Journal of Computational Physics 345 (2017) 732–751

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

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, BS8 1TR, UK

ARTICLE INFO

Article history:
Received 19 December 2016
Received in revised form 18 May 2017
Accepted 24 May 2017
Available online 31 May 2017

Keywords:
Mesh deformation
Radial basis functions
Data reduction methods
Multiscale methods
Exact surface recovery
Numerical simulation

ABSTRACT

Radial basis function (RBF) interpolation is popular for mesh deformation due to robustness and generality, but the cost scales with the number of surface points sourcing the deformation as $O(N_s^3)$. Hence, there have been numerous works investigating efficient methods using reduced datasets. However, although reduced-data methods are efficient, they require a secondary method to treat an error 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 which captures global and local motions at multiple scales using *all* the surface points, and so no correction stage is required; all surface points are used and a single interpolation built, but the cost and conditioning issues associated with RBF methods are eliminated. Moreover, the sparsity introduced is exploited using a wall distance function, to further reduce the cost. The method is compared to an efficient greedy method, and it is shown mesh quality is always comparable with or better than with the greedy method, and cost is comparable or cheaper at all stages. Surface mesh preprocessing is the dominant cost for reduced-data methods and this cost is reduced significantly here: greedy methods select points to minimise interpolation error, requiring repeated system solution and cost $O(N_{red}^4)$ to select N_{red} points; the multiscale method has no error, and the problem is transferred to a geometric search, with cost $O(N_s \log(N_s))$, resulting in an eight orders of magnitude cost reduction for three-dimensional meshes. Furthermore, since the method is dependent on geometry, not deformation, it only needs to be applied once, prior to simulation, as the mesh deformation is decoupled from the point selection process.

1. 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 interaction [1], aerodynamic optimisation [2] and articulated rotor simulations [3]. 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

* Corresponding author.
E-mail address: c.b.allen@bristol.ac.uk (C.B. Allen).

http://dx.doi.org/10.1016/j.jcp.2017.05.042
0021-9991/

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 [4].

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. In [1] and [5] 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.

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 [6,1]. However, the mesh deformation requires solution of a linear system the size of the moving control point cloud, and so the full method quickly becomes very expensive for large meshes. Hence the implementation is commonly approximated using some kind of data-reduction scheme.

Rendall and Allen [4,7] developed effective methods to improve the efficiency of RBF interpolation, by selecting a reduced set of control points based on minimising interpolation error at the non-selected 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 [3], and the surrounding volume mesh deformed accordingly. The work in [1,4,7] has in fact led to a large number of further improvements, focused on both efficient point selection methods and alternative methods to treat the correction vector, see for example [8–15]. There has also been work investigating whether the reduced points selected can be selected once, before an unsteady simulation, or need to reselected at every time level [16], adding an adaptive point selection method [17], and combining reduced data points with orthogonality improvements [18]. A further work worthy of mention is that due to Poirier and Nadarajah [19], 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.

In terms of operations, RBF mesh deformation methods need, at every time step, a system solution stage which scales with number of control points cubed, and a volume mesh update stage which scales with control points $\times$ volume points. Hence, the reduced-point methods aim to reduce both of these stages, however, reduced-point methods require an expensive 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 exactly 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.

2. Mesh deformation

Numerous mesh deformation schemes have been developed, often depending on the mesh type or the particular application; an interesting review paper was recently published by Selim and Koomullil [20]. Some of the more simple mesh deformation schemes, see for example [21,22], used a uni-parametric interpolation [23] 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 [24], for example using the spring analogy for block edges.

The spring analogy, developed by Batina [25], 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. Farhat [26] built on the method by introducing torsional stiffness to alleviate mesh crossover problem, and in Blom [27] this torsional spring was shown to be essential for moving viscous meshes. Sheta et al. [28] have also reconsidered the formulation to use solid structural elements in an effort to prevent cell inversion. The method is robust and accurate, but computationally expensive for large grids [29]. There are also PDE solution approaches, usually involving an elliptic problem solution, see for example Loehner [30,31]. Grid quality can be improved by solving a bi-harmonic set of equations that also preserve orthogonality [32,33].

The spring analogy, and PDE-based methods, are expensive to solve and fail to guarantee mesh quality for large deformations. In contrast, non-iterative methods, which generally take the form of a multivariate interpolation, are robust

and do not usually rely on connectivity information [9]. Liu et al. [34] developed a Delaunay graph interpolation scheme boasting high computational efficiency, however boundary mesh quality and robustness were poor for large deformations and rotations [9]. Allen [5] 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 were also presented in [35] and [36] and methods have also been presented using disk and then sphere relaxation methods [37,38].

Recently there has been increased interest in radial basis function (RBF) interpolation for mesh deformation [6,1,2]. RBF interpolation is a popular tool for general multivariate interpolation, see [39], 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 are highlighted (mesh/solver independence, parallelisable), 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 [4]. Wang 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 Amoigon [2] 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 [4], demonstrating two orders of magnitude cost reduction for a large mesh case 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 improved [3] by adding a correction step to recover exact displacements. A series of different data-reduction schemes followed, primarily focusing on the efficiency and error of the point-selection process, see for example [2–4,7–14,16,15,17,19]; 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. However, reduced-point methods require a surface mesh preprocessing stage to select the point set, and these are usually selected to minimise the interpolation error at the non-selected points; this requires repeated system solution and a cost of $O(N_{red}^4)$. Hence, the rationale for the method presented here 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 similar or reduced cost compared to greedy-type methods. With no requirement to minimise error, the point selection problem can be transferred to a surface point geometric search, with a significantly reduced cost of $O(N_s \log(N_s))$. The multiscale formulation developed means that although all surface points are used and a single interpolation is built, the linear system to be solved is significantly smaller than that of the full method, and the system conditioning issues are eliminated. Furthermore, unlike the greedy-based methods where the points selected depend on the exact deformation, the point selection for the new method is based purely on geometry, and so is decoupled from the deformation, meaning the method only needs to be applied once, prior to any simulation.

3. 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.

3.1. 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 [39,40].

A radial function is a function that takes the form $\phi(\mathbf{x}, \mathbf{x}_0) = \phi(\|\mathbf{x}_0 - \mathbf{x}\|)$ where $\|.\|$ represents the vector norm, usually Euclidean, and $\mathbf{x}_0$ is the RBF 'centre'. Given a set of discrete data points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ 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:

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

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

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

Name	Definition
Norm	$r = (\|\mathbf{x}_i - \mathbf{x}_j\|)/R$
C^0	$\phi(r) = (1-r)^2$
C^2	$\phi(r) = (1-r)^4(4r+1)$
C^4	$\phi(r) = (1-r)^6(35r^2+18r+3)/3$

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:

$$\mathbf{f} = \mathbf{\Phi}\boldsymbol{\gamma} \tag{2}$$

Where

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

Here, $\phi_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. The size and structure of the interpolation matrix $\mathbf{\Phi}$ 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 [40]. 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. If the support radius is large in comparison to the spatial density of the data, then close mesh points present conditioning problems. Wendland [41] 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. In a similar way to support radius, higher order functions give better interpolation accuracy but with poorer matrix conditioning.

In [3] it was shown that the existence of a smooth first derivative of the basis function is desirable for preservation of orthogonality. C^2 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 [1,7], and C^2 is used here.

3.2. 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 [6,1]. In their demonstration of RBF mesh motion, de Boer et al. [6] observed the simplicity of implementation and the superior mesh quality when compared with the spring analogy [25]. Jakobsson and Amoignon applied 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 $\mathbf{\Phi}$ 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:

$$\mathbf{u}_s^x = \mathbf{\Phi}\boldsymbol{\gamma}^x, \quad \mathbf{u}_s^y = \mathbf{\Phi}\boldsymbol{\gamma}^y, \quad \mathbf{u}_s^z = \mathbf{\Phi}\boldsymbol{\gamma}^z \tag{4}$$

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

$$\mathbf{u}_v^{x/y/z} = \mathbf{\Psi}\boldsymbol{\gamma}^{x/y/z} \tag{5}$$

Where

$$\mathbf{\Psi} = \begin{pmatrix} \phi_{v1,s1} & \phi_{v1,s2} & \cdots & \phi_{v1,N_s} \\ \phi_{v2,s1} & \phi_{v2,s2} & \cdots & \phi_{v2,N_s} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N_v,s1} & \phi_{N_v,s2} & \cdots & \phi_{N_v,N_s} \end{pmatrix} \tag{6}$$

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	–	$O(N_s N_v)$

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

Mesh points		Storage of **H** memory (GB)	Preprocessing or at each stage		Each stage mesh update
N_s	N_v (M)		Condition No.	System solution	
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	–	$O(10^{10})$	$O(10^9)$

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

3.3. 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 requires no preprocessing stage prior to simulation, and no significant data storage, and consists of two steps: 1) Solve for field vectors $\boldsymbol{\gamma}^{x/y/z}$; 2) Update volume mesh points. In this way, the operation-intensive method has no significant memory costs since the matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ do not need to be explicitly generated and stored. However, as suggested by the name, the operation-intensive method requires many processor operations per mesh deformation step, since the $N_s \times N_s$ matrix requires solution, with cost $O(N_s^3)$. 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, the system solution is performed in a preprocessing stage, and a single $N_v \times N_s$ transfer matrix $\mathbf{H}$ can be defined, as in equation (8).

$$\mathbf{u}_v = \mathbf{H}\mathbf{u}_s \tag{7}$$

Where

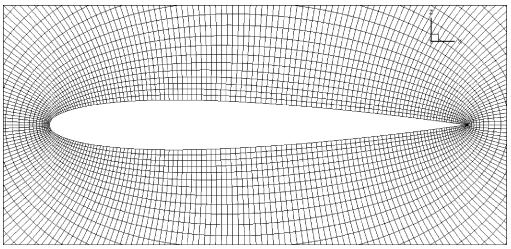
$$\mathbf{H} = \boldsymbol{\Psi}\boldsymbol{\Phi}^{-1} \in \mathbf{R}^{N_v \times N_s} \tag{8}$$

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 operations and memory complexity for each stage of the RBF mesh deformation, using the simple operation counts discussed above.

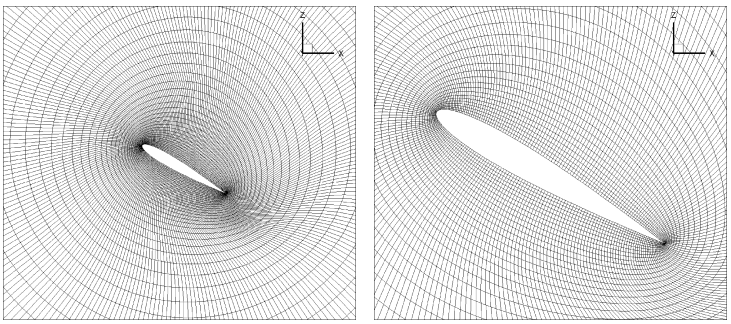
It is clear that the size of the surface mesh (N_s) is prominent in the overall cost of the RBF method. Moreover, solving the system scales with N_s^3, hence full RBF mesh deformation quickly becomes infeasible for large meshes. In addition to higher costs, increased mesh sizes also introduce numerical instability in the linear system.

3.4. 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 Fig. 1(a). Fig. 1 also shows the resulting deformed mesh using Wendland's C^2 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. These data are evaluated using the simple operation scalings presented in Table 2. 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)

Fig. 1. NACA0012 Rigid rotation case, full RBF method, $R = 4$ chords.

3.5. 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 [7,10,16]. 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 [4]. 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 algorithm [7].

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

4. 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 [42]. 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 [43]. 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 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 [2,42]. The separation distance Q of a set of points Ω is defined as the radius of the largest empty sphere centred 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:

$$Q = \max_{i\in\Omega} \min_{j\in\Omega, i\neq j} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{9}$$

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*.

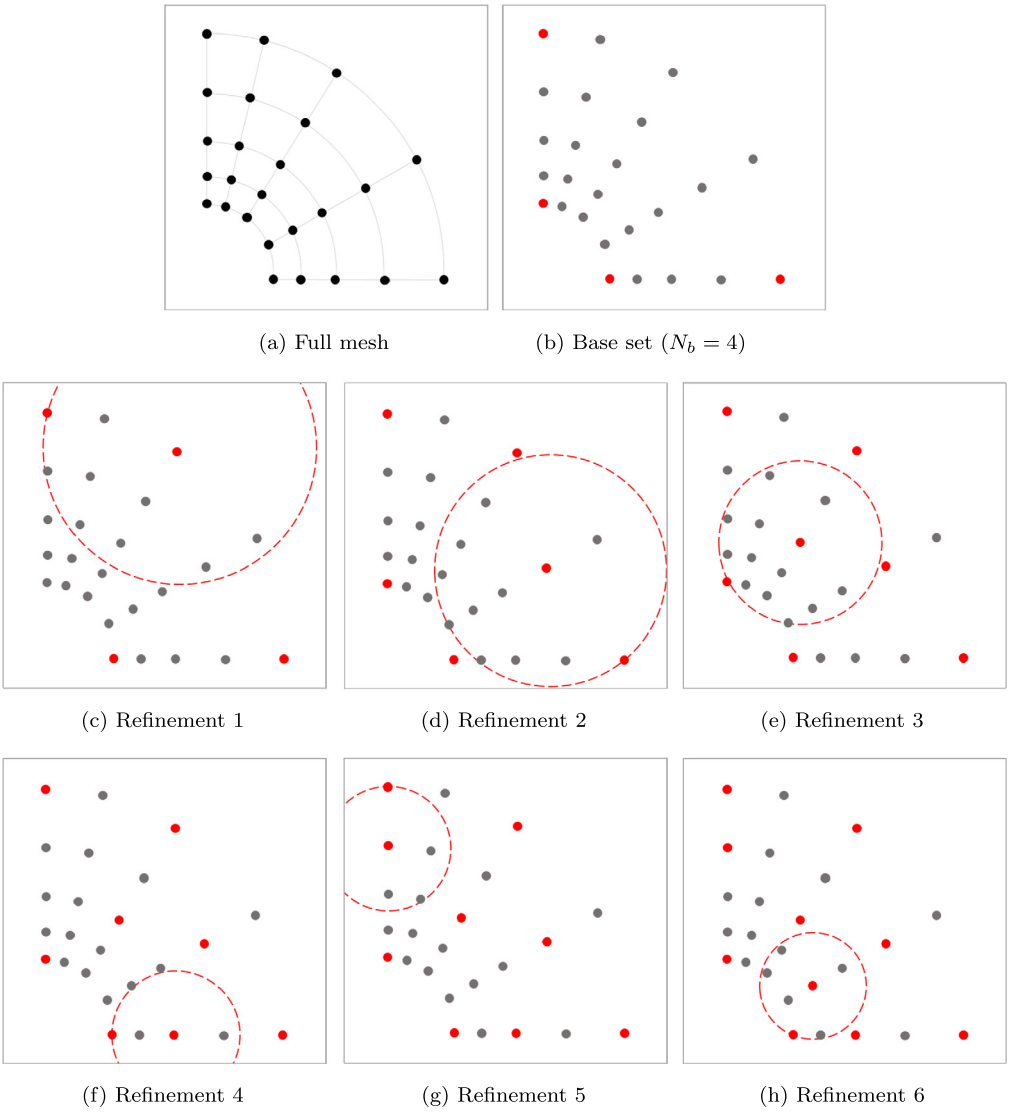
The only user-specified parameters here are the base-set support radius and the number of points in the base set. Fig. 2 illustrates an example point refinement process using separation distance, with the multiscale nature of the method clear; active points are shown as red, and points not yet included grey.

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 up to 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 structure:

$$\mathbf{\Phi} = \left(\begin{array}{cccc|cccc} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,N_b} & 0 & 0 & \cdots & 0 \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,N_b} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{N_b,1} & \phi_{N_b,2} & \cdots & \phi_{N_b,N_b} & 0 & 0 & \cdots & 0 \\ \hline \phi_{N_b+1,1} & \phi_{N_b+1,2} & \cdots & \phi_{N_b+1,N_b} & \phi_{N_b+1,N_b+1} & 0 & \cdots & 0 \\ \phi_{N_b+2,1} & \phi_{N_b+2,2} & \cdots & \phi_{N_b+2,N_b} & \phi_{N_b+2,N_b+1} & \phi_{N_b+2,N_b+2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{N_s,1} & \phi_{N_s,2} & \cdots & \phi_{N_s,N_b} & \phi_{N_s,N_b+1} & \phi_{N_s,N_b+2} & \cdots & \phi_{N_s,N_s} \end{array}\right) \tag{10}$$

And hence, the linear system to solve:

$$\begin{pmatrix} \mathbf{u}_b^x \\ \mathbf{u}_r^x \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}_b & \mathbf{0} \\ \mathbf{\Psi}_r & \mathbf{L} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^x \\ \boldsymbol{\beta}^x \end{pmatrix} \tag{11}$$



(a) Full mesh (b) Base set ($N_b = 4$)

(c) Refinement 1 (d) Refinement 2 (e) Refinement 3

(f) Refinement 4 (g) Refinement 5 (h) Refinement 6

Fig. 2. Multiscale refinement process.

$\mathbf{\Phi}_b$ is the $N_b \times N_b$ interpolation matrix for the base set, which is positive definite for suitable choice of RBF $\phi(\|.\|)$. $\mathbf{\Psi}_r$ 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 $\boldsymbol{\alpha}^x$, which are then substituted into the lower system to evaluate the residual for the refinement points: $\mathbf{\Delta} = \mathbf{u}_b^x - \mathbf{\Psi}_r \boldsymbol{\alpha}^x$. Finally, since $\mathbf{L}$ is a lower triangular matrix, the lower system is itself solved by forward substitution for $\boldsymbol{\beta}^x$. The volume mesh is then updated using the coefficient vectors $\boldsymbol{\alpha}^x$ and $\boldsymbol{\beta}^x$. Equation (11) is solved exactly and hence the original surface displacements are recovered. The only linear system to solve is that involving $\mathbf{\Phi}_b$, and since $\mathbf{\Phi}_b$ is only a coarse subset of the full surface mesh, then the cost and conditioning issues of the full RBF method are eliminated.

5. 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.

5.1. Surface mesh preprocessing: multiscale

Once the base-set problem 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.

Algorithm 2 Control point sequencing.

procedure
init:
  $inactiveList \leftarrow [1, N_s]$ ▷ All point indices start on inactive list
  for $i = 1, N_s$ **do**
    $separation(i) \leftarrow LARGE$ ▷ Initialise separation distance list
  end for
  $N_{active} \leftarrow 0$
  $activeList(N_{active}) \leftarrow k$ ▷ Add first point to active list
  $inactiveList \leftarrow inactiveList \backslash \{k\}$
  $N_{active} \leftarrow N_{active} + 1$
main:
  while $N_{active} < N_s$ **do**
    for i **in** $\{inactiveList\}$ **do** ▷ Update separation distances
      $r_{ik} \leftarrow \|x_i - x_k\|$ ▷ Distance to latest added point
      if $r_{ik} < separation(i)$ **then**
        $separation(i) \leftarrow r_{ik}$
      end if
    end for
    $k \leftarrow \arg\max_k(separation(k))$ ▷ Find point with largest separation distance
    $activeList(N_{active}) \leftarrow k$ ▷ Move to active list
    $inactiveList \leftarrow inactiveList \backslash \{k\}$
    if $N_{active} \leq N_b$ **then** ▷ Save support radius value
      $radii(N_{active}) \leftarrow r_{base}$
    else
      $radii(N_{active}) \leftarrow separation(k)$
    end if
    $N_{active} \leftarrow N_{active} + 1$
  end while
output:
  return *activeList, radii* ▷ Ordered control point indices and support radii
end procedure

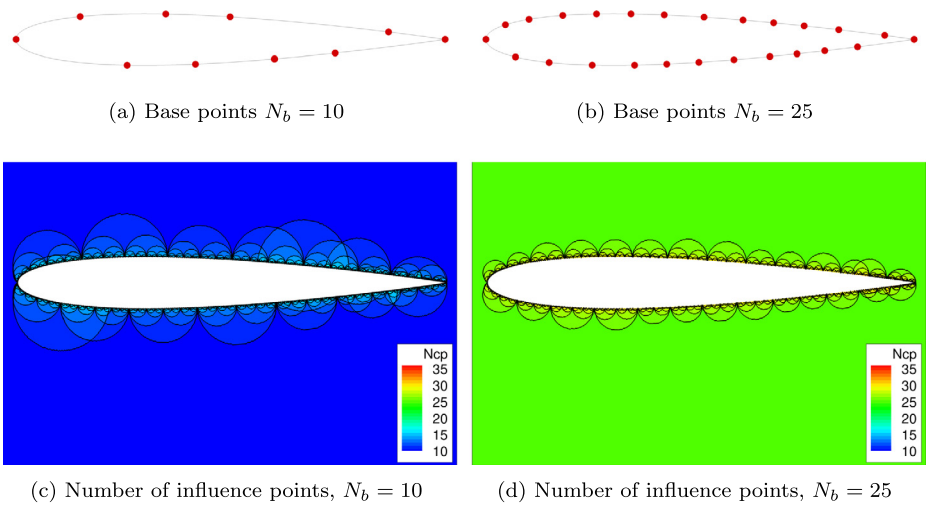
This algorithm operates on a point cloud of N_s points, and the main loop of the sequencing algorithm has $N_s - 1$ iterations. At each iteration, the algorithm calculates the separation distance of all remaining points with respect to the processed points and chooses the point with the largest separation distance to add to the processed list.

The surface mesh array can either be re-ordered, costing $O(N_s)$ time, or an ordered list of indices can be stored, costing $O(N_s)$ memory. An array of support radii is also produced costing $O(N_r)$ memory. An efficient tree search scheme is used for surface point identification, with cost of $O(N_s \log(N_s))$ to build, and cost of $O(\log(N))$ each time to interrogate N points. Hence the total number of operations is given by:

$$N^{op} = O(N_s \log(N_s)) + O(N_s) + O(N_r) + \sum_{i=1}^{N_s - 1} \log(N_s - i) = O(N_s \log(N_s)) \tag{12}$$

5.2. Surface mesh preprocessing: greedy

Algorithm 1 is an example of a greedy point reduction method. The algorithm takes the surface displacements as 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



(a) Base points $N_b = 10$ (b) Base points $N_b = 25$

(c) Number of influence points, $N_b = 10$ (d) Number of influence points, $N_b = 25$

Fig. 3. Base points and number of control points influencing volume mesh points.

N_{red} iterations, where at each iteration a linear system is both built and solved, and the interpolant evaluated on the remaining points and a search performed for the point with maximum error:

$$N^{op} = \sum_{i=1}^{N_{red}} \Big(O(i^2) + O(i^3) + O(i(N_s - i)) + O(N_s - i) \Big) = O(N_{red}^4) \tag{13}$$

for $N_{red} > \sqrt{N_s}$, which is always the case for realistic mesh sizes.

5.3. 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. 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 Fig. 3 which shows the number of actively influencing surface points at each volume mesh point, for N_b of 10 and 25, clearly demonstrating the multiscale method and efficiency; the base points selected are also shown. 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 $\boldsymbol{\Psi}$ of equation (6) will be sparse. This preprocessing stage can be implemented using any suitable wall distance function, to efficiently categorise each volume point and, for those points in the third category, to find efficiently the additional influencing indices using the tree search. For completeness, the cost of the wall-distance calculation has been included here. The search tree has already been built for the surface preprocessor, and so the wall distance cost is represented by an estimated cost of $O(N_v \log(N_s))$. 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 operations for volume mesh preprocessing is:

$$N^{op} = O(N_v \log(N_s)) + N_{v1} + N_{v2} + O(N_{v3} \log(N_r)) \tag{14}$$

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 is:

$$N^{op} = O(N_v \log(N_s)) \tag{15}$$

Note, $N_{v1} + N_{v2} + N_{v3} = N_v$.

5.4. 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 and most efficient approach is used here; a single 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:

$$\Delta \mathbf{x}_v = \phi(\|\mathbf{x}_v - \mathbf{x}_n\|/R_c)(\Delta \mathbf{x}_n^{exact} - \Delta \mathbf{x}_n^{interp}) \tag{16}$$

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

$$R_c = \max_{i,j \in N_{red}\ i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{17}$$

This type of correction may also be thought of as a highly localised single point RBF interpolation.

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. Including the same approximate distance function cost as with the multiscale method, the search tree build cost must be included here, and using the same search costs for the nearest neighbour point identification, the volume preprocessing operations are:

$$N^{op} = O(N_s \log(N_s)) + O(N_v \log(N_s)) + N_{v1} + O(N_{v2} \log(N_s)) \tag{18}$$

or, similarly to multiscale

$$N^{op} = O(N_v \log(N_s)) \tag{19}$$

Note in this case, $N_{v1} + N_{v2} = N_v$.

The multiscale and greedy costs will never be exactly the same, as the point distribution for greedy selection and multiscale base set will not be the same for $N_b = N_{red}$ due to different criteria, but R_c will normally be very similar to r_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 and N_{v2} for greedy will always be greater than N_{v2} for multiscale.

5.5. Solve and update steps: multiscale

The solution stage requires solution of the $N_b \times N_b$ system. The second stage of solving equation (11) involves forward substitution of the $N_r \times N_r$ lower triangular matrix $\boldsymbol{L}$ 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, $\boldsymbol{L}$ 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 $\boldsymbol{L}$ can be determined. If N_{nz} is the number of non-zero elements in matrix $\boldsymbol{L}$, 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:

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

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

$$N^{op} = N_b N_{v2} + N_{nz-v3} \tag{21}$$

where N_{nz-v3} is the number of non-zero influences for volume mesh points in the third category (close to the surface). Since N_{v2} is some fraction of N_v, and if N_{nz-v3} is sufficiently small then:

$$N^{op} = O(N_v N_b) \tag{22}$$

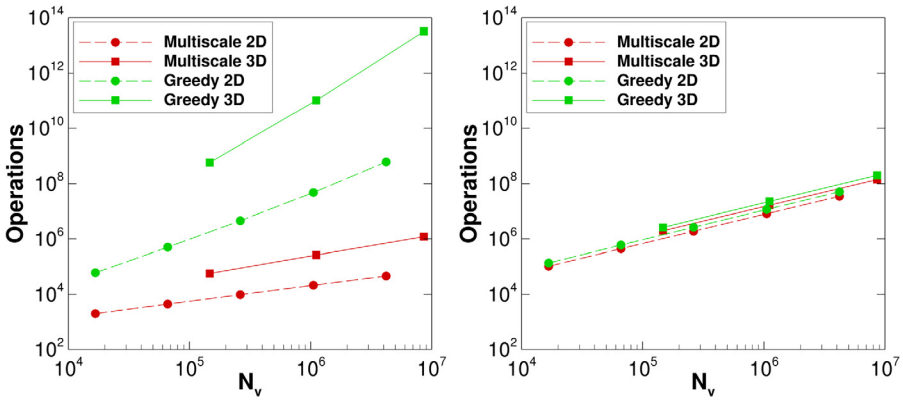
5.6. Solve and update steps: greedy

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

$$N^{op} = O(N_{red}^3) \tag{23}$$

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. Again, only category 2 points are moved, so as with the multiscale method this step will also be some fraction of N_v, hence:

$$N^{op} = N_{red} N_{v2} + N_{v2} = O(N_v N_{red}) \tag{24}$$



(a) Surface mesh preprocessing (b) Volume mesh preprocessing

Fig. 4. Comparison of cost with an efficient greedy RBF method.

5.7. 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 $257 \times 65, 513 \times 129, 1025 \times 257, 2049 \times 513, 4097 \times 1025$. A structured multiblock mesh for a wing is considered as the three-dimensional problem, with mesh sizes: $N_v = 0.15M, N_s = 3,881$; $N_v = 1.11M, N_s = 15,441$; $N_v = 8.62M, N_s = 61,601$. For all the data here, $N_b = N_{red} = 0.05N_s$, although it is shown later that this many points are not required for finer meshes.

Fig. 4 shows the cost of the preprocessing stages for surface and volume for the two methods. These data are the exact operation counts for these meshes, including the effect of dimensionality on the search costs. The system solution and update costs at each deformation stage are not presented, as these are dominated by the system solution which is a simple $O(N^3)$ cost, and hence is exactly the same for $N_b = N_{red}$. 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 extremely cheap.

6. 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.

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 [44]. For a structured mesh, consider a parametric plane, and the orthogonality at any point in that plane can be defined using the vectors to the four neighbour points:

$$q = \frac{1}{4}\left\{\frac{(\mathbf{v_1}.\mathbf{v_2})^2}{\mathbf{v_1}^2\mathbf{v_2}^2} + \frac{(\mathbf{v_2}.\mathbf{v_3})^2}{\mathbf{v_2}^2\mathbf{v_3}^2} + \frac{(\mathbf{v_3}.\mathbf{v_4})^2}{\mathbf{v_3}^2\mathbf{v_4}^2} + \frac{(\mathbf{v_4}.\mathbf{v_1})^2}{\mathbf{v_4}^2\mathbf{v_1}^2}\right\} \tag{25}$$

Similar arguments can be used in each parametric direction, to give a local orthogonality value for each grid point in a three-dimensional mesh as:

$$q_{i,j,k} = 1.0 - \frac{q_i + q_j + q_k}{3} \tag{26}$$

Orthogonality is often taken as zero to be perfectly orthogonal, i.e. all (θ) values are $\pi/2$, 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. (For the two-dimensional cases shown here, $q_{i,j} = 1.0 - q_k$.)

A rigid rotation of 30° about the origin is applied to the surface mesh. Mesh deformation is first calculated using the full RBF method, as a baseline for orthogonality change. Fig. 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.

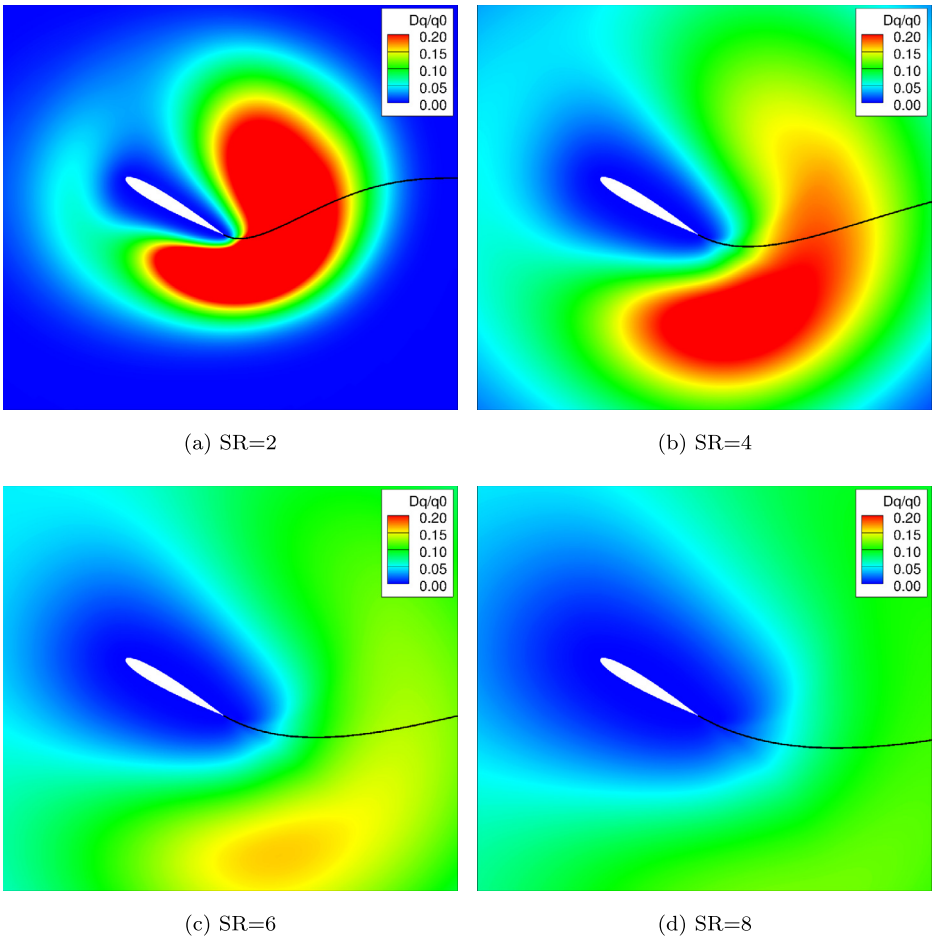


Fig. 5. Rigid rotation – full RBF method for varying support radius (SR, in chord units).

The rigid-rotation case was then tested using the new multiscale method. Fig. 6 shows the change in orthogonality for four values of N_b. The multiscale method clearly dissipates the deformation field away from the surface, matching that of the full RBF method well. Fig. 7 shows change in orthogonality now 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.

7. Three-dimensional application

The Brite-Euram MDO (multidisciplinary optimisation) wing of semi-span 35 m is used here as the surface geometry [45, 46]. The mean chord for this geometry is 10.26 m. Three structured multi-block volume meshes have been generated [47], as defined in Table 4. Fig. 8 shows the domain and block boundaries, the farfield mesh, and a view of the surface mesh, wake slit and a spanwise plane; note this is the same plane shown in orthogonality plots later. This is the 1.11 million point mesh.

An exaggerated deformation is considered. A detailed structural mesh is also available for this geometry, and Fig. 9 shows the CFD surface mesh points, and the finite-element structural model nodes. Deformation of the 18th structural mode is considered, and this mode is scaled to give a 3m maximum deflection of the wing tip nodes; this deformation is transfered to the CFD surface points using RBF interpolation [1]. Fig. 10 shows two views of the underformed 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 are the average q values over all cells in each mesh, and average q values for the first two grid planes away from the wing surface, to give a global and local orthogonality measure.

7.1. 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

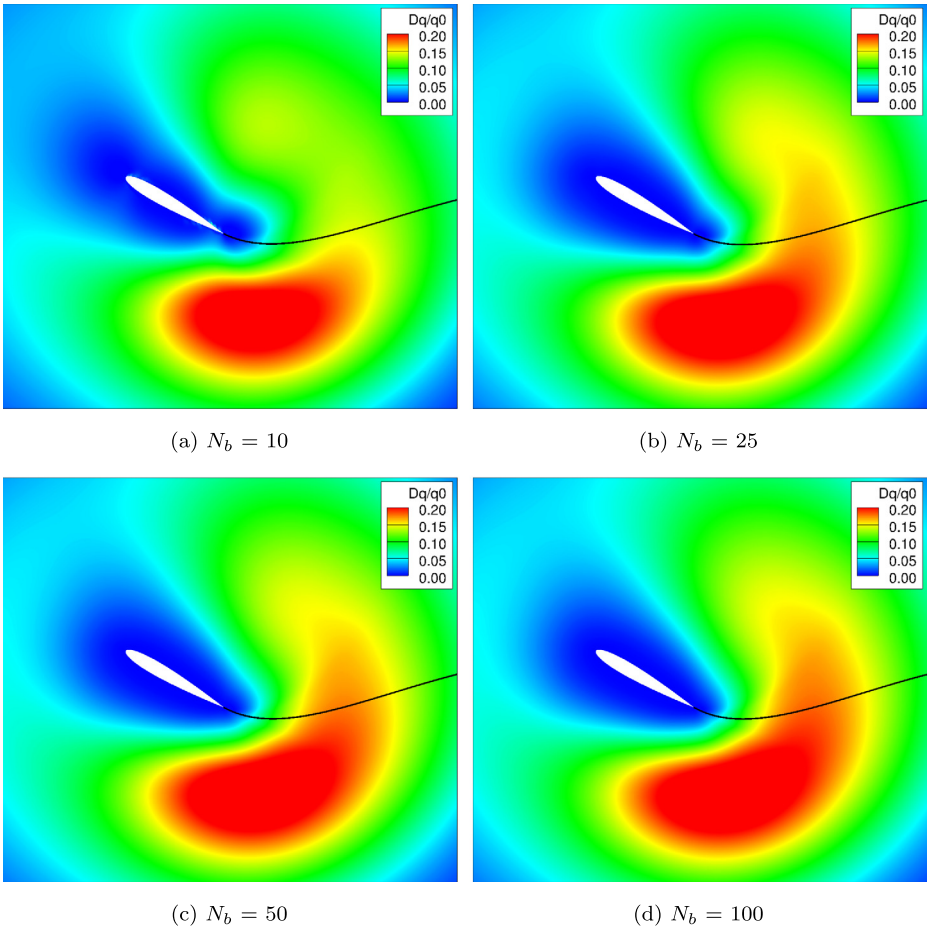


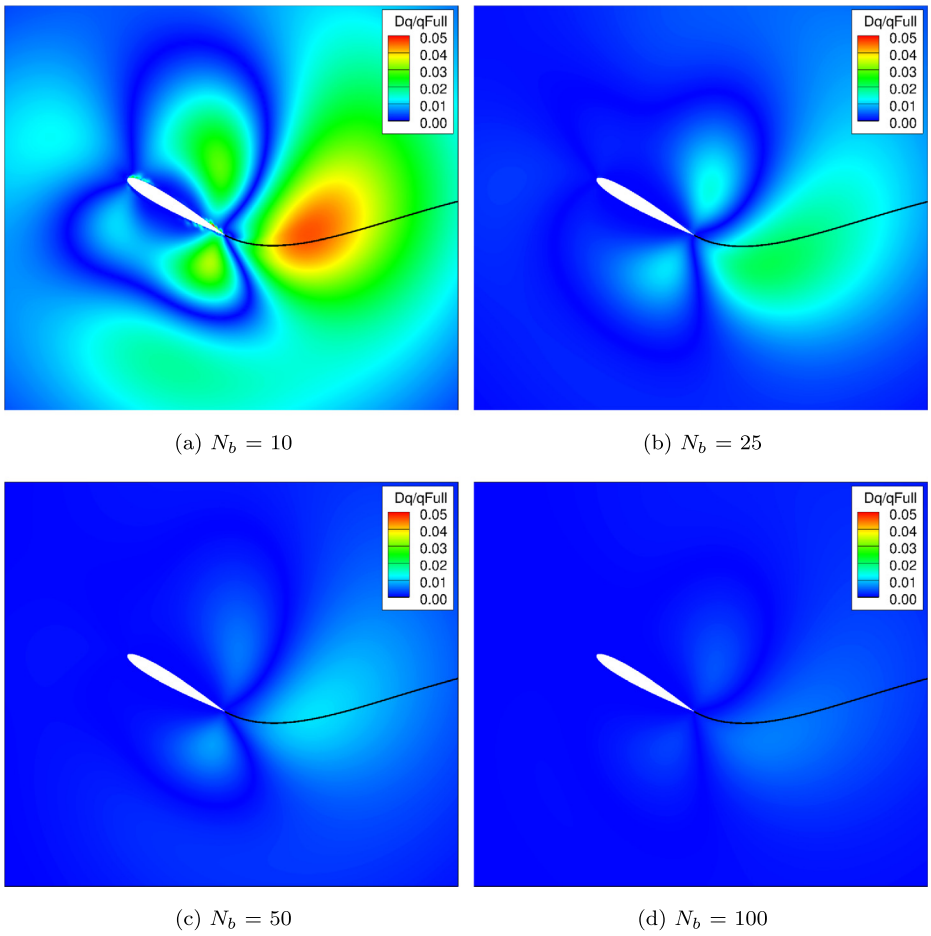
Fig. 6. Rigid rotation – multiscale RBF method for varying number of base set points.

Table 4
Transport wing: mesh densities and undeformed orthogonality.

N_s	N_v (M points)	q^0	q^0_{surf}
3,881	0.15	0.8921	0.9105
15,441	1.11	0.8900	0.9114
61,601	8.62	0.8888	0.9116

selection. Instead, an approach is adopted 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 Fig. 11a; points are coloured red and blue to represent upper and lower surface points. 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.

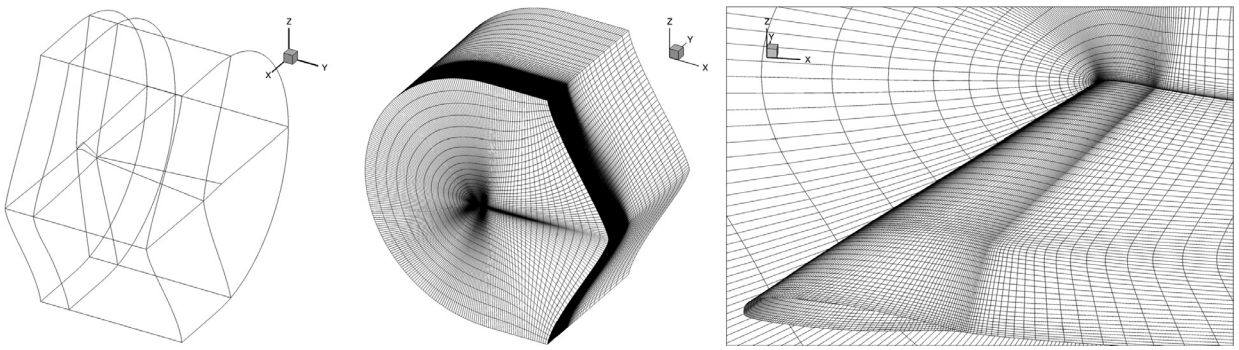
First, the space-filling method is modified to include some consideration of local mesh density. Whenever the separation distance of an inactive point is updated (Algorithm 2: line 16) the inactive point is also recorded as a 'child' of the closest active point. Then when selecting the next point to add to the active list (line 19), 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 Fig. 11b, again with upper and lower surface points identified.



(a) $N_b = 10$ (b) $N_b = 25$

(c) $N_b = 50$ (d) $N_b = 100$

Fig. 7. Rigid rotation – multiscale RBF method for varying number of base set points.



(a) Domain (b) Farfield mesh (c) Mesh planes

Fig. 8. One million cell mesh.

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

$$\|\mathbf{x} - \mathbf{x}_i\|_{\mathbf{n}_i} = \psi_i(\mathbf{x}) \cdot \|\mathbf{x} - \mathbf{x}_i\| \tag{27}$$

$$\psi_i(\mathbf{x}) = \begin{cases} 1 + a(\hat{d})^2, & \text{if } \hat{d} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{28}$$

Where

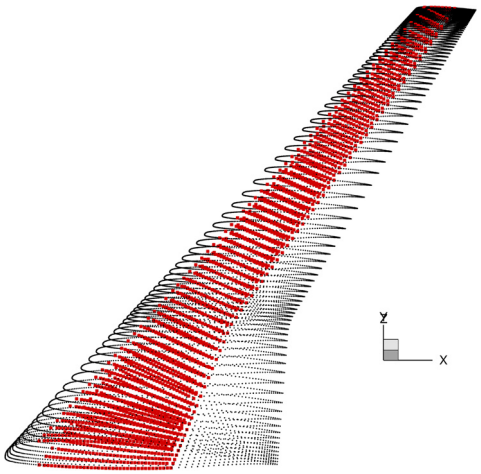


Fig. 9. MDO wing. Red – finite element nodes. Black – CFD surface points. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

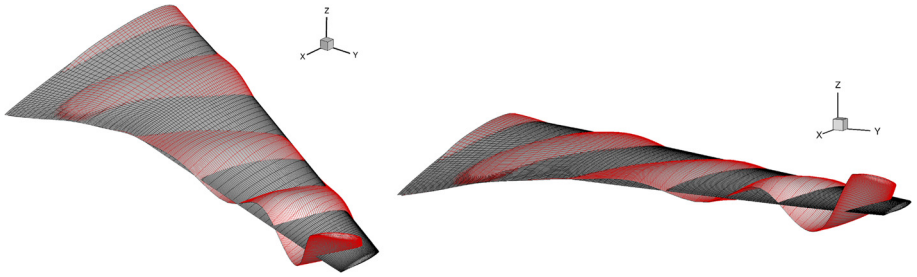
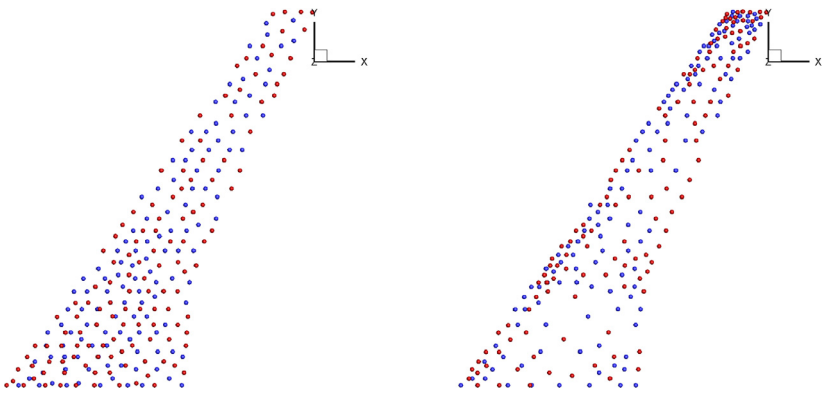


Fig. 10. MDO wing. Red – mode 18 deflection. Black – original surface. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

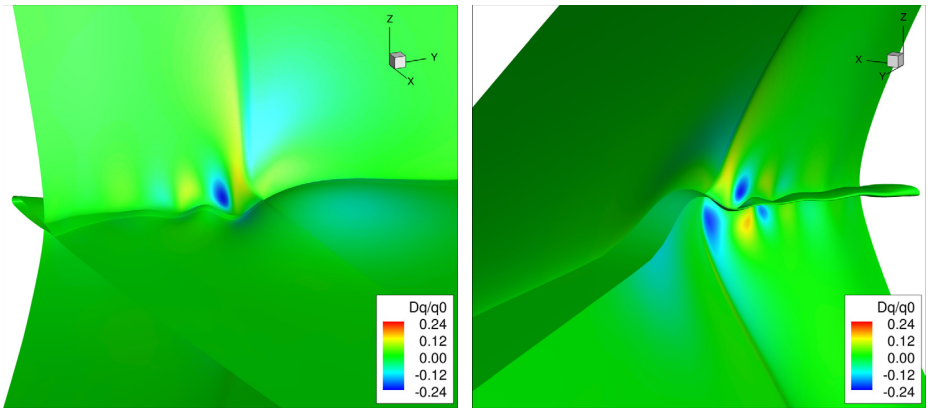


(a) Space filling (b) Clustered space filling

Fig. 11. Base sets for $N_b = 250$. (For interpretation of the colours in this figure, the reader is referred to the web version of this article.)

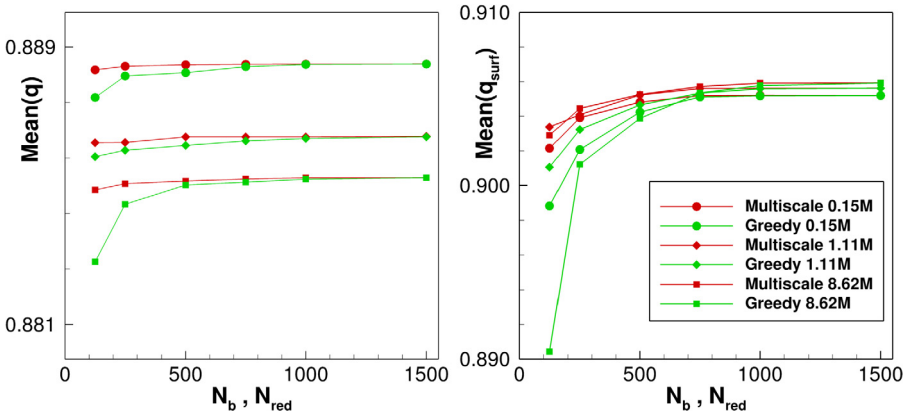
$$\hat{d} = \frac{(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{n}_i}{\|(\mathbf{x} - \mathbf{x}_i)\| \|\mathbf{n}_i\|} \tag{29}$$

Here $\mathbf{n}_i$ is the outward normal vector for surface control point $\mathbf{x}_i$ and a is the compression parameter. This is defined as $a = f(R, r_i)$, where R is the global support radius used for the base set, and r_i 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. This anisotropic norm function can be substituted when calculating separation distances (Algorithm 2: line 14) and is designed to restrict influences between separate surfaces that enclose a thin volume. For the base set, the anisotropic norm is only used for calculating the separation distance criterion; the standard Euclidian norm is used for RBF influence coefficients to maintain a positive definite influence matrix. However



(a) View from behind wing (b) View from ahead of wing

Fig. 12. Two views of mesh orthogonality change. $N_v = 1.11M$ points.



(a) Mean orthogonality (b) Mean orthogonality near surface

Fig. 13. Mesh quality variation with N_b, N_{red}. $N_v = 0.15M, 1.11M, 8.62M$ points.

for the refinement points, the anisotropic norm is used for both the separation distance criterion and the RBF influences since this is required to maintain the multiscale independence.

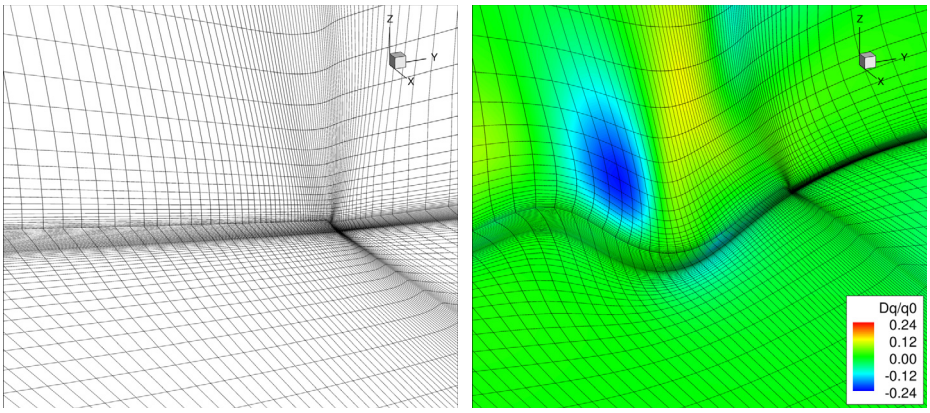
The improvements presented here could also be incorporated into conventional greedy-type methods, to improve performance of these. However, greedy-type methods are two-stage approaches: a reduced dataset of control points selected to give the most accurate interpolation of the moving surface points that are not included in the data set; a secondary stage to correct all the locations of the non-included points. Since these methods depend on the selection of a reduced dataset based on interpolation, the full interpolation system is solved regularly during the selection process. Hence, improving the point selection process for greedy using the methods above would not change the cost, and so greedy methods with these improvements have not been tested here.

7.2. Mesh quality

Mesh quality is now investigated in detail for the modern transport wing test case for the three mesh densities, with the orthogonality metric (equation (26)). The multiscale RBF method is applied to transfer the surface displacements into the volume mesh again using Wendland's C^2 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 method [7] (Algorithm 1) where $N_{red} = N_b$. The greedy point selection is performed here using unit vector displacement of the entire wing surface [7], and a single-point correction [3] is applied using a support radius equal to that of the maximum reduced surface set separation distance, as explained earlier.

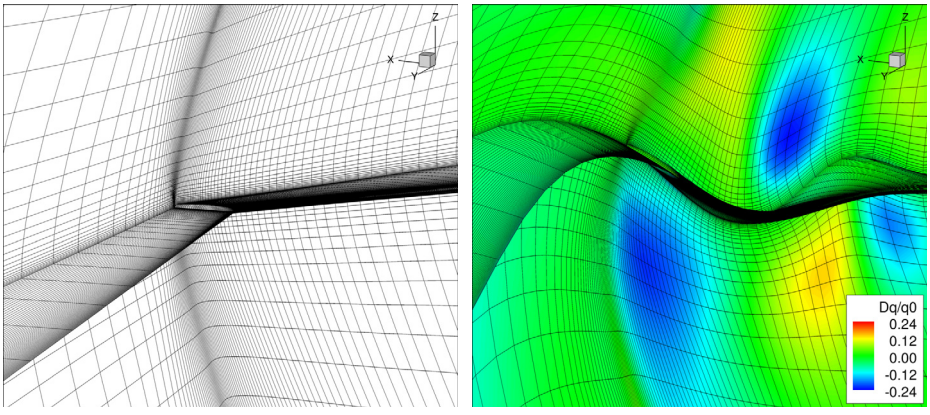
Fig. 12 shows two views of selected mesh planes, showing the extent of the deformation, and example orthogonality change; the wing surface is shown along with the wake-slit and tip-slit, and a spanwise grid plane. These are the same views as shown later in Figs. 14 and 15.

Fig. 13 shows the variation in average mesh orthogonality and mesh orthogonality in the near surface region, for varying surface set sizes, for the three mesh sizes. This clearly demonstrates that for both greedy and multiscale methods, only a



(a) Original mesh (b) Deformed mesh

Fig. 14. Mesh orthogonality change. 1.11M point mesh, view from ahead of wing.



(a) Original mesh (b) Deformed mesh

Fig. 15. Mesh orthogonality change. 1.11M point mesh, view from behind wing.

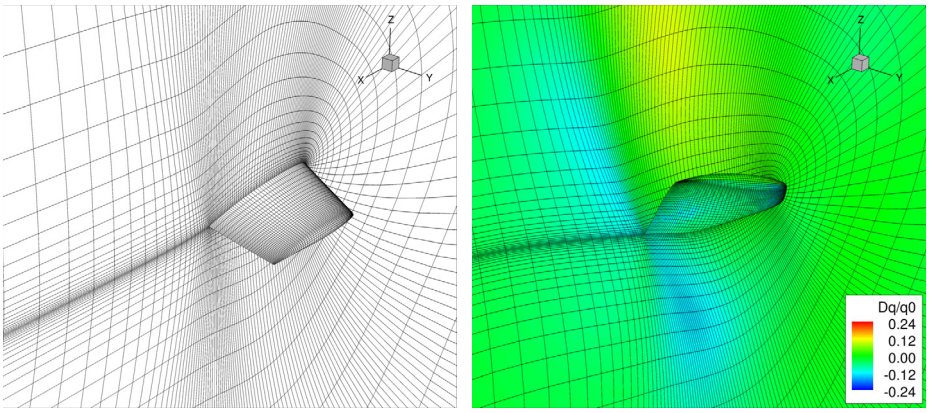
small number of surface points are required to deform the mesh effectively. The asymptotic trend is similar in all cases, with the multiscale method performing slightly better, i.e. reaching the asymptote slightly faster, in all cases. For all meshes, 500–750 points are sufficient, and this is around 1% of the finer mesh surface size. This confirms the findings in [7], where it was shown that the geometric interpolation error associated with the surface reconstructed from the reduced dataset is dependent on the geometry representation, not the surface mesh size.

Orthogonality change and mesh distributions are shown in Figs. 14, 15 and 16, which also show the same view of the initial mesh. Hence, even for a deformation of this extent, mesh orthogonality is preserved at the surface. The one million cell mesh is shown for clarity, and this deformation was computed with $N_b = 750$.

8. 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 than 1% of the full surface mesh 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



(a) Original mesh (b) Deformed mesh

Fig. 16. Mesh orthogonality change. 1.11M mesh, view from outboard of tip.

mesh preprocessing/point selection stage, and the cost is reduced significantly here from $O(N_{red}^4)$ to $O(N_s \log(N_s))$. However, unlike such reduction methods, the interpolation is exact, requiring no subsequent error correction. Furthermore, the multiscale point selection is based purely on geometry, so is decoupled from the deformation, and so the point selection only needs to be performed once, prior to any simulation.

Data Access Statement

The meshes used to produce the data here are available at the University of Bristol data repository, data.bris, at https://doi.org/10.5523/bris.2nwcp58q20uhq2u5znia4ovkdu, DOI: 10.5523/bris.

References

[1] T.C.S. Rendall, C.B. Allen, Unified fluid–structure interpolation and mesh motion using radial basis functions, Int. J. Numer. Methods Eng. 74 (10) (2008) 1519–1559.
[2] S. Jakobsson, O. Amoignon, Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization, Comput. Fluids 36 (2007) 1119–1136.
[3] T.C.S. Rendall, C.B. Allen, Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors, Int. J. Numer. Methods Eng. 81 (1) (2010) 89–105.
[4] T.C.S. Rendall, C.B. Allen, Efficient mesh motion using radial basis functions with data reduction algorithms, J. Comput. Phys. 228 (17) (2009) 6231–6249.
[5] C.B. Allen, Parallel universal approach to mesh motion and application to rotors in forward flight, Int. J. Numer. Methods Eng. 69 (10) (2007) 2126–2149.
[6] A. de Boer, M.S. van der Schoot, H. Bijl, Mesh deformation based on radial basis function interpolation, Comput. Struct. 85 (2007) 784–795.
[7] T.C.S. Rendall, C.B. Allen, Reduced surface point selection options for efficient mesh deformation using radial basis functions, J. Comput. Phys. 229 (8) (2010) 2810–2820.
[8] A. Michler, Aircraft control surface deflection using rbf-based mesh deformation, Int. J. Numer. Methods Eng. 88 (10) (2011) 986–1007.
[9] Y. Wang, N. Qin, N. Zhao, Delaunay graph and radial basis function for fast quality mesh deformation, J. Comput. Phys. 294 (2015) 149–172.
[10] G. Wang, H.H. Mian, Z.-Y. Ye, J.-D. Lee, Improved point selection method for hybrid-unstructured mesh deformation using radial basis functions, AIAA J. 53 (4) (2015) 1016–1025.
[11] B. Erzincanli, M. Sahin, An arbitrary Lagrangian–Eulerian formulation for solving moving boundary problems with large displacements and rotations, J. Comput. Phys. 255 (2013) 660–679.
[12] F. Bos, B. van Oudheudsen, H. Bijl, Radial basis function based mesh deformation applied to simulation of flow around flapping wings, Comput. Fluids 79 (2013) 167–177.
[13] L. Ding, L. Zhiliang, G. Tongqing, An efficient dynamic mesh generation method for complex multi-block structured grid, Adv. Appl. Math. Mech. 6 (1) (2014) 120–134.
[14] Y. Rozenberg, S. Aubert, G. Bénéfice, Fluid structure interaction problems in turbomachinery using rbf interpolation and greedy algorithm, in: ASME Turbo Expo 2014: Turbine Technical Conference and Exposition, Vol. 2B: Turbomachinery, ASME, 2014.
[15] M. Cordero-Gracia, M. Gomez, J. Ponsin, E. Valero, An interpolation tool for aerodynamic mesh deformation problems based on octree decomposition, Aerosp. Sci. Technol. 23 (2012) 93–107.
[16] C. Sheng, C.B. Allen, Efficient mesh deformation using radial basis functions on unstructured meshes, AIAA J. 51 (3) (2013) 707–720.
[17] T. Gillebaart, D. Blom, A. van Zuijlen, H. Bijl, Adaptive radial basis function mesh deformation using data reduction, J. Comput. Phys. 321 (2016) 1–20.
[18] T. Gillebaart, A. van Zuijlen, H. Bijl, Radial Basis Function Mesh Deformation Including Surface Orthogonality, AIAA Paper no. AIAA-2016-1674, AIAA SciTech, AIAA, San Diego, CA, Jan 2016, 2016.
[19] V. Poirier, S. Nadarajah, Efficient reduced-radial basis function mesh deformation within an adjoint-based aerodynamic optimization framework, J. Aircr. 53 (6) (2016) 1905–1921.
[20] M. Selim, R. Koomullil, Mesh deformation approaches – a survey, J. Phys. Math. 7 (2) (2016) 1–9.
[21] P.M. Hartwich, S. Agrawal, Method for perturbing multiblock patched grids in aeroelastic and design optimisation applications, in: 13th Computational Fluid Dynamics Conference, Colorado, 1997, AIAA paper AIAA1997-2018.
[22] C.B. Allen, Aeroelastic computations using algebraic grid motion, Aeronaut. J. 106 (1064) (2002) 559–570.
[23] L.E. Eriksson, Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation, AIAA J. 20 (10) (1982) 1313–1320.

[24] M.A. Potsdam, G.P. Guruswamy, A parallel multiblock mesh movement scheme for complex aeroelastic applications, in: 39th Aerospace Sciences Meeting, Reno, NV, 2001, AIAA paper AIAA2001-0716.
[25] J.T. Batina, Unsteady Euler aerofoil solutions using unstructured dynamic meshes, in: AIAA 27th Aerospace Sciences Meeting and Exhibit, AIAA, 1989, AIAA2989-8999.
[26] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, Torsional springs for two-dimensional dynamic unstructured fluid meshes, Comput. Methods Appl. Mech. Eng. 163 (1998) 231–245.
[27] F. Blom, Considerations on the spring analogy, Int. J. Numer. Methods Fluids 32 (2000) 647–668.
[28] E. Sheta, H. Yang, S.D. Habchi, Solid Brick Analogy for Automatic Grid Deformation for Fluid–Structure Interaction, AIAA Paper no. AIAA-2006-3219, 2006.
[29] R. Kamakoti, W. Shyy, Fluid–structure interaction for aeroelastic applications, Prog. Aerosp. Sci. 40 (2004) 535–558.
[30] R. Loehner, C. Yang, Improved ALE mesh velocities for moving bodies, Commun. Numer. Methods Eng. 12 (1996) 599–608.
[31] J. Bau, H. Luo, R. Loehner, E. Goldberg, A. Feldhun, 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.
[32] L. Tysell, Grid deformation of 3D hybrid grids, in: Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations, 2002.
[33] B. Helenbrook, Mesh deformation using the biharmonic operator, Int. J. Numer. Methods Biomed. Eng. 56 (2003) 1007–1021.
[34] X. Liu, N. Qin, H. Xia, Fast dynamic grid deformation based on Delaunay graph mapping, J. Comput. Phys. 211 (2006) 405–423.
[35] E. Luke, E. Collins, E. Blades, A fast mesh deformation method using explicit interpolation, J. Comput. Phys. 231 (2012) 586–601.
[36] J.A.S. Witteveen, Explicit and robust inverse distance weighting mesh deformation for cfd, in: 48th Aerospace Sciences Meeting, Reno, NV, AIAA, 2010, AIAA paper AIAA2010-016.
[37] X. Zhou, S. Li, A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing, J. Comput. Phys. 235 (2013) 199–215.
[38] X. Zhou, S. Li, A novel three-dimensional mesh deformation method based on sphere relaxation, J. Comput. Phys. 298 (2015) 320–336.
[39] M.D. Buhmann, Radial basis functions, Acta Numer. 9 (2000) 1–38.
[40] A. Beckert, H. Wendland, Multivariate interpolation for fluid–structure-interaction problems using radial basis functions, Aerosp. Sci. Technol. 5 (2001) 125–134.
[41] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 (1995) 389–396.
[42] M.S. Floater, A. Iske, Multistep scattered data interpolation using compactly supported radial basis functions, J. Comput. Appl. Math. 73 (1996) 65–78.
[43] H. Wendland, Multiscale analysis in Sobolev spaces on bounded domains, Numer. Math. 116 (2010) 493–517.
[44] B. Siebert, G. Dulikravich, Grid Generation Using a Posteriori Optimization with Geometrically Normalised Functions, AIAA Paper No. AIAA-1990-3048, Applied Aerodynamics Conference, Portland, OR, 1990.
[45] S. Allwright, Multi-discipline optimisation in preliminary design of commercial transport aircraft, in: Computational Methods in Applied Sciences, ECCOMAS, 1996, pp. 523–526.
[46] D. Haase, V. Selmin, B. Winzell, Progress in Computational Flow-Structure Interaction, in: Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, 2002.
[47] C.B. Allen, Towards automatic structured multiblock mesh generation using improved transfinite interpolation, Int. J. Numer. Methods Eng. 74 (5) (2008) 697–733, http://dx.doi.org/10.1002/nme.2170.