Copyright © 2010 Tech Science Press CMES, vol.55, no.1, pp.89-109, 2010

Node Placement Method by Bubble Simulation and Its Application

Ying Liu[1], Yufeng Nie[2], Weiwei Zhang[2] and Lei Wang[2]

Abstract: In the light of the ideas and treatment technologies about molecular dynamics simulation and bubble meshing, a new approach of node placement for the meshless method called node placement method by bubble simulation (NPBS method), is proposed. Nodes are seen as the centers of the bubbles which can be moved by their interacting forces. Through dynamic simulation, bubbles are placed into a near-optimal configuration, and the centers of bubbles will form a good-quality node distribution in the domain. This process doesn't need updating the mesh connection constantly, i.e., is totally meshfree. Some example results show that the uniform point sets and non-uniform point sets generated by NPBS method have good construction, and the non-uniform point sets have also good gradualness which is prefer to have for numerical solution methods of partial differential equations in some case. In addition, NPBS method has good adaptability to complex regions. Since the defined interbubble force is short-range, this method has inherent parallelism. Two simple parallel example results show that the parallel generated node sets are also good-quality. Furthermore, the node sets optimized by bubble simulation are used in the meshless method, and by comparison, the advantage with the less computational error is shown. And it is easy to make use of the information of neighbor bubbles generated by NPBS method for finding the nodes on the influence domain in meshless analysis. Finally, the node sets generated by NPBS method are also used for triangular mesh generation and refinement in finite element method. The results show that the mesh has excellent quality.

Keywords: node placement, meshless methods, parallel, molecular dynamics, bubble simulation.

[1] This paper is partly presented in ICCES MM08. Micro and Nano Electro Mechanical System Laboratory, Northwestern Polytechnical University, Xi'an 710072, China

[2] School of Science, Northwestern Polytechnical University, Xi'an 710129, China. Email: yfnie@nwpu.edu.cn

1 Introduction

In meshless methods, local shape functions are constructed by only using the information of nodes and needn't the mesh topology information. Meshless methods show particular advantages in some fields where mesh-based numerical methods encounter difficulties, e.g., large deformation, crack propagation and free surface flow [Belytschko (1994); Atluri (1998); Atluri (2000); Atluri (2004); Liu (2005); Li (2007); Han (2005); Rabczuk (2006)]. In contrast with an arbitrary set of nodes, a good-quality node distribution will improve the computational accuracy of numerical solution of partial different equations. So it is valuable and necessary to generate a good-quality node set for meshless analysis. And we prefer to mention that a good node distribution will help to form high quality mesh which is necessary for mesh-based numerical methods, such as finite element method [Frey (2008); Zienkiewicz (2008)], node-based finite element method [Nie (2007); Nie (2009)], etc. There exist several node placement methods of the meshless methods. In reference [Klaas (2000)], the node set is composed of the corner nodes of the octants. The biting method in reference [Li (2000)] uses an advancing front strategy to generate a good sphere packing and their centers constitute a well-spaced node set. And Du et al. proposed a node placement method using centroidal Voronoi tesselations [Du (2002)].

Furthermore, the support region of the integration point includes more nodes, and its shape functions aren't polynomials usually. So the meshless methods need significantly more computational cost to evaluate integration than the mesh-based methods. To reduce the computational cost and especially to solve large-scale problems, many parallel meshless methods are proposed. Singha et al. proposed a parallel algorithm based on the data decomposition approach for the solution of a sparse system of linear equations in the element free Galerkin method [Singha (2005)]. And Lu et al. gave the parallel algorithms of several important components in the element free Galerkin method [Lu (2006)]. In [Danielson (2000)], a parallel computational implementation of modern meshless methods is presented for explicit dynamics analysis. In [Cartwright (2006)], Cartwright et al. focused on parallel support set searches for meshfree methods.

In this paper, we focus on node placement, parallelization and searching the nodes in the meshless methods. In Section 2, in light of molecular dynamics [Srinivasan (1997); Heffelfinger (2000); Allen (2004); Fleissner (2007)] and bubble mesh generation by Shimada et al. [Shimada (1995)], we present a new node placement method and point out that this method is easy to parallelize. For uniform and non-uniform point distribution in simple and complex regions, computational example results are given. And parallel node placement method is also verified by a simple parallel example. In Section 3.1, the node set optimized by bubble simulation is used in the Element Free Galerkin (EFG) method [Belytschko (1994)], and by comparison, the computational error is less. In Section 3.2, the generated information of neighbor bubbles is used for selecting for influence radius in meshless analysis. In Section 4, a node set generated by NPBS method are connected by constrained Delaunay triangulation, and after modifying node-spacing function, initial mesh can be locally refined.

2 Node placement method by bubble simulation and its parallelism

In the light of bubble mesh method and molecular dynamics, we presented node placement method by bubble simulation (NPBS method). Bubble mesh method is a kind of unstructured mesh generation method, and has been successfully applied to two- and three-dimensional meshing [Shimada (1995); Cingoski (1997); Shimada (1998); Yokoyama (1999)], anisotropic triangular meshing of parametric surfaces [Shimada (1997); Yamakawa (2000)] and adaptive analysis in finite element method [Kim (2003)]. It is based on the observation that a pattern of tightly packed bubbles mimics a Voronoi diagram, from which a set of well-shaped Delaunay triangles and tetrahedral can be created by connecting the centers of the bubbles [Shimada (1995)]. Fig. 1 illustrates Voronoi polygons, packed bubbles and Delaunay triangles. Firstly, initial nodes are placed by recursive spatial subdivision in the domain. Secondly, the mesh is relaxed by assuming the presence of proximity-based, repulsive/attractive inter-node forces and then performing dynamics simulation for a force-balancing configuration of nodes [Shimada (1998)]. In the end, well-shaped Delaunay triangles can be generated.

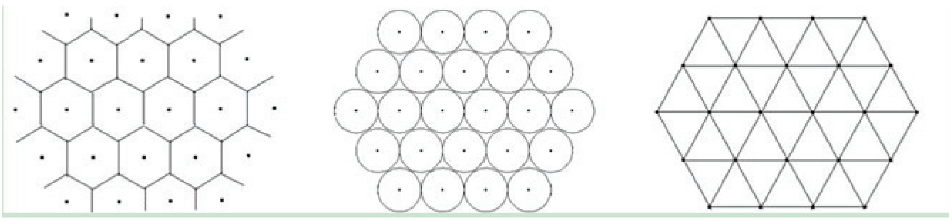


Figure 1: Voronoi polygons, packed bubbles, Delaunay triangles

Bubble mesh method usually need to find adjacent bubbles in dynamics simulation by using constrained Delaunay triangulation, and re-triangulation costs much time. However, for the meshless methods, only nodes but triangles are needed. In order to save time and make the total process of node generation being independent of the mesh, we make use of the locality of the interacting force to construct a neighbor bubble set of each bubble, and renew timely the neighbor bubble sets to compute a result force of each bubble in the simulation. This technology, called Verlet list [Allen (2004)], is usually used in molecular dynamics simulation. And the molecular dynamics method has come to be widely employed for gaining atomistic insight into many materials properties. The corresponding parallel algorithms have also been deeply studied.

2.1 Node placement method by bubble simulation

The main steps of NPBS method are given as follows: First, an initial node set is positioned in the domain. It is significant to obtain a good initial bubble configuration for speeding up the simulation. Then nodes are seen as bubbles, and bubbles are moved by their interacting forces which are analogous to the van der Waals force. When two bubbles are too close, a repulsive force is applied; when two bubbles lie farther apart than the stable distance, an attractive force is applied. Finally the centers of bubbles will form a good-quality node distribution in the domain. The flow chart of NPBS method is shown in Fig. 2.

2.1.1 Input information

The inputs for NPBS method include the computational domain and the desired node-spacing function. The computational domain in two-dimension is described by Planar Straight Line Graph (PSLG) [Zhang (2005)] which is a set of vertices and segments. And some holes can also be included. The desired node-spacing function d can be defined by users, or be changed with a priori and a posterior error estimates in the adaptive numerical computation [Babuvška (1978)].

2.1.2 Initial bubble configuration

It is important to obtain a good initial bubble configuration for saving the time of simulation. Bubbles are placed successively on the vertices, edges and surfaces of the domain by the recursive bisection technology [Shimada (1995)]. Especially, when bubbles are placed on the surfaces, rhombic cells with inside angles of 60° and 120° are used instead of square cells in order to realize hexagonal arrangement of the bubbles [Shimada (1998)]. An example is given as follows. In a square domain with a hole, the desired node-spacing function is:

$$d(x,y) = \left(x^2+y^2\right)^{1/2}/8 \tag{1}$$

The placed initial bubbles can be seen in Figure 3.

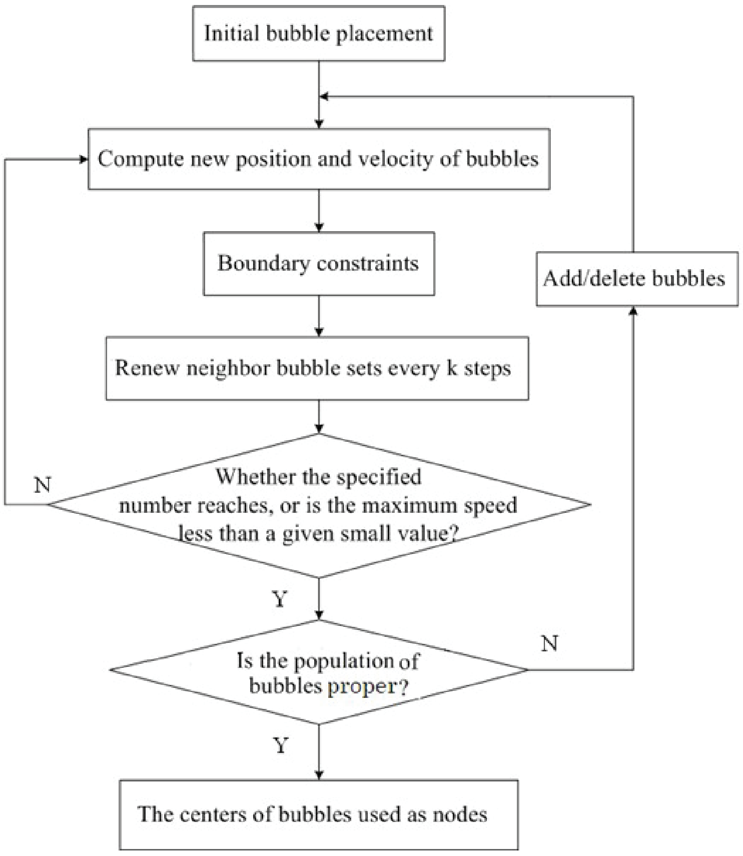

Initial bubble placement
Compute new position and velocity of bubbles
Boundary constraints
Add/delete bubbles
Renew neighbor bubble sets every k steps
N
Whether the specified number reaches, or is the maximum speed less than a given small value?
Y
N
Is the population of bubbles proper?
Y
The centers of bubbles used as nodes

Figure 2: The flow chart of node placement method by bubble simulation

2.1.3 Computation of the motion of bubbles

In order to reduce the computational cost and avoid the force from growing infinitely large when the distance between two bubbles is close to zero, the interbubble force is approximated by the 3rd order polynomials [Yamakawa (2000)]

$$f(w) = \begin{cases} k_0\left(1.25w^3 - 2.375w^2 + 1.125\right) & 0 \le w \le 1.5 \\ 0 & 1.5 < w \end{cases} \tag{2}$$

instead of the van der Waals force, where w is the ratio of the real distance and the desired distance between the bubbles, and the parameter k_0 defined by user can control the magnitude of the force.

The motion equation of the ith bubble is a second order ordinary differential equa-

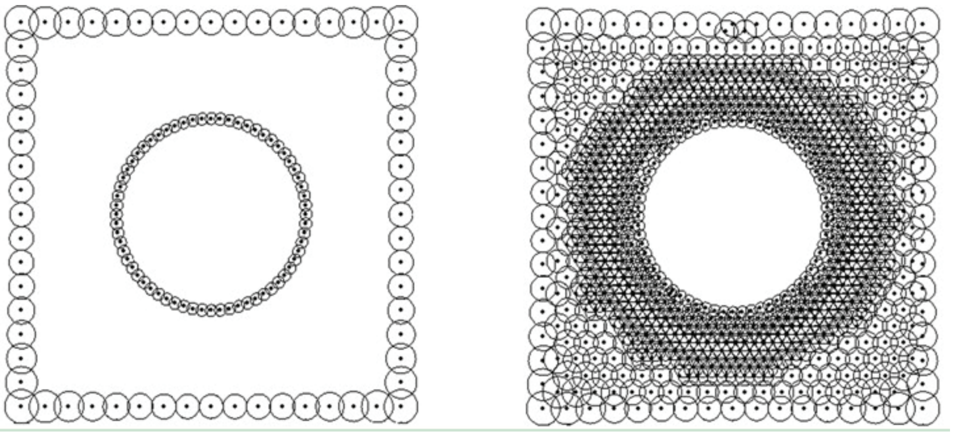


Figure 3: Initial bubble placement

tion [Shimada (1995)]:

$$m\frac{d^2x_i(t)}{dt^2}+c\frac{dx_i(t)}{dt}=F_i(t) \tag{3}$$

where m is the coefficient of mass, and c is the coefficient of damping. The resultant force F_i of the i*th* bubble depends on the interacting forces between it and its neighbor bubbles. The ordinary differential equation system is integrated through time and solved iteratively. The 4*th* order Runge-Kutta method can be used here. So after each time step Δt, the position and velocity of each bubble is renewed.

To find the neighbor bubbles of each bubble at each step, similar to the Verlet list [Allen (2004)] in the molecular dynamics simulation, the neighbor bubble sets are built initially by a series of buckets in the domain (Fig. 4 left), and must be renewed timely. The new neighbor bubble set of the ith bubble could include the old neighbor bubbles and their neighbor bubbles (Fig. 4 right), but the distances between them and the ith bubble must be smaller than 1.7σ, where σ is the ideal distance between two bubbles. So the adjacency bubble linked list of the ith bubble contains enough neighbor bubbles, and need be only renewed every k steps.

The treatment of boundary constraints is very important to the quality of the node set generated by the node placement method. After each time step Δt, except vertex bubbles on the boundaries and some special bubbles, the bubbles which moved outside the domain can be projected into the domain and keep a certain distance from the boundaries [Zhang (2005)]. And the bubbles on the boundaries must be kept on the boundary.

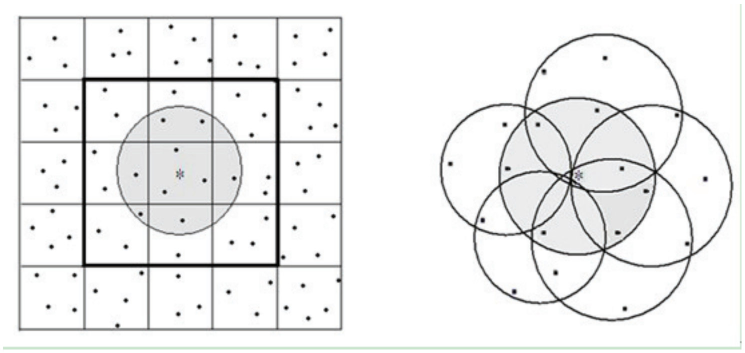


Figure 4: The neighbor bubble set of the bubble

2.1.4 Adjusting the total number of bubbles

Usually, initial bubble placement can not generate the proper number of bubbles. So by computing the overlapping ratio [Shimada (1995)], delete the bubble whose overlapping ratio is too large; add new bubbles near the bubble whose overlapping ratio is too small. The total number of node set is changed, then bubble simulation is reprocessed (Fig. 5).

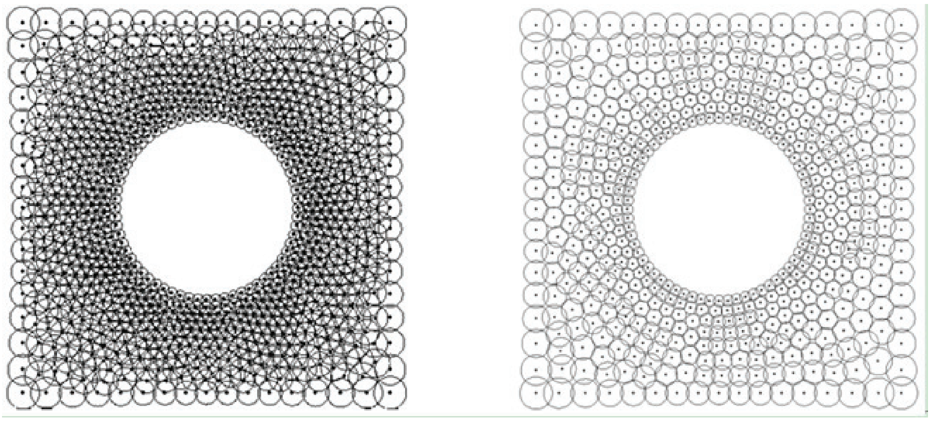


Figure 5: Before and after adjusting the total number of bubbles

2.1.5 Numerical examples

Several numerical examples in convex and non-convex domains with uniform and non-uniform distribution are provided for showing the performance of this method.

Firstly, in the above example 1, set the force coefficient $k_0 = 1$, the mass coefficient $m = 1$, the damping coefficient $c = 3.8429$ and the time interval $\Delta t = 0.02$. The left picture of Fig. 5 shows the result after 750 steps. Obviously, the total number is too large and isn't proper. So it is necessary to adjust the total number of bubbles, then bubble simulation is reprocessed. It can be seen that the quality of node distribution is improved in Fig. 6.

For the non-convex domains in Fig. 7, Fig. 8 and Fig. 9, the desired node-spacing function d is a constant, equal to 0.05. So the node distribution is uniform. The basic coefficients of simulation are the same as above. After 30 steps, 5 steps and 6 loops each with 250 steps, we can see that the node sets generated respectively have good construction in Fig. 7, Fig. 8 and Fig. 9, especially near the boundary of domain. It means that this method is adaptable to the complex regions.

Fig. 10 shows the non-uniform node set in the circle domain after 500 steps. Near the circle $x^2+y^2= 0.09$, the density of nodes is larger, and its desired node-spacing function is:

$$d(x,y) = 0.04 \cdot \left|\sqrt{x^2+y^2} - 0.3\right| / 0.3 + 0.02. \tag{4}$$

The basic coefficients of simulation are the same as above. This example illustrates that the non-uniform point set generated by this method have good construction and gradualness. In the square domain of Fig. 11, we hope node (0, 0.25) is a densified node. Its desired node-spacing function is:

$$\begin{aligned} d(x,y) = \min\Big(&0.06 \cdot \sqrt{x^2 + (y-0.25)^2}/0.5 + 0.02, \\ &0.06 \cdot \sqrt{x^2 + (y-0.75)^2}/0.5 + 0.02, 0.05\Big) \end{aligned} \tag{5}$$

The expected result is generated after 750 steps. Fig. 12 shows the non-uniform node set in the L-type domain after 10 loops with 250 steps and adjusting the total number of nodes. The corner node is a densified node and its desired node-spacing function is:

$$d(x,y) = \min\left(0.16 \cdot \sqrt{(x-0.4)^2 + (y-0.6)^2} + 0.01, 0.08\right) \tag{6}$$

In Fig. 13, seven fingers in comb structure need be densified, and its desired node-spacing function is:

$$d(x,y) = \begin{cases} 0.015 & x > 0.7 \\ \min\left((0.7-x)/5 + 0.015, 0.05\right) & x \le 0.7 \end{cases} \tag{7}$$

The final result is generated after 6 loops with 250 steps.

Through the above examples, it is obvious that this method is suitable for uniform and non-uniform distribution in simple and complex domain and can generate the good-quality node sets. And this method is simple and easy to implement. The existed node set can also be optimized by bubble simulation. In addition, it can be extended to three-dimensional node placement and anisotropic node placement by using anisotropic node-spacing function.

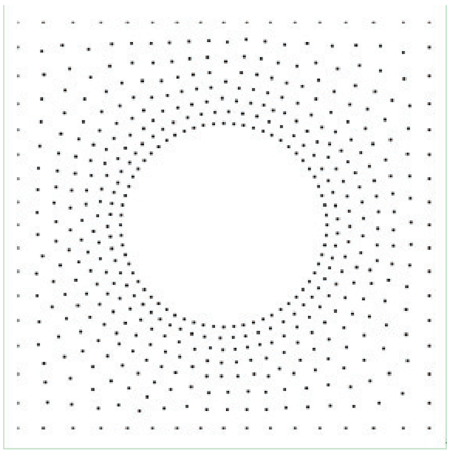


Figure 6: The generated node set of example 1

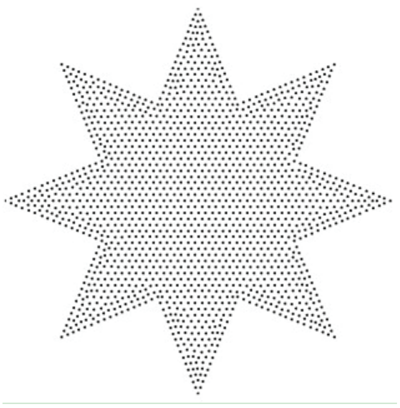


Figure 7: The generated uniform node set in an octagram domain

2.2 Parallelism

The defined interbubble force is short-range, meaning that each bubble interacts only with other bubbles that are geometrically nearby. The force computations are spatially local. So coordinate and velocity updates for two distant bubbles can be performed simultaneously and independently in the simulation, i.e., NPBS method has natural parallelism. NPBS method is analogous to a molecular dynamics simulation. And parallel molecular dynamics methods have been being widely studied. There are three main parallel decomposition schemes: domain decomposition, data decomposition and force decomposition [Heffelfinger (2000); Plimpton (1995)]. In comparison to molecular dynamics, the defined interbubble force is simple, so the computation cost will account for less percentage. In view of this feature, it is recommended to use domain decomposition for NPBS method.

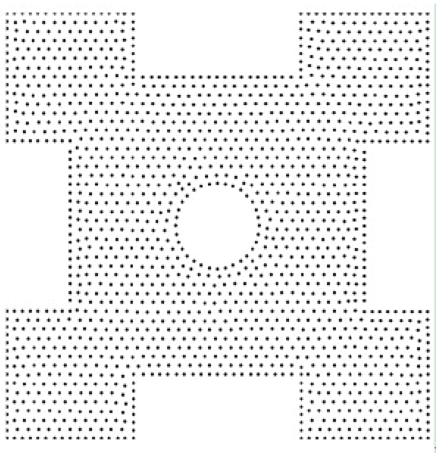


Figure 8: The generated uniform node set in a non-convex domain with a hole

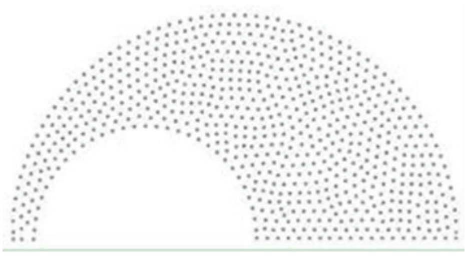


Figure 9: The generated uniform node set in a trunk-like domain

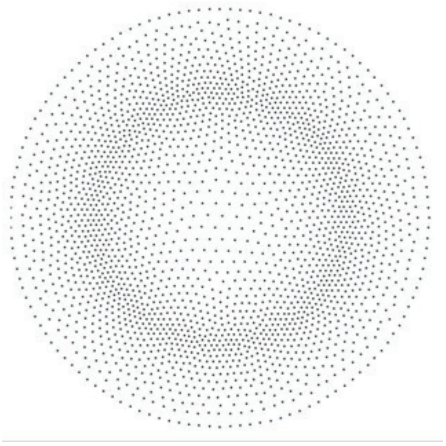


Figure 10: The generated non-uniform node set in a circle domain

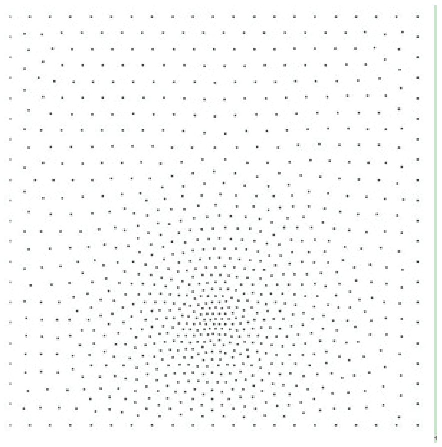


Figure 11: The generated non-uniform node set in a square domain

2.2.1 Parallel algorithm

First, the whole domain is geometrically decomposed into disjoint sub-domains (MPI programming on the parallel computer "Lenovo DeepComp 1800"). Each sub-domain is called as a computational domain which will be assigned to a different processor. Each processor only computes the motion of bubbles placed in it. But neighbor bubbles of the bubble near the domain segmentation line may fall on

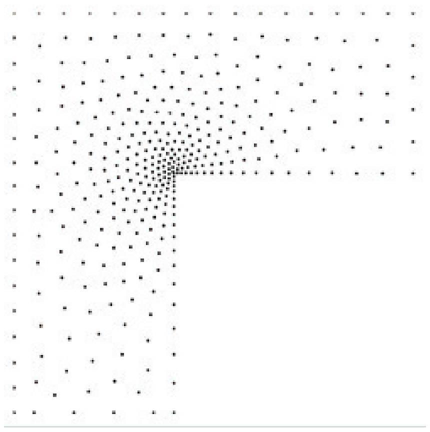


Figure 12: The generated non-uniform node set in a L-type domain

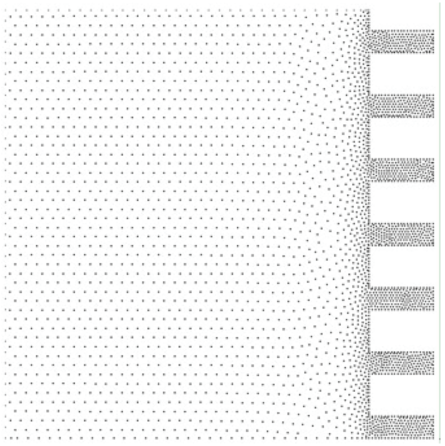
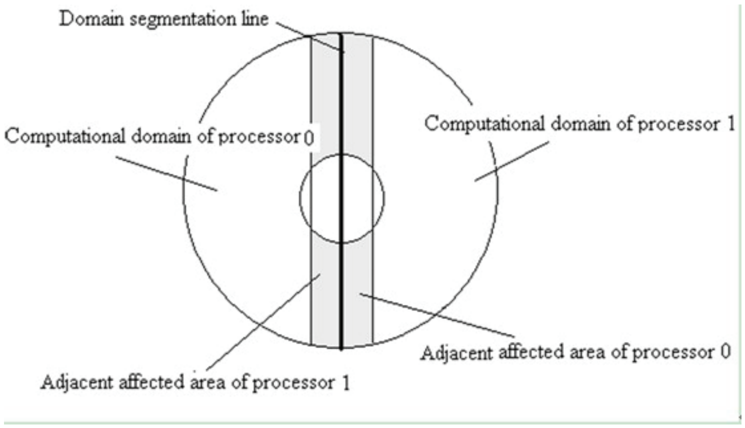


Figure 13: The generated non-uniform node set in a comb domain

neighbor sub-domains. So each processor also must store bubbles in the neighbor sub-domain and renew the position and velocity of them in the simulation. Because the interbubble force is short-range, in order to reduce unnecessary information exchange, the corresponding adjacent affected areas are built, and each processor only need to store a part of bubbles in neighbor sub-domains, see Fig. 14. The width of the adjacent affected area is: $1.5\max d(x,y)$.


Domain segmentation line
Computational domain of processor 0
Computational domain of processor 1
Adjacent affected area of processor 0
Adjacent affected area of processor 1
Figure 14: Domain decomposition

After initial node placement and domain decomposition, it begins to simulate in parallel. Every $s$steps, the bubble information must be interchanged with each other. Four concrete steps are as follows:

1. Transfer bubbles which moved to the neighbor sub-domains in the simulation to the corresponding processors;
2. Accept bubbles which moved to its sub-domain, and get a new bubble set for the next simulation stage;
3. Transfer the new bubble information of bubbles fell on the adjacent affected areas to the corresponding processors;
4. Accept the new bubble information in the adjacent affected areas from the other processors.

Therefore, if one bubble moves to the neighbor sub-domain in the process of simulation, its position is marked, and will not change in this simulation stage. At the beginning of the next simulation stage, this bubble will be transfer to the processor of the neighbor sub-domain.

2.2.2 Parallel examples

We tested the parallel performance of this method by using a simple parallel example. For a ring domain, its center is (0,0), and the desired node-spacing function in it is:

$$d(x,y) = \left(x^2+y^2\right)^{1/2}/6 \tag{8}$$

The node set is generated by parallel bubble simulation in two processors and four processors. Fig. 14 and Fig. 15 shows domain decomposition and the initial node set, respectively.

In the parallel simulation, the value of s is 50. After 20 times of data communication, the left picture of Fig. 16 shows the parallel result of two processors. Its speedup ratio is 1.6042, and its parallel efficiency is 0.8021. As shown in the right picture of Fig. 16, the parallel result of four processors is also good. But its parallel performance is not so good. One of its reasons is that this example includes so few nodes (the total number of nodes is 1192) that the communication cost takes a large proportion. Further research about parallel realization of NPBS method is still needed.

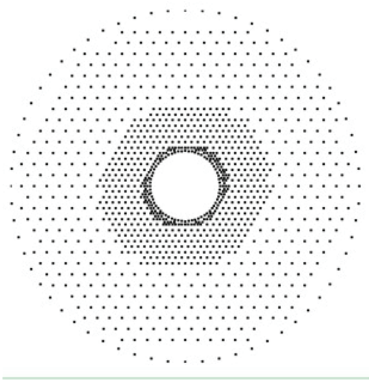


Figure 15: Initial placement

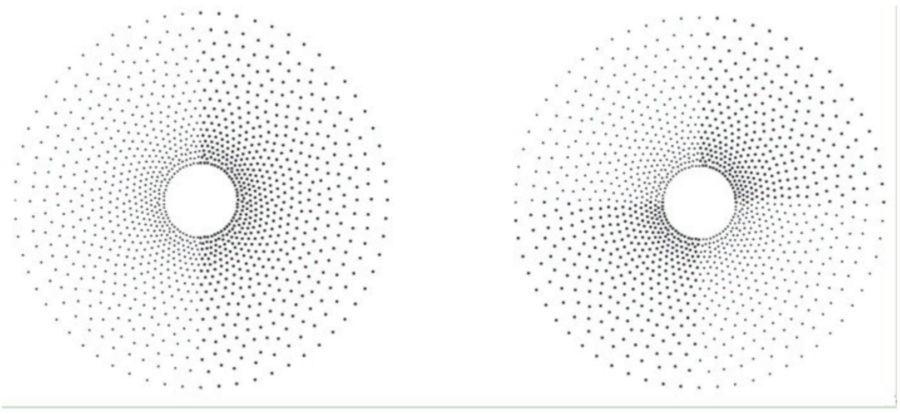


Figure 16: The result of parallel node placement by bubble simulation

3 Its application in meshless methods

3.1 Impact of node quality on the computational error in meshless method

For the sake of studying the effect using the node set optimized by bubble simulation in the meshless method on the calculation error, we solve the Poisson equation with Dirichlet boundary condition by the element free Galerkin method with four different node sets.

$$\begin{cases} -\Delta u = -\left(x^2+y^2\right) e^{xy} \;\; in\, \Omega \overset{\Delta}{=} (0,1)\times(0,1) \\ u|_{x=0} = 1,\; u|_{x=1} = e^y,\; u|_{y=0} = 1,\; u|_{y=0} = e^x \end{cases} \tag{9}$$

Shape functions are constructed by moving least squares method [Belytschko (1994)] with linear basis. The quartic spline function [Nguyen (2008)] is used as the weight function. 12×12 uniform background cells are introduced and 2×2 Gaussian integral algorithm is used in each cell. The essential boundary conditions are imposed by the penalty method [Nguyen (2008)]. The influence radius is given by 0.15 [Nie (2006)]. See Fig. 17, the picture in the top left corner is a random distributed node

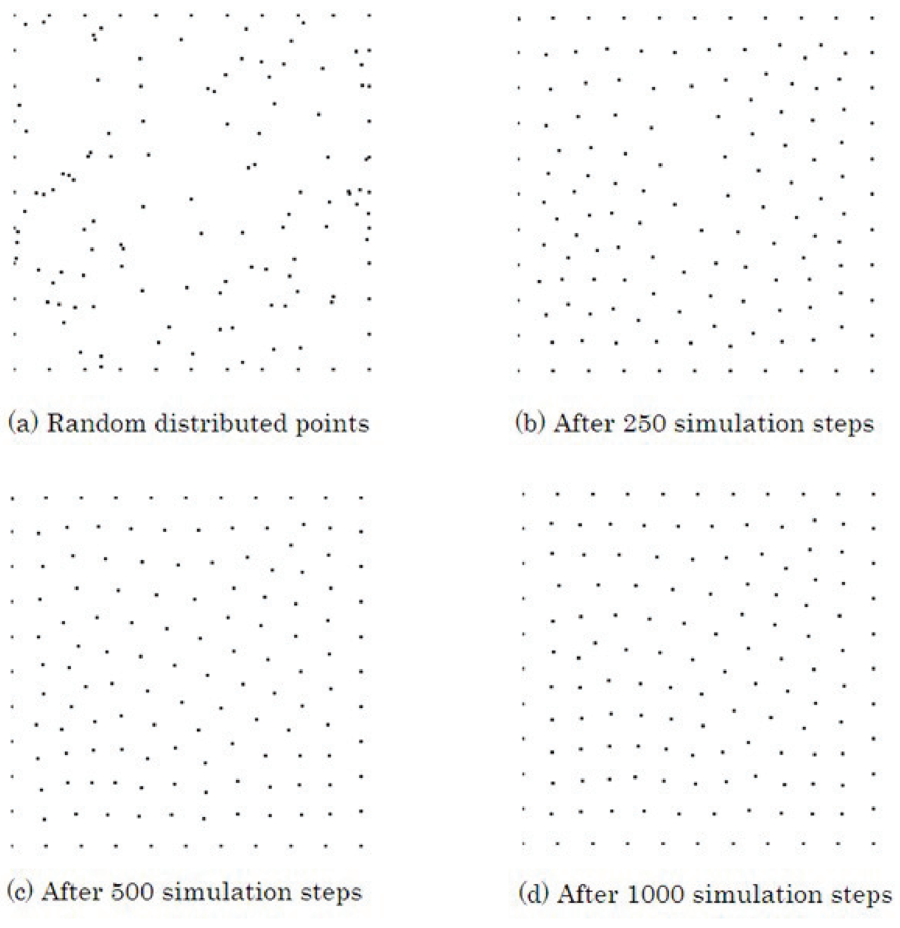


Figure 17: The result of parallel node placement by bubble simulation (The same number of nodes)

set of 144 nodes in [0,1]×[0,1]. In the top right corner is the result after 250 simulation steps. In the lower left corner is after 500 simulation steps. The last picture is after 1000 simulation steps. Its exact solution of problem (8) is $u(x,y) = e^{xy}$.

Sobolev Norm $\|\cdot\|_t$ which is defined as follows is used to measure computing error.

$$\|f\|_t = \left(\sum_{l=0}^{t} \sum_{|\alpha|=l} \int_\Omega [D^\alpha f(x)]^2 d\Omega \right)^{1/2}, \quad f \in H^t(\Omega),$$
$$\Omega \subset R^d, D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial x_2}\right)^{\alpha_2} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d}, \quad \alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d) \tag{10}$$

So Tab. 1 shows the calculation errors using these four different node sets. We can see that bubble simulation results in better node distribution. And the node set optimized by bubble simulation is suitable for the meshless method.

Table 1: The calculation errors using four different node sets

Node placement schemes	Calculation errors $\|\cdot\|_0$	Calculation errors $\|\cdot\|_1$
a	0.390010815	2.933037054
b	0.004286155	0.041298026
c	0.000408937	0.016183909
d	0.000378625	0.014812266

3.2 Finding the nodes on the influence domain

The resultant force of each bubble depends on the interacting forces between it and its neighbor bubbles. The neighbor bubble set of every bubble is stored in adjacency bubble linked list, so after the simulation, it is easy to get the information of neighbor nodes of each node. We can make use of this additional information for finding the nodes within the influence domain of the integration points in meshless analysis. This is one of the advantages of our method. It can help reduce the searching time for nodes within the influence domains after generating the node set. The influence domain sets of the above two examples are as shown in Fig. 18.

4 Its application in triangular mesh generation

Mesh generation is the first step of finite element analysis, and mesh quality has great influence to analysis result. Because of its adaptability to complex domains, Delaunay triangulation is commonly used. Node placement is a very important part of Delaunay triangulation algorithms, and the quality of node set will finally determine the quality of Delaunay mesh. So how to generate a good node set is a research hotspot.

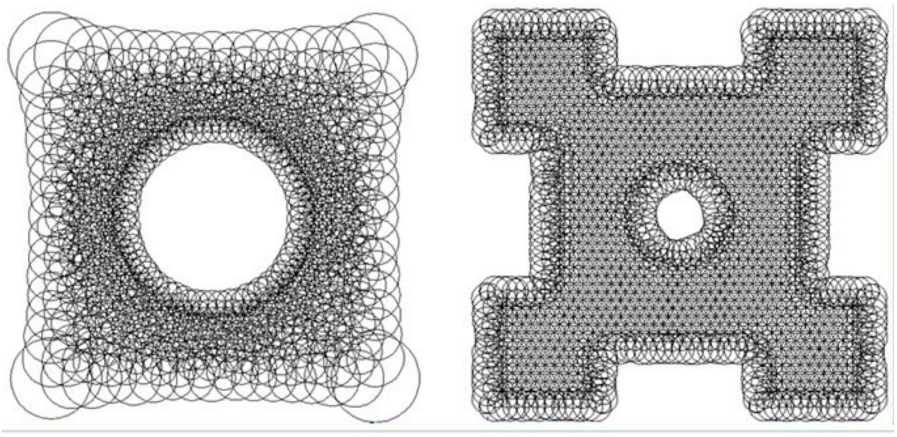


Figure 18: The influence domain set

In this section, the generated nodes of example 1 are connected by constrained Delaunay triangulation algorithm, see Fig. 19. One of mesh quality measure formulas is:

$$q(\mathrm{a,b,c}) = \frac{2r_{in}}{r_{out}} = \frac{(\mathrm{b+c-a}) \times (\mathrm{c+a-b}) \times (\mathrm{a+b-c})}{\mathrm{a} \times \mathrm{b} \times \mathrm{c}} \tag{11}$$

that is, the ratio between the radius of the largest inscribed circle (times two) and the smallest circumscribed circle, where *a*, *b*, *c* are the side lengths [Persson (2004)]. The generated mesh quality result in Fig. 19 shows NPBS method can be used to generate initial node set for triangular mesh generation, and the mesh connected by constrained Delaunay triangulation algorithm has good quality.

Furthermore, if the generated triangular mesh need be refined, firstly, modify the desired node-spacing function:

$$d(x,y) = \begin{cases} \sqrt{x^2+y^2}/8 & x^2+y^2 > 0.7^2 \\ \sqrt{x^2+y^2}/11 & 0.7^2 \geq x^2+y^2 > 0.6^2 \\ \sqrt{x^2+y^2}/14 & 0.6^2 \geq x^2+y^2 > 0.5^2 \end{cases} \tag{12}$$

Secondly, generate the new node set (Fig. 20). In the end, the new mesh is gained by the new node set and constrained Delaunay triangulation algorithm. See Fig. 21, the new mesh after densifying the node set has good quality, and the refinement effect is obvious. So like bubble mesh method, the NPBS method has also good application prospect in triangular mesh generation.

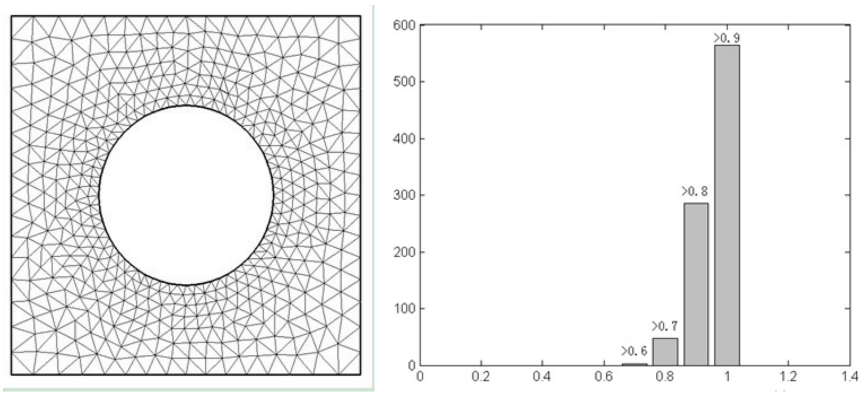


Figure 19: Triangular mesh of the generated nodes in example 1 and mesh quality

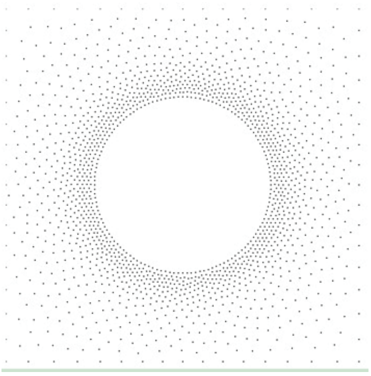


Figure 20: The densified node set

5 Conclusions

NPBS method can generate good-quality node sets efficiently and also can be easily parallelized for large-scale problems. The generated node set is suitable for the meshless analysis, and will be combined with the parallel EFG method. However, these advantages mentioned in this paper are worth further studying. Especially, to improve parallel efficiency, further work is urgently needed.

Acknowledgement: This research was supported by National Natural Science Foundation of China (No: 90916027), Program for New Century Excellent Talents in University (No: NCET-06-0881) and the Scientific Research Foundation for the

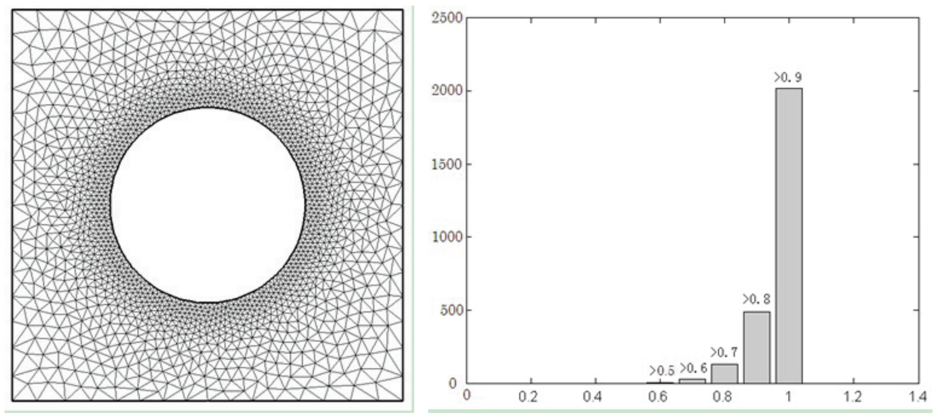


Figure 21: The refined mesh and mesh quality

Returned Overseas Chinese Scholars (No: 2009YK11).

References:

Allen, M. P. (2004): Introduction to Molecular Dynamics Simulation, *Computer Soft Matter: From Synthetic Polymers to Proteins*, Lecture Notes, Norbert Attig, Kurt Binder, Helmut Grubmuller, Kurt Kremer (Eds.), John von Neumann Institute for Computing, Julich, NIC Series, vol. 23, pp. 1–28.

Atluri, S. N. (2004): *The meshless method (MLPG) for domain & BIE discretizations.* Tech. Science Press.

Atluri, S. N.; Kim, H. G.; Cho, J. Y. (1999): A critical assessment of the truly meshless local Petrov-Galerkin(MLPG) and local boundary integral equation (LBIE) methods. Comput. Mech, vol. 24, pp. 348–372.

Atluri, S. N.; Zhu, T. (1998): A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. *Comput. Mech.*, vol. 22, pp. 117–127.

Atluri, S. N.; Zhu, T. (2000): New concepts in meshless methods. Int. J. Num. Meth. Eng., vol. 47, pp. 537–556.

Babuvška, I.; Rheinboldt, W. C. (1978): Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis,* vol. 15, pp. 736–754.

Belytschko, T.; Lu, Y. Y.; Gu, L. (1994): Element-free Galerkin methods, *Journal for Numerical Methods in Engineering,* vol. 37, pp. 229–256.

Cartwright, C.; Oliveira, S.; Stewart, D. E. (2006): Parallel support set searches for meshfree methods. *SIAM Journal on Scientific Computing*, vol. 28, no. 4, pp.

1318–1334.

Cingoski, V. Murakawa, R. Kaneda, K. Yamashita, H. (1997): Automatic Mesh Generation in Finite Element Analysis Using Dynamic Bubble System. *Journal of Applied Physics,* vol.81, no. 8, 4085–4087.

Danielson, K. T.; Hao, S.; Liu, W. K.; Aziz, R.; Li, S. (2000): Parallel computation of meshless methods for explicit dynamic analysis. *International Journal for Numerical Methods in Engineering*, vol. 47, pp. 1323–1341.

Du, Q.; Gunzburger, M.; Ju, L. L. (2002): Meshfree, probabilistic determination of points sets and support regions for meshless computing. *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 13–14, pp. 1349–1366.

Fleissner, F.; Eberhard, P. (2007): Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection. *International Journal for Numerical Methods in Engineering,* vol. 74, no. 4, pp. 531–553.

Frey, P. J.; George, P. L. (2008): *Mesh Generation.* Hermes Science Publications.

Han, Z. D.; Rajendran, A. M.; Atluri, S. N. (2005): Meshless Local Petrov-Galerkin (MLPG) Approaches for Solving Nonlinear Problems with Large Deformations and Rotations. *CMES: Computer Modeling in Engineering & Sciences*, vol. 10, no. 1, pp. 1–12.

Heffelfinger, G. S. (2000): Parallel atomistic simulations. *Computer Physics Communications*, vol. 128, no. 1–2, pp. 219–237.

Kim, J. H.; Kim, H. G.; Lee, B. C.; Im, S. (2003): Adaptive mesh generation by bubble packing method. *Structural Engineering and Mechanics,* vol. 15, no. 1, 135–149.

Klaas, O.; Shephard, M. S. (2000): Automatic Generation of Octree-based Three-Dimensional Discretizations for Partition of Unity Methods. *Computational Mechanics,* vol. 25, no. 2–3, pp. 296–304.

Li, S. F.; Liu W. K. (2007): *Meshfree Particle Methods.* Springer.

Li, X. Y.; Teng, S. H.; Ungor, A. (2000): Point placement for meshless methods using sphere packing and advancing front methods, *ICCES'00*, Los Angeles.

Liu, G. R.; Gu, Y. T. (2005): *An Introduction to Meshfree Methods and Their Programming*. Springer.

Lu, D. T.; Zeng, Q. H.; Lin, C. Y. (2006): Parallel algorithms of meshless numerical simulation. *Journal of University of Science and Technology of China*, vol. 36, no. 12, pp. 1299–1307.

Nie, Y. F.; Atluri, S. N.; Zuo, C. W. (2006): The optimal radius of the support of radial weights used in moving least squares approximation. *CMES: Computer Modeling in Engineering & Sciences,* vol. 12, no. 2, pp. 137–147.

Nie, Y. F.; Chang, S.; Fan, X. K. (2007): The Parallel Mechanism of Node-based Seamless Finite Element method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 19, no. 2, pp. 135–144.

Nie, Y. F.; Liu, Y.; Gu, Y. T.; Fan, X. K. (2009): Fast Searching Algorithm for Candidate Satellite-node Set in NLMG. *CMES: Computer Modeling in Engineering & Sciences*, vol. 45, no. 1, pp. 31–56.

Nguyen, V. P.; Rabczuk, T.; Bordas, S.; Duflot, M. (2008): Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation, v*ol. 79, no. 3, pp. 763–813.

Persson, P. O.; Strang, G. (2004): A simple mesh generator in MATLAB, *SIAM review,* vol. 46, no. 2, pp. 329–345.

Plimpton, S. J. (1995): Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics,* vol. 117, no. 1, pp. 1–19.

Rabczuk, T.; Areias, P. (2006): A Meshfree Thin Shell for Arbitrary Evolving Cracks Based on An Extrinsic Basis. *CMES: Computer Modeling in Engineering & Sciences*, vol. 16, no. 2, pp. 115–130.

Shimada, K. (1997): Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles. *Proceedings of 6th International Meshing Roundtable*, Park City, pp. 375–390.

Shimada, K.; Gossard, D. C. (1995): Bubble Mesh: Automated Triangular Meshing of Non-Manifold Geometry by Sphere Packing. *Proceedings of Solid Modeling Applications*, Salt Lake City, pp. 409–419.

Shimada, K.; Gossard, D. C. (1998): Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis. *Computer Aided Geometric Design,* vol. 15, pp. 199–222.

Singha, I.V.; Jainb, P. K. (2005): Parallel EFG algorithm for heat transfer problems. A*dvances in Engineering Software*, vol. 36, no. 8, pp. 554–560.

Srinivasan, S. G.; Ashok, I.; Jônsson, H.; Kalonji, G.; Zahorjan, J. (1997): Dynamic-domain- decomposition parallel molecular dynamics. *Computer Physics Communications*, vol. 102: 44–58.

Yamakawa, S.; Shimada, K. (2000): High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing. *Proceedings of the 9th International Meshing Roundtable*, New Orleans, pp. 263–274.

Yokoyama, T.; Cingoski, V.; Kaneda, K.; Yamashita, H. (1999): 3-D Automatic Mesh Generation for FEA Using Dynamic Bubble System. *IEEE Transactions on*

Magnetics, vol. 35, no. 3, pp. 1318–1321.

Zhang, H. Z.; Smirnov, A. V. (2005): Node placement for triangular mesh generation by Monte Carlo simulation. *International Journal for Numerical Methods in Engineering,* vol. 64, pp. 973–989.

Zienkiewicz, O. C.; Taylor, R. L.; Zhu, J. Z. (2008): *The Finite Element Method: Its Basis and Fundamentals*. Elsevier.