A PARALLEL MULTI-SELECTION GREEDY METHOD FOR THE RBF MESH DEFORMATION IN OPENFOAM

CHAO LI[1,2], XIAOWEI GUO[1,3], CHENGKUN WU[1], XIANG ZHANG[1],
YI LIU[1], LIHUAN YUAN[1], SIJIANG FAN[1], CANQUN YANG[1]
[1]*College of Computer, National University of Defense Technology, Changsha 410073, China*
[2]*Email: dirk911@nudt.edu.cn*
[3]*Email: guoxiaowei@nudt.edu.cn*

Keywords: *CFD Mesh deformation; Radial basis function; Greedy algorithm; Multi-selection; Parallelization*

For a general CFD simulation with moving boundaries, the mesh deformation procedure could be logically divided into two steps. In the first step, the deformation of the mesh on the surface is performed based on a predefined equation or coupled iterative calculations. In the second step, the volume mesh is deformed to coordinate with the surface mesh using a specific method. The radial basis function(RBF) based mesh deformation is a point-by-point mesh movement scheme, which interpolates the displacement of the surface mesh to all the nodes of the flow mesh. The whole mesh points could be grouped into three categories: moving boundary points($\boldsymbol{x}_m$), static boundary points($\boldsymbol{x}_s$) and internal points($\boldsymbol{x}_{in}$). Based on a given basis function, the motion of any of these points could be calculated by:

$$s(x)=\sum_{j=1}^{N_b}\lambda_j\phi\left(\left\|x-x_j\right\|\right) \tag{1}$$

where x_{bj} is the coordinate of the jthboundary point, φ is the radial basis function as function of the Euclidean distance $\|x\|$, N_b is the number of the boundary points and λ_j is the weight coefficients. The conventional RBF mesh deformation would use all the boundary points to calculate the weight coefficients, which results in an interpolation scale of N_b^3. During the interpolation for the motion of the internal points, a time scale of $N_{in}\times N_b$ (N_{in} is the number of internal points) is introduced. Considering the whole mesh deformation procedure, it may be observed that the number of the boundary points N_b is the dominant factor in the cost of the method. Therefore, for large-scale 3D simulations with complex moving boundaries, N_b may reach to a very high value, which will result in a huge RBF system to be solved. In addition, if the mesh deformation is performed at each time step, the computational cost will be unacceptable.

An effective solution is to select a reduced subset from the boundary points to represent the surface meshes. These selected points, labeled as control points($\boldsymbol{x}_c$), then could be used instead to interpolating the motion of the internal points. Then new problems arises: how many control points should be selected and how to determine them. An ideal criterion is that the selected control points should be the minimum subset which could represent the geometry and motion of the boundary to a given degree of accuracy. This means N_c (number of the control points) should not be too large, which would influence the efficiency of the computation, nor too small, which might reduce the accuracy of the interpolation. Considering this appealing criterion, a real surface error based greedy algorithm is proposed by Rendall and Allen and the detailed procedure could refer to [1].

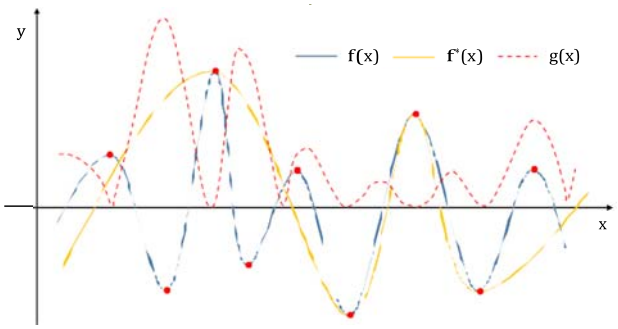

Figure 1: Illustration for the curvefitting.

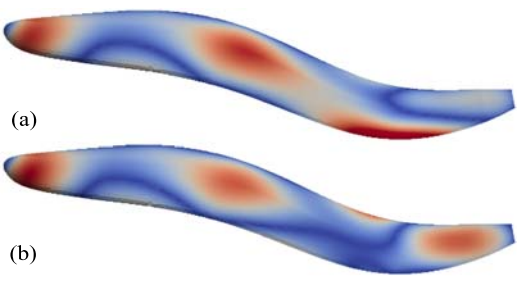

Figure 2: Distributions of the normalized boundary error when (a) $N_c = 10$ and (b) $N_c = 11$.

From the results discussed by Rendall and Allen, the greedy algorithm could effectively reduce the number of boundary points used to interpolating the motion of the volume mesh. But the time cost is reasonable only when the

final value of N_c is smaller than 400. The grey curve in Fig. 5 shows the time cost of the original greedy algorithm as a function of N_c. We could see the cost is in an exponential growth when N_c increases. Especially when N_c is larger than $1,000$, it will take several hours for the point selection procedure, which is unacceptable even for cases that only perform the greedy procedure once at the beginning. Consequently for large-scale $3D$ simulations with complex deformations, when the required N_c is large, the greedy procedure will become a bottleneck for the mesh deformation. To accelerate this procedure, a multi-selection method and a hybrid decomposition parallel algorithm are proposed and implemented respectively in OpenFOAM.

In the original greedy algorithm, only one point is added to the subset at each loop step, which will then lead to an inefficiency problem. It is easy to imagine that if multiple points are selected at each step, the loop times should be reduced accordingly. The abstract of the greedy algorithm is to select a small subset to represent a large data set, which is similar to the idea of curve fitting. As illustrated in Fig. 1, $f(x)$ is a curve with 9 peaks or valleys(red points), $f^*(x)$ is the fitting curve, $g(x)$ represents $|f(x) - f^*(x)|$. It is obvious that for $f^*(x)$, the fastest way to approaching to $f(x)$ is to select the 9 feature points as its sample data and currently 4 of them have already been included. For the remaining 5 unselected points, we could see they are generally located in the similar positions(x axis) as the points of $g(x)$ that contain the largest local maximum values(except for the far left one). That is to say, the largest local maximums of $g(x)$ are probably(not absolutely) corresponding to the feature points. As an analogy, we could take the actual boundary displacements as $f(x)$, the interpolated displacements as $f^*(x)$ and the displacement errors as $g(x)$. Thus, in a similar way, the points that contain the local maximum errors may correspond to the feature points to be selected. To validate this idea, the distributions of the surface error with $N_c = 10$ and $N_c = 11$ is plotted in Fig. 2. It could be clearly noted there are several local maximum errors scattered on the surface. This demonstrates that we could select multiple points that contain the local maximum errors at each step in the greedy algorithm.

Algorithm 1 The multi-selection greedy algorithm

Require: $\boldsymbol{x}_b, \Delta\boldsymbol{x}_b, \xi_{tol}$
Ensure: $\boldsymbol{x}_c, \Delta\boldsymbol{x}_c$
1: neighbors[][] $= ConstructBdyNeib(\boldsymbol{x}_b)$
2: $\boldsymbol{x}_c = \boldsymbol{x}_b[0]$ $\quad$ $\Delta\boldsymbol{x}_c = \Delta\boldsymbol{x}_b[0]$
3: **repeat**
4: $\quad \varepsilon_{pre} = \| \boldsymbol{\varepsilon} \|_2 \ / \ \| \Delta\boldsymbol{x}_b \|_2$
5: $\quad \boldsymbol{\lambda}_c = \boldsymbol{\Phi}_{c,c}^{-1} \Delta\boldsymbol{x}_c$
6: $\quad \Delta\boldsymbol{x}_b^* = \boldsymbol{\Phi}_{b,c} \boldsymbol{\lambda}_c$
7: $\quad \boldsymbol{\varepsilon} = \Delta\boldsymbol{x}_b^* - \Delta\boldsymbol{x}_b$
8: $\quad \varepsilon_{cur} = \| \boldsymbol{\varepsilon} \|_2 \ / \ \| \Delta\boldsymbol{x}_b \|_2$
9: $\quad \boldsymbol{\varepsilon}_{mxm}[] = 0$ $\quad$ indices$_{mxm}[] = 0$
10: $\quad$ **for** $i = 0$ to $N_b - 1$ **do**
11: $\quad\quad$ **if** $\| \boldsymbol{\varepsilon}[i] \| > \| \boldsymbol{\varepsilon}$[neighbors$[i][0]] \| \ \&\& \ldots \&\& \ \| \boldsymbol{\varepsilon}[i] \| > \| \boldsymbol{\varepsilon}$[neighbors$[i][end]] \|$ **then**
12: $\quad\quad\quad \boldsymbol{\varepsilon}_{mxm} = (\boldsymbol{\varepsilon}_{mxm}, \boldsymbol{\varepsilon}[i])$ $\quad$ indices$_{mxm} =$ (indices$_{mxm}, i$)
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ Select multiple control points based on $\boldsymbol{\varepsilon}_{mxm}$ and indices$_{mxm}$
16: **until** $\varepsilon_{cur} < \xi_{tol}$

Based on the above idea, we have proposed a multi-selection greedy algorithm as presented in Algorithm 1. Step 1 is to construct a $2D$ array which contains the neighbors of each boundary point. After the initialization, the greedy loop is performed from Step 3 to 16. In Steps 5-7, the boundary displacement errors are firstly calculated. Then a loop is executed to calculate the local maximum errors in Steps 10-14. The arrays $\boldsymbol{\varepsilon}_{mxm}$ and indices$_{mxm}$ respectively contain the specific values and indices of current local maximum errors. In step 15, multiple points with local maximum errors will be selected. A remaining question to consider is how many exactly control points should be selected. Here we implemented tow different strategies. Strategy 1 is to select a constant number(N_{cst}) of control points at each loop step and Strategy 2 is an adaptive strategy based on the gradient of the error reduction. For either Strategy 1 or Strategy 2, multiple points will be selected at each loop step, thus compared to the original greedy algorithm, Algorithm 1 is thought to have a faster convergence speed.

The RBF mesh deformation with data reduction could be divided into two major steps: (1) Reduce the number of control points by the greedy algorithm; (2) Interpolate the motion of the internal points based on the selected control points. Our parallelization work is mainly focused on $\boldsymbol{\Phi}_{b,c}$(a matrix to calculate the interpolated displacements of the boundary points) in the greedy procedure and $\boldsymbol{\Phi}_{in,c}$(a matrix to calculate the motion of the internal points) in the interpolation procedure. At present, most CFD tools employ parallel algorithms based on the mesh decomposition. The mesh points are decomposed into multiple parts at beginning of the simulation, and each processor reads a portion of the

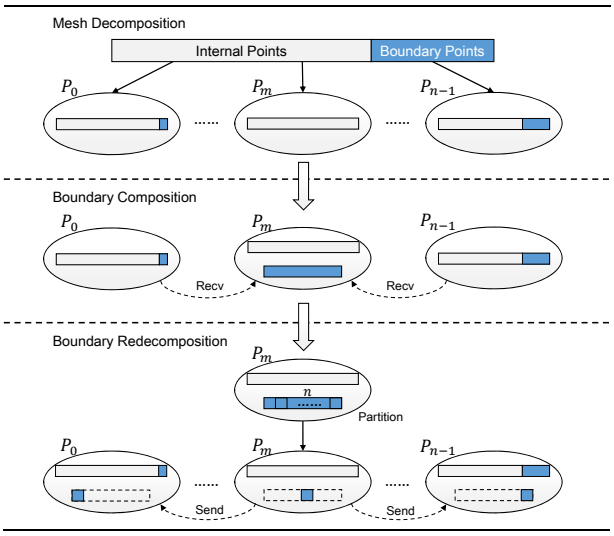

Figure 3: The hybrid decomposition parallel method.

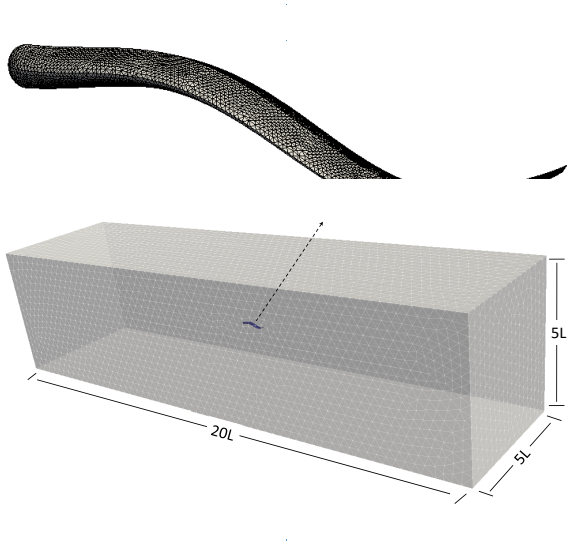

Figure 4: Computational mesh for the $3D$ undulating fish.

points and then begins the computation. In most cases the graph partition algorithm is used based on the relations between mesh cells. Generally when the degree of parallelism is small, both the boundary points and the internal points could be evenly allocated to each processor. However, when the degree of parallelism increases, it's hard to ensure all processors be allocated with the boundary points. This will then lead to a load imbalance problem.

To address the problem above, we proposed a hybrid decomposition parallel method as illustrated in Fig. 3. The arrays of the coordinates of internal points and boundary points are separately colored by grey and blue. Firstly in the mesh decomposition procedure, all the mesh points are allocated to n processors. We could see the boundary points are unevenly distributed. Then in the following step, a boundary composition procedure is performed. All the processors send their local arrays of the boundary points to a specific processor P_m. By assembling these scattered data, a global array containing the coordinates of all the boundary points is constructed on P_m. At last in the boundary re-decomposition procedure, P_m partitions the global array into n uniform portions and then evenly sends them to other processors according to their ranks. Thus each processor will obtain a same-size portion of the boundary points and then could perform the computations for $\mathbf{\Phi}_{b,c}$ in parallel without any load imbalance.

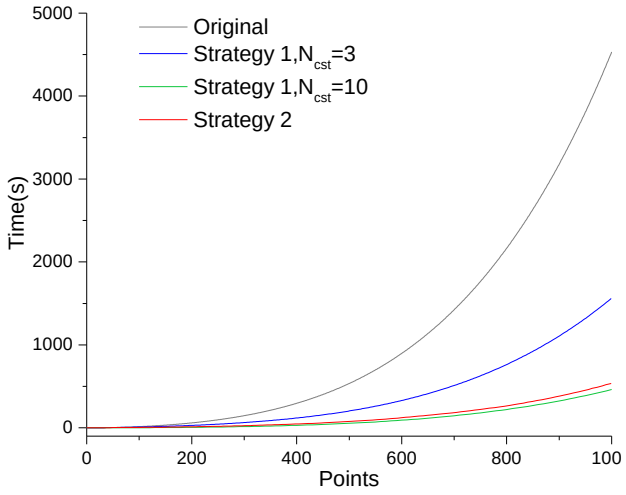

Figure 5: Greedy time cost vs. number of selected points.

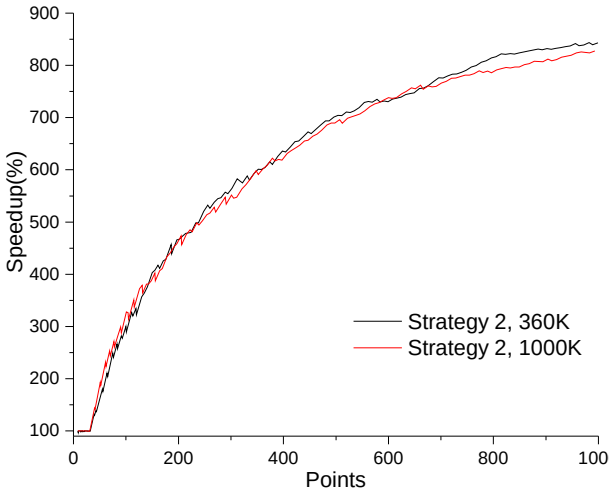

Figure 6: Speedup of 360×10^3 and 10^6 cell meshes.

To assess the effectiveness and efficiency of our method, a $3D$ moving fish test case is considered. As illustrated in Fig. 4, a fish with length L is undulating in the middle of a $20L \times 5L \times 5L$ cubic tank. In terms of the convergence speed, Fig. 5 plots the greedy time as a function of the number of selected control points. We could see as N_c increases, the cost of the original greedy method is in a very fast exponential growth. For our multi-selection method, using Strategy 1 with $N_{cst} = 3$, the cost could obtain a reduction of more than 50%. Further more, using either Strategy 1 with $N_{cst} = 10$ or Strategy 2, the cost could be reduced almost to a linear growth. This verifies the high efficiency of our multi-selection method. Comparison of Strategy 2 applied on different meshes is illustrated in Fig. 6. It can be seen that the speedup of Strategy 2 is mesh independent. For both two meshes, as the number of selected points increases, the speedup will

accordingly grow up and gradually approach to a limit value. Fig. 8 illustrates with contours the manner in which the surface errors decay as more points are added using Strategy 2. Typically, it is possible to achieve maximum errors less than 0.025% of the undulating amplitude($0.125L$) with 683 control points on the 10^6 cell meshes.

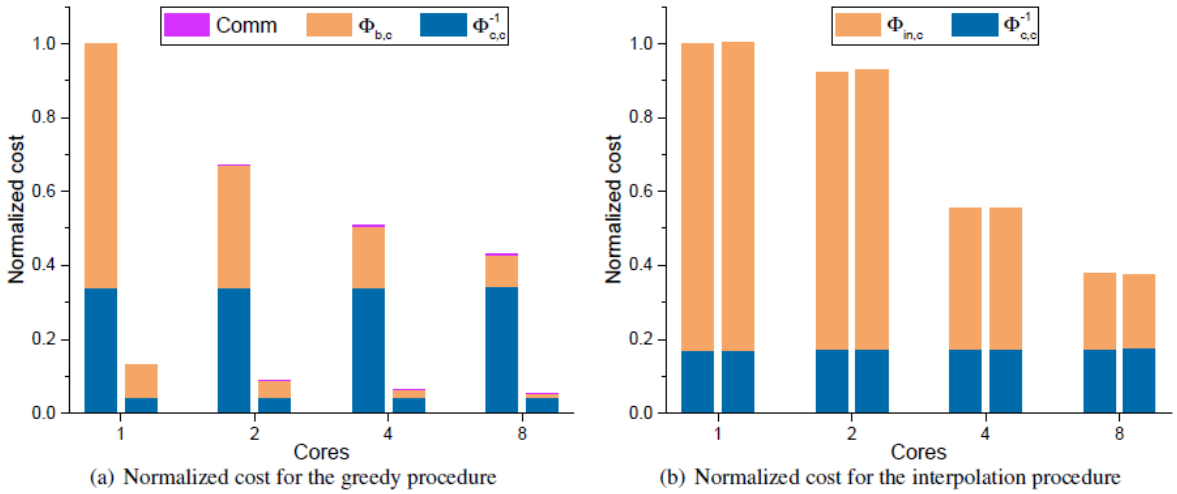

Figure 7: Time Cost versus degree of parallelism for each part of the RBF mesh deformation using the original greedy method(left columns) and the multi-selection greedy method(right columns) on the 10^6 cell mesh with error tolerance of 1e–4.

The time cost of each part of the RBF based mesh deformation procedure on the 10^6 cell mesh is illustrated against the degree of parallelism in Fig. 7. For the greedy procedure, the cost is reduced completely due to the reduction of $\boldsymbol{\Phi}_{b,c}$, while $\boldsymbol{\Phi}^{-1}_{c,c}$ (a matrix calculation that could not be parallelized) remains unchanged at all parallelism. In addition, as the parallelism increases, the communication cost will increase and thus lead to a decrease of the parallel efficiency. Similarly for the interpolation procedure, the cost is reduced completely due to the reduction of $\boldsymbol{\Phi}_{in,c}$ and $\boldsymbol{\Phi}^{-1}_{c,c}$ also remains the same at all parallelism. As shown in Fig. 9, the total speedup of Strategy 2 for both meshes is illustrated against the degree of parallelism. The parallel speedup using the original greedy algorithm is also plotted as a comparison. We could see using our multi-selection method at parallelism of 72, a total speedup near to 12 and 20 will be obtained respectively for the 360×10^3 cell mesh and the 10^6 cell mesh at last.

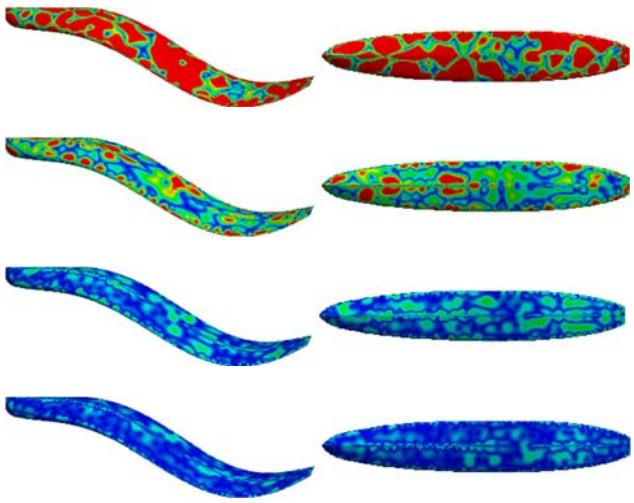

Figure 8: Evolution of the surface errors on the 10^6 cell mesh.

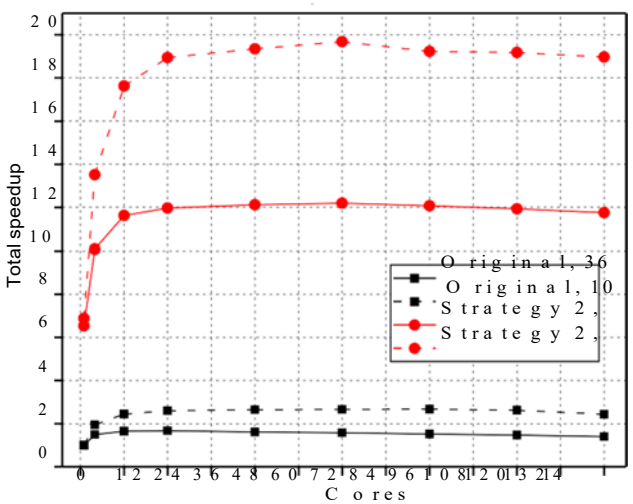

Figure 9: Total speedup vs. degree of parallelism.

References

[1]T. C. Rendall and C. B. Allen, "Efficient mesh motion using radial basis functions with data reduction algorithms," *Journal of Computational Physics*, vol. 228, no. 17, pp. 6231–6249, 2009.