A Quality and Efficient Tetrahedral Mesh Generation Method

Cheng Huang[1], Jianming Zhang[1], Luping Liu[2], Guangyao Li[1]

[1] *State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, College of Mechanical and Vehicle Engineering, Hunan University, Changsha, 410082 China*

[2] *Hydrochina Zhongnan Engineering Corporation, Changsha 410014 China*

Correspondence to: Jianming Zhang
College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China
Telephone: +86-0731-88823061
E-mail: zhangjm@hnu.edu.cn

Abstract

A method of unstructured tetrahedral mesh generation for general three-dimensional domain of arbitrary shape is presented. The method consists of two steps. First, generating the boundary tetrahedral mesh by the classic Delaunay method, then, using the advancing front method to create internal points and inserting them into the mesh by the Delaunay kernel procedure. The algorithm combines the advantages of the high point placement quality of the advancing front method and the high efficiency and convergence guaranty of the Delaunay algorithm. To get better efficiency, a new front identification and field point generation method is proposed and applied. Several examples have demonstrated the computational efficiency of our method and the high quality of meshes that generated within a reasonable time limit.

Key words: tetrahedral mesh generation, Delaunay method, advancing front method

1. Introduction

Many physical phenomena in science and engineering can be modeled by partial differential equations (PDEs). These equations can be solved by numerical methods, for instance, the finite element method (FEM), the finite volume method (FVM) and the boundary element method (BEM). The first step of using these methods is to partition the problem domain (or in the BEM, just the boundary of the problem

domain) into small pieces of simple shape. These pieces are called elements, and are usually triangles or quadrilaterals (in two dimensions), or tetrahedral or hexahedral bricks (in three dimensions). Because of the high ability to adaptive the arbitrary shape of the problem domain, triangle and tetrahedron are widely used. The problem of triangular and tetrahedral mesh generation is of considerable interest in engineering. Elements are usually obtained by mesh generation procedures. For practical simulation, it is desirable to perform the mesh generation automatically. Many mesh generation methods have been proposed. Among these methods, the advancing front method [1-3] and the Delaunay algorithm [4-6] are most popular for unstructured mesh generation.

This paper presents a new quality and efficient Delaunay tetrahedral mesh generation method coupled with the advancing front method. The combined method has the high quality point placement strategy of the advancing front method and the high efficient and the mathematical properties of the Delaunay approach. The method that combined the Delaunay and advancing front method has been described first by Rebay [6] and Muller et al. [7]. And Pascal J. Frey [8] extends this method to three dimensions. In addition to their work, we use a new efficient front identification and point generation procedure to further improve the efficiency. Our method can generate high quality mesh within a reasonable time limit. And high quality tetrahedral mesh can be generated by our method even in the case where the gradient of controlling element size is very steep. The resulting tetrahedral mesh will be used in the boundary face method (BFM) [9] to obtain and display the results of the arbitrary internal points. We have developed an interface between BFM and UG-NX(R) (Unigraphics NX). We have also developed an adaptive triangular mesh generation method over an arbitrarily parametric surface based on the advancing front method combined with a quad-tree procedure to generate boundary triangular mesh [10]. After surface triangular mesh generation, the points of the boundary are inserted using the Delaunay kernel to obtain the boundary tetrahedral mesh. Then, internal points are iteratively created using the advancing front method and inserted into the existing mesh with the Delaunay insertion algorithm. In our method, the front identification and field point creation process is different from the previous work. We define the front which is suitable for creating a new point as active front. Therefore, not all fronts are used to create a new point, just the active fronts. So, our method is more efficient than the previous work. The mesh size is controlled by the control space [11]. The final mesh

quality is improved by optimization techniques.

In the following sections, section 2 describes the general scheme of the mesh generation algorithm. Within section 3 the creation of field point and the identification of the front are described. The mesh control is discussed in section 4. Section 5 provides three application examples and in section 6 the current work is summarized.

2. Scheme of the method

The most popular approaches to triangular and tetrahedral mesh generation can be divided into two classes: Delaunay triangulation methods and advancing front methods. Usually, these methods start from the discretization of the domain boundary. The domain boundary is represented by a set of edges and faces in three dimensions.

There are many algorithms for Delaunay triangulation. However, the most used practical algorithm, applicable in both two and three dimensions, is the Bowyer-Watson algorithm [4-6]. This method adds points sequentially into an existing Delaunay triangulation. It usually starts from a very simple triangulation enclosing all the points to be triangulated. The basic operation of this method is the Delaunay insertion of a vertex into the existing triangulation. Take two-dimensional case as an example. At the beginning of the Delaunay insertion of a vertex, find all existing triangles whose circumcircle contains the new point. First, the triangle which contains the new point has to be found. Then the neighbours of this triangle are searched and then their neighbours, etc., until no more neighbours have the new point in their circumcircle. Delete these triangles, which create (always) a convex cavity. Join the new point to all the vertices on the boundary of the cavity. After all of the field points have been inserted, the boundary is recovered. Then we get the final mesh.

Subsequently we describe the conventional advancing front method. The advancing front method starts from the discretization of the domain boundary. A base front is selected from the set of fronts. A new tetrahedron is generated by the base front with a new point. The new point is created by the base front. After a new tetrahedron is generated, the old fronts are deleted and new fronts are added. As this procedure is repeated, the front advances over the domain and changes continuously. Repeat the creation loop as long as the front is not empty [12].

The advancing front method has the high quality point placement strategy. But this method is very time consuming. The point creation process requires the determination

whether the new element intersects any existing element and if so, whether the new element is too close to any existing element. Most of the time is spent on these determinations. It is a complex task to design an efficient and robust advancing front method program satisfying the desirable criteria.

In an attempt to overcome this difficult task, we suggest to use the advancing front method as an internal point creation tool, the Delaunay algorithm as an insertion tool and the Delaunay boundary tetrahedral mesh as a background mesh. In the present method, similar to the conventional advancing front method, the front is identified and internal points are created to form optimal tetrahedral elements in accordance with an element-size defined by the control space. A Delaunay kernel procedure is used to insert these points into the existing triangulation. Thus the convergence of the process is always ensured.

In our method, the advancing front method is used to generate the boundary triangular mesh. And the element size as well as element densities is only related to the given boundary triangular discretization. The corresponding scheme of the tetrahedral mesh generation algorithm can be briefly described as follows:

Scheme of the tetrahedral mesh generator

(1) Boundary tetrahedral mesh generation

1. Creation a box enclosing the domain;
2. Insertion of the boundary point into the initial mesh by Delaunay kernel;
3. Regeneration the boundary;

(2) Domain tetrahedral mesh generation

We define all the triangles of the surface mesh as the initial fronts. Identify the initial front to get the active front.

While (the set of active fronts is not empty)

Loop over facets of the set of active fronts to create new points

Calculate the location of the new point

Filtration of the new point

If the new point satisfies the criterion

Insert it using Delaunay kernel

Add the new triangle into the set of fronts if it belongs to two elements of new tetrahedrons

Loop over the new fronts to get active front

Identification of the new front

If the front satisfies the criterion

Add it into the set of active fronts

First step of the tetrahedral mesh generation algorithm is boundary tetrahedral mesh generation. At first, we construct a bounding box enclosing the domain. The box as a convex environment encloses all the boundary points. The box is defined by eight vertices and it is divided into five tetrahedrons. All the boundary points are inserted into the initial mesh using the Delaunay kernel. Then, the boundary is recovered, and we obtain the boundary tetrahedral mesh.

After the boundary tetrahedronization, the present method is used to generate the domain tetrahedral mesh. The tetrahedral mesh is obtained by adding field points inside the existing mesh and then optimized so as to complete the final mesh of the domain. Similar to the conventional advancing front method, we define the suitable triangle generated in the last step as the front. And all the triangles of the boundary triangular discretization are defined as the initial fronts. We define the front which can be used to create new points as the active front. The internal points are iteratively created by the active front to form optimal tetrahedral elements and inserted into the previous mesh by Delaunay kernel. The quality of the generated mesh can be improved dramatically by several techniques of mesh optimization. So, after the mesh generation the Laplacian smoothing and topological optimization process are used to improve the quality of the mesh.

Each of these steps is briefly described in the following sections. First, the creation of field point and the front identification are detailed step by step. Afterwards, the mesh control is described.

3. Creation of field point and identification of front

In the previous work, the field point is created with respect to a front face. The new vertex creation process is the same as advancing front method. The new vertex is created in a normal direction to the front at a distance of mesh size h. The front simply takes the form of a set of triangles between two elements. And these fronts are all used to create a new field point [8].

In our method, the field point is created with respect to a front, but the new point creation process is different from the conventional advancing front method. The front also simply takes the form of a set of faces between two elements. But the front needs to be identified in order to obtain the active front, because not all fronts are suitable

for creating a new point. We define the front which is suitable for creating a new point as active front. Therefore, not all fronts are used to create a new point, just the active fronts. And the process of the front identification is simple. So, our method is more efficient than the previous work.

3.1 Creation of field point

In this section, we describe the creation of an optimal point with respect to an active front, leading to the creation of an ideal tetrahedron. At first, we describe how to get an ideal tetrahedron. After the insertion of new points, many new tetrahedrons and triangles are generated. Then we need to get the set of fronts. After the front identification, we get a set of active fronts. Assume that $\triangle P_1P_2P_3$ is a triangle of the set of active fronts. In two dimensions, equilateral triangle is suitable for the numerical computation. In three dimensions, we always want to get regular tetrahedron. Assume that h is the requested size. So the length of the tetrahedron's edges is desired to equal to h, as shown in Figure 1.

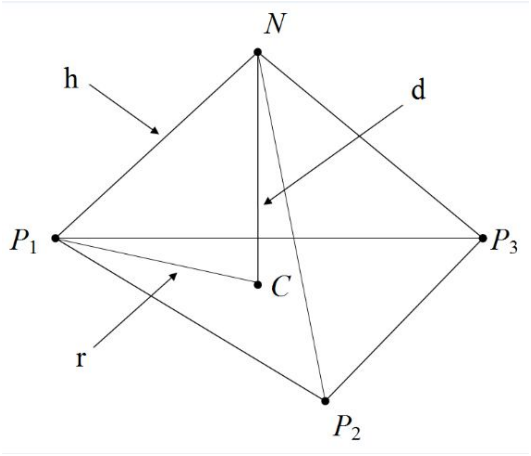

Figure 1. The ideal tetrahedron

h is calculated by control space which is described in section 4. We have to determine the position of the new ideal node N, so that the resulting element is as equilateral as possible while respecting the element size h. Let point C be the circumcircle center of $\triangle P_1P_2P_3$. We can prove that line NC is perpendicular to the plane $P_1P_2P_3$. Then, we just have to calculate the location of point C to get the coordinate of point N. Let us assume now that we know the coordinates of points $P_1(x_0, y_0, z_0)$, $P_2(x_1, y_1, z_1)$, $P_3(x_2, y_2, z_2)$. We can calculate the coordinate of point C by solve the following equations. $\boldsymbol{n}$ (n_1, n_2, n_3) is the normal vector of the plane $P_1P_2P_3$.

$$A=\begin{bmatrix} n_x & n_y & n_z \\ x_1-x_0 & y_1-y_0 & z_1-z_0 \\ x_2-x_0 & y_2-y_0 & z_2-z_0 \end{bmatrix} \tag{1}$$

$$X=\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{2}$$

$$B=\frac{1}{2}\begin{bmatrix} 2(n_x x_0+n_y y_0+n_z z_0) \\ x_1^{\,2}+y_1^{\,2}+z_1^{\,2}-x_0^{\,2}-y_0^{\,2}-z_0^{\,2} \\ x_2^{\,2}+y_2^{\,2}+z_2^{\,2}-x_0^{\,2}-y_0^{\,2}-z_0^{\,2} \end{bmatrix} \tag{3}$$

$$AX=B \tag{4}$$

Then we can calculate the circumcircle's radius r of the $\triangle P_1P_2P_3$.

$$r=\sqrt{(x_c-x_0)^2+(y_c-y_0)^2+(z_c-z_0)^2} \tag{5}$$

The length of the new point to the plane $P_1P_2P_3$ is $d=(h^2-r^2)^{0.5}$.

Then the location of the new point N can be calculated by

$$(x_n, y_n, z_n)=(x_c, y_c, z_c) \pm d\boldsymbol{n}. \tag{6}$$

The distributions of field points must satisfy the mesh size distributions. As every new point is created apart from the others, a filtration process can remove the point which violates the size criterion when compared with a previously selected point. And new points are generated in both two sides of the active fronts. But both two sides of the active fronts may not be suitable for creating a new tetrahedron. So, new points need to be filtrated.

1. Let $L(NP)$ be the length between new point N and arbitrary point P. N is filtered if the length $L(NP) < 0.8h$.

2. Let $T(P_1, P_2, P_3, P_4)$ be the tetrahedron which links to the current front. $\triangle P_1P_2P_3$ is the current front. Let L be the average length of three edges which link to P_4. N is filtered if $L < h$.

3.2 Identification of front

As mentioned in the previous section, the identification and update of the front is one of the most important steps of the present algorithm. In the conventional advancing front method, it is important to design an explicit data structure to management the front. In our algorithm, we only use the active front to create a field point. And the

fronts need to be identified to obtain the active fronts.

At the beginning of the algorithm, we define all the triangles of the boundary triangular mesh as the initial fronts. A set of new triangles is created after the insertion of points at stage i. In order to improve the efficiency of the algorithm, we need to identify these new triangles to get active front. Only the active front is suitable for generating a tetrahedron satisfying the mesh size distribution. At first, we create the front. Let F_i be a triangle of the set of new triangles created at stage i. If F_i belongs to two new tetrahedrons created at stage i, it is added to the set of fronts. At the second step, the front is identified in order to obtain the active front, because not all fronts are suitable for creating a new point. Let r be the Circumcircle radius of the front and h be the mesh size of the center point of the front. Mesh size h is extracted from the control space. The distance between new point and current front is calculated by $d=(h^2-r^2)^{0.5}$. So, we must insure $h > r$. The front satisfied this criterion can be used to create a new point. If $h > r$, the front is added into the set of active front. If $h < r$, the new tetrahedron generated by this front can not satisfy the mesh size distribution, and the front is eliminated. During the mesh generation, if the front can not create a new point, it is deleted.

After all the fronts are identified, ideal points are generated with every active front. We have to determine which side of the triangle the new point will locate at. We prescribe that the normal vector of the initial front point to the inner of the domain. In our algorithm, new points are generated in both two sides of the active fronts. And these new points need to be filtrated, as described in previous section.

4. Mesh control

In our algorithm, we use the background mesh to specify mesh size distributions over arbitrarily shaped domains. The background mesh covers the problem domain completely, and consists of triangles and tetrahedrons in two and three dimensions, respectively [11]. The mesh size attributes are assigned to the nodes of the background mesh. A linear variation of the mesh size is assumed within each element. Hence, the mesh size at an arbitrary point within the domain is calculated by linear interpolation of the nodal values of the element which the point is contained in.

The background mesh is defined by the boundary mesh of the domain. To obtain the mesh size of an arbitrary point within the domain, it is only needed to define the size at every vertex of background mesh. Let Q be a vertex of the background mesh, the

size of the vertex is calculated by

$$h(Q)=\frac{L_{\max}(Q)+L_{\min}(Q)}{2} \tag{7}$$

$L_{max}(Q)$ (resp. $L_{min}(Q)$) represents the length of the largest (resp. smallest) edge connected to Q.

For every point Q within the domain, there is a tetrahedron T enclosing it, as shown in Figure 2. Let $\alpha_i(1\leq i\leq 4)$ be the volume coordinates of Q in T, $h(Q)$can be defined by the way of an arithmetic interpolation

$$h(Q)=\sum_{i=1}^{4}\alpha_i h(P_i) \tag{8}$$

P_i, $(1\leq i\leq 4)$ are the vertices of T.

Before we calculate the mesh size of the point Q, we have to work out the location of the given point. It takes a lot of time to locate the point. In order to improve the efficiency we use the grid to manage the background mesh [13]. At the beginning, the background mesh is covered by structured grids with a large size. We create an array, and each element in the array corresponds to a cell of the grids. The element of the array stores the index of tetrahedrons which overlapped the cell. At first, we find the cell which contains the given point Q. Then, we can quickly find the tetrahedron which contains the point Q.

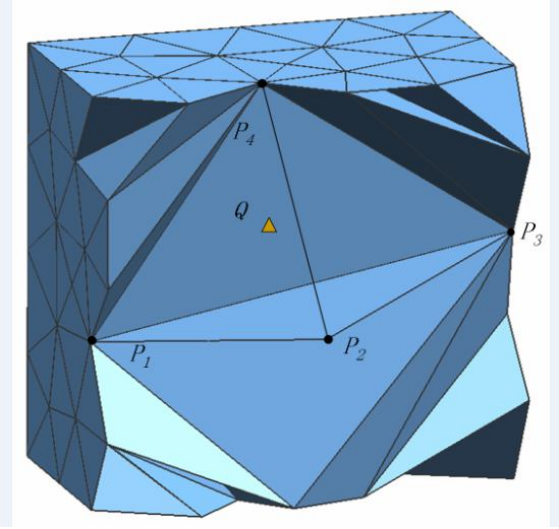

Figure 2. Background mesh

5. Examples of application

In this section we select three examples to demonstrate the robustness and efficiency of the proposed method. All the mesh generation process are carried out on the desktop computer with Intel (R) Core(TM)2 Duo CPU E7400 (2.80GHz). The first

example is a crankshaft. Figure 3 shows the generated tetrahedral mesh. The mesh consists of 2062 surface elements, 5465 volume elements, and 1522 mesh nodes. The total CPU time needed for the mesh generation is less than 1 seconds. Figure 4 shows the CPU times needed for the mesh generation of the first example with different element number. We can see that the CPU time increases linearly with the number of elements. More than 140000 elements can be generated by our method within 40 seconds.

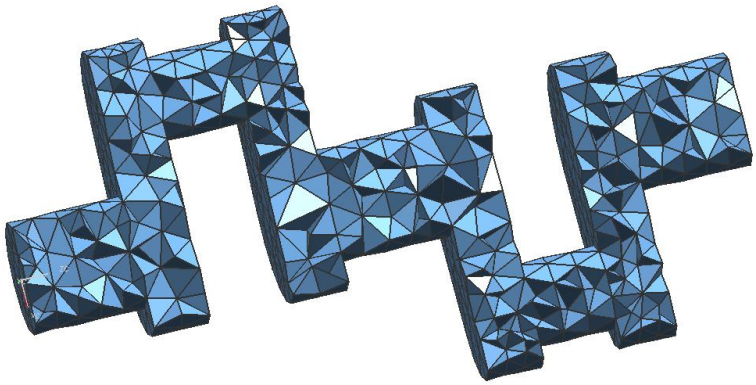

Figure 3. Example 1: tetrahedral mesh

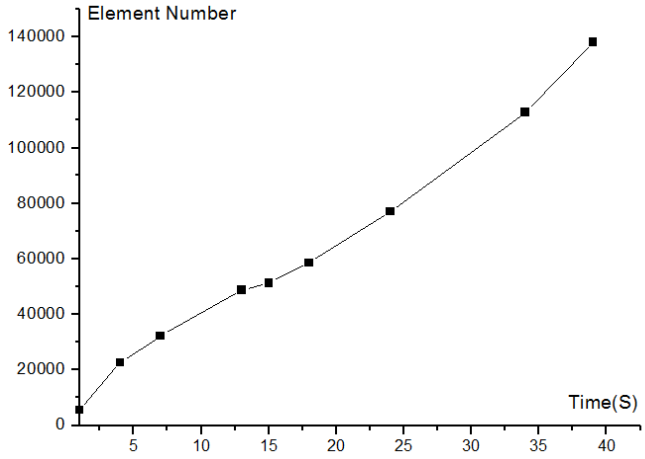

Figure 4. CPU times for example 1

The second example involves a complex geometric model drawn in Figure 5. This example is presented to demonstrate the ability of our method to generate meshes even in the case where the gradient of controlling element size is very steep. The mesh consists of 3254 surface elements, 13510 volume elements, and 3134 mesh nodes. The total CPU time needed for the mesh generation is 4 seconds. Figure 6 shows the generated tetrahedral mesh.

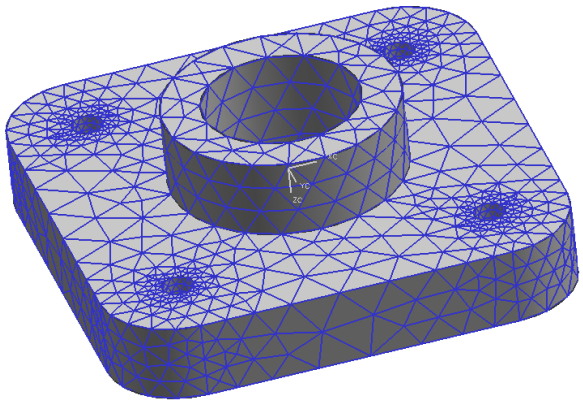

Figure 5. Example 2: surface mesh

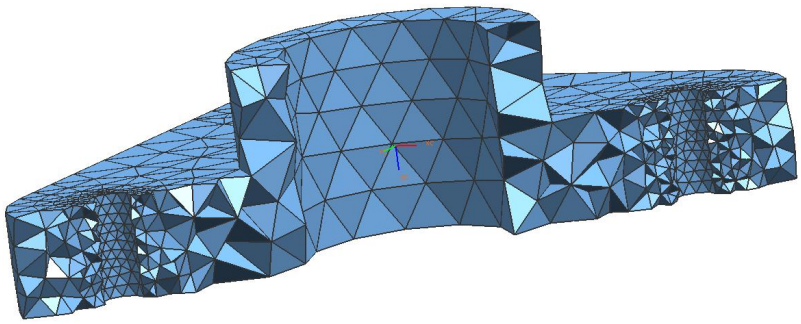

Figure 6. Example 2: tetrahedral mesh

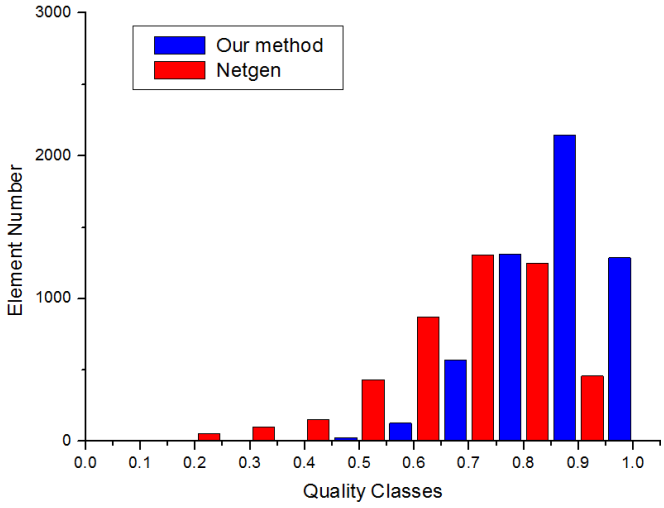

Figure 7. Example 1: quality classes

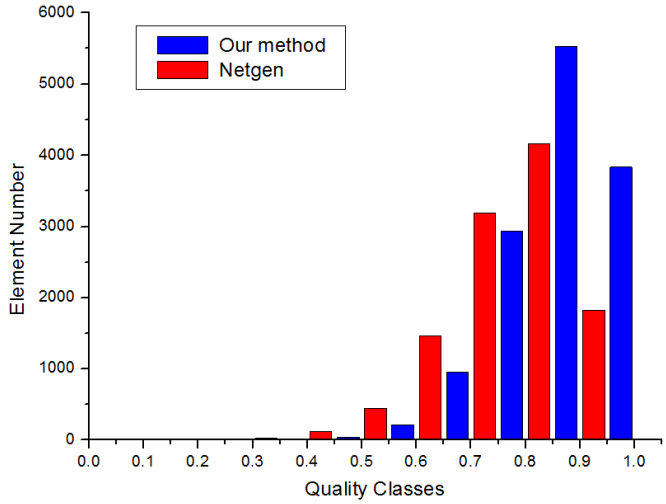

Figure 8. Example 2: quality classes

These two examples are compared to the mesh generated by Netgen. Netgen is an automatic tetrahedral mesh generator. The Netgen uses conventional Delaunay algorithm to generate tetrahedral mesh. Our method and Netgen use the same surface triangular mesh to generate tetrahedral mesh. The details of our surface triangular mesh generation algorithm can be found in references [10]. The quality of the tetrahedron is evaluated by element shape measures [14]. An element shape measure approaches a maximum value only for the equilateral tetrahedron, and approaches zero for degenerated tetrahedral of zero volume. In this paper, the element measure is the radius ratio ρ. The radius ratio of a tetrahedron T is defined as $\rho=3r/R$, where r and R are, respectively, the inradius and circumradius of T. Figure 7 and 8 show the quality classes of the volume elements generated by our method and Netgen. Table 1 shows the number of elements of each quality class.

Quality		Element Number										Total Element Number
		⩽0.1	⩽0.2	⩽0.3	⩽0.4	⩽0.5	⩽0.6	⩽0.7	⩽0.8	⩽0.9	⩽1.0	
1	Our Method	0	0	0	6	25	127	567	1309	2145	1286	5465
	Netgen	1	8	52	100	151	431	870	1302	1245	457	4617
2	Our Method	0	0	2	6	40	212	958	2930	5527	3835	13510
	Netgen	0	0	7	28	118	445	1467	3188	4162	1825	11240

Table 1. Distribution of quality classes

Table 1 shows that the mesh generated by our method doesn't contain poor quality element while the mesh generated by Netgen contains several poor quality element. And Figure 8 and 9 show that the mesh quality of the mesh generated by our method

is better than the mesh generated by Netgen. The number of elements generated by our method whose quality coefficient is between 0.7 and 1.0 accounts for 86.73% and 90.98% of the total element number while the number of elements generated by Netgen whose quality coefficient is between 0.7 and 1.0 accounts for 65.06% and 81.62% of the total element number. Compared to conventional Delaunay algorithm, our method can generate high quality mesh within a reasonable time limit. And our method can generate high quality tetrahedral mesh even in the case where the gradient of controlling element size is very steep.

The third example is a complex machine component model drawn in Figure 9. The mesh generated by our method consists of 11722 surface elements, 43979 volume elements, and 10435 mesh nodes. The total CPU time needed for the mesh generation is 9 seconds. Figure 10 shows the generated tetrahedral mesh.

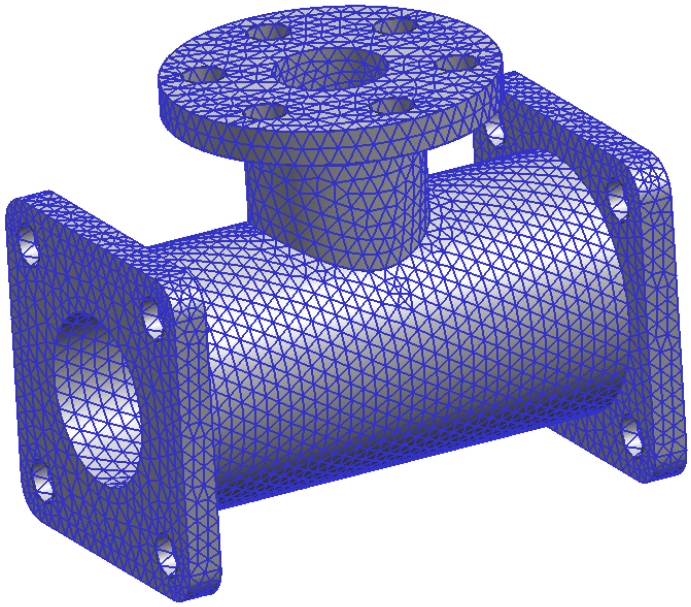

Figure 9. Example 3: surface mesh generated by our method

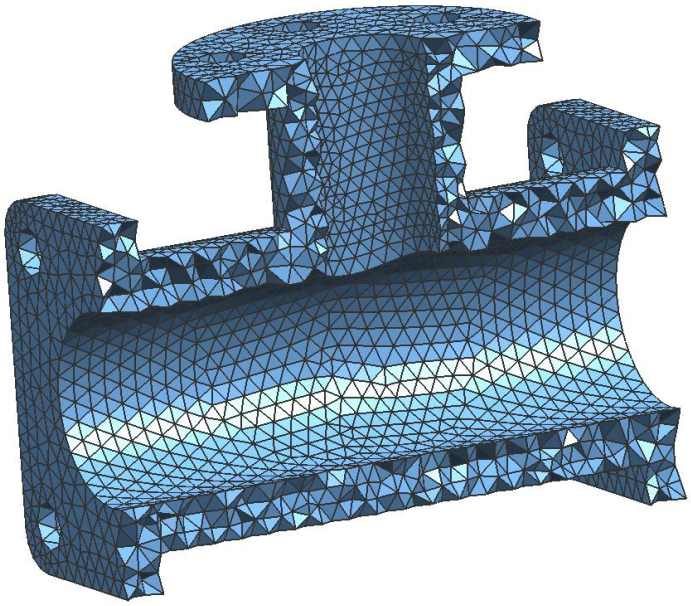

Figure 10. Example 3: tetrahedral mesh generated by our method

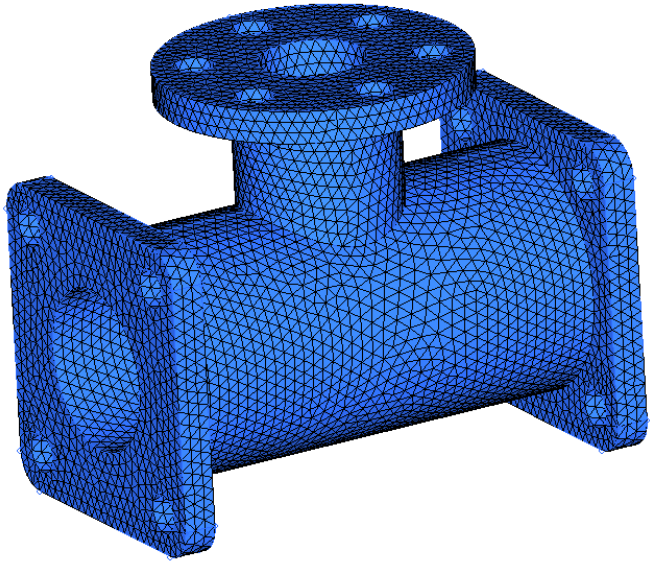

Figure 11. Example 3: mesh generated by Hypermesh

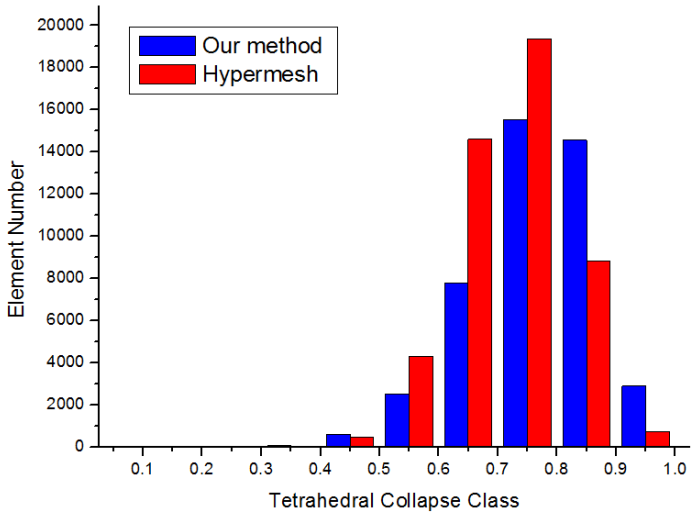

Figure 12. Example 3: tetrahedral collapse classes

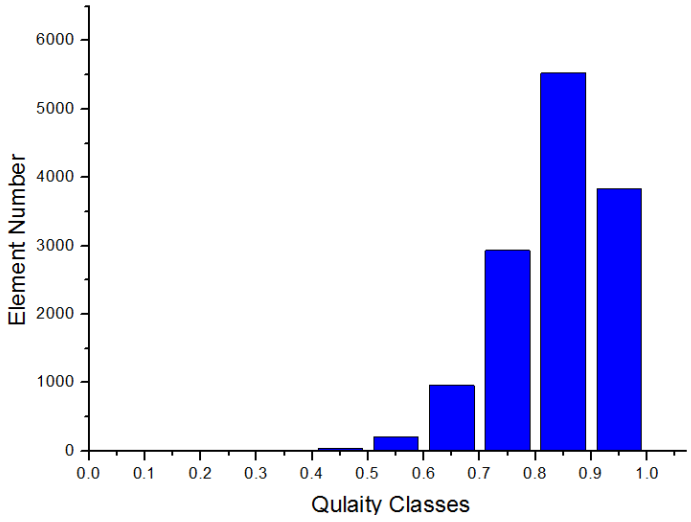

Figure 13. Example 3: quality classes

collapse	Element Number										Total Element Number
	≤0.1	≤0.2	≤0.3	≤0.4	≤0.5	≤0.6	≤0.7	≤0.8	≤0.9	≤1.0	
Our Method	0	0	2	63	619	2519	7782	15547	14563	2884	43979
Hypermesh	0	0	0	22	473	4324	14610	19362	8843	733	48367

Table 2. Distribution of tetrahedral collapse classes

This example is compared to the mesh generated by Hypermesh. Hypermesh is a powerful meshing software. The mesh generated by Hypermesh contains 17728 surface elements, 48367 volume elements. Figure 11 shows the mesh generated by Hypermesh. Hypermesh spends less time to generate tetrahedral mesh than our method. Our method may be less efficiency than the powerful commercial software which have been highly optimized and paralleled in code.

All of the jacobians of the elements generated by Hypermesh are greater than 0.7. The minimum tetrahedral collapse is 0.32. Hypermesh uses tetrahedral collapse as one of the mesh quality estimate criteria. Hypermesh calculates tetrahedral collapse by the following procedure. At each of the four nodes of the tetrahedron, the distance from the node to the opposite side of the element is divided by the square root of the area of the opposite side. The minimum value found is normalized by dividing it by 1.24. As the tetrahedral collapses, this value approaches 0.0. For a perfect tetrahedron, this value is 1.0. Figure 12 shows the tetrahedral collapse classes of the volume elements generated by Hypermesh and our method. Table 2 shows the number of elements of each tetrahedral collapse class. Figure 13 shows the quality classes of the volume elements generated by our method. The mesh generated by our method doesn't contain poor quality element. Elements generated by our method whose quality coefficient is between 0.7 and 1.0 account for 96% of the total element number. We can see that the mesh quality of the mesh generated by our method is as good as the mesh generated by Hypermesh.

6. Conclusion

In this paper, we proposed a Delaunay mesh generation method coupled with advancing front method for arbitrary three-dimensional domains. The proposed method combines the advantages of efficiency and nice mathematical properties of the Delaunay algorithm and the high quality point placement strategy of the advancing front method. A new front identification and field point generation method is proposed and applied to get better efficiency. Complex numerical examples have been shown to illustrate that the proposed method is numerically reliable, has favorable efficiency and has good mesh quality characteristics.

Acknowledgements

This work was supported in part by National Science Foundation of China under grant numbers 10972074 and 11172098, in part by national project under grant number 2011ZX04003-011, and in part by National 973 Project of China under grant number 2010CB328005.

References

[1] Lohner R, Parikh P. Generation of three-dimensional unstructured grids by the advancing-front method. International Journal for Numerical Methods in Fluids, 1988, 8(10):1135-1149.

[2] Jin H, Tanner R I. Generation of unstructured tetrahedral meshes by advancing front technique. International Journal for Numerical Methods in Engineering, 1993, 36(11):1805-1823.

[3] Rassineux A. Generation and optimization of tetrahedral meshes by advancing front technique. International Journal for Numerical Methods in Engineering, 1998, 41(4):651-674.

[4] Watson D. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. The Computer Journal, 1981, 24(2):167-172.

[5] Bowyer A. Computing Dirichlet tessellations. The Computer Journal, 1981, 24(2):162-166.

[6] S. Rebay. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. Journal of Computational Physics. 1993.106(1):125-138.

[7] J.D. Muller, P.L. Roe and H.Deconinck. A frontal approach for node generation in Delaunay triangulations. Int. J. Numer. Meth. Fl. 1993.17(3):241-255.

[8] Pascal J. Frey, Houman Borouchaki, Paul-Louis George. 3D delaunay mesh generation coupled with an advancing front approach. Comput. Meth. Appl. Mech. Engrg. 1998. 157 :115-131.

[9] Jianming Zhang, Xianyun Qin, Xu han and Guangyao Li. A boundary face method for potential problems in three dimensions. Int. J. Numer. Meth. Engrg. 2009.80(3):320-337.

[10] Xianyun Qin, Jianming Zhang, Guangyao Li, Xiaomin Sheng, Qiao Song and Donghui Mu. An element implementation of the boundary face method for 3D potential problems, Engineering Analysis with Boundary Elements, 2010, 34:934–943.

[11] Lo S H and Lee C K. Generation of gradation meshes by the background grid technique. Computers & Structures, 1994, 50:21-32.

[12] Jan Frykestiong. Advancing front mesh generation technique with application to the finite element method. Department of Structural Mechanics of Chalmers University of Technology, 1994, 10.

[13] Krause R and Rank E. A fast algorithm for point location in a finite element mesh. Computing. 1996. 57:49-62.

[14] Lo S H. Optimization of tetrahedral meshes based on element shape measures. Computers & Structures, 1997, 63(5):951-961.