Some Improvements on the FDTD mesh generation algorithm

Hui Zhu, Cheng Gao, Hai-lin Chen
(National Key Laboratory on Electromagnetic Environment Effects and Electro-optical Engineering, PLA University of Science and Technology, Nanjing, Jiangsu 210007, China)
zhuhuiblog@gmail.com

Abstract: Mesh generation is the prerequiste of Finite Difference Time Domain (FDTD) method. In this paper, singular points in mesh is classified and a new strategy is proposed to deal with these points directly. In addition, a parallell algoritm is implemented with OpenMP to accelerate the mesh process. In the end, a test example is provided to verify the accuracy and efficiency of the proposed improved algorithm.

Key words: FDTD, Grid Mesh, Parity of Intersections Number Method, singular points, OpenMP.

I. INTRODUCTION

Mesh generation is the prerequiste of the Finite Difference Time Domain (FDTD) method [1], which is a powerful and robust numerical method for many applications such as lightning electromagnetic pulse (LEMP), nuclear electromagnetic pulse (NEMP), and high power microwave (HPM) simulation. However, there are no mature commercial software for this purpose. Some algorithms have been proposed to mesh complex models [2-7].

The parity of intersection number method is used in this work to mesh models. This method is performed in a point-by-point way and judge the grid point whether in the model by the parity of the number of intersections. More details can be found in our former work in [6]. It should be noted that this method requires that the model used must be convex bodies and consits of triangular facets.

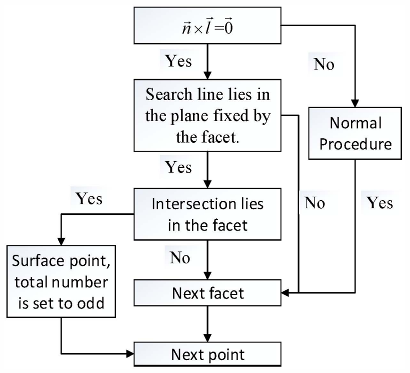
II. STRATEGY FOR SINGULAR POINTS

Some special situations will occor when the parity of intersection number method is employed to mesh models. Firstly, a search line may parallel to one of the facets or in the plane fixed by the facet, thus no or infinite intersection points can be found and the method does not work. Secondly, the intersection points of seach lines and some facets may lies in the characteristic elements (edges or vertexes) of a triangular facet. Beacause of the clourse property of models, all of the characteristic elements will be shared by two or more triangular facets, which will lead to repeated count of intersection points, thus the wrong total number is acquired. The intersection points in the above two situations are called sigular points, strategies for them are defferent and will be discussed as the follow two parts.

To deal with singular points, two methods were proposed. The first method is to change the search line or the model itself with a small translation length to avoid these singular points, however, this needs to compute the intersections points of every facet with the search line once again, which not only increases the computational burden, but also can not ensure singular points disappear for other facets. The second method proposed in [4] judges singular points by neighbouring grid points. However, this method assumes that all of the nerghbouring grid points are not singular, which is apprently can not be used in the case where some of these points are also singular. To solve this problem, a new scheme is proposed in this paper to directly deal with these singular points, which is proved to be efficient and robust in the example provided in the example bellow.

A. Sigular points of the first category

Sigular points of the first category can be judged by the vector product $\vec{n}\times\vec{l}$, with $\vec{n}$ the normal vector of a facet and $\vec{l}$ the directional vector of a search line. When the product is equall to zero, the point is determined as a first category sigular point. The procedure to handle these points is summarized as three steps, as is illustrated in the fig. 1.



$\vec{n}\times\vec{l}=\vec{0}$
Yes
No
Search line lies in the plane fixed by the facet.
Normal Procedure
Yes
Yes
Intersection lies in the facet
No
Yes
Surface point, total number is set to odd
No
Next facet
Next point

Fig. 1 procedure to handle sigular points of the first category

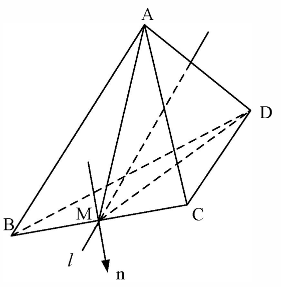
B. Sigular points of the second category

The solution to points of the second category are more complex. Two different strategies are adopted according to the type of facet elements penetrated by the search line.

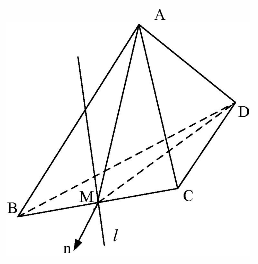
a) search line intersecting with the facet at edges

In this case, the search line intesrsects with the edge shared by two neighbouring facets. The search line may penetrate the model or just pass by with only one intersection point. For the latter situation, the intersection point should be counted only once, while for the former, both of the two intersections should beneglected, thus the total number remainsunchanged.

In order to judge whether a search line penerates into the model, the only thing we need is to determine whether the two facets lies at the same side of the plane fixed by the edge and search line. That is to say, if they lie at the same side, we can determin that the search line penetrate into the model, and vice verse. The way to peform this judgement is summarized in fig. 3.

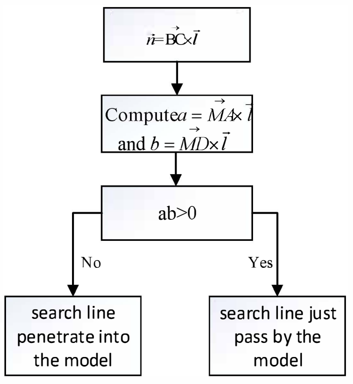


(a) search line intersect with the facet at edges



(b) search line intersect with the facet at vertexes.

Fig. 2 Sigular points of the second category



$\vec{n}=\vec{BC}\times\vec{l}$
Compute $a=\vec{MA}\times\vec{l}$ and $b=\vec{MD}\times\vec{l}$
ab>0
No
Yes
search line penetrate into the model
search line just pass by the model

Fig. 3 procedure for search line intersecting with the facet at edges

b) search line intersect with the facet at vertexes

It is the same as the above case, the search line may penetrate or pass by the model. For generisity, the situation in fig. 4 is considered. The seach line l pass by the vertese M shared by several facets. Note that the polygon A-B-C-D-E-F may not lie in a same plane. Assume edge M-D, M-E, M-F, and line l intersect with the plane fixed by A, B, and C at *D*’, *E*’, *F*’, *M*’. Then the original 3D problem can be transformed to a 2D one. If *M*’ lies in the new polygon A-B-C-D’-E’-F’, *l* penetrates into the model and vice verse. In addition, when *M*’lies in the edge or conincide with one vertex of the new polygon, this can be catogorized into rhe first kind of singular points and handled as the methid described before.

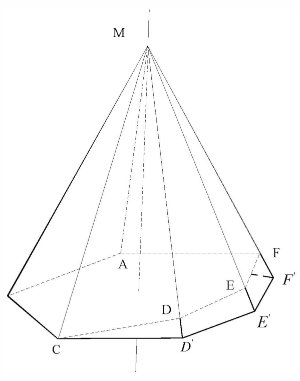


Fig. 4 search line intersect with the facet at vertexes

III. Parallelization of Mesh Algorithm

To accelerate mesh speed, a parallel version of the mesh algorithm need to be implemented. In this paper, OpenMP is used because of its simplicity.

OpenMP is a set of memory-share compile-guided directives, which supports many languages such as C++ and Fortran and is portable on machines with defferent operating systems, including WindowsNT and most Unix-like systems. A fork-join excuting model is adopted by OpenMP, by which a program starts with a main thread and forks new threads to peform parallel tasks when necessary. When tasks are finished, all derived threads will exit or hang on, and the main thread excute serially again. The pseudo-code of the parallel algorithm is listed in fig. 6.

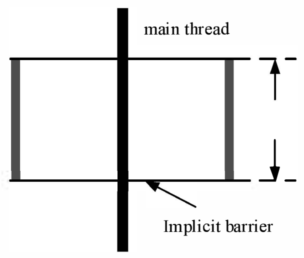


Fig. 5 The fork-join model of OpenMP

```
// parallel region begin， m threads
#pragma omp parallel for num_threads(m) private(i, j, k, n,
v, f)
for(i = 0; i <= imax; i++) {
   for(j = 0; j <= jmax; j++) {
      for(k = 0; k <= kmax; k++) {
         // determine the cell to be judged
         v = cell(i, j, k) ;
         for( n = 0; n <= nmax; n++) {
            // determine facet
            f = find_face(n) ;
            // judge the relation between search line and facet
         judge_position(v, f) ;
         }
         // handle singular points
         if( v.is_singular ) {
            handle_singular() ;
         }
         // judge the cell whether in the model
         judge_cell(v) ;
         // save cell
         #pragma omp critical
         save_cell(v) ;
      }
   }
}
```

Fig. 6 pseudo-code for parallel mesh algorithm

IV. Mesh example

In this section, a test model illustrated in fig. 7 is used to verify the proposed mesh algorithm. The test is implemented on a machine with a 4-core i7 CPU. With the visualization system described in [6], mesh result is rendered and plotted as fig. 8.

The curve in fig. 9 depicts the relation between speed-up ratio and the number of threads. The curve comprises three nearly linear segments. When the number of threads n_t is less than the number of CPU cores n_c, mesh time decreases leanearly as n_t decreases; After that, if n_t continuously increases, mesh time stay still until n_t exceeds 10. When n_t increase from 10, mesh time increases dramatically, this is because the threads scheduling time increases.

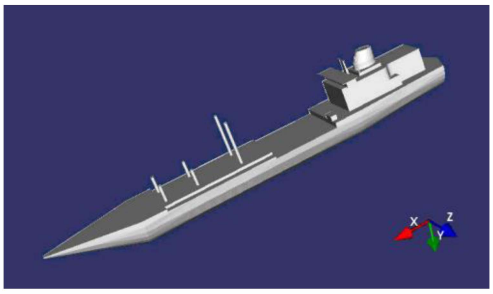
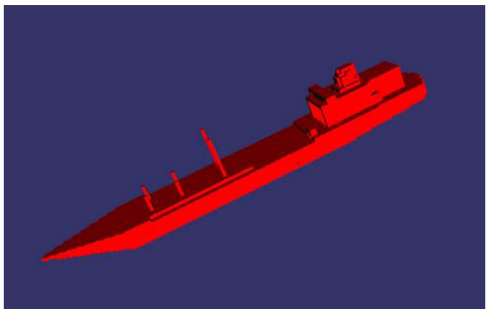
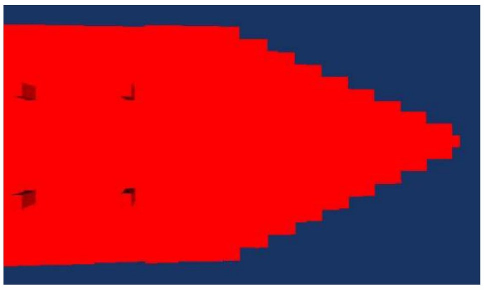


Fig. 7 Original Model



a) Overall mesh result



b) Detail of the mesh result

Fig. 8 Mesh result

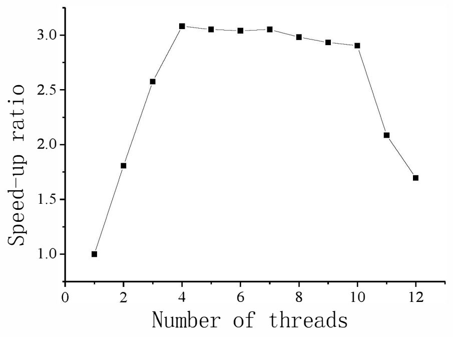


Fig. 9 the relation between speed-up ratio and number of threads

V. CONCLUSION

A new stratagy for singular points is proposed in this paper, which can deal with singular points directly and is robust and efficient. In addition, a parallell mesh algorithm is implemnted to save mesh time. In the end, a test is provided, which verifies the accuracy and speed of the proposed improved algorithm.

REFERENCES

[1] A. Taflove and S. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 2nd ed. Boston, MA: Artech House, 2000.

[2] Jeff T. MacGillivray. Trillion Cell CAD-Based Cartesian Mesh Generator for the Finite-Difference Time-Domain Method on a Single-Processor 4-GB Workstation[J]. IEEE Transactions on Antennas and Propagation, 2008, 56(8): 2187-2190.

[3] Y. Srisukh, Nehrbass J, Teixeira F L, et al. An approach for automatic grid generation in three-dimensional FDTD simulation of complex geometries[J]. IEEE Antenna and propagation magazine, 2002, 44(4): 75-80.

[4] Jonathan Hill. Efficient Implementation of Mesh Generation and FDTD Simulation of Electromagnetic Fields[D], Worcester: Worcester Polytechnic Institute, 1996.

[5] M W Yang, Y C Chen. AutoMesh: An automatically adjustable, non-uniform, orthogonal FDTD mesh generator[J]. IEEE Antennas and Propagation Magazine, 1999, 41(2): 13-19.

[6] Zhu Hui, Gao Cheng, Shi Zhen-hua, et al The Research on FDTD Mesh Generation and Visualization Technology[D]. The Sixth Asia-Pacific Conference on Environmental Electromagnetics Proceedings, 2012, 282-184.

[7] Liang Hui, Song Zu-xun. FDTD Mesh-Generating and Visual Realization Based on Triangle-Patch [J]. Computer Simulation, 2009, 26(11): 106-109(in Chinese).

[8] R. Chandra, R. Menon, L. Dagum, et al. Parallel Programming in OpenMP[M]. Morgan Kaufmann, 2000.