Kedward, L. J., Allen, C., & Rendall, T. (2017). Efficient and Exact Mesh Deformation using Multiscale RBF Interpolation. In *55th AIAA Aerospace Sciences Meeting* [AIAA 2017-0586] American Institute of Aeronautics and Astronautics Inc. (AIAA).
<https://doi.org/10.2514/6.2017-0586>

Peer reviewed version

Link to published version (if available):
[10.2514/6.2017-0586](https://doi.org/10.2514/6.2017-0586)

[Link to publication record in Explore Bristol Research](#)
PDF-document

This is the accepted author manuscript (AAM). The final published version (version of record) is available online via AIAA at <http://dx.doi.org/10.2514/6.2017-0586> . Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research

General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/>

Efficient and Exact Mesh Deformation using Multiscale RBF Interpolation

L. Kedward ^{*}, C.B. Allen [†], T.C.S. Rendall [‡]

Department of Aerospace Engineering, University of Bristol, Bristol, BS8 1TR, U.K.

Mesh deformation schemes are vital in many areas of numerical simulation, and radial basis function(RBF)-based interpolation schemes are popular due to their excellent quality preservation properties. However, the system solution cost scales with moving surface points as $N_{surface}^3$, and so there have been numerous works investigating efficient methods of reducing the data set. However, reduced data methods require the addition of a secondary ‘correction vector field’ to ensure surface points not included in the primary deformation are moved to the correct location, and the volume mesh moved accordingly. A new method is presented here, which captures global and local motions at multiple scales using *all* the surface points, and so there is no need for a correction stage. The multiscale formulation developed means that although all surface points are used and a single interpolation built, the cost and conditioning issues associated with RBF methods are eliminated while still retaining exact recovery of the surface. Moreover, the sparsity introduced can be exploited, using an existing wall distance function, to further reduce the cost. The method is compared with a conventional greedy method on two- and three-dimensional meshes with large deformations. It is shown that mesh quality is always comparable with or better than with the greedy method, and cost and complexity is also comparable or cheaper for all stages. The most expensive stage of reduced point methods is surface mesh preprocessing, and the cost is reduced significantly here; a three or four orders of magnitude reduction in cost is demonstrated compared to greedy-type methods.

I. Introduction

The need for dynamic mesh deformation within computational fluid dynamics (CFD) solvers arises in problems that involve moving geometric boundaries, such as fluid-structure interaction¹ or aerodynamic optimisation.² In such problems a complete regeneration of the volume mesh at each cycle would be prohibitively expensive and is also undesirable due to the elimination of time-history. Instead, an updating procedure is required that deforms the existing volume mesh in accordance with the new surface geometry at each time-step or optimisation cycle.

Numerical flow solution is notably sensitive to the quality of the mesh, particularly for viscous problems, and so mesh deformation schemes need to maintain the original mesh quality, as changes in orthogonality can lead to an increased truncation error while, in the worst case, mesh crossover and cell inversion can invalidate the mesh completely. Hence, robustness and preservation of mesh quality are of prime importance for mesh deformation schemes. Usually in contention with the quality-preserving ability of a mesh deformation method is the computational expense; during simulation or optimisation it is important that mesh deformation is computationally cheap to implement such that the overhead it introduces is minimal.³

A particularly desirable quality of a mesh deformation scheme is generality; a generic or universal method that can be applied equally to many problems, irrespective of mesh structure or configuration with little or no modification. This is in contrast to tailored solutions that are suited only to a specific problem. In¹ and⁴ Rendall and Allen present key requirements for a universal mesh motion scheme, namely: to operate regardless of mesh type; to require no connectivity information; to be able to handle multiple bodies in relative motion; to be suitable for parallelisation.

^{*}Graduate Student. Email: laurence.kedward@bristol.ac.uk

[†]Professor of computational aerodynamics, AIAA Senior Member. Email: c.b.allen@bristol.ac.uk

[‡]Lecturer, Aerospace Engineering Department. Email: thomas.rendall@bristol.ac.uk

Interpolation using radial basis functions (RBFs) has recently become a prominent mesh deformation method boasting excellent robustness and quality-preserving properties. Moreover, the method is completely generic, operating on point-clouds alone, and is perfectly parallel. A global deformation field can be generated analytically, from a cloud of control point movements, normally the surface points, and this has been shown to be particularly effective.^{1,5} However, the mesh deformation requires solution of a linear system the size of the moving control point cloud, and so the full method becomes very expensive for large meshes. Hence the implementation is commonly approximated using some kind of data-reduction scheme. Rendall and Allen^{6,3} developed very effective methods to improve the efficiency of RBF interpolation significantly, by selecting intelligently a reduced set of control points. Reducing the dataset reduces both the system solution cost and the mesh update cost but, equally as importantly, can improve the system conditioning significantly by increasing control point spacing. The important issue here is that only the selected control points are moved exactly by the reduced system interpolation; all other points are moved by the global interpolation, and so a vital part of the deformation method is a second stage which requires the treatment of a ‘correction vector field’ to ensure the non-selected control points move to their correct position, to recover exact surface movement,⁷ and the surrounding volume mesh deformed accordingly. The work in^{1,6,3} has in fact led to a large number of further methods, focussed on both efficient point selection methods and alternative methods to treat the correction vector, see for example.^{8,9,10,11,12,13,14} There has also been work investigating whether the reduced points selected can be selected once, before an unsteady simulation, or need to be reselected at every time level,¹⁵ adding an adaptive point selection method,¹⁶ and combining reduced data points with orthogonality improvements.¹⁷ A further work worthy of mention is that due to Poirier and Nadarajah,¹⁸ wherein both the primary and secondary RBF mesh movement algorithms are formulated such that they can be included in the mesh sensitivities of an adjoint-based optimisation method.

All reduced points methods aim to reduce both the system solution stage and the mesh update stage at each timestep or optimisation evolution. However, these methods require a both a pre-processing stage, either prior to simulation or at each unsteady time-step, *and* a method to treat the error or correction vector field that needs to be added to the approximate deformation field from the reduced points. The objective of the work presented here is to develop a new implementation of efficient RBF interpolation that is both fast and exact, *i.e.* computationally more efficient than the full method while still recovering all original data. In fact, the goal is to develop a single interpolation that includes *all* the data points, but is still significantly cheaper than any reduced point method in the preprocessing stage, cheaper or comparable for the system solution and mesh update stages and, most significantly, requires no correction stage as all control points are included.

II. Mesh Deformation

Numerous mesh deformation schemes have been developed, often depending on the mesh type or the particular application. Some of the more simple mesh deformation schemes, see for example,^{19,20} used a uni-parametric interpolation²¹ along grid lines between an inner surface definition, for example a wing, and the outer boundary. Since the resulting mapping is purely algebraic it is very computationally efficient. However, the method suffers from singularities and is limited to structured single-block domains. This can be extended to multi-block domains when used in combination with another deformation scheme for the block vertices,²² for example using the spring analogy for block edges.

The spring analogy, developed by Batina,²³ is one of the earliest and most commonly-used methods, wherein the connections between nodes are assigned compliances proportional to the connection length. This method can be used for both structured and unstructured meshes. Farhat²⁴ built on the method by introducing torsional stiffness to alleviate mesh crossover problem, and in Blom²⁵ this torsional spring was shown to be essential for moving viscous meshes. Sheta *et al*²⁶ have also reconsidered the formulation to use solid structural elements in an effort to prevent cell inversion. The method is fairly robust and accurate, but computationally expensive for large grids.²⁷ There are also PDE solution approaches, usually involving an elliptic problem solution, see for example Loehner.^{28,29} Grid quality can be improved by solving a bi-harmonic set of equations that also preserve orthogonality.^{30,31}

The spring analogy, and PDE-based methods, are expensive to solve and fail to guarantee mesh quality for large deformations. In contrast, mathematical-based methods, which generally take the form of a multivariate interpolation, are robust and do not usually rely on connectivity information.⁹ Liu *et. al.*³² developed a Delaunay graph interpolation scheme boasting high computational efficiency, however boundary mesh

quality and robustness were poor for large deformations and rotations.⁹ Allen⁴ developed a fast algebraic interpolation strategy using inverse distance weighting, and particular attention was paid to the preservation of orthogonality. Similar inverse distance weighting methods are also presented in³³ and³⁴ and methods have also been presented using disk and then sphere relaxation methods.^{35,36}

Recently there has been increased interest in radial basis function (RBF) interpolation for mesh deformation^{5,1,2} due to its generality, being able to operate on scattered data sets (requiring no connectivity) and having no limit on the dimensionality of the problem. Rendall and Allen used RBF interpolation in a unified approach to fluid-structure interaction,¹ wherein the universal characteristics of RBF interpolation as a mesh motion scheme were highlighted, and mesh quality is shown to be preserved well. However the full RBF method is prohibitively expensive to implement for large meshes.

The dominant factor in the cost of RBF interpolation is the number of control points.⁶ Qin *et. al.* address this problem with a hybrid method combining the Delaunay graph scheme with RBF interpolation.⁹ RBF interpolation is applied separately to each Delaunay graph sub-domain using the vertices as control points. Therefore, in three dimensions, there are only four control points in each interpolation and hence the size of the interpolation matrix is greatly reduced, resulting in improved computational efficiency.

Alternatively, the efficiency of the RBF method can be improved by using only a subset of the surface mesh points as control points. This approximation exploits the fact that surface mesh deformations can be reproduced, with minimal error, using only a fraction of the original surface mesh as control points. Hence, the important issue becomes how to reduce the system size and there has been a large amount of effort devoted to this area. Jakobsen and Amoignon² applied a simple mesh coarsening method based on maintaining uniformity of the reduced dataset. A smart coarsening method was presented by Rendall and Allen in which greedy algorithms were used to minimise the interpolation error at non-control points,⁶ demonstrating large cost reductions with a maximum error of less than 0.1%. Moreover the resulting scheme was shown to scale linearly with the number of volume points. The scheme was further improved⁷ by adding a correction step to recover exact surface displacements. A series of different data-reduction schemes followed, primarily focusing on the efficiency and error of the point-selection process, see for example;^{2,7,6,3,8,9,10,11,12,13,15,14,16,18} however all such reduction methods are still only approximate, requiring some treatment of an error vector field at the points not included in the interpolation.

Driving volume mesh deformation using a reduced set of surface mesh points makes sense since global deformations can be represented sufficiently well on a coarse subset. The rationale for the method presented herein is to capture global and local motions of multiple scales using all the surface points, and so there is no need for an error vector, but with a reduced cost compared to greedy-type methods. The multiscale formulation developed means that although all surface points are used and a single interpolation is built, the dimension of the linear system to be solved is significantly smaller than that of the full method.

III. Formulation

A general introduction to RBF interpolation is first given, followed by a description of RBF interpolation as applied to mesh motion, and a typical reduced data-set algorithm.

III.A. RBF Interpolation

RBF methods have become a popular tool across a wide variety of disciplines due to the flexibility they present. A particular advantage of RBF methods is the ability to operate on scattered data in any general multidimensional space. Moreover, RBF methods do not require any connectivity information, they are a mesh-free method. As such, much attention has been given to exploiting them.^{37,38}

A radial function is a function that takes the form $\phi(\mathbf{x}, \mathbf{x}_0) = \phi(\|\mathbf{x}_0 - \mathbf{x}\|)$ where $\|\cdot\|$ represents the vector norm, usually Euclidean, and $\mathbf{x}_0$ is the RBF ‘centre’. Given a set of discrete data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ in some space $\mathbf{R}^d$, and an associated set of scalar values $f_i = f(\mathbf{x}_i)$, then radial functions can be used to form a basis that spans this space and approximate the scalar field $f(\mathbf{x})$. The resulting RBF interpolant has the form:

$$f(\mathbf{x}) \approx s(\mathbf{x}) = \sum_{i=1}^N \gamma_i \phi(\|\mathbf{x}_i - \mathbf{x}\|) \quad (1)$$

A low-degree polynomial is sometimes included in the interpolation to provide an underlying global

trend, however this is normally undesirable in mesh deformation applications, since deformations would be amplified away from moving boundaries. The RBF coefficients γ_i are determined by requiring exact recovery of the known function values at the data sites, this results in a linear system to solve:

$$\mathbf{f} = \Phi \gamma \quad (2)$$

Where

$$\mathbf{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}; \Phi = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,N} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,N} \end{pmatrix}; \gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_N \end{pmatrix} \quad (3)$$

Here, $\phi_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. The size and structure of the interpolation matrix Φ is of key importance in determining the ease with which the system is solved. Solution methods are either direct or iterative, and the latter can readily take advantage of sparse matrix structure and become the only viable option for very large systems.

Radial basis functions can broadly be categorised as either global or local in nature, where the former grow with radial distance and the latter decay. Despite giving good interpolation accuracy, global basis functions tend to smooth out local effects and result in a dense interpolation matrix which is undesirable both physically and practically.³⁸ Consequently, local RBFs, specifically compactly supported RBFs, have become increasingly popular.

Compactly supported RBFs, a subset of local RBFs, decay to zero at a finite radial distance, known as support radius R , and remain zero thereafter. Compact functions present computational benefits via sparseness of the interpolation matrix, though at the cost of reduced interpolation accuracy; the trade-off between the two is controlled by the choice of support radius. Of more concern is the effect on the conditioning of the interpolation matrix, a function of the ratio of the support radius to the spatial density of the data. Wendland³⁹ derived positive definite compactly supported functions having the lowest order for a given order of smoothness, see Table 1, where r is the normalised Euclidean norm.

Table 1: Wendland's compactly-supported radial basis functions

Name	Definition
C0	$\phi(r) = (1 - r)^2$
C2	$\phi(r) = (1 - r)^4(4r + 1)$
C4	$\phi(r) = (1 - r)^6(35r^2 + 18r + 3)$

In⁷ it is shown that the existence of a smooth first derivative of the basis function is desirable for preservation of orthogonality. C2 is the lowest order function with a smooth first derivative and has already been shown to give a good balance between quality-preservation and system conditioning^{1,3}. Hence, C2 is used here, applied as:

$$r = (\|\mathbf{x}_i - \mathbf{x}_j\|)/R \quad (4)$$

$$\phi(r) = \begin{cases} (1 - r)^4(4r + 1) & r \leq 1 \\ 0 & r > 1 \end{cases} \quad (5)$$

III.B. RBF Mesh Deformation

The mesh-free property of RBF interpolation is particularly well-suited to mesh deformation since it removes any flow-solver dependency and requires no manual user input.^{5,1} In their demonstration of RBF mesh motion, de Boer *et. al.*⁵ observe the simplicity of implementation and the superior mesh quality when compared with the spring analogy.²³ Jakobsson and Amoignon apply RBF mesh motion to aerodynamic shape optimisation, demonstrating the availability of gradients with respect to control point positions.² However, the RBF formulation suffers from high cost and reduced accuracy as mesh size increases.

Volume mesh deformation is driven by the motion of the surface mesh. The target function to interpolate is the vector of nodal displacements $\mathbf{u}_s$ for the surface mesh, and Φ is the $N_s \times N_s$ interpolation matrix constructed using the surface mesh points as RBF centres, and N_s is the number of surface mesh points. A

separate interpolation is required for the displacements in each dimension, however the interpolation matrix is the same for each. Hence in three dimensions:

$$\mathbf{u}_s^x = \Phi \gamma^x, \quad \mathbf{u}_s^y = \Phi \gamma^y, \quad \mathbf{u}_s^z = \Phi \gamma^z \quad (6)$$

The displacements of the N_v volume mesh points can be obtained using an $N_v \times N_s$ evaluation matrix Ψ :

$$\mathbf{u}_v^{x/y/z} = \Psi \gamma^{x/y/z} \quad (7)$$

Where

$$\Psi = \begin{pmatrix} \phi_{v1,s1} & \phi_{v1,s2} & \cdots & \phi_{v1,N_s} \\ \phi_{v2,s1} & \phi_{v2,s2} & \cdots & \phi_{v2,N_s} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N_v,s1} & \phi_{N_v,s2} & \cdots & \phi_{N_v,N_s} \end{pmatrix} \quad (8)$$

Clearly, whether this matrix is actually constructed is dependent on the system size.

III.C. Implementation

There are two main approaches to implementing an RBF mesh deformation scheme at each time step of mesh movement; an *operation-intensive* method and a *memory-intensive* method. The operation-intensive approach consists of two steps: 1) Solve for field vectors $\gamma^{x/y/z}$; 2) Update volume mesh points. In this way, the operation-intensive method does not have any significant memory costs since the matrices Φ and Ψ do not need to be explicitly generated and stored in full. However, as suggested by the name, the operation-intensive method requires many processor operations per mesh deformation step. In contrast, the memory-intensive approach takes advantage of the fact that the relationship between surface mesh displacements $\mathbf{u}_s$ and volume mesh displacement $\mathbf{u}_v$ is constant and entirely linear. Hence a single $N_v \times N_s$ transfer matrix $\mathbf{H}$ can be defined, as in equation 10.

$$\mathbf{u}_v = \mathbf{H} \mathbf{u}_s \quad (9)$$

Where

$$\mathbf{H} = \Psi \Phi^{-1} \in \mathbb{R}^{N_v \times N_s} \quad (10)$$

This matrix can be generated once and saved prior to commencing optimisation iterations or time stepping. Therefore at each mesh deformation step only a matrix-vector multiplication is required. However the full transfer matrix $\mathbf{H}$ is of dimension $N_v \times N_s$ and can hence be too costly to store for large meshes. The operation- and memory-intensive approaches can be summarily compared by their complexity with respect to the size of the input meshes. Table 2 gives the time and memory complexity for each stage of the RBF mesh deformation.

Table 2: Comparison of implementation complexity

	Cost	
	Operation-intensive	Memory-intensive
Preprocessing	-	$O(N_s^3)$
Solve	$O(N_s^3)$	-
Update	$O(N_s N_v)$	$O(N_s N_v)$
Memory	-	$O(N_s N_v)$

It is clear that the size of the surface mesh (N_s) is prominent in the overall cost of the RBF method. Moreover, solving the system scales with the N_s^3 , hence full RBF mesh deformation quickly becomes infeasible for large meshes, due to both cost and system conditioning issues.

Table 3: Mesh refinement: memory and operation costs for memory- and operation-intensive approaches

Mesh		Storage of $\mathbf{H}$	Preprocessing or at each stage		Each stage
N_s	N_v (Mpoints)	Memory (GB)	Condition No.	System Solution	Mesh Update
257	0.02	0.03	$O(10^{10})$	$O(10^7)$	$O(10^6)$
513	0.07	0.27	$O(10^{11})$	$O(10^8)$	$O(10^7)$
1025	0.26	2.16	$O(10^{12})$	$O(10^9)$	$O(10^8)$
2049	1.05	17.23	-	$O(10^{10})$	$O(10^9)$

III.D. Example problem

To demonstrate the RBF method, a 30° rotation about the origin is applied to a NACA-0012 surface mesh consisting of 257 points. RBF interpolation is used to transfer this deformation to the volume mesh, a structured 257×65 O-Mesh shown undeformed in Figure 1(a). Figure 1 also shows the resulting deformed mesh using Wendland's C2 function with a support radius of 4 chords. The good quality-preservation is clearly demonstrated, with mesh orthogonality at the surface being well maintained and dissipating away smoothly. Orthogonality is quantitatively analysed for the full RBF method in section 5. Several refinements have been made to the surface and volume mesh, as given in Table 3, to demonstrate the cost. The memory cost for storing the full transfer matrix of the memory-intensive method (for double precision storage) along with approximate costs for this and the operation-intensive approach is given in Table 3, along with the order of magnitude of the condition number for the solve stage. At 2049 surface points, the memory cost has grown beyond practicality and the conditioning of the system has exceeded machine double precision. It is clear that the solve cost is dominant for the operation-intensive method and so reducing the system size is essential for general application.

(a) NACA 0012 Aerofoil, structured O-Mesh

(b)

(c)

Figure 1: NACA0012 Rigid rotation case, full RBF method, SR = 4

III.E. Conventional reduced-point method

It is clear from simple costing that RBF interpolation can not be used in its full form for larger meshes, but instead implemented using some data-reduction method.^{3,10,15} These methods use a reduced number $N_{red} < N_s$ of surface points as control points. The reduced dataset is normally chosen in an intelligent way, such that the interpolation error is minimised on points outside of the reduced dataset. Algorithm 1 shows one such method, a greedy algorithm for selecting the reduced data set.⁶ The method is termed *greedy* since at each cycle it chooses the point with the largest error. Note, however, that an error vector always exists on those points not selected for the interpolation, and hence some correction method is always required to deform the volume mesh correctly to recover the exact surface geometry. Methods of this type have become very common, and are significantly cheaper than the full method, but require a preprocessing stage, either prior to simulation or at every time step, and a correction vector field addition method at every time step. This method is used later for comparison with the multiscale method presented below.

Algorithm 1 Greedy full point selection algorithm³

```

1: procedure
2: init:
3:    $\mathbf{x}_a \leftarrow \{\dots\}$                                 ▷ Select initial set of points and add to the active set
4:    $\mathbf{f}^{exact} = \mathbf{u}^{x/y/z}$                                 ▷ Set function to interpolate to surface displacements
5: main:
6:   while  $N_{active} < N_{red}$  do
7:      $\boldsymbol{\alpha}_a = \boldsymbol{\Phi}_a^{-1} \mathbf{f}_a^{exact}$                     ▷ Solve interpolation on active set
8:      $\mathbf{f}^{eval.} = \boldsymbol{\Psi} \boldsymbol{\alpha}_a$                                 ▷ Evaluate active set interpolant at all points
9:      $\mathbf{e} = F(\mathbf{f}^{exact} - \mathbf{f}^{eval.})$                         ▷ Evaluate interpolation errors
10:     $i_{worst} = \arg \max_{i \in [1, N_x]} \mathbf{e}(i)$                 ▷ Identify the point with the worst error
11:     $\mathbf{x}_a \leftarrow \mathbf{x}(i_{worst})$                             ▷ Add point to active set
12:   end while
13: output:
14:   return  $\mathbf{x}_a$                                             ▷ Reduced point set
15: end procedure

```

IV. Multiscale RBF Interpolation

The new method is presented here. Methods named *Multistep* (or *multilevel*) RBF interpolation have been presented previously as an approach to interpolating scattered data of varying density and that contain phenomena on multiple length scales.⁴⁰ These use multiple support radii within the interpolation to match the multiple length scales required, however, these produce a non-symmetric interpolation matrix which therefore cannot be guaranteed to be non-singular.⁴¹ In the multistep method an interpolant is first constructed on a coarse subset of the data, then the dataset is refined and another interpolant constructed for the residual (the error due to reduction) from the previous level. At the refined level a smaller support radius is used to match the data density and capture finer details. This refinement process is repeated until the full dataset is reached. The sum of all the interpolants interpolates the target function on multiple scales while still recovering values at all data sites.

The work herein will be named a *multiscale* approach; here only a single interpolation is constructed. Despite varying the support radii within the interpolation the system is still uniquely soluble due to the construction of the interpolation function. First an interpolation system is built on a (very) coarse subset of the data, named here the *base set*. From here, successive refinements are made where the interpolation from the new control points are formulated such that the mesh refinement does not influence points in the preceding mesh level. Hence the interpolation system for the incremental refinement can be solved in isolation. The existing interpolation does, of course, affect the interpolation at the added points but the resulting matrix structure can be exploited to allow a simple solution procedure.

To ensure each level of refinement is as uniform as possible, the selection of refinement points is made such that it maximises the separation distance of the resulting mesh.^{2,40} The separation distance Q of a set of points Ω is defined as the radius of the largest sphere centered on a data-point. Put another way: each

point has an individual separation distance equal to the distance to the nearest point, the largest of these over a group of points is the separation distance for that group of points:

$$Q = \max_{i \in \Omega} \min_{j \in \Omega, i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\| \quad (11)$$

Now consider if each refinement only adds a single point to the mesh, then the interpolation system at each refinement simply becomes an explicit expression for the RBF coefficient in terms of error function. The procedure is outlined as follows:

1. Take a coarse subset of N_b base control points as the *active set*;
2. Solve the interpolation problem on the active set, using a user-specified base support radius r_0 ;
3. Evaluate the residual error at all points not in the active set;
4. Identify the next refinement point - the point having the largest separation distance with respect to active set - and add to the active set;
5. Set the support radius of the refinement point equal to the separation distance;
6. Set the RBF coefficient for the refinement point equal to the residual error at that point;
7. If not all points added, *go to step 3*.

Figure 2: Multiscale refinement process

The only user-specified parameters here are the base-set support radius and the number of points in the base set. Figure 2 illustrates the single-point refinement process using separation distance. Consider now the structure of the resulting interpolation matrix, assuming points are sequenced in the order that they are included into the interpolation, *i.e.* the first N_b points are the *base set* and $N_b + 1$ is the first refinement point added *etc.* Subscript r refers to the *refinement* points - those not included in the base set - such that $N_s = N_r + N_b$. The first N_b rows and N_b columns are those formed by the base set of points alone. When a refinement point is added, the matrix is augmented by one row and one column. Since the support radius is chosen such that it has no influence on preceding points, all entries in the new column upto the diagonal are zero. Entries on the new row are non-zero and represent the evaluation of the existing interpolant at the new refinement point. Hence, the final interpolation matrix has the following block structure:

$$\Phi = \left(\begin{array}{cccc|cccc} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,N_b} & 0 & 0 & \cdots & 0 \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,N_b} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{N_b,1} & \phi_{N_b,2} & \cdots & \phi_{N_b,N_b} & 0 & 0 & \cdots & 0 \\ \hline \phi_{N_b+1,1} & \phi_{N_b+1,2} & \cdots & \phi_{N_b+1,N_b} & \phi_{N_b+1,N_b+1} & 0 & \cdots & 0 \\ \phi_{N_b+2,1} & \phi_{N_b+2,2} & \cdots & \phi_{N_b+2,N_b} & \phi_{N_b+2,N_b+1} & \phi_{N_b+2,N_b+2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{N_s,1} & \phi_{N_s,2} & \cdots & \phi_{N_s,N_b} & \phi_{N_s,N_b+1} & \phi_{N_s,N_b+2} & \cdots & \phi_{N_s,N_s} \end{array} \right) \quad (12)$$

And hence, the linear system to solve:

$$\begin{pmatrix} \mathbf{u}_b^x \\ \mathbf{u}_r^x \end{pmatrix} = \begin{pmatrix} \Phi_b & \mathbf{0} \\ \Psi_r & \mathbf{L} \end{pmatrix} \begin{pmatrix} \alpha^x \\ \beta^x \end{pmatrix} \quad (13)$$

Φ_b is the $N_b \times N_b$ interpolation matrix for the base set, which is positive definite for suitable choice of RBF $\phi(\|\cdot\|)$. Ψ_r is the $N_r \times N_b$ evaluation matrix for the base set interpolant onto the refinement points. $\mathbf{L}$ is an $N_r \times N_r$ lower triangular interpolation matrix for the refinement points. The multiscale process outlined above solves this system by forward substitution. First, the upper system is solved for the base set of coefficients α^x , which are then substituted into the lower system to evaluate the residual for the refinement points: $\Delta = \mathbf{u}_b^x - \Psi_r \alpha^x$. Finally, since $\mathbf{L}$ is a lower triangular matrix, the lower system is itself solved by forward substitution for β^x . The volume mesh is then updated using the coefficient vectors α^x and β^x . Equation (13) is solved exactly and hence the original surface displacements are recovered. The only linear system to solve is that involving Φ_b , and since Φ_b is only a coarse subset of the full surface mesh, then the cost and conditioning issues of the full RBF method are eliminated.

V. Computational Aspects

In this section, computational implementation of the multiscale method is considered, specifically the opportunities presented for cost reduction, and the costs compared to a conventional greedy-type method. For clarity, the greedy method chosen is the simplest; mesh update with a reduced set of data points, followed by a single correction update for each volume point based on a nearest surface point interpolation.

The entire mesh deformation process can be split into four stages, listed below:

1. **Surface Preprocessing:** The preprocessing step required to select the reduced surface points for greedy, or to sequence the surface control points in order of decreasing separation distance for multiscale.
2. **Volume Preprocessing:** The preprocessing step for volume mesh points, either selecting surface points for greedy correction, or processing the reduced influencing points for multiscale.
3. **Solve:** The system solution is either full solution for the N_{red} greedy points, or a full solution on the N_b base points and sequential explicit updates on the remaining refinement points for multiscale.
4. **Update:** Each volume point moved by summation over the N_{red} points and a single correction vector addition for greedy, or summation over the influencing points for multiscale.

V.A. Surface Mesh Preprocessing: Multiscale

Once the base set problem set has been solved, an iteration is performed over the remaining refinement points in order of decreasing separation distance, using the separation distance as the support radius. The sequencing of control points and saving of support radii can be performed once, prior to any simulation. This preprocessing step is detailed in Algorithm 2.

This algorithm operates on a point cloud of N_s points, requiring no connectivity information (structured/unstructured) or topology (multiple surfaces). The main loop of the sequencing algorithm has $N_s - 1$ iterations. At each iteration, the algorithm calculates the separation distance of all remaining points with respect to the processed points and chooses the point with the largest separation distance to add to the processed list.

The surface mesh array can either be re-ordered, or an ordered list of indices can be stored. An array of support radii is also produced. Hence the total number of operations is given by:

$$N_{preproc_surf}^{op} = \sum_{i=1}^{N_s-1} (N_s - i) + O(N_s) + O(N_r) = O(N_s^2) \quad (14)$$

V.B. Surface Mesh Preprocessing: Greedy

Equivalent expressions for number of operations are required for a representative greedy point reduction method. Algorithm 1 is an example of one such method. The algorithm takes the surface displacements as

Algorithm 2 Control Point Sequencing

```
1: procedure
2:   init:
3:      $inactiveList \leftarrow [1, N_s]$                                  $\triangleright$  All point indices start off on the inactive list
4:     for  $i = 1, N_s$  do
5:        $separation(i) \leftarrow LARGE$                                  $\triangleright$  Initialise separation distance list
6:     end for
7:      $activeList(N_{active}) \leftarrow k = 1$                              $\triangleright$  Add first point to active list
8:      $inactiveList \leftarrow inactiveList \setminus \{k\}$ 
9:   main:
10:    while  $N_{active} < N_s$  do
11:      for  $i$  in  $\{inactiveList\}$  do                                 $\triangleright$  Update separation distances
12:         $r_{ik} \leftarrow \|x_i - x_k\|$                                  $\triangleright$  Distance to most-recently added point
13:        if  $r_{ik} < separation(i)$  then
14:           $separation(i) \leftarrow r_{ik}$ 
15:        end if
16:      end for
17:       $k \leftarrow \arg \max_k (separation(k))$                              $\triangleright$  Select point with largest separation distance
18:       $activeList(N_{active}) \leftarrow k$                                  $\triangleright$  Move to active list
19:       $inactiveList \leftarrow inactiveList \setminus \{k\}$ 
20:      if  $N_{active} \leq N_b$  then                                 $\triangleright$  Save support radius value
21:         $radii(N_{active}) \leftarrow r_{base}$ 
22:      else
23:         $radii(N_{active}) \leftarrow separation(k)$ 
24:      end if
25:    end while
26:   output:
27:    return  $activeList, radii$                                  $\triangleright$  Ordered list of control point indices & support radii
28: end procedure
```

input for selecting the reduced point set. Therefore the point selection can occur every time the surface is displaced or it can be performed once at the beginning using some representative displacement (*e.g.* mode shapes). Here it is assumed that the point selection occurs in advance as a preprocessing step. To produce a reduced set of N_{red} points, the algorithm performs N_{red} iterations, where at each iteration a linear system is solved, the interpolant evaluated on the remaining points and a search performed for the point with maximum error. Hence, the cost for realistic mesh sizes is:

$$\bar{N}_{preproc_surf}^{op} = \sum_{N_i=1}^{N_{red}} (O(N_i^3) + N_i(N_s - N_i) + (N_s - N_i)) = O(N_{red}^4) \quad (15)$$

V.C. Volume Mesh Preprocessing: Multiscale

The support radius of the base set r_0 is specified by the user and controls how far out the base interpolation reaches. The support radius of the remaining points is set according to their separation distances with respect to the current active set. Let r_1 be the support radius of the first refinement point after the base set. All remaining refinement points will have a support radius less than or equal to r_1 .

The wall distance d_i for a volume mesh point is the shortest distance between that point and a geometric boundary, and is used here to categorise points. This quantity is often available anyway within a CFD solver, so is a convenient quantity to use.

Figure 3: Number of influencing surface points per volume mesh point ($N_b = 25$)

Volume mesh points are categorised as follows:

1. $r_0 < d_i$ - Volume points which do not fall within the base set influence, and hence are not influenced by any control points;
2. $r_1 < d_i \leq r_0$ - Volume mesh points only influenced by the base set;
3. $d_i \leq r_1$ - Volume mesh points influenced by the base set and additional refinement points

Furthermore, r_1 will be of the order of the spatial density of the base set. Hence, the volume mesh will mostly consist of points in the first two categories, with only a few points near the surface being in the third category. This can be seen in Figure 3 which shows the number of actively influencing surface points at each volume mesh point. The multiscale nature is also clearly illustrated. Moreover, those points near the surface in the third category will only be influenced by a small local subset. Therefore the matrix Ψ of equation (8) will be sparse. A preprocessing step can be designed, using any suitable wall distance function, to efficiently categorise each volume point and, for those points in the third category, to find and store the additional influencing indices. If N_{v1} , N_{v2} and N_{v3} are the number of volume mesh points in categories one, two and three respectively, then the number of inner-loop operations for volume mesh preprocessing is:

$$N_{preproc_vol}^{op} = N_{v1} + N_{v2} + N_{v3}N_r \quad (16)$$

Only the category three points need a surface point search, and this number is a function of N_b , as the r_1 value depends on the maximum base point separation. Hence, the number of operations are:

$$N_{preproc.vol}^{op} = \alpha N_v N_r \quad (17)$$

where $\alpha < 1$.

V.D. Volume Mesh Preprocessing: Greedy

The volume preprocessing stage for the greedy algorithm is required for the error correction vector update. Several methods have been developed for this since the original greedy work, but the simplest approach is used here; a nearest point correction, weighted by a distance function. This means every point on the surface may then be moved exactly to its required position, and this small perturbation is dissipated gradually into the volume mesh based on the weighting function, so that the correction to each volume mesh point is given by:

$$\Delta \mathbf{x}_v = \phi(\|\mathbf{x}_v - \mathbf{x}_n\|/R_c)(\Delta \mathbf{x}_n^{exact} - \Delta \mathbf{x}_n^{interp}) \quad (18)$$

where n represents the nearest surface neighbour point. The correction vector support radius, R_c , should be smaller than that for the RBF interpolation and, for consistency with the multiscale method, exactly the same value is used as for refinement points, i.e. maximum separation distance of the reduced point set.

Points outside the distance R_c from the surface can, in a similar argument to that for category 1 and 2 points in the multiscale method, be left out of the correction. Hence, the conventional greedy correction above can also be improved significantly by also using the wall distance function. As with the multiscale method, the volume preprocessing operations are:

$$N_{preproc.vol}^{op} = \alpha N_v N_s \quad (19)$$

where $\alpha < 1$. Hence, for $N_b = N_{red}$ the multiscale method volume preprocessing will always be slightly cheaper than the greedy method since N_s is always greater than N_r .

V.E. Solve and Update Steps: Multiscale

The solution stage requires solution of the $N_b \times N_b$ system. The second stage of solving equation 13 involves forward substitution of the $N_r \times N_r$ lower triangular matrix $\mathbf{L}$ which requires N_r^2 operations. However, the multiscale support radii mean that each refinement point only influences a small local set of points and as such, $\mathbf{L}$ is sparse. When the surface mesh is preprocessed, influence distances between pairs of control points are calculated for separation distance. The same distances are used in evaluating the interpolation matrix and hence the sparsity of $\mathbf{L}$ can be determined. If N_{nz} is the number of non-zero elements in matrix $\mathbf{L}$, then the cost for storing the sparse structure is $2N_{nz} + N_r + 1$ and the cost for forward substitution reduces to only N_{nz} . Hence, the number of operations required to solve the multiscale system is:

$$N_{solve}^{op} = N_b^3 + N_{nz} = O(N_b^3) \quad (20)$$

Assuming the volume mesh has been preprocessed as presented above, the cost of the update step is:

$$N_{update}^{op} = N_b N_{v2} + N_{nz-v3} \quad (21)$$

where N_{nz-v3} is the number of non-zero influences for volume mesh points in the third category. Since N_{v2} is some fraction of N_v :

$$N_{update}^{op} = O(N_v N_b) \quad (22)$$

V.F. Solve and Update Steps: Greedy

The solve step simply involves solution of the reduced point set interpolation:

$$\bar{N}_{solve}^{op} = O(N_{red}^3) \quad (23)$$

Only the reduced point set is used to update the volume mesh, then the extra step is required to correct the volume mesh for the actual surface point positions. As with the multiscale method this correction step will also be some fraction of N_v , hence:

$$\bar{N}_{update}^{op} = N_v N_{red} + N_v = O(N_v N_{red}) \quad (24)$$

V.G. Detailed Cost Comparison

The two methods are compared here in terms of pure cost alone, with mesh quality considered later. Two-dimensional and three-dimensional case are considered: the aerofoil meshes discussed previously are used, with sizes 257×65 , 513×129 , 1025×257 , 2049×513 , 4097×1025 . A structured multiblock mesh for a wing is considered as the three-dimensional problem, with mesh sizes: $N_v = 0.148M$, $N_s = 3,881$; $N_v = 1.111M$, $N_s = 15,441$; $N_v = 8.620M$, $N_s = 61,601$. For all the data here, $N_b = N_{red} = 0.05N_s$, although it is shown later that this many points is not required for finer meshes.

Figure 4: Comparison of cost with a conventional greedy RBF method

Figure 4 shows the cost of the preprocessing stages for surface and volume for the two methods. The volume preprocessing stage cost presented is using the standard greedy method, to show the efficiency gained using the wall distance function efficiency; the cost of the greedy method using the wall distance function reduces to almost exactly the same as for multiscale. The surface preprocessing stage is significantly more expensive for the greedy method, and renders the volume preprocessing cost almost negligible, whereas the multiscale volume preprocessing stage dominates, although this is still very cheap. The system solution and mesh update stages are not presented here as they are identical for $N_b = N_{red}$.

VI. Two-Dimensional Application

The multiscale method is now demonstrated in two dimensions and the mesh quality considered. A NACA-0012 surface mesh of 1025 points and a 1025×257 volume mesh is used here, and mesh quality is investigated using the change in orthogonality between undeformed and deformed meshes. The orthogonality metric q of a volume mesh point is calculated here using a similar method to that of Siebert and Dulikravich.⁴² Orthogonality is often taken as zero to be perfectly orthogonal, i.e. all internal mesh edges are at $\pi/2$ to each neighbour, but it is the relative change in orthogonality that is important for a mesh deformation scheme, not the absolute value, and so the value here is reversed to mean 1.0 is perfectly orthogonal, and the value can then be normalised by the undisturbed value.

A rigid rotation of 30° about the origin is applied to the surface mesh, and mesh deformation is first calculated using the full RBF method, as a baseline for orthogonality change. Figure 5 shows the relative change in orthogonality (relative to the undisturbed mesh) for the full RBF method for four different support radii; the mesh is an O-mesh, and the wake/slit line is also shown for illustration purposes. Note that all orthogonality plots present the magnitude of change. The good preservation of mesh quality is clearly demonstrated, with surface orthogonality being maintained even for the smallest support radius. The effect of support radius is also clearly shown to control the extent of the RBF influence. A support radius of 4 chords sufficiently offsets the displacement field away from the aerofoil surface with an acceptable maximum change in orthogonality and also gives a good base for comparison for the multiscale method; a support radius of 4 chords will be used henceforth.

Figure 5: Rigid rotation - full RBF method for varying support radius (SR, in chord units)

The rigid-rotation case was then tested using the new multiscale method. Figure 6 shows change in orthogonality relative to the deformed mesh using the full RBF method with support radius of four. Hence, the full scheme is matched very closely even for only 5% of the surface points as the base set. Since only the base set is responsible for deformations far from the surface, this confirms the observation that deformation fields for global motions can be sufficiently represented using only a fraction of the surface data.

VII. Three-Dimensional Application

VII.A. Modern transport wing

The Brite-Euram MDO (multidisciplinary optimisation) wing of semi-span 35m is used here as the surface geometry.⁴³ Three structured multi-block volume meshes have been generated,⁴⁴ as defined in table 4. An exaggerated deformation is considered; the 18th structural mode is transferred to the surface points using RBF interpolation,¹ such that there is a 2m maximum deflection at the wing tip nodes. Figure 7 shows the undeformed and deformed surface, showing the harsh nature of the deformation, particularly at the tip; this is the one million cell mesh surface. Also presented in table 4 is the average orthogonality values over all cells in each mesh, and average values for the first two grid planes away from the surface, to give a global and local orthogonality measure.

Table 4: Transport wing: mesh densities and undeformed orthogonality

N_s	N_v (M points)	q_0	q_0^{surf}
3,881	0.148	0.8921	0.9105
15,441	1.111	0.8900	0.9114
61,601	8.620	0.8888	0.9116

Figure 6: Rigid rotation - multiscale RBF method for varying number of base set points

Figure 7: MDO wing. Red - mode 18 deflection. Black - original surface.

VII.B. Improving Performance

In this work the selection of the base set occurs without any prior displacement information. This is as opposed to the greedy method which uses characteristic displacements (e.g. modal deformations) or actual displacements to drive point selection. Instead, methods are adopted which heuristically attempt to maximise the accuracy of the base set interpolant. The simplest approach is to maximise the spatial coverage of the domain; the base set is built up by sequentially adding points that have the furthest distance to the existing base set points (Algorithm 2). In this method, termed space-filling, the criterion evaluated at inactive points is the separation distance (distance to nearest active point), and the point with the maximum criterion is

added at each step. A space-filling base set for the MDO wing test case is shown in figure 8a. However, while this method is suitable in two dimensions, it can suffer from reduced accuracy for large three-dimensional meshes with thin bodies and large variations in mesh density. Two enhancements have been made to the base set selection method to improve robustness with respect to the surface mesh.

First, the space-filling method is modified to include some consideration of local mesh density. Whenever the separation distance of an inactive point is updated (Algorithm 2: line 14) the inactive point is also recorded as a ‘child’ of the closest active point. Then when selecting the next point to add to the active list (line 17), a search is performed for the active point with the most children and, for this active point, the child with the largest separation distance is added to the active set. This method, termed clustered space filling, only affects the selection of the base set (*i.e.* $N_{active} < N_b$), normal space-filling is maintained for refinement point sequencing. The result of this improved point selection method is shown in Figure 8b.

Figure 8: Base sets for $N_b = 250$

The second enhancement has been developed to minimise the issue of deformations on the opposite side of a surface influencing each other. An anisotropic pseudo norm function has been defined as:

$$\|\mathbf{x} - \mathbf{x}_i\|_{\mathbf{n}_i} = \psi_i(\mathbf{x}) \cdot \|\mathbf{x} - \mathbf{x}_i\| \quad (25)$$

$$\psi_i(\mathbf{x}) = \begin{cases} 1 + a(\hat{d})^2, & \text{if } \hat{d} \leq 0 \\ 1 & \text{otherwise} \end{cases} \quad (26)$$

where a is a compression parameter. This is defined as $a = f(R, r_j)$, where R is the global support radius used for the base set, and r_j is the local mesh separation distance. The dot product $\hat{d}$ is negative when the outward surface is not facing the relative location of point $\mathbf{x}$ which has the effect of exaggerating the distance to points ‘behind’ the local surface.

VII.C. Mesh quality

Mesh quality is now investigated in detail for the modern transport wing test case for the three mesh densities. The multiscale RBF method is applied to transfer the surface displacements into the volume mesh again using Wendland’s C2 function with a base support radius of one semi-span (35 m). For comparison, the test case mesh deformation is also performed using the greedy full point method³ (Algorithm 1) where $N_{red} = N_b$. A unit displacement is used for point selection and a single-point correction⁷ is applied for exactness using a support radius equal to that of the maximum reduced surface set separation distance, as explained earlier.

Figures 9 to 11 show the variation in average mesh orthogonality and mesh orthogonality in the near surface region, for varying surface set sizes. This clearly demonstrates that for both greedy and multiscale methods, only a small number of surface points are required to deform the mesh effectively. Furthermore, the multiscale method performs slightly better in all cases. For all meshes, 500-750 points are sufficient, and

Figure 9: Mesh quality variation with N_b, N_{red} ($N_v = 0.148M$)

Figure 10: Mesh quality variation with N_b, N_{red} ($N_v = 1.111M$)

this is around 1% of the finer mesh surface size. This confirms the findings in,³ where it is shown that the error is dependent on the geometry representation, not the surface mesh size.

VIII. Conclusions

An efficient RBF mesh deformation method has been presented that uses all surface points to ensure an exact surface representation, but still only requires a single interpolation to be built. The RBF interpolant is constructed using a coarse base set followed by multiple single-point refinements of scaled support radii, resulting in capture of global and local motions of multiple scales. Selecting support radii to match the mesh density at each level of refinement guarantees the existence of a unique interpolant and limits the size of the linear system to solve to the size of the base set. This overcomes the cost and conditioning issues associated with the full RBF method while still retaining exact recovery of the full surface. Moreover, the sparsity introduced by the multiscale method can be exploited, using an existing wall distance function, to further reduce the cost of the solve and update steps, and this has been demonstrated.

The multiscale method has been tested on two-dimensional and three-dimensional structured meshes for large deformations, and it has been shown that mesh orthogonality is well-maintained for base sets of less

Figure 11: Mesh quality variation with N_b, N_{red} ($N_v = 8.620M$)

than 1% of the surface points for the finer mesh. In all cases the mesh quality is comparable or slightly better with the multiscale method than the greedy method. It has also been shown that the cost and complexity of the multiscale method are comparable, if not cheaper, than current greedy methods at all stages. The most expensive part of reduced point methods is the surface mesh preprocessing/point selection stage, and the cost is reduced significantly here from N_{red}^4 to N_s^2 . However, unlike such reduction methods, the interpolation is exact, requiring no subsequent error correction.

References

- ¹Rendall, T. C. S. and Allen, C. B., "Unified fluid-structure interpolation and mesh motion using radial basis functions," *International Journal for Numerical Methods in Engineering*, Vol. 74, No. 10, 2008, pp. 1519–1559.
- ²Jakobsson, S. and Amoignon, O., "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization," *Computers and Fluids*, Vol. 36, 2007, pp. 1119–1136.
- ³Rendall, T. C. S. and Allen, C. B., "Reduced Surface Point Selection Options for Efficient Mesh Deformation using Radial Basis Functions," *Journal of Computational Physics*, Vol. 229, No. 8, 2010, pp. 2810–2820.
- ⁴Allen, C. B., "Parallel universal approach to mesh motion and application to rotors in forward flight," *International Journal for Numerical Methods in Engineering*, Vol. 69, No. 10, 2007, pp. 2126–2149.
- ⁵de Boer, A., van der Schoot, M. S., and Bijl, H., "Mesh deformation based on radial basis function interpolation," *Computers and Structures*, Vol. 85, 2007, pp. 784–795.
- ⁶Rendall, T. C. S. and Allen, C. B., "Efficient Mesh Motion using Radial Basis Functions with Data Reduction Algorithms," *Journal of Computational Physics*, Vol. 228, No. 17, 2009, pp. 6231–6249.
- ⁷Rendall, T. C. S. and Allen, C. B., "Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors," *International Journal for Numerical Methods in Engineering*, Vol. 81, No. 1, 2010, pp. 89–105.
- ⁸Michler, A., "Aircraft control surface deflection using RBF-based mesh deformation," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 10, 2011, pp. 986–1007.
- ⁹Wang, Y., Qin, N., and Zhao, N., "Delaunay graph and radial basis function for fast quality mesh deformation," *Journal of Computational Physics*, Vol. 294, No. , 2015, pp. 149–172.
- ¹⁰Wang, G., Mian, H. H., Ye, Z.-Y., and Le, J.-D., "Improved Point Selection Method for Hybrid-Unstructured Mesh Deformation Using Radial Basis Functions," *AIAA Journal*, Vol. 53, No. 4, 2015.
- ¹¹Erzincanlı, B. and Sahin, M., "An arbitrary Lagrangian-Eulerian formulation for solving moving boundary problems with large displacements and rotations," *Journal of Computational Physics*, Vol. 255, 2013, pp. 660–679.
- ¹²Bos, F., van Oudheuden, B., and Bijl, H., "Radial basis function based mesh deformation applied to simulation of flow around flapping wings," *Computers & Fluids*, Vol. 79, 2013, pp. 167–177.
- ¹³Rozenberg, Y., S., A., and Bnife, G., "Fluid Structure Interaction Problems in Turbomachinery Using RBF Interpolation and Greedy Algorithm," *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition, Volume 2B: Turbomachinery*, ASME, 2014.
- ¹⁴Cordero-Gracia, M., Gomez, M., Ponsin, J., and Valero, E., "An interpolation tool for aerodynamic mesh deformation problems based on octree decomposition," *Aerospace Science and Technology*, Vol. 23, 2012, pp. 93–107.
- ¹⁵Sheng, C. and Allen, C. B., "Efficient Mesh Deformation Using Radial Basis Functions on Unstructured Meshes," *AIAA Journal*, Vol. 51, No. 3, 2013, pp. 707–720.

- ¹⁶Gillebaart, T., Blom, D., van Zuijlen, A., and Bijl, H., "Adaptive Radial Basis Function Mesh Deformation using Data Reduction," *Journal of Computational Physics*, Vol. 321, 2016, pp. 1–20.
- ¹⁷Gillebaart, T., van Zuijlen, A., and Bijl, H., "Radial basis function mesh deformation including surface orthogonality," AIAA Paper no. AIAA-2016-1674, AIAA SciTech, San Diego, CA, Jan 2016.
- ¹⁸Poirier, V. and Nadarajah, S., "Efficient Reduced-Radial Basis Function Mesh Deformation within an Adjoint-based Aerodynamic Optimization Framework," *Journal of Aircraft*, Vol. 53, No. 6, 2016, pp. 1905–1921.
- ¹⁹Hartwich, P. M. and Agrawal, S., "Method for perturbing multiblock patched grids in aeroelastic and design optimisation applications," *AIAA*, 1997.
- ²⁰Allen, C., "Aeroelastic Computations using Algebraic Grid Motion," *The Aeronautical Journal*, Vol. 106, No. 1064, 2002, pp. 559–570.
- ²¹Eriksson, L. E., "Generation of Boundary-Conforming Grids Around Wing-Body Configurations Using Transfinite Interpolation," *AIAA Journal*, Vol. 20, No. 10, 1982.
- ²²Potsdam, M. A. and Guruswamy, G. P., "A parallel multiblock mesh movement scheme for complex aeroelastic applications," *AIAA*, 2001.
- ²³Batina, J. T., "Unsteady Euler aerofoil solutions using unstructured dynamic meshes," *AIAA 27th Aerospace Sciences Meeting and Exhibit*, AIAA, 1989.
- ²⁴Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional springs for two-dimensional dynamic unstructured fluid meshes," *Computer methods in applied mechanics and engineering*, Vol. 163, 1998, pp. 231–245.
- ²⁵Blom, F., "Considerations on the Spring Analogy," *International Journal for Numerical Methods in Fluids*, Vol. 32, 2000, pp. 647–668.
- ²⁶Sheta, E., Yang, H., and Habchi, S. D., "Solid Brick Analogy for Automatic Grid Deformation for Fluid-Structure Interaction," AIAA Paper no. AIAA-2006-3219, 2006.
- ²⁷Kamakoti, R. and Shyy, W., "Fluid-structure interaction for aeroelastic applications," *Progress in Aerospace Sciences*, Vol. 40, 2004, pp. 535–558.
- ²⁸Loehner, R. and Yang, C., "Improved ALE Mesh Velocities for Moving Bodies," *Communications in Numerical Methods in Engineering*, Vol. 12, 1996, pp. 599–608.
- ²⁹Bau, J., Luo, H., Loehner, R., Goldberg, E., and Feldhun, A., "Application of Unstructured Moving Body Methodology to the Simulation of Fuel Tank Separation from an F-16 Fighter," AIAA Paper no. AIAA-1997-0166, 1997.
- ³⁰Tysell, L., "Grid Deformation of 3D Hybrid Grids," *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, 2002.
- ³¹Helenbrook, B., "Mesh Deformation using the Biharmonic Operator," *International Journal for Numerical Methods in Engineering*, Vol. 56, 2003, pp. 1007–1021.
- ³²Liu, X., Qin, N., and Xia, H., "Fast dynamic grid deformation based on Delaunay graph mapping," *Journal of Computational Physics*, Vol. 211, 2006, pp. 405–423.
- ³³Luke, E., Collins, E., and Blades, E., "A fast mesh deformation method using explicit interpolation," *Journal of Computational Physics*, Vol. 231, 2012, pp. 586–601.
- ³⁴Witteveen, J. A. S., "Explicit and Robust Inverse Distance Weighting Mesh Deformation for CFD," *AIAA 48th Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, AIAA, 2010.
- ³⁵Zhou, X. and Li, S., "A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing," *Journal of Computational Physics*, Vol. 235, 2013, pp. 199–215.
- ³⁶Zhou, X. and Li, S., "A novel three-dimensional mesh deformation method based on sphere relaxation," *Journal of Computational Physics*, Vol. 298, 2015, pp. 320–336.
- ³⁷Buhmann, M. D., "Radial basis functions," *Acta Numerica*, Vol. 9, 2000, pp. 1–38.
- ³⁸Beckert, A. and Wendland, H., "Multivariate interpolation for fluid-structure-interaction problems using radial basis functions," *Aerospace Science and Technology*, Vol. 5, No. 125-134, 2001.
- ³⁹Wendland, H., "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in Computational Mathematics*, Vol. 4, 1995, pp. 389–396.
- ⁴⁰Floater, M. S. and Iske, A., "Multistep scattered data interpolation using compactly supported radial basis functions," *Journal of Computational and Applied Mathematics*, Vol. 73, 1996, pp. 65–78.
- ⁴¹Wendland, H., "Multiscale analysis in Sobolev spaces on bounded domains," *Numerical Math*, Vol. 116, 2010, pp. 493–517.
- ⁴²Siebert, B. and Dulikravich, G., "Grid Generation using A Posteriori Optimization with Geometrically Normalised Functions," AIAA Paper No. AIAA-1990-3048, Applied Aerodynamics Conference, Portland, OR, 1990.
- ⁴³Allwright, S., "Multi-Discipline Optimisation in Preliminary Design of Commercial Transport Aircraft," *Computational Methods in Applied Sciences, ECCOMAS*, 1996, pp. 523–526.
- ⁴⁴Allen, C. B., "Towards Automatic Structured Multiblock Mesh Generation using Improved Transfinite Interpolation," *International Journal for Numerical Methods in Engineering*, Vol. 74, No. 5, 2008, pp. 697–733.