Incremental approach for radial basis functions mesh deformation with greedy algorithm

Mohamed M. Selim a, b, *, Roy P. Koomullil a, Ahmed S. Shehata b, c

a Mechanical Engineering Department, University of Alabama at Birmingham, Birmingham, AL, 35294, USA
b College of Engineering, Arab Academy for Science and Technology and Maritime Transport, Alexandria, B.O. Box 1029, Egypt
c Department of Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, Glasgow G4 0LZ, UK

ABSTRACT

Mesh Deformation is an important element of any fluid–structure interaction simulation. In this article, a new methodology is presented for the deformation of volume meshes using incremental radial basis function (RBF) based interpolation. A greedy algorithm is used to select a small subset of the surface nodes iteratively. Two incremental approaches are introduced to solve the RBF system of equations: 1) block matrix inversion based approach and 2) modified LU decomposition approach. The use of incremental approach decreased the computational complexity of solving the system of equations within each greedy algorithm’s iteration from O(n^2) to O(n^2). Results are presented from an accuracy study using specified deformations on a 2D surface. Mesh deformations for bending and twisting of a 3D rectangular supercritical wing have been demonstrated. Outcomes showed the incremental approaches reduce the CPU time up to 67% as compared to a traditional RBF matrix solver. Finally, the proposed mesh deformation approach was integrated within a fluid–structure interaction solver for investigating a flow induced cantilever beam vibration.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Many dynamic computational simulations such as Fluid Structure Interaction (FSI), control surface deflections, and aerodynamic geometries optimization involve moving/deforming meshes [1–5]. A critical step in the analysis of this class of problems is a successful mesh deformation due to structural deformations. For an accurate simulation, the fluid volume mesh needs to move conformal to the structure, with very little degradation in quality. Due to the repeatability of the mesh deformation and the large number of fluid cells, an efficient and reliable approach is needed for a successful analysis. In addition, the fluid mesh is typically partitioned and handled by different processors. This necessitates the mesh deformation algorithms to be parallelizable for efficient simulations. The main complexity of the moving mesh approach is to find an optimum technique that is suitable for different mesh topologies and physical situations. At the same time, it should preserve, as much as possible, the quality of the mesh while keeping the computational cost low. The objective of this study is to investigate the feasibility of using radial basis function (RBF) based interpolation technique with greedy algorithm to deform large-scale generalized meshes and to improve the traditional greedy algorithm using an incremental approach.

* Corresponding author at: Mechanical Engineering Department, University of Alabama at Birmingham, Birmingham, AL, 35294, USA.
E-mail address: mselim@uab.edu (M.M. Selim).

http://dx.doi.org/10.1016/j.jcp.2017.03.037
0021-9991/ © 2017 Elsevier Inc. All rights reserved.
Strategies for deforming the fluid mesh conforming to the deformation of structure can be divided into two basic classes: physical analogy and interpolation. The most popular algorithms of each of these classes are briefly discussed below.

The physical analogy approach describes the fluid mesh deformation according to a physical process that can be modeled using numerical methods. One of the popular methods in this class is the tension spring analogy developed by Batina [6]. In this approach, each edge of the mesh is replaced by a tension spring with the spring stiffness inversely proportional to the edge length. The drawback of this approach, especially for fine meshes or high amplitude movements, is mesh crossing. One of the improvements to this approach is the torsional spring approach introduced by Farhat et al. [7], in which fluid mesh is considered as a network of torsional springs added at the nodes to prevent cell collapse and to allow cell rotation. There are modifications reported in the literature in which the mesh is considered as a combination of tension and torsion springs and also the stiffness of the springs is varied locally based on mesh cell size or proximity to the deforming surface. The main drawback of these methods is that they involve large systems of equations, implying a higher computational cost. Besides, these methods require grid connectivity information which results in more storage requirements and difficulties in parallelization.

A later advancement under the physical analogy approach is the use of a multidimensional linear elasticity analogy [8], in which the mesh is interpreted as a continuous elastic medium. The modulus of elasticity is chosen as inversely proportional to the cell volume or to the distance from the deforming boundary. In this approach, each displacement component of a mesh movement is governed by a partial differential equation, such as Laplacian equation [9]. Further details about different mesh deformation techniques can be found in [10].

On the other hand, the interpolation analogy uses an interpolation function to transfer prescribed boundary point displacements to the fluid mesh. In general, these schemes do not require connectivity information. Therefore, these algorithms can be applied to arbitrary mesh types that contain general polyhedral elements or hanging nodes [11].

Recently, a novel interpolation based scheme has been developed by Luke [11]. In this scheme, the deformation of the volume mesh is viewed as a projection that is independent of the surface deformation into the volume. Using a tree-code optimization, the algorithm cost is demonstrated to be \( O(n \log(n)) \), where \( n \) is the total number of nodes in the simulation, with mesh quality that is competitive to the radial basis function (RBF) scheme.

McDaniel and Morton [12] developed a technique that is based on a two-pronged approach where the viscous layers of nodes are deformed rigidly and the outer region is deformed with two different interpolation techniques. Several different rigid deformation schemes were investigated by the authors. However, the results showed that the best performing scheme was based on a semi-rigid connection to the owner surface nodes defined as part of the mesh parsing, which provided smoother deformation in convex regions. The last layer of the viscous region was used as the deforming boundary surface for the outer region deformation.

The radial basis function interpolation method, such as the method developed by Boer et al. [13,14], is one of the promising interpolation schemes. RBFs have become a well-established tool to interpolate scattered data. RBF can also be used as an interpolation function to transfer the displacements known at the boundaries of the structural mesh to the fluid mesh. This scheme produces high-quality meshes with reasonable orthogonality preservation near deforming boundaries [14]. Other advantages of RBF include: 1) Avoiding the need for mesh connectivity information, 2) the system of equations which needs to be solved is linear, and 3) the size of the linear system of equation is proportional to the number of boundary nodes, not all fluid nodes. Moreover, many studies have investigated different techniques for improving RBF based mesh deformation. The most influential study was made by Rendall and Allen [15]. They proposed the use of data reduction algorithm along with RBF interpolation. This technique will be discussed in the following sections. Another study, which builds on Rendall and Allen [15], is the work by Sheng and Allen [16], which put forward specific criteria for selecting the nodes involved in the interpolation. Michler [17] proposed a confinement technique that restricts the mesh deformation to the surrounding region of the moving surface. He achieved this by assigning an auxiliary geometry encompassing the region targeted by the interpolation, instead of using a cut-off function. Moreover, Michler [17] proposed to choose different
The following influence (1) to sum 2. incremental a currently method centers 558 M.M. Selim et al. / Journal of Computational Physics 340 (2017) 556–574 robust, formulated. Where \( \phi \) functions Two the \( \{ \) RBFs \( \) compact \( \} \) support. Examples of different RBFs types.

<table>
<thead>
<tr>
<th>No.</th>
<th>Function ( \phi )</th>
<th>Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((1 - \xi)^2)</td>
<td>Compact support</td>
<td>( \xi = x/r )</td>
</tr>
<tr>
<td>2</td>
<td>((1 - \xi)^4(4\xi + 1))</td>
<td>Compact support</td>
<td>( \xi = x/r )</td>
</tr>
<tr>
<td>3</td>
<td>((1 - \xi)^p(35/3\xi^2 + 6\xi + 1))</td>
<td>Compact support</td>
<td>( \xi = x/r )</td>
</tr>
<tr>
<td>4</td>
<td>(e^{-\xi^2})</td>
<td>Global support</td>
<td>Gaussian</td>
</tr>
<tr>
<td>5</td>
<td>(x^2 \log(x))</td>
<td>Global support</td>
<td>Thin plate spline</td>
</tr>
<tr>
<td>6</td>
<td>(1 + x^2)</td>
<td>Global support</td>
<td>Quadric biharmonics</td>
</tr>
</tbody>
</table>

centers for different directions instead of using the same set of centers for all displacement directions, which results in reducing the required CPU time. Wang et al. [18] proposed a hybrid method that combines the Delaunay graph mapping method with RBF interpolation. They showed that this method is capable of handling extremely large deformations without compromising the mesh quality. In this study, a novel incremental methodology for solving the RBF system of equations is formulated. Two different incremental approaches are formulated and compared against the RBF method that is being currently used. As will be presented in this article, the proposed approaches decrease the CPU time significantly and exhibit a robust performance. The following sections present the formulation of the RBF interpolation and explain the proposed incremental solving methodology.

2. RBF formulation

Using RBF, the interpolation function, \( S \), describing the displacement in the whole domain can be approximated by a sum of basis functions as [19],

\[
S(X) = \sum_{j=1}^{n_b} \alpha_j \phi(||X-X_{bj}||) + P(X) \tag{1}
\]

where \( X_{bj} = [x_{bj}, y_{bj}, z_{bj}] \) are the boundary nodes in which the deformations are known and these are called the centers for RBF, \( P \) is a polynomial, \( n_b \) is the number of boundary nodes, and \( \phi \) is the selected basis function with respect to the Euclidean distance \( ||x|| \). The coefficients \( \alpha_j \) and the polynomial \( P \) are determined by the interpolation conditions

\[
S(X_{bj}) = d_{bj} \tag{2}
\]

\[
\sum_{j=1}^{n_b} \sum_{j=1}^{n_b} \alpha_j x_j = \sum_{j=1}^{n_b} \sum_{j=1}^{n_b} \alpha_j y_j = \sum_{j=1}^{n_b} \sum_{j=1}^{n_b} \alpha_j z_j = 0 \tag{3}
\]

Equation (3) ensures that the polynomial \( P \) is coinciding with the interpolant \( S \), by forcing the polynomial in equation (1) to zero when \( X = X_{bj} \) [20].

The values for the coefficients \( \alpha_j \) and the linear polynomial coefficients \( \beta_j \) can be obtained by solving the system

\[
\begin{bmatrix}
\phi_{b,b} & p^T \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} =
\begin{bmatrix}
d_b \\
0
\end{bmatrix} \tag{4}
\]

Where \( \alpha \) is a vector containing the coefficients \( \alpha_j \), \( \beta \) is a vector containing the coefficients of the linear polynomial \( \beta_j \), \( \phi_{b,b} \) is an \( n_b \times n_b \) matrix containing the evaluation of the basis function \( \phi_{b,b} = \phi(||X_{bj} - X_{bj}||) \), and \( p \) is an \( n_b \times 4 \) matrix with row \( j \) given by \( [1 x_{bj}, y_{bj}, z_{bj}] \) [13,14].

In this study, the polynomial \( P \) was omitted, since it was concluded in previous studies that it does not have a large influence on the quality of the deformed mesh [21]. In this case, the system of equations in (4) will be simplified as the following

\[
[\phi_{b,b}][\alpha] = [d_b] \tag{5}
\]

Generally, RBFs can be divided into two groups: 1) functions with compact support, and 2) functions with global support. The main difference between the two types is that the function with compact support is scaled with a support radius \( r \) to control the extent of influence of the basis function. Functions with global support cover the entire interpolation space, which leads to dense matrix systems. Table 1 lists some RBFs for both these types [13,14,22].

Functions with compact support are forced to satisfy the following condition

\[
\phi_{b,b} = \begin{cases} 
\phi(||X_{bj} - X_{bj}||) & \text{if } 0 \leq \xi \leq 1 \\
0 & \text{if } \xi > 1
\end{cases} \tag{6}
\]
2.1. Direct RBF scheme limitations and improvements

The radial basis function interpolation method produces high-quality meshes with good orthogonality preservation near deforming boundaries. On the other hand, in its most straightforward implementation, it is too costly to be used for large 3-D problems. A direct solution of such systems requires $O(n^3)$ operations and $O(n^2)$ memory usage which becomes prohibitive for more than a few thousand data points. Great progress has been made in recent years towards alleviating this computational burden.

An approximation algorithm for RBF based mesh deformation has been suggested by Rendall and Allen [15]. In this algorithm, the RBF is applied using a coarsened subset of the surface mesh. Displacements of the omitted surface nodes are calculated using the interpolation method and the error is calculated as the difference between the interpolated values and the actual displacement. A greedy algorithm is used to add points that have the largest error. Rendall and Allen report that this algorithm improves the performance of the RBF method by approximately two orders of magnitude. The use of the greedy algorithm reduces the cost remarkably without compromising the accuracy.

The presented approach is an extension of the work done by Rendall and Allen [15] by utilizing incremental approaches to solve the system of equations resulting from RBF formulation. This study takes advantage of solving similar systems of equations within each iteration by using an incremental approach [23] to reduce the CPU time by avoiding the use of any expensive linear system solver such as full LU decomposition. Two incremental approaches will be presented and compared in this article: 1) Matrix inversion based approach, and 2) Incremental LU decomposition based approach. The proposed method does not require any pre-conditioning, since it is not affected by the matrix’s condition number, which makes it very robust and suitable for any compact support RBF. The procedure of picking the centers using the greedy algorithm and the use of the incremental approaches are described in the following sections.

3. Greedy algorithm

A direct RBF method would use all boundary nodes as interpolation centers. For large-scale unstructured-grid problems, the number of boundary nodes is usually of the order of $10^4$ to $10^5$, therefore calculating the interpolation weight coefficients tend to be very costly. Moreover, if the mesh is required to be deformed at each time step, direct RBF method cost will be impractical. It was reported in the literature that to achieve high mesh deformation accuracy, only a small subset of the total boundary nodes is sufficient [15–17].

Centers selection is highly dependent on the shape of the geometry, displacement vector, and the support radius in the case of compact support RBFs. Therefore, there is no direct way to predict which nodes to be selected in advance. Hence, a greedy algorithm is used to iteratively select these centers. Initially, two boundary nodes are selected during the first iteration. In each subsequent iteration, the selected centers are used to predict the deformation of the unselected centers. Based on this prediction, the error is calculated at all boundary points, which is the difference between the actual and the predicted deformations. The greedy algorithm is stopped if the maximum error is below a specified tolerance. Otherwise, the node with the maximum error is added to the selected subset, and the iteration continues. Sheng and Allen [16] investigated the possibility of selecting the centers based only on the geometry. Thus, the greedy algorithm is used only once and the same subset of centers is used for all time steps for different deformations. A typical greedy algorithm procedure is expressed in Algorithm 1.

Algorithm 1

\begin{algorithm}
\begin{procedure}[H]
\caption{Greedy_RBF}
\begin{algorithmic}
\Procedure{Greedy_RBF}{} \EndProcedure
\State Read in real deformation for all boundary nodes \State Choose an initial subset (two nodes) to begin with \While{\text{Maximum Error $>$ Tolerance}} \State Calculate $\phi_{s,s}$ and $\phi_{u,s}$ \State Solve equation (5) \State Evaluate deformation for all unselected centers \State Calculate the error \State Select node with largest error and add it to selected centers list \EndWhile \State End Do \State Evaluate volume deformation using only reduced subset of centers \State Update volume mesh
\end{algorithmic}
\end{procedure}
\end{algorithm}

\begin{align*}
\text{where } \phi_{s,j} &= \phi(\|X_{\text{selected}} - X_j\|) \text{ and } \phi_{u,j} = \phi(\|X_{\text{unselected}} - X_j\|). \\
\text{Assume the greedy algorithm starts with } n \text{ selected centers, equation (5) becomes} \\
\begin{bmatrix}
\phi_{1,1} & \cdots & \phi_{1,n} \\
\vdots & \ddots & \vdots \\
\phi_{n,1} & \cdots & \phi_{n,n}
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_n
\end{bmatrix}
= 
\begin{bmatrix}
d_1 \\
\vdots \\
d_n
\end{bmatrix}
\end{align*}

\begin{align*}
\begin{bmatrix}
\phi_n \\
\vdots \\
\phi_n
\end{bmatrix}
\begin{bmatrix}
\alpha_n
\end{bmatrix}
= 
\begin{bmatrix}
d_{b_n}
\end{bmatrix}
\end{align*}

Then for the next iteration, the node with the largest error will be added to the list, and the system will become

\[
\begin{bmatrix}
\phi_{1,1} & \ldots & \phi_{1,n} & \phi_{1,n+1} \\
\vdots & \ddots & \vdots & \vdots \\
\phi_{n,1} & \ldots & \phi_{n,n} & \phi_{n,n+1} \\
\phi_{n+1,1} & \ldots & \phi_{n+1,n} & \phi_{n+1,n+1}
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_n \\
\alpha_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
d_1 \\
\vdots \\
d_n \\
d_{n+1}
\end{bmatrix}
\] (9)

Or

\[
[\phi_{n+1}] [\alpha_{n+1}] = [d_{n+1}] 
\] (10)

Note that

\[
[\phi_{n+1}] = 
\begin{bmatrix}
\phi_n & \phi_{i,n+1} \\
\phi_{n+1,i} & \phi_{n+1,n+1}
\end{bmatrix}
\] (11)

where \(i = [1, 2, \ldots, n]\).

For compact support RBFs, the matrix \(\phi_{b,b}\) is symmetrical with ones in the diagonal. Therefore, the system can be simplified as

\[
\begin{bmatrix}
\phi_n & \phi_{\text{add}} \\
\phi_{\text{add}}^T & 1
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_n \\
\alpha_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
d_1 \\
\vdots \\
d_n \\
d_{n+1}
\end{bmatrix}
\] (12)

where \(\phi_{\text{add}}\) is a vector that contains the RBF evaluations for the newly added node with respect to each selected center.

The following sections describe the proposed approaches for solving equation (5) incrementally.

3.1. Incremental matrix inversion based approach

The most straightforward technique for solving the above system is by calculating the inverse of the coefficients matrix. Since the coefficient matrix consists of the matrix from the previous iteration with the addition of one row and one column, it is possible to use an incremental approach to compute the inverse using the inverse from the previous iteration.

The inverse of a general block matrix can be calculated as [24]

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}^{-1} = 
\begin{bmatrix}
(A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\
-(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1}
\end{bmatrix}
\] (13)

Applying this principle to the calculation of the inverse of the coefficient matrix in Equation (11) will result in

\[
[\phi_{n+1}]^{-1} = 
\begin{bmatrix}
\phi_n & \phi_{\text{add}} \\
\phi_{\text{add}}^T & 1
\end{bmatrix}^{-1} = 
\begin{bmatrix}
(\phi_n - \phi_{\text{add}}\phi_{\text{add}}^T)^{-1} & -\frac{1}{k}\phi_n^{-1}\phi_{\text{add}} \\
1\phi_{\text{add}}\phi_n^{-1} & \frac{1}{k}
\end{bmatrix}
\] (14)

where \(k = 1 - \phi_{\text{add}}^T\phi_n^{-1}\phi_{\text{add}}\).

Since the matrix \(\phi_n\) is symmetrical, the previous equation can be simplified as

\[
[\phi_{n+1}]^{-1} = \frac{1}{k} 
\begin{bmatrix}
k\phi_n^{-1} + \xi\xi^T & -\xi \\
-\xi^T & 1
\end{bmatrix}
\] (15)

where \(\xi = \phi_n^{-1}\phi_{\text{add}}\).

This step has \(O(n^2)\) complexity [23] instead of \(O(n^3)\) complexity required by Gauss elimination or LU decomposition [25]. The matrix system resulting from RBFs based formulation is highly ill-conditioned [16,26], which makes it difficult to use traditional iterative matrix solvers. The current incremental approach is independent of the condition number of the matrix. As shown in the following section, it requires less computational time as compared to LU decomposition. Moreover, this step is repeated for every iteration until the greedy algorithm converges, which makes the complexity reduction more significant. However, it must be noted here that during the first greedy algorithm iteration, the inverse of the matrix must be computed using any traditional method. At this specific iteration, the system is very small in size, typically \(2 \times 2\), which is very feasible to solve.

3.2. Incremental LU decomposition based approach

LU decomposition of the matrix \(\phi_{n+1}\) defined in equation (12) can be calculated incrementally using the LU decomposition of the matrix \(\phi_n\). This can be performed by the addition of one row and one column to the LU decomposition of the
matrix $\phi_h$, as shown in equation (16). In this case, performing full LU factorization is needed only once at the first iteration. For all subsequent iterations an incremental LU decomposition will be used.

\[
[LU_{n+1}] = \begin{bmatrix}
    u_{1,1} & \cdots & u_{1,n+1} \\
    \vdots & \ddots & \vdots \\
    l_{1,n+1} & \cdots & u_{n,n+1}
\end{bmatrix} = \begin{bmatrix}
    [LU_n] & u_{1,n+1} \\
    \vdots & \vdots \\
    l_{1,n+1} & l_{n,n+1} & u_{n,n+1}
\end{bmatrix}
\]

(16)

The incremental LU decomposition procedure requires only two nested do loops to perform the decomposition instead of three nested do loops which are regularly required by the full LU decomposition. The resulted $LU_{n+1}$ from this code is used, alongside with $d_{b_0}$, to calculate the interpolation weight coefficients using forward and backward substitutions.

4. Results and discussion

Several benchmark test cases have been selected to study the performance of the proposed approaches. These test cases vary in mesh topology, mesh size, and deformation shape and magnitude. First, a 2D mesh surface with different mesh refinements has been tested to be deformed into different shape functions. Second, a 3D Rectangular Supercritical Wing (RSW) has been tested for several wing bending deflection magnitudes. Third, three different mesh refinements for the RSW case with mesh sizes up to 8 million cells have been tested for bending-twisting wing deformation. Finally, a complete FSI problem for the flow-induced cantilever beam vibration case has been demonstrated.

4.1. Two-dimensional shape functions

To investigate the accuracy and efficiency of the presented method, four analytical test functions were chosen to deform a structured mesh. A mesh refinement study is also conducted to analyze influence of the total number of nodes on the performance of the algorithm. Three different structured meshes used for this study are shown in Fig. 1. The $x$- and $y$-coordinates of the computational domain varies from $-1.5$ to $1.5$. The $z$-coordinate of the computational domain is specified as zero before the deformation and it is specified as the analytic functions after the deformation. The specified deformations of the mesh for this study are listed below.

\[
F1(x, y) = (1 + 9x^2 + 16y^2)^{-1}
\]

(17)

\[
F2(x, y) = 1 - ((x^2 + y^2)/2)^{1/2}
\]

(18)

\[
F3(x, y) = 1.5xe^{-x^2-y^2}
\]

(19)

\[
F4(x, y) = (1.25 + \cos(5.4y))/(6 + 6(3x - 1)^2)
\]

(20)

The displacement at each node is pre-calculated using the above functions, and two random nodes are selected as initial centers for the RBFs. The greedy algorithm uses the initial centers to calculate the RBF weight coefficients and evaluate the displacement for all the remaining nodes. The error is calculated as the difference between the pre-calculated displacement and the evaluated displacement to check the stopping criteria for the greedy algorithm. If the maximum error is greater than the specified tolerance, the node with the maximum error is added to the list of centers for RBF. If the maximum error is less than the specified tolerance, the greedy algorithm stops the iteration and proceeds with deformation of the fluid mesh using the selected RBF centers.
Fig. 2. Computational cost percentages of greedy algorithms steps for different shape functions using full LU decomposition for element size \( h = 0.05 \).

Table 2
Time comparison between RBF with greedy algorithm using full LU decomposition and incremental approaches for element size \( h = 0.05 \).

<table>
<thead>
<tr>
<th>Analytical shape function</th>
<th>Centers percentage</th>
<th>Greedy RBF (tolerance = ( 5 \times 10^{-4} ))</th>
<th>Time saving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CPU time (seconds)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Full LU</td>
<td>Matrix inv.</td>
</tr>
<tr>
<td>F1</td>
<td>6.9%</td>
<td>0.92</td>
<td>0.4</td>
</tr>
<tr>
<td>F2</td>
<td>11.5%</td>
<td>4.84</td>
<td>1.57</td>
</tr>
<tr>
<td>F3</td>
<td>3.5%</td>
<td>0.32</td>
<td>0.13</td>
</tr>
<tr>
<td>F4</td>
<td>5.6%</td>
<td>0.82</td>
<td>0.31</td>
</tr>
</tbody>
</table>

From a computational cost point of view, the greedy algorithm consists of three main steps:

Step 1: Evaluating the RBF between the nodes and forming \( \phi_{s} \) & \( \phi_{u} \)

Step 2: Solving equation (5) for the weight coefficients

Step 3: Evaluating the interpolated deformations for all unselected centers

Using conventional methods for solving equation (5) makes Step 2 the most expensive step within the greedy loop. Fig. 2 illustrates the computational cost of each step as a percentage of the total CPU time of the greedy algorithm for the different shape functions deformations when using Full LU decomposition. In this specific case, Step 2 formed approximately an average of 70% of the total CPU time of the greedy algorithm. Thus, reducing the cost of solving equation (5) would highly improve the greedy algorithm’s performance, which the proposed method aims to.

To assess the enhancement gained by using the greedy algorithm with incremental approach, the performance of the proposed algorithms were compared against the use of a full LU decomposition. In this test case, the compact support RBF \( \psi = (1 - \xi)^{3}(4\xi + 1) \) was used with a predefined error tolerance of \( 5 \times 10^{-4} \). Table 2 shows the results of this comparison, for the case of element spacing \( h = 0.05 \), which clearly illustrates the superiority of the use of greedy algorithm along with incremental approach. Shape function F2 requires picking relatively a higher number of centers, to achieve the predefined tolerance, compared to other shape functions. This leads to perform more iterations for the greedy algorithm, which consequently leads to a longer CPU time especially for the full LU decomposition case.

The number of selected centers and total number of nodes on the surface for three meshes with different resolution and for four different deformations are compared in Fig. 3. In these calculations \( 1.0 \times 10^{-3} \) is set as the tolerance criteria for stopping the greedy algorithm and a support radius of 1.5 units is used. It can be seen from the figure that the mesh refinement did not cause the number of selected centers to increase. The number of centers required for each of the analytical deformation remains more or less the same irrespective of the number of surface nodes. However, the number of selected nodes varies with the analytical functions used for the deformation. This confirms that the number of selected centers is a function of the type of deformation.

The specified four different analytic deformations used for this study. The selected RBF centers for the interpolation for three different meshes are shown in Fig. 4. In this figure the color contour is specified based on the deformation. It can be seen from the figure that the pattern of the selected RBF centers are the same for all different mesh resolutions.

Fig. 5 shows the deformed mesh shaded with the absolute error for a specified error tolerance of \( 1 \times 10^{-4} \) for all four analytical benchmark deformations. This figure shows the error is smaller in the vicinity of the selected nodes and relatively higher near the unselected nodes.

A study is also conducted to analyze the effect of error tolerance on the number of selected centers and the CPU time required for the selection of appropriate centers. The fine mesh \( h = 0.025 \) from the mesh refinement study is used for this analysis. The number of selected centers and CPU time requirements were calculated for error tolerances of \( 1 \times 10^{-3} \), \( 5 \times 10^{-4} \), and \( 1 \times 10^{-4} \). As expected, the finer the specified tolerance, the higher the number of centers picked and longer
Fig. 3. Total number of nodes versus selected number of centers for the three different mesh refinements: (a) $h = 0.1$, (b) $h = 0.05$, and (c) $h = 0.025$.

Fig. 4. Original surface mesh and selected centers (top view) for coarse, medium, and fine meshes.

the CPU time required. Fig. 6 shows the variation of CPU time and number of selected centers versus the specified tolerance. It is noticeable from the chart that the number of selected centers and the CPU time for the greedy algorithm increases with the decrease in error tolerance. To determine the dependency of the total number of nodes ($N$) on the interpolation error, the Root Mean Square (RMS) error is plotted against the number of nodes using a logarithmic scale in Fig. 7. It can
Fig. 5. Deformed mesh shaded by the absolute error.

Fig. 6. Tolerance versus number of centers (left), tolerance vs. CPU time (right).
be seen from the figure that all different analytical functions for deformation showed an approximate slope of 0.4 or less, indicating that the total number of nodes has a minimal effect on the interpolation error.

4.2. Rectangular Supercritical Wing (RSW)

To assess the proposed approaches on a real-life FSI application, the computational time savings for the presented RBF interpolation with greedy algorithm and incremental approach, and traditional RBF interpolation with greedy algorithm are compared using the rectangular supercritical wing. RSW test case was used at the Aeroelastic Prediction Workshop sponsored by the Structural Dynamics Technical Committee, American Institute of Aeronautics and Astronautics (AIAA) [27, 28]. This wing has a span of 48 inches and root chord length of 24 inches. The mesh used for this analysis is one of the meshes provided for the Aeroelastic Prediction Workshop and is available to download from the workshop website.¹ A fully tetrahedral mesh was selected for this test case as shown in Fig. 8. The definition used for the skewness results in approximately 34% of the total number of elements having a skewness of 1.0 due to the highly skewed tetrahedral elements within the boundary layer; see Fig. 9. The fluid domain has the dimensions of $4824 \times 2400 \times 4800$. The mesh

¹ https://c3.nasa.gov/dashlink/static/media/other/RSW.htm.
consists of 17,453,792 cells, 2,944,006 nodes, and 56,272 boundary nodes. A direct RBF interpolation technique will require operations of the order of \((56,272)^3\) to evaluate the weighting coefficients and operations of the order of \(2,944,006 \times 56,272\) to evaluate the deformation for all interior nodes. In addition, this process will be repeated for each time step, which will make it impractical for FSI analysis.

The RSW structure is assumed to be fixed at the root of the wing and deformed using sinusoidal shape function. Three deformation angles are tested \(1^\circ, 3.5^\circ,\) and \(5^\circ\) to assess the performance of the presented approach. In these simulations, a specified tolerance of \(5 \times 10^{-4}\) for the greedy algorithm and a support radius of 1200 units are used. The support radius was calculated as half the distance between the furthest fluid node and any boundary node.

Fig. 10 shows no noticeable change in the skewness before and after the deformation. However, to get a better understanding about the effect of the magnitude of boundary deformation on the mesh quality, the percentage change of number of elements with respect to the skewness before and after the deformation is also plotted in Fig. 10. From this graph, it is evident that the new distribution of skewness is within an acceptable range and the number of elements that became highly skewed after the deformation is below 0.1\% for 3.5 degrees deformation and below 0.2\% for 5 degrees deformation. Furthermore, all the deformed meshes have been tested and neither negative volume cells nor cells degeneration has occurred. Pictorial views of the mesh after deformation are shown in Fig. 11.

The computational costs and number of selected centers for the full LU decomposition method compared against the incremental matrix inversion and LU decomposition based approaches are compared in Table 3. The reason of using LU decomposition method as the base method for comparison instead of using an iterative method is that the matrix system is highly ill-conditioned and convergence issues were reported in the literature [16,26]. It is clear that the number of selected centers and accordingly the corresponding computational time increases for larger deformations. However, it can be noted from the results that the time required for the greedy algorithm with incremental approaches is significantly smaller than that of performing a full LU decomposition. Moreover, the matrix inversion based approach is found to have slightly better performance over LU decomposition based approach for all cases. Also the superiority of the incremental approaches over the full LU decomposition increases for larger deformations and accordingly for larger meshes, as illustrated by the increase of the computational time saving with the increase of the deformation angle.

### 4.3. Bending and twisting RSW deformation

The computational time savings for hybrid mesh topologies are demonstrated using a combined bending and twisting deformations of the RSW case. Three different meshes were used for this study. The count of boundary and interior nodes and the count of cells for each mesh are given in Table 4. These meshes were also attained from the Aeroelastic Prediction Workshop website. Fig. 12 illustrates the different refinements for the three meshes by visualizing a part of the side wall around the wing.

The deformation of the wing is specified as a combination of a vertical bending displacement and a twist using the following formula

![Fig. 9. Highly skewed elements within the boundary layer: a) mounting wall normal view, b) zoom into the wing leading edge, and c) zoom into the wing trailing edge.](image)
Fig. 10. (a) Mesh skewness versus number of elements and (b) mesh skewness versus percentage change in number of elements.

Table 3
Number of selected centers and CPU time in seconds for different deformation angles.

<table>
<thead>
<tr>
<th>Deformation angle</th>
<th>Tolerance</th>
<th>No. of centers</th>
<th>Greedy algorithm time</th>
<th>Total saving %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Full LU</td>
<td>Matrix inv.</td>
</tr>
<tr>
<td>1.0°</td>
<td>5.0E-4</td>
<td>285</td>
<td>10.31</td>
<td>5.29</td>
</tr>
<tr>
<td>3.5°</td>
<td>5.0E-4</td>
<td>570</td>
<td>39.34</td>
<td>18.80</td>
</tr>
<tr>
<td>5.0°</td>
<td>5.0E-4</td>
<td>703</td>
<td>78.84</td>
<td>25.48</td>
</tr>
</tbody>
</table>

Table 4
Count of boundary and interior nodes for the three mesh refinements.

<table>
<thead>
<tr>
<th></th>
<th>No. of boundary nodes</th>
<th>No. of interior nodes</th>
<th>No. of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse</td>
<td>30,146</td>
<td>1,325,155</td>
<td>1,355,301</td>
</tr>
<tr>
<td>Medium</td>
<td>60,101</td>
<td>3,062,697</td>
<td>3,122,798</td>
</tr>
<tr>
<td>Fine</td>
<td>134,754</td>
<td>8,476,975</td>
<td>8,611,729</td>
</tr>
</tbody>
</table>
\[ \Delta z = \delta_0 \sin\left(\frac{y}{2b}\pi\right); \quad \delta_0 = 0.5c \quad (21) \]
\[ \varphi = \varphi_0 \sin\left(\frac{y}{2b}\pi\right); \quad \varphi_0 = 30^\circ \quad (22) \]

Where \( y \) is the coordinate along the span of the wing, \( z \) is the coordinate normal to the wing, \( b \) is the span of the wing, \( c \) is the root chord length, and \( \varphi \) is the twist angle. A comparison of the original and the deformed wings is shown in Fig. 13. This test case represents a real-application of FSI problem since it involves more complicated deformations with more than 130 thousands boundary nodes and 8.4 million interior nodes. The focus here will be directed toward determining which incremental approach provides better performance.

In this test case, the support radius is taken as four times the span of the wing (220 units). Therefore, the deformation is confined within 220 units away from all boundaries. The error tolerance for stopping the greedy algorithm is taken as \( 5.0 \times 10^{-4} \). A comparison of the original and the deformed meshes for the medium resolution of the discretized domain is shown in Fig. 14. Fig. 15 compares the CPU time required by the greedy algorithm using the matrix inversion and LU decomposition based approaches for each of the three meshes. For all meshes the matrix inversion based approach required slightly less CPU time and resulted in average time saving of 5.5% over LU decomposition based approach.

It should be mentioned that in some cases, due to round off error and slight variation in solvers accuracies, matrix inversion and LU decomposition based approaches do not pick exactly the same number of centers. However, the difference in the final number of selected centers is within the range of \( \pm 5 \) centers. Fig. 16 shows the convergence trends for the case of the fine mesh. The convergence trends for both algorithms are identical for all test cases. However, the variation in the
Fig. 13. Two different views of the original (blue) and deformed (red) wings. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 14. Comparison of original (blue) and deformed (red) meshes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 15. CPU time's comparison.

The total number of centers occurs at the end when a few centers are added to reach the predefined tolerance. Therefore, for ensuring fair time comparison; in such case both algorithms were forced to pick the same subset of centers with the same sequence.

Moreover, the matrix inversion based approach shows unstable behavior when reducing the error tolerance below $1 \times 10^{-4}$, unlike LU decomposition based approach, see Fig. 17. This instability is due to the accumulation of numerical errors for the matrix inverse calculation during the greedy iterations which results in error in the weight calculation. The matrix inverse at the current level is calculated as a function of the matrix inverse from the previous step through the calculation of $k$ and $\xi$ in equation (15). Therefore, any error in the calculation of the inverse in the previous step will be carried to the next level. Based on this observation, it is recommended to use the LU decomposition based approach for the greedy algorithm, even though the CPU time requirement is slightly higher.
4.4. Flow-induced cantilever beam vibration

To examine the proposed technique for FSI problem, the mesh deformation algorithm was integrated within a fluid–structure interaction solver. The case of a cantilever beam under the effect of incoming flow is investigated [29–33]. As shown in Fig. 18, the beam has a cross section area of 0.2 m × 0.2 m and a height of 2 m. The computational domain has the dimensions of 8 m × 1.2 m × 2.5 m and the beam is placed at 2 m from the inlet.

The fluid was assumed to be water with a density of 1000 kg/m³. The fluid flow was assumed to be inviscid and incompressible. Regarding the boundary conditions, all walls, including interface walls, were assigned as slip walls, thus no
boundary layer was needed. An inlet velocity boundary and an atmospheric pressure outlet boundary were used. An inlet velocity of 1.0 m/s have been assigned as the inlet condition. For the structural side, the beam was assumed to have a density of 1000 kg/m³ and a Poisson’s ratio of 0.3. A fixed zero displacement boundary condition is applied to the bottom surface of the beam, while all other beam surfaces are subject to FSI traction boundary condition. For these conditions, the cantilever beam vibration frequency was calculated to be 1 Hz and the simulation was run for three complete vibration cycles (t = 3 seconds). Details about structural and fluid solvers are available in [34,35].

Unstructured tetrahedral meshes has been used for both domains. Fig. 19 shows the generated surface meshes for both fluid and structural domains. The CFD mesh consists of 105,568 cells, 206,408 faces, and 19,164 nodes. The CSD mesh consists of 20,346 cells, 37,114 faces, and 5169 nodes. The surface meshes for both domains at the interface are identical.

An RBF support radius of 1.0 m has been used for this test case. However, regarding the error tolerance, it has been found that using a fixed error tolerance is not practical for such FSI analysis. Since the beam is expected to have a wide range of deflections, using a fixed error tolerance that suits the minimum deflection will cause the greedy algorithm to pick almost all boundary nodes when the deflection reaches its peak. This will lead to a very time consuming mesh deformation process. Instead, a relative error tolerance has been used in order to relate the error tolerance to the maximum deflection predicted at each time step. In this case, the error tolerance has been set as 0.1% of the maximum total beam deflection at every time step. At the same time, the minimum allowed error tolerance was set to $1 \times 10^{-6}$ in order to avoid expensive mesh deformation at the beginning of the vibration cycle.
Fig. 21. Velocity distribution of cutting planes normal to Y at different simulation times: (a) 0.0 seconds, (b) 0.2 seconds, (c) 0.3 seconds, and (d) 0.5 seconds.

The RBF based mesh deformation approach with incremental solver showed a very efficient and robust performance. Using a varying error tolerance, that is a percentage of the maximum deflection, improved the algorithm’s performance dramatically. For all cases, the total number of nodes in the deformed region is 19,164 and the surface mesh consists of 4730 points. Only an average of 470 surface points, i.e. around 10%, were selected to drive the volume mesh motion. A support radius of 1 m was selected, which gave a deforming region that was large enough to reasonably accommodate the motion, as shown in Fig. 20. It is noticeable in this figure that for higher beam deflections, the mesh cells located above the beam tend to become more skewed. This issue happened because the mesh is coarse on the top wall, and the gap between the top wall and the beam tip is relatively small. This can be avoided by refining the mesh at this region. Fig. 21 shows the velocity distributions around the cantilever beam at simulation times ranging between 0.0 and 0.5 seconds.

Fig. 22 shows the percentage of selected centers versus simulation time. It is noticeable that the number of selected centers increases up to 16% at the beginning of each cycle. Since the predicted deflections tend to be relatively small at the beginning of each cycle, the algorithm picks more nodes in order to achieve the error tolerance. However, limiting the minimum value of the error tolerance to be $1 \times 10^{-6}$ prevents the algorithm from keep adding more points at this stage of the deflection cycle. Moreover, the RBF interpolation error is calculated based on the known deflections of the unselected centers. The interpolation error percentage is shown in Fig. 22. It can be seen that the error is fixed at 0.1%.

Fig. 23 shows the mesh deformation CPU time versus simulation time for both incremental approaches. It is noticeable that both approaches exhibited an efficient performance with an average CPU time of 2 seconds. Both approaches showed very similar performances.
5. Conclusion

An efficient mesh deformation technique using radial basis functions, by combining the advantages of merging the use of both greedy algorithm and incremental approach, for fluid–structure interaction simulations is presented in this paper. A greedy algorithm is used to reduce the number of centers used for the RBF interpolation and an incremental approach is used for the inversion of the matrix system during each greedy iteration. Two different incremental approaches were implemented and tested: 1) Matrix inversion based, and 2) LU decomposition based. The use of incremental approach decreased the computational complexity of solving the system of equations within each greedy algorithm’s iteration from $O(n^3)$ to $O(n^2)$. This technique does not need any cell, face, or edge connectivity information and it depends only on the set of points and the boundary deformation. Therefore, it could be easily parallelized. However, in this study, the proposed approaches have not been implemented in a parallel framework.

Benchmark test cases with four different analytic deformations are used to evaluate the performance of the presented approach. The results from the numerical experiments showed that the number of centers required to perform the interpolation is independent of the total number of nodes and mainly depends on the deformation. This makes the technique optimal for fluid–structure interaction simulations where the meshes are very fine. Moreover, the algorithm’s order of accuracy is also independent of the total number of nodes. Results are also presented for mesh deformations for bending and twisting of a rectangular supercritical wing. These simulations showed both proposed incremental approaches take significantly less CPU time as compared to the traditional full LU decomposition. The present results show that improvement in CPU time saving increases as the number of selected centers for RBF increases. Moreover, the matrix inversion based approach showed instability issues when error tolerance is less than $1 \times 10^{-4}$. Therefore, it is recommended to use the LU decomposition based approach even though its CPU time requirement is slightly higher. In addition, the application of the incremental RBF mesh deformation for FSI problems showed that the proposed approaches are efficient and robust. However, to avoid picking many centers by the greedy algorithm when the structural deflections are minimum, a relative error tolerance is recommended to be used. Defining the RBF error tolerance as a fraction of the maximum structural deflection at every time step ensures an efficient mesh deformation process, especially when the structural deflections are negligible.

References

Materials
Computational
J. Comput.

Rendall,
Wang,
Tukovic,
Jo,
Farhat,
Sheng,
Li,
Pasquariello,
Jasak,
Bretscher,
Su,
Skala,
Trahan,
Shehata,
McDaniel,
Michler,
Tian,
Boer,
Ricketts,
Selim,
Koomullil,
Koomullil,
E.

J.

576–582.
Allen,
J.W.

258–270.

1381–1388.

231–245.

1155–1163.

1285–1300.

586–601.

149–172.

209–213.

955–959.

9–10,

7–49.

986–10079.

the

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.

9–10,

986–10079.