Vertex-Ball Spring Smoothing: An efficient method for unstructured dynamic hybrid meshes

T.J. Lin[a], Z.Q. Guan[a,*], J.H. Chang[a], S.H. Lo[b]

a. State Key Laboratory of Structural Analysis for Industrial Equipment, Dept. of Engineering Mechanics, Dalian University of Technology, Dalian 116024, China

b. Dept. of Civil Engineering, The University of Hong Kong, Pokfulam, Hong Kong, China

ABSTRACT：Spring analogy approach is one of the most popular dynamic mesh deformation methods. In the Ball-vertex method, vertical linear springs are introduced to deal with the element collapse problem. However, it is not very efficient as a large system of linear equations has to be resolved. In order to overcome this difficulty, the Vertex-Ball Spring Smoothing algorithm (VerBSS) is proposed in this paper. Following the mesh smoothing concept, a sub-spring system derived from the "Ball-Vertex" model is built and solved on a node by node basis using a LDL^T solver. Interior nodes are smoothed layer by layer in an iterative manner to achieve the best result. Parallel scheme is also introduced in the smoothing process for further improvement of the efficiency. Numerical examples in two and three dimensions show that VerBSS is much more efficient than the Ball-vertex method, and is capable of dealing with practical engineering objects with complex geometries subject to large deformations. VerBSS can be applied to complicated mesh topologies as well, not only to 2D/3D dynamic mesh, but also to the hybrid dynamic mesh.

Key words：Mesh deformation; dynamic mesh；ball-vertex method；Vertex-Ball spring

1. INTRODUCTION

In modern numerical simulation of engineering applications, we often need to deal with many unsteady flow problems, such as free surface flows, bio-fluid mechanics problems, forced vibration and fluid-structure interaction problems. The mesh of the solution domain needs to be updated at each time step of the numerical simulation process when flow parameters and geometries of the computational field are changing with time. The dynamic mesh approach is one of the most popular

* Correspondence to: guanzhq@dlut.edu.cn.

methods to solve this class of problems, especially for multi-block structures moving in unsteady flow with irregular boundary displacement specifications. There are generally three ways to dynamically update unstructured meshes [1,2]. (1) Remeshing: according to the changes in the domain, local or global mesh is regenerated by mesh generator, in which the flow properties of the new mesh are obtained by interpolation. However, generating new meshes will not only increase the computational cost, but will also bring additional errors [3] in the computation. (2) Mesh deformation: through nodal repositioning, the shape and the size of the elements are changed while keeping nodal connectivity intact. It is simple to implement and it does not require topological modifications. Nevertheless, it is difficult to be applied in large displacement problems. (3) Combination of the two approaches: compared to mesh deformation, remeshing is much more computationally expensive, especially for the three-dimensional applications.

Mesh deformation method is mainly divided into three categories: the PDE-based mapping, the pseudo-material and the spring analogy method. Lohner [4] and Helenbrook [5] proposed Laplacian or bi-harmonic equation methods, which are based on PDE. This kind of methods employs potential equations to determine how nodes should be repositioned. By avoiding mesh degenerating and maintaining the size of elements, they can efficiently solve the problem of intersection between mesh edges. Due to the restriction of non-coupling property between displacement variables, PDE-based method is not suitable for complex geometries and large-scale problems. It is usually applied to small-scale mesh or optimization problems. For the pseudo-material approach [6,7], the basic idea is to map the fluid domain to a pseudo-material and to calculate the mesh deformation following the classical laws of mechanics governed by a set of boundary displacement conditions. The main drawback of the pseudo-material approach is its low computational efficiency. The most widely used mesh deformation technique is the spring analogy method, which was first introduced by Batina [8], in order to deform a mesh around a pitching airfoil. By this method, a network of springs connecting all vertices (nodes) in the mesh is created, that is, each edge in the mesh is replaced by a fictitious spring whose stiffness is inversely proportional to its length. In this method, the equilibrium length of the spring is equal to the initial length of the edge. When the boundary moves, the interior nodes will be repositioned so that the spring system will converge to a new static equilibrium. Spring analogy is more often applied to aero-elastic calculations [9,10]. The method is also applied to structured meshes as well, for example, by Nakahashi

and Deiwert [11]. The system works well as long as the displacement is small compared to the element size. However, in some applications, the fluid domain boundary undergoes a motion with relatively large amplitude. It does indeed fail, when local invalid elements occur at some parts of the mesh. The reason why it fails has been studied for years. First of all, the resulting equation from the spring system is elliptic; it means that local perturbations only have local impact. Boundary perturbation cannot propagate into the interior solution domain effectively. Therefore, the spring analogy can only solve problems of small deformations. Some researchers have made improvements to the spring analogy method. By increasing the stiffness near the boundary, Blom [12] relieved the localization of deformation and enhanced the deformation capacity of the mesh. Secondly, the method lacks control mechanisms for element collapse. The spring method only considers the stretching effect of the spring, the stiffness of the edges is not related neither to the areas of the connected triangles nor to the angles of these triangles. To address this issue, torsional springs were added to the linear springs by Farhat et al. [13,14] in order to prevent the collapse of the elements. Later, Bottasso et al. [15] proposed a ball-vertex method which is arithmetically less complex than the torsional spring analogy method. The method introduces additional vertical linear springs that restrict the motion of the vertex towards its corresponding opposite face. These vertical springs effectively confine each vertex within the polyhedral ball that encloses it. The resulting linear equation is solved by a Gauss–Seidel solver. When dealing with repeated cycles of severe deformations, such as airfoil oscillation, the ball-vertex method shows a remarkably consistent behavior, which seems to be more robust than the torsional spring. The method can be implemented with only a little additional computational cost comparing with the spring analogy method. The combination of the vertical springs and the original springs can greatly improve the deformation capacity of the spring analogy method. However, it is still rather time consuming as a large system of linear equilibrium equations has to be resolved. Recently, a novel method [16,17,18] based on the creation of a background graph of the original mesh is developed. The mesh movement is carried out using the background graph with ease and efficiency. The mesh is then mapped back onto the deformed graph to provide the new mesh. Zhang [19] and Lin [20] et al. recommended improved methods for two-dimensional mesh deformation by adding new nodes into the background graph. Although the mesh update time is reduced, difficulties occur when it is extended to three-dimensional applications.

In this paper, a dynamic mesh deformation method named VerBSS (Vertex-Ball Spring Smoothing Method) is proposed. The algorithm is found to be robust for substantially distorted mesh and the solution strategy based on the LDL^T solver can significantly improve the computational efficiency.

2. THE SPRING BASED METHODS

2.1. Spring analogy method

The classical edge spring analogy method is easy to implement. A basic spring system model is shown in Figure 1. The edges of the mesh are considered as springs with stiffness inversely proportional to its length, and the spring system is in a balanced state. Given a mesh edge e_{ij}, which is connected by vertices i and j, the force on vertex i exerted by vertex j can be written as

$$\boldsymbol{f}_{ij}^{Edge} = k_{ij}(\boldsymbol{u}_j - \boldsymbol{u}_i)\cdot \boldsymbol{n}_{ij}\boldsymbol{n}_{ij} = -\boldsymbol{f}_{ji}^{Edge} \tag{1}$$

where k_{ij} is the stiffness of edge e_{ij}. $\boldsymbol{n}_{ij}$ is the unit vector from i to j. The displacement of vertex i and j are denoted by $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ respectively. Considering the effect of all its n vertices connected to vertex i, the equilibrium of node i is given by

$$\sum_{j=1}^{n} \boldsymbol{f}_{ij}^{Edge} = 0 \tag{2}$$

The elastic problem is solved for each interior node of the mesh and a large linear system $\boldsymbol{Kx}=\boldsymbol{b}$ is established. The global matrix $\boldsymbol{K}$ is formed with the edge spring stiffness k_{ij}, the vector $\boldsymbol{x}$ represents the displacements of the mesh vertices (nodes), and the vector $\boldsymbol{b}$ is formed based on the given boundary conditions.

By solving the linear system, the displacements of the interior nodes are obtained. A new mesh can be created by updating the nodal coordinates of the mesh according to the displacement vector.

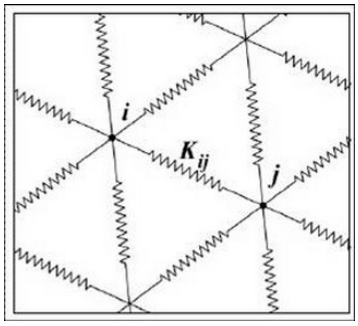

Fig.1. Spring system.

2.2. Ball-Vertex method

The disadvantage of the spring analogy method is the lack of collapse controlling mechanisms. That is, element inversions cannot be detected by the spring system. Node penetrations may occur as shown in Figure 2. This is usually the case when large amplitude of motion is involved. To prevent elements inversion, additional vertical linear springs are added in the ball-vertex method.

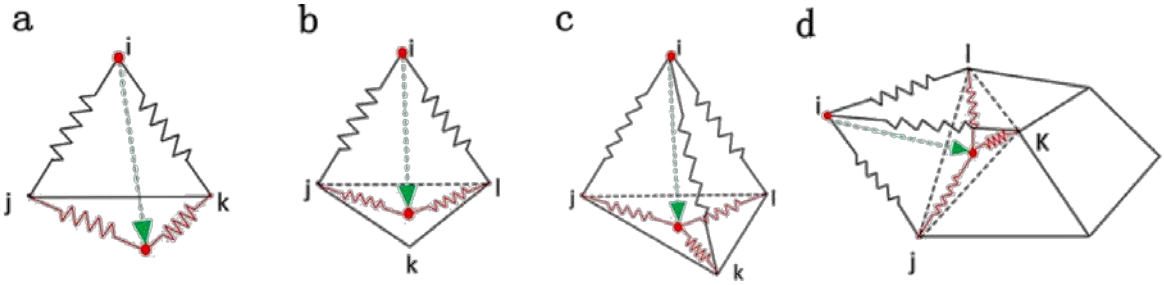

Fig.2. Collapse mechanism. (a) Tri, (b) Quad, (c) Tet, (d) Hex.

As is shown in Figure 3 [21], a convex polyhedron can always be defined around an interior node (the set of element faces that are one layer adjacent to the node). Vertical springs that connect node *i* and its projection points on the face of the ball are constructed as depicted in Figure 4. The node can be restricted by the vertical springs not to leave its ball.

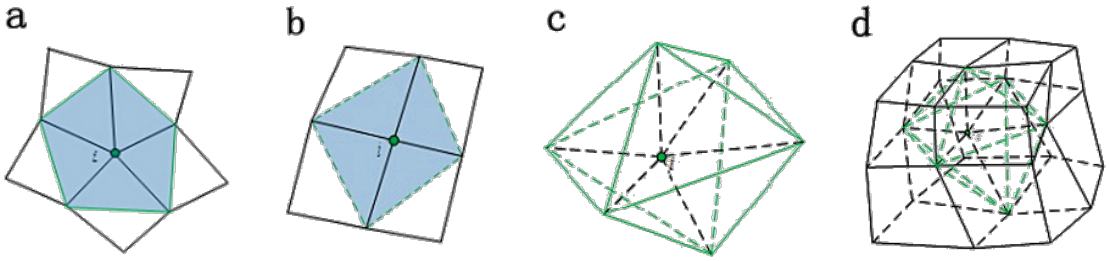

Fig.3. “Ball-vertex” concept. (a) Tri, (b) Quad, (c) Tet, (d) Hex.

For the sake of simplicity, we only take a single tetrahedron for consideration. Let T_{pijk} denote a tetrahedron whose vertices are respectively *p i j* and *k*, and *q* is the projection of vertex *p* on to its opposite face. Vertical spring S_{pq} is constructed with the stiffness inversely proportional to its length. Similarly to the previous spring equilibrium equations, the resulting force of spring S_{pq} can be expressed as

$$\boldsymbol{f}_{ip}^{ball-vertex} = k_{ip}(\boldsymbol{u}_p - \boldsymbol{u}_i).\boldsymbol{n}_{ip}\boldsymbol{n}_{ip} = -\boldsymbol{f}_{pi}^{ball-vertex} \tag{3}$$

where k_{ip} is the stiffness of the new spring, $\boldsymbol{n}_{ip}$ is the unit vector from *i* to *p*. The displacement of vertex *i* and *p* are denoted respectively by $\boldsymbol{u}_i$ and $\boldsymbol{u}_p$.

In the triangle F_{ijk}, since node p is an artificial point, its displacement is interpolated linearly with the existing vertices (i, j, k) such that

$$\boldsymbol{u}_p = \xi \boldsymbol{u}_j + \eta \boldsymbol{u}_k + (1-\xi-\eta)\boldsymbol{u}_l \tag{4}$$

where ξ and η are the area coordinates of the point p.

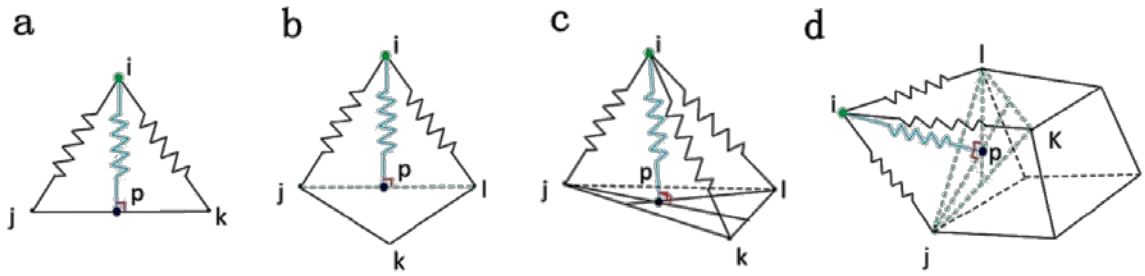

Fig.4. Ball-vertex springs: additional vertical springs. (a) Tri, (b) Quad, (c) Tet, (d) Hex.

The position of each mesh node is determined by the equilibrium under the combined action of its edge-connected springs and the additional ball-vertex springs as shown in Eq. (5)

$$\sum_{j=1}^{n} \boldsymbol{f}_{ij}^{Edge} + \sum_{p=1}^{m} \boldsymbol{f}_{ip}^{ball-vertex} = 0 \tag{5}$$

where n and m are the number of nodes and elements in the convex ball of vertex i.

The contribution of the vertical springs and the original edge springs are combined to form a new spring system. Similar to the edge spring method, a global linear equation is established for all the interior nodes. Nodal displacements are determined by solving the system of linear equations with a SOR solver [22].

In summary, among the mesh deformation methods, spring analogy method is applicable to problems with relative small movements, and the ball-vertex method introduces additional vertical springs to control on the element deformation and avoid the occurrence of element inversions.

3. THE VERBSS ALGORITHM

3.1. Basic principle

Compared to the spring analogy method, the ball-vertex method effectively constrains the interior node within its polyhedral ball. Though the scheme is able to deal with large deformations, it is rather time consuming as the solution for a large system of equilibrium equations to determine the new positions of the interior nodes

is required in each incremental step. When a large number of nodes are involved, the computational efficiency may be pretty low. As an improvement to the original ball-vertex algorithm, the VerBSS method (Vertex-Ball Spring Smoothing Method) is introduced in this paper. From the basic concept of mesh smoothing, each vertex-ball spring system is solved by means of the LDL^T solver, and the interior nodes are smoothed layer by layer in an iterative manner for a higher efficiency.

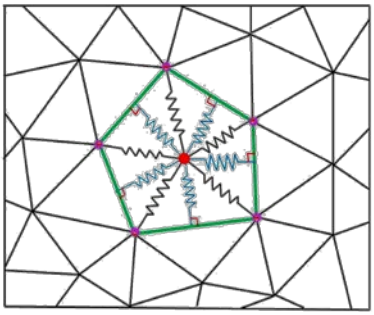

Fig.5. 2D sub-system of VerBSS algorithm.

Following the vertex-ball spring system, a sub-system for a vertex i is formed by edge springs together with the ball-vertex springs as shown in Figure 5. The global system equation $\boldsymbol{K}\,\boldsymbol{u} = \boldsymbol{b}$ is reduced to 2×2 sub-systems whose right-hand vector $\boldsymbol{b}$ is written as

$$\boldsymbol{b}_i = \sum_{j=1}^{n} k_{ij}(\boldsymbol{u}_j \cdot \boldsymbol{n}_{ij}\boldsymbol{n}_{ij}) + \sum_{p=1}^{m} k_{ip}(\boldsymbol{u}_p \cdot \boldsymbol{n}_{ip}\boldsymbol{n}_{ip}) = \mathbf{B}_i \boldsymbol{u}_i' \qquad (6)$$

where $\boldsymbol{u}_i'$ is the nodal displacement vector of the ball. n and m are the number of nodes and elements in the ball. $\boldsymbol{B}_i$ is a matrix of constant coefficients.

As displacements are solved by iteration for each node in turn, LDL^T matrix decomposition is employed for repeated calculations. Introduction of a relaxation factor can further speed up convergence, that is

$$\boldsymbol{u}^{new}{}_i = w\,\boldsymbol{u}^{new}{}_i + (1\text{-}w)\,\boldsymbol{u}^{0ld}{}_i \qquad (7)$$

Numerical examples show that w can be set between 1.5 and 1.7 (over-relaxation). There are various alternatives for the termination control of VerBSS algorithm. Since solutions for mesh deformation do not require a high accuracy, we use formula (8) as the termination criterion.

$$\sum_{i=1}^{D\cdot N} | x_i - x_i^{(prev)} | / (D \cdot N) < \varepsilon_q \qquad (8)$$

where $x_i^{(prev)}$ is the result from the previous iteration step. N is the number of interior nodes and D is the spatial dimension. That is, for a given tolerance $\varepsilon_q \in R+$, when the

mean value of the difference between two adjacent iteration results falls below a prescribed value ε_q, the mesh smoothing will terminate. However, a maximum number of iteration steps k_{max} can also be defined for practical applications.

3.2. Procedure of the algorithm

The procedure of the new mesh deformation algorithm is given as follows:

Step 1： **For(*i* = 0; *i* < Number_of_Interior_Nodes; *i* ++_)**

{ Find the polyhedral ball O_i of interior node(vertex) *i* according to the topological relations (set of triangle faces, one layer adjacent to vertex *i*); **store** the topological data of ball O_i. **}**

Step 2： **For(*i* = 0; *i* < Number_of_Interior_Nodes; *i* ++_)**

{ Construct the vertex-ball spring system of interior node(vertex) *i* so as to **form** stiffness matrix $\boldsymbol{K}_i$ and coefficient matrix $\boldsymbol{B}_i$; **decompose** matrix $\boldsymbol{K}_i$ into $\boldsymbol{L}_i$ and $\boldsymbol{D}_i$ using a $\mathrm{LDL}^{\mathrm{T}}$ solver; **store** the matrix $\boldsymbol{L}_i$, $\boldsymbol{D}_i$ and $\boldsymbol{B}_i$ for subsequent steps. **}**

Step 3： Set displacements of boundary nodes to the specified values.

Step 4： **For(*i* = 0; *i* < Number_of_Interior_Nodes; *i* ++_)**

{ Form linear equilibrium equations for interior node(vertex) *i*; **solve** the equations by $\mathrm{LDL}^{\mathrm{T}}$ matrix decomposition and **update** the displacements. **}**

Step 5： Repeat *step 4* until the result of the displacements meets the computational accuracy requirements.

Step 6： Option 1: update the coordinates of the nodes according to the displacement result.

Option 2: output the displacement result directly.

Step 7： Repeat *step2-6* (for option 1) or *step3-6* (for option 2) for the next boundary displacement increment step until the simulation process is completed.

3.3. A simple parallel scheme

Parallel technology is popular as today's microcomputers are all equipped with more than one processor. An efficient parallel algorithm making the full use of all the processors will boost the efficiency of computers. In mesh deformation applications, the CUP time required is increased as the problem size becomes larger. Generally, the construction of parallel solution for large system of linear equations may not be a simple task.

In this paper, based on the VerBSS algorithm framework, a generic parallel scheme can be easily constructed according to the characteristic of the sub-spring

system of VerBSS. Each interior node is a new system and multiple sub-systems can be solved simultaneously in parallel by several processors using the LDL^T solver. The parallel method is used in section 5.2 and 5.3 with 3D large-scale problems. Results show that the efficiency is much improved compared to the classical sequential process. Furthermore, there is no additional memory requirement for the parallel process.

On OpenMP platform, the simple parallel scheme can be described as follows:

…
{*Step 3*}
 # pragma omp parallel for NP(the Number of Processors)
{*Step 4*}
…

4. TWO-DIMENSIONAL APPLICATIONS

In this section, the performance of the proposed VerBSS algorithm will be compared to the classical spring analogy method and the ball-vertex method for two-dimensional dynamic meshes. For this purpose, applications of translational rectangle, pitching and plunging aerofoil and multi-element aerofoil are considered. The numerical simulations were all carried out on a 2.66MHz Duo Core computer.

Mesh quality measures expressed in terms of geometric criteria are used for assessing the characteristics of these methods. In this paper, we consider quality measures based on the ratio of the radii of the inscribed and circumscribed circles (or spheres in three dimensions) [23], which gives a measure of the stretching of an element. It is evaluated as

$$\rho = C \cdot r / R \qquad (0 < \rho \leq 1) \tag{9}$$

where r and R are respectively the radii of the inscribed circle and the circumscribed circle. C equals 2 and 3 for 2D and 3D applications respectively, and it is noted that for equilateral triangles and tetrahedra, the value of ρ equals to 1.

4.1. Translational motion of the rectangle

A model of translational rectangle with 496 triangles and 298 vertices is studied. Large vertical displacements are applied to the rectangle inside while the outer geometries remains fixed. The initial mesh configuration is shown in Figure 6(a). The tolerance adopted for the SOR algorithm in the test is 10^{-6} and the maximum iterative

number for SOR is limited to 300.

Apart from efficiency, the robustness of the scheme is also an important consideration. In Figure 6(b), we observe that invalid triangles are formed when the vertical displacement reaches 2.6 units for the spring analogy method. On the other hand, the rectangle can move to a maximum displacement of 5.0 units for both the ball-vertex and the VerBSS algorithms (Figure6(c)).

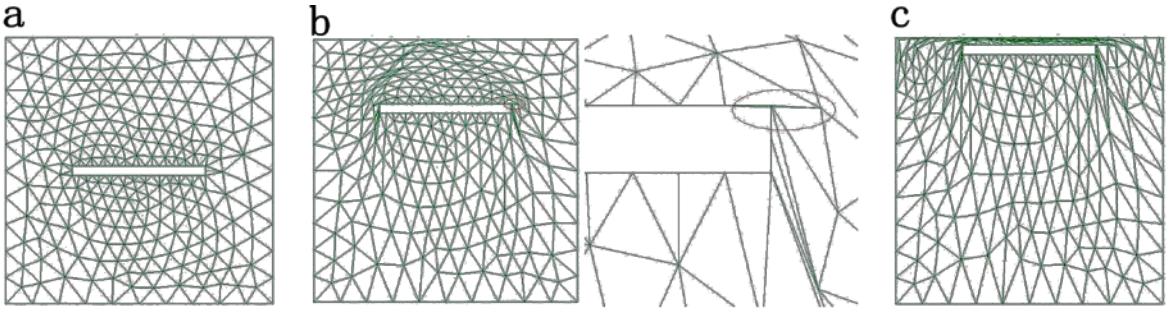

Fig.6. Dynamic meshes of translational rectangle. (a) Initial configuration, (b) Spring analogy method: failed, (c) VerBSS algorithm.

In fact, the proposed algorithm is equivalent to the Block-SOR iterative method in principle. As can be seen from table 1, the VerBSS algorithm and the ball-vertex method converge to the same result. These two methods are capable of yielding a valid mesh, even when the rectangle is very close to the domain boundary, and they both have strong ability of dealing with large boundary deformations. However, the required CPU time of the ball-vertex method is 20ms for a single time step, whereas the proposed algorithm takes only 0.5ms, and VerBSS is much faster than the ball-vertex method for this example.

Table 1.
Quality distribution of triangular elements (a movement of 4.0 units).

Method / Quality	ball-vertex method	VerBSS algorithm
0.0-0.02	6	6
0.02-0.1	35	35
0.1-0.4	81	81
0.4-0.7	126	126
0.7-1.0	248	248

Next, we give a further investigation on the efficiency of the scheme, using exactly the same model with much finer mesh, which consists of 3055 elements and

5830 vertices as shown in Figure 7. In this refined model, the computational time of the ball-vertex method needed 3.714s and the VerBSS algorithm took 0.021s.

The comparison of the CPU time of the two methods for different mesh density is shown in Table 2, and the corresponding graph is shown in Figure 8. From the results, we can see that efficiency is further improved for more complicated large-scale meshes.

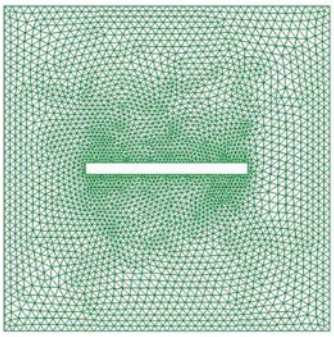

Fig.7 Model with finer mesh.

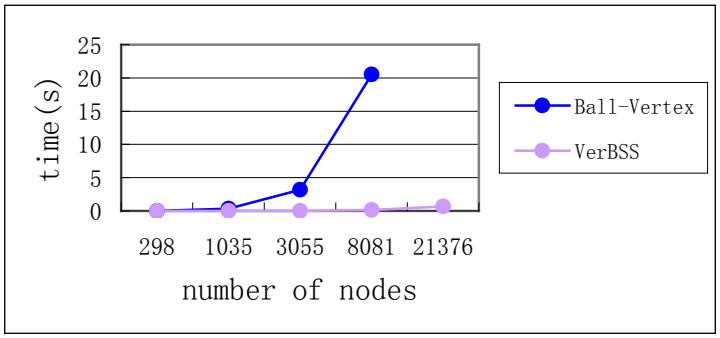

Fig.8. CPU time for meshes of different density.

Table 2.
CPU time required by the two methods in a single time step.

tim e / numbers	Ball-vertex (s)	VerBSS (s)
298	0.020	0.0005
1035	0.349	0.004
3055	3.174	0.021
8081	20.532	0.165
21376	---	0.685

4.2. Pitching and plunging airfoil

In this example, the aerodynamic benchmark problem of a standard wing, the NACA0012 airfoil [24] is chosen as a test case. The pitching and plunging airfoil of a unit chord moves according to Eq. (10) with pitching amplitude θ_0=40°and plunging amplitude h_0 =0.2 chord. The flow domain is discretized into 8664 triangles (4485 nodes).

$$\theta_n = \theta_0 \sin(2\pi n/N)$$

$$h_n = h_0 \sin(2\pi n/N) \qquad (10)$$

N=160 stands for the total number of incremental time steps. The maximum number of SOR iterations is limited to 300 and the SOR convergence criterion chosen is $\varepsilon_q=10^{-5}$ .The performance of the proposed method is compared with the spring analogy method and the ball-vertex method with regard to robustness, efficiency and mesh quality. For the spring analogy, the elements near the trailing edge is distorted when the airfoil rotates to 18°, whereas full required displacement is achieved within the allowable number of displacement increments by the ball-vertex and VerBSS algorithms. The mesh quality has also been well maintained at each stage of the movement as shown in Tables 4 and 5. Figure 9 shows the upper and lower mesh configurations at the end of the first cycle by the VerBSS algorithm.

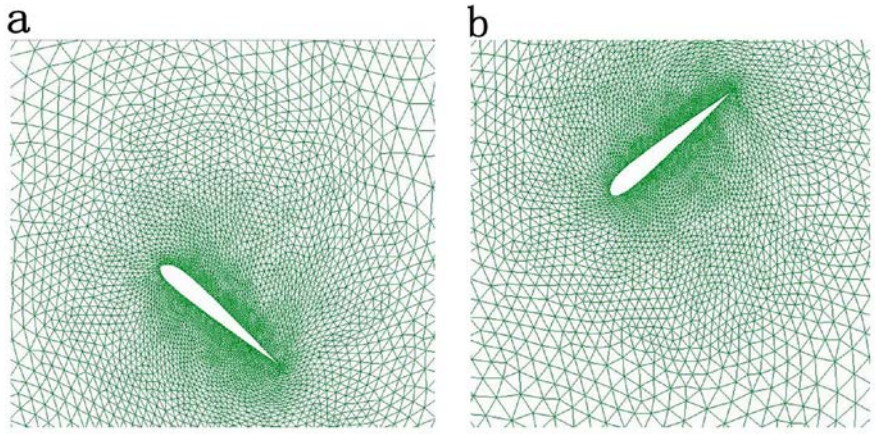

Fig.9. Pitching and plunging airfoil with upper and lower configurations.

For efficiency comparison, as the solution for a large system of equations consumes much more time, the computational time of spring analogy is similar to that of ball-vertex method with 9.11s and 9.328s for a single time step respectively, which is much more than that of the VerBSS algorithm. The average number of iterations is 65 for the VerBSS algorithm, and the CPU time is only 0.11s for a single time step.

Table 3.
CPU time required for a single mesh update step.

VerBSS algorithm	Time
Part Ⅰ: Formation of equations for each vertex	0.031s
Part Ⅱ: Solution for displacements	0.076s
Part Ⅲ: Mesh update time	0.003s
Total time	0.110s

Quality of the meshes is presented in the following tables for rotation angle of 15° and 30°, which verify the consistent convergence property of the two methods. The

local mesh quality around the airfoil is also maintained after large deformations. A rotation angle as large as 96° can be achieved by VerBSS as shown in Figure 10.

Table 4.
Quality comparison with a rotation angle of 15°.

method / quality	spring analogy	ball-vertex	VerBSS
0.0-0.02	2	0	0
0.02-0.1	3	0	0
0.1-0.4	28	6	6
0.4-0.7	158	120	120
0.7-1.0	8473	8538	8538

Table 5
Quality comparison with a rotation angle of 30°.

method / quality	Ball-vertex	VerBSS
0.0-0.02	0	0
0.02-0.1	1	1
0.1-0.4	12	12
0.4-0.7	206	206
0.7-1.0	8445	8445

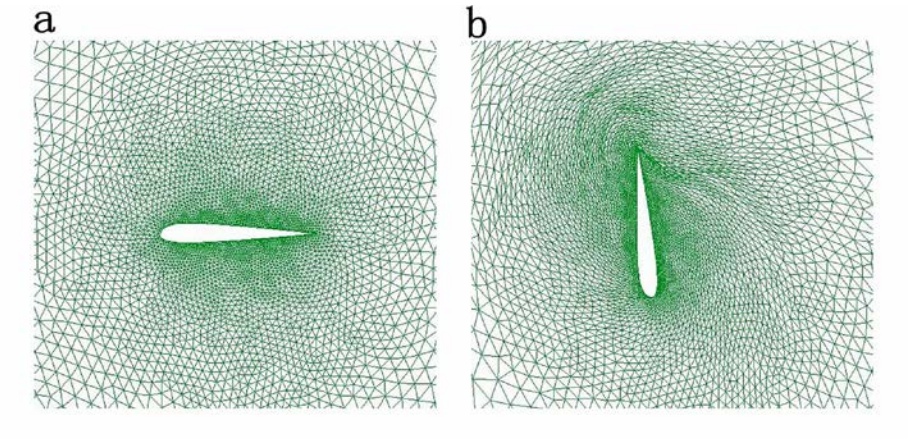

Fig.10. Original configuration and airfoil rotates to 96°.

4.3 Multi-element aerofoil with large dynamic flap deployment

A three-element aerofoil is studied in the last test. The aerofoil is composed of a leading edge slat, the main airfoil section and a trailing edge flap that has been used as

an effective high-lift device. Both the front and back flaps move towards the main configuration. The fluid mesh has a total of 11372 mesh vertices (nodes) and 22177 elements. The SOR convergence criterion is the same as the previous section. The average number of iterations is 80 by the VerBSS algorithm, and the required CPU time for the formation of equations and solution of the aerodynamic problem by the LDL^T solver is given in Table 6.

Table 6.
CPU time required for a single mesh update step.

VerBSS algorithm	Time
Part Ⅰ: Formation of equations for each vertex	0.094s
Part Ⅱ: Solution for displacements	0.196s
Part Ⅲ: Mesh update time	0.018s
Total time	0.308s

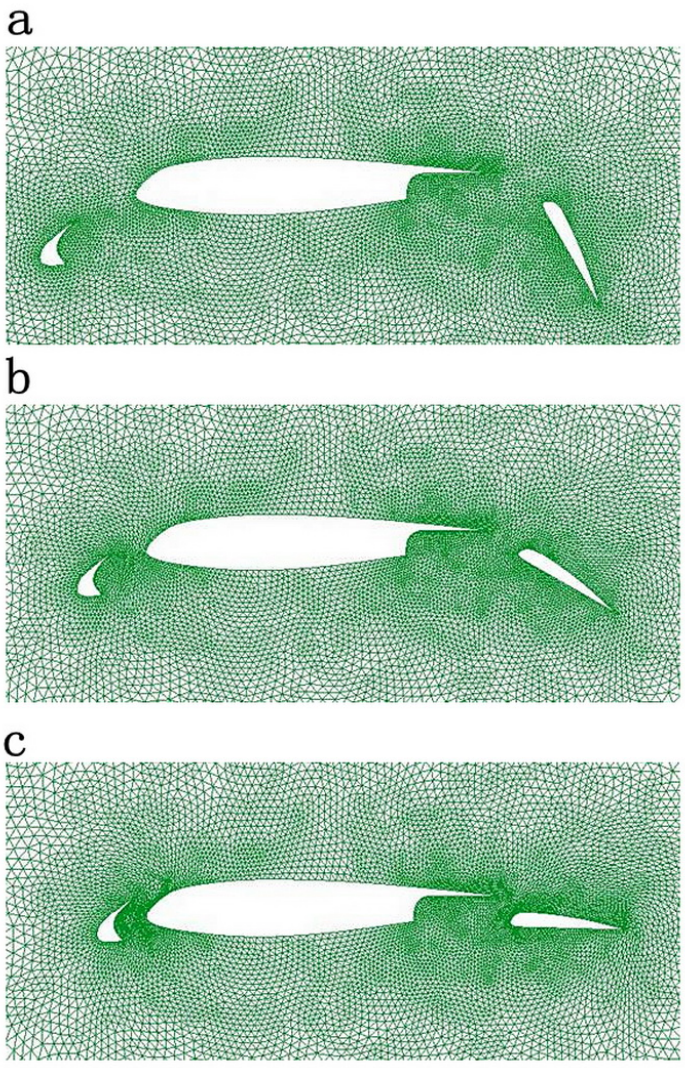

Fig.11. Mesh deformation of multi-element aerofoil.

A sequence of the moving mesh for a relatively large displacement using the approach described in Section 3 is depicted in Figure 11. Results show that the VerBSS algorithm manages to update the mesh around the wing's flaps. Although the mesh is subjected to a strong compression in the slot areas, the entire process is well completed without any node penetration or element inversion.

5. THREE-DIMENSIONAL APPLICATIONS

The commonly used spring analogy method, classical Laplacian equation method [4] and the ball-vertex method are chosen to compare with the VerBSS algorithm for three-dimensional applications. The tolerance adopted for Section 5.1 is 10^{-5}, and 10^{-4} for Section 5.2 and 5.3. In order to demonstrate the improvement by the parallel scheme, the last two cases are carried out on a 2.20MHz four cores computer, and only the CPU time for the solution part is considered for comparison in this section.

5.1 Movement of a sphere in a fixed cube

We consider the dynamic mesh for a sphere of radius 5 moving from the center to the left-hand side within a 50×50×50 cube. The mesh consists of 2058 nodes and 9496 elements. The original configuration is shown in Figure 12(a).

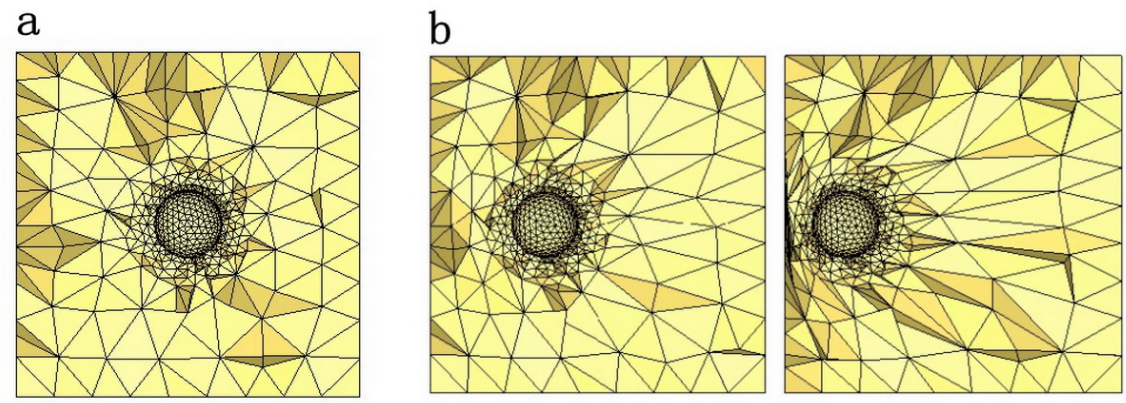

Fig.12. Moving meshes around a sphere in the cube. (a) Initial configuration, (b) Moved mesh by VerBSS algorithm.

The maximum displacements of the sphere are both approximately 10 units for the spring analogy method and the classical Laplacian equation method; whereas the ball-vertex and the VerBSS algorithms can move the sphere to a total displacement of 16 units. The deformed mesh obtained by the VerBSS algorithm is presented in Figure 12(b). The number of iteration is 60 for the VerBSS algorithm, and the

required CPU time for one single time step is shown in table 7.

Table 7.
CPU time comparison between different methods in one step .

Spring analogy [8]	1.683s
Ball-vertex method [15]	1.954s
Classical Laplacian equation [4]	0.621s
VerBSS algorithm	0.067s

From this example, we can see that the VerBSS algorithm and the ball-vertex method are superior to the spring analogy and the classical Laplacian equation method in terms of mesh deformations. As for efficiency, the methods based on the spring concept need to solve a large system of linear equations, which leads to a low computational efficiency. Since the displacements in three directions, x,y,z are independently solved out in the Laplacian method, the scale of the problem is reduced. On the other hand, by considering a sub-system for each individual node to avoid the resolution of a global system of equilibrium equations, the proposed VerBSS algorithm tremendously improves the computational efficiency.

5.2 Dynamic mesh around swimming fish

In the field of bionics, we use the undulatory flexible body as a simplified model to study the relevant mechanisms of swimming animals. In this example, model of swimming fish is carried out to investigate the viscous flow around a flexible body. A mesh of two fish swimming side by side is generated with 20840 nodes and 108322 tetrahedra. The geometric model consists of two fish with anti-phase undulation, of which the spacing is 0.5 length of the body. The Initial configuration is shown in Figure 13.

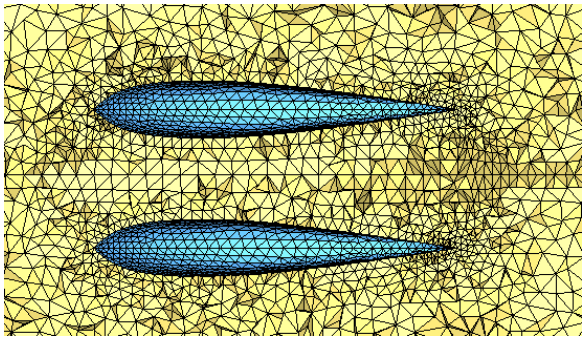

Fig. 13. Initial configuration

Movement of fishes is as follows:

$$y(x,t) = A(x)\cos[2\pi/\lambda\,(x-ct)] \tag{11}$$

where x is the distance from a point on the body to the head of the fish. λ=1.0 is the wavelength of the swing and the phase speed c is taken as 1.5 in this example. $A(x)$ is the function which is formed based on the amplitude of the movement at different parts of the fish. Figure 14 shows four typical moments of the swimming fish in one cycle. Each deformation cycle is broken into a sequence of 60 steps. The CPU time for a single time-step movement by the serial scheme is 0.205s and the whole process takes 14.409s. Parallel technology is used in this example to further improve the computational efficiency. The parallel computing time is shown in table 8.

$$Speed\,up = \frac{CPU\,time\,of\,one\,processor}{CPU\,time\,of\,multiple\,processors}$$

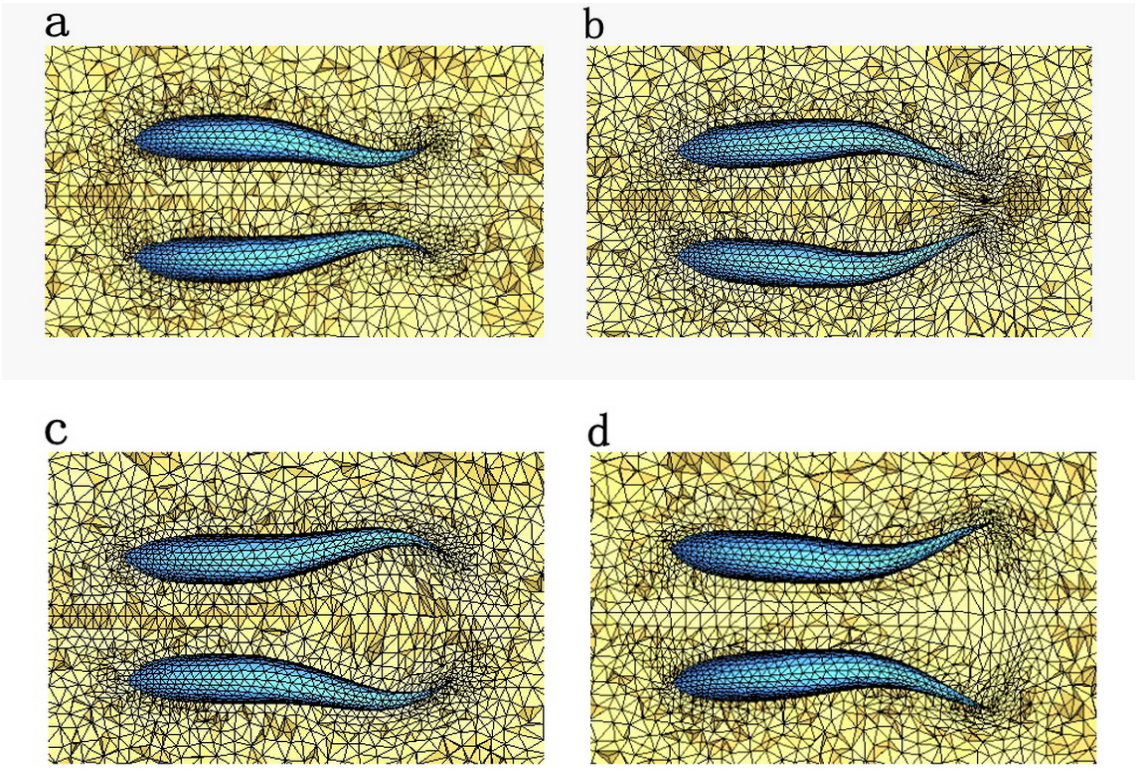

Fig.14. Four typical moments of the swimming fishes.

Table 8.

CPU time for one step with different number of processors, NP= number of processors.

NP	CPU time (s)	Speed up
1	0.205	1.000
2	0.112	1.830

4	0.076	2.697

5.3 Deformation of a bending wing

For the last case, the geometry of a wing-body combination is considered for aero-elastic studies, which is discretized into 106755 nodes and 610252 tetrahedra. A view of the initial mesh is given in Fig.15(a). We assume the wing is clamped at its root and bend with a tip bending amplitude of 0.4*L*, where *L* is the span of the wing. The bending angle of the tip is 40° approximately. The deformed configuration is achieved in 40 steps, i.e. about 1° per step. Figure 15(b) shows the wing at the upstroke positions.

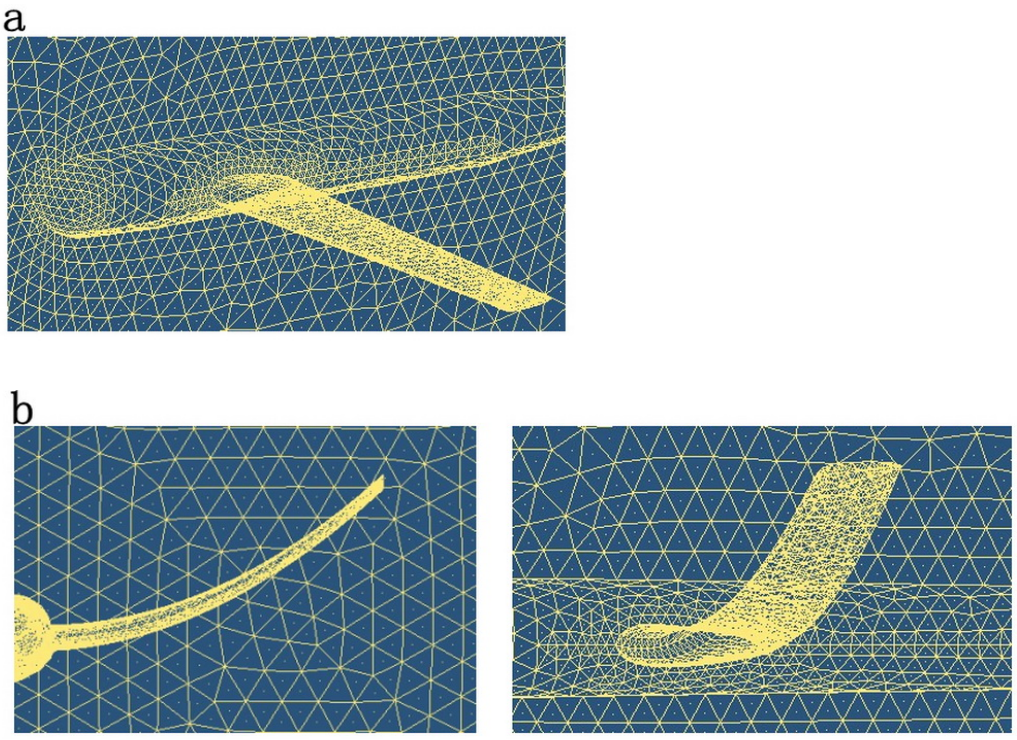

Fig.15. The bending wing with deformation of 40°. (a) Initial configuration, (b) Mesh after the deformation.

Table 9.

CPU time for one step with different number of processors, NP= number of processors.

NP	CPU time (s)	Speed up
1	1.198	1.000
2	0.743	1.612
4	0.463	2.587

In terms of efficiency, The CPU time by the serial process was 1.198s in a single time-step movement and the entire process took 55.236s. The corresponding parallel CPU time is shown in table 9.

6 ANALYSIS OF VERBSS ALGORITHM

In terms of mathematical modeling, the three methods, namely, the spring analogy, ball-vertex method and the VerBSS method, are all based on a system of *D*N* linear equations, where *N* is the number of interior nodes and *D* represents the spatial dimensions. In the solution phase, the first two methods use the same classic SOR iterative procedure, and the VerBSS algorithm uses a different solution strategy which is similar to the Bolck-SOR iterative method. By solving the vertex-ball spring system of each node in turn, the three displacement components of each node are obtained by means of LDL^T matrix decomposition, which is equivalent to reducing the *D*N* linear equations into *N* sub-blocks of much smaller sizes. As the vertex-ball spring system is a linear elastic structure, the stiffness matrix of the sub-block system is symmetric positive definite. These characteristics enable VerBSS algorithm to converge faster with the highest computation efficiency among the three methods. Moreover, the Block-SOR and SOR algorithms will converge to the same solution as solution is unique for a linear elastic system irrespective how the solution is determined. However, the order in processing the nodes has an impact on the CPU time and the symmetric SOR method (SSOR) can improve the computational efficiency in some situations. Furthermore, only non-zero components of large sparse stiffness matrix $\boldsymbol{K}$ are stored instead of the whole matrix in the VerBSS algorithm to save memory space.

7. CONCLUSIONS

As an improvement over spring analogy method in terms of robustness and efficiency, a dynamic mesh deformation approach named VerBSS is proposed. From the basic mesh smoothing concept, the sub-spring system derived from the "Ball-Vertex" model is solved by means of a LDL^T solver. Interior nodes are smoothed layer by layer in an iterative manner. In principle, this procedure bears the same concept as the Block-SOR iterative method. The algorithm simplifies the global spring system to sub-systems of individual interior nodes. Because of the

characteristic of the sub-block, a parallel scheme can be easily implemented. As a result, the efficiency is significantly enhanced, especially in three dimensional large-scale meshes. Furthermore, memory required is greatly reduced, as only the non-zero coefficients of the sparse matrix are stored. Numerical examples in two and three dimensions have shown that the VerBSS algorithm is simple to implement and is capable of dealing with practical engineering problems of complex geometries subject to large boundary deformations. In comparison with other dynamic mesh deformation methods, the new method exhibits both robustness and computational efficiency.

ACKNOWLEDGEMENT

The financial support from HKSAR GRF Grant to the research project HKU715110E on "Drifted based seismic fragility analysis of high-rise RC building with transfer structures" is greatly acknowledged. The authors would also like to appreciate the joint supports to this project by the National Natural Science Foundation of China (Grant No. 10872040 and 11272074).

Appendix: The LDLT approach

The VerBSS algorithm only needs solving the sub-spring system. It converts large linear equations into a 2×2 (three-dimensional 3×3) symmetric linear equations. The corresponding fictitious stiffness matrix is symmetric and positive definite.

In certain applications, the coefficient matrix of the stiffness is constant, and only the right-hand vector changes from time to time. To deal with this situation, a great deal of calculations can be saved if matrix decomposition is done only once for different right-hand side vectors. One of the most effective methods is the LDL^T decomposition approach, especially, if the coefficient matrix is symmetric.

The analysis of computational cost is as follows:

(a) The analytical method

$$\begin{cases} k_{11}x_1 + k_{12}x_2 + k_{13}x_3 = b_1 \\ k_{21}x_1 + k_{22}x_2 + k_{23}x_3 = b_2 \\ k_{31}x_1 + k_{32}x_2 + k_{33}x_3 = b_3 \end{cases} \tag{12}$$

Eq. (12) shows the equation of three dimensional applications whose solution is evaluated as

$$x_1 = D_1 / D \ , \ x_2 = D_2 / D \ , \ x_3 = D_3 / D.$$

where

$$D = \begin{vmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{vmatrix} \quad D_1 = \begin{vmatrix} b_1 & k_{12} & k_{13} \\ b_2 & k_{22} & k_{23} \\ b_3 & k_{32} & k_{33} \end{vmatrix} \quad D_2 = \begin{vmatrix} k_{11} & b_1 & k_{13} \\ k_{21} & b_2 & k_{23} \\ k_{31} & b_3 & k_{33} \end{vmatrix} \quad D_3 = \begin{vmatrix} k_{11} & k_{12} & b_1 \\ k_{21} & k_{22} & b_2 \\ k_{31} & k_{32} & b_3 \end{vmatrix}$$

Each determinant needs 12 multiplications to evaluate. While determinant *D* keeps unchanged, *D1*, *D2*, *D3* change with vector ***b***, at least 36 multiplications and 3 divisions are needed for each iteration.

(b) The LDL^T method

The LDL^T decomposition of matrix ***K*** to solve ***Kx***=***b*** is referred to as the LDL^T decomposition approach, in which

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \cdots & \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1 \end{pmatrix}, \quad D = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix},$$

Since the coefficient matrix ***K*** will not be used again after decomposition, memory can be saved by putting the reduced coefficients in their original position,

Solving symmetric linear equation ***Kx***=***b*** according to the decomposition ***K***=***LDL***$^{\boldsymbol{T}}$, we have (***LDL***$^{\boldsymbol{T}}$) ***x***=***b***, which can be broken into three steps, i.e. ***Ly***=***b***，***Dz***=***y*** and ***L***$^{\boldsymbol{T}}$***x***=***z***. Thus the solution are obtained as

$$\begin{cases} y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k, & i = 1, 2, \cdots, n \\ z_i = y_i / d_i, & i = n, n-1, \cdots, 1 \\ x_i = z_i - \sum_{k=i+1}^{n} l_{ki} x_k, & i = n, n-1, \cdots, 1 \end{cases} \qquad (13)$$

As seen from equation (13), each iteration requires 6 multiplications and 3 divisions, when *n* equals to 3. If the inverse matrix of ***D*** is also stored, there needs only 9 multiplications

REFERENCES

[1] B.M. Klingner, B.E. Feldman, N. Chentanez, J.F. O'Brien, Fluid animation with dynamic meshes, ACM. T. GRAPHIC. (SIGGRAPH Proceedings). 25 (2006) 820-825.

[2] D.M. Huang, S.G. Dong, D.H. Chen, 2D unstructured dynamic mesh research on store separation, Acta Aerodyn. Sin. 24 (1) (2006) 73-79.

[3] Z. Guo, J. Liu, Dynamic unstructured grid method with applications to 3D unsteady flows involving moving boundaries, Acta Mech. Sin. 35 (2) (2003) 141-146.

[4] R. Löhner, C.Yang, Improved ALE mesh velocities for moving boundaries, Commun. Numer. Methods Eng. 12 (1996) 599-608.

[5] B.T. Helenbrook, Mesh deformation using the biharmonic operator, Int. J. Numer. Methods Eng. 56 (2003) 1007-1021.

[6] E.J. L´opez, N.M. Nigro, M.A. Storti, J.A. Toth, A minimal element distortion strategy for computational mesh dynamics, Int. J. Numer. Methods Eng. 69 (2007) 1898-1929.

[7] S.Y. Hsu, C.L. Chang, Mesh deformation based on fully stressed design: The method and 2-D examples, Int. J. Numer. Methods Eng. 72 (2007) 606-629.

[8] J.T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA J. 28 (8) (1990) 1381-1388.

[9] W.L. Kleb, J.T. Batina, M.H. Williams, Temporal adaptive Euler/Navier - Stokes algorithm involving unstructured dynamic meshes, AIAA J. 30 (8) (1992) 1980-1985.

[10] K.P. Singh, J.C. Newman JC, O. Baysal, Dynamic unstructured method for flows past multiple objects in relative motion, AIAA J. 33 (4) (1995) 641-659.

[11] K. Nakahashi, G.S. Deiwert, Three-dimensional adaptive grid method, AIAA J. 24 (6) (1986) 948-954.

[12] F.J. Blom, Considerations on the spring analogy, Int. J. Numer. Methods Fluids 32 (2000) 647-668.

[13] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, Torsional springs for two-dimensional dynamic unstructured fluid meshes, Comput. Methods Appl. Mech. Eng. 163 (1998) 231-245.

[14] C. Degand, C. Farhat, A three-dimensional torsional spring analogy method for unstructured dynamic meshes, Comput. Struct. 80 (2002) 305-316.

[15] C.L. Bottasso, D. Detomi, R. Serra, The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes, Comput. Methods Appl.

Mech. Eng. 194 (2005) 4244-4264.
[16] E. Lefrancois, A simple mesh deformation technique for fluid-structure interaction based on a submesh approach, Int. J. Numer. Methods Eng. 75 (2008) 1085-1101.
[17] X.Q. Liu, N. Qin, H. Xia. Fast dynamic grid deformation based on Delaunay graph mapping, J. Comput. Phys. 211 (2006) 405-423.
[18] J.H. Ko, S.H. Park, H.C. Park, D. Byun, Finite macro-element-based volume grid deformation for large moving boundary problems, Int. J. Numer. Methods Biomed Eng. 26 (2010) 1656-1673.
[19] L.P. Zhang, X.P. Duan, X.H. Chang, Z.Y. Wang, H.X. Zhang, A hybrid dynamic grid generation technique for morphing bodies based on Delaunay graph and local remeshing, Acta Aerodyn. Sin. 27 (1) (2009) 32-40.
[20] T.J. Lin, Z.Q. Guan, Fast dynamic mesh moving based on background grid morphing, Chin. J. Comput. Mech. 29 (1) (2012) 105-110.
[21] H. Wang, G.H. Xu, Grid deforming method for flow field simulation of elastic helicopter rotors, J. NanJing Univ. Aeronaut. Astronaut. 43 (1) (2011) 1-6.
[22] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., PWS Publishing Company, Boston, 2003.
[23] V.N. Parthasarathy, C.M. Graichen, A.F. Hathaway, A comparison of tetrahedron quality measures, Finite Elem. Anal. Des. 15 (1993) 255-261.
[24] H. Shan, L. Jiang, C.Q. Liu, Direct numerical simulation of flow separation around a NACA 0012 airfoil, Comput. Fluids. 34 (2005) 1096-1114.