Acceleration of Dynamic Bubble Mesh Generation for Large-Scale Model

So Noguchi, Fumiaki Nobuyama, and Hajime Igarashi

Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan

An automatic mesh generation method employing a dynamic bubble system can provide a high-quality mesh for electromagnetic finite element analysis. However, it takes very long time to compute bubbles' movement when a mesh with a huge number of elements is generated. The bubbles move according to a force among them like the van der Waals force. However, the force received from bubbles of very far distance can be ignored. Therefore, it is unnecessary to calculate the force from all the bubbles. In this paper, we propose the subdivided dynamic bubble system in order to shorten the computation time. The proposed method considers an analysis region to divide into some tetrahedral subdivisions, and then the bubbles moves independently in the tetrahedral subdivisions. As the result, the acceleration of the computation time is achieved.

Index Terms— **Dynamic bubble system, finite element analysis, mesh generation, parallel computing.**

I. Introduction

Recently, very large-scale Finite Element Analysis (FEA) with a huge number of elements is often performed for design and performance survey of electromagnetic machines. Mesh generation is necessarily required as a preprocessing of FEA. The quality and size of mesh affect the simulation accuracy and the computation time [1], [2]. That is, a mesh of good quality yields high analysis accuracy and short computation time. However, the mesh generation of large-scale model is usually very laborious and time consuming. In order to solve this issue, a 3-D automatic mesh generation method adopting a dynamic bubble system has been proposed [3]-[7]. It composes a dynamic bubble system and a Delaunay division. The dynamic bubble system generates a set of vertices inside an entire analysis domain, and then the Delaunay division completes the connection of the vertices to generate a tetrahedral mesh.

The dynamic bubble system proposed in [3] has a high ability to generate vertices with adequately dense and coarse distribution. However, the computation of the bubbles' movement is time consuming, and it is difficult to apply it to a very large-scale model. It requires a high-performance CPU with a large memory to simulate the bubbles' movement. The bubbles move according to the forces acting on them. However, the force of one bubble is nearly zero from other bubbles adequately far from it. Accordingly, it is unnecessary to simultaneously compute the force from all the bubbles. To solve this problem, the proposed bubble system generates initial coarse tetrahedral subdivisions, and it is generated from the nodes of the initial input data using the Delaunay division. The bubbles' movement is independently performed in every individual initial tetrahedral subdivision. By subdividing the analysis region, it is possible to reduce the used memory and the computation time. In recent years, an ordinary PC has a very large memory, however the use of a GPGPU is proposed for acceleration of a mesh generation [8]. As of now the memory of the GPU is not so large.

Manuscript received June 29, 2013. Corresponding author: S. Noguchi (e-mail: noguchi@ssi.ist.hokudai.ac.jp).

Digital Object Identifier inserted by IEEE

In this paper, we propose a subdivided dynamic bubble system for a large-scale model, and the bubbles move independently in every initial coarse tetrahedral subdivision. Moreover, the proposed method is parallelized with GPGPU in order to accelerate an automatic tetrahedral mesh generation. Here, the large-scale model means that a generated mesh has a huge number of elements, more than one million.

II. Dynamic Bubble System

A. Dynamic Bubble System

The dynamic bubble system [3] is a physical model using multiple bubbles. It generates and moves the bubbles in an analysis domain, and computes a dense disposition of them as possible. The disposition of nodes obtained by the dynamic bubble system has a smooth variation of sparse and dense. Therefore, it is promised that a high-quality mesh is generated [9], [10].

The bubble is usually represented as a spherical particle with a node at the center point, a radius, mass, and volume. The center location and radius of bubbles determine the bubbles' motion with their interaction in the analysis domain, wherein the bubbles ultimately reach their stabilized disposition. The radius of each bubble is proportional to the desired mesh size, and every bubble acts on a force from the other bubbles. Fig. 1 shows the procedure of the ordinary dynamic bubble system [3].

The ordinary dynamic bubble system has an issue that it takes long computation time and a large memory is consumed for a large-scale problem. Let the number of bubbles be N, and $N \times (N–1)$ computations is necessary in every time step. Therefore, the computation time explosively increases with increase of bubbles. In addition, since one bubble receives the force from all the other bubbles, a common large memory uses for storing the position and size of the bubbles. However, recently, large-scale FEAs with more than one million elements have been occasionally done; therefore the dynamic bubble system has to be improved to deal with more than one million elements at least with short time as possible.

To address the issue of the time consumption, the

acceleration with a GPGPU was proposed [8]. Of course, it was achieved to shorten the computation time however the memory of GPU is too small to employ a large-scale model. When more than one million nodes are generated, the computation time becomes remarkably long. The acceleration with small GPU memory is a new subject.

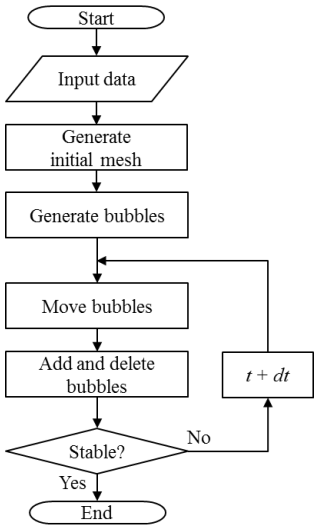


Fig. 1. Flowchart of the ordinary dynamic bubble system.

B. Algorithm of Bubble Motion

The following equation of the bubble motion is considered in the dynamic bubble system.

$$m_i \frac{\mathrm{d}^2 p_i}{\mathrm{d}t^2} + c_i \frac{\mathrm{d}p_i}{\mathrm{d}t} = f_i \quad (i = 1,2,\ldots,N) \tag{1}$$

where m_i is the mass of bubble i, c_i is the viscosity coefficient, p_i is the position of the bubble center, N is the total number of bubbles, and f_i is the resultant force acting on bubble i, respectively. The force f_i, which depends on the center position and the distances from its center to the center of the neighboring bubbles, is mathematically modeled by the van der Waals force [4]. Let d denote the distance between the centers of two adjacent bubbles, and the van der Waals force acting on the bubbles is approximated by the 3rd order polynomials, as shown in Fig. 2. The approximation function is set to the following equations:

$$f(d) = \begin{cases} Ad^3 + Bd^2 + Cd + D, & 0 \le d \le d_1 \\ 0, & d < 0,\ d_1 < d \end{cases} \tag{2a}$$

$$f(d_0) = 0 \quad f(d_1) = 0 \quad f'(0) = 0 \quad f'(d_0) = -k_0 \tag{2b}$$

where $f(d)$ indicates the magnitude of the force between two neighboring bubbles with the distance d and k_0 is the elastic coefficient at the stabilized distance d_0. Note that the force at $d = 0$ is not infinite in this model, and the forces are activated within a restricted distance ($d < d_1 = 1.5\ d_0$). The finite force at $d = 0$ prevents a singular situation when the center of bubbles coincide with each other. To reduce the computation cost, only the bubbles in the restricted distance ($d < d_1$) are considered in the dynamic bubble system. k_0 is the absolute value of the differential coefficient of the van der Waals' force and is expressed as follows:

$$k_0 = \frac{72}{d_0^2}. \tag{3}$$

The initial forces $f_i(d)$ are defined from the disposition of the initial bubbles, and the final position should be determined by enforcing the system of (1) by iteration until a stable state is achieved. A filling ratio Q is used as the stability criterion. Q is defined as follows:

$$Q = \frac{V_{\text{bubble}}}{V_{\text{region}}} \tag{4}$$

where V_{bubble} is the volume of total bubbles and V_{region} is the volume of the analysis region, respectively.

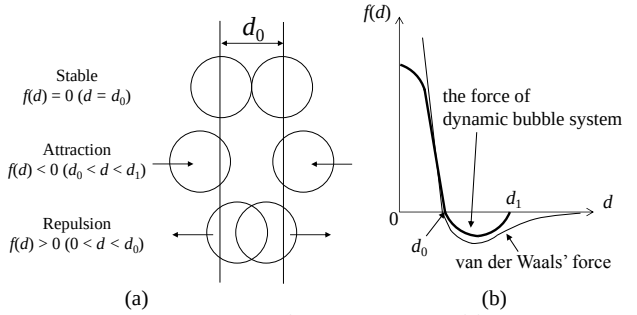


Fig. 2. Interbubble proximity-based forces. (a) Change of force depending on distance between two bubbles, and (b) simplified approximate force.

III. Subdivided Dynamic Bubble System

A. Subdivided Dynamic Bubble System

In the proposed subdivided dynamic bubble system, firstly, bubbles are generated independently in every tetrahedral subdivision of the initial coarse mesh. Here, the tetrahedral subdivisions are generated by connecting the nodes of the initial input data with the Delaunay division, as shown in Fig. 3. Then, all the generated bubbles move according to the force between the bubbles within each tetrahedral subdivision. Remote bubbles adequately far from one bubble are not considered to practically affect it. In the proposed system, since the bubble movement is performed independently in every tetrahedral subdivision, the total computation for distance between far bubbles is reduced for shortening the computation cost.

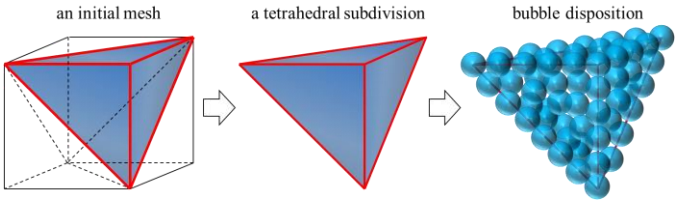


Fig. 3. Examples of an initial coarse mesh, a tetrahedral subdivision, and bubble disposition.

The previous dynamic bubble system [4] has a problem with respect to the memory size of GPU. Because the space of considering the bubble force was restricted to the initial subdivision in the proposed method, the used memory of GPU can be saved. Eventually, employing subdivisions makes the computation cost lower and the used memory size smaller, and the problem of the pervious bubble system using GPGPU is

dissolved.

We propose the subdivided dynamic bubble system, which moves the bubbles independently in every tetrahedral subdivision of the initial mesh. Fig. 4 shows the procedure of the subdivided dynamic bubble system. Here, N_e is the number of tetrahedral subdivisions of the initial coarse mesh.

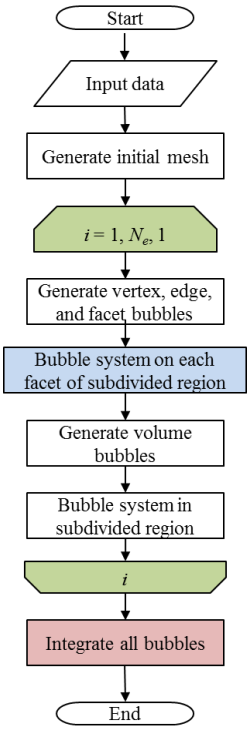


Fig. 4. Flowchart of the subdivided ordinary dynamic bubble system.

B. Treatment of Bubbles on Subdivision Facet

After the activation of the bubble system in each subdivision, it is necessary to merge the bubbles in all the subdivisions. However, there is a problem that the facet belonging to two subdivisions has the different bubble disposition. Since the facet bubbles are generated randomly and move on facet in each subdivision, the bubble dispositions on the common facet differ from each other. It is, therefore, impossible to merge the common facet with the different bubbles dispositions, as shown in Fig. 5. Hence, the following procedure is added in order to identify the bubble disposition on the common facet in the proposed algorithm.

In the loop process of Fig. 4, the facet bubble system is doubly applied to every common facet of subdivisions. In the middle of this process, when the bubbles disposition on the facet belonging to a subdivision has been already obtained in the previous loop, the identical bubble disposition is adopted on the common facet of the other subdivision. By this algorithm, the common facet of the neighboring subdivisions has the same bubble disposition. The bubbles inside the subdivisions, which are surrounded the subdivision facets with the bubble disposition, are disposed using the ordinary dynamic bubble system.

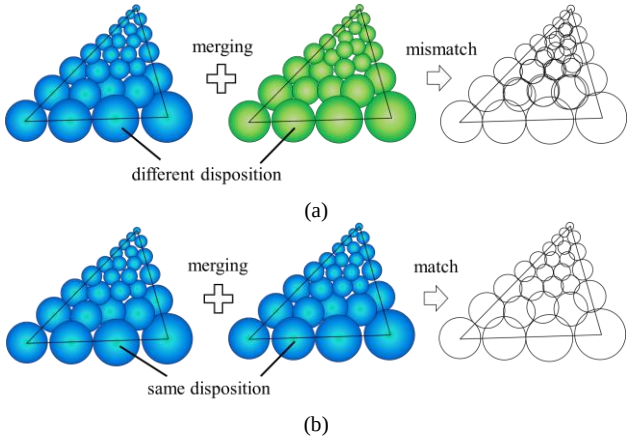


Fig. 5. In the procedure of merging facets, (a) facets with different bubble's disposition are mismatched and (b) facets with same disposition are matched.

IV. Comparison of Computation Time

The usefulness of the automatic mesh generation employing the subdivided dynamic bubble system even with CPU is verified on the TEAM Workshop Problem 20. Fig. 6 shows the schematic drawing of the TEAM Workshop Problem 20 in 3-D space. Fig. 7 shows the initial subdivision which is generated to parallelize the mesh generation. The number of subdivisions in the initial mesh is 80.

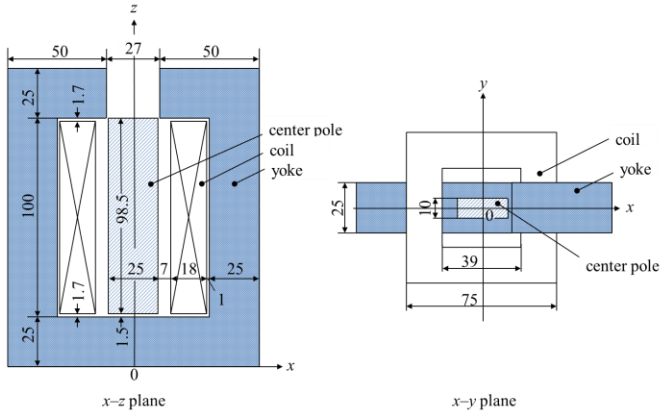
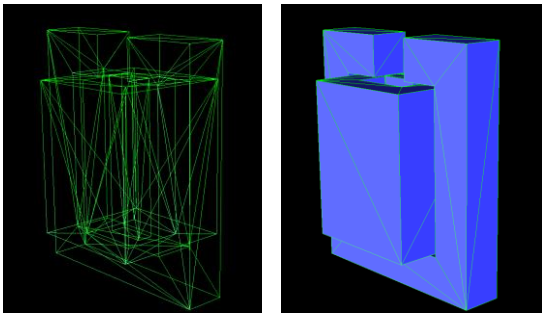


Fig. 6. The schematic drawing of TEAM Workshop Problem 20 in 3-D space.



(a) (b)

Fig. 7. Initial subdivision of TEAM Workshop Problem 20. It is generated by the parallelized dynamic bubble system. (a) General view and (b) outside surface view. The mesh of air region is not depicted.

Fig. 8 shows the comparison of the computation time with the ordinary and the subdivided dynamic bubble system with CPU (Intel Core i7 950 QuadCore 3.06 GHz, 6 GB). Here, an

acceleration rate γ defined as follows is used as an indicator of acceleration.

$$\gamma_1 = \frac{t_o}{t_s} \tag{5}$$

where t_o and t_s are the computation time in the case of the ordinary and the subdivided dynamic bubble system, respectively.

The acceleration rate γ_1 is approximately 7. However, when the number of bubbles was more than one million, the ordinary dynamic bubble system could not converge in a realistic time. The reason is that it is necessary to access all over a wide common memory to obtain the position and size of bubbles. In addition, it takes too long time to compute bubbles' movement. However, the subdivided dynamic bubble system could converge since the memory accessed to calculate the force of one bubble was restricted in a narrow area. The acceleration of the dynamic bubble system is achieved for the mesh generation of a large-scale model.

In order to aim to accelerate furthermore, the subdivided dynamic bubble system is parallelized with GPU [8]. Fig. 9 shows the comparison with the ordinary dynamic bubble system with CPU and the proposed subdivided system parallelized with GPGPU (NVIDIA Tesla C2050, 448 cores, 1.15 GHz, 3 GB). Here, t_{sp} is the computation time in the case of the later, respectively. The acceleration rate γ_2 (= t_o / t_{sp}) exceeds 10 and much higher than that without GPGPU.

Fig. 10 shows the comparison with the subdivided dynamic bubble system with CPU and that parallelized with GPGPU. The subdivided dynamic bubble system with GPGPU successfully generates a mesh consisted of 1,500,000 bubbles faster than that without GPGPU. The more bubbles are generated, the higher acceleration rate γ_3 (= t_s / t_{sp}) will be obtained. However, when the number of bubbles is small, the acceleration rate is less than 1. The reason is that the computation power of CPU exceeds the parallelization power of GPGPU because of smallness of the number of bubbles. The computation time of both systems becomes nearly equal in case of 300,000 bubbles. Therefore, 300,000 bubbles is the criterion whether a user select a use of CPU or GPGPU.

V. Conclusion

In this paper, we propose an automatic mesh generation using a subdivided dynamic bubble system and parallelize the method with GPGPU. As the result, we have achieved a mesh generation of a large-scale model that the ordinary dynamic bubble system failed to generate a mesh, and the acceleration rate exceeds 5 by adopting both subdivision and parallelization with GPGPU.

References

[1] I. Tsukerman, "Accurate computation of 'ripple solutions' on moving finite element meshes," *IEEE Trans. Magn.*, vol. 31, no. 3, pp. 1472-1475, May 1995.

[2] I. Tsukerman, "A general accuracy criterion for finite element approximation," *IEEE Trans. Magn.*, vol. 34, no. 5, pp. 2425-2428, Sep. 1998.

[3] V. Cingoski, R. Murakawa, K. Kaneda, and H. Yamashita, "Automatic mesh generation in finite element analysis using dynamic bubble system," *Journal of Applied Physics*, vol. 81, no. 8, pp. 4085-4087, Apr. 1997.

[4] T. Yokoyama, V. Cingoski, K. Kaneda, and H. Yamashita, "3-D Automatic Mesh Generation for FEA Using Dynamic Bubble System," *IEEE Trans. Magn.*, vol. 35, no. 3, pp. 1318-1321, May 1999.

[5] S. Nagakura, S. Noguchi, K. Kaneda, H. Yamashita, and V. Cingoski, "Automatic quadrilateral mesh generation for FEM using dynamic bubble system," *IEEE Trans. Magn.*, vol. 37, no. 5, pp. 3522-3525, May 2001.

[6] Y. Marechal, D. Armand, and D. Ladas, "An adaptive remeshing technique ensuring high quality meshes," *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 1222-1225, Jun. 2008.

[7] D. N. Shenton and Z. J. Cendes, "Three-Dimensional Finite Element Mesh Generation Using Delaunay Tessellation," *IEEE Trans. Magn.*, vol. 21, no. 6, pp. 2535-2538, Nov. 1985.

[8] F. Nobuyama, S. Noguchi, and H. Igarashi, "The Parallelized Automatic Mesh Generation Using Dynamic Bubble System with GPGPU," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1677-1780, May 2013..

[9] K. Shimada and D. C. Gossard, "Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing," *ACM 3rd Symposium on Solid Modeling and Applications*, pp. 409-419, 1995.

[10] J. H. Kim, H. G. Kim, B. C. Lee, and S. Im, "Adaptive mesh generation by bubble packing method," *Structural Engineering and Mechanics*, vol. 15, no. 1, pp. 135-149, 2003.

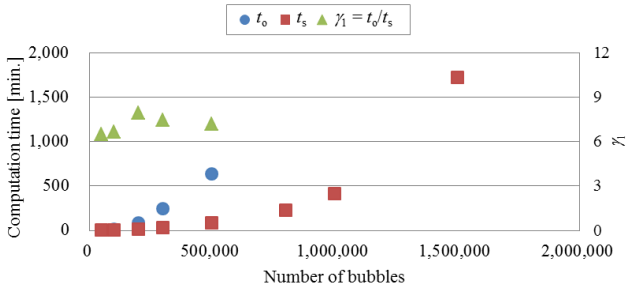


Fig. 8. Comparison of computation times with the ordinary dynamic bubble system and the subdivided dynamic bubble system (CPU computation).

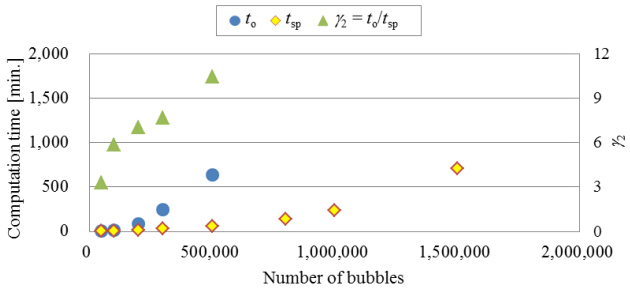


Fig. 9. Comparison of computation times with the ordinary dynamic bubble system with CPU and the subdivided dynamic bubble system with GPGPU.

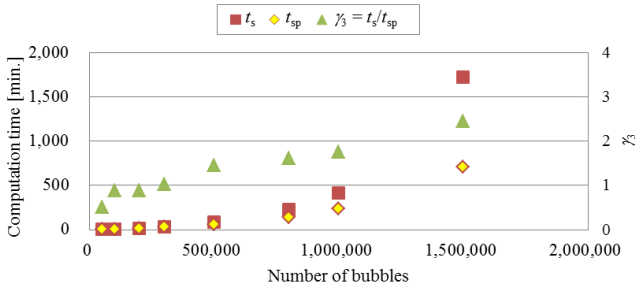


Fig. 10. Comparison of computation times with the subdivided dynamic bubble system (CPU computation) and the one with GPGPU.