Efficient and Robust Parallel Mesh Motion Solver Using Radial Basis Functions

Xiang Gao[1]; Chuanfu Xu[2]; Yidao Dong[3]; Min Xiong[4]; Dali Li[5]; Zhenghua Wang[6]; and Xiaogang Deng[7]

Abstract: This paper presents a parallel mesh deformation solver using radial basis function (RBF) interpolation. The solver computes the displacement of each internal point independently without using the topological relations, and is further accelerated by an incremental approach based on the data reduction algorithm. The incremental approach makes full use of the matrix and solution of the previous step during the greedy selection procedure, and gives a better initial solution of the current RBF system of equations. To enhance the robustness and efficiency of the solver in parallel, for nonpredefined boundary movement, each CPU process computes the same interpolation function; for predefined movement, an additional process can be used to calculate the interpolation function one step earlier and broadcast it to other processes. Four typical mesh motion cases are simulated to demonstrate the deforming capability and parallel performance of the proposed method. Finally, several parametric setting rules of the deformation approach are presented for better usage. **DOI: 10.1061/(ASCE)AS.1943-5525.0000874.** *© 2018 American Society of Civil Engineers.*

Introduction

Unsteady computational fluid dynamics (CFD) applications such as aeroelastic computation, aerodynamic shape optimization, and fluid–structure interaction usually involve moving boundaries. In order to be able to perform the flow calculations accurately and reliably, a robust, accurate, and efficient method which redistributes the mesh in accordance with the movement of the domain is needed. One natural idea is to regenerate the mesh at every time step, but mesh generation is very time-consuming for complex geometry problems. Moreover, after the mesh is regenerated, the solution at the new mesh must be mapped with an interpolation step. The interpolation step can produce additional errors and decrease the accuracy of the flow solver. The other popular method uses the so-called mesh deformation approach, which just moves the position of grid points without changing their topological relations.

Among all the possible strategies available to deform a mesh during a simulation, finding a general technique suitable for all types of meshes and physical situations would be preferable. The transfinite interpolation method interpolates the displacements of boundary points into inner points along with the grid lines, which is fast and accurate but is only applicable to a structured mesh (Wang and Przekwas 1994). Two types of strategies can be used independent of the flow solver. The first uses the connectivity information of the grid points, by analogizing the edges as springs (Batina 1991) or solid body elasticity (Lynch and O'Neill 1980). Computational costs for these methods are huge because a system of equations involving all mesh points must be solved. The other point-by-point (connectivity-free) strategy moves each grid point independently based on its coordinates and distance to the boundary, and is easily implemented in parallel. In recent years, some point-by-point interpolation methods have been applied for mesh deformations. The inverse distance weighting (IDW) interpolation method computes the displacements of interior points explicitly, but needs different power parameters for different kinds of movement (Witteveen and Bijl 2009; Witteveen 2010; Luke et al. 2012). The machine learning–based methods using artificial neural networks (Stadler et al. 2011) or support vector machines (Gao et al. 2017) first train a system for each coordinate direction via the displacements of the boundary points, then predict the displacement of every internal point individually. This kind of method is efficient but the deforming capability is relatively small. The radial basis function (RBF) interpolation method is more flexible and robust for addressing large deformations (De Boer et al. 2007; Rendall and Allen 2008), and is widely applied in a variety of research fields (Biancolini et al. 2014; Li et al. 2017).

However, although the RBF approach is particularly attractive to obtain a high-quality mesh motion, the motion requires the solution of a system of equations whose size equals the number of boundary points. It is still quite expensive for large-scale simulations.

[1]Ph.D. Candidate, College of Computer, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China. E-mail: gaoxiang12@nudt.edu.cn

[2]Associate Professor, College of Computer, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China (corresponding author). E-mail: xuchuanfu@nudt.edu.cn

[3]Ph.D. Candidate, College of Aerospace Science and Engineering, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China. E-mail: dongyidao08@nudt.edu.cn

[4]Ph.D. Candidate, College of Computer, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China. E-mail: xiongmin@nudt.edu.cn

[5]Assistant Professor, College of Aerospace Science and Engineering, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China. E-mail: achhx@nudt.edu.cn

[6]Professor, College of Computer, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China; Professor, State Key Laboratory of Aerodynamics, China Aerodynamics Research and Development Center, P.O. Box 211, Mianyang 621000, P.R. China. E-mail: zhhwang188@sina.com

[7]Professor, College of Aerospace Science and Engineering, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China; Professor, College of Computer, National Univ. of Defense Technology, No. 109 Deya Rd., Changsha 410073, P.R. China. E-mail: xgdeng2000@vip.sina.com

Note. This manuscript was submitted on October 6, 2017; approved on February 2, 2018; published online on March 8, 2018. Discussion period open until August 8, 2018; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Aerospace Engineering***, © ASCE, ISSN 0893-1321.

To alleviate these difficulties, a data-reduction algorithm based on minimizing an error function was proposed to reduce the selection of boundary points (called control points) (Rendall and Allen 2009; Rendall and Allen 2010; Michler 2011; Sheng and Allen 2013), which has been shown to decrease significantly the cost of the mesh motion in terms of both time and memory requirements. Furthermore, Gillebaart et al. (2016) proposed an adaptive RBF mesh deformation method, which ensures the set of control points always represents the geometry up to a certain criterion by keeping track of the boundary error throughout the simulation and reselecting when needed. Niu et al. (2017) used a dynamic set of control points in each time step of mesh deformation. The neighboring boundary point near the cell with the worst quality is added into the control point set, whereas the neighboring control point near the cell with the best quality is removed from the control point set. This permits larger mesh deformation with a small increase in computational cost. Kedward et al. (2017) presented a reformative method called multiscale RBF interpolation. This method enlarges the control points from a base subset to all the surface points with the use of different support radii. As a result, the method modifies the distance matrix and makes it much easier to solve the equation. Another way to reduce the time cost is to apply a hybrid method which combines the moving submesh approach (MSA) and RBF interpolation (Liu et al. 2012). It uses the RBF method to deform the background mesh. For global support kernel functions, which lead to a dense matrix system, (Coulier and Darve 2016) proposed the inverse fast multipole (IFMM) method to exploit the hierarchical structure of the matrix in order to convert the original dense system into an extended sparse system, and then solve the linear system in a more efficient way. The IFMM algorithm is suitable for large-scale problems with complicated deformation, but the code implementation is not easy and its parallelization issue has not been addressed yet. PetRBF (Yokota et al. 2010) is a highly parallel RBF interpolation algorithm. It uses the fast decay property of the Gaussian kernel function that allows rapid convergence of the iterative solver. However, De Boer et al. (2007) proved that the Gaussian function is not good enough for mesh deformation. Estruch et al. (2013) presented a parallel algorithm based on the pure form of the RBF interpolation. It computes different interpolating function in each process. This may cause mesh cells intersection near shared boundaries and is difficult to avoid, so the robustness of the method needs to be improved.

This paper implements a robust and efficient parallel mesh motion solver using RBF interpolation based on the framework of *OpenFOAM* (*Open Field Operation And Manipulation*) (Jasak et al. 2007; Bos et al. 2013). In the aspect of parallelism, for non-predefined boundary movements such as aerodynamic shape optimization, each process maintains a copy of the entire boundary points and forms the interpolation function simultaneously. For predefined boundary movements such as forced airfoil pitching, only an additional process is needed to compute the interpolation function one step earlier than CFD calculation. As a result, this hides the time cost of interpolation function computing. The proposed motion solver uses the error-based greedy method to reduce the boundary point selection, and an incremental iterative linear solver that fully uses the previous solution is proposed to further accelerate every selection step. A detailed analysis of the influence of each calculation module and each parameter is presented, and several universal parametric setting guidances are presented. Four test cases as representatives of dynamic mesh applications are used to demonstrate the superiority of this method. Because of the hierarchical design and encapsulation of the *OpenFOAM* platform, the proposed parallel motion solver can be directly coupled with many flow solvers integrated in *OpenFOAM*.

Methodology

Radial Basis Function Interpolation

In recent years, the RBF approximation has become increasingly popular as a method to interpolate scattered data. For mesh deformation, the interpolation function of the RBF method, describing the displacement of the entire computational domain in each coordinate direction, usually takes the form

$$g^{(k)}(\mathbf{x}) = \sum_{i=1}^{N_b} \alpha_i^{(k)} \phi(\|\mathbf{x}-\mathbf{x}_i\|) + p(\mathbf{x}), \quad k \in \{x, y, z\} \tag{1}$$

where $g(\mathbf{x})$ = function to be evaluated at location $\mathbf{x}$ and gives the offset of the grid points; $k = x, y, z$-direction; ϕ = basis function with respect to the Euclidean distance $\|\cdot\|$; $\mathbf{x}_i$ = known centers (i.e., mesh boundary points); and N_b = number of boundary points. The coefficients α_i are determined by recovering known values of the offset at the boundary in the form

$$d_i^{(k)} = g^{(k)}(\mathbf{x}_i) \tag{2}$$

where $d_i^{(k)}$ = known displacement of the boundary point $\mathbf{x}_i$ in the k-direction. Typically, the term $p(\mathbf{x})$ in Eq. (1) is a linear polynomial which can recover rigid body translations exactly (De Boer et al. 2007). Estruch et al. (2013) obtained quantitative comparison between final meshes considering and discarding the polynomial term, demonstrating that the resultant meshes are very similar. Therefore this paper omits the additional linear term, and the global form of Eq. (2) can be written as

$$\mathbf{Y}^{(k)} = \mathbf{M}\alpha^{(k)} \tag{3}$$

where

$$\mathbf{Y} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N_b} \end{pmatrix}, \qquad \alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N_b} \end{pmatrix} \tag{4}$$

and

$$\mathbf{M} = \begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N_b} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N_b} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N_b1} & \phi_{N_b2} & \cdots & \phi_{N_bN_b} \end{pmatrix}, \qquad \phi_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) \tag{5}$$

Various radial basis functions are available and they can be divided into two categories: functions with global support and functions with compact support. Global functions are not equal to zero inside the whole interpolation space, which leads to dense matrix $\mathbf{M}$. Compact functions decay moving away from the center, and finally reach zero at the support radius R. Hence proper low values of R result in sparse matrix systems which can be solved more efficiently.

This paper uses Wendland's continuous polynomial (CP) C^2 function with compact support as the basis function. It was found to provide the best combination of deformation quality and matrix conditioning (De Boer et al. 2007; Rendall and Allen 2009; Rendall and Allen 2010), and it is defined as

$$\phi(\|\mathbf{x}_i - \mathbf{x}_j\|) = f(\|\mathbf{x}_i - \mathbf{x}_j\|/R) = \begin{cases} (1-\xi)^4(4\xi+1) & \xi < 1 \\ 0 & \xi \geq 1 \end{cases} \quad (6)$$

where $\xi = \|\mathbf{x}_i - \mathbf{x}_j\|/R$. Furthermore, with compact support, the far field static boundary cannot be involved as control points in some cases. This diminishes the size of the matrix system, reducing the calculation capacity. The selection criteria for R are given in section "Result."

With the preceding configuration, the distance matrix $\mathbf{M}$ is sparse, symmetric, and strictly positive definite, and the preconditioned conjugated gradient (CG) algorithm with the diagonal smoothing preconditioner (Saad 2003) can be applied to solve Eq. (3). In theory, the conjugate gradient algorithm is guaranteed to converge in N iterations, where N is the number of unknowns in the system. However, in practice, convergence is usually achieved at a much faster rate. On the other hand, direct solvers such as lower and upper (LU) factorization have the advantage of reusability, because the matrix $\mathbf{M}$ is factorized only once and applicable for three coordinate directions. Section "Result" quantitatively compares these two methods in detail. Once the interpolation function is obtained, the displacement of internal points can be calculated directly using their coordinate values.

Data Reduction Algorithm with Acceleration

In order to increase the speed of the RBF method for large-scale problems, the literature proposes a data reduction algorithm with greedy point selection (Rendall and Allen 2009, 2010; Sheng and Allen 2013). Its main idea is to reduce the number of boundary points selected as RBF centers and control the interpolation error. These selected points are also called control points. The typical procedure is as follows: first choose a subset of boundary points to compute the interpolation function, then calculate the displacement error of other boundary points with their computed and known displacement values; if the largest displacement error is smaller than a given criterion, use this function as the final interpolation function; otherwise, add one or more boundary points with the largest displacement error to the subset list and repeat the procedure. Fig. 1 is a flowchart of this greedy procedure.

The control points for each coordinate direction may or may not remain the same, depending on the error control criterion. This paper uses a general definition of the displacement error based on each point

$$D_{\text{error}} = \sqrt{(\Delta x - \Delta x_{\text{greedy}})^2 + (\Delta y - \Delta y_{\text{greedy}})^2 + (\Delta z - \Delta z_{\text{greedy}})^2} \quad (7)$$

This definition has a clear physical meaning and the resultant control points are selected one by one for all coordinate components. Moreover, the stop criterion ε should be less than the finest mesh spacing divided by the number of deforming steps, i.e., $\varepsilon <$ (minimum size/number of steps). This is important to control the accumulative error and avoid the negative control volumes in viscous flow computations.

Eq. (3) should be solved in each loop with an increasing size. Some acceleration techniques can be used to further accelerate the selection procedure. First, due to the symmetry of matrix $\mathbf{M}$, only half the values need to be computed and stored. Second, compared with the previous loop step, just one additional row and column are added to matrix $\mathbf{M}$, so the formation of the new matrix can be based on the old formation. Third, because there is only one new unknown compared with the previous linear system, efficiency can be further improved by making full use of the solution of the previous equation. For iterative techniques, the initial solution of the equation can be set as $\{\alpha_1, \alpha_2, \ldots, \alpha_k, 0\}$ based on the final solution of the last linear system, $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. This incremental approach further accelerates the data reduction algorithm.

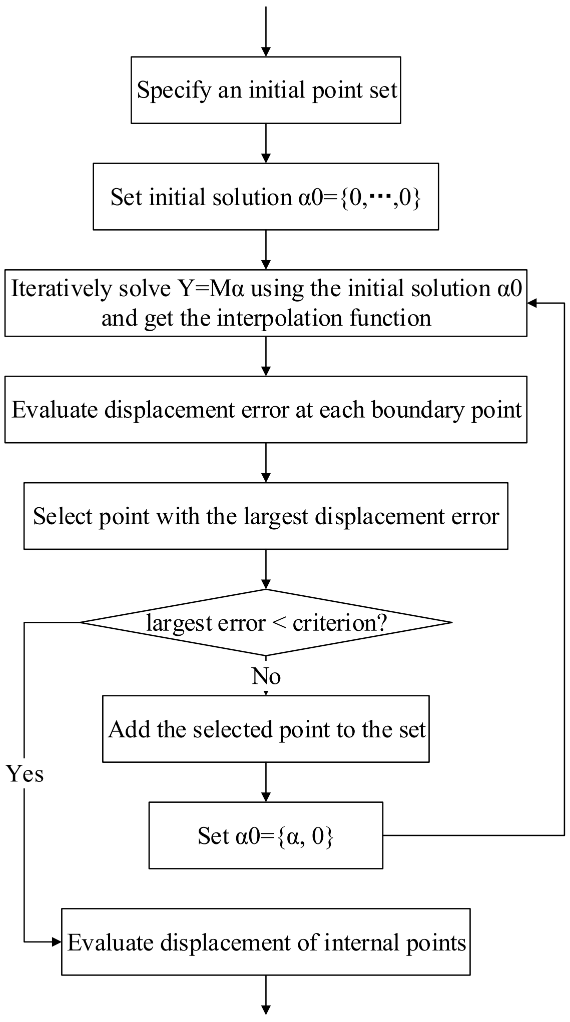
Specify an initial point set
Set initial solution α0={0,⋯,0}
Iteratively solve Y=Mα using the initial solution α0 and get the interpolation function
Evaluate displacement error at each boundary point
Select point with the largest displacement error
largest error < criterion?
No
Yes
Add the selected point to the set
Set α0={α, 0}
Evaluate displacement of internal points

Fig. 1. Flowchart of greedy point-selection procedure

For radial basis function interpolation, the fast multipole method (FMM) is usually applied to calculate the sums of weighted basis functions [see Eq. (1), or in matrix-vector product form] with massive data (Darve 2000; Fong and Darve 2009). However, the number of RBF centers is significantly reduced with the use of a data reduction algorithm. Rendall and Allen (2009) and Gillebaart et al. (2016) concluded that the number of control points selected by a greedy algorithm is grid-size independent, and is related only to the deformed geometry and its structural deformation. Therefore the number of control points will not be large even in large-scale three-dimensional (3D) meshes, so the FMM method is not suitable for use with a data reduction algorithm, and the authors' tests demonstrated this viewpoint with the usage of the black-box FMM (Fong and Darve 2009). Other optimizations such as adaptive (Gillebaart et al. 2016) and dynamic (Niu et al. 2017) control points could be applied in the present framework.

Global Procedure of Dynamic Motion Solver

Although the mesh deformation technique has been introduced, it needs to be coupled with CFD calculation in moving boundary

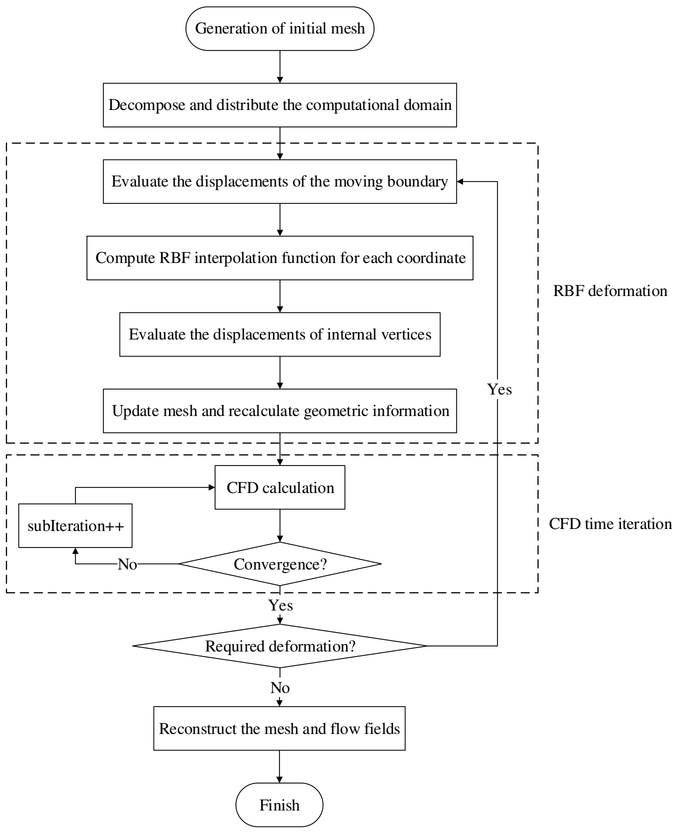
Generation of initial mesh
Decompose and distribute the computational domain
Evaluate the displacements of the moving boundary
Compute RBF interpolation function for each coordinate
Evaluate the displacements of internal vertices
Update mesh and recalculate geometric information
RBF deformation
Yes
CFD calculation
subIteration++
No
Convergence?
CFD time iteration
Yes
Required deformation?
No
Reconstruct the mesh and flow fields
Finish

Fig. 2. Global algorithm flowchart of parallel dynamic motion solver

numerical simulations. The displacements of the moving boundary points are usually obtained by computational structural dynamics (CSD) codes. Two approaches can be used to tackle the CFD/CSD coupling problem. The tight coupling approach requires solving both CFD and CSD as one coupled set of equations, and would require the complete rewrite of both solvers. In contrast, the loose coupling approach decouples the CFD and CSD sets of equations and uses projection methods to transfer interface information between the CFD and CSD domains (Baum et al. 2001; Biedron and Lee-Rausch 2008). The latter is more efficient with similar accuracy, and it can be coupled with different CSD applications without code change, which is more robust. The authors adopted the latter methodology in the proposed dynamic motion solver.

This section summarizes the global procedure of the parallel dynamic motion solver for generating the deformed computational mesh. Fig. 2 presents a flowchart of the algorithm. The solver first evaluates movement of the dynamic boundary based on the last information of the flow fields or some predefined motion equations. After the entire mesh is deformed, the CFD time iteration is executed. This paper mainly focuses on the RBF deformation.

When the motion solver runs in parallel, it needs to decompose and distribute the computational domain to each process in the preprocessing stage. Therefore the boundary points are scattered in each process at first. To ensure robustness, the authors keep a global interpolation function for the entire computational domain. All processes maintain a copy of the entire boundary points via a point-to-point communication, then each process performs the same operation to obtain the interpolation function. Because only a small number of control points is used, this takes an insignificant amount of time. Once the interpolation function is calculated, the solver evaluates the offsets of internal points in each domain simultaneously. After the displacements of the internal vertices are obtained, the solver updates the mesh and recalculate geometric quantities such as cell volumes. Finally, once the unsteady simulation is complete, the solver reconstructs the mesh and flow fields as a whole.

Implementation

The implementation of the proposed mesh motion solver is based on the prototype developed by Bos et al. (2013) under the framework of *OpenFOAM*. *OpenFOAM* is a C++ toolbox for the customization and extension of numerical solvers for continuum mechanics problems (Jasak et al. 2007). Benefiting from its hierarchical and modular design, many of the existing functions can be coupled with the new mesh motion solver for moving boundary simulations.

For convenience, all the parametric settings are centrally managed in the file called dynamicMeshDict. With the use of the run-time type selection mechanism in *OpenFOAM*, it allows the user to choose different flow solvers and boundary movement equations at run time. Because the matrix solvers in *OpenFOAM* are specially built for all mesh points based on the mesh data structure, the authors applied the general iterative linear solvers of the Eigen library (Guennebaud et al. 2010) for solving Eq. (3).

The implementation uses two parallel strategies. For predefined movement cases, because the position of the boundary points in next time step can be precalculated when the time step length is fixed, a special CPU process is applied to compute the interpolation function one step earlier than flow calculation. As a result, when a new time step begins, the ordinary processes first receive the function to predict the displacements of internal points sent by the special process, and then the special process start calculating the function for the next step. This strategy totally hides the function-forming cost, but is only valid for known boundary motions. For other cases, each process maintains a copy of the entire boundary points and forms the interpolation function simultaneously. To ensure that the solution of the equation is consistent, the order of global boundary points should be kept exactly the same in each process while using gather-scatter operations. There are duplicated shared boundary points after decomposition. These duplicated points need to be removed to form the global boundary list. In addition, the authors simply selected the first, middle, and last points of the global list as the initial control point set.

In *OpenFOAM*, two-dimensional problems are extended to three dimensions by applying special boundary conditions on any patch in the plane normal to the direction of interest. Original two-dimensional meshes are extended one layer in the third direction, and then cell elements such as triangles become triangular prisms. For two-dimensional applications, an additional correction step after the RBF deformation method ensures that the upper and lower surfaces of the grid are exactly the same in the viewpoint of the third direction.

Furthermore, to ensure maximum efficiency and robustness, this new mesh motion solver can run in parallel and sequential mode, which preserves the quality of the mesh and keeps the computational resources affordable. For the purpose of simplicity, this paper investigates only the performance of mesh deformation.

Results

The new mesh deformation method was tested with four numerical examples to demonstrate its efficiency and deforming capability. Typical cases involving rotation of an airfoil, relative motion of a multibody, pitching of a reentry capsule, and a three-dimensional

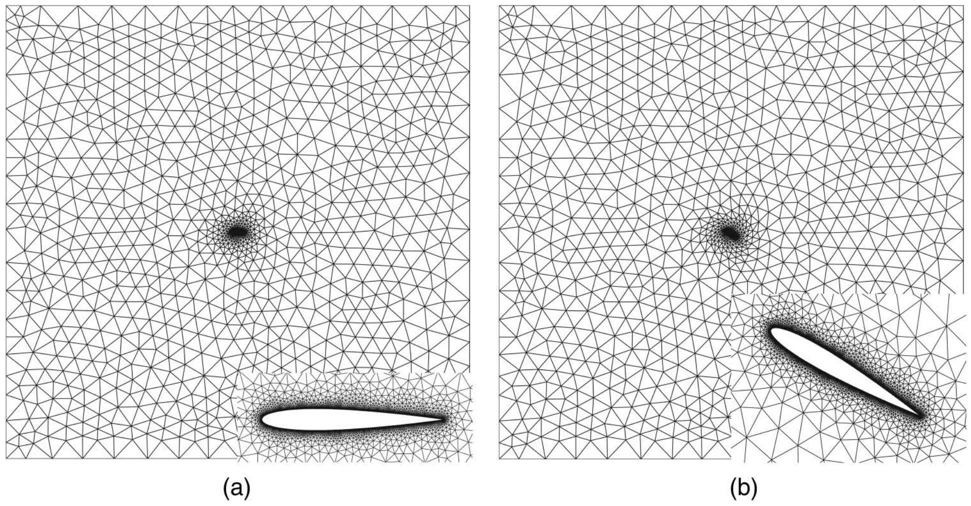

Fig. 3. Initial and final meshes of NACA 0012 airfoil with detailed view

bending wing were simulated. All cases were tested on a high-performance cluster node with two Intel Xeon E5-2692 v2 CPUs, with a total of 24 cores at 2.20 GHz, and 64 GB memory.

The authors used the size-skew metric f_{ss} to quantitatively measure the mesh quality for each mesh element (Knupp 2003; De Boer et al. 2007)

$$f_{ss} = \sqrt{f_{\text{size}} f_{\text{skew}}} \tag{8}$$

where f_{size} = relative size metric that reveals the change in area/volume; and f_{skew} = skew metric is to detect element distortions. The size-skew metric is 0 when the element is degenerate or has negative area/volume. When f_{ss} reaches the maximum of 1, the area/volume of the element is not changed after motion and the deformed element attains the ideal node configuration. The minimum and average f_{ss} values of the mesh are calculated to represent its mesh quality. Both parallel strategies introduced in section "Implementation" were applied in the parallel cases to investigate their performance.

Rotation of Airfoil

In the first test case, a National Advisory Committee for Aeronautics (NACA) 0012 airfoil was rotated 30° clockwise at its quarter-chord length, which is a very simple and typical example for mesh deformation. Fig. 3 shows the initial and final meshes. The mesh had 13,112 inner points, 3,128 boundary points, and 29,352 triangular elements, which were extended to three dimensions. The size of the computational domain was $20C \times 20C$, where C is the chord length of the airfoil. The mesh deformation was performed in three steps, and the airfoil rotated 10° in each step. The capability of the RBF method to perform large deformations in a single step is well demonstrated in the literature (De Boer et al. 2007; Rendall and Allen 2009). This paper mainly focuses on investigating the performance of the parallel RBF approach; therefore the latter three cases use relatively more deforming steps to ensure a sufficient amount of calculation to obtain its parallel performance.

To investigate the influence of the support radius R of the CP C^2 function for mesh quality, the authors performed the deformation with different radii and used the chord length C as the unit of the radius. The results in Fig. 4 show that the minimum size-skew metric of the final mesh exhibited a wavy attenuation. When the radius was five times the chord length, the final mesh reached its best quality (Fig. 3). Moreover, Fig. 4 shows that the mesh quality was nearly the same with or without including outer static boundary points in the preselected list for control points. This is because the outer boundary points are not in the working scope of the RBF centers with small support radius, so the displacements of these outer points are zero if calculated, which equals their expected values. Therefore users can exclude outer static boundary points for interpolation function calculation. In this case, this reduced the computation time by 16%.

Table 1 presents the performance of different matrix solvers in the function coefficient solving procedure. Here, the support radius was $5C$ and the stop criterion ε of the data reduction algorithm was set to 10^{-5}. The LU and CG methods use all the moving boundary points as control points in the LU factorization and CG algorithm, respectively. This demonstrates that iterative methods are much more efficient than direct solvers in solving large sparse linear systems. When using the data reduction algorithm, the performances of the LU-Greedy and CG-Greedy methods were close, and were approximately 20 times better than the performance of the CG method. The CG-Greedy-LastValue method uses the solution of the previous step as the initial solution of this step in the greedy selection procedure; this technique further accelerated the performance nearly 50%.

To evaluate the parallel performance of the mesh motion solver, a refined NACA 0012 mesh with over 10^6 grid points

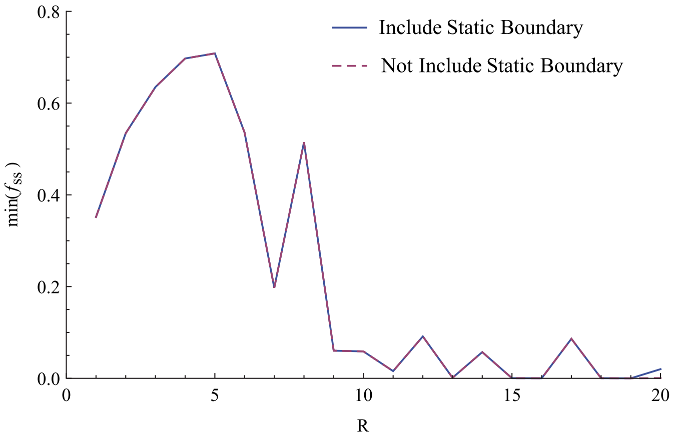

Fig. 4. Minimum size-skew metric of final NACA 0012 mesh with different support radii

Table 1. Performance of Different Matrix Solvers in Case of Airfoil Rotation

Method	Time (s)
LU	1,517.21
CG	303.6
LU-Greedy	14.35
CG-Greedy	14.38
CG-Greedy-LastValue	9.65

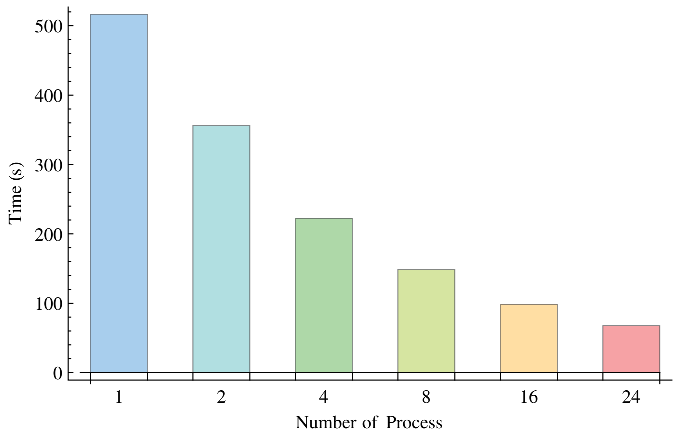

Fig. 5. Parallel performance of motion solver using mesh of more than 10^6 points

was generated to perform the airfoil rotation. Fig. 5 shows the total CPU time of the motion solver running with different processes. As the number of processes increased, the cost of the solver decreased rapidly. Because the solver calculates the interpolation function in all the processes, the separated parallel scalability of the motion solver is not particularly desirable. If it were coupled with CFD time iteration, the parallel performance of the solver could be more efficient.

Relative Motion of Multibody

This case demonstrates the deforming capability of the RBF method for relative motion of a multibody with a complex boundary. The tail wing of a 30P30N multielement airfoil was rotated 30° clockwise in 30 steps. Fig. 6 illustrates the initial and final

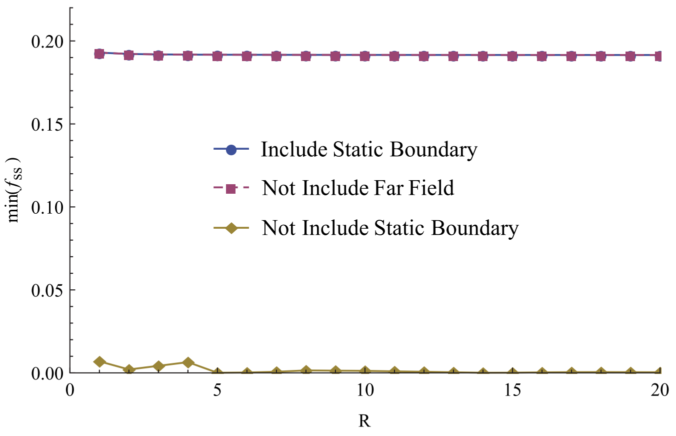

Fig. 7. Minimum size-skew metric of final 30P30N mesh with different support radii

meshes of the 30P30N multielement airfoil. The mesh had 65,170 internal points and 1,440 boundary points, and extended $25C$ from the airfoil with a circular outer boundary, where C is the chord length of the tail wing.

Fig. 7 shows the minimum f_{ss} of the final mesh with different support radii. Again, the authors used the chord length C as the unit of the radius. If the preselected list for control points does not include any static boundary points, the final mesh is almost invalid after deformation. On the other hand, if it includes all the static boundary points or excludes the far field boundary points, the quality of the final meshes is very high and nearly identical. In this case, it can be inferred that only far field static boundary points that are far from the objects can be neglected for RBF function calculation. Furthermore, the mesh quality was not sensitive to support radius in this case. The following tests used a radius of $4C$.

Fig. 8 depicts the size-skew metric of the elements near the tail wing after deformation. The boundary meshes still moved together with the tail, and their mesh quality was preserved well during the rotation. In addition, Fig. 9 gives the average and minimum mesh quality of the 30P30N mesh in each deforming step. The average quality of the grid was basically stable during the deformation process and tended to be ideal. The minimum value of f_{ss} was high in the first eight steps. With the increase of the deformation amplitude, the minimum cell quality gently decreased to 0.2, and only several mesh cells reached this value shown in Fig. 8.

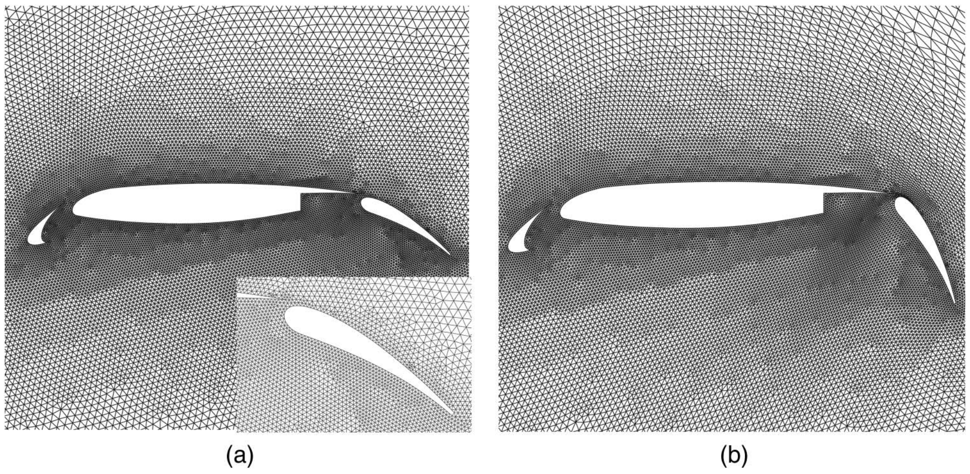

Fig. 6. Initial and final meshes of 30P30N multielement airfoil with detailed view

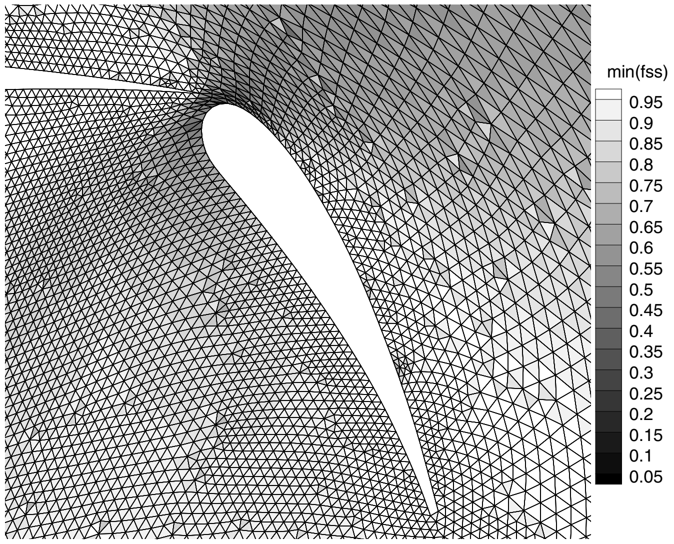

Fig. 8. Closeup of tail wing after rotating 30°

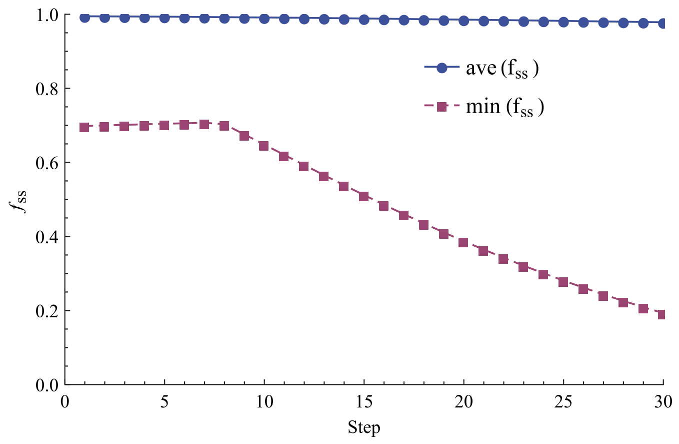

Fig. 9. Average and minimum quality of 30P30N mesh in each deforming step

The results presented in Table 2 demonstrate similar behaviors of each matrix solver as those in Table 1 in the first case. The stop criterion ε of the greedy method was set to 10^{-6} based on the minimum mesh spacing divided by the number of deforming steps in this case. The greedy selection method greatly accelerated the function coefficient solving procedure, and with the incremental approach, the performance of the coefficient solving procedure further accelerated by approximately two times for this case. Moreover, due to the further overhead compression of this serial part, the parallel efficiency of the mesh motion solver was improved.

Pitching of Reentry Capsule

At present, the space and earth transport system is mainly dependent on the blunt cone–type reentry capsule, and the dynamic stability of the capsule is critical. This case presents the oscillatory and transient pitching of a reentry capsule. Fig. 10 shows the computation domain of the initial mesh. The tetrahedral mesh had 303,927 internal points and 61,514 boundary points. The pitching center was located at 20% of the axis of the capsule, and the amplitude was 20° in the y-direction.

Table 2. Performance of Different Matrix Solvers in Case of Airfoil Relative Motion

Method	Time (s)
LU	244.42
CG	122.95
LU-Greedy	49.51
CG-Greedy	43.3
CG-Greedy-LastValue	25.53

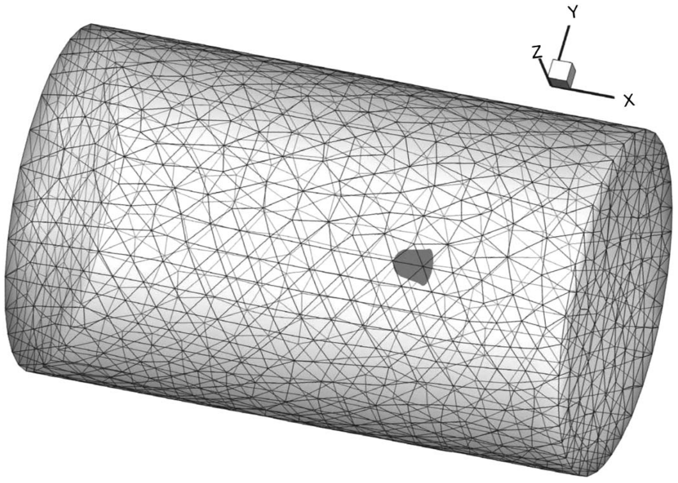

Fig. 10. Computation domain of initial mesh for reentry capsule

The deformation performed three cycles in 120 steps. In accordance with the previous conclusion, the static points of the far field boundary were not included in the list to perform greedy selection. The stop criterion ε of the greedy method was 2×10^{-3}, which was equal to the finest mesh spacing divided by 40, the number of steps for a deforming cycle. In addition, the authors set the support radius of CP C^2 function to $3C$, where C is the length of the central axis of the capsule. Fig. 11 illustrates the surface meshes of the reentry capsule at initial and the maximum amplitude.

Fig. 12 presents the average and minimum quality of the tetrahedral mesh in each deforming step. The mesh quality was maintained very well during the entire deformation process. In this case, the mesh quality was similar whether it selected control points in every deforming step or just selected once in the first step. Therefore, for relatively simple motion, the solver can select control points once to further improve its efficiency.

Fig. 13 presents the parallel acceleration of the motion solver relative to the number of processes. Because the number of internal points was not very large in this case, the precalculated parallel strategy had a superlinear acceleration subjected to cache effects. When the number of processes is larger than 10, all the data can be loaded into the cache to greatly improve the memory access efficiency. On the other hand, because the shape of the capsule was complex, the solver needed to select more control points to represent the deformation. Hence the acceleration of the normal strategy with every step selection was limited by the matrix solving.

Bending of 3D Wing

This three-dimensional case was constructed by artificially bending an Office National d'Etudes et de Recherches Aérospatiales (ONERA) M6 wing, which was previously presented by Liu et al. (2012).

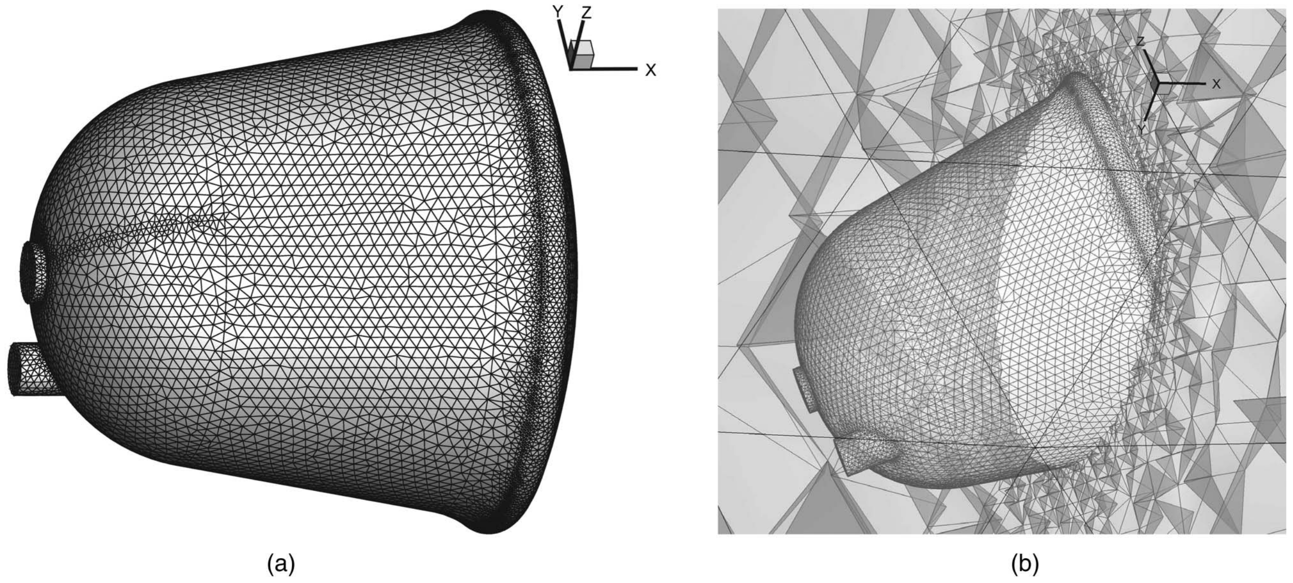

Fig. 11. Surface mesh of reentry capsule at initial and maximum amplitudes

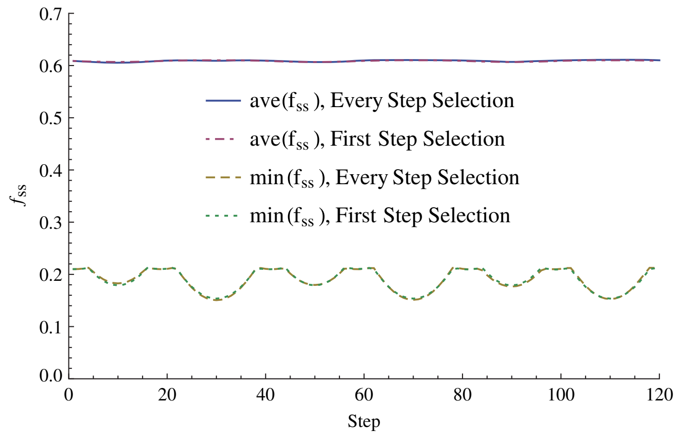

Fig. 12. Average and minimum quality of reentry capsule mesh in each deforming step

A periodic vertical movement was prescribed at the wing tip, and the motion can be formulated as follows:

$$h_{\mathrm{tip}} = h_0 \sin(2\pi t)$$
$$h(z) = (z/z_{\mathrm{tip}})^2 h_{\mathrm{tip}} \tag{9}$$

where h_{tip} = displacement of the wing tip in the y-direction; and z_{tip} = distance from the wing tip to the root. The wing root was fixed, and a quadratic variation of the bending motion was prescribed between the wing tip and the root; z is the distance from the point on the wing surface to the symmetric plane at the wing root, and $h(z)$ is the vertical displacement of the point. In this case, the bending magnitude h_0 equaled C, where C is the chord length of the wing.

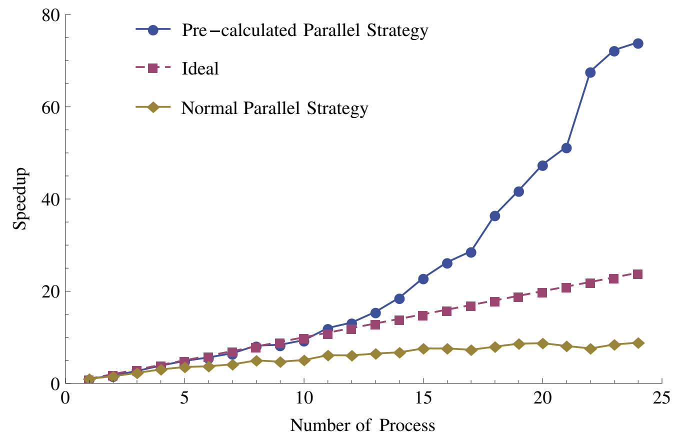

Fig. 13. Parallel speedup of different calculation module in the case of pitching reentry capsule

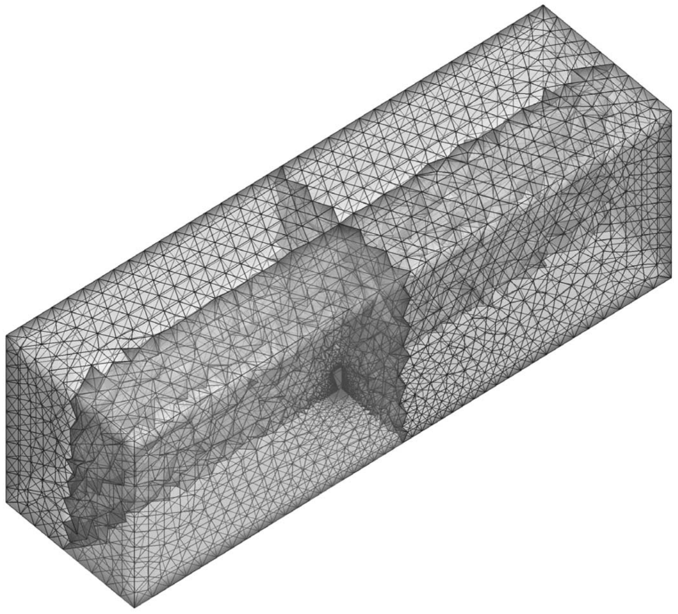

Fig. 14. Computation domain of initial ONERA M6 mesh decomposed into four parts

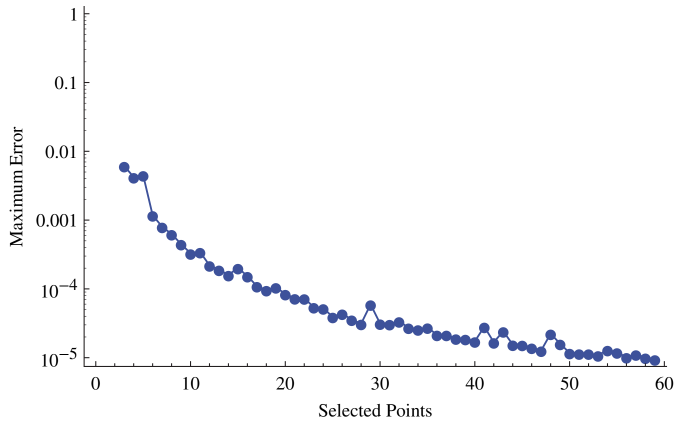

Fig. 15. Maximum displacement error in greedy selection procedure

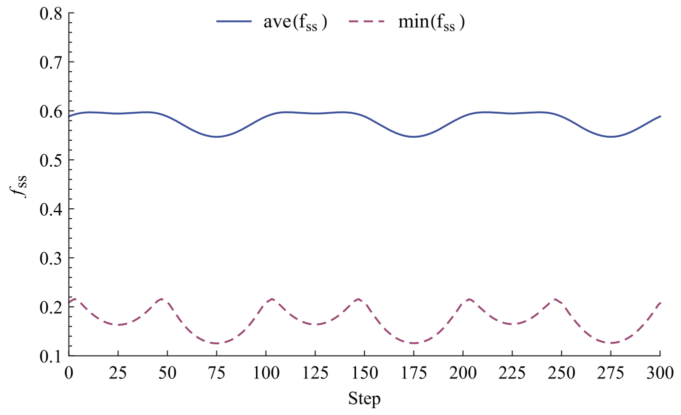

Fig. 17. Average and minimum quality of M6 mesh in each deforming step

Fig. 14 shows the computational domain of the wing. It was decomposed into four parts for parallel computing. The mesh had 1,579,843 internal points and 34,992 surface points. The number of boundary points was so large that an unacceptable run time was needed to obtain the results directly using the full RBF interpolation on this platform. Therefore data reduction with an incremental approach was directly applied in this test case, and the far field boundary points were neglected in the control points selection.

Fig. 15 shows the maximum displacement error of the boundary points during the greedy selection procedure of the first deforming step; its value decreased rapidly as the maximum error point was selected one by one. In this case, the authors set the stop criterion ε to 10^{-5}, and the support radius of the kernel function was set to $3C$, i.e., three times the chord length of the wing.

The test performed three deformation cycles in 300 steps. Fig. 16 presents the surface meshes of the M6 wing at the initial and largest bending displacements after deformation. Fig. 17 shows the average and minimum tetrahedral mesh quality after each deforming step. The mesh quality changed cyclically, indicating that when the boundary moves to its original position, the mesh quality returns to its original level. This demonstrates that the proposed motion solver is robust enough to handle periodic motion problems.

In this case, the authors tested the parallel acceleration of two proposed parallel strategies of the motion solver (Fig. 18). The precalculated strategy computed the displacement of each inner point separately in simulating processes, so there was no data dependency. The parallel acceleration of this part was nearly ideal. The acceleration of the normal strategy was also considerable; its parallel efficiency while using 24 processes was approximately 50%.

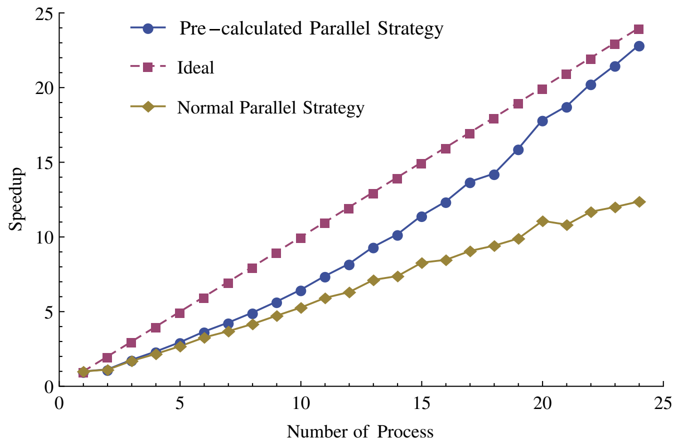

Fig. 18. Parallel acceleration of different calculation module in case of 3D bending wing

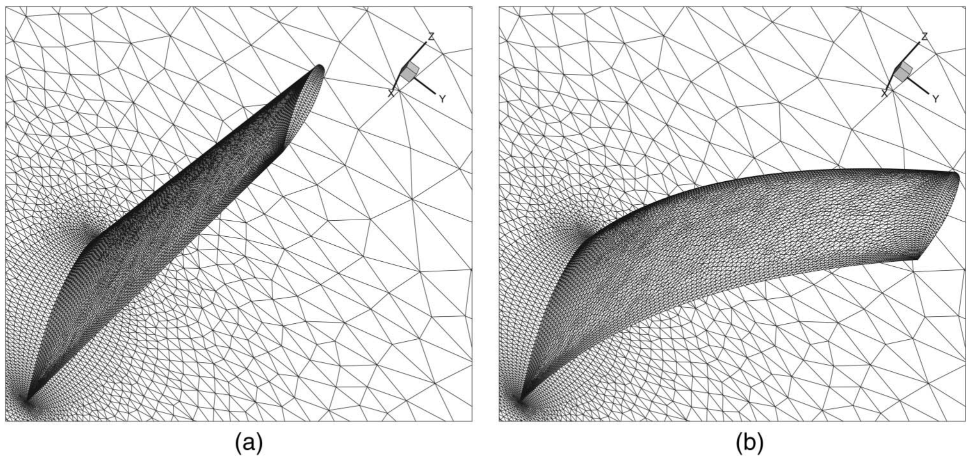

Fig. 16. Surface mesh of M6 wing at initial and largest bending displacements

Conclusions

This paper developed an efficient and robust parallel mesh motion solver using the radial basis function interpolation. The proposed mesh deformation method uses only the coordinates of grid points, without their topological relations. Based on the data reduction algorithm, an incremental approach was presented to further accelerate the formation of the interpolation function. The new motion solver was developed based on the *OpenFOAM* platform, and it can be directly coupled with different flow solvers and boundary motion equations. Therefore the solver is applicable for various moving boundary problems and all types of grids. The parallel method treats predefined boundary movements differently, and a more efficient parallel strategy is applied. For non-predefined boundary movements, the solver maintains a copy of the entire boundary points in each process, then evaluates the displacements of internal points in each domain with the same interpolation function. The method is easily parallelized and no additional checking mechanism is required.

The proposed motion solver was tested with the compact supported CP C^2 function for four typical cases to demonstrate its performance and robustness, and several parametric setting rules were provided. The rotation of an airfoil case showed that the support radius of the function can be set to two to six times the length of the deformable part of the object. The relative motion of a multibody case demonstrated that the static points of the far field boundary do not need to be included for the greedy selection in external flow problems. The stop criterion ε of the greedy selection approach should be less than the finest mesh spacing divided by the number of deforming steps to avoid negative control volumes. The pitching of a reentry capsule case and the bending of a three-dimensional wing case demonstrated the deforming capability of the solver applied to three-dimensional problems. The parallel performance of the proposed solver is limited by the interpolation function computing for small grids, but the proportion of the function-forming cost is much lower in large-scale cases, which makes the parallel solver efficient and robust in large-scale simulations.

Further research includes thread-level parallelism with OpenMP (2015) for the conjugated gradient algorithm and the internal evaluation, and applying the mesh motion solver to real flow simulations such as aerodynamic shape optimization problems.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China under Grants Nos. 11502296, 61772542, and 61561146395; the Foundation of National University of Defense Technology under Grant No. ZDYYJCYJ20140101; the Open Research Program of China State Key Laboratory of Aerodynamics under Grant No. SKLA20160104; and the Defense Industrial Technology Development Program under Grant No. C1520110002.

References

Batina, J. T. (1991). "Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis." *AIAA J.*, 29(3), 327–333.

Baum, J. D., et al. (2001). "Recent developments of a coupled CFD/CSD methodology." *Proc., Int. Conf. on Computational Science–ICCS 2001*, Springer, Berlin, 1087–1097.

Biancolini, M., Viola, I., and Riotte, M. (2014). "Sails trim optimisation using CFD and RBF mesh morphing." *Comput. Fluids*, 93(1), 46–60.

Biedron, R., and Lee-Rausch, E. (2008). "Rotor airloads prediction using unstructured meshes and loose CFD/CSD coupling." *26th AIAA Applied Aerodynamics Conf.*, AIAA, Reston, VA.

Bos, F. M., van Oudheusden, B. W., and Bijl, H. (2013). "Radial basis function based mesh deformation applied to simulation of flow around flapping wings." *Comput. Fluids*, 79(1), 167–177.

Coulier, P., and Darve, E. (2016). "Efficient mesh deformation based on radial basis function interpolation by means of the inverse fast multipole method." *Comput. Methods Appl. Mech. Eng.*, 308(1), 286–309.

Darve, E. (2000). "The fast multipole method: Numerical implementation." *J. Comput. Phys.*, 160(1), 195–240.

De Boer, A., Der Schoot, M. S. V., and Bijl, H. (2007). "Mesh deformation based on radial basis function interpolation." *Comput. Struct.*, 85(11–14), 784–795.

Estruch, O., Lehmkuhl, O., Borrell, R., Segarra, C. P., and Oliva, A. (2013). "A parallel radial basis function interpolation method for unstructured dynamic meshes." *Comput. Fluids*, 80(1), 44–54.

Fong, W., and Darve, E. (2009). "The black-box fast multipole method." *J. Comput. Phys.*, 228(23), 8712–8725.

Gao, X., Dong, Y., Xu, C., Xiong, M., Wang, Z., and Deng, X. (2017). "Developing a new mesh deformation technique based on support vector machine." *Int. J. Comput. Fluid Dyn.*, 31(4–5), 246–257.

Gillebaart, T., Blom, D., van Zuijlen, A., and Bijl, H. (2016). "Adaptive radial basis function mesh deformation using data reduction." *J. Comput. Phys.*, 321(1), 997–1025.

Guennebaud, G., Jacob, B., Avery, P., Bachrach, A., and Barthelemy, S. (2010). "Eigen v3." ⟨http://eigen.tuxfamily.org⟩ (Apr. 8, 2017).

Jasak, H., Jemcov, A., and Tukovic, Z. (2007). "OpenFOAM: A C++ library for complex physics simulations." *Int. Workshop on Coupled Methods in Numerical Dynamics*, Vol. 1000, Inter-University Center Dubrovnik, Dubrovnik, Croatia, 1–20.

Kedward, L., Allen, C. B., and Rendall, T. (2017). "Efficient and exact mesh deformation using multiscale RBF interpolation." *J. Comput. Phys.*, 345(1), 732–751.

Knupp, P. M. (2003). "Algebraic mesh quality metrics for unstructured initial meshes." *Finite Elem. Anal. Des.*, 39(3), 217–241.

Li, C., Yang, W., Xu, X., Wang, J., Wang, M., and Xu, L. (2017). "Numerical investigation of fish exploiting vortices based on the Kármán gaiting model." *Ocean Eng.*, 140(1), 7–18.

Liu, Y., Guo, Z., and Liu, J. (2012). "RBFS-MSA hybrid method for mesh deformation." *Chin. J. Aeronaut.*, 25(4), 500–507.

Luke, E. A., Collins, E., and Blades, E. (2012). "A fast mesh deformation method using explicit interpolation." *J. Comput. Phys.*, 231(2), 586–601.

Lynch, D. R., and O'Neill, K. (1980). "Elastic grid deformation for moving boundary problems in two space dimensions." *Finite Elem. Water Res.*, 2(7), 111–120.

Michler, A. K. (2011). "Aircraft control surface deflection using RBF-based mesh deformation." *Int. J. Numer. Methods Eng.*, 88(10), 986–1007.

Niu, J., Lei, J., and He, J. (2017). "Radial basis function mesh deformation based on dynamic control points." *Aerosp. Sci. Technol.*, 64(1), 122–132.

OpenMP. (2015). "OpenMP application programming interface." ⟨http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf⟩ (Nov. 2015).

Rendall, T., and Allen, C. (2008). "Unified fluid-structure interpolation and mesh motion using radial basis functions." *Int. J. Numer. Methods Eng.*, 74(10), 1519–1559.

Rendall, T., and Allen, C. B. (2009). "Efficient mesh motion using radial basis functions with data reduction algorithms." *J. Comput. Phys.*, 228(17), 6231–6249.

Rendall, T., and Allen, C. B. (2010). "Reduced surface point selection options for efficient mesh deformation using radial basis functions." *J. Comput. Phys.*, 229(8), 2810–2820.

Saad, Y. (2003). *Iterative methods for sparse linear systems*, SIAM, Philadelphia.

Sheng, C., and Allen, C. B. (2013). "Efficient mesh deformation using radial basis functions on unstructured meshes." *AIAA J.*, 51(3), 707–720.

Stadler, D., Kosel, F., Celic, D., and Lipej, A. (2011). "Mesh deformation based on artificial neural networks." *Int. J. Comput. Fluid Dyn.*, 25(8), 439–448.

Wang, Z. J., and Przekwas, A. J. (1994). “Unsteady flow computation using moving grid with mesh enrichment.” *Technical Rep. AIAA-94-0285*, CFD Research Corporation, Huntsville, AL.

Witteveen, J., and Bijl, H. (2009). “Explicit mesh deformation using inverse distance weighting interpolation.” *19th AIAA Computational Fluid Dynamics, Fluid Dynamics and Co-Located Conf.*, AIAA, Reston, VA.

Witteveen, J. A. (2010). “Explicit and robust inverse distance weighting mesh deformation for CFD.” *48th AIAA Aerospace Sciences Meeting*, Vol. 165, AIAA, Reston, VA.

Yokota, R., Barba, L. A., and Knepley, M. G. (2010). “PetRBF—A parallel O(N) algorithm for radial basis function interpolation with Gaussians.” *Comput. Methods Appl. Mech. Eng.*, 199(25), 1793–1804.