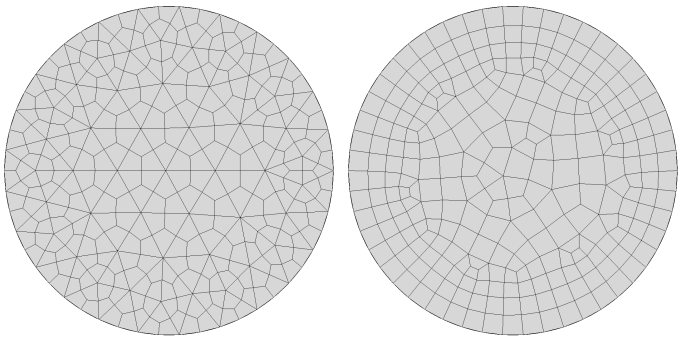


Fig. 2. Before and after quadrilateral improvement on a triangle mesh that has been split into quadrilaterals.

3.2 Surface Mesh Improvement

The triceratops and rabbit examples demonstrate the effectiveness of the algorithm on three-dimensional surface meshes, illustrated in Figure 3. The topology results are illustrated in Figure 3 and the element quality and node valences are further analyzed in Table 2. The cleanup algorithm presented in this paper improves the average scaled Jacobian of the meshes by significantly increasing the number of structured nodes and generating many near square elements.

3.3 Limitations

The methods demonstrated in this paper are straight-forward to implement and can be applied to any quadrilateral mesh. Since the improvement operations tend to increase structure, the likelihood of algorithmic circularity is

Table 1. Before and after comparisons of the number of quadrilateral elements, the average, minimum and maximum scaled Jacobians (SJ), and the valence histograms (number of nodes |N| and % of the total) of mesh shown in Figure 2.

Quality Metric	Circular Disk				
	Before		After		% Change
|Quad|	333		269		−19.2
Ave. SJ	0.804		0.951		18.2
Min. SJ	0.648		0.717		10.7
Max. SJ	0.878		1.000		13.8
Histogram	|N|	%	|N|	%	
Val. 2	0	0.0	0	0.0	
Val. 3	111	30.9	26	8.8	
Val. 4	179	49.9	222	75.3	
Val. 5	13	3.6	47	15.9	
Val. 6	49	13.6	0	0	
Val. 7^+	7	1.9	0	0	

Table 2. Before and after comparisons of the number of quadrilateral elements, the average, minimum and maximum scaled Jacobians (SJ), and the valence histograms (number of nodes |N| and % of the total) of the two meshes shown in Figure 3.

Quality Metric	Rabbit					Triceratops				
	Before		After		% Change	Before		After		% Change
|Quad|	21714		19675		−9.4	19274		17607		−8.6
Ave. SJ	0.904		0.968		7.1	0.903		0.962		6.5
Min. SJ	0.453		0.436		−3.8	−0.500		0.031		106.2
Max. SJ	1.000		1.000		0.0	1.000		1.000		0.0
Histogram	|N|	%	|N|	%		|N|	%	|N|	%	
Val. 2	0	0.0	0	0.0		0	0.0	0	0.0	
Val. 3	5099	23.5	1078	5.5		4190	21.7	880	5.0	
Val. 4	12163	56.0	17530	89.1		11434	59.3	15859	90.1	
Val. 5	3817	17.6	1068	5.4		3122	16.2	868	4.9	
Val. 6	637	2.9	1	0.0		530	2.7	2	0.0	
Val. 7^+	0	0.0	0	0.0		0	0.0	0	0.0	

low. The order of operations can affect the level of results achieved and the heuristic ordering that we have chosen for our algorithm may not be generally appropriate for all mesh generation methods. However, for the examples shown, this recipe has led to pleasing results, especially for the triangle subdivision mesh.

4 Conclusion

In this paper, we introduce topological changes in an existing quadrilateral mesh to improve quadrilateral shape and nodal valence. Our purpose is to

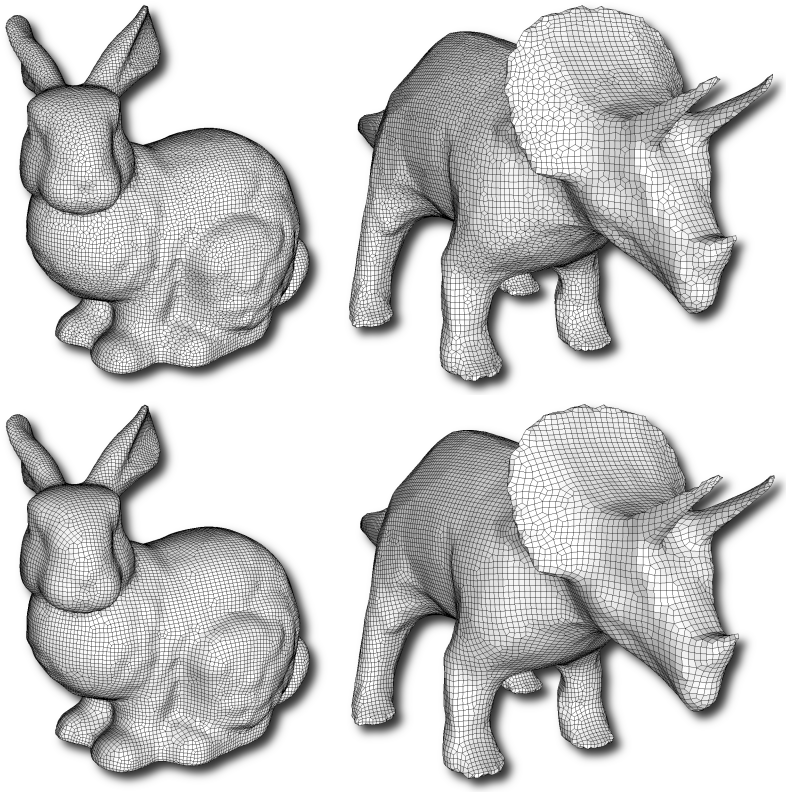


Fig. 3. Before and after quadrilateral improvement on two meshes constructed as grafted quadrilateral surfaces. Despite the generation type and original connectivity, our improvement methods greatly increase the percentage of ideal nodes and element quality.

show the impact of topological improvement operations for all quadrilateral generation methods. We have demonstrated these mesh improvement operations with several different quadrilateral mesh schemes and have shown dramatic improvement in the quality of these meshes.

References

1. Paul Kinney. Cleanup: Improving quadrilateral finite element meshes. In *Proceedings of the 6th International Meshing Roundtable*, pages 449–461, October 1997.
2. Thouis R. Jones, Fredo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, 2003.

Adaptive mesh refinement for the numerical modelling of complex microstructural evolution applications

T. Coupez, L. Silva, M. Bernacki, H. Resk, W. Zerguine

Mines ParisTech, CEMEF- Centre de Mise en Forme des Matériaux, CNRS UMR 7635, BP 207, 1 rue Claude Daunesse 06904 Sophia Antipolis Cedex, France

1. Introduction

The objective of this note is to demonstrate the necessity of using an appropriate automatic adaptive meshing technique so as to handle complex microstructural evolutions. Examples of such microstructural evolutions include the deformation and recrystallization of polycrystalline aggregates or the growth of dendrites occurring during the solidification of metallic alloys. In such applications, it is required to track and/or capture accurately the interface between two or several domains having different mechanical behaviour. Also, the interface is a region where strong physical and/or kinematic incompatibilities, stemming from the heterogeneity of the mechanical response, occur. Gradients of the various mechanical fields develop across the interface and are expected to be significant in the direction perpendicular to the boundary. For these reasons, anisotropic mesh refinement along the interface, with a smaller mesh size in the perpendicular direction to it, is needed. This strategy represents a good compromise between accuracy and computation time as the mesh refinement is performed only where it is necessary. The level set framework used in our approach to represent and follow the boundaries is fully compatible with this meshing technique and allows for the remeshing to be handled quite easily throughout the evolution of the microstruture as often as needed.

After presenting briefly the basis of the anisotropic meshing and remeshing tool, two illustrative applications are presented.

2. Anisotropic meshing and remeshing based on the level set framework

In order to construct an anisotropic mesh, an appropriate anisotropic metric has to be defined. In our approach, the construction of the metric is based on the level set framework which is used for domain representation. The level set is defined as a signed distance function ϕ with a zero level corresponding to the interface Γ:

$$\begin{cases} \phi(x) = d(x,\Gamma), x \in \Omega \\ \Gamma = \{x \in \Omega,\ \phi(x) = 0\} \end{cases} \qquad (1)$$

The direction of mesh refinement is the normal to the interface which is defined by the gradient of the level set function $\vec{n} = \overrightarrow{\nabla\phi}$. Figure 1 illustrates the concept in 2D. Let h_1 be the desired mesh size far from the interface where isotropic mesh is needed and h_2 the desired mesh size in the direction of $\overrightarrow{\nabla\phi}$ in the region close the to the interface where anisotropic mesh is needed. As seen on figure 1, h_1 is also the mesh size in the direction perpendicular to $\overrightarrow{\nabla\phi}$ in the region surrounding the interface.

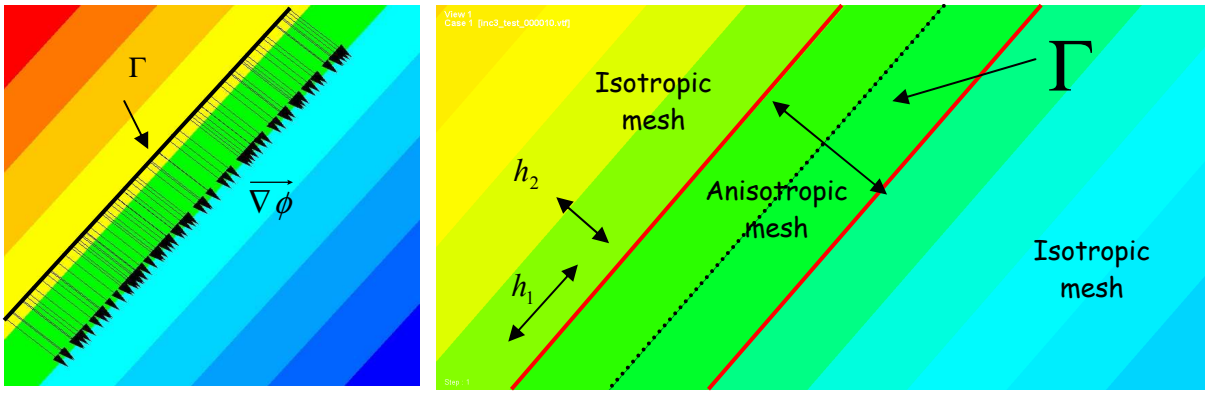


Fig. 1. Left: level set function in 2D with its zero level and the direction of its gradient; right: a close up view of the region near the zero level.

The metric M is defined as follows:

$$M = C\left(\overrightarrow{\nabla\phi} \otimes \overrightarrow{\nabla\phi}\right) + \frac{Id}{h_1^{\,2}} \qquad (2)$$

where Id is the identity tensor, C a scalar parameter which is equal to $\frac{1}{h_2^{\,2}} - \frac{1}{h_1^{\,2}}$ in the anisotropic region and 0 in the isotropic region. One can verify that such common definition for the metric leads to an isotropic mesh far from the interface and an anisotropic mesh near a specified region around the boundary.

The metric M is fed into MTC [1] which builds the corresponding mesh. Most importantly, the level set description is fully compatible with automatic remeshing. The values of the level set function are directly interpolated from the old mesh to the new one and the corresponding anisotropic metric and mesh are reconstructed. Thus, periodic adaptive automatic remeshing can be easily performed, thereby always ensuring appropriate refinement near the interface.

3. Application 1: deformation and subsequent primary recrystallization of polycrystalline aggregates

Recrystallization phenomena inevitably occur during thermal and mechanical processes and have a major impact on the final in-use properties of the polycrystalline metallic materials [2]. The objective of this study is to model the deformation of 3D grain structures subjected to important strains in an attempt to provide reliable quantitative data to be used for the modelling of the recrystallization.

3D polycrystalline volumes are constructed using random Voronoï tesselation or based on voxelized experimental data. When dealing with a polycrystalline aggregate, an individual level set function is used for each grain: $\{\phi_i, 1 \le i \le N_G\}$, with NG the total number of grains in the aggregate. One can also define a global level set function as $\phi_{glob} = \max\{\phi_i, 1 \le i \le N_G\}$ with a zero value corresponding to the interface between all grains. The obtained digital microstructures are meshed anisotropically and adaptively, with refinement close to the grain boundaries as seen on figure 2. Digital mechanical testing then takes place using classical crystal plasticity constitutive law [3] coupled to a non-linear finite element solver in an updated Lagrangian framework. In this context, the level set function of each grain moves according to the material velocity, which corresponds here to the mesh velocity. The large deformation of the aggregate is responsible for strong mesh distortion as seen on figure 2, hence the need for frequent remeshing.

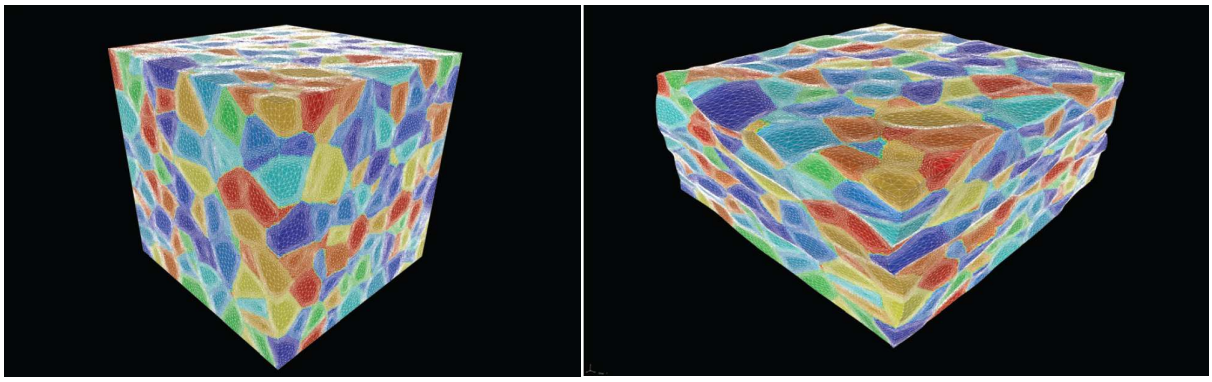


Fig. 2. 1000 grains polycrystal. Left: Anisotropic mesh before deformation (2 125 688 nodes, 12 385 889 elements) ; right : mesh after 70% thickness reduction

The deformation simulation provides an estimate of the spatial distribution of strain energy within the polycrystalline aggregate. The latter quantity is used as input for modelling subsequent primary recrystallization as seen on figure 3. As in the deformation stage, level set functions are used to represent grains during the recrystallization stage. The microstructure obtained at the end of the deformation step is used as input to the recrystallization simulation. New nuclei are introduced in the simulation by creating new level set functions based on the gradient of the stored energy.

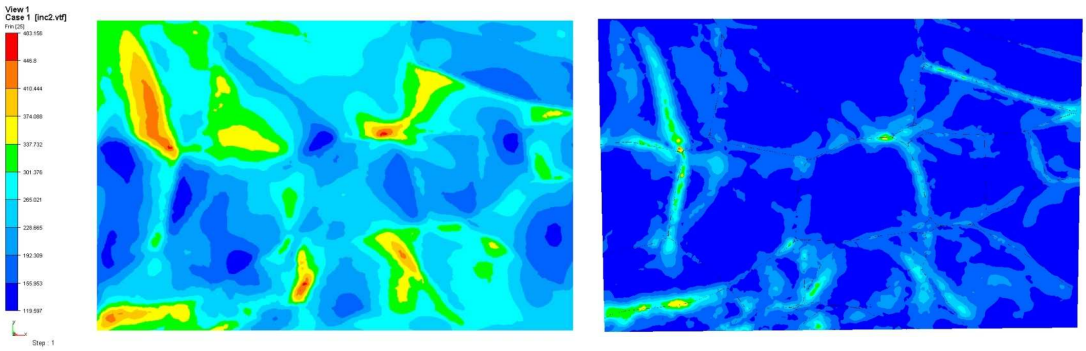


Fig. 3. Left: stored energy at the grains interface after deformation; right: the gradient of the stored energy

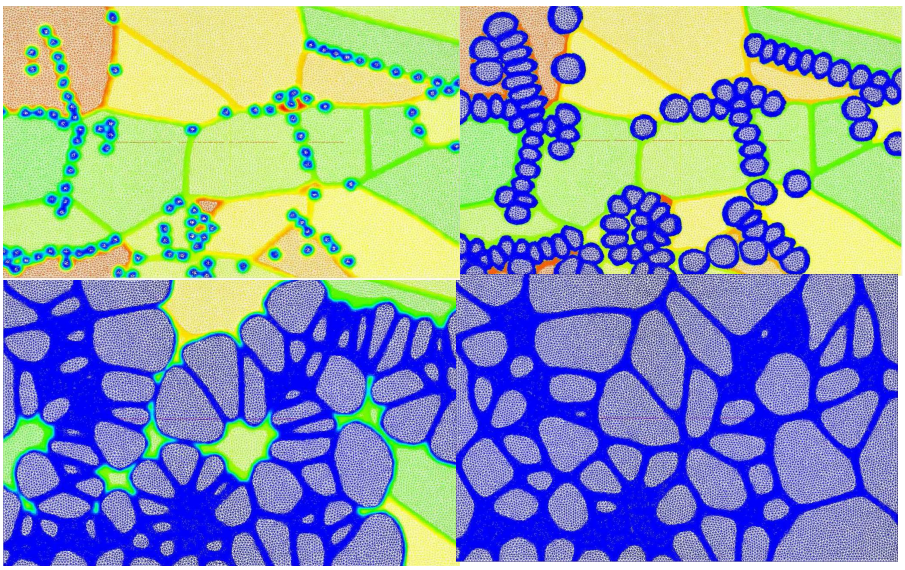


Fig. 4. From left to right and top to bottom: evolution of the recrystallized part in blue

As opposed to the deformation step, the microstructure evolution during the recrystallization stage is perfomed in Eulerian context. Thus, the level set function of each grain is convected throughout the mesh. In order to correctly evolve the fronts, several issues need to be addressed. The main problem is the development of vacuum and/or overlapping regions. A highly refined mesh is needed near the interfaces of the grains so as to ensure a good accuracy in the computation of the interface velocities. The adaptive meshing technique described previously has proved to be an efficient tool in achieving this objective [4]. Frequent remeshing is needed as the microstructure evolves as seen on figure 4.

4. Application 2: dendritic growth

Dendritic growth occurs during complex heat/solute diffusion processes associated to curvature effects and all at their own length and time scale. First numerical results concerning the simulation of dendritic growth of a single dendrite in two dimensions is presented in this section. Material parameters and geometry were taken from [5]. The initial solid grain is a circle with an initial concentration corresponding to the equilibrium liquid concentration of the phase diagram and is inserted in a square domain. As with recrystallization simulation, the level set framework is used in an eulerian context for the evolution of the microstructure. Frequent remeshing is needed and for the case presented here the frequency of remeshing was 20, for several thousands of time steps. Figure 5 illustrates the evolution of the different computed variables during dendritic growth. One can observe a logical anisotropic growth, as well as the solute's migration between phases and the different thermal and solutal diffusion layers. Even though these first results are promising, dendrite final shape is not quite well achieved: due to curvature effects, dendrite's tip should be rounded and instabilities occuring at the interface should vanish with time. If we look at the mesh during growth (figure 6) we can see that although a relatively fine adaptation near the interface was performed, it may not be sufficient to capture the anisotropic curvature correctly. Moreover, we may also need to enlarge the adaptation to the whole solutal diffusion layer (due to the abrupt concentration's pattern in this zone). Perspectives of this work include the use of a posteriori error estimators both on level set and concentration solutions to better guide the meshing procedures.

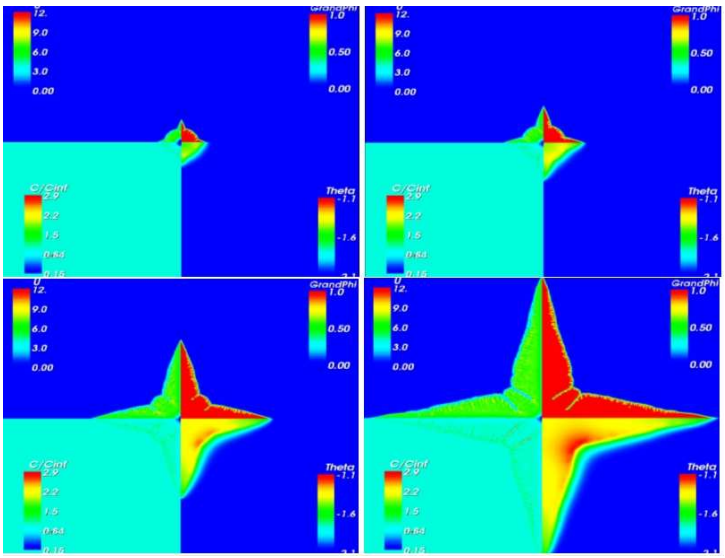


Fig. 5. Growth of one dendrite. From left to right and from top to bottom: representation of the adimensional concentration, of a normalized [0,1] function of the level set, of the real concentration and of the adimensional temperature

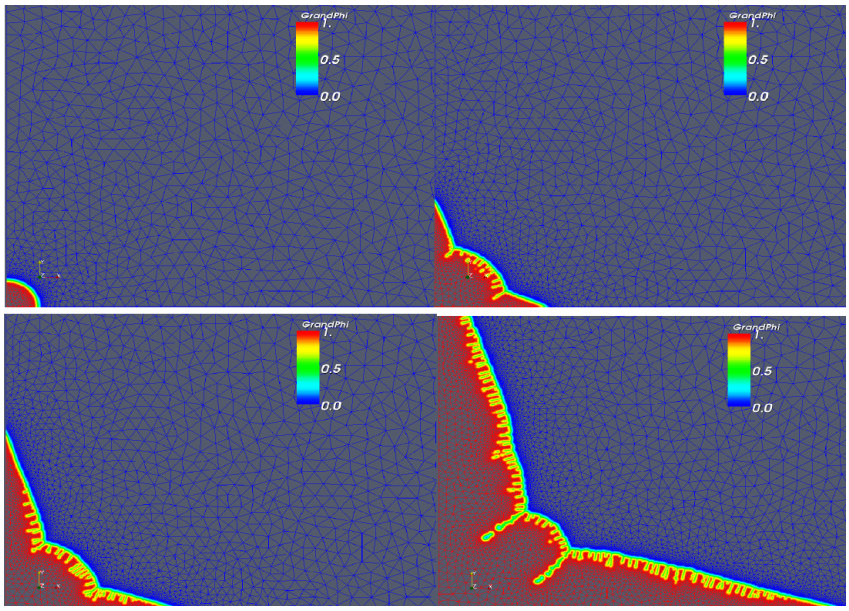


Fig. 6. Mesh evolution during dendritic growth showing anisotropic refinement near the moving front

5. Conclusion

It has been shown that an adaptive anisotropic automatic meshing and remeshing technique coupled to a level set framework is a promising tool to describe complex phenomena such as the deformation and subsequent primary recrystallization in polycrystalline materials and dendritic growth during solidification of metallic alloys. Indeed, in the case of polycrystalline aggregates, large plastic deformation is achieved on a representative volume element of 1000 grains and for primary recristallization modelling, full kinematic compatibilities are enforced, and nucleation and growth phenomena are implemented. Perspectives for recrystallization modelling include simulations with a statistical number of grains, the prediction of recrystallization kinetics and the comparison with other models. Concerning dendritic growth, perspectives include a better guidance of the adaptive meshing procedure using error estimators. Furthermore, a local level set method can also be adopted, to avoid advection of the function over the whole computational domain.

It should be noted that this research note presents a first promising step in the simulation of complex phenomena at the microscale. The adaptative meshing technique presented here seems to be an unavoidable tool if we wish to simulate microstructural evolutions at even smaller scales.

References

[1] Coupez T., Digonnet H., Ducloux R. (2000) Math. Modelling 25:153-175
[2] Humphreys F.J., Haterly M. (1995) Recrystallization and related annealing phenomena. Oxford: Pergamon Press
[3] Delannay L., Jacques P.J., Kalidindi S.R., (2006) Int. J. Plasticity 22:1879-1898
[4] Bernacki M., Chastel Y., Coupez T., Logé R.E. (2008) Scripta Mat. 58(12):1129-1132
[5] Ramirez J.C, Beckermann C. (2004) Phys. Rev. E, 69
[6] Kugler G., Turk R. (2004) Acta Mat. 52:4659-4668
[7] Gibou F., Fedkiw R., Caflisch R., Osher, S. J. Sci. Comput. 19(1-3):183–199
[8] Sarma & al, Comp. Mat. Sci, 1998, 12, 105-123.
[9] Narski J., Picasso M. (2006) FDMP, 1(1):1-13

A New Technique for Quad-Dominant Adaptive Mesh Generation

Mohamed Ebeida[1], Roger Davis[2], and Roland Freund[3]

[1] University of California, Davis msebeida@ucdavis.edu
[2] University of California, Davis davisrl@ucdavis.edu
[3] University of California, Davis freund@math.ucdavis.edu

A new technique for adaptive grid generation is presented. This technique is fast and based on quad-tree decomposition. The resulting mesh is quad-dominant and ready for the application of finite element and multigrid methods. The lower bound of the interior angles of any element is set to be 45° except at those parts of the boundaries where the geometry is discontinuous. Narrow regions of the domain are automatically detected and represented efficiently. An algorithm for mesh modification in the boundary layer region is presented. Several application examples are provided to emphasize the main features of this new approach.

1 Introduction

Grid generation is a first step in the solution of many computational problems, especially those arising in scientific computing and computer graphics. For example, the accuracy of the numerical solution of partial differential equations depends on the quality of the grid used in the discretization of the problem.

Over the last two decades, adaptive methods have stirred much interest in the engineering community. For compressible flow applications, such methods are crucial because of the pressing need for accurate computation of flows with variable density and/or shock waves. Adaptive methods offer a means of tackling complex flow problems at a reasonable cost and of controlling the accuracy of numerical simulations.

The purpose of this paper is to describe a high-quality mesh generation technique for planar domains of arbitrary shape based on quad-tree decomposition. Our goal for this new technique is to achieve the following requirements:

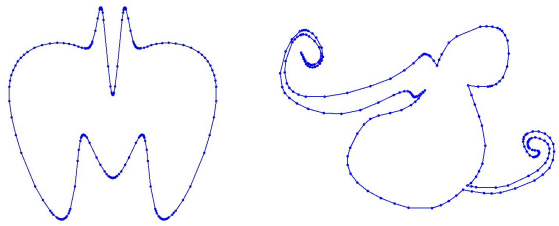
- Fast and adaptive.
- Capable of automatic detection and efficient representation of narrow and slope-discontinuous regions in the domain.
- Efficient representation of the boundary layer region.

- Ready for the application of finite element methods.
- Ready for the application of multigrid and line-solver methods.
- Most of the domain is covered by squares.
- Minimum angle in any element should be greater than or equal to 45° with the exception of those elements with a node where the boundary of the domain is slope-discontinous.

Our mesh generation technique can be summarized in the following steps:

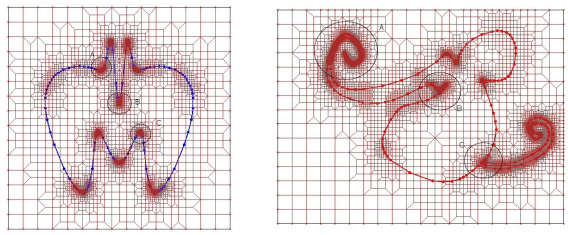
1. Linear representation of the domain boundaries based on its curvature:
   - Each closed boundary curve is represented initially with a two-line segment ploryline.
   - The number of the line segements in each polyline increases through an iteraive adaptive refinement procedure.
   - A parameter, ϵ, is responsible for the termination of the iterative refinement procedure which stops when all the angles, α_i, of the closed polyline satisfy $180^\circ - \epsilon < \alpha_i < 180^\circ + \epsilon$
2. Refinement of a background Cartesian mesh based on quad-tree decomposition and distribution of boundary edges:
   - The size of the near boundary elements is determined based on the length of the nearest boundary edge.
   - Narrow domain regions are detected and refined.
3. Detection of the points out of the domain.
   - Standard Delaunay triangulation is used to cover each closed curve with triangles.
   - Exterior elements are then removed.
4. Optimization of near-boundary elements:
   - Buffer zone creation around each shape and removal of external elements.
   - Covering the buffer zone between the terminal elements and the boundaries of the domain.
   - Modification of the boundary layer elements to enable efficient capturing of the velocity variables for viscous flows.
5. Mesh adaption based on some error function during numerical simulation.

The output of these steps is demonstrated using the following figures. Figure 1 shows the linear representation of two different domain boundaries; a smooth one with varying curvature and another one with discontinuous-slope and narrow regions. Figure 2 shows the spatial decomposition around the two types of boundaries (varying curvature and narrow region). The result from step 3 in which the domain is covered with triangles is shown in Fig. 3. The optimization step and the modification of the elements in the boundary layer region is demonstrated using Figs. 4 and 5. Finally the solution based adaptation and the multigrid levels are demonstrated in Figs. 6 and 7 for unsteady flow over two vertical cylinders..

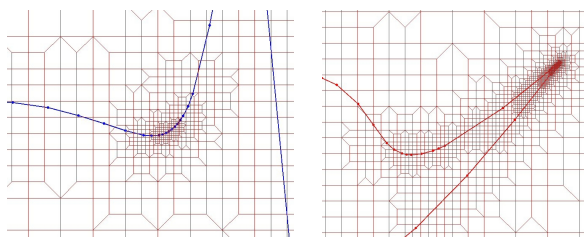


(a) $\epsilon = 10°$ (b) $\epsilon = 30°$

Fig. 1. Linear representation of the domain boundaries

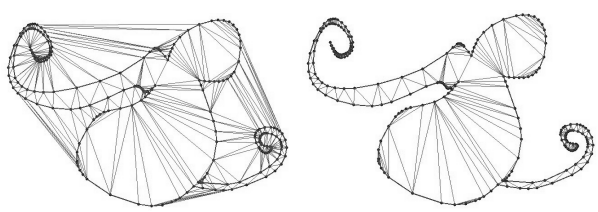


(a) Domain with boundaries of varying curvature (b) Domain with narrow regions



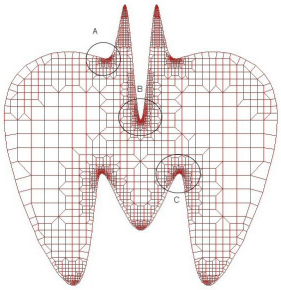
(c) Zoom A (d) Zoom C

Fig. 2. Spatial decomposition based on quad-tree and boundary edge distribution

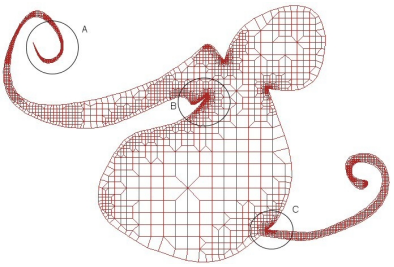


(a) Output of Standard Delaunay algorithm (b) External Elements Removed

Fig. 3. Covering the domain with triangles

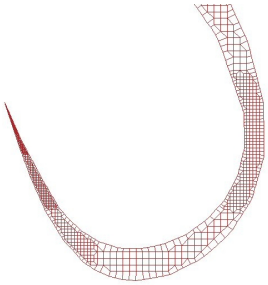


(a) Domain with boundaries of varying curvature



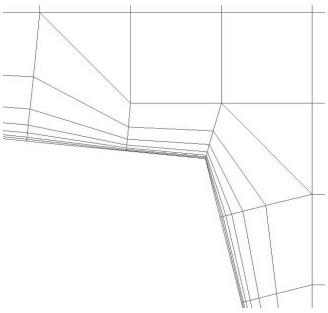
(b) Domain with narrow regions

(c) Zoom A

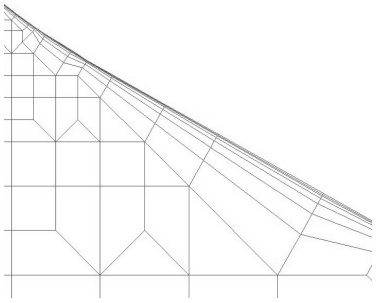


(d) Zoom A

Fig. 4. Optimization Step

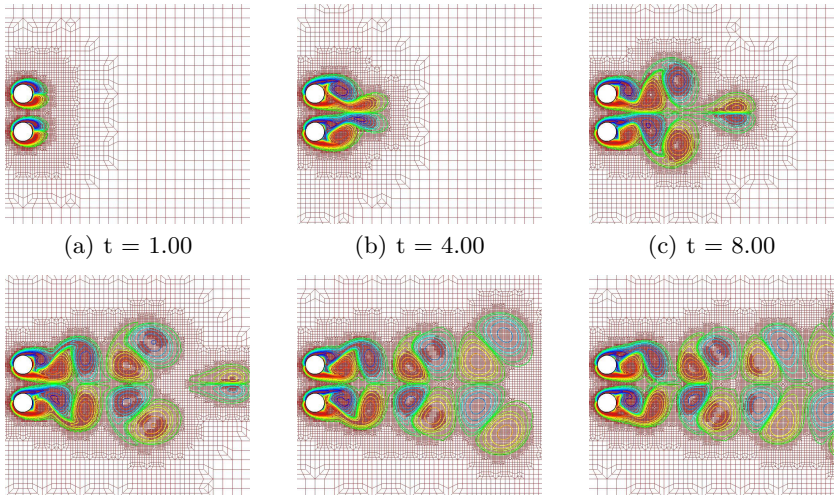


(a)



(b)

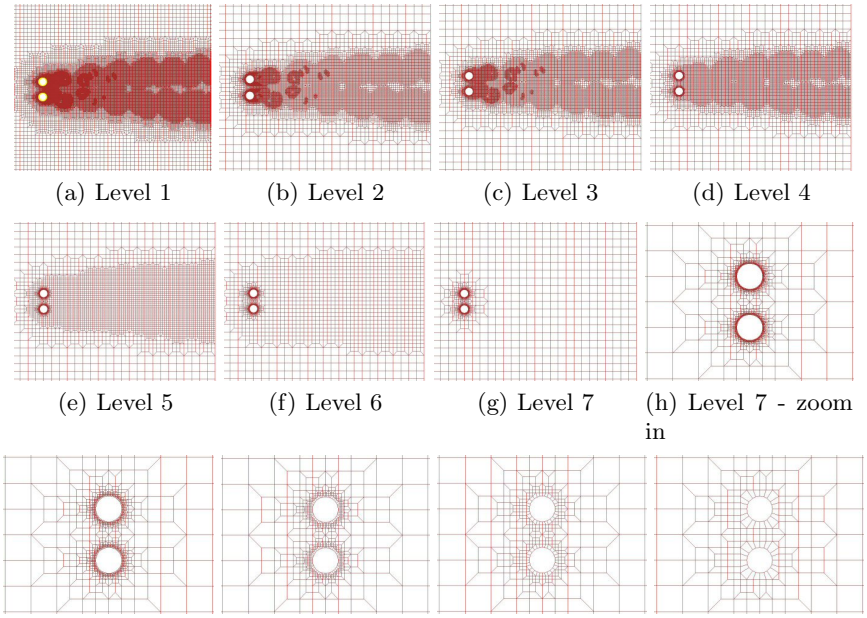
Fig. 5. Modification of the elements in the boundary layer region



(a) t = 1.00 (b) t = 4.00 (c) t = 8.00

(d) t = 12.00 (e) t = 16.00 (f) t = 20.00

Fig. 6. Evolution of grid and vorticity contours for flow over two vertical cylinders at $Re = 200$



(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 4

(e) Level 5 (f) Level 6 (g) Level 7 (h) Level 7 - zoom in

(i) Level 8 (j) Level 9 (k) Level 10 (l) Level 11

Fig. 7. Grid Levels for a domain with two vertical cylinders

Safe Points for Delaunay Refinement

Benoît Hudson[1]

Toyota Technological Institute at Chicago
bhudson@tti-c.org

Summary. Since 1995, when Ruppert invented his Delaunay mesh refinement algorithm and proved that it always produces a mesh barely larger than is optimal, a number of techniques have extended his work to various settings: higher dimension, curved features, fast runtime, parallelism, etc. Each extension needs a different method to choose vertices for the mesh, and thus each extension needs its own tedious proof of the output size guarantee. In this research note, I show how to classify points as safe or potentially unsafe; any algorithm that chooses only safe points achieves the sizing guarantee. Furthermore, I show that it is easy to classify as safe the vertices chosen in prior work. This work frees future meshing researchers to more easily consider varying the choice of vertices in order to achieve important new properties.

1 Introduction

The Delaunay refinement method [Rup95, She98] sits at a happy confluence of theoretically provable qualities and good behaviour in practice. The method starts by computing a Delaunay triangulation, then incrementally adds vertices one by one, retriangulating after each insertion, until the mesh has the desired properties: every mesh element has good quality, and the geometric model being meshed is accurately represented. According to Shewchuk [She98]: *The central question of any Delaunay refinement algorithm is "where should the next vertex be inserted?"* He gives what he calls a "reasonable" answer, but it is not the only one.

In particular, Boivin and Ollivier-Gooch show how to adapt Ruppert's algorithm for some curved inputs [BOG02], adding vertices near the middle of a curved segment, instead of at the midpoint of a linear segment. There have recently been several results on meshing piecewise smooth inputs in higher dimension (*e.g.* [RY07, CDL07]) that discuss where to put new vertices on the curved surfaces. Away from any surfaces, Üngör recommends using the *off-center* [Üng04] rather than the traditional circumcenter, nearly halving the output size in practice. Har-Peled and Üngör [HPÜ05] use off-centers to achieve fast runtime. Alternately, Hudson, Miller, and Phillips [HMP06] achieve fast runtime on a broader class of inputs by starting with a very coarse mesh, then alternating between inserting input points and Steiner (created) vertices. Refinement can also be done in parallel: the question becomes to find a *set* of vertices to insert simultaneously [STÜ07, STÜ04, HMP07].

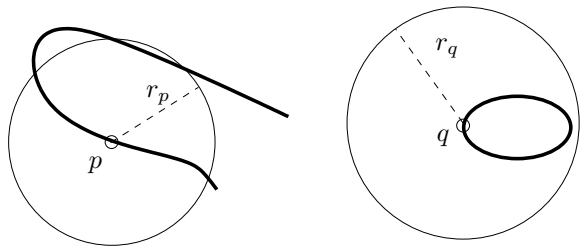


Fig. 1: Two curved features: a curve (left), and an ellipse (right). The ball $B(p, r_p)$ intersects the curve twice, so $\mathrm{lfs}(p) < r_p$. Similarly, $B(q, r_q)$ intersects the ellipse in a sphere, not a ball, so $\mathrm{lfs}(q) < r_q$. The local feature size is the maximum function that satisfies these constraints.

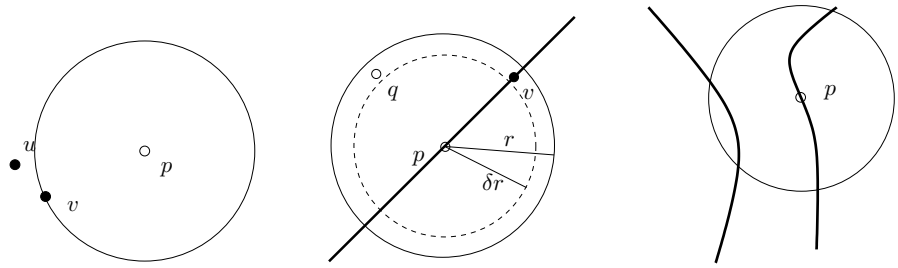
All of these articles reproduce, with minor adjustments, a proof pioneered by Ruppert. The proof states that the algorithm terminates, and outputs a mesh only slightly larger than is optimal. It is typically two pages long. Chernikov and Chrisochoides [CC07] have a more general proof that encompasses a few algorithms, but the present work provides a much broader generalization, allowing far more radical departures from tradition.

Contributions: I describe simple rules for categorizing one or more vertices as safe to insert, even in parallel. The rules appear in Section 3. In this research note, I formally state the main theoretical result (Section 4), but leave the proofs to an online manuscript [Hud08]. Finally, I provide an example of how to apply the proof.

2 Preliminaries

The quality criterion I use in this work is the Voronoi quality, which is closely related to the well-known radius-edge ratio. Let $\mathrm{NN}(v)$ be the distance from a vertex v to its nearest neighbour. Let $R(v)$ be the distance from v to the farthest node of its Voronoi cell. Then the vertex v is ρ**-good** if $R(v) \leq \rho\,\mathrm{NN}(v)$. As input, I assume the mesher receives a **piecewise smooth complex**: a set $\mathcal{X}$ of smooth manifolds—which I call features—that do not intersect except at their boundaries. In the end, the mesh should respect the geometry: each input feature should be represented or approximated by a union of simplices (segments, or triangles) in the output mesh.

To respect both the geometry and the spacing function, the mesh must pack vertices sufficiently densely. This induces a so-called **local feature size** function: in the output mesh, for every point $p \in \Omega$, there must be a mesh vertex within a distance of $O(\mathrm{lfs}(p))$. The function is minimal on the features, where it is defined as illustrated in Figure 1, and elsewhere grows proportionally as we move away from features. See the full version of this paper for a technical definition. One unfortunate fact about the local feature size function I use is that it is only bounded away from zero if the complex is **non-acute**—intersecting features must meet at right or obtuse angles. This is a traditional problem with all Delaunay mesh refinement algorithms; some specially handle small angles, while others fail or risk failing when faced with acute angles. Properly handling inputs with acute angles remains a major open problem.



(a) QUALITY (b) YIELDING (c) ENCROACHMENT

Fig. 2: Examples of almost-safe points.

3 Safe Points

Fundamentally, the safety rules must ensure that if the algorithm were to insert a point p, then, in the output, the relation $\mathrm{NN}(p) \in \Omega(\mathrm{lfs}(p))$ holds—that is, not only is the mesh at least as dense as local feature size, but it is no denser. However, the safety rules should also resemble conditions that algorithms can easily test. For the analysis, I define safety not on points but on triples $\langle p, r, f \rangle$. They should be read as denoting an open ball $B(p, r)$ with radius r, centered on a point p that lies on a feature f. Ideally, the ball should be empty of any vertices; the parameter δ allows accounting for either purposeful perturbation or rounding errors.

Definition 1 (Almost-safe). *Let $\mathcal{X}$ be a piecewise manifold complex. Let $B(p, r)$ be a ball centered on an input feature $f \in \mathcal{X}$, but not centered on any lower-dimensional feature. The triple $\langle p, r, f \rangle$ is* ***almost safe*** *if no vertex on f, nor any 0-manifold that is a subfeature of f, lies within $B(p, \delta r)$, and also one or more of the following conditions hold (see Figure 2):*

- INPUT: *$p \in \mathcal{X}$. Then we define $r \equiv \infty$.*
- ENCROACHMENT: *For some $f' \in \mathcal{X}$, $B(p, r) \cap (f \cup f')$ is not contractible.*
- QUALITY: *$B(p, r)$ contains a vertex v, and $|pv| \geq \rho\,\mathrm{NN}(v)$.*
- YIELDING: *$B(p, r)$ contains both a point q that lies on f_q and a vertex v that lies on f; f is a subfeature of f_q; and q is the center of an almost-safe triple. Under these conditions, we say that q yields to p.*

I call these triples merely almost-safe, because they could be arbitrarily close to a feature. The QUALITY and YIELDING rules would then allow arbitrary amounts of refinement. Following prior work, I solve this issue by requiring that an almost-safe triple that can yield must not be inserted:

Definition 2 (Full Safety). *Let $\langle p, r, f \rangle$ be an almost-safe triple. The triple is fully safe if p does not yield to any point on a subfeature of f.*

The safety rules are all local checks, so we should not be surprised that we can apply them in parallel. I use the technique of serialization: an algorithm is allowed to

insert a set of safe triples in parallel if there exists at least one sequential ordering of the insertions under which each triple is safe when it is inserted. This is equivalent to the following geometric condition (see the full-length paper):

Definition 3 (Safe Set). *Let $\mathcal{S} = \{\langle p_i, r_i, f_i\rangle, \ldots\}$ be a set of triples. We say the set is safe if $\mathcal{S}$ does not have a pair of triples on the same feature, $\langle p, r, f\rangle$ and $\langle p', r', f\rangle$ with $|pp'| < \delta \min(r, r')$.*

Definition 4 (Safe Meshing Algorithm). *An algorithm is a safe meshing algorithm for an input $\mathcal{X}$ if it incrementally inserts safe sets, and no other points.*

In the next section, I claim that a safe meshing algorithm will necessarily terminate, as long as ρ is set sufficiently large (we cannot ask for triangles arbitrarily close to equilateral). In the case of piecewise linear complexes, the requirement on ρ can be improved by replacing the YIELDING rule with:

PROJECTION*: $\langle p, r, f\rangle$ satisfies the* YIELDING *conditions, and, additionally, the orthogonal projection of q onto f is closer to v than is p, where v is defined as in the* YIELDING *rule.*

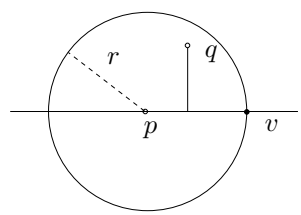


Fig. 3: PROJECTION

4 Termination proof

The key theorem shows that the lfs function acts as a lower bound on the spacing between vertices in the final output mesh of a safe algorithm, assuming that the quality criterion is appropriately set. Then the safe algorithm must terminate. The proof [Hud08] invokes a parameter β. In an algorithm that uses the PROJECTION rule, $\beta = \sqrt{2}$. If we can only prove that it uses the YIELDING rule, then $\beta = 2$. Most extant algorithms assume they make no numerical error, and thus $\delta = 1$.

Theorem 1. *Consider the output of a safe meshing algorithm with ρ such that*

$$\delta^d \rho > \beta^{d-1}$$

Then, at any vertex v in the mesh, $\mathrm{NN}(v) \in \Omega(\mathrm{lfs}(v))$. On a piecewise linear complex, this means that the mesh output by the algorithm is within a constant factor of the smallest possible mesh. For any input, this means the algorithm terminates as long as lfs is bounded away from zero everywhere.

5 Application of the proof

Ruppert and Shewchuk give algorithms to mesh non-acute piecewise linear complexes. For brevity, I discuss only Shewchuk's algorithm, which is the three-dimensional analogue of Ruppert's. The initial step is to compute the Delaunay triangulation of the input points. This corresponds to repeatedly invoking the INPUT rule,

which is always safe. Shewchuk also recommends maintaining a triangulation for each input feature (each input polygon is a set of triangles, and each input segment is a set of subsegments). Shewchuk defines an *encroached* subsimplex as being one whose associated ball contains either a vertex of a disjoint feature, or a point being considered for insertion. When a subfacet is encroached, we should insert its circumcenter p — unless the circumcenter itself encroaches a subsegment. This corresponds to the ENCROACHMENT and YIELDING rules. Finally, if a tetrahedron is skinny (that is, if it has circumradius r, shortest edge uv, and $r \geq \rho|uv|$), then we may attempt to insert its circumcenter, yielding as necessary, which directly corresponds to the QUALITY rule. Shewchuk proves a projection lemma: when a subsimplex is encroached, another subsimplex on the same feature is also encroached for which the PROJECTION rule applies. Shewchuk uses $\delta = 1$ in theory (in practice, numerical errors may reduce δ). Theorem 1 thus proves that the algorithm terminates for any $\rho > 2$, matching Shewchuk's bound; or, in two dimensions, for $\rho > \sqrt{2}$, matching Ruppert's bound.

References

[BOG02] Charles Boivin and Carl Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Inter. J. Num. Meth. Eng.*, 55(10):1185–1213, 2002.

[CC07] Andrey N. Chernikov and Nikos P. Chrisochoides. Three-dimensional semi-generalized point placement method for Delaunay mesh refinement. In *IMR*, pages 25–44, 2007.

[CDL07] Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *IMR*, pages 477–494, 2007.

[HMP06] Benoît Hudson, Gary L. Miller, and Todd Phillips. Sparse Voronoi Refinement. In *IMR*, pages 339–356, 2006. See also Tech. Report CMU-CS-06-132.

[HMP07] Benoît Hudson, Gary L. Miller, and Todd Phillips. Sparse parallel Delaunay mesh refinement. In *SPAA*, pages 339–347, 2007.

[HPÜ05] Sariel Har-Peled and Alper Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. In *SoCG*, pages 228–236, 2005.

[Hud08] Benoît Hudson. Safe Steiner points for Delaunay refinement, 2008. `http://www.cs.cmu.edu/~bhudson/manuscripts/hudson08safe.pdf`.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[RY07] Laurent Rineau and Mariette Yvinec. Meshing 3D domains bounded by piecewise smooth surfaces. In *IMR*, pages 442–460, 2007.

[She98] Jonathan Richard Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *SoCG*, pages 86–95, 1998.

[STÜ04] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement with off-centers. In *EUROPAR*, 2004.

[STÜ07] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement: Algorithms and analyses. *IJCGA*, 17:1–30, 2007.

[Üng04] Alper Üngör. Off-centers: A new type of Steiner point for computing size-optimal quality-guaranteed Delaunay triangulations. In *LATIN*, pages 152–161, 2004.

Meshing of heterogeneous unbounded domains

Benjamin Kettner[1] and Frank Schmidt[1]

Zuse Institute Berlin, Takustrasse 7, D-14195 Berlin, Germany kettner@zib.de

Traditionally, methods for solving partial differential equations such as the Finite Element method and the Finite Difference method can not deal with open region problems. On the other hand, when applying these methods, the computational domain has to be truncated to keep the computational costs at a reasonable level. To resolve this discrepancy, many different approaches have been followed. Techniques that treat the interior as well as the exterior problem by discretising the underlying partial differential equation rely on a discretisation of the exterior domain that satisfies several crucial conditions. We will discuss the problems these conditions pose and present an algorithm that will provide such a discretisation if one exists.

1 Background

The ambivalence between bounded computational domains and unbounded problem formulations is inherent in many applications. Each method applied to resolve this discrepancy requires the coupling of interior and exterior domains. This is either accomplished by the construction of transparent boundary conditions or by combining differentiation techniques for the solution of the coupled problem. The techniques we consider are the Perfectly Matched Layers (PML) Method as formulated by Bérenger in [1] and the Pole Condition formulated by F. Schmidt in [2]. Both require a discretisation of the unbounded exterior domain. Yet, while the triangulation of inhomogeneous bounded interior domains via constraint delaunay algorithms and Rupperts algorithms for refinement and mesh quality enforcement is already very elaborate [3, 4, 5], to our knowledge the discretisation of inhomogeneous exterior domains was not approached so far.
Surprisingly enough, it is sometimes non-trivial to find discretisations of heterogeneous exterior domains and we will show that even in the two dimensional case we can find simple examples where such a discretisation does not exist at all.

2 Discretisation in two dimensional space

In this section we will start by defining the conditions a discretisation of the exterior will have to fulfil in order to be admissible for our application. We will then introduce an algorithm that can produce such discretisations and prove the existence of solutions and the consistence of the algorithm. Both our applications require a discretisation of the exterior domain with quadrilaterals (two dimensions) or prisms (three dimensions) for our discretisation to be admissible we will require the following conditions:

A1: The quadrilaterals have to have one side on the boundary of the computational domain and the other side infinitely far from it

A2: Each quadrilateral or prism has to be homogenous in its interior. So if the inhomogeneities in the exterior domain such as waveguides in optics, are depicted as infinite rays leaving the computational domain, these rays must be contained in exactly one quadrilateral or prism for all times.

A3: For our application these prisms are cut off parallel to the boundary of the computational domain at a finite distance from it. It is necessary for both methods, that the so-created edges parallel to the computational domain form a closed polygon or in the three dimensional case a closed polyhedron, a scaled and distorted version of the computational domain.

While the conditions A1 and A2 are merely local criteria, condition A3 is a global condition to be fulfilled which is one of the difficulties to be mastered to gain a valid discretisation of the exterior domain.

2.1 Challenges

We will first depict the challenges of creating a valid discretisation and then show an example where no valid discretisation can exist. For our example we will choose the unit square as computational domain. Assume light scattering off an object containing four waveguides that leave this computational domain, each on one side of the square, and stretch to infinity. Now let these waveguides stretch into the directions $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ and $\mathbf{c_4}$ numbered counter clockwise starting from the waveguide leaving the domain through the edge on the x-axis (c.f. Fig. 1).

If one considers the corners of the computational domain, rays for the discretisation of the exterior, that attach to these corners have to lie within a cone that is spanned by the prolongation of the edges. If an inhomogeneity is given, this cone is narrowed by a side parallel to this inhomogeneity going through the corresponding corner. In Fig. 1 at the edges of the computational domain, these narrowed cones are marked grey and the discretisation rays are depicted as dashed lines. Note that each inhomogeneity only affects one cone. This is not the general case since in computational domains with inner angles larger than 90° some inhomogeneities might affect several edges.

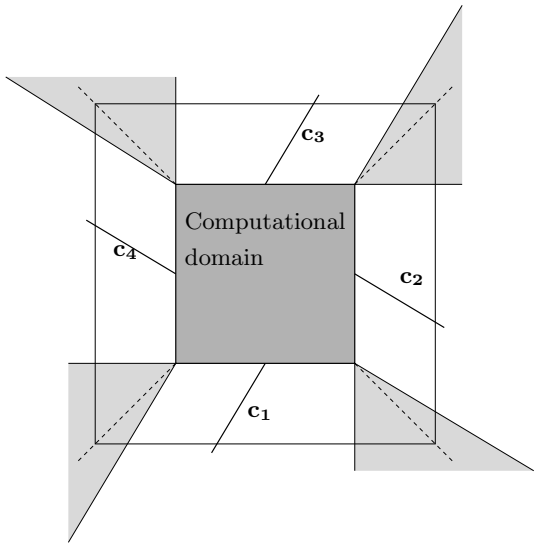


Fig. 1. Four waveguide inhomogeneities leaving the computational domain each through one edge. Dashed lines are exemplatory discretisation rays.

Let α_i denote the angle between $\mathbf{c_i}$ and the edge through which the inhomogeneity defined by $\mathbf{c}_i$ leaves the computational domain.

Lemma 1. *Let the computational Domain be the unit square and the inhomogeneities* $\mathbf{c}_i$ *be defined as before. Let* $0 < \alpha_i < \frac{\pi}{4}$ *for* $i = 1, ..., 4$*. Then there exists no discretisation of the exterior domain that satisfies the conditions A1, A2 and A3.*

Proof. The situation is depicted in Fig. 2. If the first quadrilateral of an exterior domain discretisation is cut off at a distance d_1 from the computational domain, then the closest cut off distance for its neighbour such that a line connecting the corner of the computational domain with the corner created by both cut offs for trigonometrical reasons has a distance of $d_2 = \frac{d_1}{\tan \alpha_1}$ to the computational domain. Since $\alpha_1 < 45°$, $d_2 > d_1$. So if all four inhomogeneities have angles less than 45°, by applying the same argument at each corner of the computational domain, we see that the minimal distance for the last cutoff is $d_4 > d_1$. Via d_1, the x-coordinate of the intersection between the first and the last cutoff is already fixed while $-d_4$ determines the y-coordinate. For the discretisation to be conforming to the given criteria, said intersection point has to lie in the cone formed by the x-axis and $\mathbf{c}_1$. Yet a line parallel to the second edge of the computational domain with a distance of d_1 to it, intersects this cone only in the interval $[0, -d_1 * \tan(\alpha_1)]$ which contains only points whose y-coordinate is less than d_1. Thus both lines can not intersect in the given cone and thus not form a valid discretisation of the exterior domain.

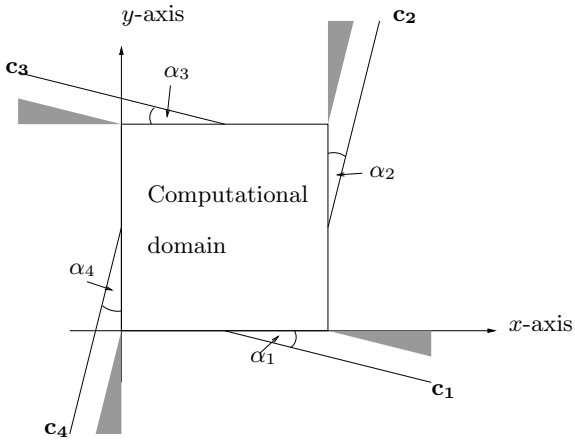


Fig. 2. Schematic of a computational domain together with constraints that make a valid discretisation of the exterior domain impossible.

2.2 Algorithmic solution

We will now present an algorithmic solution of the problem of creating a valid discretisation of the exterior domain if it exists and subsequently prove the existence of a solution and the consistence of the algorithm. The algorithm is given for 2D computational domains for graphic reasons, but it may also be extended to 3D computational domains.

Theorem 1. *Let the following algorithm find appropriate ξ_i in step 5 and thus not terminate in step 6:* **Algorithm:**

1. *Embed the problem into $\mathbb{R}^3$ ("lifting"), leaving the computational domain in the (x_1, x_2) plane;*
2. *Define an axis Ξ that is orthogonal to the (x_1, x_2) plane and intersects the computational domain in an interior point;*
3. *For each edge e_i of the computational domain, define a plane E_i containing e_i and a point on Ξ having $-\xi_i$ as x_3-coordinate;*
4. *At each corner V_i of the computational domain, form the intersection of the planes E_{i-1} and E_i containing the adjacent edges e_{i-1} and e_i thus obtaining lines of intersection $S_i \subset \mathbb{R}^3$;*
5. *Select appropriate ξ_i such that the projections of the S_i into the (x_1, x_2)-plane do not conflict with any constraints given by inhomogeneities and do not intersect each other;*
6. *If such ξ_i exist, continue. Otherwise exit, there is no valid discretisation;*
7. *Intersect all S_i with a plane parallel to the (x_1, x_2)-plane;*
8. *Project the S_i into the (x_1, x_2)-plane. The points of intersection formed in the last step now define the corners for the quadrilaterals required for a discretisation of the exterior domain.*

Then the following holds true:

1. **Existence:** *The algorithm it produces a valid discretisation of the exterior domain, if one exists.*
2. **Consistence:** *For each valid discretisation of the exterior domain, there exist ξ_1, ..., ξ_n for which the algorithm will produce this discretisation.*

Proof. The first proposition is evident from the construction principle. If the algorithm terminates, in step 5, appropriate ξ_i were chosen. The resulting lines of intersection of the E_i will then form a valid discretisation of the exterior domain and since each E_i contains one edge e_i of the computational domain, the endpoints of the projected cut off intersection lines will form edges parallel to the edges of the computational domain which in turn form a closed polygon thus resulting in a valid discretisation of the exterior domain. To prove the second proposition, we lift the scaled computational domain created by the given discretisation of the exterior domain to a plane in $\mathbb{R}^3$ that is parallel to the (x_1, x_2)-plane. The corners of this lifted scaled version of the computational domain are then connected to the corners of the original computational domain with rays, yielding the E_i and thereby unambiguously defining the ξ_i.

2.3 Conclusions

The algorithm presented here may easily be extended to three dimensions. Yet the selection of the ξ_i in step 5 is an open issue that will require utilisation of optimisation algorithms for non-convex, non-linear programming problems. Also the lifting of the surface grids onto the parallel sides of the prisms in 3D is an open issue that will have to be tackled in the near future. Thus the first steps towards a fully automatic discretisation of the exterior domain have been presented, while more work will still have to be done to arrive there.

References

1. J. Bérenger, (1994) A perfectly matched layer for the absorption of electromagnetic waves. In: Springer, J. Comput. Phys.
2. F. Schmidt (2002) Solution of Interior-Exterior Helmholtz-Type Problems Based on the Pole Condition Concept: Theory and Algorithms. Habilitation Thesis, Free University Berlin, Fachbereich Mathematik und Informatik
3. J. R. Shewchuck (2002) Delaunay Refinement Algorithms for Triangular Mesh Generation In: Computational Geometry: Theory and Applications
4. P-L. George (1991) Mesh Generation: Application to Finite Elements, J. Wiley & Sons
5. J. F. Thompson, B. K. Soni and N. P. Weatherill (1999) Handbook of Grid Generation, CRC Press
6. Benjamin Kettner (2006) Ein Algorithmus zur prismatoidalen Diskretisierung von unbeschränkten Aussenräumen in 2D und 3D unter Einhalten von Nebenbedingungen. Diploma Thesis, Free University Berlin, Fachbereich Mathematik und Informatik

GMDS: A Generic Mesh Data Structure

F. Ledoux[1], J.-C. Weill[1], and Y. Bertrand[2]

[1] CEA/DAM Île de France, Arpajon, France
{franck.ledoux,jean-christophe.weill}@cea.fr
[2] Laboratoire IRCOM-SIC, Université de Poitiers, France
yves.bertrand@univ-poitiers.fr

1 Introduction

The development of a mesh data structure is a crucial task that needs to get the right balance between memory consumption and speed performances and to predict the useful and pertinent characteristics of the data structure. Today, working in a massive parallel context increases the difficulty of such a development. For software engineering reasons (code reusing, development cost) and technical reasons (mesh connectivity management, generality level, parallelism), the need for general meshing infrastructures has been recognized. Several development efforts have been done in the last few years [2, 3, 4, 5]. From our point of view, FMDB [5] is the most complete work. It is a C++ library providing a mathematically-founded mesh data structure which represents any cellular mesh in a memory distributed context. But, essentially due to an intensive use of STL containers, it consumes too much memory. In our applicative context, this drawback is very penalizing that is why we develop a new generic parallel C++ mesh data structure dedicated to mesh generation and modification. The proposed data structure, called Generic Mesh Data Structure (GMDS) is based on two mathematical models: a traditional cellular model [5] optimized in memory consumption, and the combinatorial map model [1] which allows us to handle global unpredictable topological modifications and provides a high level of genericity that you can not reach with traditional approachs.

This paper provides a snapshot of this work in progress. Sections 2 and3 describe our cellular model and the one based on combinatorial maps. Section 4 provides first memory consumption comparisons between our data structure, FMDB and MSTK [2].

2 Cellular Model

A n-dimensional cellular mesh model can be defined by the cells it handles and the available connectivities between these cells[3]. Such a description is topological, it gives an unambiguous shape-independent representation of the mesh. Some mesh models are given in Fig. 1. The description we use to modelize a mesh consists of a n-dimensional vector C and a $n \times n$ matrix[4] D. Vector C describes the stored cells and matrix D the stored connectivities. $D_{i,j} = 1$ (respectively 0) means that connectivities from i-cells to j-cells are stored (resp. not stored). A second $n \times n$ matrix I is computed from D and C to know which indirect adjacencies are available in the mesh model. Direct and indirect matrices of models M^3 and M^4 are given in Fig. 2. Indirect adjacencies are computed on the fly to satisfy a user request. A similar representation is used in FMDB where a single 4×4 matrix stores the available cells in the diagonal and cell adjacency otherwise. Adjacency between same dimension cells can not be stored while it is important in many meshing algorithms.

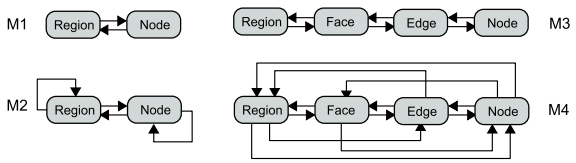


Fig. 1. Some mesh models. M^1 and M^2 are reduced models where only nodes and regions are present. Connectivities are modelized by arrows. In M^2, a region stores its nodes and some adjacent regions.

GMDS is a C++ library that intensively uses object-oriented concepts and generic programming. Generic classes are used to optimize storage consumption while interface classes provide flexibility and a user-friendly interface. every template classes has a parameter defining available cells and connectivities in the mesh. Direct and indirect matrices are computed from it. Generic programming techniques, like traits usage, optimize accesses and basic modifications throw exceptions without any test when an operation is not available for a specific mesh model. Generic cell classes have an extra parameter specifying the cell type (quad, triangle, hexahedron, tetrahedron,etc). This parameter coupled with the mesh model description allows us to optimize cell storage: Every cell class has a pointer C-style tabular whose size is defined at compile time considering mesh model and cell type. Downward connectivities are stored with the just necessary space while upward connectivities use two extra pointers to handle an extensible collection of connectivities. Let us

[3] Such a definition is not sharp enough to define a family of meshes. Generally, supported n-dimensional meshes are equivalent to n-dimensional quasi-manifolds.

[4] In this paper, we do not mathematically define what a cellular mesh is.

consider models M^1 and M^2 of Fig. 1. Connectivities of a tetrahedron in M^1 are stored inside a 4-size tabular while connectivities of the same tetrahedron in M^2 are stored in a 8-size tabular, the four first item store node adjacency while four last items store region adjacency. In M^2, hexahedron connectivities are stored using a 16-size tabular.

$$D^3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \; I^3 = \begin{pmatrix} 1 & \times & 1 & 1 \\ \times & 0 & 1 & 2 \\ 1 & \times & 1 & \times \\ 2 & 2 & \times & 2 \end{pmatrix} \; D^4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \; I^4 = \begin{pmatrix} 1 & \times & \times & \times \\ \times & 0 & \times & \times \\ \times & \times & 1 & \times \\ \times & \times & \times & 2 \end{pmatrix}$$

Fig. 2. Direct and indirect matrices of mesh models M^3 and M^4 given in Fig. 1. With matrix I^3 we know that to get the adjacent nodes of a face we have to go through edges ($I^3_{2,0} = 1$ and $I^3_{1,0} = \times$).

3 Combinatorial Map Model

Combinatorial maps define a general framework to encode any subdivision of n-dimensional topological spaces orientable or non-orientable with or without boundaries. Traditional cells are replaced by algebraic definitions where topology is built from a set of abstract elements, the darts, and applications α_i, with $0 < i \leq n$, defined on these darts. In 3D, we get the following definition which can be generalized to any dimension.

Definition 1 *A combinatorial* 3*-map* M *is a 4-tuple* $(D, \alpha_1, \alpha_2, \alpha_3)$ *where* α_1 *is a permutation*[5] *on* D *and* α_2 *and* α_3 *are involutions*[6] *on* D*.*

Usual mesh cells can be built as dart subsets obtained using α_i applications. For instance, on Fig. 3, the quad face can be described by dart d and application α_1. Note that a cell is a particular case of orbit[7]. An example of orbit is a face inside a region which allows you to attach properties to the two sides of a face. Combinatorial map model provides us a generic data structure which eases algorithm development: orbit notion gives flexibility to attach new types of properties; the management of any 3D cell is direct as cells are implicitly defined; all topological traversals and adjacency retrievals are possible. Moreover, it is a performing issue to handle dual meshes, block-structured meshes and non-conforming meshes.

Combinatorial maps are defined using three classes for maps, darts and orbits. Parameters defining a map-based mesh are the mesh dimension and

[5] A permutation f on X is a one-to-one mapping on X such that $\forall x \in X, \exists k \geq 1, f^k(x) = x$.

[6] An involution f on X is a one-to-one mapping on X such that $f^2 = id$.

[7] In 3D, there exist 16 orbits and 4 cells.

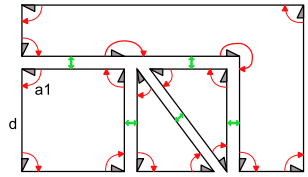


Fig. 3. A 2-dimensional combinatorial map. Darts are modelized by arrows as they are oriented. α_1 (resp. α_2) links darts belonging to adjacent edges (resp. faces).

the available orbits. Adjacency relations have not to be specified since they intrinsically exist in the combinatorial map model. The storage is very simple since the only entities to store are darts and orbits. Darts are defined by pointers given relation α_1, α_2 (and α_3 in 3D) and pointers to the orbits containing them. Orbits are defined by a pointer on a dart to access to the topology and an id defining it.

4 Memory Consumption Comparison

We have compared memory consumption between our data structure, FMDB and MSTK to evaluate the potential gain of our approach. This comparison is based on source code analysis by keeping comparable information. For instance, in FMDB, we do not consider elements relative to parallelism and geometric classification. To estimate "cell cost", we consider the type of attributes (`int`, `long`, `T*`), container strategies and potential virtual table pointers. The cost of a mesh is then computed using the next formula given in [2, 3]:

$$Storage(M) = \sum_{d=0}^{3} \sum_{i=1}^{N_d} (S_{cell} + \sum_{q=0}^{3} |\{M_i^d\{M^q\}\}| \times S_{adj})$$

where N_d is the number of d-dimensional cells, S_{ent} is the amount of memory each cell uses, S_{adj} is the amount of memory each adjacency uses, and $|\{M_i^d\{M^q\}\}|$ is the number of q-cells adjacent to d-cell M_i^d and stored in the mesh. The formula is different for combinatorial maps:

$$Storage(M) = \sum_{d=0}^{3} \sum_{i=1}^{N_d} S_{cell} + \sum_{k=0}^{N_k} S_{dart}$$

where N_k is the number of darts and S_{dart} is the amount of memory each dart uses[8]. We get results of Tab. 1 where the cost of mesh is the specified number of bytes $\times$ the number of mesh nodes. For any models M^1, M^2, M^3 and M^4, we consider tetrahedral (T) and hexahedral (H) meshes. The current GMDS cellular model consumes less memory than FMBD and MSTK while

[8] That is, 2 or 3 pointers to store α_i and a pointer by available orbit type.

combinatorial maps are expensive for reduced models. The low cost of cellular models is due to the fact we do no store downward adjacencies with generic containers. The high cost of combinatorial maps is due to the presence of all connectivities whatever the mesh model it is. Note this overcost is not prohibitive for full models.

Table 1. Memory Consumption Comparison on 64 bit machines (in bytes).

	M^1 (T)	M^1 (H)	M^2 (T)	M^2 (H)	M^3 (T)	M^3 (H)	M^4 (T)	M^4 (H)
FMDB	193	66	×	×	882	284	1247	408
MSTK	145	50	203	70	513	168	822	272
GMDS(cell)	71	28	117	44	325	112	506	178
GMDS(map)	642	254	642	254	1015	392	1015	392

5 Conclusion and Future Works

In this note, we have shortly introduced a mesh data structure based on a generic cellular model and combinatorial maps. This combination will allow us to represent any kind of meshes while optimizing memory consumption for cellular models. Combinatorial maps are memory expensive but they provide a high genericity level allowing us to easily handle complex kinds of meshes. Moreover it is a perfect framework to prototype new algorithms before converting them into an memory-optimized mesh model. The development of this data structure just begins and several studies and developments are expected in little time. Other memory consumption comparisons and speed comparisons will be then done. Some other developments are also planned: conversion from a model to another one, managment of memory-distributed applications, user-friendly mecanism to develop algorithms above GMDS and so on.

References

1. Lienhardt P (1991) Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design 23(1):59–82
2. Garimella RV (2002) Mesh data structure selection for mesh generation and FEA applications. International Journal for Numerical Methods in Engineering 55:451–478
3. Remacle JF and Karamete BK and Shephard MS (2003) Algorithm Oriented Mesh Database. International Journal for Numerical Methods in Engineering 58:349–374
4. Tautges T, Ernst C., Merkley K, Meyers R and Stimpson C (2005) Mesh Oriented datABase (MOAB). `http://cubit.sandia.gov/cubit`
5. Seol ES (2005) FMDB: Flexible Distributed Mesh Database for Parallel Automated Adaptative Analysis. Thesis, Faculty of Renssealer Polytechnic Institute

Hexahedral Meshing of complex and invalid CAD Geometries

Torsten Sadowski[1] and Robert Liebich[2]

[1] Berlin Institute of Technology torsten.sadowski@tu-berlin.de
[2] Berlin Institute of Technology robert.liebich@tu-berlin.de

Summary. The primary requirement for a mesh generator used in an automated tool is robustness. Surface based meshing algorithms fail very often especially with invalid boundary representation geometries. With generic boundary representation formats like STEP or IGES, invalid geometries are very common due to the loss of information and precision in CAD Data Exchange. This paper presents an approach to a robust hexahedral mesh generator that is insensitive to invalid geometries and produces meshes which can be used for stiffness calculations.

1 Motivation

A special preprocessor for the multibody system simulation of crankshafts (CADSimShaft [SL07]) is under development by the Chair for Engineering Design and Product Reliability at Berlin Institute of Technology. An important part of the preprocessing is the stiffness calculation of crankshaft parts. Stiffness calculation is a common task in mechanical engineering and easily done using the finite element method. But first of all a model preparation is needed that includes the most important part, the mesh generation.

CADSimShaft uses OpenCASCADE [OPE08] as its geometry kernel. The CAD models of the crankshafts are imported using generic CAD data exchange formats like STEP and IGES resulting in invalid geometries sometimes. But even if the geometry is valid the surface based mesh algorithms are not always successful. Of 44 different CAD crankweb models only 9 could be meshed using the available algorithms in the Salome [OPE06] project. Common cause for the failed meshes was a failure to create a consistent and closed surface mesh in order to initialize the volume meshing algorithm.

Robustness of the mesh generation process is an absolute necessity for the CADSimShaft software, even with geometrically invalid boundary representation models. All failures are surface related, therefore a new volume based mesh generator is being developed. Hexahedral meshes are preferred over tetrahedral meshes because of usually better computational results [BPM$^+$95].

2 Volume based Hexahedral Meshing

Volume meshes can be build from a mesh of the surface of the model or directly without an existing surface mesh. The volume meshing algorithms presented in [Owe98] can be classified into both groups. The algorithms working directly with the surface or a surface mesh are: Boundary Constrained Triangulation, Advancing Front, Medial Surface, Plastering and Whisker Weaving.

Volume oriented algorithms do not require a surface mesh and cope better with dirty geometries[WS02][LLG+06]. They can be used as long as it is possible to check if a point is inside or outside of a volume. This can be successfully determined using the CAD kernel even if the surface representation is not completely valid. The Octree-based and Grid-based algorithms can be used for tetrahedral and for hexahedral meshing. An interesting variation of the octree for hexahedral meshes, the 27-tree, is presented in [SSW96].

3 CAD Model Preparation

Complete CAD models are unnecessarily complex for mesh generation. A reduction of their complexity is possible with the CAD software used to design the model. It is quite easy to export the design before all fillets and chamfers are applied if the full design model and not just a generic exchange file is available. A different approach is using feature recognition [Tau01] to remove needless detail. Much research effort is spent on the closely related automatic decomposition [Bla96][LG96][LGT99][LGT01]. Automatic decomposition is difficult to implement for complex shapes where fillets on other fillets might have erased the basic geometry.

Two more simple approaches do exist. Either the many small faces are combined to fewer but larger approximations [Mez08] or the individual faces are ignored and used to support a mesh node only if necessary.

Boundary Representations (BRep) of solid parts are constructed from six different topological entities: Vertex, Edge, Wire, Face, Shell and Solid [OPE07]. Three of these (Wire, Shell, Solid) carry organizational information only, the other three (Vertex, Edge, Face) have geometrical information as well.

Most of the edges result from fillet patches. These patches can be very small or narrow and have edges connecting tangent surfaces. They are detrimental to the mesh because of the small face they define. In order to maintain the shape it is necessary to identify those topological faces, edges and vertices which are important for the model.

Automatic classification can be based on entities carrying geometric information only. Possible classifications for faces are tangency at face boundaries, surface area and aspect ratio. Edges can be classified by tangency of connecting faces and length.

The general shape of CAD models is preserved good enough for most applications if all sharp edges and vertices at corners are represented in the mesh. Only a single parameter, the "sharpness" of an edge is necessary for an automatic complexity reduction of the CAD model. The required information is given by the standard exchange formats STEP and IGES. Tangency is therefore the most important classification possibility and can either be checked for each edge or for each face. Edge

based classification is easier to implement because every edge is connected to exactly 2 faces whereas faces are bounded by n edges connecting them to n more faces. This is similar to [Mez08] but individual faces are not used to rebuild a new surface in the present approach.

Edge tangency checking is done by calculating the angle between the surface normals on the faces at several points along the connecting edge. If an angle is larger than a given threshold the edge must be considered in the mesh. The result is a number of master edges of the shape. Vertices at non tangent master edge connections are master vertices. Approximately one third of the topological edges and one eighth of the topological vertices are master edges and master vertices for all tested crankwebs.

Figures 1 and 2 give an impression of the reduction in complexity for a crankweb using an threshold angle of 6°. Only a fraction of the topological edges shown in fig. 1 is selected as master edges. The few master vertices are shown in fig. 2 as small crosses.

4 Implementation

The mesh can be created with an octree based algorithm or with a grid based algorithm. The grid based approach is easier to implement for a BRep model and results in a regular mesh for the first step. The grid is created with the help of the spatial twist continuum (STC) [MBBM]. The STC of a regular hexahedral grid is a regular hexahedral grid as well. Hence, the additional effort seems unjustified. On the other hand it is easier to manipulate hexahedral meshes using their dual STC [BMT+97][TK03][HBO04]. Only one input parameter is used, the average element edge length.

The mesh generation is done in six steps: creation of the STC as a regular grid using ray-casting, deformation of the STC to fit the surface (similar in effect to [Dho99]), construction of the primal mesh, projection of primal mesh surface nodes on the model surface, paving of identified master wires, moving mesh nodes on master vertices.

5 Results and Conclusion

The robust meshing of the crankwebs as the most important target could be achieved. All crankwebs could be meshed into hexahedra like the example Fig. 3. First tests for stiffness calculations with hexahedral meshes showed good results. Because the true stiffness is not known, the results can only be checked against results from other FEM calculations which are currently used for the multibody dynamic simulation.

The meshes presented in this work are useful for stiffness calculations but not yet for stress analysis. But the mesh and its STC is a useful base for improvements using sheet insertion [HBO04]. The current study showed some elements along the master wires being deformed into concave shapes. Although this has not been a challenge for the method so far, sheet insertion to be used in the future would solve this[MT95][ZHB08][MESB08][PBSB08].

Fig. 1. Crankweb with all topological edges

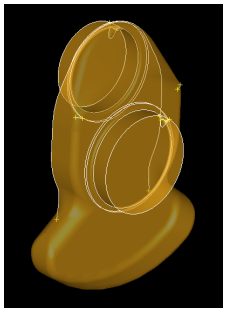


Fig. 2. Crankweb showing only master (sharp) edges

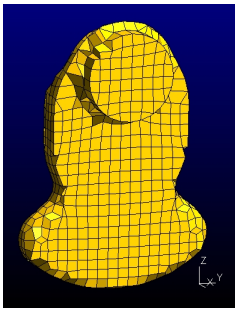


Fig. 3. Finished Mesh

References

[Bla96] Ted Blacker. The cooper tool. In *5th International Meshing Roundtable*, pages pp.13–30. Sandia National Laboratories, October 1996.

[BMT+97] T.D. Blacker, S.A. Mitchell, T.J. Tautges, P. Murdock, and S.E. Benzley. Forming and resolving wedges in whisker weaving and the spatial twist continuum. *Engineering with Computers*, 13:pp. 35–47, 1997.

[BPM+95] Steven E. Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardema. A comparison of all-hexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings, 4th International Meshing Roundtable*, pages pp.179–191. Sandia National Laboratories, October 1995.

[Dho99] Guido D. Dhondt. Unstructured 20-node brick element meshing. In *Proceedings, 8th International Meshing Roundtable*, pages pp.369–376, South Lake Tahoe, CA, U.S.A., October 1999.

[HBO04] Nathan J. Harris, Steven E. Benzley, and Steven J. Owen. Conformal refinement of all-hexahedral element meshes based on multiple twist plane insertion. In *Proceedings, 13th International Meshing Roundtable*, volume SAND #2004-3765C, pages pp.157–168, Williamsburg, VA, September 19-22 2004. Sandia National Laboratories.

[LG96] Shang-Sheng Liu and Rajit Gadh. Basic logical bulk shapes (blobs) for finite element hexahedral mesh generation. In *5th International Meshing Roundtable*, pages pp.291–306. Sandia National Laboratories, October 1996.

[LGT99] Yong Lu, Rajit Gadh, and Timothy J. Tautges. Volume decomposition and feature recognition for hexahedral mesh generation. In *Proceedings, 8th International Meshing Roundtable*, pages pp.269–280, South Lake Tahoe, CA, U.S.A., October 1999.

[LGT01] Yong Lu, Rajit Gadh, and Timothy J. Tautges. Feature based hex meshing methodology: Feature recognition and volume decomposition. *Computer Aided Design*, 33:221–232, 2001.

[LLG+06] Y. K. Lee, Chin K. Lim, Hamid Ghazialam, Harsh Vardhan, and Erling Eklund. Surface mesh generation for dirty geometries by shrink wrapping

using cartesian grid approach. In *Proceedings, 15th International Meshing Roundtable*, pages pp.325–338. Springer-Verlag, 2006.

[MBBM] P. Murdoch, S. Benzley, T. Blacker, and S. Mitchell. The spatial twist continuum captures the global connectivity of an all–hexahedral finite element mesh.

[MESB08] Karl Merkley, Corey Ernst, Jason F. Shepherd, and Michael J. Borden. Methods and applications of generalized sheet insertion for hexahedral meshing. In *Proceedings of the 16th International Meshing Roundtable*, pages pp.233–250. Springer-Verlag, 2008.

[Mez08] Andrey Mezentsev. An efficient geometrical model for meshing applications in heterogeneous environments. *Proceedings of the 16th International Meshing Roundtable*, pages 151–168, 2008.

[MT95] Scott A. Mitchell and Timmothy J. Tautges. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In *Proceedings, 4th Internatinal Meshing Roundtable*, pages pp.231–240. Sandia National Laboratories, October 1995.

[OPE06] OPEN CASCADE SAS. Salome : The open source integration platform for numerical simulation. http://www.salome-platform.org, 2006.

[OPE07] OPEN CASCADE SAS. *Modeling Data User's Guide*, 2007.

[OPE08] OPEN CASCADE SAS. Open CASCADE Technology, 3d modeling & numerical simulation. http://www.opencascade.org, 2008.

[Owe98] Steven J. Owen. A survey of unstructured mesh generation technology. In *Proceedings, 7th International Meshing Roundtable*, pages pp.239–267. Sandia National Lab, October 1998.

[PBSB08] Michael Parrish, Michael Borden, Matthew Staten, and Steven Benzley. A selective approach to conformal refinement of unstructured hexahedral finite element meshes. In *Proceedings of the 16th International Meshing Roundtable*, pages pp.251–268. Springer-Verlag, 2008.

[SL07] Torsten Sadowski and Robert Liebich. Crankshaft slicing for multibody-system-simulation. In *SimDrive International User Meeting*, 2007.

[SSW96] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *5th International Meshing Roundtable*, pages pp.205–216. Sandia National Laboratories, October 1996.

[Tau01] T. Tautges. Automatic detail reduction for mesh generation applications. In *Proceedings 10th International meshing roundtable*, pages pp 407–418, 2001.

[TK03] Timothy J. Tautges and Sarah E. Knoop. Topology modification of hexahedral meshes using atomic dual-based operations. In *Proceedings, 12th International Meshing Roundtable*, pages pp.415–423. Sandia National Laboratories, Sept. 2003.

[WS02] Z. J. Wang and Kumar Srinivasan. An adaptive cartesian grid generation method for 'dirty' geometry. *Int. J. Numer. Meth. Fluids*, 39:703–717, 2002.

[ZHB08] Yongjie Zhang, Thomas J.R. Hughes, and Chandrajit L. Bajaj. Automatic 3d mesh generation for a domain with multiple materials. In *Proceedings of the 16th International Meshing Roundtable*, pages pp.367–386. Springer-Verlag, 2008.

Local remeshing of CAD models for finite element analysis: a feature-based approach

M. Sypkens Smit and W. F. Bronsvoort

Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
m.sypkensmit@tudelft.nl, w.f.bronsvoort@tudelft.nl

Summary. An original approach to locally remeshing CAD models for finite elements analysis is proposed. The method analyses the geometry of two feature model variants by individually comparing features. The ideas are validated by experimental results.

1 Introduction

Finite element analysis is nowadays an integral part of the product development cycle. It stands apart from the principal design step, as the two fields evolved largely separately, each bearing their particular history of developed solutions and associated practices.

This difference is reflected in practice by design and analysis each having their distinct geometries. One of the areas addressed by research has been the link between the design and the analysis model. The former is nowadays commonly a feature model and the latter a mesh. There does not really exist a link between the two yet, as the mesh is often completely regenerated at each step of the design cycle. With models getting more complex, the approach of reusing meshes by locally adapting them, instead of completely reconstructing them at every step, is getting more attractive.

2 Basic idea of remeshing

The concept of remeshing for finite element analysis is that, instead of completely regenerating the mesh for each analysis, the mesh is adapted to reflect the modifications in the geometry it is linked to. The concept is not new, but its broader application to the development cycle has yet to catch on. A likely reason is that the effort is perceived to be too large for the gain: meshes can be generated fairly quickly and dealing with modifications between geometries is far from easy. Quality meshes, however, are often more costly to generate.

What constitutes 'a quality mesh' depends on the particular problem context, but obtaining it usually involves some kind of optimisation procedure or comprehensive exploration of a set of possible solutions, e.g. [1] and [2]. Especially in the case of small modifications to large models that already have a quality mesh, remeshing is likely to be beneficial.

Many research efforts have been, and still are, directed at the mesh deformation / morphing approach. Here the existing mesh is reshaped, as if it were rubber, by stretching, compressing and bending. See [3] for a recent example. The quality of the mesh elements can be severely reduced by such mesh deformation and changes in topology cannot be handled generically. Amongst the few contributions that did consider modifications to the topology in the context of remeshing are [4] and [5]. These approaches are lacking, however, on genericness, flexibility, or implementation. Efficiently handling generic topological changes is in particular relevant when looking at the remeshing problem in the context of the product development cycle, where changes in model topology are common.

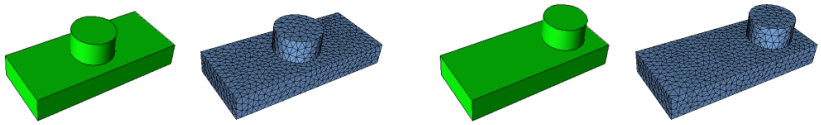
In our research, we approach the remeshing problem from the angle of feature modelling and the desire to maintain the analysis model as an integral part of the design model. To this end, we have come up with a new way to associate geometry between two different feature models.

3 Remeshing feature models

Feature modelling today is the standard in CAD. There is no uniform definition of what feature modelling is. Generally it enables a designer to construct a model by combing higher level objects (the features), instead of manipulating lower level geometry. Examples of common features are holes, ribs, bumps and fillets. Features are manipulated through their parameters, which affect their shape and position in the model. In advanced modelling systems, they also carry semantics that actively support the designer in his decisions and can warn him of unintended consequences.

The design space is thus essentially parameterised by means of the features. This makes it a natural choice to reason by means of the features when looking at the difference between two models, such as needed when remeshing. The idea is that each additive feature corresponds to some part(s) of the model's mesh. We could say that these features 'carry' the mesh. After a modification, we want to create the new mesh by copying, as much as possible, the mesh that the features carried in the previous mesh. Figure 1 shows two variants of a simple model and the corresponding meshes. It is easy to imagine that the mesh of Figure 1(b) could *almost* be acquired by just recombining the mesh covering the base block and the mesh covering the cylinder.

In reality, however, we cannot recombine the meshes corresponding to features without some extra work. The principal reason is that features interact, i.e. they overlap at faces or even with their volumes. Volumetric meshes do



(a) Original model: base block with a cylindrical feature on top, and its mesh

(b) Modified model: base block with cylindrical feature translated, and its mesh

Fig. 1. Original and modified model and their meshes

not, and should not, respect the internal feature boundaries, and thus mesh elements will not align on the whole boundary of features. When the relative positioning between features changes, such as in Figure 1, then the mesh needs to be adapted in the areas where the interaction changes.

For our feature-based remeshing approach, we thus need to find all areas where feature interactions change. Generally, feature modelling systems are history-based, i.e. they do not explicitly store the interactions of all features, but compute a new boundary representation (BRep), and continue the rest of the operations with the new geometry. This approach is thus not suitable for finding and describing the changes in interaction.

A data structure that does store the complete geometry of all the features is the *cellular model* [6]. It does this by keeping the complete geometry of all features, and where features intersect they are divided into cells. A feature that covers a cell, is said to *own* that cell. Each cell is owned by at least one feature; where features overlap, the cell has multiple owners.

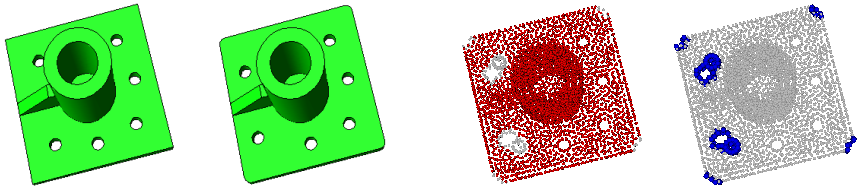
In the cellular model, we have a complete geometric description of each feature, which includes those parts of its boundary that are not part of the BRep. It also contains an explicit partition of the volume of the model. After a model has been modified, we can thus analyse for each feature how its geometric interaction with the other features has changed. If nothing has changed, then all mesh elements covering the feature can be copied to the mesh of the new model.

For the case that something has changed, we need a consistent classification of what has changed and in what respect. For example, feature geometry that previously was internal to the model, can have become part of the BRep; cells that used to be empty, due to overlapping subtractive features, can now contain material. For this classification, we use the *feature difference* [7]. The feature difference compares the geometry of a feature, as it was in a previous variant of the model, with the geometry of the feature in a new variant of the model. This geometry includes all cells, faces, edges and vertices that emerged through interaction with other features in the model.

Based on the feature difference, we can thus decide for each feature which parts of it could contribute to the new mesh model and where/how the mesh needs to be adapted. The final challenge is to combine this information, on

each individual feature, for the construction of a single mesh for the complete model. Since features intersect, we must be careful not to copy points multiple times to the same region. Also, we need to identify the regions for which no mesh can be copied, which can be due to new features or the relocation of subtractive features. Lastly, we must pinpoint the regions where meshes need some further work, e.g. because mesh with different origins ended up copied next to each other, or because of interaction changes (such as in Figure 1.

The outline of the algorithm is: 1) analyse the difference between the two feature models, 2) identify the regions in the new model for which mesh can be copied and actually copy it, 3) create new mesh in the remaining areas of the new model, 4) connect and optimise the mesh in all regions where a (good) connection is lacking. Figure 2 illustrates the regions of mesh that are fixed (red) or should be optimised (blue) at the start of step 4 of the algorithm.



(a) Original and modified model

(b) Copied nodes (red) and new/active nodes (blue)

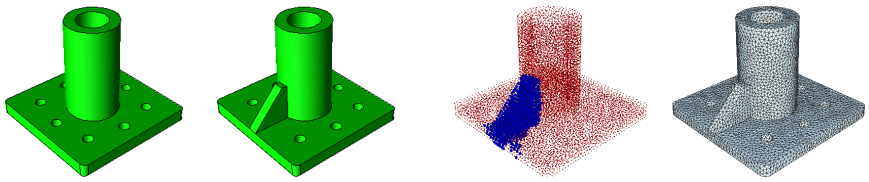
Fig. 2. Example of model modification with corresponding copied nodes (red) and new/active nodes (blue)

4 Experimental results

To test our feature-based approach to remeshing, we have implemented the variational tetrahedral meshing algorithm [8], with some minor modifications that improve its applicability to mechanical models. This algorithm results in high quality mesh elements, by alternating optimisation steps of the boundary and interior mesh. This algorithm produces a Delaunay mesh, which is a major advantage here as it avoids the need to explicitly copy the connectivity of the elements. Instead, we can just copy the nodes.

Figure 3 illustrates a result of this approach. We compared the result with that of the full meshing procedure. The remeshing took *30*% of the normal meshing time. The element quality distrubition is nearly identical.

In general, our experiments indicate a efficiency gain between 50% and 80%. The gain depends on the number of mesh nodes, the amount of surface area relative to the volume, and the extent of change in feature intersections.



(a) Original and modified model **(b)** Region of optimisation and result of remeshing

Fig. 3. The result of remeshing a tool with a new stiffener feature

5 Conclusions and looking ahead

We have demonstrated that our approach to remeshing, based on the feature difference, is feasible and that it can improve efficiency. The gains depend on the number of mesh elements, the extent of the model modifications, and the complexity of the meshing algorithm.

Our current approach is not universal. It relies on the Delaunay criterion for the connectivity. In order to support other kinds of meshes and meshing algorithms, we need to be able to reliably copy the mesh connectivity between the nodes. The handling of graded meshes is another challenge. Can a sizing field efficiently be maintained under model modification? What are good strategies to maintain the conformity of the mesh to the sizing field when remeshing? What is the impact on efficiency? Opportunities for further research are abundant.

References

1. Lagüe, J.F., Hecht, F.: Optimal mesh for P_1 interpolation in H^1 seminorm. In: Proc. of the 15th International Meshing Roundtable, pp. 259–270. Birmingham, AL (2006)
2. Klingner, B.M., Shewchuk, J.R.: Aggressive tetrahedral mesh improvement. In: Proc. of the 16th International Meshing Roundtable, pp. 3–23. Seattle, WA (2007)
3. Date, H., Onosato, M.: Triangular mesh deformation based on dimensions. Computer-Aided Design and Applications **5**(1–4), 287–295 (2008)
4. François, V., Cuillière, J.C.: 3D automatic remeshing applied to model modification. Computer-Aided Design **32**(7), 433–444 (2000)
5. Sheffer, A., Ungor, A.: Efficient adaptive meshing of parametric models. Journal of Computing and Information Science in Engineering **1**(4), 366–375 (2001)
6. Bidarra, R., de Kraker, K.J., Bronsvoort, W.F.: Representation and management of feature information in a cellular model. Computer-Aided Design **30**(4), 301–313 (1998)
7. Sypkens Smit, M., Bronsvoort, W.: The difference between two feature models. Computer-Aided Design and Applications **4**(6), 843–851 (2007)
8. Alliez, P., Cohen-Steiner, D., Yvinec, M., Desbrun, M.: Variational tetrahedral meshing. ACM Transactions on Graphics **24**(3), 617–625 (2005)

A Relaxation Method for Surface-Conforming Prisms

Ross Whitaker, Robert M. Kirby, Zhisong Fu
Scientific Computing and Imaging Institute, University of Utah
Salt Lake City, UT 84112

Abstract:
This paper presents a method for computing thin layers of high-quality, triangular prisms that conform to surfaces that are specified as level sets of an implicit function. Triangular prisms are important in circumstances where volumetric meshes need to capture the geometries materials with thin layers–*i.e.* that are smooth relative to their thickness–or where, as in fluid mechanics simulations, solutions perpendicular to boundaries exhibit boundary layer complexities that exceed that of the bounding surfaces. "High-quality" triangular prisms exhibit a roughly linear structure, with nearly regular triangles at the ends and side faces that are nearly perpendicular to the triangular faces. The proposed method relies on an iterative relaxation of point samples, which we call *dynamic particles*, that simultaneously regularize inter-point distance on the surfaces while shortening distances to corresponding points on a nearby offset surface, which establishes the layer. This paper describes the method and results on meshes of medical data sets that model human vasculature.

1 Introduction

This paper presents a method for computing thin layers of high-quality, triangular prisms that conform to surfaces that are specified as level sets of an implicit function. This problem statement is motivated by a number of applications in medicine and biology where simulations are conducted on geometric models derived from 3D images, such as CT, MRI, or various forms of microscopy. These applications, in fields such as electrophysiology [4], orthopedics, and cardiology [3], rely on a wide range of governing equations, such as electrostatics, fluid dynamics, solid mechanics, and chemical dynamics. Here we consider three particular driving problems: electrostatic simulations in the head for EEG, in which thin layers around the skin and skull present thin, layered geometries; hyperelastic simulations of mouse bone loading where stress boundary layers normal to the surface must be captured; and computational fluid dynamics of human vasculature, where meshes of complex, branching structures must be augmented with thin layers to capture the properties of fluid flow near boundaries. In this paper we present meshing results for the latter.

Figure 1(a)–(b) shows an example of a segmentation from head MRI that demonstrates the thin layers of tissue which can have very different electrical conductivity—an important consideration in the simulation of biolelectric fields. Figure 1(c)–(d) shows a segmentation and an adaptive mesh of a rela-

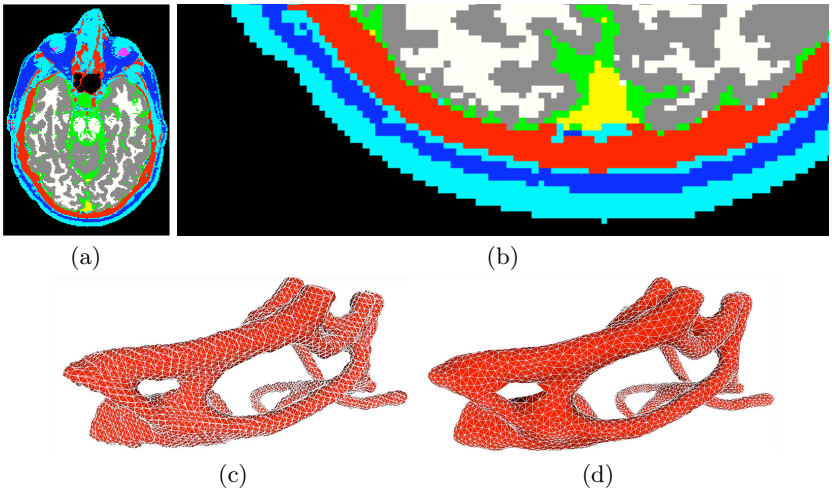


Fig. 1: (a)–(b) Segmentations of MRI head data show thin layers of skin, skull, fat and cerebral spinal fluid. (c) Isosurfaces of CT angiography give boundaries of vascular lumen, where blood flows. (d) Adaptive meshes using the method of dynamic particles [2] give good quality triangles and watertight surfaces.

tively small section of a CT data set from a human head. Fluid simulations rely on thin elements near the vessel boundaries to capture high gradients in flow.

One typical strategy for constructing such prisms is to start with a good quality triangle surface and simply offset the triangle vertices in the direction of the surface normals (established from the adjacent triangles, *e.g.* an average), and build prisms from the set of resulting triangle pairs. The first problem with this is that the prisms may not be valid. Here we define *validity* as nonself-intersecting prisms—that represent a single, connected solid. For instance, in regions of high curvature, the offsets that form the uprights of the prisms can cross the quadrilateral faces. The second problem is that even if the prisms are valid, the *quality* of the prisms could be compromised with this approach. This is because surface normals for each vertex are not parallel and because the inner and out triangular faces are not parallel. This can change the shape of the individual triangles and cause uprights that are skewed relative to the faces.

Figure 2a shows quantitative results of this strategy. For triangle quality we use the radius ratio: $Q = 3r/R$, where r and R are the radii of inscribing and circumscribing circles, respectively. This figure shows the effect of the offset on Q, as represented by histograms, which is quite good for the original triangle mesh (Fig. 2a—blue), but degrades significantly in the offset (Fig. 2a—red). The average radius ratio drops from 0.92 to 0.72. The angles between the quadrilateral faces and the triangles is $A = \text{asin}(\mathbf{u} \cdot \mathbf{n})$, where $\mathbf{u}$ is the unit vector along the quadrilateral edge, and $\mathbf{n}$ is the unit surface normal. Ideally,

this would be 90 degrees. The average angle is 74 degrees and the minimum is 0.2 degrees—there are some very skewed prisms. Finally, the number of incorrect prisms is significant—160 out of 11,854 prisms are of poor quality (1.3%) and would need to be corrected either by direct user interaction or some other mechanism. The purpose of this paper is to examine an alternative to this simple approach.

2 Methods

The meshing method uses a set of points, which we call *particles* [5, 1], whose positions are updated along the gradient of an objective function that prefers regular configurations. In previous work, we have shown that these objective functions can be constructed to achieve adaptive, high-quality sampling of implicit surfaces [2]. The objective function is a sum of local potential functions of the form $E(r_{ij}) \sim \mathrm{cotan}(\mathrm{r_{ij}}/\alpha)$, where r_{ij} is the distance between points (particles) i and j, $r_{ij} = |\mathbf{x}_i = \mathbf{x}_j|$. The total energy, for N particles is $P = \sum_{i=1}^{N} \sum_{j \neq i} E(r_{ij})$. The derivative of this energy (Figure 2c) is compact, and particles move with a gradient descent on this potential, constrained to the isosurface $V(\mathbf{x}) = k$.

For prisms, we establish a set of corresponding particles, with potential $P' = \sum_{i'=1}^{N} \sum_{j' \neq i'} E(r_{i'j'})$. These particles are constrained to an alternative isosurface $V(\mathbf{x}) = k'$, and we obtain an ϵ offset if $k' = k + \epsilon$ and $V(\mathbf{x})$ is a distance function to the original surface. Finally the two systems of particles are coupled, in order to keep correspondences close, through a quadratic energy, $S_i = |x_i - x'_i|^2$, and we have

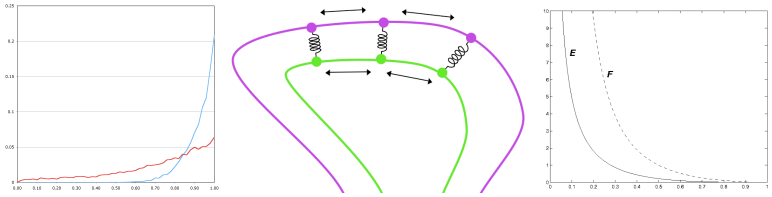
$$R = P + P' + \alpha \sum_{i=1}^{N} S_i, \tag{1}$$

as depicted in Figure 2b.

To initialize, we use the *simple* method described above, where we first optimize on the original surface, place twin particles on the offset surface, and then optimize the joint energy in Equation 1, which optimizes the particles on the two surface while keeping corresponding particles close. For all the results in this paper we use $\alpha = 10$. As in previous work [2], we insert and remove particles so that each particle (on the original surface) has a distance to its neighbors that is to within a specified tolerance of the local feature size. The local feature size is determined by measuring both the surface curvature and the distance to the medial axis.

3 Results and Conclusions

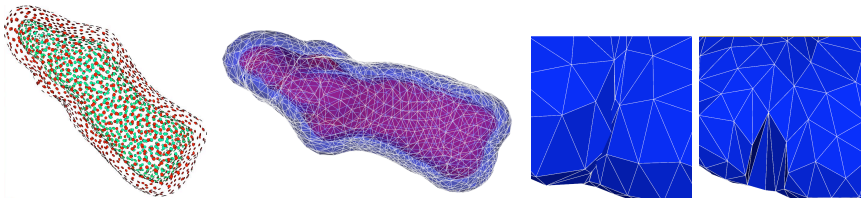
We begin with a simple geometric structure, which is a metacarpal of a mouse (finger/paw bone). Figure 3 shows a particle distribution from the proposed



(a) (b) (c)

Fig. 2: (a) Histograms of radius ratios of the original surface and offset (blue and red respectively), show a degradation of the triangle quality in the offset surface (Radius ratio of 1.0 is ideal). (b) Particles on each surface, the original and the offset, repel each other to maintain good configurations, but are attracted to the corresponding particle on the other surface. (c) The potential energy and repulsive force between particles falls off as *cotan*, approximating an electrostatic potential, but with compact support.

method and the resulting mesh. Table 3 summarizes the quantitative results. The proposed mesh eliminates the five poor quality prisms, improves the average and minimum radius ratios, but introduces slightly more *skew* in the prisms, as indicated by the average and maximum prism angles. Figure 3 shows the prism mesh of the vessel data. The quantitative results in Table 3 demonstrate an overall improvement but not the ideal results. The number of poor quality elements drops by 75% (160 to 40), and the triangles improve, but not entirely.

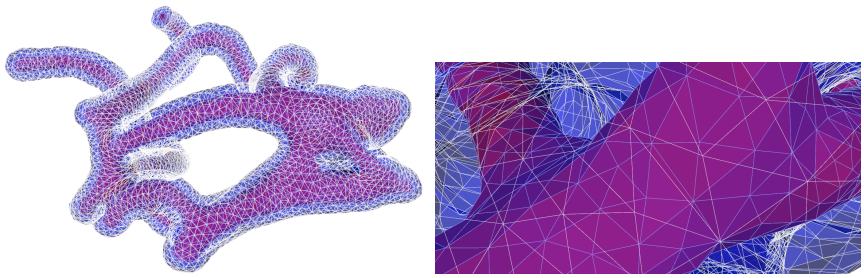


(a) (b) (c) (d)

Fig. 3: (a) Distribution of points/particles on a single surface—red and green are original and offset surfaces, respectively. (b) Prisms from the proposed method. (c) Offset surface from the *simple* method (normal offsets), show invalid (intersecting) elements. (d) Same view as (c) for the proposed method—showing valid prisms.

The results, therefore, are promising but mixed. In the case of invalid prisms, one can show that such prisms must always come in pairs, and that the energy of the system must be improved by swapping vertices. This suggests that the problem is the minimization, which is certainly getting trapped in local minima. Also, the parameters themselves—the tradeoff between particle repulsion and particle attraction—needs further attention. Another issue is that we use a DT triangulation method on the initial surface, and let the offset surface inherit this triangulation from the correspondences. We have noticed

that the triangles are consistently better on the initial surface, suggesting the need for a more holistic approach for the final meshing phase of the algorithm. Also, the offsets in these results are large relative to the surface geometry, which is intentional so that we could examine the behavior of the system as it fails. For the future, the integration of this system with the construction of multiple, thinner prismatic layers and tetrahedral volumes could help with the overall quality and allow better testing of performance.



Method	Rad Ratio Avg	Rad Ratio Min	Angle Avg	Angle Min	Invalid Elements
Mouse Bone					
Simple	0.87	3.0×10^{-2}	79.8	2.1×10^{-1}	5
Particle	0.88	7.2×10^{-2}	67.5	7.2×10^{-2}	0
Vessels					
Simple	0.82	4.9×10^{-4}	72.7	2.1×10^{-2}	160
Particle	0.83	4.0×10^{-3}	74.4	9.7×10^{-2}	40

References

1. M. Meyer, P. Georgel, and R. Whitaker. Robust particle systems for curvature dependent sampling of implicit surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI)*, pages 124–133, June 2005.
2. M. Meyer, R. M. Kirby, and R. Whitaker. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1704–1711, September/October 2007.
3. C. A. Taylor and M. T. Draney. Experimental and computational methods in cardiovascular fluid mechanics. *Annual Review of Fluid Mechanics*, 36:197–231, 2004.
4. R. V. Uitert, C. Johnson, and L. Zhukov. Influence of head tissue conductivity in forward and inverse magnetoencephalographic simulations using realistic head models. *IEEE Transactions on Biomedical Engineering*, 51(12):2129–2137, 2004.
5. A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM Press, 1994.

Multimaterial Meshing of MRI Head Data for Bioelectric Field Simulations

Ross Whitaker, Robert M. Kirby, Jeroen Sinstra, Miriah Meyer
Scientific Computing and Imaging Institute, University of Utah
Salt Lake City, UT 84112

1 Introduction

The problem of body fitting meshes that are both adaptive and geometrically accurate is important in a variety of biomedical applications in a multitude of clinical settings, including electrocardiology, neurology, and orthopedics. *Adaptivity* is necessary because of the combination of large-scale and small-scale structures (*e.g.* relatively small blood vessels spanning a human head). *Geometric accuracy* is important for several reasons. In some cases, such as computational fluid dynamics, the fine-scale structure of the fluid domain is important for qualitative and quantitative accuracy of the solutions. More generally, finite element approximations of elliptic problems with rough coefficients require increased spatial resolution normal to material boundaries [3]. The problem of constructing meshes from biomedical images is particularly difficult because of the complexity and irregularity of the structures, and thus tuning or correcting meshes by hand is quite difficult and time consuming. Many researchers and, indeed, commercial products simply subdivide the underlying hexahedral image grid and assign material properties to tetrahedra based on standard decomposition of each hexahedron into tetrahedra.

This paper presents a small case study of the results of a recently developed method for multimaterial, tetrahedral meshing of biomedical volumes [6]. The method uses an iterative relaxation of surface point point positions that are constrained to subsets of the volume that correspond to boundaries between different materials. In this paper we briefly review the method and present results on a set of MRI head images for use in bioelectric field simulation and source localization.

2 Mesh Generation Methodology

The goal of this paper is to examine the problem of forming anatomy-conforming tetrahedral meshes from segmented, three-dimensional MRI images of the human head for the purposes of finite element simulations of the electric fields. The pipeline for generating tetrahedral meshes from images consists of three parts: volume preprocessing, surface meshing, and volume meshing. This meshing method is presented in [6], but we briefly review it here for completeness.

We represent interfaces in a multimaterial dataset using a model that describes each material with a smooth, volumetric *indicator function*, f_i [4]. A

set of N indicator functions $F = \{f_i | f_i : V \mapsto \Re\}$ represents n materials. In practice these functions are formed by smooth interpolations of the output of segmented volumes or label maps, such as shown in Figure 1. For this work, we process each label map using a level-set implementation of the grayscale morphology algorithm proposed by Williams and Rossignac [10] called *tightening*, which limits the radius of curvature of the resulting boundary.

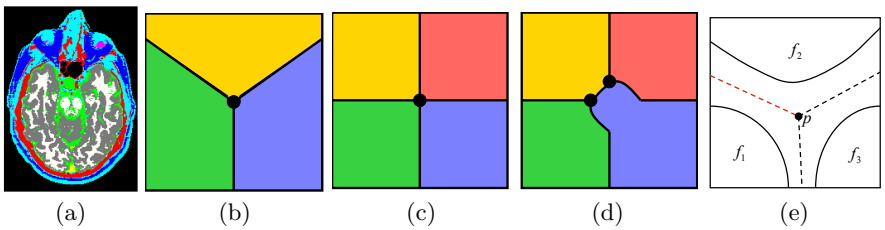


Fig. 1: (a) Multimaterial volumes, such as this segmentation of a head MRI are used to generate 3D meshes. (b) Generically, in 2D, two or three materials can meet (up to four in 3D). (c)–(d) Four material junctions are not generic, and result in generic conditions with small perturbations. (e) The generic case can be represented through smooth indicator functions.

A material label i is assigned to a point $x \in V$ if (and only if) $f_i(x) > f_j(x) \ \forall \ j \neq i$. In the case where $N > 2$, the boundaries that separate materials are not necessarily manifold, and can form sharp corners and edges. We characterize each material interface in terms of the number of material indicator functions that are maximal (and equal) at that junction. After this preprocessing, we can define *inside/outside* (IO) functions for each material so that the boundaries boundaries between different materials, represented as zero sets of the IO functions, coincide. We define these functions:

$$\tilde{f}_i = f_i - \max_{j=1, j \neq i}^{n} f_j. \tag{1}$$

Material junctions occur at points where two or more of the indicator functions (f_i) have equal, maximal value. We characterize the order of the junction by the number of coincident materials. For $V \subset \Re^2$, 2-junctions and 3-junctions occur generically, as shown in Figure 1 (b–c), while a 4-junction is a nongeneric case, as shown in Figure 1 (d–e). For $V \subset \Re^d$ each K-junction forms a subset of V that is topologically equivalent (homeomorphic) to a P-disk, where $P = d - K + 1$. Thus each type of material junction can be considered a P-cell, as described in the literature on discrete topology [1]. Generically, for $d = 3$ we have 4-junctions (0-cells), points; 3-junctions (1-cells), and 2-junctions (2-cells) or surfaces. For the case of the head data in this paper, we use six materials: grey matter, white matter, CSF, skin, bone, and air. Thus we have the possibility of 15 different types of 2-junctions (2-cells), 20 types of 3-junctions (1-cells), and 15 types of 4-junctions (0-cells).

The meshing method uses a set of points, which we call *particles* whose positions are updated along the gradient of a objective function that prefers regular configurations. In previous work, we have shown that these objective functions can be constructed to adaptive, high-quality sampling of implicit surfaces [5]. In this work each particle is constrained to a particular material junction. This formulation for each type of cell includes a set of projection operators that force particle motions to remain in the tangent space of the associated material boundary. A hierarchy of particle systems then samples each type of generically occurring material junction such that each junction is represented in the final mesh. That is, in 3D the 0-cells (points) are sampled first, followed by the 1-cells (curves), and concluding with the 2-cells (surfaces). Finally, a simple labeling algorithm extracts the multimaterial surface meshes as a subset of a Delaunay tetrahedralization of the samples, as described below.

To represent the interface between sets of materials, we define a *cell indicator function* that identifies points in V where a set of IO functions evaluate to zero, such as the material interface between materials 1 and 2 shown as the red dashed line in Figure 1. For a set of materials M (where $2 \leq |M| \leq 4$ generically in 3D), we have

$$J_M = \sum_{i \in M} \tilde{f}_i^2, \tag{2}$$

and the zero set of this function is the cell defined by the interface of the materials M.

We use the particle system framework of Meyer et. al [5, 6] for placing points along each material junction. This is an iterative algorithm that relaxes the particle positions, prior to a triangulation or meshing, in order to minimize an energy function that favors equal distributions of points or *particles*. This framework uses a *sizing field* that informs particles of how far they should be from their neighbors. We set this sizing field to be proportional to the *local feature size* (LFS) [2], which is the distance to the nearest point on the medial axis. In this way, the sizes of triangles on the surface meshes will be proportional to the sizes of the tetrahedra that will be needed to fill the adjacent volumes. We generate a sizing field volume, at the same resolution as the input data, for a multimaterial dataset, by first computing an approximate medial axis of each IO function. We then store at each grid point in the sizing field volume the minimum LFS for the set evaluated at the grid point location. We smooth this field using a gradient-limiting PDE proposed by Persson [7].

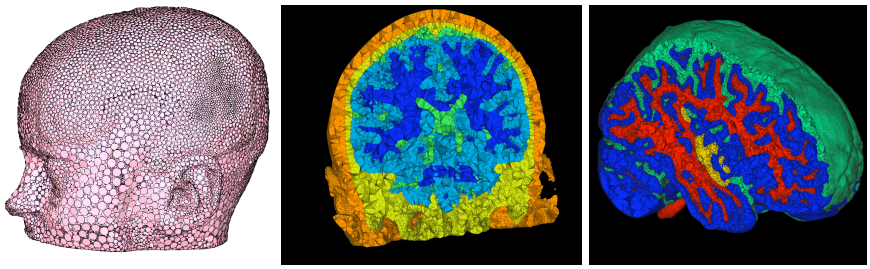
For the tetrahedral meshing we use the open-source software package *Tetgen*[1]. We run Tetgen in a constrained-Delaunay mode so that it adds points to the volume interior but does not add points to the surface. This Delaunay tetrahedralization (DT) uses the constrained DT method proposed in [9]. The Tetgen software includes a Delauney refinement approach [8] for ensuring mesh quality, as measured by *edge-radius ratios*, which does not, unfortu-

[1] http://tetgen.berlios.de

nately, penalize slivers. Tetgen does include a sliver removal step, consisting of edge-flips and peeling away of slivers near the boundary, which is only moderately effective.

3 Case Studies

In this section we present a small set of case studies that demonstrates the use of this meshing strategy for indicator functions derived from smoothed label maps of segmented, 3D, MRI images. The three data sets are from two pediatric patients suffering from neurological symptoms (*e.g.* seizures), ages 12 and 15, and a single normal volunteer, age 28, which we call cases 1, 2, and 3, respectively. The segmentations come from a semiautomated tissue classification algorithm followed by a manual inspection and hand editing of mislabeled pixels.



(a) (b) (c)

Fig. 2: (a) A particle distribution for Case 2 shows the adaptivity of the particles for thin regions of the skin/skull. (b) The tetrahedral mesh for Case 2. (c) The tetrahedral mesh for cortex and CSF of Case 2b, showing higher resolution.

Cases	1	2	2b (high res)	3
Num tris	265,989	450,945	3,510,322	321,532
Num tets	1,067,541	1,765,216	15,561,759	1,190,125
Tris RR avg(min)	0.89(0.031)	$0.89(1.8 \times 10^{-4})$	$0.93(2.6 \times 10^{-4})$	$0.89(7.6 \times 10^{-5})$
Tets RR avg(min)	0.61(0.013)	$0.61(1.0 \times 10^{-5})$	$0.64(2.0 \times 10^{-5})$	$0.62(5.1 \times 10^{-5})$

Each material is extracted into a binary volume, and a distance transform is constructed to the interfaces for that material, and the surface is smoothed with a *tightening* of radius 0.5, and approximate medial axes of each material are constructed by detecting coincidences of foot points on a grid with the same resolution as the original image. The feature size volume resulting from the distance to this medial axis is gradient limited (smoothed to enforce maximum on gradient values [7]) to a value of 0.5 (unitless), and then multiplied

by a value ϵ, which controls the resolution of the final mesh, is 2.5, except Case 2b, which is the high-resolution version of Case 2, with $\epsilon = 1.0$. Figure 2 shows surface and tetrahedral meshes that result from the proposed, particle-based multimaterial meshing method. Table 3 shows quantitative analyses of the triangles and tetrahedra for the three cases.

The results of this method are quite promising. The triangle mesh quality is excellent, especially given the adaptivity and high geometric accuracy exhibited by the method. The tetrahedral quality, as measured by radius ratio (which is indicative of the conditioning of our stiffness matrix in the resulting linear system), is not as good, but acceptable for simulations. The worst tetrahedra are slivers, which are an expected outcome of the use of Tetgen for the final volumetric step. Particularly important is that the system used one set of parameters for all three cases and required no manual intervention after the initial tuning of parameters. Future work will focus on the computation time, which is 8-12 hours for each of these datasets, and is mostly spent of preprocessing and distributing particles, and the tetrahedralization, which would benefit from one of the variety of method that explicitly reduces slivers.

References

1. A. Adamson and M. Alexa. Point-sampled cell complexes. *ACM Trans. Graph.*, 25(3):671–680, 2006.
2. N. Amenta, M. Bern, and D. Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphic Models and Image Processing*, 60(2):125–135, Mar. 1998.
3. I. Babuska, B. Andersson, B. Q. Guo, and H. Oh. Finite element method for solving problems with singular solutions. *J Computational and Applied Math*, 74:51–70, 1996.
4. B. Merriman, J. K. Bence, and S. J. Osher. Motion of multiple junctions: A level set approach. *J. Comput. Phys.*, 112(2):334–363, 1994.
5. M. Meyer, R. M. Kirby, and R. Whitaker. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1704–1711, September/October 2007.
6. M. Meyer, C. Ledergerber, H.-P. Pfister, and R. Whitaker. Particle-based sampling and meshing of multimaterial volumes. In *Proc IEEE Visualization*, page To appear, 2008.
7. P.-O. Persson. Mesh size functions for implicit geometries and pde-based gradient limiting. *Eng. with Comput.*, 22(2):95–109, 2006.
8. J. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proc Fourteenth Annual Sym on Computational Geometry*, pages 86–95, 1998.
9. H. Si and K. Gartner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proc 14th Int Meshing Roundtable*, 2005.
10. J. Williams and J. Rossignac. Tightening: Curvature-limiting morphological simplification. In *Proceedings of the Symposium on Solid and Physical Modeling*, pages 107–112, 2005.