Journal of Computational Physics 229 (2010) 2810–2820

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Reduced surface point selection options for efficient mesh deformation using radial basis functions

T.C.S. Rendall [*], C.B. Allen

Aerospace Engineering Department, University of Bristol, Bristol BS8 1TR, UK

ARTICLE INFO

Article history:
Received 4 September 2009
Accepted 7 December 2009
Available online 4 January 2010

Keywords:
CFD mesh deformation
Radial basis functions
Greedy algorithms

ABSTRACT

Previous work by the authors has developed an efficient method for using radial basis functions (RBFs) to achieve high quality mesh deformation for large meshes. For volume mesh deformation driven by surface motion, the RBF system can become impractical for large meshes due to the large number of surface (control) points, and so a particularly effective data reduction scheme has been developed to vastly reduce the number of surface points used. The method uses a chosen error function on the surface mesh to select a reduced subset of the surface points; this subset contains a sufficiently small number of points so as to make the volume deformation fast, and a correction function is used to correct those surface points not included. Hence, the scheme is split such that both parts are working on appropriate problems. RBFs are an excellent way of finding smooth orthogonality preserving global deformations, but are less suitable for enforcing an exact geometry for a large number of points, while a simpler approach is ideal for diffusing small changes evenly but has quality (and possibly expense) drawbacks if used for the entire volume. However, alternatives exist for the error function used to select the reduced data set, so here a comparison is made between three different options: the surface error function, the unit function and the power function. Tests run on structured and unstructured meshes show that the surface error function gives the lowest errors, but this also requires a deformed surface shape to be known in advance of the simulation. The unit and power functions both avoid the need for a deformed surface, and the unit function is shown to be superior.

1. Introduction

Current aerodynamic design procedures make extensive use of CFD codes. However, these methods are not routinely extended to include aeroelastic behaviour, which can significantly influence performance. One of the reasons for this is the increased cost of aeroelastic simulation, but the requirement for a mesh deformation method is also a restriction. To address this aspect of the problem, recent work by the authors has developed an efficient high quality mesh deformation scheme [1,2] using radial basis functions (RBFs) that is entirely independent of the mesh type, and easily parallelised. This has been validated previously with aeroelastic and optimisation test cases [3,4], as well as complex examples with multiple surfaces, such as high lift and rotor configurations [5]. Example mesh deformation cases are shown in Fig. 1 for deflections of the multi-disciplinary optimisation wing [6], on an 8×10^6 cell structured mesh. However, while the RBF-based mesh motion is mathematically elegant, robust, and produces high quality mesh motion, it is expensive in its pure form, since the resulting system is of size $N_{sp} \times N_{vp}$, where N_{sp} is the number of surface points defining the motion, and N_{vp} is the number of volume mesh points to be moved.

[*] Corresponding author. Tel.: +44 0117 331 7641; fax: +44 0117 927 2771.
E-mail addresses: thomas.rendall@bristol.ac.uk (T.C.S. Rendall), c.b.allen@bristol.ac.uk (C.B. Allen).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.12.006

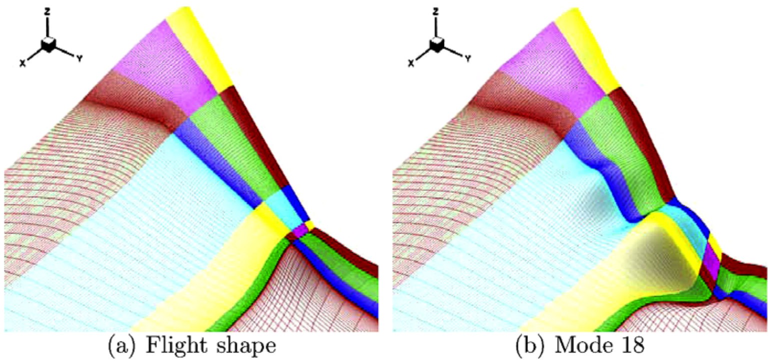


Fig. 1. MDO wing mesh motion on an eight million cell mesh.

To increase the efficiency of the RBF mesh motion method the authors have developed a method wherein the number of surface points used to define the surface motion is vastly reduced through a point selection process [2,5]. This minimises the position error across all surface points, whilst also minimising the number of surface points required. Hence, when the volume mesh is to be moved, it depends on a smaller number of surface points, and therefore the volume motion process is much faster. Of course, this means the positions of surface points not included in the reduced set will be interpolated and so will not be moved to exactly the correct position, requiring a correction function to adjust the positions of those points. Hence, the scheme is optimum, as both stages are working on problems appropriate given their computational strengths; RBFs are an excellent way of finding smooth orthogonality preserving global deformations, but are less suitable for enforcing an exact geometry for a large number of points, while a simpler approach is ideal for diffusing small changes evenly but has quality (and possibly expense) drawbacks if used for the entire volume. It was shown in [2] that the number of surface points required is independent of the mesh size, since the problem is one of geometry representation, and that an 11,593 point surface mesh could be represented by only 200 points. However, clearly the points selected are critical to the accuracy of the resulting interpolation, and alternatives exist for the error function used, so the objective of this paper is to assess and compare different error functions that may be used for selecting the reduced set of surface points.

2. Formulation

Mesh deformation using RBFs is achieved by interpolating displacements through the volume mesh away from the deflected surface. This interpolation takes the form [1]

$$I(\mathbf{x}) = \sum_{i=1}^{i=N} \alpha_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{1}$$

Here, $I(\mathbf{x})$ is the function to be evaluated at location $\mathbf{x}$ and defines the deformation of the volume points, ϕ is the basis function (in this work Wendland's C^2 is used [7]) and the index i identifies the centres for the RBFs, while $\mathbf{x}_i$ is the location of that centre. For mesh deformation the centres correspond to mesh points located on moving surfaces. The coefficients α_i are found by requiring exact recovery of the original function at these points $\mathbf{x}_i$.

Using subscript s to denote a surface mesh point, and using the change in position of a mesh point as the variable to be interpolated, the problem is written in the following fashion:

$$\Delta\mathbf{x}_s = \mathbf{M}\mathbf{a}_x \tag{2}$$
$$\Delta\mathbf{y}_s = \mathbf{M}\mathbf{a}_y \tag{3}$$
$$\Delta\mathbf{z}_s = \mathbf{M}\mathbf{a}_z \tag{4}$$

where

$$\Delta\mathbf{x}_s = \begin{pmatrix} \Delta x_{s_1} \\ \vdots \\ \Delta x_{s_N} \end{pmatrix} \quad \mathbf{a}_x = \begin{pmatrix} \alpha^x_{s_1} \\ \vdots \\ \alpha^x_{s_N} \end{pmatrix} \tag{5}$$

(analogous definitions hold for $\Delta\mathbf{y}_s$ and $\Delta\mathbf{z}_s$ and their $\mathbf{a}$ vectors)

$$\mathbf{M} = \begin{pmatrix} \phi_{s_1 s_1} & \phi_{s_1 s_2} & \cdots & \phi_{s_1 s_N} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{s_N s_1} & \phi_{s_N s_2} & \cdots & \phi_{s_N s_N} \end{pmatrix} \tag{6}$$

with

$$\phi_{s_1 s_2} = \phi(\xi_{s_1 s_2}) = \phi(\|\mathbf{x}_{s_1} - \mathbf{x}_{s_2}\|/R) \tag{7}$$

indicating the basis function evaluated on the distance between points s_1 and s_2 with a support radius of R, where $\|\cdot\|$ is the Euclidean norm. If a point lies outside the support radius then it will not be moved by the motion scheme as the basis function is zero beyond this point. The C^2 function of Wendland [7] was found to provide the best combination of deformation quality and matrix conditioning, and this is defined as:

$$\begin{aligned} \phi(\xi) &= (1-\xi)^4(4\xi+1) \quad \xi < 1.0 \\ \phi(\xi) &= 0 \quad \xi \geqslant 1.0 \end{aligned} \tag{8}$$

To determine the dependence of M volume points on the N surface points the following matrix must be formed, where v indicates a volume node:

$$\mathbf{A} = \begin{pmatrix} \phi_{v_1 s_1} & \phi_{v_1 s_2} & \cdots & \phi_{v_1 s_N} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{v_M s_1} & \phi_{v_M s_2} & \cdots & \phi_{v_M s_N} \end{pmatrix} \tag{9}$$

so that the positions of the volume points, given by the vectors $\Delta\mathbf{x}_v$, $\Delta\mathbf{y}_v$ and $\Delta\mathbf{z}_v$ are

$$\Delta\mathbf{x}_v = \mathbf{A}\mathbf{a}_x = \mathbf{A}\mathbf{M}^{-1}\Delta\mathbf{x}_s = \mathbf{H}\Delta\mathbf{x}_s \tag{10}$$
$$\Delta\mathbf{y}_v = \mathbf{A}\mathbf{a}_y = \mathbf{A}\mathbf{M}^{-1}\Delta\mathbf{y}_s = \mathbf{H}\Delta\mathbf{y}_s \tag{11}$$
$$\Delta\mathbf{z}_v = \mathbf{A}\mathbf{a}_z = \mathbf{A}\mathbf{M}^{-1}\Delta\mathbf{z}_s = \mathbf{H}\Delta\mathbf{z}_s \tag{12}$$

Or, more concisely

$$\begin{pmatrix} \Delta\mathbf{x}_v \\ \Delta\mathbf{y}_v \\ \Delta\mathbf{z}_v \end{pmatrix} = \begin{pmatrix} \mathbf{H} & 0 & 0 \\ 0 & \mathbf{H} & 0 \\ 0 & 0 & \mathbf{H} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}_s \\ \Delta\mathbf{y}_s \\ \Delta\mathbf{z}_s \end{pmatrix} \tag{13}$$

It is clear that there are two stages for accomplishing the mesh deformation. First comes the 'solve', which is defined as either finding $\mathbf{M}^{-1}$ or solving for $\mathbf{a}_{x/y/z}$. This is followed by the 'update', which is equivalent to either multiplying $\mathbf{a}_{x/y/z}$ by $\mathbf{A}$ or multiplying through by $\mathbf{H}$, depending on which option was used for the solve. Hence, reducing the number of surface points speeds up both the solve and update stages, since the number of control points is reduced. However, the update is the largest cost by some way, and this is reduced since the summation for each volume point deformation is calculated over fewer points.

3. Greedy point selection

It is not difficult to reduce the number of surface points used; for example, a simple structured coarsening or unstructured agglomeration could be used. However, it was decided to make the approach more intelligent, and to base the point selection on minimising the surface interpolation error. To select the surface points defining the motion, a greedy-type algorithm [8–11] was developed, and a hybrid variant developed by the authors [2] has been found to be particularly effective for minimising the interpolation error. The basic philosophy of the greedy method may be divided into four stages. Before starting it is necessary to populate the 'active list' (the list of surface points currently in use) by selecting an initial point, or set of points, for the first iteration to begin. Since the initial choice of point(s) can have only a minor influence on the later development of the point set, a simple answer that has been found to be adequate is to select the first point in the list of surface points. Once this step has been taken, the greedy loop begins.

In each cycle of the loop the active list is used to build a surface interpolation. After this, that interpolation is used to calculate the errors, which are the difference between the exact value of the function and the interpolated value of the function at all surface points. These errors are then checked, and the point with the largest error is marked. Finally, the marked point is added to the active list, which means it is included in the interpolation for the next iteration, thereby forcing the error at that point to be set to zero. In this fashion the method progressively 'irons out' errors in the interpolation. The process is repeated until the error is less than a certain limit at all of the unused centres. The strengths of this approach are that it is simple and potentially very effective, as any reduction in the number of surface points used gives a proportional reduction in the cost of the update (the major cost), which is a substantial saving.

The objective of this paper is to consider the different error signals that may be used for guiding the greedy method when selecting surface points.

4. Error signal descriptions

It is possible to use either geometrical arguments [10] or interpolation errors [12], but so far in this work interpolation errors are selected as they have the scope to produce more efficient point selections (i.e. showing a lower surface position error for a given number of points).

Suitable error candidates are the error function itself, including the actual interpolation errors, the power function [11] or any other prescribed function. The choice between these options is driven by considering how well the function reduces the error, versus data independence of that selection. It is clearly desirable to have a function that produces a low error using only a few points, but if the resulting point selection is very sensitive to the data itself then the result is of little practical use for mesh deformation. This is because mesh deformation involves repeated fitting of a time dependent function, so whatever point selection is made at the beginning of the process this must be suitable for a wide range of surface deformations.

According to de Marchi [10,11] in the limit the optimal distribution of centres for RBF interpolation is evenly spaced. This reasoning is based on a 'near' optimal data independent centre distribution found [11] through an analysis using Lagrange functions, and leads to a definition of the 'power function', which will be considered in Section 4.2. However, the best selection of surface points must be dependent on both the shape of the function to be interpolated and the set of points available from which to interpolate.

4.1. Surface error function

The errors $\mathbf{E}^x,\ \mathbf{E}^y,\ \mathbf{E}^z$ in either $x,\ y$ or z come from the actual surface deflection and are one good error signal. The error may be expressed at each point

$$\mathbf{E}^x = (\Delta\mathbf{x}_s - \mathbf{A}_{red}\mathbf{M}_{red}^{-1}\Delta\mathbf{x}_{red}) \tag{14}$$
$$\mathbf{E}^y = (\Delta\mathbf{y}_s - \mathbf{A}_{red}\mathbf{M}_{red}^{-1}\Delta\mathbf{y}_{red}) \tag{15}$$
$$\mathbf{E}^z = (\Delta\mathbf{z}_s - \mathbf{A}_{red}\mathbf{M}_{red}^{-1}\Delta\mathbf{z}_{red}) \tag{16}$$

These errors may be squared and added to give a single scalar error at point i

$$E_i = \sqrt{\left(\mathbf{E}_i^x\right)^2 + \left(\mathbf{E}_i^y\right)^2 + \left(\mathbf{E}_i^z\right)^2} \tag{17}$$

where $\mathbf{A}_{red}$ and $\mathbf{M}_{red}$ are the evaluation and interpolation matrices for the centres on the active list. $\Delta\mathbf{x}_{red},\ \Delta\mathbf{y}_{red},\ \Delta\mathbf{z}_{red}$ consist of the entries of $\Delta\mathbf{x}_s,\ \Delta\mathbf{y}_s,\ \Delta\mathbf{z}_s$ for those points on the active list (which is a reduced list of all points, denoted by the subscript *red*).

4.2. Power function

Lagrange functions arise commonly in polynomial interpolation and are functions that return 1 at the datum site that they are based around, but 0 at every other site. In this context these functions may be RBF interpolations themselves, with exactly the same property, and there are as many Lagrange functions as there are data points. To find each one it is only necessary to solve a single RBF interpolation problem, though it may be observed that the coefficients for any Lagrange function of this form are by definition a single column of the inverse of the interpolation matrix. Calculating the set of Lagrange functions is therefore synonymous with finding the inverse of the interpolation matrix. The neat property of these functions is that the final interpolation is just a sum of each function multiplied by its corresponding data value but, of course, this is a trivial point, amounting simply to multiplication of the data vector by the inverse interpolation matrix.

The important point about Lagrange functions is not that they offer quick interpolation (the cost of finding them is normally too high for the point set, the same as the inverse matrix cost) but that they help to separate the mathematics, which is constant, from the data, which are naturally not. This makes them a useful tool for error analysis. The interpolation, $I(\mathbf{x})$, expressed in the Lagrange function basis is [13,10,11,7,14]

$$I(\mathbf{x}) = \sum_{i=1}^{i=N} u_i(\mathbf{x})f_i \tag{18}$$

where u_i are the Lagrange functions and f_i their associated data. An error may be formulated purely in the Lagrange basis, normally referred to as the 'power function', P. The power function is useful because it gives a limit on the error in the interpolation as [15,10]

$$(f(\mathbf{x}) - I(\mathbf{x}))^2 \leqslant |f|^2 P(\mathbf{x})^2 \tag{19}$$
$$P(\mathbf{x})^2 = \left(\phi(0) - \mathbf{q}^T\mathbf{M}_{red}^{-1}\mathbf{q}\right)^2 \tag{20}$$

where

$$\mathbf{q}(\mathbf{x}) = \begin{pmatrix} \phi(\|\mathbf{x}-\mathbf{x}_1\|) \\ \phi(\|\mathbf{x}-\mathbf{x}_2\|) \\ \vdots \\ \phi(\|\mathbf{x}-\mathbf{x}_N\|) \end{pmatrix} \tag{21}$$

Usually P^2 is used to ensure the function is positive, and then the function is used in a similar way to the surface error function to decide which points to add to the active list. The subtraction means that the best value the function may take is 0, and anything larger implies a region of larger error. Thus, it is straightforward to add to the active list the point where P^2 is largest.

A conceptual explanation of the power function is as follows. Consider a single data site, and compute the basis function evaluated on the distances to all the other data sites. This vector is then the 'data' vector to be interpolated (the coefficients which are found with the stage $\mathbf{M}_{red}^{-1}\mathbf{q}$ above). Next, evaluate this interpolation by multiplying by $\mathbf{q}^T$, to give a number that is an approximate value for $\phi(0)$. Since the real value is of course known to be $\phi(0)$, the difference is the 'error' or power function. If the data site was included in the construction of the interpolation then collocation forces the power function there to be 0.

4.3. Unit function

A simple approach is to use a constant function, such as the unit function so that

$$\mathbf{E}^x = \mathbf{E}^y = \mathbf{E}^z = \left(\mathbf{1} - \mathbf{A}_{red}\mathbf{M}_{red}^{-1}\mathbf{1}_{red}\right) \tag{22}$$

This corresponds to prescribing a translation of the surface along a unit vector and then using the surface error function.

5. Results

5.1. MDO wing structured mesh

In order to test the suitability of the three suggested error functions test cases were considered using a flight shape, the 1st and the 18th mode shapes of the 35 m semispan MDO wing (see Fig. 2 for illustration of these shapes). Since the power

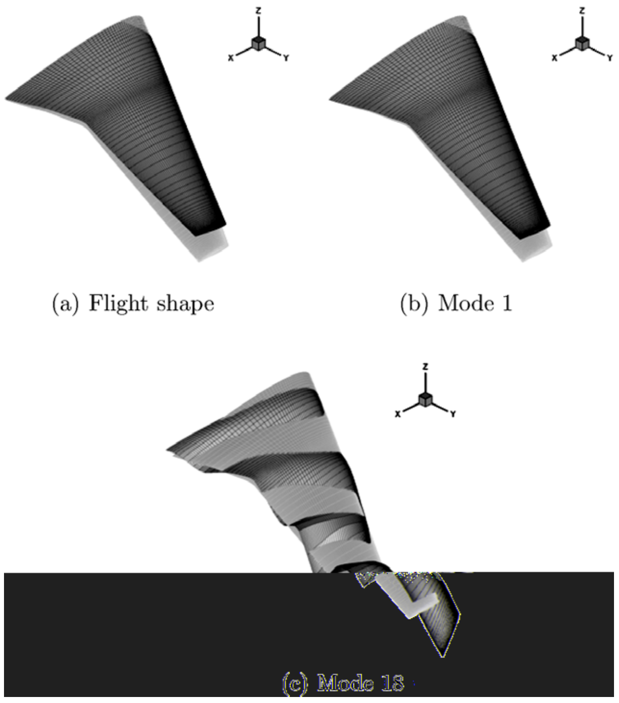


Fig. 2. MDO wing mesh deformation deflected shapes. Grey – original shape, black – deflected shape.

function may only easily be used when the inverse interpolation matrix is available (see Eq. (20)) and a comparison between all functions was required, a greedy method using the full matrix inversion at each step was chosen to perform the function tests. Four hundred points from the surface mesh (consisting of 11,593 points) of the 10^6 cell structured mesh were selected using Wendland's C^2 function, with a support radius of 10 m, using each of the error functions. Once these points had been selected, the interpolation was fitted and evaluated across the full set of surface points, allowing the error relative to the correct deformed surface to be calculated. This error was calculated both as an average error for all the points, and as the maximum error of any point. Maximum motions of any surface point were 2.5 m, 2.2 m and 4.8 m for the flight, 1st mode and 18th mode shapes, respectively.

Tables 1–3 present these results, and show an unexpected result. As might have been anticipated, using the actual surface errors leads to the lowest final error, but surprisingly the unit function performs better than the power function by a noticeable margin. Selection of a suitable error function is made quite simple by this result; if a suitable deformed shape exists then this should be used to select surface points, and if not, then the unit function should be applied. It does not appear that the power function offers any advantages, being less effective and more complicated to compute than the unit function.

One approach to explaining this result is to suggest that the unit function is not superior, rather, the power function is inferior. It has been observed [11] that the power function leads to an isotropic set of centres, as evidenced again here by Fig. 3(b). This is because the power function is the difference between real and interpolated values for $\phi(0)$, and ϕ is a radially

Table 1
Comparison of error functions using MDO wing flight shape (maximum point motion 2.5 m), for the surface error, unit or power function.

	Surface error	Unit	Power
Average error (cm)	0.045	0.115	0.461
Maximum error (cm)	0.136	0.672	4.327

Table 2
Comparison of error functions using MDO wing 1st mode shape (maximum point motion 2.2 m), for the surface error, unit or power function.

	Surface error	Unit	Power
Average error (cm)	0.033	0.091	0.397
Maximum error (cm)	0.102	0.527	3.843

Table 3
Comparison of error functions using MDO wing 18th mode shape (maximum point motion 4.8 m), for the surface error, unit or power function.

	Surface error	Unit	Power
Average error (cm)	0.713	1.169	2.049
Maximum error (cm)	2.256	9.249	11.394

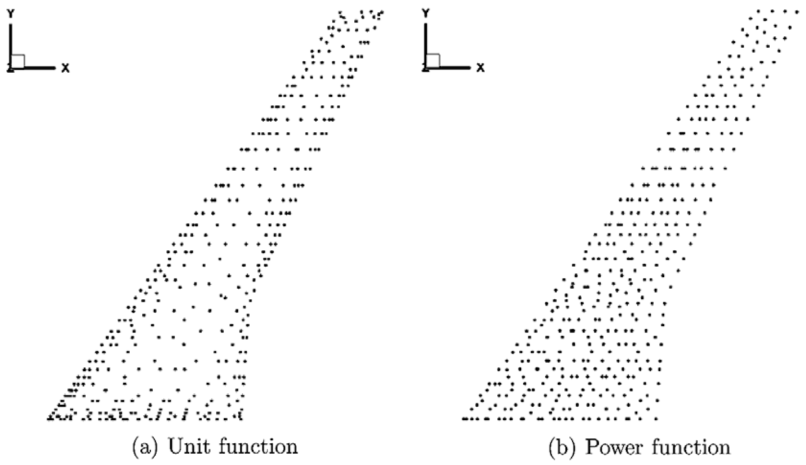
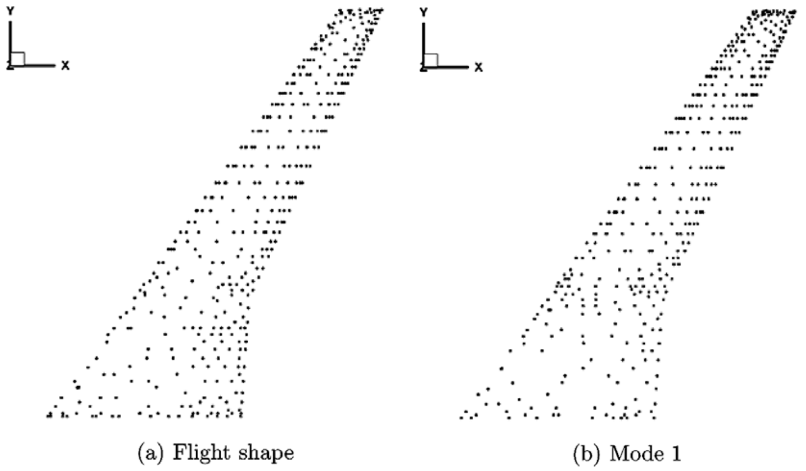


Fig. 3. Comparison of 400 surface points selected on the 10^6 cell mesh using the unit and power functions. These results are independent of the deformed shape, and are therefore only presented once.



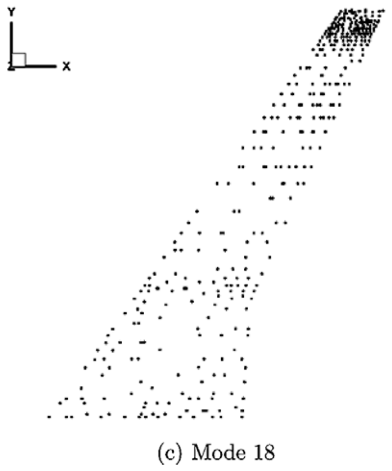


Fig. 4. Comparison of 400 selected surface points for three deformed shapes on the 10^6 cell mesh, found using the surface error function. These results are dependent on the deformed shape, and are therefore presented for different shapes.

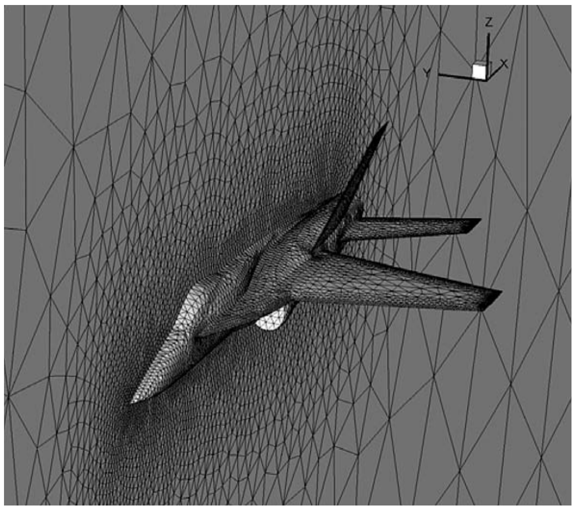


Fig. 5. YF-17 CGNS unstructured mesh.

symmetric function by definition. When interpolating a radially symmetric function across a set of points it might be suspected that a good point distribution shares a similar symmetry, with equal point spacing. Fundamentally this behaviour is observed because the influence of f, the actual function being interpolated, has been lost. Using the real position errors (thereby including the influence of f) leads to a highly anisotropic centre distribution as shown in Fig. 4, and one which is found to be the best of the three functions. Comparing the centre distributions from the unit and power functions in Fig. 3(a) and (b), respectively, shows that the unit function leads to a result somewhere between the surface error function and the power function. An interesting feature is that the unit function tends to add more points near the leading and trailing edges than the power function. It may therefore again be suspected that an isotropic centre distribution is a handicap rather than an advantage, and leads to the poor performance of the power function.

5.2. YF-17 unstructured mesh

A particularly harsh case was then considered. A 529,000 cell tetrahedral CGNS mesh for the YF-17, shown in Fig. 5, was used as the unstructured test case, and the deformed surface was defined by rotating the entire aircraft through 90° about the y-axis (giving a maximum change in surface position of 8.5 m in x and z). The surface mesh comprises 6394 points, and only 200 surface points, i.e. around 3.1%, were selected from this surface using the surface error, unit and power functions. These 200 points were then used to drive the volume mesh motion. A support radius of 50 was selected, which gave a deforming region that was large enough to reasonably accommodate the motion, as shown in Fig. 6.

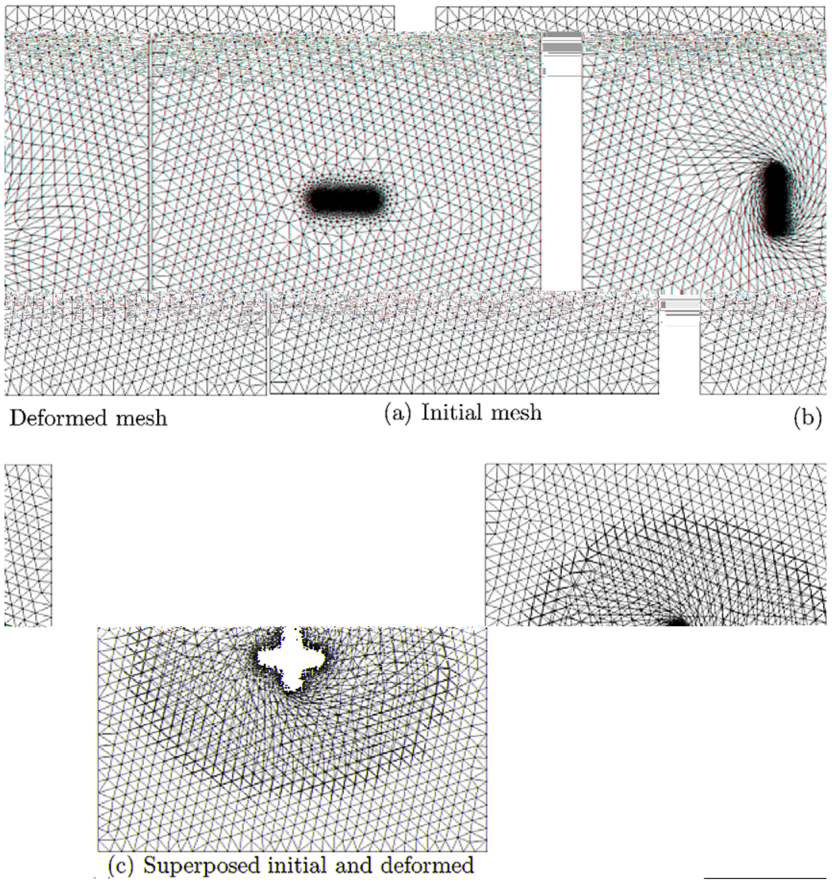


Fig. 6. YF-17 meshes.

Table 4
Comparison of error functions using rotated YF-17, for the surface error, unit or power function.

	Surface error	Unit	Power
Average error (cm)	0.009	0.019	0.050
Maximum error (cm)	0.037	0.191	0.517

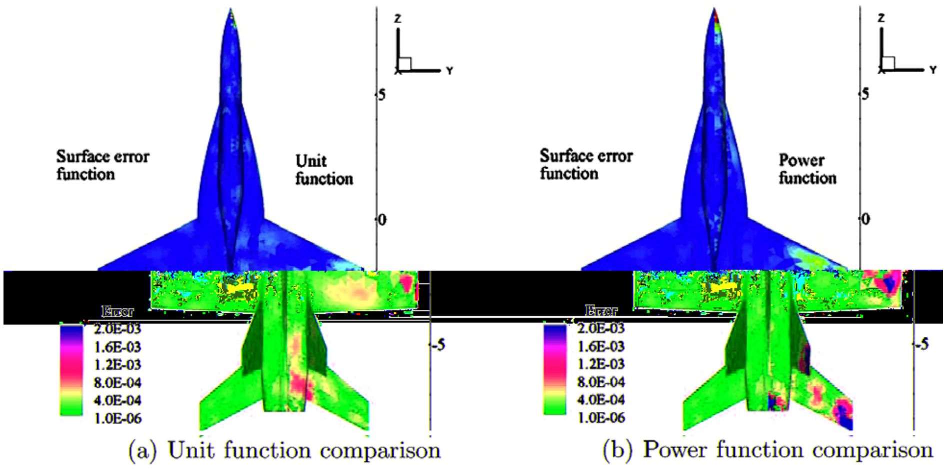


Fig. 7. YF-17 error comparison between surface error, unit and power functions.

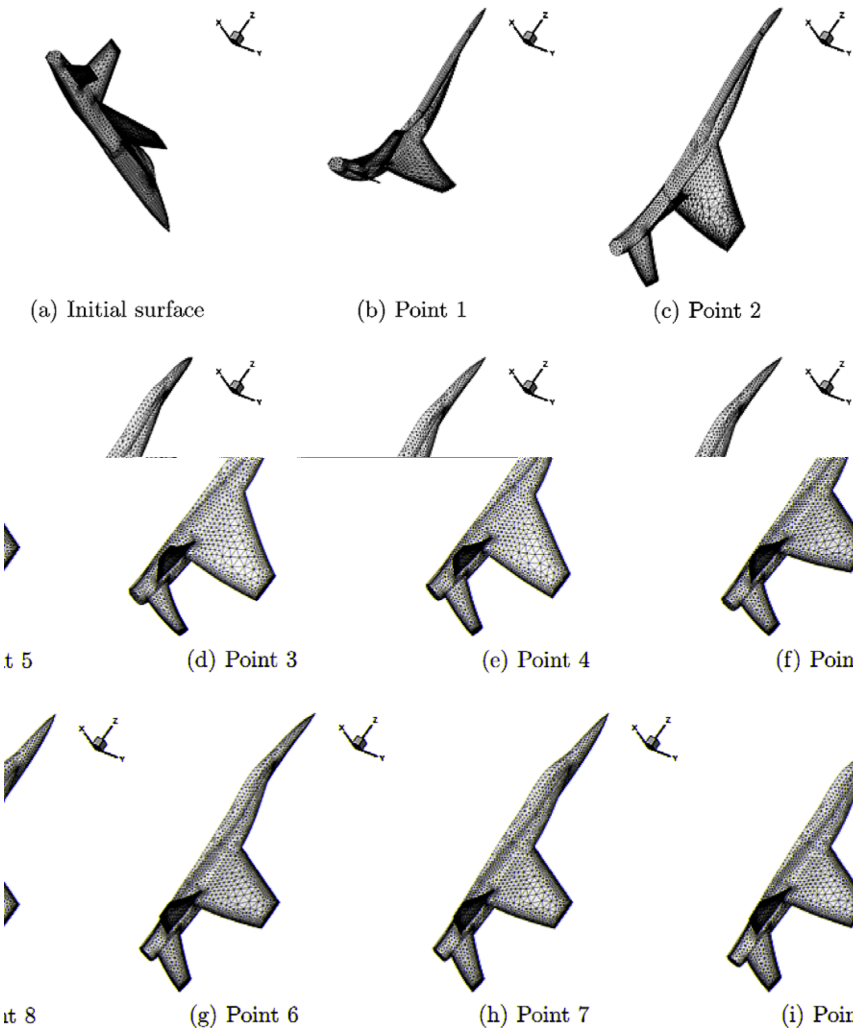


Fig. 8. Evolution of rotated YF-17 surface as first eight points on the surface are used.

Table 4 compares the maximum and average errors for the three different functions, and confirms that the surface error function is the most accurate, followed by the unit function. The power function again displays the largest errors. Fig. 7 shows the distribution of the errors across the surface of the aircraft, revealing that the extremities (nose, tail, wingtip and tailfin) show the largest, while Fig. 8 shows how the rotated surface evolves as more points are added to the interpolation. After only eight are used the surface already visually resembles the rotated surface very well, although more points would be required to achieve a surface accurate enough for a CFD analysis. Each function chooses a different set of points and these are illustrated in Fig. 9. Once more the surface error function picks points where the displacements are greatest, with the unit function showing a tendency to pick points close to edges. The power function chooses a completely even distribution of points and it is this even selection tendency that is posited to lead to its larger errors; comparison to the other two functions suggests an isotropic distribution of points is not optimal.

The undeformed and deformed meshes (surface and symmetry plane) for the YF-17 are shown in Fig. 10. Despite the large amplitude of the motion the quality of the cells in the near-field is preserved well, while the motion is accommodated in the mid-field before smoothly blending back to the far-field.

It should be stressed that, even with such an excessive deformation, the deformed mesh is obtained in a single stage, i.e. a single deformation of the initial mesh, and not computed over several steps. This is significant since, during an unsteady calculation, the initial mesh will always be recovered for a zero deformation. Furthermore, the maximum error with the unit function is still only 0.022% of the maximum deformation, and so a very simple correction scheme can be used to recover the exact surface position.

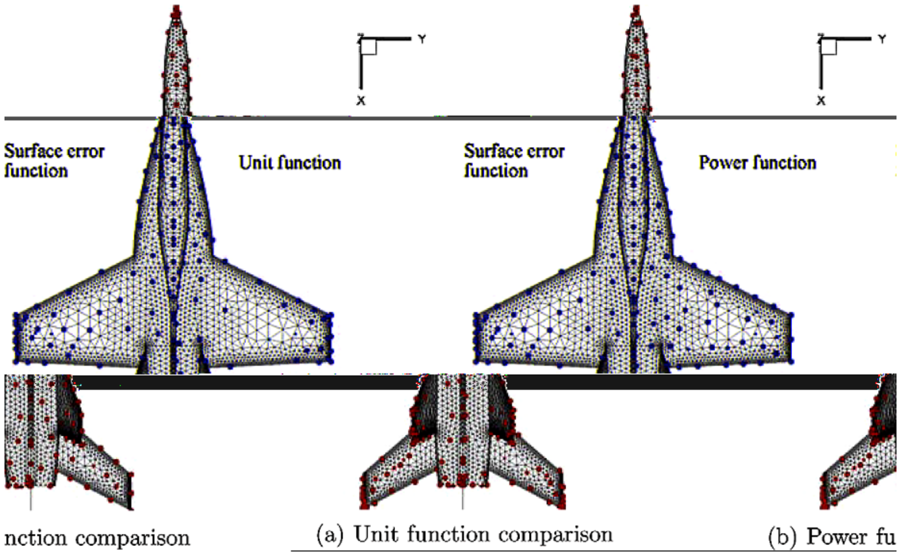


Fig. 9. YF-17 surface point selection comparison between surface error, unit and power functions.

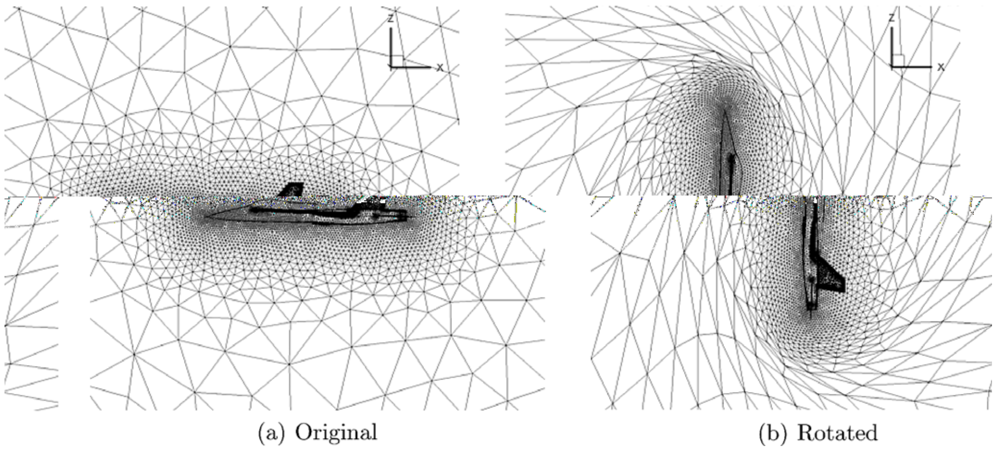


Fig. 10. YF-17 meshes.

6. Conclusion

Previous work by the authors has shown RBF-based mesh deformation to be extremely effective, preserving orthogonality and producing extremely high quality meshes, but potentially expensive owing to the dependence of the entire volume mesh on a large number of surface points. Hence, the scheme has been significantly improved to allow a vastly reduced set of surface points to define the deformation. This reduces both the 'solve' and 'update' stage costs, and splits the update stage into two parts, wherein both parts are working on problems appropriate given their computational strengths; RBFs are an excellent way of finding smooth orthogonality preserving global deformations, but are less suitable for enforcing an exact geometry for a large number of points, while a simpler approach is ideal for diffusing small changes evenly but has quality (and possibly expense) drawbacks if used for the entire volume. A greedy-type method has been used to select only the points that are strictly necessary, but this process demands a suitably defined error function to assess the points against each other.

The work presented here has compared three different functions. The first and most logical choice is the actual surface error, but this requires the deformed surface to be known in advance. Results have also been presented for the unit and power functions, both of which are independent of the exact deformation. Structured and unstructured meshes have been deformed in a high quality fashion showing both the flexibility and quality of the approach, and the unstructured mesh, which contains 6394 surface points, is moved using only 200 of them, showing the efficiency.

The smallest final errors are achieved when the actual surface position errors are used to drive the surface point selection. However, the deformed surface may not be known in advance, and it is desirable to keep the surface points used constant for a range of deflections (as might be the case for an unsteady simulation) anyway, so the unit function provides a simple and effective substitute. For the unstructured case, using only 3.1% of the surface points, the maximum error with the unit function has been shown as only 0.022% of the maximum deformation, demonstrating this is an extremely effective approach.

References

[1] T.C.S. Rendall, C.B. Allen, Fluid-structure interpolation and mesh motion using radial basis functions, International Journal for Numerical Methods in Engineering 75 (10) (2008) 1519–1559, doi:10.1002/nme.2219.

[2] T.C.S. Rendall, C.B. Allen, Efficient mesh motion using radial basis functions with data reduction algorithms, Journal of Computational Physics 228 (17) (2009) 6231–6249.

[3] A.M. Morris, C.B. Allen, T.C.S. Rendall, CFD-based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation 58 (8) (2008) 827–860, doi:10.1002/fld.1769.

[4] A.M. Morris, C.B. Allen, T.C.S. Rendall, Domain element method for aerodynamic shape optimization applied to modern transport wing, AIAA Journal 47 (7) (2009) 1647–1659.

[5] T.C.S. Rendall, C.B. Allen, Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors, International Journal for Numerical Methods in Engineering 81 (2010) 89–105.

[6] S. Allwright, Multi-discipline optimisation in preliminary design of commercial transport aircraft, in: J.-A. Desideri, C. Hirsch, P. Le Tallec, E. Onate, M. Pandolfi, J. Periaux, E. Stein (Eds.), Computational Methods in Applied Sciences, ECCOMAS, Wiley, 1996, pp. 523–526.

[7] H. Wendland, Scattered Data Approximation, first ed., Cambridge University Press, 2005.

[8] R. Schaback, H. Wendland, Adaptive greedy techniques for approximate solution of large RBF systems, Numerical Algorithms 24 (3) (2000) 239–254.

[9] Y.C. Hon, R. Schaback, X. Zhou, Adaptive greedy algorithm for solving large RBF collocation problems, Numerical Algorithms 32 (2003) 13–25.

[10] S. De Marchi, On optimal locations for radial basis function interpolation: computational aspects, Rendiconti Del Seminario Matematico 63 (3) (2003) 343–357.

[11] S. De Marchi, R. Schaback, H. Wendland, Near-optimal data-independent point locations for radial basis function interpolation, Advances in Computational Mathematics 23 (2005) 317–330.

[12] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, T.R. Evans, Reconstruction and representation of 3D objects with radial basis functions, in: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics, 2001, pp. 67–76.

[13] R. Schaback, Comparison of radial basis function interpolants, in: In Multivariate Approximation. From CAGD to Wavelets, World Scientific, 1995, pp. 293–305.

[14] M. Buhmann, Radial Basis Functions, first ed., Cambridge University Press, 2005.

[15] R. Schaback, Reconstruction of multivariate functions from scattered data, Technical report, Georg-August-UniversitSt Goettingen, 1997.