

Research Article

A Novel Mesh Quality Improvement Method for Boundary Elements

Hou-lin Liu, Cui Dai, Liang Dong, and Ming-gao Tan

Research Center of Fluid Machinery Engineering and Technology, Jiangsu University, Zhenjiang, Jiangsu 212013, China

Correspondence should be addressed to Liang Dong, edongliang@yahoo.com

Received 18 June 2012; Revised 5 September 2012; Accepted 19 September 2012

Academic Editor: Mehmet Sezer

Copyright © 2012 Hou-lin Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to improve the boundary mesh quality while maintaining the essential characteristics of discrete surfaces, a new approach combining optimization-based smoothing and topology optimization is developed. The smoothing objective function is modified, in which two functions denoting boundary and interior quality, respectively, and a weight coefficient controlling boundary quality are taken into account. In addition, the existing smoothing algorithm can improve the mesh quality only by repositioning vertices of the interior mesh. Without destroying boundary conformity, bad elements with all their vertices on the boundary cannot be eliminated. Then, topology optimization is employed, and those elements are converted into other types of elements whose quality can be improved by smoothing. The practical application shows that the worst elements can be eliminated and, with the increase of weight coefficient, the average quality of boundary mesh can also be improved. Results obtained with the combined approach are compared with some common approach. It is clearly shown that it performs better than the existing approach.

1. Introduction

Numerical simulation is an important component in diverse activities such as medical imaging, engineering design, and cinematic special effects. These simulations routinely rely on tetrahedral meshes to model the physical domain of interest. The popularity of tetrahedral meshes stems from their ability to accurately model extremely complex geometries. High resolution tetrahedral meshes are often particularly desirable, as they can improve numerical accuracy greatly at previously infeasible scales. Fortunately, the power of modern computing and data acquisition technologies has enabled the production of tetrahedral models with enormous size. However, it is normally difficult to acquire mesh in which all the elements are suitable for numerical computation. Poorly shaped tetrahedra in a mesh can result in numerical errors and increase the time cost to find a solution [1, 2]. Hence, there is a market for mesh improvement tools which can enhance the quality of tetrahedra for an existing mesh.

Recently, there are two main techniques to improve the mesh quality. The first modifies topology by inserting or deleting nodes as well as changing connectivity of nodes [3, 4]. The operations include local face swapping and element or vertex insertion/deletion. The second, called smoothing method [5, 6], preserves mesh topology by applying appropriate node placement techniques. Mesh improvement techniques have been shown to be effective in improving tetrahedral meshes. However, most of them regard the boundary configuration of a mesh as “untouchable.” For example, vertices on the boundary cannot be smoothed. Moving the vertices on the boundary of a mesh is fraught with peril. In engineering applications, the mesh is considered to be noise-free and any change to the surface may induce some errors in the computational domain. For this, the quality improvement algorithms usually keep boundary vertices locked in place, which limits their ability to improve the quality of tetrahedra.

As to improving boundary quality of a surface mesh, considerable research has been conducted. Garimella et al. [7, 8] proposed an optimization-based vertex repositioning procedure to improve the quality of a surface mesh containing triangles and quadrilaterals. It has been shown that the method is capable of keeping the nodes on the original mesh faces and close to their original locations. Semenova and Savchenko [9] presented two novel techniques to improve the quality of triangle surface meshes while preserving surface characteristics as much as possible. Frey and Borouchaki [10] proposed a suitable method for the construction of an enriched geometric finite element mesh from a given arbitrary surface triangulation. The initial triangulation is optimized with respect to geometry and element shape quality. Actually, the surface mesh quality has an important effect on high quality solid mesh generation. Surface mesh quality improvement algorithms can reduce the difficulty of generating solid meshes and improving mesh quality. However, when it comes to numerical simulation, they cannot improve its accuracy efficiently.

In this study, we are more concerned with optimizing boundary vertices to obtain better quality tetrahedra, rather than surface meshes. However, little research has been done to improve boundary surface quality in solid meshes. Klingner [11] presented a quadric smoothing method for smoothing boundary vertices of surfaces in solid meshes. It permits all surface vertices to move, but they are encouraged to move along the original surface, and discouraged from making noticeable changes to the shape of the domain. By balancing tetrahedron quality against a quadric error measured at each vertex, the method controls the domain shape error. However, it cannot preserve mesh nodes on the original discrete surface and cannot guarantee boundary conformity in a real sense.

The goal of this paper is to improve the quality of boundary tetrahedra. It should be noted that, unlike Klingner’s work, surface vertices will not be moved. Topology optimization technique is adopted for converting elements that have all their vertices on the boundary into other type of elements, and the optimization-based smoothing algorithm is used for improving mesh quality. Section 2 briefly describes the overall combined algorithm. Section 3 introduces the modified optimization-based smoothing algorithm and Section 4 discusses the topology optimization technique. We then present the results of numerical experiments on several test meshes. Finally, Section 5 concludes the paper.

2. The Overall Combined Optimization Algorithm

Due to inherent defect, the optimization-based smoothing algorithm cannot improve boundary mesh quality through moving boundary nodes without distorting discrete surface.

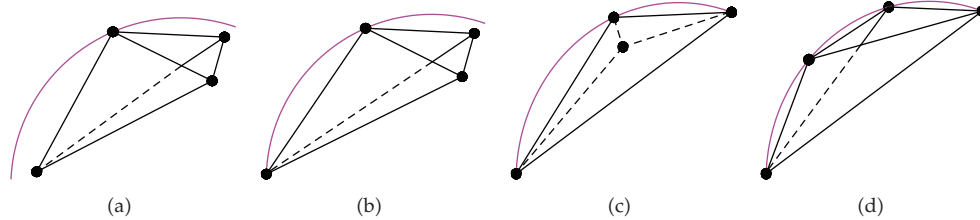


Figure 1: Four kinds of boundary elements.

The boundary conformity cannot be guaranteed when surface nodes are permitted to move. For example, the elements in Figures 1(a), 1(b), and 1(c) might be improved by moving a node if the node is an interior one. However, for the element in Figure 1(d), there are no interior nodes to move to improve its quality. It is called “all boundary element” in the paper. The quality improvement for this type of element with all their nodes on the boundary surface is beyond the capability of smoothing. Therefore, there are a lot of poor quality elements on the boundary after performing the optimization-based smoothing algorithm. To address this problem, a combined optimization algorithm based on optimization-based smoothing and topology optimization algorithm is proposed. The algorithm can improve the overall and boundary mesh quality and won't destroy boundary integrity of the initial mesh.

The overall scheme is presented. Algorithmic details for each of the major steps in this scheme will be presented later in the paper.

Input: initial mesh, poor quality threshold value γ , iteration number n , objective function $E(\Gamma)$, and weight coefficient C .

Output: high quality mesh.

- (1) Compute the number of nodes and elements in initial mesh. If the element is a boundary one, mark the element with the boundary;
- (2) calculate the mesh quality of initial mesh based on the quality measure α (see Section 3.1), and store the nodes of poor quality element in Φ , whose quality is less than γ ;
- (3) in Φ , select a node randomly and determine whether it is a boundary node. If the node is a boundary one and all its neighboring nodes are on the boundary, go to Step (4). Otherwise, carry out Step (5).
- (4) Apply *Topology optimization algorithm* (see Section 4) to convert elements that have all their vertices on the boundary into other type of elements, and go to Step (6).
- (5) Apply *improved optimization-based smoothing algorithm* (see Section 3) to optimize mesh.
- (6) Repeat Steps (3) to (5) until there are no poor quality elements in Φ .

3. The Improved Optimization-Based Smoothing

3.1. Objective Function

Through studying the quality measures and optimization algorithms of tetrahedral meshes, the error function is derived by means of transforming the measures. The element' distortion

is disposed as the error. And the larger the distortion is, the bigger the value of the error. The maximum value of the error function is infinite, and the minimum is zero. The total error function of the mesh is the sum of the elements' errors, and it is adopted as the objective function of the optimization-based smoothing. The minimum value of objective function is solved for improving the mesh quality.

Generally speaking, a reasonable quality measure for elements should possess the following attributes.

- (1) It is invariant under translation, rotation, and scaling.
- (2) Normalization by an optimal value within a range $[0,1]$, where 1 is for an equilateral tetrahedron and 0 is for a degenerate tetrahedron.
- (3) Ability to detect all possible badly shaped elements.

So an optimal quality measure α is employed, and an error function is derived based on it:

$$\alpha = \frac{6\sqrt{6}V}{\left(\left(\sum_{i=1}^4 S_i\right) \times \max_{i=1,\dots,6} L_i\right)}, \quad (3.1)$$

where V is the volume of tetrahedron, S_i is the surface area of a triangular facet, and L_i is the length of any edge i .

The reciprocal of (3.1) is the error function of an element:

$$e(T) = \frac{\left(\sum_{i=1}^4 S_i\right) \times \max_{i=1,\dots,6} L_i}{6\sqrt{6}V}, \quad (3.2)$$

where T denotes a tetrahedron.

When the element is a regular tetrahedron, the value of the function is 1. As one element degenerates, the value tends to be infinite. If the element is reverse or the volume is negative, the value of error is also infinite. The total error function of the mesh is defined as $E(\Gamma)$, as shown in the following:

$$E(\Gamma) = \sum_{T \in \Gamma} e(T). \quad (3.3)$$

$E(\Gamma)$ is the objective function of the optimization-based smoothing. In order to control the boundary and interior mesh quality simultaneously, the objective function is expressed as follows:

$$f(\Gamma) = (1 - C) \sum_{T \in \Gamma_1} e(T) + C \sum_{T \in \Gamma_2} e(T). \quad (3.4)$$

where Γ is the set of all elements, Γ_1 is the set of all interior elements, Γ_2 is the set of all boundary elements, and C ($0.5 < C < 1$) is the weight coefficient which indicates the proportion of boundary mesh quality with respect to the overall. The same method is executed for boundary and interior mesh when $C = 0.5$. Through (3.4), the interior and boundary mesh quality is considered simultaneously.

- (1) **function SOLUTION** ($f(x), \lambda_0, \varepsilon, X_n^0, N$)
 - ▶ $f(x)$ = objective function
 - λ_0 = initial step size
 - ε = desired degree of accuracy
 - N = iteration number
 - X_n^0 = initial coordinate for vertices X
- (2) get free vertex H_n^0 and $k = 0$
 - ▶ H_n^0 = Hessian matrix
 - k = Iteration number
- (3) $\min f(X_n^k + \lambda_k d_n^k)$
 - ▶ d_n^k = optimal search direction
- (4) compute λ_k^* using line search algorithm (see Algorithm 2)
 - ▶ λ_k^* = optimal step size
- (5) $X_n^{k+1} = X_n^k + \lambda_k^* d_n^k$
- (6) if $|f(X_n^{k+1}) - f(X_n^k)| < \varepsilon$ or $k > N$ then
- (7) $X_n^* \leftarrow X_n^{k+1}$ and **break**
 - ▶ X_n^* = optimal point coordinate
- (8) else $\Delta X_n^{k+1} = X_n^{k+1} - X_n^k, \Delta g_n^k = g_n^{k+1} - g_n^k$
 - ▶ $g_n^{k+1} = f(x)$ gradient at vertices X_n^{k+1}
 - $H_n^{k+1} = (H_n^k + \Delta X_n^k (\Delta X_n^k)^T / (\Delta X_n^k)^T \Delta g_n^k) + (H_n^k \Delta g_n^k (\Delta g_n^k)^T H_n^k / (\Delta g_n^k)^T H_n^k \Delta g_n^k)$
 - $d_n^{k+1} = -H_n^{k+1} g_n^{k+1}, k = k + 1$
 - go to step (3)
- (9) **end function**

Algorithm 1: Nonsmooth optimization algorithm.

3.2. Optimization Algorithm

An efficient and robust solver for the large system of equations presented by the optimization problem is needed. Better smoothing algorithms are based on numerical optimization [12, 13]. Early algorithms define a smooth objective function that summarizes the quality of a group of elements (e.g., the sum of squares of the qualities of all the tetrahedra adjoining a vertex) and use a numerical optimization algorithm such as steepest descent or Newton's method to move a vertex to the optimal location. Freitag et al. [14] proposes a more sophisticated nonsmooth optimization algorithm, which makes it possible to optimize the worst tetrahedron in a group, for instance, to maximize the minimum dihedral angle among the tetrahedra that share a specified vertex. A nonsmooth optimization algorithm is needed because the proposed objective function is not a smooth function of the vertex coordinates in this paper. The gradient of this function is discontinuous wherever the identity of the worst tetrahedron in the group changes. Pseudopodia for nonsmooth optimization algorithm are presented in Algorithm 1.

It is well known that the line search methods play a pivotal role on optimization problems. Locating a local minimum in the optimization problem with no constrains are prepared. All methods have the basic structure in common. In each iteration, a direction d_n^k is chosen from the current location X_n . The next location, X_{n+1} , is the minimum of the function along the line that passes through X_n in the direction d_n^k . The line search algorithm is conducted in Algorithm 2. In order to find the minimum of a function $f(X) : R \rightarrow R$, we need to bracket it. To bracket a minimum means fining a triple $a, b, c \in R, a < b < c$, so that

```

(1) function LINE_SEARCH ( $\varepsilon_1, \varphi(f(x), X_n^0, h_0, \alpha)$ )
    ►  $f(x)$  = objective function
     $\varepsilon_1$  = allowable error
     $\varphi(f(x), X_n^0, h_0, \alpha)$  = the function for search region
        [ $a_1, b_1$ ] see below Algorithm 3
(2) compute  $\mu_1 = a_1 + 0.382(b_1 - a_1)$  and  $\nu_1 = a_1 + 0.618(b_1 - a_1)$ 
    ►  $\mu_1, \nu_1$  = initial tentative point
        Set  $i = 1$ 
         $i$  = iteration number
(3) if  $|\mu_i - \nu_i| < \varepsilon_1$  then
(4) return  $\lambda_i^* = (\mu_i + \nu_i)/2$  and break
    ►  $\lambda_i^*$  = optimal step size
(5) else if  $f(\mu_i) < f(\nu_i)$  go to step (7)
(6) else  $f(\mu_i) \geq f(\nu_i)$  go to step (8)
(7) set  $a_{i+1} = a_i, b_{i+1} = \nu_i, \nu_{i+1} = \mu_i, f(\nu_i) = f(\mu_i)$ 
(8) compute  $\mu_{i+1} = 0.618a_{i+1} + 0.382b_{i+1}$  and  $f(\mu_{i+1})$ 
(9)  $i = i+1$  go to step (3)

```

Algorithm 2: Line search algorithm.

```

(1) function SEARCH_INTERVAL ( $f(x), X_n^0, h_0, \alpha$ )
    ►  $f(x)$  = objective function
     $X_n^0$  = initial coordinate for vertices X
     $h_0$  = initial step size
     $\alpha$  = coefficient larger than 1
(2) compute  $f_0 = f(X_n^0)$  and Set  $j = 0$ 
    ►  $j$  = iteration number
(3)  $X_n^{j+1} = X_n^j + h_j$  and compute  $f(X_n^{j+1})$ 
(4) if  $f(X_n^{j+1}) < f(X_n^j)$  then go to step (6)
(5) else go to step (7)
(6)  $h_{j+1} = \alpha h_j, X_n^* = X_n^j, X_n^j = X_n^{j+1}$ 
     $f(X_n^j) = f(X_n^{j+1}), j = j + 1$ 
(7) if  $j = 0$  then  $h_j = -h_j, X_n^* = X_n^j$  go to step (3)
(8) else  $a = \min\{X_n^*, X_n^{j+1}\}, b = \max\{X_n^*, X_n^{j+1}\}$ 
(9) return  $a$  and  $b$ 

```

Algorithm 3: Interval search algorithm.

$f(a) < f(b)$ and $f(b) < f(c)$. This indicates that the minimum is in the interval $[a, c]$. The interval search algorithm is in Algorithm 3.

4. Topology Optimization

Topology optimization algorithm is employed, which converts boundary elements that have all their nodes on the boundary into other types of boundary elements. Furthermore, the topology optimization does not necessarily improve the quality of elements, but

- (1) Find all the elements which include the node in the all boundary elements, and E denotes those elements.
 ▶ EG : Initial all boundary elements
- (2) Edge removal can be performed if an edge in the EG is shared by three tetrahedra.
- (3) Edge Swapping can be performed If an edge in the EG is shared by four or more tetrahedra.
- (4) Face Swapping can be performed If a tetrahedron and E share a face
- (5) end function

Algorithm 4: Topology optimization algorithm.

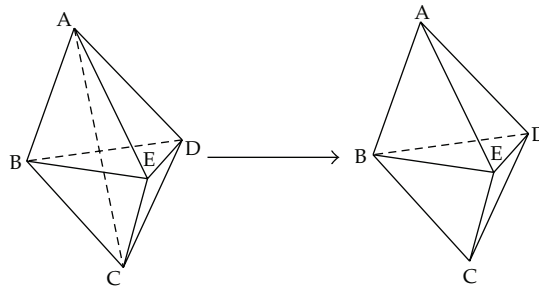


Figure 2: Edge removal.

those boundary elements must be eliminated. Pseudopodia for topology optimization are presented in Algorithm 4.

4.1. Edge Removal

Proposed by George [15], edge removal is a topological transformation that removes a single edge from the mesh, along with all the tetrahedra that include it. Figure 2 illustrates replacing three tetrahedra with two. The three tetrahedra ($ABCD$, $ABCE$ and $ACDE$) share the edge AC , and $ABCD$ tetrahedron is all boundary element. When AC is removed, the three tetrahedra in the region are transformed to two tetrahedra $ABDE$ and $BCDE$.

4.2. Edge Swapping

Edge swapping is a more complicated procedure [4]. If there are k tetrahedra containing an interior edge e in the mesh, then e is removed and the original k tetrahedra are replaced with $2k-4$ tetrahedra. In Figure 3, the edge TB is perpendicular to the page, and the five tetrahedra ($01BT$, $12BT$, $23BT$, $34BT$, and $40BT$) originally incident to edge TB can be replaced by six new tetrahedra: $012T$, $021B$, $024T$, $042B$, $234T$, and $234B$, where $01BT$ is all boundary element.

4.3. Face Swapping

Face swapping changes the local connectivity of a simplified mesh which converts all boundary elements into other boundary elements. Each interior face in a tetrahedral

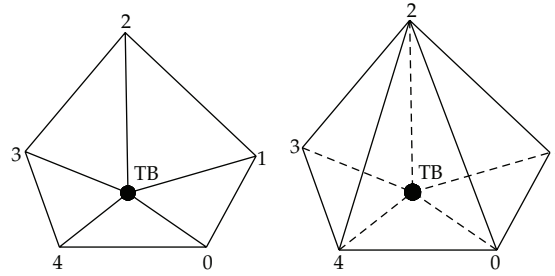


Figure 3: Edge swapping.

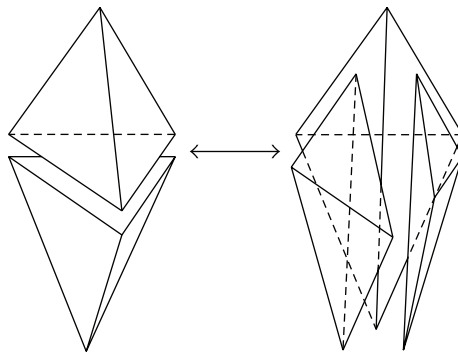


Figure 4: Face swapping configuration of five points in three dimensions.

mesh separates two tetrahedra consisting of a total of five points. A large number of nonoverlapping tetrahedral configurations can be formed with these five points, but only two of them can be reconnected satisfactorily. Figure 4 shows the face swapping configuration of five points. It is evident that either two or three tetrahedra can be used to fill the convex hull of a set of five points. Switching from two to three tetrahedra requires an additional edge interior to the convex hull. So all boundary elements can be converted into other type of boundary elements.

5. Numerical Experiments

Figure 5 shows the distribution of poor quality elements for an impeller in centrifugal pump before and after optimization. The quality for the worst element is 0.08 for initial mesh, 0.15 for $C = 0.6$, and 0.16 for $C = 0.8$ based on the quality measure α . So the improved optimization-based smoothing algorithm can improve the quality of the worst elements in a centrifugal impeller. Furthermore, through increasing the weight coefficient, the quality of boundary and overall elements is improved.

Figure 6 shows the effect of weight coefficient on optimization time. After increasing the weight coefficient C , the optimization time for the overall and boundary mesh ascends dramatically. That is because, to obtain optimal solution of objective function, we need to reduce the value of objective function through adjusting interior nodes' locations. So, it needs more iteration number (i.e., costing more time) when the value of objective function is larger.

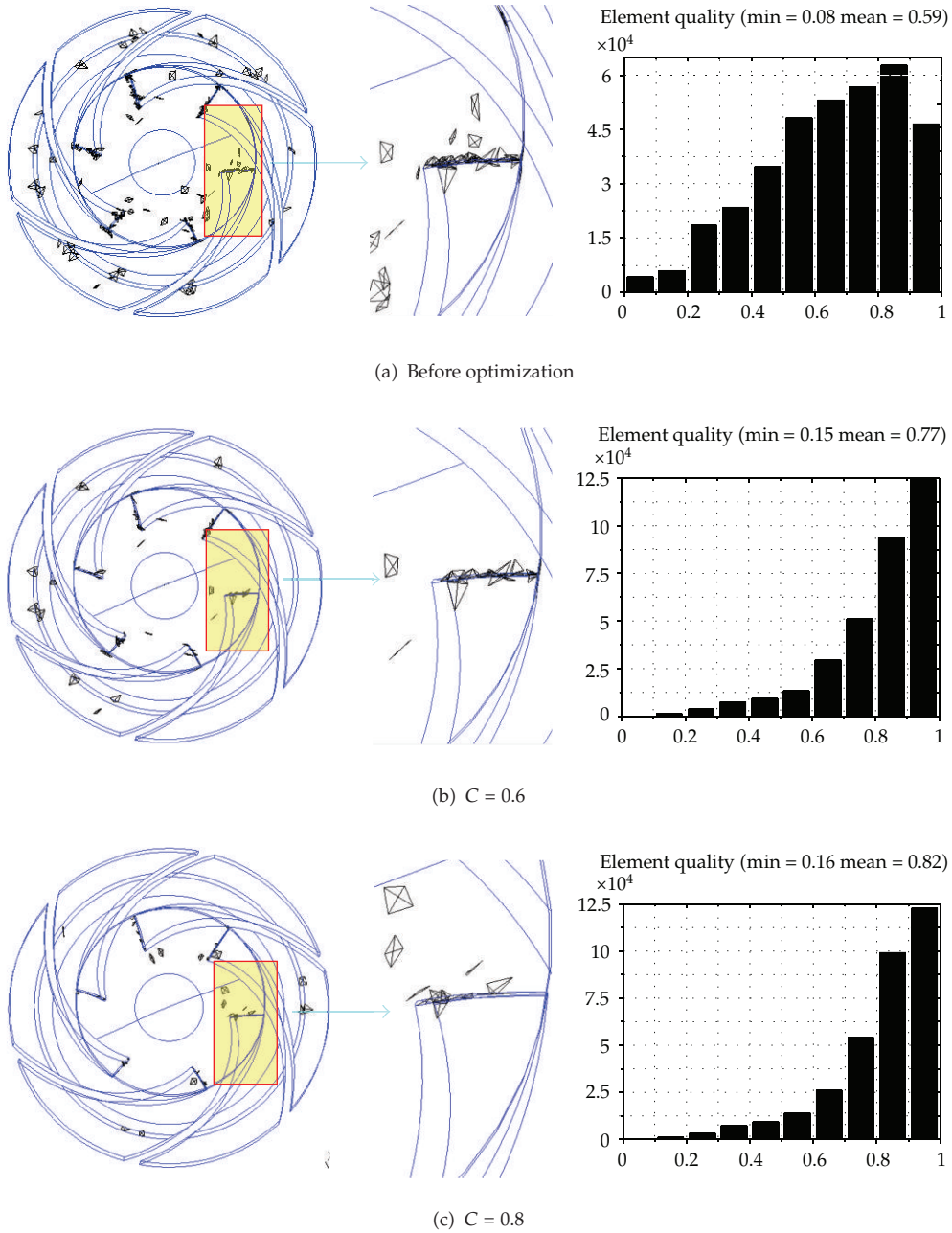


Figure 5: Distribution of poor quality elements.

The conclusion can be drawn that the proposed algorithm can dramatically reduce the number of poor quality elements in boundary and interior meshes. With the increase of weight coefficient, the overall and boundary mesh quality rises, while the time consumed increases rapidly. So in consideration of optimizing effect and efficiency, it is recommended the weight coefficient lies in the range of 0.7~0.8.

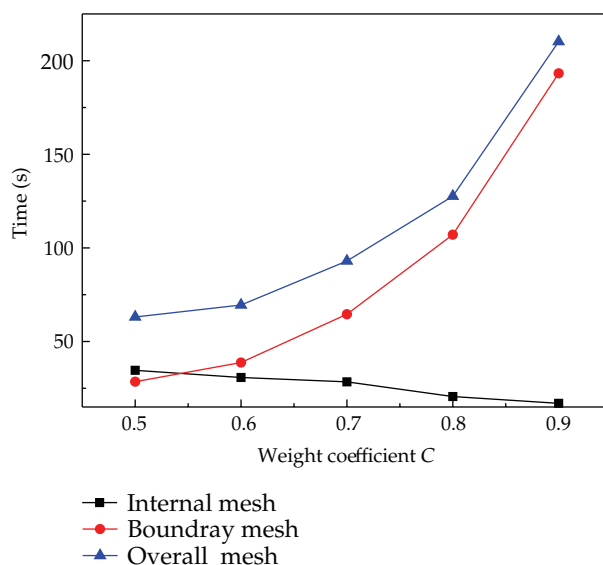


Figure 6: Time for different weight coefficients.

Table 1: Statistics of the examples before optimization.

Model	Mesh size		Overall mesh quality		Boundary mesh quality	
	Vertex number	Elem#	Worst	Average	Worst	Average
Impeller	77710	428049	0.0052	0.5105	0.0052	0.4823
Volute	36994	229101	0.0039	0.5074	0.0039	0.4761

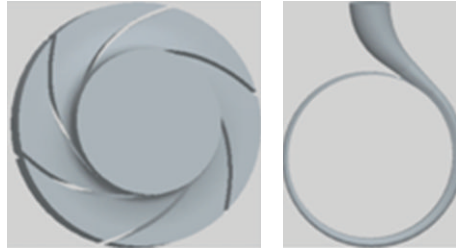
Figure 7 shows two examples to illustrate the effect of different mesh quality improvement algorithms. In Table 1, the number of nodes and elements in each mesh, and the initial mesh quality measured by α are given.

In Table 2, the results of each mesh quality improvement technique in terms of the quality of the overall and boundary mesh after optimization are reported. The time consuming is also given. It can be seen that, for the two algorithms, the quality of overall and boundary elements can be improved. And, the combined algorithm performs better in the quality improvement, while it consumes 1.3 times more time than the improved smoothing algorithm.

To evaluate our mesh improvement algorithm, comparison with Freitag and Ollivier-Gooch's method is made. And two cases of mesh (TIRE and RAND2) come courtesy of Freitag and Ollivier-Gooch [4]. TIRE is a tire incinerator with 2,570 vertices and 11,099 tetrahedral elements. Its initial mesh quality can be found in Table 3. RAND2 are lazy triangulations generated by inserting randomly located vertices into a cube one by one. Each vertex is inserted by splitting one or more tetrahedra into multiple tetrahedra. The random meshes have horrible quality and poor dihedral angles at both extremes. The RAND2 mesh has 5,086 vertices and 25,704 tetrahedral elements; its initial quality can be found in Table 3. Table 3 compares the minimum and maximum dihedral angles reported by Freitag and Ollivier-Gooch to that achieved by our mesh improvement algorithm. It can be seen that the mesh quality is bad before optimization, and there are badly shaped elements whose dihedral

Table 2: Statistics of the examples after optimization.

Model	Method	Overall mesh quality		Boundary mesh quality		Time/s
		Worst	Aver	Worst	Aver	
Impeller	Smoothing	0.0339	0.5760	0.0339	0.5360	122.9
	Combined	0.0671	0.5937	0.0671	0.5494	149.8
Volute	Smoothing	0.0359	0.5679	0.0359	0.5486	102.8
	Combined	0.0593	0.5861	0.0593	0.5603	128.7



(a) Impeller

(b) Volute

Figure 7: Models for mesh generation.

angles tend to be 0° or 180° . Dihedral angles are improved to be between 14° and 160° for Freitag and Ollivier-Gooch's algorithm with both swapping and smoothing, and between 23° and 141° for our proposed algorithm.

The two mesh optimization algorithms have been implemented and tested for RAND2 with the distribution of dihedral angles, as shown in Figure 8. Figure 8(a) shows the initial RAND2 mesh, and Figures 8(b) and 8(c) show the optimized mesh by Freitag algorithm and the proposed algorithm, respectively. It can be seen that they can both successfully eliminate poorly shaped elements from the mesh. Comparing with Freitag algorithm, the proposed algorithm is more successful in eliminating poor dihedral angles at both extremes.

6. Conclusions

A new combined algorithm based on optimization-based smoothing and topology optimization is proposed for boundary mesh quality improvement. Error functions of elements determined based on an inverse of measure α compose the objective function of a modified optimization-based smoothing. In the objective function, the function terms considering boundary and interior mesh quality are included, and a weight coefficient controlling boundary mesh quality is added, too. By means of minimizing an error function in local iterations, the modified smoothing algorithm can eliminate poor quality elements and improve the overall and boundary mesh quality as much as possible. Topology optimization is employed, by which those bad elements with all their vertices on the boundary are converted into other types of elements whose quality can be improved by the modified smoothing algorithm. Through comparison to Freitag and Ollivier-Gooch's method with RAND2 mesh used as example, the combined algorithm is confirmed to be capable of improving the boundary mesh quality considerably without distorting discrete surface. With the increase of weight coefficient, the overall and boundary mesh quality rises, while

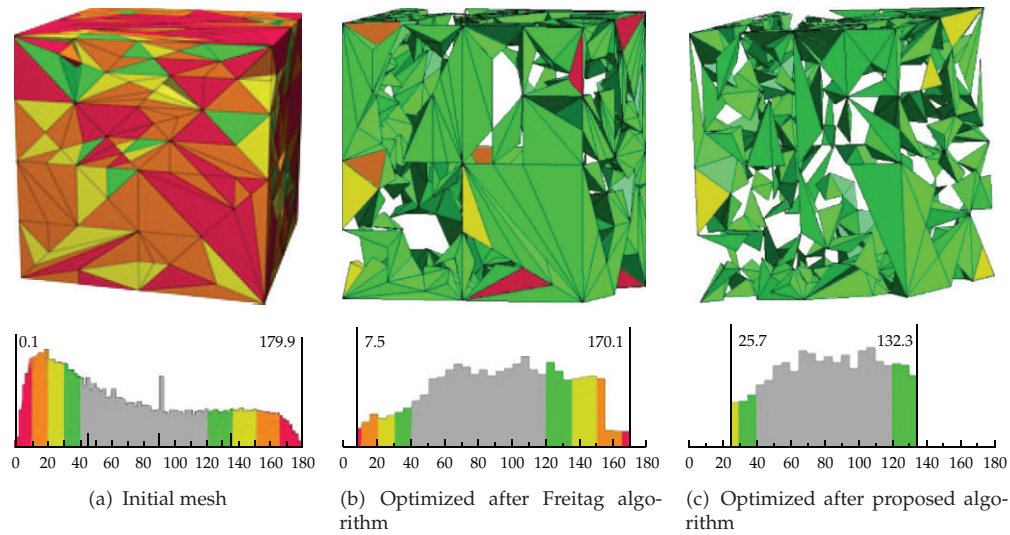


Figure 8: Mesh quality improvement for RAND2 with two algorithms.

Table 3: Statistics of the examples before and after optimization.

Model	Method	Before optimization		After optimization	
		Min	Max	Min	Max
TIRE	Freitag	0.66°	178.88°	13.67°	159.82°
	Combined			22.57°	141.54°
RAND2	Freitag	0.10°	179.84°	7.50°	170.09°
	Combined			25.69°	132.27°

the improvement time increases rapidly. To strike a balance between optimizing effect and time, the weight coefficient is suggested in the range of 0.7~0.8.

Acknowledgments

This work was supported by The National Natural Science Foundation of China (no. 50825902, 51079062, 51109095, and 51179075), a Project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Natural science fund in Jiangsu Province (BK2010346, BK2009006), and Postgraduate Innovation Foundation of Jiangsu Province (CXLX11_0576).

References

- [1] T. Munson, "Mesh shape-quality optimization using the inverse mean-ratio metric," *Mathematical Programming*, vol. 110, no. 3, pp. 561–590, 2007.
- [2] J. Park and S. M. Shontz, "Two derivative-free optimization algorithms for mesh quality improvement," *Procedia Computer Science*, vol. 1, no. 1, pp. 387–396, 2010.
- [3] H. Edelsbrunner and N. R. Shah, "Incremental topological flipping works for regular triangulations," *Algorithmica*, vol. 15, no. 3, pp. 223–241, 1996.

- [4] L. A. Freitag and C. Ollivier-Gooch, "Tetrahedral mesh improvement using swapping and smoothing," *International Journal for Numerical Methods in Engineering*, vol. 40, no. 21, pp. 3979–4002, 1997.
- [5] K. Xu, Z. Q. Cheng, Y. Wang, Y. Xiong, and H. Zhang, "Quality encoding for tetrahedral mesh optimization," *Computers and Graphics*, vol. 33, no. 3, pp. 250–261, 2009.
- [6] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun, "Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation," *ACM Transactions on Graphics*, vol. 28, no. 3, article 75, pp. 1557–1571, 2009.
- [7] R. V. Garimella, M. J. Shashkov, and P. M. Knupp, "Triangular and quadrilateral surface mesh quality optimization using local parametrization," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 9–11, pp. 913–928, 2004.
- [8] R. V. Garimella and M. J. Shashkov, "Optimization of surface mesh quality using local parameterization," in *Proceedings of the 11th International Meshing Roundtable*, pp. 41–52, 2002.
- [9] I. B. Semenova and V. V. Savchenko, "Two techniques to improve mesh quality and preserve surface characteristics," in *Proceedings of the 13th International Meshing Roundtable*, pp. 37–48, 2004.
- [10] P. J. Frey and H. Borouchaki, "Geometric surface mesh optimization," *Computing and Visualization in Science*, vol. 3, no. 1, pp. 113–121, 1998.
- [11] B. M. Klingner, *Improving Tetrahedral Meshes*, University of California, 2008.
- [12] S. A. Canann, M. B. Stephenson, and T. Blacker, "Optsmoothing: an optimization-driven approach to mesh smoothing," *Finite Elements in Analysis and Design*, vol. 13, no. 2-3, pp. 185–190, 1993.
- [13] V. N. Parthasarathy and S. Kodiyalam, "A constrained optimization approach to finite element mesh smoothing," *Finite Elements in Analysis and Design*, vol. 9, no. 4, pp. 309–320, 1991.
- [14] L. Freitag, M. Jones, and P. Plassmann, "An efficient parallel algorithm for mesh smoothing," in *Proceedings of the 4th International Meshing Roundtable*, pp. 47–58, 1995.
- [15] P. L. George, "Optimization of tetrahedral meshes. Modeling, mesh generation, and adaptive numerical methods for partial differential equations," *IMA Volumes in Mathematics and its Applications*, vol. 75, pp. 97–128, 1995.