Journal of Computational Design and Engineering 5 (2018) 305–318

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

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

(Re-) Meshing using interpolative mapping and control point optimization

Ivan Voutchkov [a,*], Andy Keane [a], Shahrokh Shahpar [b], Ron Bates [b]

[a] Computational Engineering & Design Group, Southampton Boldrewood Innovation Campus, Building 176, University of Southampton, SO16 7QF, UK
[b] Aerothermal Design System, Rolls-Royce plc., ML-83, P.O. Box 31, Derby DE24 8BJ, UK

ARTICLE INFO

Article history:
Received 21 September 2017
Received in revised form 13 December 2017
Accepted 14 December 2017
Available online 15 December 2017

Keywords:
Mesh
Mesh smoothing
Mesh re-interpolation
Mesh mapping
Mesh optimization
Radial basis functions

ABSTRACT

This work proposes a simple and fast approach for re-meshing the surfaces of smooth-featured geometries prior to CFD analysis. The aim is to improve mesh quality and thus the convergence and accuracy of the CFD analysis. The method is based on constructing an interpolant based on the geometry shape and then mapping a regular rectangular grid to the shape of the original geometry using that interpolant. Depending on the selected interpolation algorithm the process takes from less than a second to several minutes. The main interpolant discussed in this article is a Radial Basis Function with cubic spline basis, however other algorithms are also compared. The mesh can be optimized further using active (flexible) control points and optimization algorithms. A range of objective functions are discussed and demonstrated. The difference between re-interpolated and original meshes produces a metric function which is indicative of the mesh quality. It is shown that the method works for flat 2D surfaces, 3D surfaces and volumes.

1. Introduction

The research discussed in this paper originates from a project aiming at mesh improvement of structured airfoil Computational Fluid Dynamic (CFD) meshes. The desired orientation of such meshes is often designed to follow the flow of the fluid around the airfoil. One of the most important areas is that adjacent to the airfoil which is used to predict boundary effects. Extra measures are usually taken in this area to ensure high quality meshing. PADRAM (Milli & Shahpar, 2012) – a tool developed by and used in Rolls Royce plc, does this by constructing an O-mesh, which grows around the airfoil to a certain thickness with increasing cell height. Subsequently four additional blocks are added to represent leading and trailing edges, and the pressure and suction sides.

A number of papers can be found in the public domain which describe the use of RBF for geometric smoothing, morphing and reconstruction (Car et al., 2001; Jakobsson & Amoignon, 2007; Savchenko, Savchenko, Egorova, & Hagiwara, 2008; Sieger, Menzel, & Botsch, 2013; Staten, Owen, Shontz, Salinger, & Coffey, 2011). Although the development described here has not been directly influenced by any of the quoted publications, this method carries a number of similarities. The main difference is that our approach starts with a minimum number of nodes used to construct the interpolant, only adding more if necessary (as will be discussed later), while existing published work starts with a large number of points. Some researchers have tried to use various algorithms to later reduce the number of these points. The large number of points is mainly caused by the intended application of the published methods – which is the recreation of a complex 3D objects with a high level of detail. The CFD meshes considered in this paper exhibit smoother features, hence the use of large number of points is not justified. ANSYS users may find resemblance to the "Mapped face meshing" capability of that product (ANSYS Meshing Controls: Mapped Face Meshing; Mapped Face Meshing in ANSYS Workbench). A mesh using this method is shown in Fig. 6 and will be discussed further on. In its basic implementation the idea is indeed very similar. However, readers will find significant further developments towards the middle and the end of this paper. Most notably the usage of additional fixed points (whilst ANSYS uses only four-corners) and the ability to move chosen control points to obtain an optimized mesh. The expansion into the 3D domain introduces further novelty. The technique also bears resemblance to free-form deformation methods described widely in papers such as (Keane, 2004).

Abbreviations: CFD, computational fluid dynamics; RBF, radial basis functions.

Peer review under responsibility of Society for Computational Design and Engineering.

* Corresponding author.
E-mail addresses: I.I.Voutchkov@soton.ac.uk (I. Voutchkov), Andy.Keane@soton.ac.uk (A. Keane), Shahrokh.Shahpar@Rolls-Royce.com (S. Shahpar), Ron.Bates@Rolls-Royce.com (R. Bates).

https://doi.org/10.1016/j.jcde.2017.12.003
2288-4300/

2. Materials and methods

Unlike a more traditional structured mesh, which is a continuation of the O-mesh into the periodic boundary, blocking ensures better fluid path tracking close to and away from the airfoil with a reduced number of elements. Depending on the geometrical complexity, e.g., during an automatic optimization workflow, it is sometimes possible for the resultant mesh to be less than perfect. One such mesh is shown in Fig. 2, where block 4 lacks smoothness due to a higher concentration of nodes in one particular area (Fig. 3). Blocks 1 and 3 also have minor defects around the middle of the airfoil length.

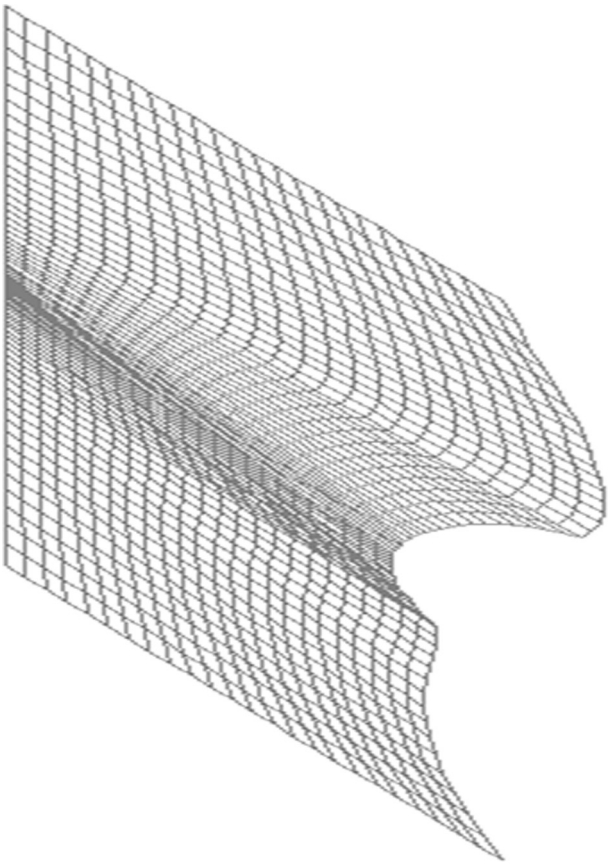

Fig. 1. Mesh produced by ANSYS 14.0, using Mapped Face Meshing feature.

Attempts have been made to improve such meshes by running a derivative aided, gradient based optimizer which moves mesh nodes freely within blocks. Objective functions tested include the difference of element angles to 90°, crowding distance (defined as the reciprocal value of the Euclidean distance) and difference of connection lengths within an element and its neighbours. Minimizing combinations of these goals proved successful for small meshes (less than 100 nodes). However when applied to a larger number of nodes the objective functions became extremely multimodal and the quality of the resulting mesh became highly sensitive to which local minima was chosen. The search for global minima was not feasible due to the significant computational cost.

Another approach using direct node manipulation is associated with using elliptical equations (Winslow, 1967). However, often there are regions in the mesh (especially for CFD simulation) where a uniform mesh is not desired, e.g., the stretched elements needed in the boundary layer next to viscous surfaces, as well as orthogonality to the surfaces for the boundary condition enforcements. The success of the method also depends highly on the iterative scheme chosen to converge the elliptical equations, the definition of which for large and complex geometries could be non-trivial.

Yet another technique, is a derivative of a weighted Laplacian smoothing (Shontz & Vavasis, 2004), where a vertex is relocated to the centroid of the vertices connected to that vertex – as shown in Fig. 4. The movement is then computed as follows:

$$P_1 = P + W \times \left(\frac{1}{n} \sum_{i=1,n}^{n} C_i - P \right),$$

where P is the original position of the node, P_1 is the new position of the node, C_i are the cell centroids of the n adjacent cells. Adding a weighting coefficient (W) and adding constraints can help avoiding inverted (negative volume) elements. This is relatively fast technique, but still requires several iterations. This method of smoothing attempts to smooth the mesh based on cell size, i.e., nodes of smaller cells tend to move towards larger cells. Higher number of

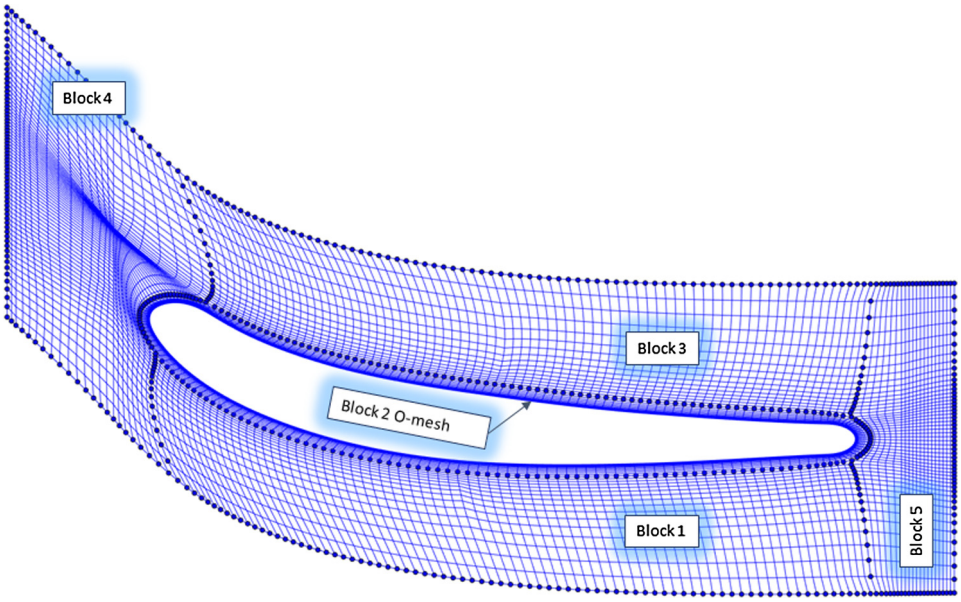

Fig. 2. A multi-block structured CFD mesh generated by PADRAM (Milli & Shahpar, 2012) around a turbomachinery blade.

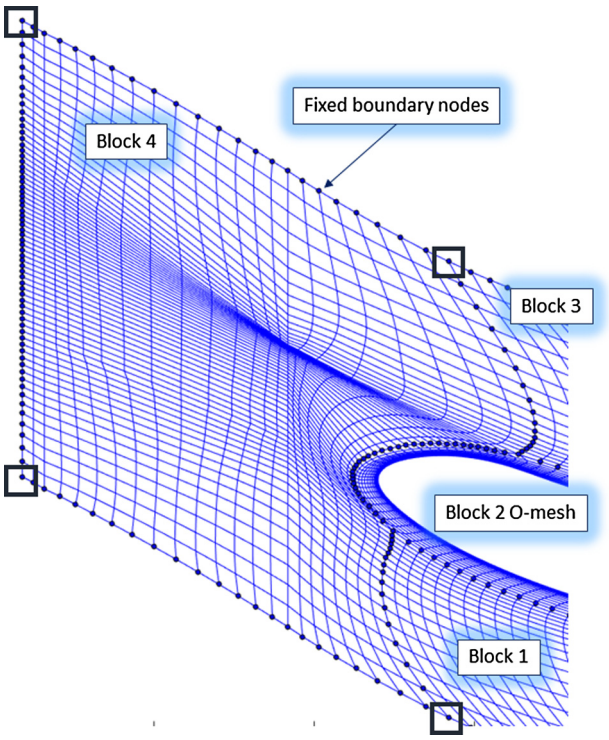

Fig. 3. Zoom-in region of Block 4 (of Fig. 2).

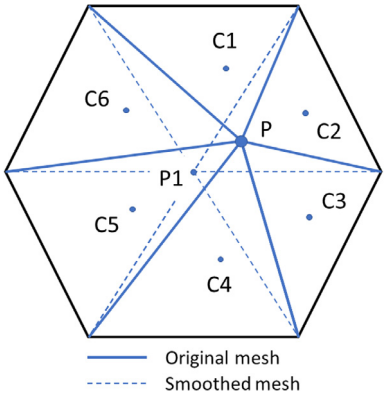

Fig. 4. Smoothing based on centroid-averaged movement of cell vertices.

iterations will make the mesh more uniform. For larger and complex meshes shapes it may stop prematurely due to its averaging nature.

Further reading and comparison of direct nodal movement methods can be found in Erten, Ungor, and Zhao (2009).

Rather than manipulate node positions directly this paper considers an alternative transformation approach.

3. Theory and calculations

3.1. Interpolative mapping

Let us first focus on Block 4, which requires most attention. We will use it to illustrate the idea of interpolative mapping. Being a structured grid, Block 4 has four distinct corners marked with squares in Fig. 3. We also have a desired position of nodes around the perimeter, which accurately describes the geometric shape of this domain.

These boundary nodes are also used to attach block 4 to blocks 1, 3 and the O-mesh (block 2). Their distribution along the length is also deliberate and serves to add more nodes around the leading edge flow. Hence it is important for them to stay fixed. Another effect, based on the fact that this is a structured mesh, is that the number of nodes on the opposite sides of this four-cornered shape need to be the same. The goal is then to have a "smooth" mesh within these bounds.

Given the four corners and the number of nodes around the edges, but ignoring the geometric shape, an ideal mesh exists in the form of a uniform rectangular grid (Fig. 5). The number of elements along X and Y correspond to block 4.

Therefore, a one-to-one function exists which describes the relationship between the node coordinates of the desired shape (block 4) and the template coordinates of Fig. 6. Note that only the fixed edge nodes are used to construct this function. Once this relationship is established we can use it to derive the coordinates of the internal nodes from the evenly spaced and right-angled elements of the template.

For two-dimensional meshes, a pair of mapping functions are required – one for each dimension (X and Y are on the arbitrary shape domain and $\underline{Xsq}$, Ysq in the Square domain):

$$X = F_x(X_{sq}(fixed), Y_{sq}(fixed)), \tag{1}$$

$$Y = F_y(X_{sq}(fixed), Y_{sq}(fixed)) \tag{2}$$

We have experimented with linear and non-linear polynomials up to 6th order, Radial Basis Functions (RBF) with linear, thin plate, cubic, Gaussian, multi-quadratic and inverse multi-quadratic basis and Kriging (Jones, Schonlau, & Welch, 1998; Nurdin, Bressloff, & Keane, 2012). It was found that RBF with cubic spline basis produces a good mapping with the least amount of time (computational expense). We will use this interpolant to explain the mechanics of the method, however we also discuss the use of other interpolants later in this article. The theory behind RBF is described elsewhere (Car et al., 2001; Jakobsson & Amoignon, 2007; Keane, 2004; Savchenko et al., 2008; Sieger et al., 2013; Staten et al., 2011) therefore we will not go into great detail. Readers who are not familiar with RBF mathematics, should still be able to reproduce the results following the equations below.

Following the above notation, we can express the coordinates of the edge nodes as follows:

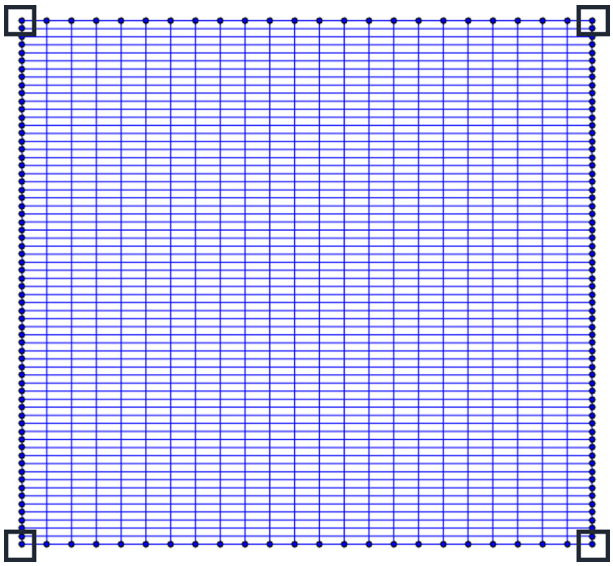

Fig. 5. Ideal four corner grid representing Block 4 with corresponding number of nodes along X and Y.

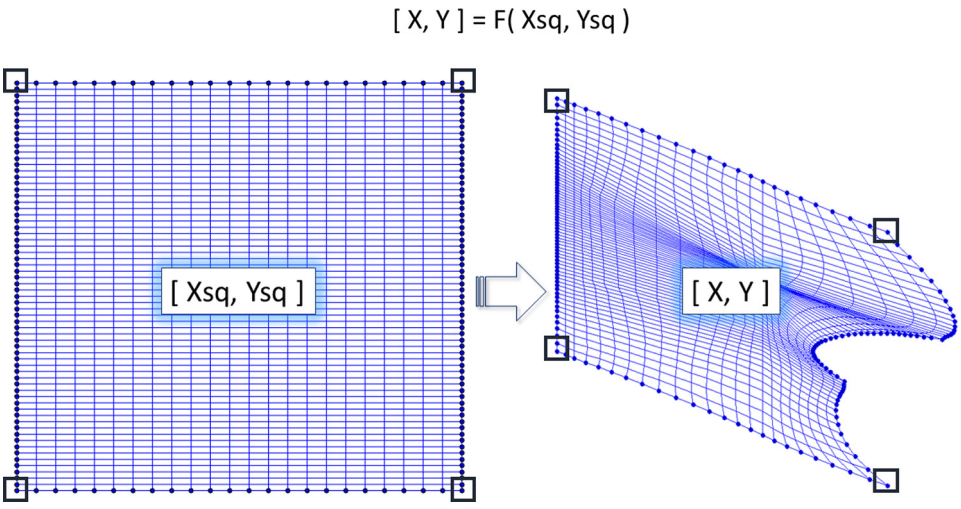

Fig. 6. Functional relationship between the ideal template (left) and the desired shape (right).

$$X_j(fixed) = \sum_{i=1}^{N} w_i^{(x)} R_{ij},\ i,j = 1 \ldots N \text{ (number of nodes)} \tag{3}$$

where *fixed* is the set of fixed control points and the distance matrix with cubic spline base, representing the distance between each point in the [*X*, *Y*] domain and every point on the [X_{sq}, Y_{sq}] domain is

$$R_{ij} = \left(\sqrt{(X_j(fixed) - X_{sq,i}(fixed))^2 + (Y_j(fixed) - Y_{sq,i}(fixed))^2} \right)^3 \tag{4}$$

The only unknowns in the above system of *j* equations are the weights $w_i^{(x)}$ and since *i* = *j* it is fully defined set. Hence, written in matrix form, the solution for the weights is

$$\mathbf{w}^{(x)} = R^{-1}\mathbf{X}(fixed). \tag{5}$$

We use the Choleski factorisation instead of direct matrix invert for better numerical stability in solving this equation. It is then possible to express the internal coordinates as follows:

$$X_j(int) = \sum_{i=1}^{N} w_i^{(x)} R_{ij} \tag{6}$$

where *internal* is the set of internal nodes and

$$R_{ij} = \left(\sqrt{(X_i(fixed) - X_{sq,j}(int))^2 + (Y_i(fixed) - Y_{sq,j}(int))^2} \right)^3. \tag{7}$$

Note the swap of *i* and *j* indices. This allows us to convert the internal template nodes [*Xsq*, *Ysq*] into internal nodes of the desired shape, relative to its boundary of *fixed* nodes. In a similar manner, the mapping function *F* for the second dimension can be derived:

$$Y_j(int) = \sum_{i=1}^{N} w_i^{(y)} R_{ij} \tag{8}$$

Fig. 7 illustrates the re-interpolated mesh using the above RBF mapping. The resultant mesh is now significantly smoother than the original mesh. It is also more uniform than the equivalent mesh produced by ANSYS 14.0, shown in Fig. 1, using the Mapped Face Meshing technique, which is the closest we have found in the existing state of the art to the method in this paper. Improving this mesh further will be discussed later in the paper. By replacing *X* with X_{sq} in (3) and (6) and swapping places of *X* and X_{sq}, as well as *Y* and Y_{sq} in (7), it is possible to compute the reverse mapping, showing how the current mesh would map into a square shape, as shown in Fig. 8. By computing the Euclidian distance between corresponding nodes in Figs. 8 and 5 it is possible to derive a mesh quality metric function as illustrated in Fig. 9. Similarly, the Euclidian distance between the original block 4 and transformed block 4 can be mapped onto the block 4 mesh to display how much each node moved during the transformation, which can also be indicative of the initial mesh smoothness as shown in Fig. 10.

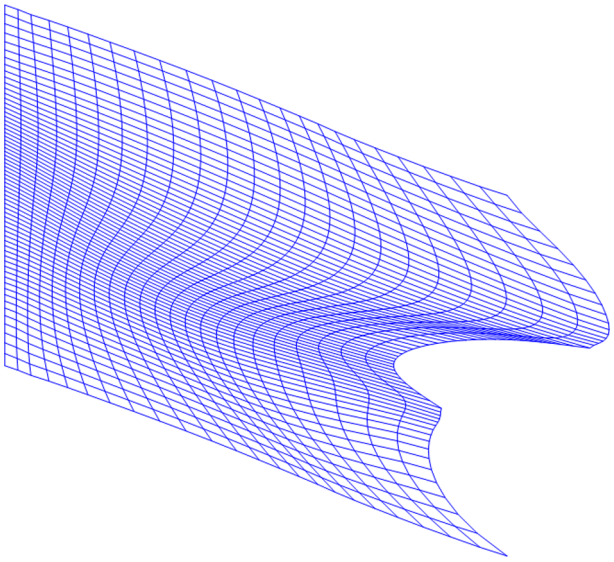

Fig. 7. Block 4 – transformed using interpolative mapping.

3.2. Interpolative mapping using active control points

The mesh shown in Fig. 7 is an improved version of block 4, however one can observe that some nodes are still too close to each other. This can be rectified by adding *Active* control points which can change their coordinates as opposed to the *Fixed* control points used for the edges. Two are shown in Fig. 11 (left). Eqs. (1)–(8) remain valid if we replace all occurrences of $X_i(fixed)$ with $X_i(cp) + \Delta X_i(cp)$ and $Y_i(fixed)$ with $Y_i(cp) + \Delta Y_i(cp)$, where the set *cp* is a union of *Fixed* and *Active* control points:

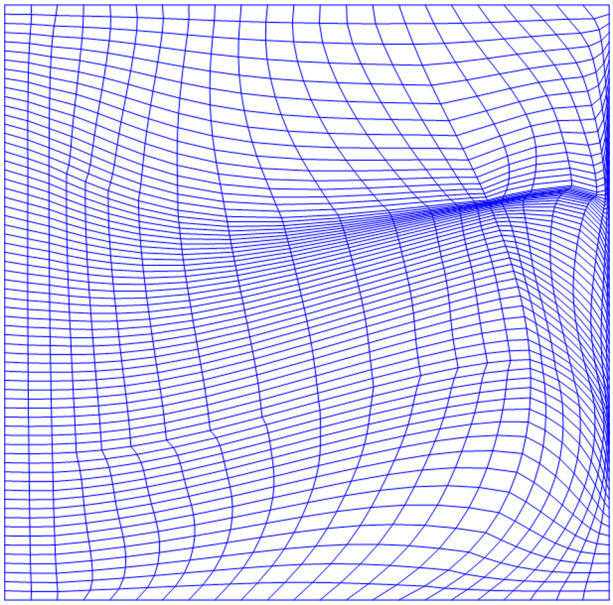

Fig. 8. Original Block 4 – mapped into a square shape.

$$cp = Fixed \cup Active$$

$\Delta X_i(cp)$ and $\Delta Y_i(cp)$ are zero for the *Fixed* and non-zero for the *Active* elements.

To demonstrate this idea two control points with grid positions [2117] and [2142] were picked manually by trial and error for illustration purposes. They were chosen in a region close to a highly curved boundary line. These are the ones shown in Fig. 11 (left). The corresponding increments are selected as follows:

$$\begin{pmatrix} \Delta X_1 & \Delta Y_1 \\ \Delta X_2 & \Delta Y_2 \end{pmatrix} = \begin{pmatrix} 3\% & -15\% \\ 10\% & 4\% \end{pmatrix}.$$

The resultant mesh is shown in Fig. 11 (right). Empty circles represent the original position of the control points, whilst the filled circles are the incremented coordinates. This mesh is better than the one on the left due to the orthogonality of the mesh lines to the boundary around the leading edge.

3.3. Optimization of the active control points

Fig. 11 was obtained by manual movement of the control points, which visually seems to have improved the mesh smoothness and uniformity. To reduce the subjectivity of human perception, one should define a quality metric and aim to optimize it, using an optimization technique. After a number of experiments the following objective function was determined as most successful:

$$Obj = w_1 O_1 + w_2 O_2 + w_3 O_3$$

where

$$O_1 = nor\left[\sum_{i=1}^{N}\sum_{j=1}^{M}(\alpha_{i,j} - 360/M)^4\right].$$

$$O_2 = nor\left[\sum_{i=1}^{N}\sum_{j=2}^{M}[\|Node(i) - Node(j)\| - \|Node(i) - Node(j-1)\|]^4\right]$$

and

$$O_3 = nor\left[\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{1}{\|Node(i) - Node(j)\|}\right]$$

Here N is the number of nodes and M is the number of connections for node i. The $nor[\ldots]$ operator is a normalized value against the initial mesh, which is computed before any changes are made to the mesh. This ensures that all three objective functions are of the same order of magnitude, which is important for a weighted objective function approach.

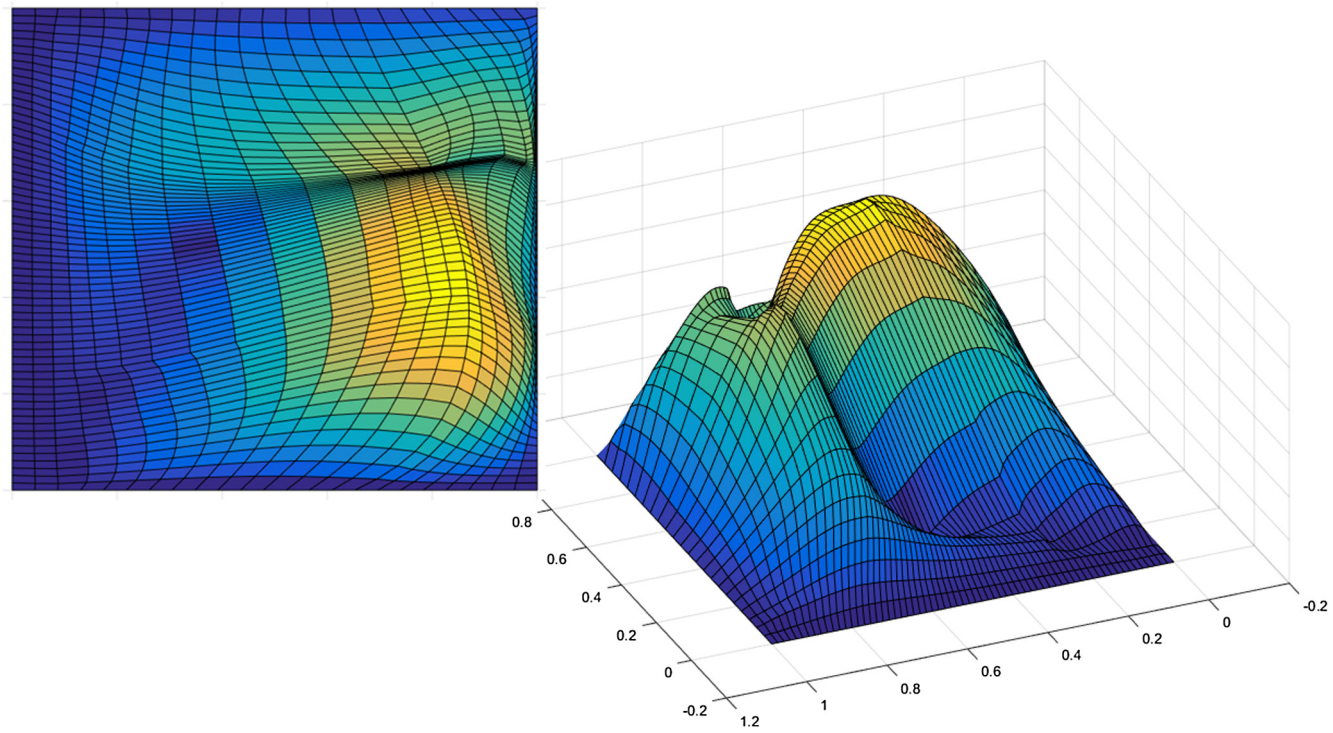

Fig. 9. Difference metric between the square and mapped original mesh.

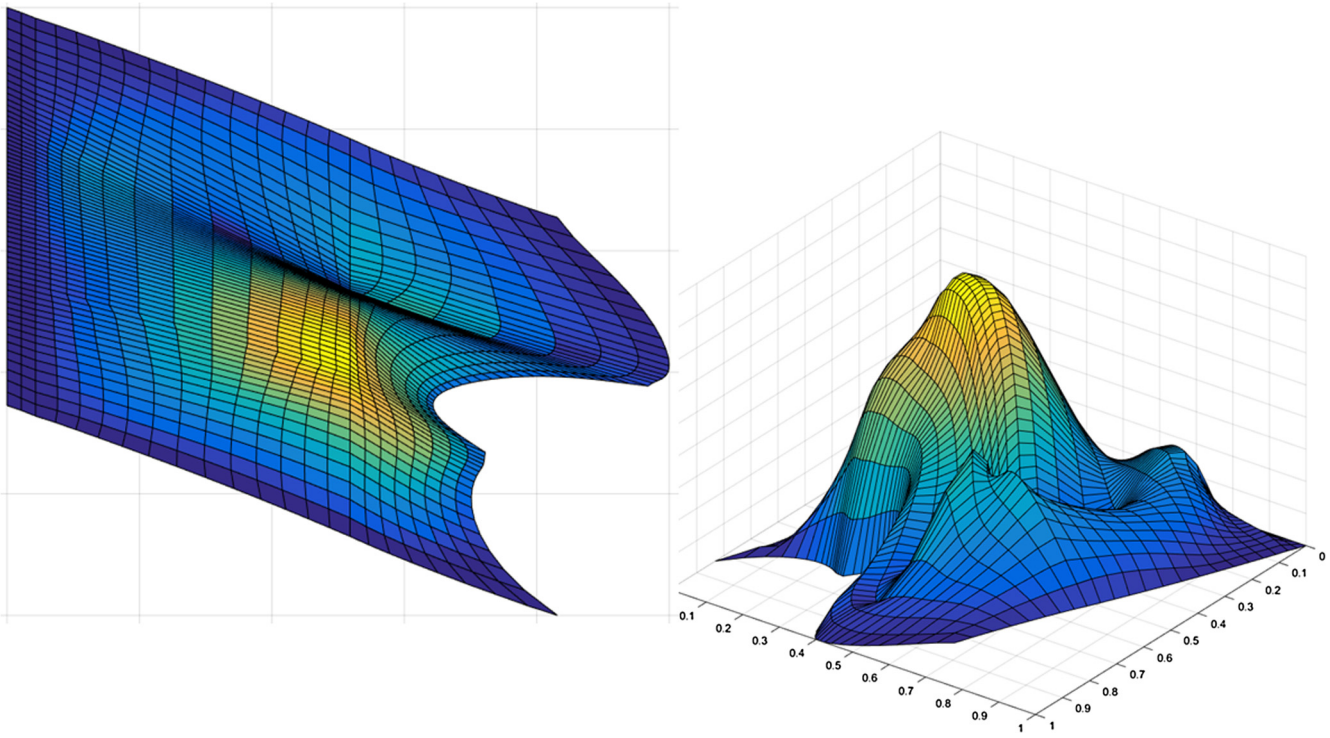

Fig. 10. Difference between transformed and original Block 4.

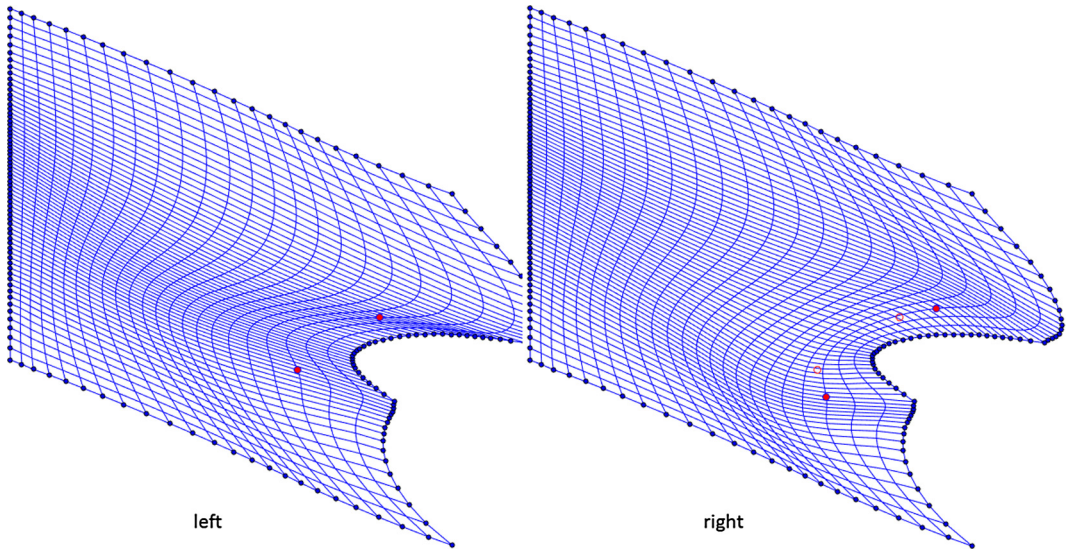

Fig. 11. Interpolative mapping using additional active control points, left – starting positions of the control points, right – new (filled) and old (empty) positions of the control points.

O_1 represents the difference of all nodal angles $\alpha_{i,j}$ and 90°, as shown in Fig. 12. For nodes with more than four connections ($M > 4$), the angles are adjusted respectively. Minimizing this function should make all angles around the nodes equal.

O_2 has a similar effect to keeping the Aspect ratio equal to 1, but also makes opposite connections equal and not just the adjacent ones. The $\|Node(i) - Node(j)\|$ operator represents the Euclidean distance between nodes i and j. Reducing this function will cause the connections to neighbouring nodes to become equal for all nodes.

O_3 is the crowding distance. Minimizing this function will cause neighbouring nodes to be pushed apart to avoid clustering.

To determine best values for the weighting coefficients w_1, w_2 and w_3, all possible combinations of 0, 0.25, 0.5, 0.75 and 1, which sum to 1 for the three coefficients were tested and the following best values were chosen by observation:

$$w_1 = 0.25;\ w_2 = 0.25;\ w_3 = 0.5.$$

Hence, minimizing the objective function *Obj* aims to produce a uniform, declustered, unskewed mesh.

To study mesh variation, a predefined set of active control points was selected. To demonstrate this idea a set of nine control points positioned in a grid of 3 × 3 was selected as shown in Fig. 13 left. The position of each control point was optimized using 18

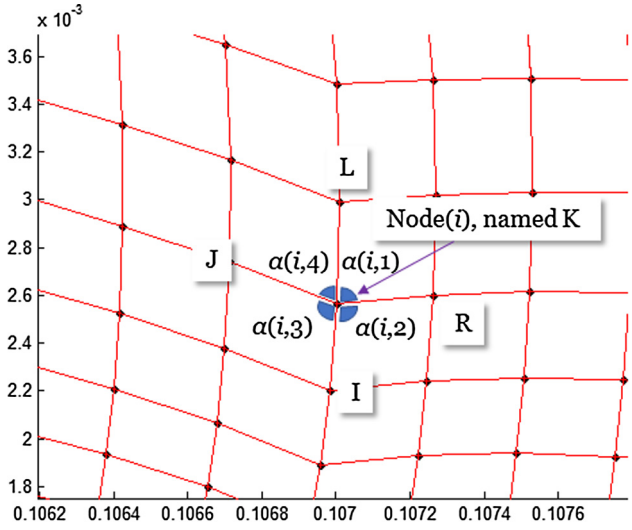

Fig. 12. Interpolative mapping using additional active control points – calculation of angle metric.

variables (X and Y increments for each of the points) varied by the MATLAB's 2014b, SQP algorithm (fmincon) (http://uk.mathworks.com/help/matlab/ref/scatteredinterpolant-class.html?refresh=true). The resultant mesh is shown in Fig. 13 right.

Grids of Active control points of 4×4, 5×5 and 6×6 were also studied without showing significant improvement for this block. Also tests were conducted using an exchange algorithm in an attempt to determine the optimal number and position of the control points. An initial collection of three control points were spaced out inside the mesh and their position optimized. If a control point displacement was less than a predefined value it was excluded from the set, indicating that there is no need of a control point at that location. A new control point is then added, maximizing the distance to existing points and the procedure repeated. The results were not satisfactory. The value of the objective function changed once a control point was removed, often leading to a situation where subsequent values would become worse than previously optimized. If old control points were fixed and only the newly added ones moved – then they produced local effects around them, disturbing the overall smoothness of the mesh. The iterations continued for considerable amount of time, which did not justify the cost of the search. It is therefore recommended to simply choose a set of points in a grid, spaced-out or latin hypercube manner and carry out a single optimization using all points.

3.4. Feature preservation

Let's next move our attention to the minor irregularity around the middle of Block 1, as shown in Fig. 14. We will create a patch around this area and apply the RBF cubic interpolant to the internal nodes.

The resultant mesh has lost the initial irregularity, but acquired another deformation, which is not deemed favourable. A logical conclusion is that the RBF cubic interpolant was unable to capture correctly the trend of this mesh using the fixed control points around the edges. Just as described above, we can add a set of active control points and run SQP to minimize the combined objective function. This produces a better mesh, shown in Fig. 15 right. Although it is a good looking solution one should keep in mind that it is achieved by optimizing nodes inside this local patch. In other cases the effect of not applying this optimization process on the rest of the nodes outside the patch could cause significant differences between the nodes inside and outside the patch. This might be a desired effect in some cases, but not if the goal is smoothness across all nodes (and block boundaries).

A slightly different and faster approach is to increase the number of fixed points, as shown in Fig. 16 and not to use active control points. It is faster, because it does not involve an iterative optimization procedure as with the previous approach.

Fixed control points are chosen in a manner to preserve the curves and features of the original mesh but leave the nodes needing repair free to move. The effect is a fixed mesh with preserved features.

3.5. The effect of the interpolant

Another option for the above problem is to change the interpolant. We next show the use of other interpolants, based on the initial set of fixed nodes around the edges. As seen in Figs. 17

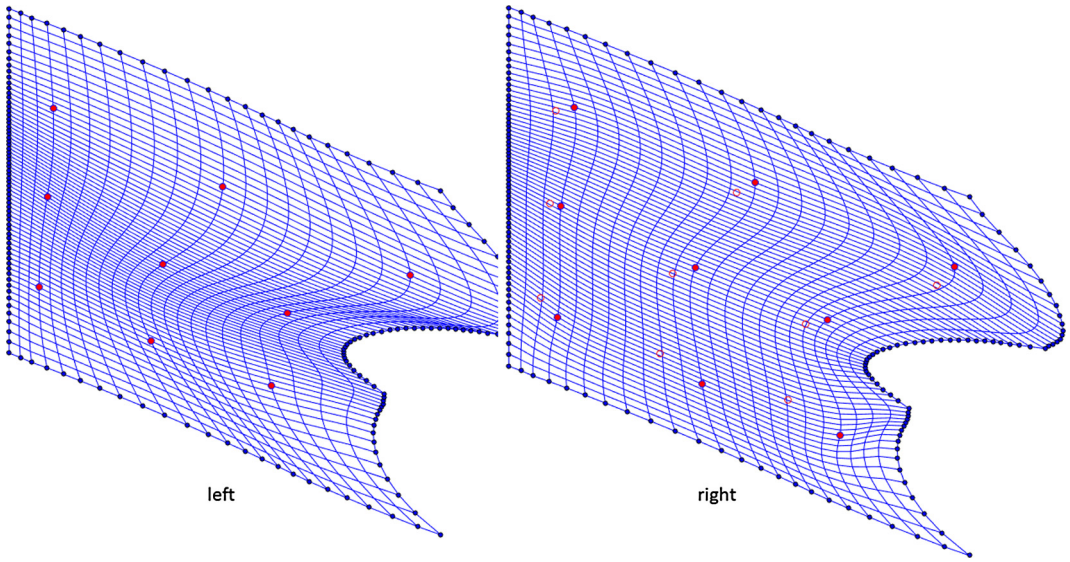

Fig. 13. Interpolative mapping using optimized active control points, left – starting positions of the control points, right – optimized (filled) and starting (empty) positions of the control points.

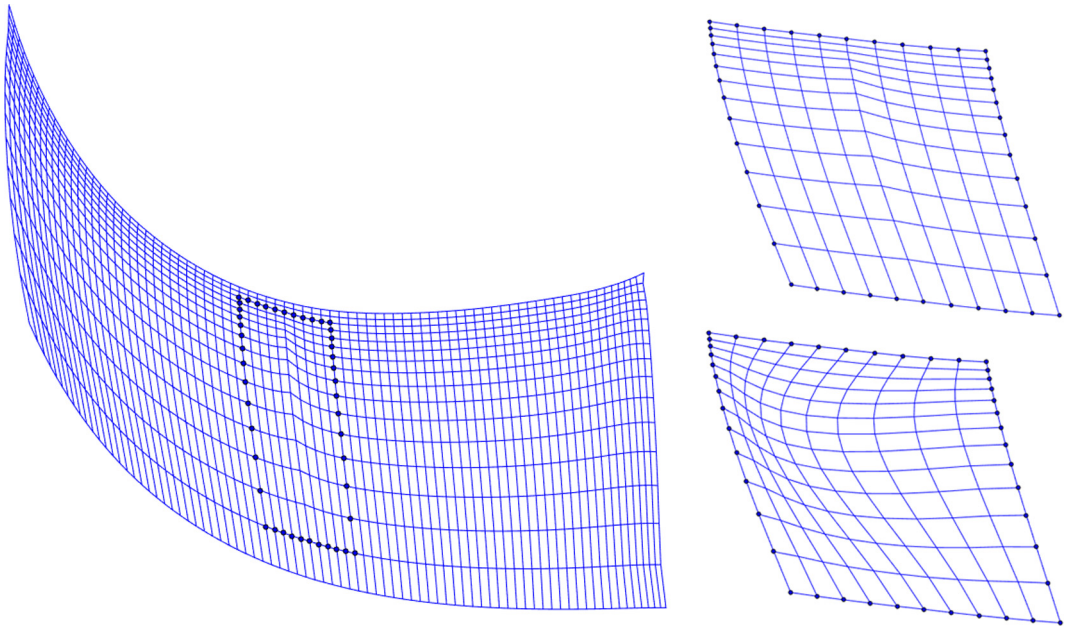

Fig. 14. Block 1 (left), patching (right top) and re-interpolation (right bottom) using RBF cubic spline.

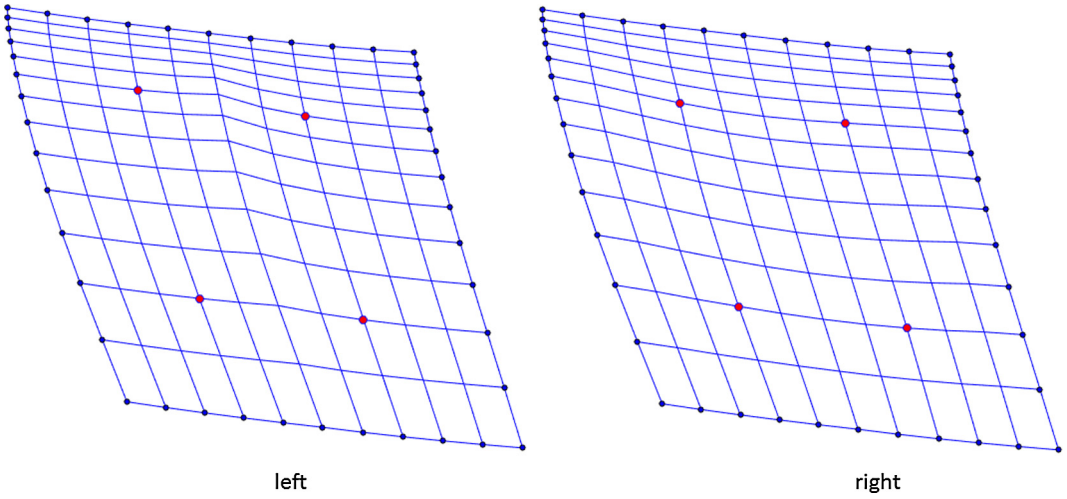

Fig. 15. RBF cubic spline and optimized control points – left – initial mesh, right – optimized mesh.

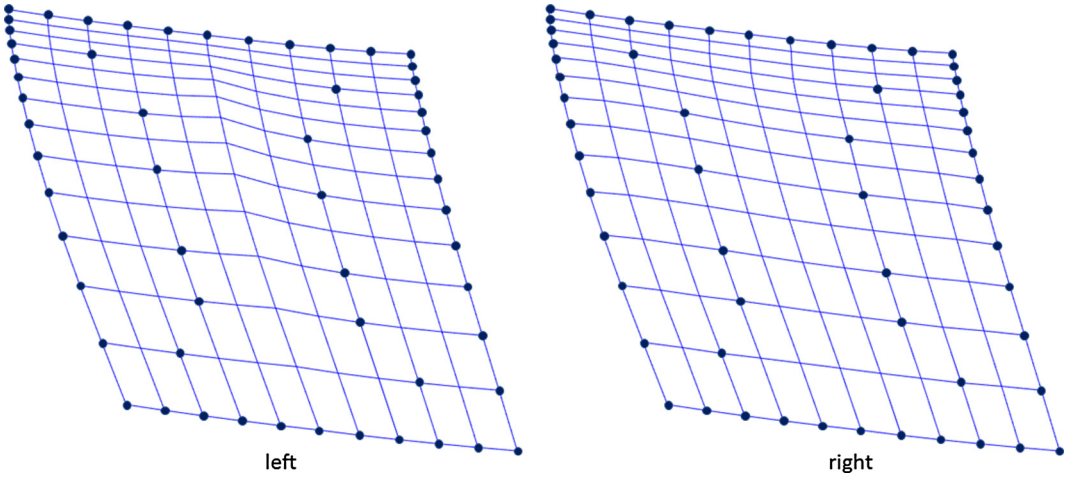

Fig. 16. RBF cubic spline with additional fixed control points – left – initial mesh, right – optimized mesh.

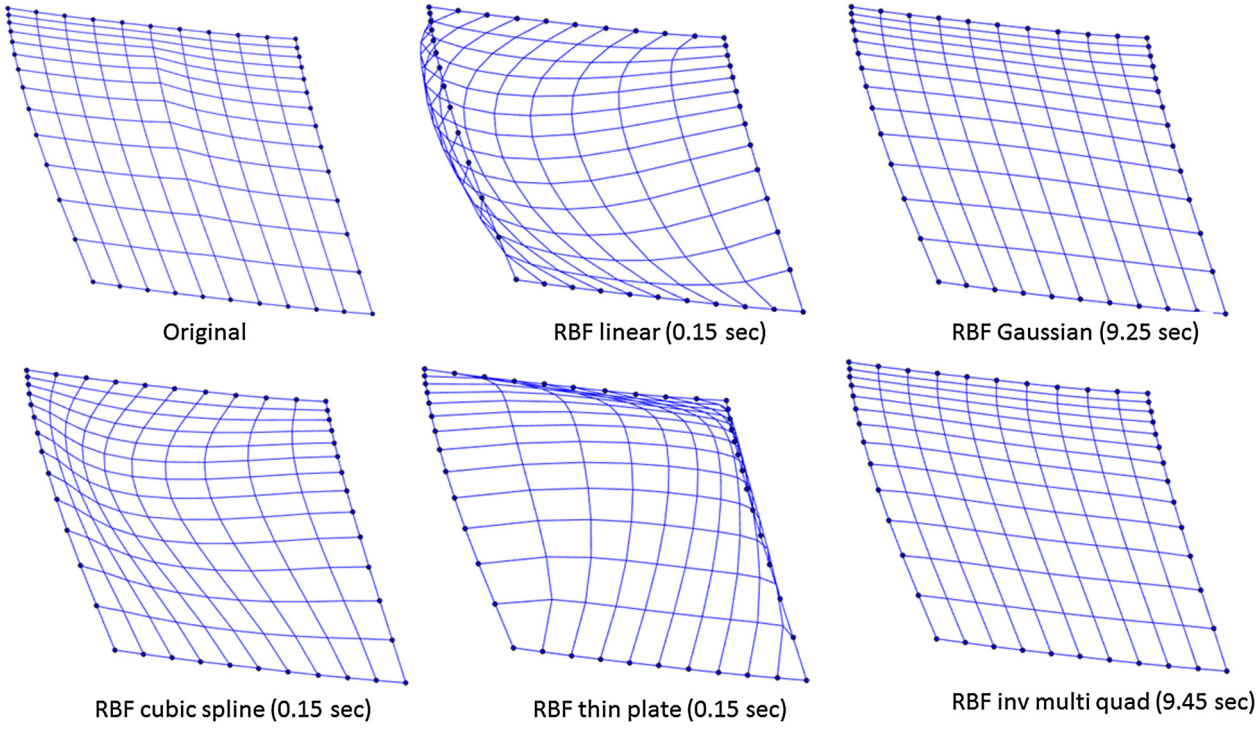

Fig. 17. RBF interpolants.

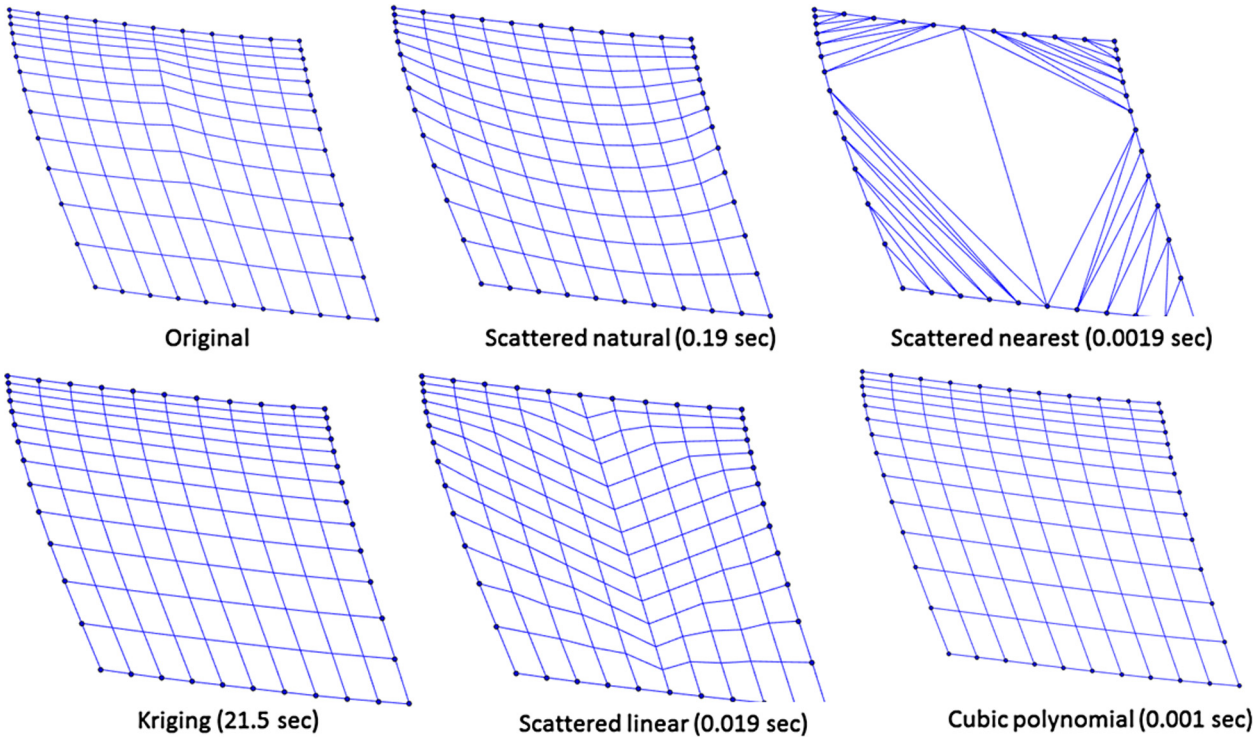

Fig. 18. Other interpolants and curve fitters.

and 18, the choice of interpolant has a significant effect on the re-interpolated mesh. Some are better than others – for instance RBF with Gaussian and inverse multi quadratic basis perform best amongst the RBF interpolants for this particular patch. Kriging (Jones et al., 1998; Nurdin et al., 2012) is another good technique, however with considerable time penalties due to the need to selectively tune hyper-parameters.

Note the result obtained by the cubic polynomial – it appears to have correct node shapes, however a closer inspection shows that the shape of the outer edge is slightly changed from the original.

This is possible because the polynomial does not refer to the training dataset during prediction, hence the prediction error is transferred to the training points as much as for the internals. All other interpolants would predict with no error at a training point. The 'Scattered' interpolants noted in Fig. 18 refer to the MATLAB's 'scatteredInterpolant' command, which uses Delaunay triangulation of scattered set of data to produce a surface. This surface can then be evaluated at any location (http://uk.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html).

Fig. 19 shows some of the better methods from the previous two figures applied to block 4. Results are significantly different due to the more complex shape of the geometry. The cubic polynomial significantly changes the boundary shape. Two interpolants have performed better but some have folded the mesh into itself creating negative volumes: 'Scattered natural' and 'RBF inverse multi quadratic'. Kriging has produced more accurate shape replication, however reducing the overall smoothness. It is possible to repair these problems using active control points, however it would significantly increase the cost for Kriging and RBF inverse multi quadratic. The ones with negative volumes can be fixed by carefully placing Active or Fixed Control points close to the problematic areas.

Based on these results and the time required to obtain each of them, we conclude that the RBF cubic spline should be the recommended first choice. Repair options should include adding fixed and active control points. If none of the above provides good results one should think about dividing the geometry into simpler/smaller blocks or patches, testing the number and position of the control points and changing the interpolants.

3.6. Working in three dimensions

We have illustrated the idea of interpolative mapping using a simple 2D case, however the method is not restricted to any particular number of dimensions. For three dimensions we update Eqs. (4) and (7)

$$R_{ij} = \left(\sqrt{(X_j(fixed) - X_{sq,i}(fixed))^2 + (Y_j(fixed) - Y_{sq,i}(fixed))^2 + (Z_j(fixed) - Z_{sq,i}(fixed))^2}\right)^3 \tag{9}$$

$$R_{ij} = \left(\sqrt{(X_i(fixed) - X_{sq,j}(int))^2 + (Y_i(fixed) - Y_{sq,j}(int))^2 + (Z_i(fixed) - Z_{sq,j}(int))^2}\right)^3 \tag{10}$$

and create a third interpolant:

$$Z_j(int) = \sum_{i=1}^{N} w_i^{(z)} R_{ij}. \tag{11}$$

The application of the method is shown in Fig. 20. Just as we used the edges in the 2D case, we will need to use all points on the walls for the 3D case. In the Block 4 case, some of the walls are used to join other blocks and that is why it is imperative that

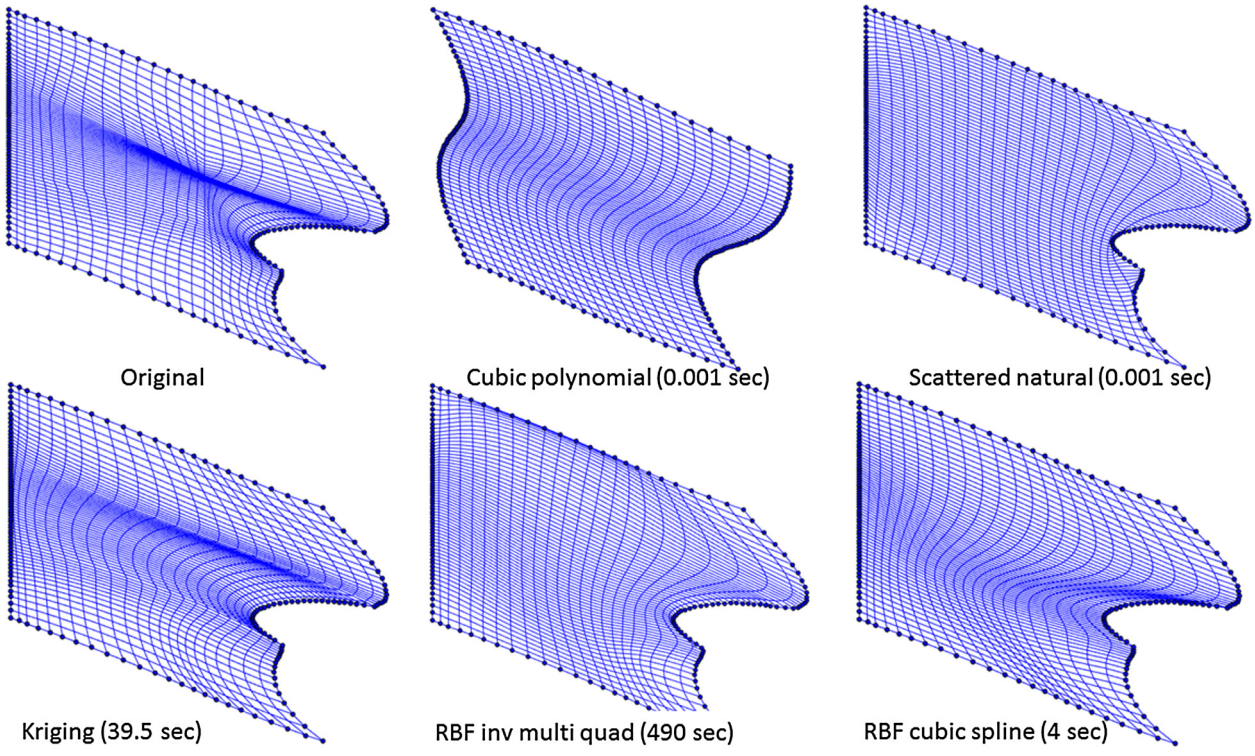

Fig. 19. Interpolants and curve fitters applied to block 4.

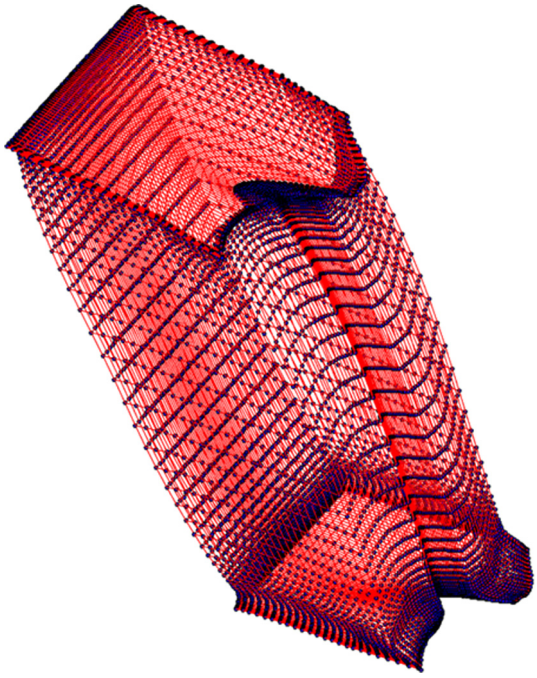

Fig. 20. Re-interpolated block 4 in 3D.

we use the walls exactly and not allow any movement there. Even the slightest offset will cause that wall to fail to connect to its neighbour. However there are walls that are free and some of the nodes on that wall can either be omitted or converted to active control points. This can be useful if a wall needs to be re-interpolated. Note that walls need not necessarily be flat. They could be curved surfaces and that is why the selection of fixed and active control points should be done with some care to preserve all 3D features.

With minor modification, we have adapted the control point procedure to suit 3D surfaces and volumes. Although results are similar, the time for completing an optimization run rises significantly from a few seconds to around an hour. The increase is due to both objective function calculation and RBF construction and prediction and the increased number of nodes. When dealing with large meshes it would clearly be necessary to work with small sub-mesh blocks, although many could be dealt with in parallel.

4. Results and automation

When dealing with structured meshes (Ali, Tucker, & Shahpar, 2017; Zhang, 2011), a 3D mesh (as shown in Fig. 20) can be viewed as a stack of surfaces and the process of extracting each surface is quick and straightforward.

It is also often true that adjacent surfaces differ only slightly from each other, although the end surfaces could be quite different. This leads to the idea that one could optimize control points for the first surface and whilst subsequent surfaces are similar one could simply re-interpolate them, using the change between the current coordinates and the optimized in the previous surface, relative to the change in the wall shape (Blanchard & Loubere, 2016; Gillebaart, Blom, van Zuijlen, & Bijl, 2016; Holloway, Grimm, & Ju, 2016; Imai, Hiraoka, & Kawaharada, 2014; Lu, Quadros, & Shimada, 2017; Niu, Lei, & He, 2017; Poirier & Nadarajah, 2016; Renka, 2015; Sieger, Gaulik, Achenbach, Menzel, & Botsch, 2016).

Let us consider the first surface of the 3D stack, as shown in Fig. 21. Drawn in a 3D coordinate system (Fig. 21, left) it is seen that the surface is oriented so that it is not parallel to any of the three axes and it is slightly curved. Our first step would be to re-interpolate the surface within its bounds, by mapping them to a perfect square as discussed earlier in this paper, however some changes are required to facilitate the fact that it is 3D surface instead of 2D.

We build RBF interpolants for X with respect to (x, y), for Y with respect to (x, y) and for Z also with respect to (x, y). After the mapping we will have predicted values for Xpred(x, y) and Ypred(x, y) which we then use to predict Zpred$(Xpred, Ypred)$. This ensures that points moving in X and Y will stay on the same surface with respect to Z.

Because the curvature of the surface is only small, we will use pictures of 2D projections, as in Fig. 21 right to illustrate the proposed idea, whilst keeping in mind that this is a 3D surface and any movement in X, Y automatically map to correct Z coordinate using the above interpolation.

As before, the process starts with mapping the edges of this shape to a perfect square and then we use this mapping to interpolate back the internal points as illustrated in Fig. 22. On some occasions more than others, RBF smoothing leads to lines getting too

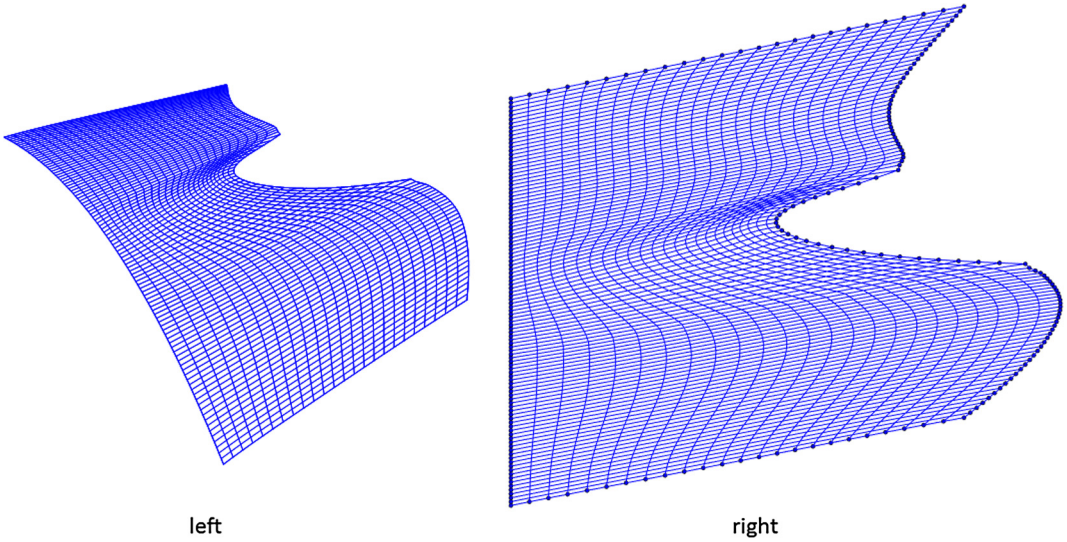

Fig. 21. Extract of the first 3D surface: left – 3D view, right – 2D view.

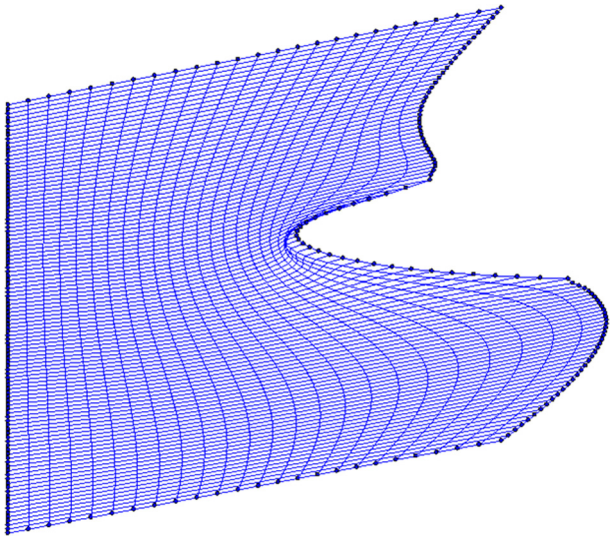

Fig. 22. Re-interpolated surface.

close to each other, or even crossing, resulting in inverted elements. We have constructed a function which automatically checks for such occurrences and we use this as a constraint. In the event of inverted elements, such as in Fig. 22, we can remedy the situation by detecting which nodes are causing it and setting them as active control points, followed by a new re-interpolation step. The result is shown in Fig. 23. The re-interpolation was repeated with no movement of the active control point (in effect acting as a fixed one), and the line crossing is prevented. Compared to Fig. 21, this mesh is already much better in its smoothness and was achieved very quickly without any iterative optimization steps.

As earlier, one may wish to improve certain metrics of the mesh, to achieve for instance a better CFD convergence. Let us assume that one such metric is related to the angles at each node and they need to be as close as possible to 90 degrees. Intuitively we would like to be able to manipulate the positions of the worst nodes, without worsening the metric around the rest of the nodes. To automate the process, we calculate the metric function for all nodes and choose a percentage of the worst – Fig. 24. Then we identify N clusters using the k-mean algorithm and extract their

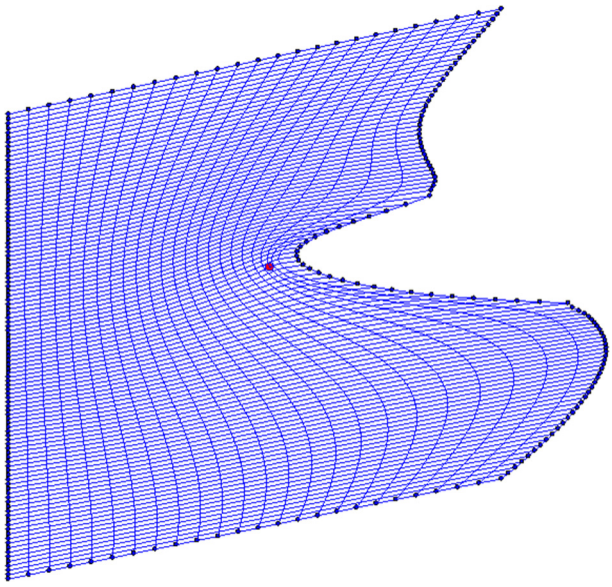

Fig. 23. Re-interpolated surface with a control point.

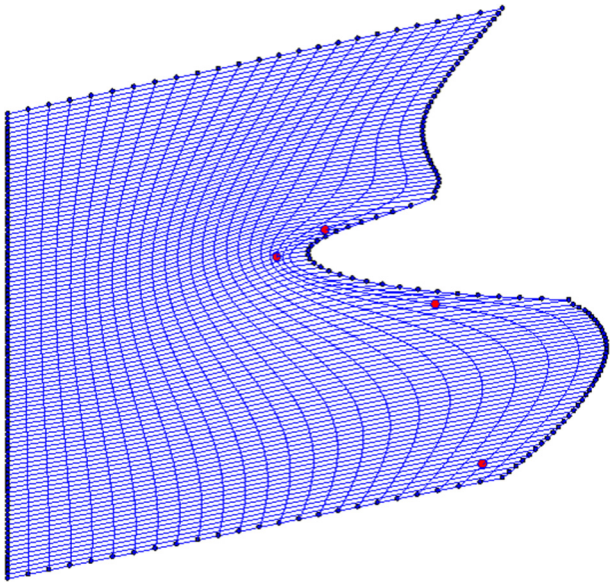

Fig. 25. Adding three k-mean centroids as Active points for the angles.

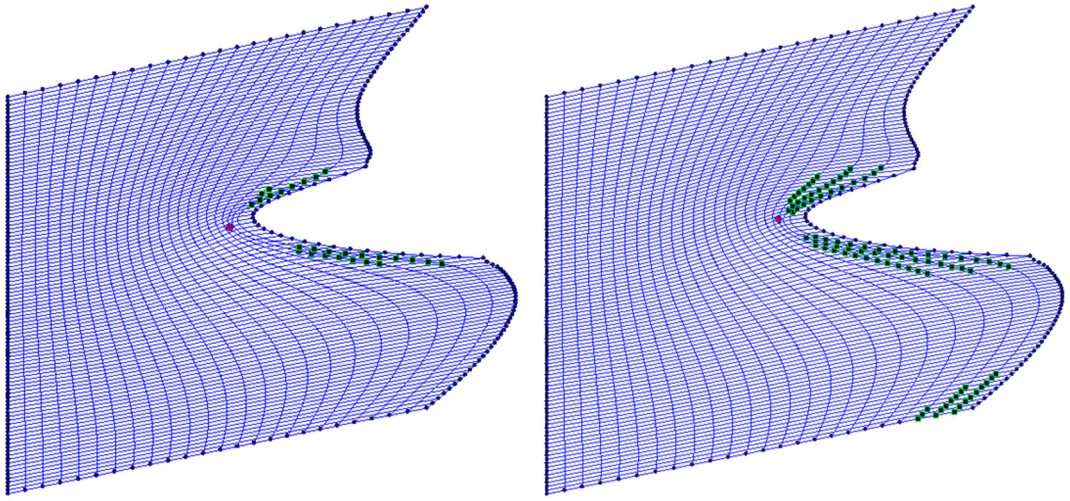

Fig. 24. Respectively 10% (left) and 20% (right) worst nodes with respect to angles.

centroids, as shown in Fig. 25. The percentage of the worst points as well as the number of clusters are variables which need to be set depending on how distorted the original mesh is. Often only few Active control points are necessary, but with more complex shapes the required number increases.

We repeat the process applying all other mesh quality metrics such as aspect ratio and size and we also add few control points around the rest of the mesh by identifying the cloud of points which is the result of subtracting the ones chosen above from all nodes and again applying k-mean clustering. The final set of control points is shown in Fig. 26. Coordinates of the Active Control Points are then manipulated using an optimization algorithm that minimizes the combined metric. Sequential quadratic programming (SQP), Nedler-Mead or Pattern search are all suitable algorithms for this problem. Due to the number of variables being twice as much as the Active Control Points (X and Y) it may take significant amount of time to complete the optimization with high accuracy.[1] An optimized mesh is shown in Fig. 27. It constitutes a better mesh for CFD due to the proper formation of high quality nodes around the boundary area.

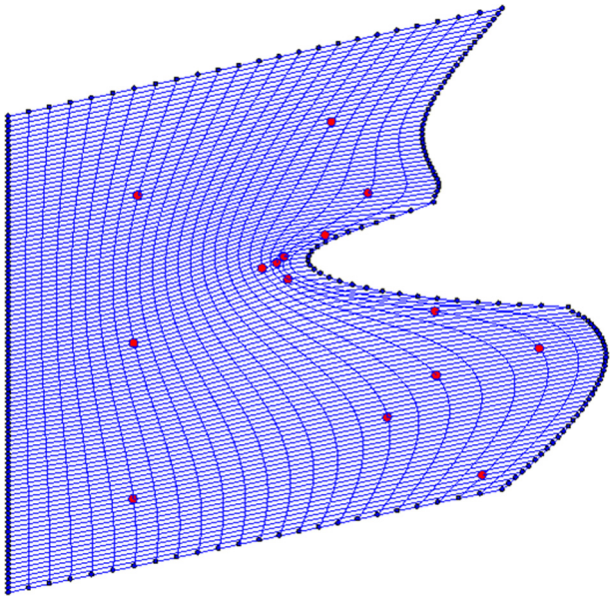

Fig. 26. Final active points setup.

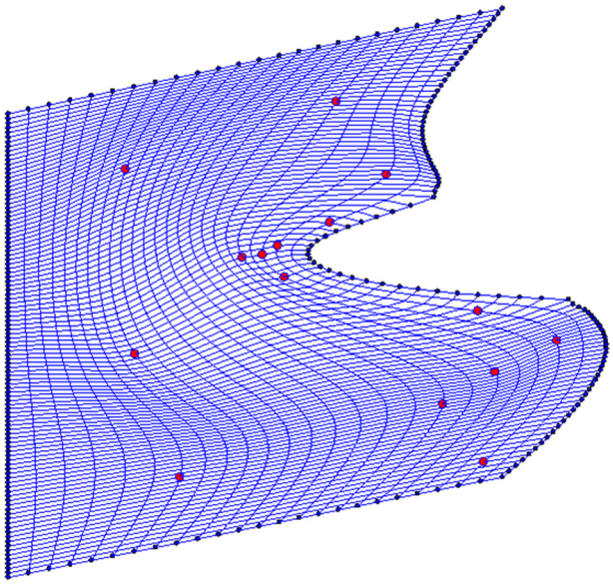

Fig. 27. Optimized mesh.

The above procedure can be repeated for every slice in the 3D mesh, by using the control points and their optimized movement of the previous slice as a starting point. Since neighbouring slices are similar to each other, the optimization takes less time and sometimes doesn't even begin as a move in every direction is worse than the current. This greatly reduces the amount of time and computer resources required to optimize the entire 3D mesh.

5. Conclusions

The mesh smoothing methods proposed here are based on interpolative mapping and provide a quick and simple approach for local mesh improvement. The procedure starts with a relatively low number of control points. One could envisage a software package which allows the user to interactively select control points, move them around manually, see the results immediately or use an optimization strategy. The objective functions can be redefined to suit a range of needs. The method can be used to construct new meshing algorithms or improve existing ones.

The ability to add control points makes the method applicable for a variety of applications. It has been shown that adding an appropriate combination of fixed and active control points can keep the number of points being manipulated down whilst producing good results.

We have shown that the method is applicable to 2D meshes as well as to 3D surfaces and 3D volumes with increased computational effort and proposed when possible just dealing with individual 2D or 3D surface slices. We have also suggested an automated control point selection based on quality metrics and k-mean clustering.

Interpolative mapping can be applied to a geometry with any number of walls, provided that one uses an equivalent primitive geometry.

Conflict of interest

None.

References

Ali, Z., Tucker, P. G., & Shahpar, S. (2017). Optimal mesh topology generation for CFD. *Computer Methods in Applied Mechanics and Engineering, 317*, 431–457. 15 April 2017.

ANSYS Meshing Controls: Mapped Face Meshing. <http://www.ansys.com/Products/Workflow+Technology/ANSYS+Workbench+Platform/ANSYS+Meshing/Features/Meshing+Controls:+Mapped+Mesh+Controls>.

Blanchard, G., & Loubere, R. (2016). High order accurate conservative remapping scheme on polygonal meshes using a posteriori MOOD limiting. *Computers & Fluids – UK, 136*, 83–103. https://doi.org/10.1016/j.compfluid.2016.06.002. Published: SEP 10 2016.

Car, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., & Evans, T. R. (2001). Reconstruction and representation of 3D objects with Radial Basis Functions, SIGGRAPH 2001 Conference proceedings, Book Series: *Computer graphics* (pp. 67–76). <http://www-lb.cs.umd.edu/class/spring2005/cmsc828v/papers/siggraph01.pdf>.

Erten, H., Ungor, A., & Zhao, C. (2009). Mesh smoothing algorithms for complex geometric domains. In *Proceedings of the 18th international meshing roundtable* (pp. 175–193), Sandia. <http://www.imr.sandia.gov/papers/imr18/Erten.pdf>.

Gillebaart, T., Blom, D. S., van Zuijlen, A. H., & Bijl, H. (2016). Adaptive radial basis function mesh deformation using data reduction. *Journal of Computational Physics, 321*, 997–1025. https://doi.org/10.1016/j.jcp.2016.05.036. Published: Sep 15 2016.

[1] Accurate determination of the optima is difficult, because the objective function is flat around the basin of attraction, where large variation of the control points coordinates result in minimal changes of the objective function, which in turn may lead to locating local minima or premature triggering of the search procedure stop criteria.

Holloway, M., Grimm, C. & Ju, T. (2016). Template-based surface reconstruction from cross-sections. Computers & Graphics – UK, 58(Special Issue: SI), 84–91. http://doi.org/10.1016/j.cag.2016.05.012. Published: AUG 2016.

Imai, Y., Hiraoka, H., & Kawaharada, H. (2014). Quadrilateral mesh fitting that preserves sharp features based on multi-normals for Laplacian energy. *Journal of Computational Design and Engineering, 1*(2), 88–95. ISSN 2288-4300.

Jakobsson, S., & Amoignon, O. (2007). Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids, 36*(6), 1119–1136 <http://www.sciencedirect.com/science/article/pii/S0045793006001320?np=y>.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization, 13*, 455–492.

Keane, A. (2004). Design search and optimisation using radial basis functions with regression capabilities. In I. Pramee (Ed.), *Adaptive computing in design and manufacture VI, Pramee I* (pp. 39–49). Springer.

Lu, J. H.-C., Quadros, W. R., & Shimada, K. (2017). Evaluation of user-guided semi-automatic decomposition tool for hexahedral mesh generation. *Journal of Computational Design and Engineering, 4*(4), 330–338. ISSN 2288-4300.

Mapped Face Meshing in ANSYS Workbench. <http://www.padtinc.com/blog/the-focus/mapped-face-meshing-in-ansys-workbench>.

Milli, A., & Shahpar, S. (2012). PADRAM: Parametric Design and Rapid Meshing System for Complex Turbomachinery Configurations, Paper No. GT2012-69030, pp. 2135–2148, ASME Turbo Expo 2012, *Turbine Technical Conference and Exposition*, Volume 8: Turbomachinery, Parts A, B and C, Copenhagen, Denmark, June 11–15, 2012, ISBN: 978-0-7918-4474-8, Copyright 2012 by Rolls-Royce plc. <http://www.researchgate.net/publication/267569134_PADRAM_Parametric_Design_and_Rapid_Meshing_System_for_Turbomachinery_Optimisation>.

Niu, J. P., Lei, J. M., & He, J. D. (2017). Radial basis function mesh deformation based on dynamic control points. *Aerospace Science and Technology, 64*, 122–132. https://doi.org/10.1016/j.ast.2017.01.022. Published: May 2017.

Nurdin, A., Bressloff, N., & Keane, A. (2012). Shape optimisation using CAD linked free-form deformation. *The Aeronautical Journal, 115*(1183), 915–939.

Poirier, V., & Nadarajah, S. (2016). Efficient reduced-radial basis function-based mesh deformation within an adjoint-based aerodynamic optimization framework. *Journal of Aircraft, 53*(6), 1905–1921. https://doi.org/10.2514/1.C033573. Published: Nov 2016.

Renka, R. (2015). Mesh improvement by minimizing a weighted sum of squared element volumes. *International Journal for Numerical methods in Engineering, 101*(11), 870–886. March 2015.

Savchenko, V., Savchenko, M., Egorova, O., & Hagiwara, I. (2008). Mesh quality improvement: Radial basis functions approach. *International Journal of Computer Mathematics, 85*(10), 1589–1607. https://doi.org/10.1080/00207160802033624.

Shontz, S., & Vavasis, S. (2004). *A mesh warping algorithm based on weighted Laplacian smoothing*. Ithaca: Center for Applied Mathematics Cornell University. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.7349&rep=rep1&type=pdf>.

Sieger, D., Menzel, S., & Botsch, M. (2013). High quality mesh morphing using triharmonic radial basis functions. In *Proceedings of the 21st international meshing roundtable* (pp. 1–16). Springer, ISBN 978-3-642-33572-3. <http://www.imr.sandia.gov/papers/imr21/Sieger.pdf>.

Sieger, D., Gaulik, S., Achenbach, J., Menzel, S., & Botsch, M. (2016). Constrained space deformation techniques for design optimization. *Computer-Aided Design, 72*, 40–51. March 2016.

Staten, M. L., Owen, S. J., Shontz, S. M., Salinger, A. G., & Coffey, T. S. (2011). A comparison of mesh morphing methods for 3D shape optimization. In *Proceedings of the 20th international meshing roundtable* (pp. 293–311). <http://www.imr.sandia.gov/papers/imr20/Staten.pdf>.

Winslow, A. M. (1967). Numerical solution of the quasilinear poisson equations in a nonuniform triangle mesh. *Journal of Computational Physics, 2*(149–172), 1967.

Zhang, Q. (2011). High-order, multidimensional, and conservative coarse–fine interpolation for adaptive mesh refinement. *Computer Methods in Applied Mechanics and Engineering, 200*(45–46), 3159–3168. 15 October 2011.