ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 159

International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)

A Tetra-hex Hybrid Mesh Generation Method Based on Delaunay Triangulation

Pengfei Zhan[1,2], Xianhai Meng[1,2,*], Zhongxiang Duan[1,2] and Qin Yang[1,2]

[1]School of Computer Science and Engineering, Beihang University (BUAA), Beijing 100191, PR China

[2]State Key Laboratory of Software Development Environment, Beijing 100191, PR China

[*]Corresponding author

***Abstract*—This paper presents an automatic method to generate a hex-dominant hybrid mesh with the input being a Delaunay tetrahedral mesh. In this method, we replace the interior tetrahedra in the original mesh with regular hexahedra, and fill the cavity between the boundary tetrahedral and the interior structured elements with tetrahedral elements. The mesh generated by this method is a tetra-hex hybrid mesh which only consists of two types of grids. The result demonstrated that our method can perform on most tetrahedral mesh successfully. The total number of grids in the hybrid mesh is reduced markedly, and moreover, the resultant mesh can keep the conformity of the boundary and consist of grids with high quality.**

Keywords—hybrid mesh; advancing front technique; delaunay triangulation

I. INTRODUCTION

Owing to the development of computing hardware, many numerical simulation methods, such as finite element method (FEM), have been widely used in industry. However, as an essential preprocessing step, the mesh generation directly impacts on the accuracy and convergence of the simulation [1]. Currently, the generated mesh can be roughly classified into the structured and unstructured mesh, while the simulation on the structured grids require less computational memory and cost [2] and the unstructured mesh can adapt to the complicated boundaries of the domain with arbitrary shape [3][4]. Since the hybrid grids combine the advantages of both unstructured and structured mesh, the methods for the hybrid grid generation have received much attention in recent decades.

The basic elements in hybrid grids include tetrahedra, prisms, hexahedrons, and pyramids. The researchers usually use two or more of basic elements above to create a hybrid mesh. For instance, the hybrid grids consisting of prismatic and tetrahedral elements were first proposed by Nakahashi [5], and developed extensively by Kallinderis et al. [6]. In 2009, Yamakawa and Shimada [7] presented a new method of prism-tetra hybrid mesh generation that replaced tetrahedral elements filling a sweepable volume with prismatic elements. This method effectively reduced the number of elements, and the resultant hybrid mesh yielded a more accurate solution than a tetrahedral mesh. In addition, the hex-dominant hybrid mesh [8] is also a popular type of hybrid mesh. Steven J. Owen [9] presented a hex-dominant mesh generation method which guarantees the boundary conformity using 3D constrained triangulation. In the work of Jesse Chan et al.[10], they create the hex-dominant hybrid mesh of hexahedra, wedge, pyramid, and tetrahedral with the same size by taking every regular hexahedron as the reference hexahedron. In most cases, the purpose of most hybrid mesh generation methods is to reduce the computational cost and to keep the computational accuracy as well.

In this paper, we propose an automatic method to convert a Delaunay tetrahedral mesh to a tetra-hex hybrid mesh. In our method, the interior tetrahedra in the original mesh are replaced with regular hexahedra, and the tetrahedra near the boundary remain. The hybrid mesh generated by our method can keep the conformity of the boundary and consist of grids with high quality. Moreover, the total number of grids can be markedly reduced.

II. METHOD FOR HYBRID GRID GENERATION

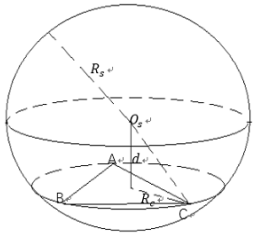
In this section, we introduce our algorithm to generate the tetra-hex hybrid mesh from the Delaunay tetrahedral mesh. The input of our algorithm is a Delaunay tetrahedral mesh with uniform size and high quality and we intend to fill the domain with structured elements inside and tetrahedra around the boundary. The main difficulty is to fill the cavity between boundary tetrahedral elements and inner structured elements. In our method, we design a modified advancing front Delaunay triangulation algorithm [11] to address this issue and we present the algorithm in the rest of this section.

A. Advancing front Initialization

Let TeS_1 denote the boundary tetrahedral set, where every tetrahedron in TeS_1 contains at least one vertex on the boundary. The initial front TS_{ori} is the inner triangle faces of these tetrahdra which are stored in a linked list. Denote one single front triangle face as T_m ($0 \le m \le n-1$) and n is the number of the front faces.

B. Hexahedron Insertion

In this step, the domain inside the initial front is filled with regular hexahedra $Cube_{in}$. However, the insertion process has an important principle that the interior hexahedral grids need keep a distance with the initial front to avoid too many silvers generated. We use the extra bounding box to control the



△ABC is the triangle on the front.
O_c is the center of facet circumcircle.
R_c is the radius of facet circumcircle.
O_s is the center of circumsphere.
R_s is the radius of facet circumsphere.
d is the distance between O_s and O_c.

FIGURE I. INITIAL SEARCH REGION.

between the initial front and the inner grids. The extra bounding box expands the bounding box of the triangular faces on the initial front, so that the inner cube which is too close to the front will be removed.

Let FS_{hex} be the set of outside surfaces of inner hexahedra and let l_{hex} be the length of inner hexahedron. The minimal edge length of the initial front, which is denoted as l_{min} is related to l_{hex}. In order to generate tetrahedra with the ratio of the radius to the shortest edge smaller than 3.6, we make l_{hex} smaller than $1.35l_{min}$. In our experiment, the value of l_{hex} is between $1.00 \sim 1.30l_{min}$.

C. Tetrahedron insertion

After the pretreatment above, we get an initial front TS_{ori} and a finishing surface set FS_{hex}, and then the empty region between TS_{ori} and FS_{hex} need be filled with new tetrahedra. Firstly, a list of candidate nodes PS_c is formed from all the nodes lying on TS_{ori} and FS_{hex}. For the first triangle Tri_0 on the initial front, choose a best point P_0 from PS_c by a Delaunay rule, which is known as the empty circumsphere criterion. We create the new tetrahedron Te_N in the empty region.

The algorithm of choosing a point satisfying the empty circumsphere criterion is shown as follows:

Algorithm Choosing Process(PS_c, Tri_0, P_0)

Input the set of candidate points PS_c;
the first triangle Tri_0 on the front;
Output the point P_0 satisfying the empty circumsphere criterion;
1. **If** PS_c is empty, **then return** NULL.
2. Create an initial search region D_1 for Tri_0 (Fig 1), and then points inside D_1 forms a sub-candidate pointset PS_{sub}.
3. **If** PS_{sub} is empty, **then return** NULL.
Otherwise, PS_{sub} contains at least one point.
(a)**If** PS_{sub} contains only one point P_x, **then** mark P_x to be P_0, **return** P_0.
(b)**Otherwise**, choose the point P_0 satisfying the empty circumsphere criterion from PS_{sub}.
4. **return** P_0.

In this algorithm above, we create a provisional circumsphere for Tri_0 for the purpose of removing the candidate points which is on the opposite direction or too far from Tri_0, so the distance d in Fig 1 satisfies $d = R_c$, and the radius of this sphere is $R_s = \sqrt{2}\,R_c$. Moreover, if the output of this algorithm is NULL, we can insert a new point P_1 in a reasonable position, and use P_1 and Tri_0 to create the new tetrahedron.

After a new tetrahedron Te_N has been created, Te_N will be searched for any collisions with the TS_{ori} or FS_{hex}. If Te_N is intersected with any facet, this new tetrahedron will be discarded, and we will choose another point P_0` for Tri_0. If Te_N passes the collision detection, the initial front TS_{ori} will be updated by the new three triangles T_{new} on Te_N. The result of front update is divided into three conditions as follows:

(a) If TS_{ori} contains T_{new}, then remove T_{new} from TS_{ori}.

(b) If T_{new} overlaps with FS_{hex}, then do nothing.

(c) If T_{new} overlaps with no facet of TS_{ori} and FS_{hex}, then add T_{new} into TS_{ori} .

Choose the first triangle Tri_0 from TS_{ori}, and repeat the processes above, until the TS_{ori} is empty. Finally, we yield a set of new tetrahedra TeS_{in} between the front and the finishing surface. The whole algorithm of tetrahedron insertion is shown as follows:

Algorithm TetraInsertion(TS_{ori} , FS_{hex} , TeS_{in})
Input the list of the initial front TS_{ori} ;
the set of the finishing surface FS_{hex} ;
Output the set of new tetrahedra TeS_{in} ;
1. Create PS_c by TS_{ori} and FS_{hex}.
2. **While** $TS_{ori} \neq \emptyset$
(a) Choose first triangle $TS_{ori} \in TS_{ori}$, and call **ChoosingProcess**(PS_c, Tri_0, P_0).
(b) **If** P_0 = NULL, **then** add a new point P_1 to PS_c, and mark P_1 to be P_0.
(c) Create Te_N by P_0 and Tri_0.
(d) **If** Te_N fails the intersection test, **then** discard Te_N, and choose another point P_0`, use $P_0$` and Tri_0 to create new tetrahedron Te_N`.
(e) Add Te_N or Te_N` to TeS_{in}.
(f) Update TS_{ori} .
End while.
3. **return** TeS_{in} .

III. ALGORITHM ANALYSIS

A. Termination

This method is an automatic hybrid mesh generation method which can terminate in a finite time. In our method, two issues may bother the process. The first problem is that the first-circumscribed sphere of Tri_0 contains no candidate point, and another one is that T_{new} fails the intersection test. We can solve the first problem by adding a suitable point P_1 into the cavity and choose P_1 as the fittest candidate point.

The second one can also be solved by the step of choosing fittest candidate point. In practice, this step is simple as it puts points on the triangles, which are adjacent to Tri_0, into sub-candidate pointset PS_{sub}, and choose the fittest point P_0 from PS_{sub}. There must be at least one point that is in the forward direction of Tri_0, otherwise no triangular facet can intersect with the previous tetrahedron Te_N. On another hand, unless

there is a point P_i inside the new tetrahedron Te_N`, the new tetrahedron $Te_N$` can surely pass the intersection test this time. Even if P_i exists, we can choose P_i as the fittest point at first time. As a result, it is certain to get a suitable point P_0 that can create a new tetrahedron which can pass the intersection test. Finally, all the triangular faces can connect to FS_{hex}, and the algorithm terminates.

B. *Algorithm complexity*

There are two main parts in this method, the pretreatment and the process of tetrahedron insertion. In the pretreatment, we mainly traverse the set of all the input tetrahedra or all the hexahedra inserted, and use every hexahedra to do intersection test with the initial front, so the algorithm complexity is $O(Num_{tetra})$ and $O(Num_{cube} \times Num_{front})$.

However, the main process is an iterative process, and the algorithm complexity depends on the total number of input tetrahedra. At each iteration, we need create a candidate subset, using $O(Num_{pset})$, and then choose the fittest point, using $O(Num_{subpset})$ or $O(Num_{subpset}^2)$ in the worst case. While a new tetrahedron is created, we will use it to do intersection test with the front and the surface of inner hexahedra. The algorithm complexity of each intersection test is $O(Num_{front} + Num_{fhex})$. With the increasing number of the total elements in the mesh, it will take more iterations to complete the hybrid mesh generation. For example, when the number of elements in the resultant mesh is 16905, it needs 9246 iterations. However, when the number of elements in the resultant mesh is 31584, it needs 14104 iterations to terminate.

IV. EXPERIMENTAL RESULTS

We present several mesh examples of our hybrid grid generation algorithm in Figure 2 and the statistic on the reduced quantity of the grid elements by our method is shown in Table 1 In the examples, we choose the hexahedra edge length l_{hex} to be 1.20 l_{min} .

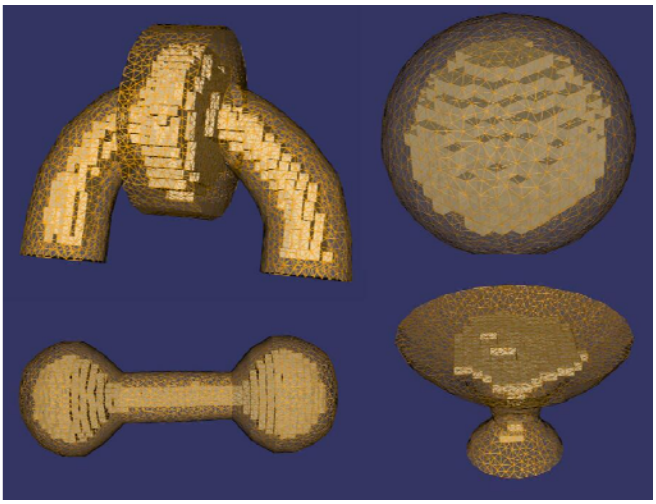


FIGURE II. EXAMPLES OF HYBRID MESH, ARCH (UPPER LEFT), SPHERE (UPPER RIGHT), DUMBBELL (LOWERLEFT), GRAIL (LOW RIGHT).

TABLE I. COMPARISON OF ELEMENTS QUANTITY BETWEEN TETRAHEDRA AND HYBRID MESH.

Number of grids	**Tetrahedral mesh**	**Resultant hybrid mesh**			
		Tetra	***Hex***	***total***	***H_r/T(%)***
Arch	**57662**	29948	1636	**31584**	**45%**
Sphere	**65947**	25053	3581	**28634**	**56%**
Dumbbell	**50176**	25144	1583	**26727**	**46%**
Grail	**30888**	18557	432	**18989**	**38%**

a. H_r is the number of the grids reduced, and T is the number of tetrahedral mesh

FIGURE III. THE GRID OF ARCH WITH DIFFERENT HEXAHEDRA SIZE. FROM LEFT TO RIGHT: LHEX EQUALS TO 1.00/1.15/1.30 LMIN.

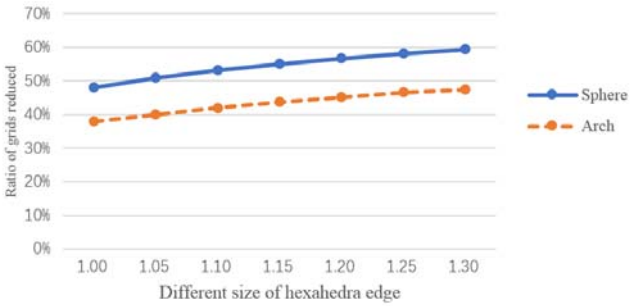


FIGURE IV. THE RATIO OF GRIDS REDUCED ON DIFFERENT SIZE OF HEXHEDRA EDGE. (LHEX EQUELS TO X * LMIN , AND X IS THE VALUE OF X-AXIS.)

It is shown that our method can truly generate the qualified grids and decrease the number of the grid elements, especially the example of Sphere, where the grids can be reduced by more than a half. Although the hexahedra are much less than tetrahedra in each example, they play an important role in reducing total elements number as they not only can replace the tetrahedra, but also can reduce the number of points inside the mesh.

The proportions of the grid elements reduced when the hexahedra size is different are shown in Fig. 3 and Fig. 4. As shown from the line chart, as the size of hexahedra extends, the more grids are reduced. The reason for this phenomenon is that the extension of the hexahedral size decreases the number of the interior hexahedra and the candidate points. Besides, as we can see, our method always reduced more elements for the example Sphere compared to the Arch. This indicates that when the models contain larger cavities inside, the algorithm performs better on the reducing of the mesh quantity.

V. CONCLUSION

In this paper we have presented an automatic hybrid generation mesh method with the input being a Delaunay tetrahedral mesh. This method replaces the interior tetrahedra with regular hexahedra and in the transition region, and it

connects two triangular faces to a quadrilateral face. Almost all tetrahedral mesh can be converted to a tetra-hex hybrid mesh by this method. The elements number of resultant mesh is reduced significantly and this method can perform much better on the model with a large cavity.

ACKNOWLEDGMENT

National Natural Science Foundation of China (Grant 61003110) and Fund of the State Key Laboratory of Software Development Environment (Grant SKLSDE-2010ZX-10) jointly supported this work.

REFERENCES

[1] K. Ho-Le, "Finite element mesh generation methods: a review and classification," *Comput.-Aided Des.*, vol. 20, no. 1, pp. 27–38, 1988.

[2] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and G. Sjaardama, "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis," presented at the Proceedings, 4th International Meshing Roundtable, 1995, vol. 17, pp. 179–191.

[3] J. R. Shewchuk, "Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery.," presented at the IMR, 2002, pp. 193–204.

[4] S. Lo, "3D Delaunay triangulation of non-uniform point distributions," *Finite Elem. Anal. Des.*, vol. 90, pp. 113–130, 2014.

[5] K. NAKAHASHI, "Viscous flow computations using a composite grid," presented at the 8th Computational Fluid Dynamics Conference, 1987, p. 1128.

[6] Y. Kallinderis and S. Ward, "Prismatic grid generation with an efficient algebraic method for aircraft configurations," presented at the 10th Applied Aerodynamics Conference, 1992, p. 2721.

[7] S. Yamakawa and K. Shimada, "Converting a tetrahedral mesh to a prism–tetrahedral hybrid mesh for FEM accuracy and efficiency," *Int. J. Numer. Methods Eng.*, vol. 80, no. 1, pp. 74–102, 2009.

[8] R. J. Meyers, T. J. Tautges, and P. M. Tuchinsky, "The" Hex-Tet" Hex-Dominant Meshing Algorithm as Implemented in CUBIT.," presented at the IMR, 1998, pp. 151–158.

[9] S. J. Owen, "Hex-dominant mesh generation using 3D constrained triangulation," *Comput.-Aided Des.*, vol. 33, no. 3, pp. 211–220, 2001.

[10] J. Chan, Z. Wang, A. Modave, J.-F. Remacle, and T. Warburton, "GPU-accelerated discontinuous Galerkin methods on hybrid meshes," *J. Comput. Phys.*, vol. 318, pp. 142–168, 2016.

[11] R. Radovitzky and M. Ortiz, "Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-Delaunay triangulation," *Comput. Methods Appl. Mech. Eng.*, vol. 187, no. 3–4, pp. 543–569, 2000.