Available online at www.sciencedirect.com
ScienceDirect
Computers and Structures 85 (2007) 784–795
Computers & Structures
www.elsevier.com/locate/compstruc

Mesh deformation based on radial basis function interpolation

A. de Boer *, M.S. van der Schoot, H. Bijl

Faculty of Aerospace Engineering, Delft University of Technology, P.O. Box 5058, 2600 GB Delft, The Netherlands

Received 18 July 2006; accepted 2 January 2007
Available online 8 March 2007

Abstract

A new mesh movement algorithm for unstructured grids is developed which is based on interpolating displacements of the boundary nodes to the whole mesh with radial basis functions (RBF's). A small system of equations, only involving the boundary nodes, has to be solved and no grid-connectivity information is needed. The method can handle large mesh deformations caused by translations, rotations and deformations, both for 2D and 3D meshes. However, the performance depends on the used RBF. The best accuracy and robustness with the highest efficiency are obtained with a C^2 continuous RBF with compact support, closely followed by the thin plate spline. The deformed meshes are suitable for flow computations as is shown by performing calculations around a NACA-0012 airfoil.

Keywords: Mesh deformation; Unstructured meshes; Radial basis function interpolation; Fluid–structure interaction; Point-to-point scheme; Mesh quality quantification

1. Introduction

Fluid–structure interaction computations typically involve moving boundaries for the flow due to the deformation of the structure. Examples can be found in: flutter simulation of wings, blood flow through veins and stability analysis of bridges and tall buildings subjected to windloads. To be able to perform the unsteady flow computations accurately and efficiently, a fast and reliable method is needed to adapt the computational grid to the new domain. Regenerating a grid each time step in an unsteady computation is a natural choice. However, the generation of a complex grid is a time-consuming and nontrivial task. Therefore, a fast and accurate algorithm is needed to update the grid automatically.

For structured meshes there are efficient techniques available to deform the mesh, such as Transfinite Interpolation [1]. The displacements of points at the boundaries of the mesh are interpolated along grid lines to points in the interior of the mesh. However, these techniques are unsuitable for unstructured grids. The greater flexibility of unstructured grids is required for the meshing of complex domains and grid adaptation. Therefore, we are in this paper interested in efficient mesh movement techniques for unstructured grids.

Two different mesh movement strategies are known for unstructured grids. The first exploits the connectivity of the internal grid points. The connection between the grid points is represented for example by springs [2–4] or as solid body elasticity [5]. Special instances of this continuous approach include moving grids based on Laplacian and Biharmonic operators [6]. The method based on Laplacian smoothing is for example successfully applied in [7 and 8]. All the methods based on grid connectivity involve solving a system of equations involving all the flow points and can therefore be quite expensive. Hanging nodes, often encountered in unstructured meshes, require special treatment.

The other strategy moves each grid point individually based on its position in space and this results in the so called point-by-point schemes. Hanging nodes are no problem and also the implementation for partitioned meshes, occurring in parallel flow computations, is straightforward.

* Corresponding author. Tel.: +31 15 27 82046; fax: +31 15 27 87077.
E-mail address: A.deBoer@tudelft.nl (A. de Boer).

0045-7949/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2007.01.013

This might be especially useful for Finite Volume flow solvers which do not incorporate efficient algorithms to deform the mesh with a pseudo-structural approach. However, until now point-by-point schemes are only applied to the boundary nodes of multi-grid blocks [9]. The interior mesh of the blocks is adapted with fast techniques available for structured grids.

Radial basis functions (RBF's) have become a well-established tool to interpolate scattered data. They are for example used in fluid–structure interaction computations to transfer information over the discrete fluid–structure interface, which is often non-matching [10,11]. An interpolation function is used to transfer the displacements known at the boundary of the structural mesh to the boundary of the aerodynamic mesh. But why not interpolate the displacement to all the nodes of the flow mesh, instead of only to the boundary? This idea has already been applied to the block boundaries in multi-block grids [9,12]. There it was mentioned that applying it to the whole internal grid would be computationally very expensive. This is because for the structured part of multi-block meshes much more efficient techniques are known. We want to investigate if interpolation of the displacement with radial basis functions does result in an efficient point-by-point mesh movement scheme for completely unstructured grids.

The objective of this paper is to develop a new mesh movement scheme for unstructured meshes based on interpolation with radial basis functions. We outline the principle of interpolation with RBF's applied to mesh movement. There are various RBF's available literature that can be used for the new method and we want to determine which one generates the best meshes and which one is the most efficient. To be able to compare the deformed meshes generated with the different RBF's, a mesh quality metric is introduced. This metric is used to determine the best RBF's for our mesh movement scheme, by applying the method to several severe test cases. For one of the test cases the results are also compared with mesh deformation using semi-torsional springs. Furthermore realistic results on a distorted mesh are presented, where flow computations are performed around a NACA-0012 airfoil. Finally it is shown that the method is capable of deforming 3D unstructured hexahedral meshes.

2. Radial basis function interpolation

Radial basis function interpolation can be used to derive the displacement of the internal fluid nodes given the displacement of the structural nodes on the interface. The interpolation function, s, describing the displacement in the whole domain, can be approximated by a sum of basis functions

$$s(\mathbf{x}) = \sum_{j=1}^{n_b} \alpha_j \phi(\|\mathbf{x} - \mathbf{x}_{b_j}\|) + p(\mathbf{x}), \tag{1}$$

where $\mathbf{x}_{b_j} = [x_{b_j}, y_{b_j}, z_{b_j}]$ are the centers in which the values are known, in this case the boundary nodes, p a polynomial, n_b the number of boundary nodes and ϕ a given basis function with respect to the Euclidean distance $\|\mathbf{x}\|$. The coefficients α_j and the polynomial p are determined by the interpolation conditions

$$s(\mathbf{x}_{b_j}) = \mathbf{d}_{b_j}, \tag{2}$$

with $\mathbf{d}_b$ containing the discrete known values of the displacement at the boundary, and the additional requirements

$$\sum_{j=1}^{n_b} \alpha_j q(\mathbf{x}_{b_j}) = 0, \tag{3}$$

for all polynomials q with a degree less or equal than that of polynomial p. The minimal degree of polynomial p depends on the choice of the basis function ϕ. A unique interpolant is given if the basis function is a conditionally positive definite function. If the basis functions are conditionally positive definite of order $m \leqslant 2$, a linear polynomial can be used [10]. In this paper we only apply basis functions that satisfy this criterion. A consequence of using a linear polynomial is that rigid body translations are exactly recovered.

The values for the coefficients α_j and the linear polynomial can be obtained by solving the system

$$\begin{bmatrix} \mathbf{d}_b \\ 0 \end{bmatrix} = \begin{bmatrix} M_{b,b} & P_b \\ P_b^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{4}$$

with $\boldsymbol{\alpha}$ containing the coefficients α_j, $\boldsymbol{\beta}$ the coefficients of the linear polynomial p, $M_{b,b}$ an $n_b \times n_b$ matrix containing the evaluation of the basis function $\phi_{b_i b_j} = \phi(\|\mathbf{x}_{b_i} - \mathbf{x}_{b_j}\|)$ and P_b an $n_b \times 4$ matrix with row j given by $\begin{bmatrix} 1 & x_{b_j} & y_{b_j} & z_{b_j} \end{bmatrix}$. Solving the system can be done by fast iterative techniques [13,14] or using partition of unity [15].

The values for the displacement in the interior of the flow mesh $\mathbf{d}_{in}$, can then be derived by evaluating the interpolation function (1) in the internal grid points:

$$\mathbf{d}_{in_j} = s(\mathbf{x}_{in_j}). \tag{5}$$

The displacement can be interpolated separately for each spatial direction. The process of mesh deformation with RBF's for one direction is visualized in Figs. 1 and 2. Each point is moved individually based on its position in space according to the interpolation function, and this means that no mesh-connectivity information is needed at all. However, to be able to see the effects on the mesh clearly, the points are connected in a very simple mesh as is shown in Fig. 1. The block in the middle is moved to the right and the resulting interpolation function is shown in Fig. 2. This function is equal to zero at the outer boundaries and equal to the displacement of the block at the location of the block boundaries. The displacement of the nodes in the interior of the mesh can then be derived from this function and the resulting deformed mesh is shown in Fig. 2.

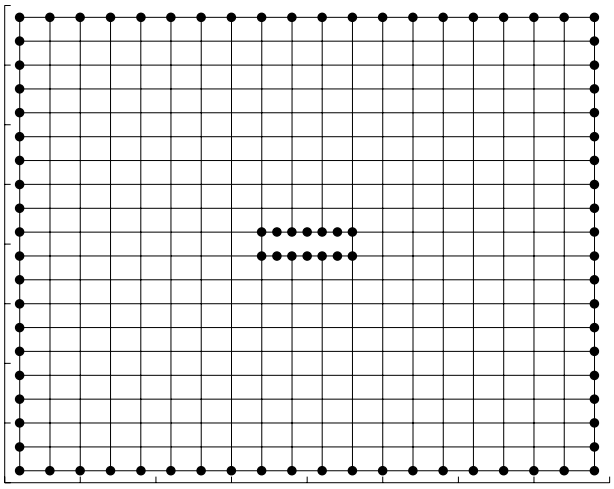

Fig. 1. Initial mesh.

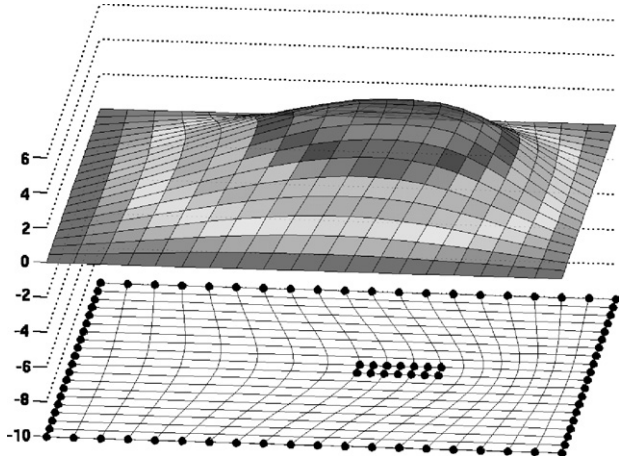

Fig. 2. Interpolation function and resulting deformed mesh.

The size of the system that has to be solved in (4) is equal to $(n_b + 4) \times (n_b + 4)$ which is usually very small compared to the systems that have to be solved in mesh-connectivity schemes. The systems encountered there are approximately as large as $n_{in} \times n_{in}$, with n_{in} the total number of mesh points. The total number of mesh points is a dimension higher than the number of points on the boundary of the mesh. The new moving mesh technique is very easy to implement, even for 3D applications, because no mesh-connectivity information is needed. Also the implementation for partitioned meshes, occurring in parallel flow computations, is straightforward.

There are various radial basis function available in literature which are suitable for interpolating multivariate data. They can be divided in two groups: functions with compact and functions with global support. Functions with compact support have the following property:

$$\phi(x) = \begin{cases} f(x) & 0 \leqslant x \leqslant 1, \\ 0 & x > 1, \end{cases} \tag{6}$$

Table 1
Radial basis functions with compact support, from [16]

No.	Name	$f(\xi)$
1	CP C^0	$(1-\xi)^2$
2	CP C^2	$(1-\xi)^4(4\xi+1)$
3	CP C^4	$(1-\xi)^6\left(\frac{35}{3}\xi^2+6\xi+1\right)$
4	CP C^6	$(1-\xi)^8(32\xi^3+25\xi^2+8\xi+1)$
5	CTPS C^0	$(1-\xi)^5$
6	CTPS C^1	$1+\frac{80}{3}\xi^2-40\xi^3+15\xi^4-\frac{8}{3}\xi^5+20\xi^2\log(\xi)$
7	CTPS C^2_a	$1-30\xi^2-10\xi^3+45\xi^4-6\xi^5-60\xi^3\log(\xi)$
8	CTPS C^2_b	$1-20\xi^2+80\xi^3-45\xi^4-16\xi^5+60\xi^4\log(\xi)$

where $f(x) \geqslant 0$. The function is generally scaled with a support radius r to control the compact support, so $\phi_r = \phi(x/r)$. When a radial basis function with compact support is used, mainly the mesh nodes inside a circle (2D) or sphere (3D) with radius r around a center x_j are influenced by the movement of this center. Therefore, higher values for the support radius lead generally to more accurate solutions. However, high values of the support radius r also result in dense matrix systems, whereas low values of r result in sparse matrix systems which can be solved more efficiently.

In Table 1 various radial basis functions with compact support are given. In this paper all compact RBF's are scaled with r, so we use $\xi = x/r$. The first four are based on polynomials [16]. These polynomials are chosen in such a way that they have the lowest degree of all polynomials that create a C^n continuous basis function with $n \in \{0, 2, 4, 6\}$. The last four are a series of functions based on the thin plate spline which create C^n continuous basis functions with $n \in \{0, 1, 2\}$ [16]. There are two possible CTPS C^2 continuous functions which are distinguished by subscript a and b.

Functions with global support are not equal to zero outside a certain radius, but cover the whole interpolation space, which leads to dense matrix systems. In Table 2 six radial basis functions are given which are frequently used, for example in neural networks, the computer graphics community [17] and for data transfer in fluid–structure interaction computations [11]. The MQB and IMQB methods use a parameter a, that controls the shape of the basis functions. A large value of a gives a flat sheetlike function, whereas a small value of a gives a narrow conelike function. The value of a is typically chosen in the range 10^{-5}–10^{-3} and in this paper we use the value $a = 10^{-3}$. More information about RBF's and their error and convergence properties can be found in [13,18,19].

Table 2
Radial basis functions with global support

No.	Name	Abbreviation	$f(x)$
9	Thin plate spline	TPS	$x^2\log(x)$
10	Multiquadric biharmonics	MQB	$\sqrt{a^2+x^2}$
11	Inverse multiquadric biharmonics	IMQB	$\sqrt{\frac{1}{a^2+x^2}}$
12	Quadric biharmonics	QB	$1+x^2$
13	Inverse quadric biharmonics	IQB	$\frac{1}{1+x^2}$
14	Gaussian	Gauss	e^{-x^2}

The new mesh movement scheme based on interpolation with radial basis functions will be tested with the different RBF's introduced in this section, but first a mesh quality metric is given to be able to compare the quality of the meshes after deformation.

3. Mesh quality metrics

To be able to compare the quality of different meshes after mesh movement we introduce mesh quality metrics [20]. The mesh quality metrics are based on a set of Jacobian matrices which contain information on basic element qualities such as size, orientation, shape and skewness.

It is assumed that the initial mesh is generated in an optimal way and therefore the element shapes should be changed as little as possible after deformation. This means that both the volume and the angles of the elements should be preserved. These two properties can be measured with the relative size and skew metric.

The relative size metric measures the change in element size. Let τ be the ratio between the current and initial element volume. The *relative size metric* [20] is then given by $f_{\text{size}} = \min(\tau, 1/\tau)$. Essential properties of the relative size metric are: $f_{\text{size}} = 1$ if and only if the element has the same total area as the initial element and $f_{\text{size}} = 0$ if and only if the element has a total area of zero. The relative size metric can detect elements with a negative total area (degenerate) and elements which change in size due to the mesh deformation.

The skew metric measures the skewness and therefore the distortion of an element. If a node of an element possesses a local negative area, this metric value is set to zero. The expressions for the *skew metric* for triangular, quadrilateral, tetrahedral and quadrilateral elements can be found in [20]. Essential properties of the skew metric are: $f_{\text{skew}} = 1$ if and only if the element has equal angles and $f_{\text{skew}} = 0$ if and only if the element is degenerate.

To measure both the change in element size and the distortion of an element, the *size-skew metric* [20] is introduced which is defined as the weighted product of the relative size and skew metrics: $f_{\text{ss}} = \sqrt{f_{\text{size}}} f_{\text{skew}}$, since changes in element volume have a smaller influence on the mesh quality than element distortion. Essential properties of the quadrilateral size–skew metric are:

- $f_{\text{ss}} = 1 \Longleftrightarrow$ element has equal angles and same size as the initial element.
- $f_{\text{ss}} = 0 \Longleftrightarrow$ element is degenerate.

This is the quality metric we will use to measure the quality of a mesh after deformation.

The average value of the metric over all the elements indicates the average quality of the mesh. The higher the average quality of the mesh, the more stable, accurate and efficient the computation will be. The minimum value of the metric over all the elements indicates the quality of the cell with the lowest quality. This value is required to be larger than zero, otherwise the mesh will contain degenerate cells. Degenerate elements have a very negative influence on the stability and accuracy of numerical computations. In the next section we will use both the average and minimal value of the size–skew metric to compare meshes after mesh movement.

4. Results

The new mesh movement strategy is tested with the 14 radial basis functions introduced in Section 2. First, three simple 2D test problems are performed to investigate the difference in quality of the mesh obtained with the RBF's after movement of the boundary. The tests include mesh movement due to rigid body rotation and translation of a rectangle block and deformation of an airfoil–flap configuration. After that the efficiency of the most promising RBF's is investigated. Furthermore realistic results on a distorted mesh are presented, where flow computations are performed around a NACA-0012 airfoil. Finally it is shown that the method is capable of deforming 3D unstructured hexahedral meshes.

The quality and robustness of the new method depends on the value of the support radius when a radial basis function with compact support is used. When the support radius is chosen large enough, the quality and robustness converge to an optimum. Therefore, a relatively high value, r is 2.5 times the characteristic length of the computational domain, is used in the first three test cases, where we only investigate the accuracy of the different RBF's. The effect of varying the compact support radius r on the computation time is investigated for the most promising RBF's in Section 4.4.

4.1. Test case 1: Rotation and translation

The first test case consists of mesh movement due to severe rotation and translation of a block in a rather small domain. The mesh nodes on the block follow its movement, while the nodes on the outer boundary are fixed. The block has dimension $5D \times 1D$, with D the thickness of the block, and is initially located in the center of a domain which has dimension $25D \times 25D$. The initial mesh is triangular and given in Fig. 5. The block is translated $10D$ down and to the left and is rotated 60° around the center of the block. The mesh deformation is performed in a variable number of steps between the initial and final location, with a minimum of 1 step and a maximum of 15 steps. The less intermediate steps are taken, the larger the deformation between two steps and the harder the total mesh deformation becomes.

The minimum value of f_{ss} after mesh movement with the different RBF's is shown in Fig. 3 for an increasing number of intermediate steps. It can be seen that for all RBF's the minimum value of f_{ss} indeed increases when more intermediate steps are taken. Only for the 5 best performing RBF's 2, 6, 7, 8 and 9 the ranking is given in Table 3. Fig. 4 shows

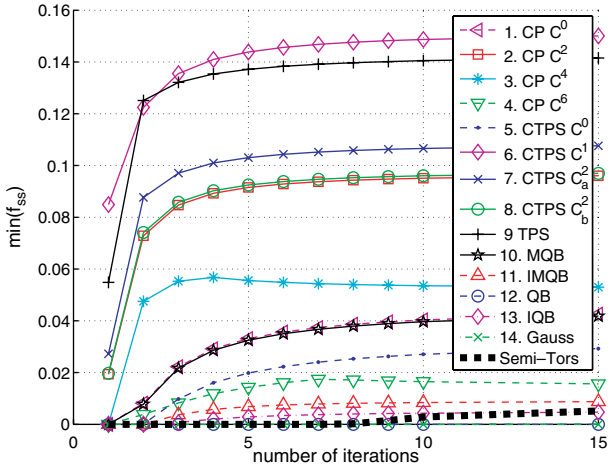

Fig. 3. Quality of the worst cell of the mesh for the different RBF's (test case 1).

Table 3
Ranking for the test cases

Test case 1		Test case 2		Test case 3	
Min	Mean	Min	Mean	Min	Mean
6	9	2, 8	2, 8	2, 7, 8	6, 9
9	7	7	7, 9	6, 9	7
7	6	9	6		2, 8
2, 8	2, 8	6			

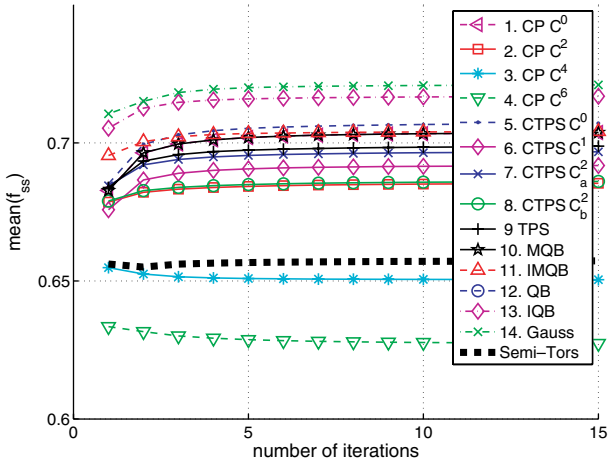

Fig. 4. Average quality of the mesh for the different RBF's (test case 1).

the average value of the mesh quality metric. The Gaussian basis function (no. 14), has the best average quality, however, the minimum of f_{ss} for this function is equal to zero. This results in highly distorted meshes as can be seen in Fig. 6 where the final mesh generated with the Gaussian basis function with 15 intermediate steps is shown. Only cells at a certain distance from the block are heavily deformed, resulting in a high average mesh quality, but the flow solver will probably crash due to the degenerate cells. The mesh with the highest minimum value for the mesh quality is generated with CTPS C^1 (no. 6) and is

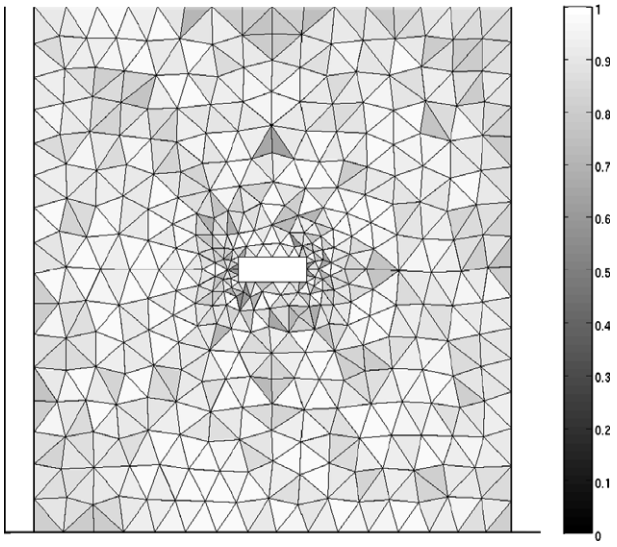

Fig. 5. Initial mesh.

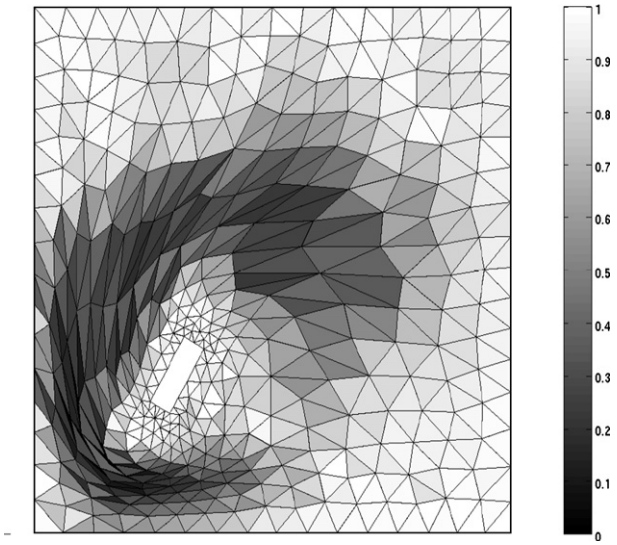

Fig. 6. Final mesh using Gaussian basis function with 15 intermediate steps.

shown in Fig. 7. The average value of the mesh quality metric is lower than for the Gaussian function, because all cells are deformed, but this results in a much smoother mesh. This will have a positive effect on the accuracy, stability and efficiency of an unsteady flow computation. In the next two test cases we will only consider the RBF's 6, 9, 7, 2 and 8, because they are the most promising.

The mesh deformation is also performed with a method based on semi-torsional springs [21], which is an improvement of the popular spring analogy [2]. This method is based on elastic deformation of element edges where element edges are modeled as springs producing forces to propagate boundary displacements into the interior of the mesh. The formation of mis-shaped elements is penalized by incorporating angle information into the spring stiffness. A boundary improvement technique [22] is

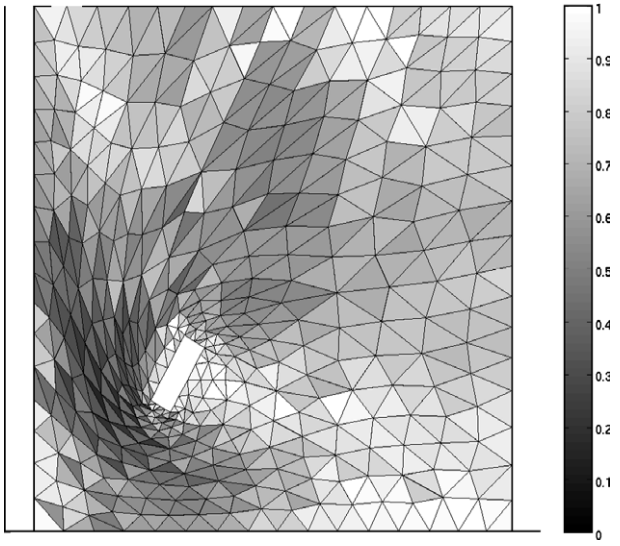

Fig. 7. Final mesh using CTPS C^1 with 15 intermediate steps.

implemented which increases the stiffness of springs close to the moving boundary so that surface displacement spreads further into the mesh. In accordance with [21] we imposed one layer of boundary stiffness by increasing the spring stiffness with a factor 3.5. The results for the minimal and average value for the mesh quality metric are added to Figs. 3 and 4, respectively (bold dashed line). More than 8 intermediate steps are needed to avoid degenerate cells and the minimum value of the mesh quality metric is very low. The final mesh using 15 intermediate steps is shown in Fig. 8. It can be seen that the displacement does not spread very far into the domain and the cells close to the moving block are heavily deformed. The mesh quality obtained with the 5 best RBF's is much higher, because the deformation is more evenly spread through the domain.

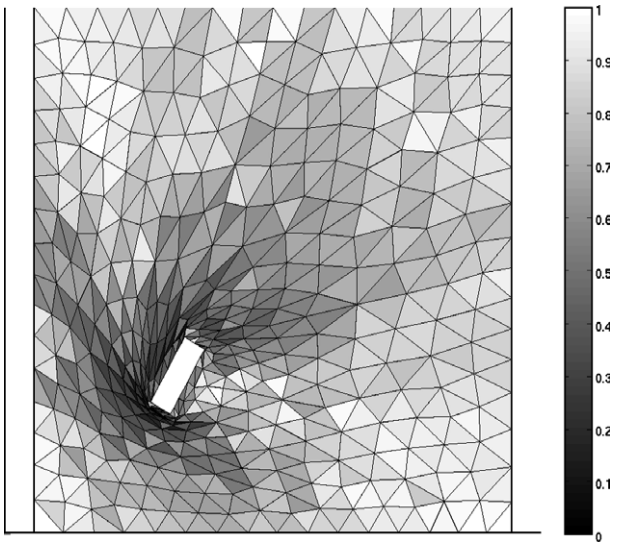

Fig. 8. Final mesh using semi-torsional springs with 15 intermediate steps.

4.2. Test case 2: Rigid body rotation

For rigid body translations the interpolation is exact and therefore the initial mesh is recovered when the domain returns to its initial form. However, it is not guaranteed that this is the case when rotations are present. Therefore we investigate the mesh quality when the block is severely rotated and brought back to its initial position. The nodes on the outer boundary can freely move along this boundary. The initial mesh is the same as for test case 1 (Fig. 5). First the block is rotated 180° counterclockwise, then 360° clockwise and back to the starting position by rotating it again 180° counterclockwise. The rotation is performed with a variable number of intermediate steps. In Fig. 9 again the minimal value and in Fig. 10 the average value of the mesh quality metric is shown against the number of intermediate steps. The resulting ranking for the RBF's is shown in Table 3.

As can be seen from Fig. 9, more than 10 intermediate steps are needed with all functions to obtain a positive

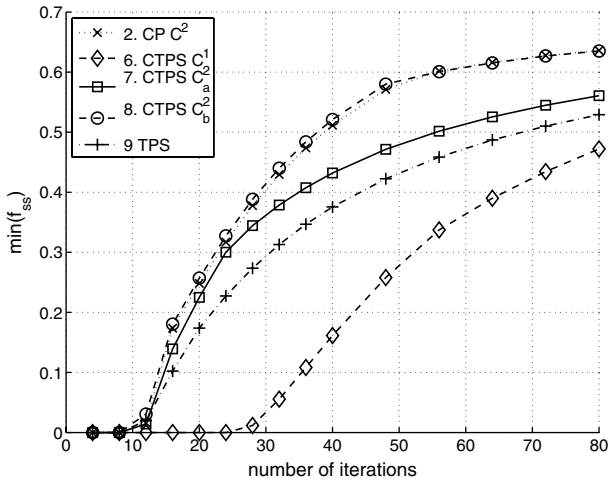

Fig. 9. Quality of the worst cell of the mesh for the different RBF's.

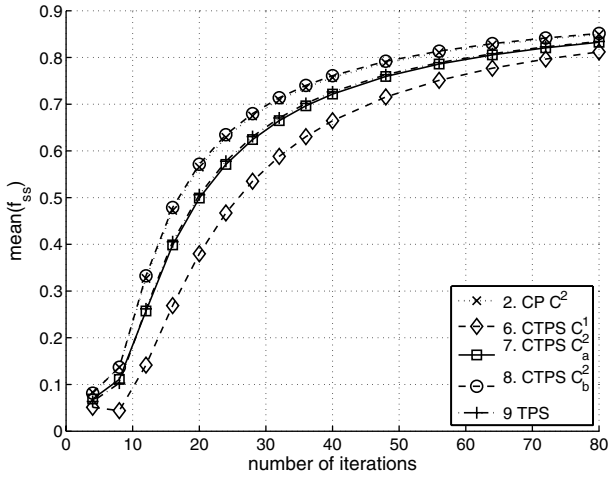

Fig. 10. Average quality of the mesh for the different RBF's.

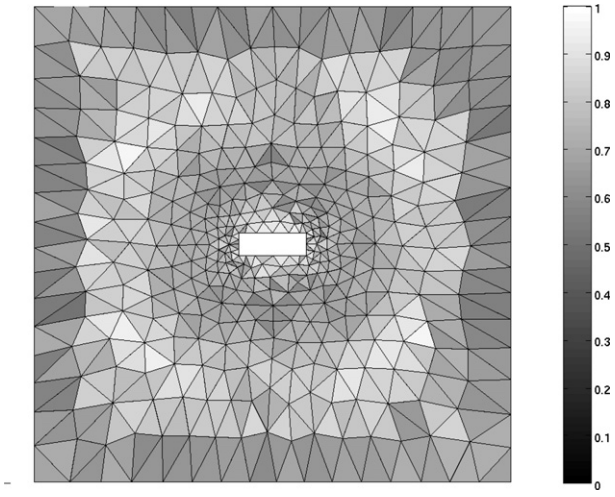

Fig. 11. Final mesh using CTPS C_b^2 after 40 intermediate steps.

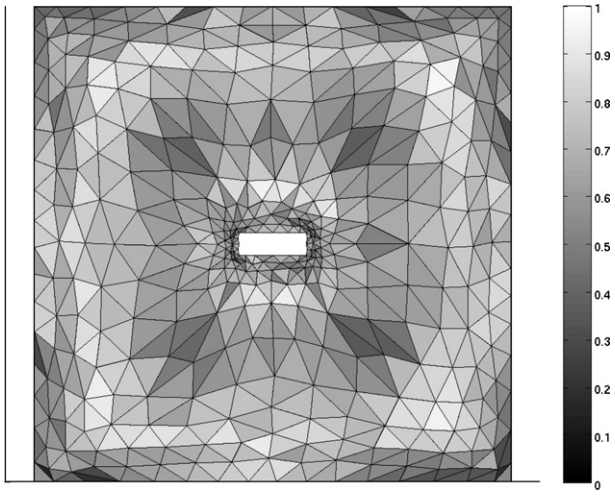

Fig. 12. Final mesh using CTPS C^1 after 40 intermediate steps.

value of f_{ss} for all cells. Figs. 11 and 12 show the final meshes with 40 intermediate steps using the best, CTPS C_b^2 (no. 8) and the worst RBF, CTPS C^1 (no. 6), respectively. Fig. 12 shows that especially the cells close to the block and boundary are deformed. With the CTPS C_b^2 function the mesh is also distorted compared to the initial mesh. However, this distortion is rather small considering the very large rotation of the block. The final meshes obtained with TPS, CP C^2 and CTPS C_a^2 are very similar to that of CTPS C_b^2.

4.3. Test case 3: Airfoil–flap

Until now we only studied rigid body rotation and translation of a block. In the third test case we investigate a more realistic situation of an airfoil–flap configuration. The test case starts with the initial mesh given in Fig. 13. A close up of the mesh around the gap between the airfoil and flap is shown in Fig. 14. The flap is rotated 30° in clockwise direction around an axis a tenth of a cord below

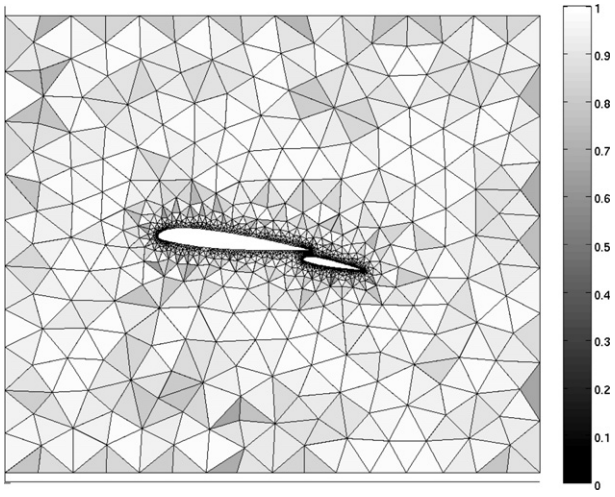

Fig. 13. Initial mesh.

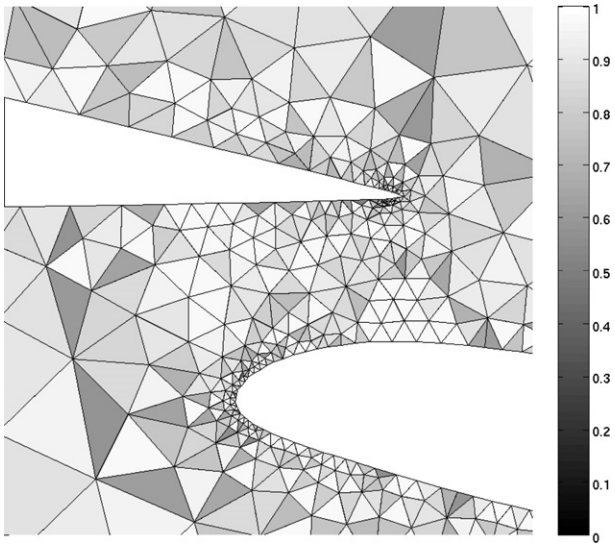

Fig. 14. Close up of initial mesh.

the trailing edge of the airfoil. In Fig. 15 the minimal value and in Fig. 16 the average value of f_{ss} is shown for the remaining RBF's. It can be seen that all the RBF's are able to deform the mesh well, and all functions give meshes of similar quality. The ranking is given in Table 3.

The final configuration of the airfoil–flap and the resulting mesh is shown in Fig. 17 using 10 intermediate steps with the CTPS C_a^2 function. Fig. 18 displays a close up of this mesh and shows that the mesh quality is still very good in the region where the largest deformation takes place and suitable for flow computations. The final meshes obtained with the other functions look very similar.

4.4. Efficiency

The first test case with a rotating and translating block is also used to investigate the efficiency of the five remaining

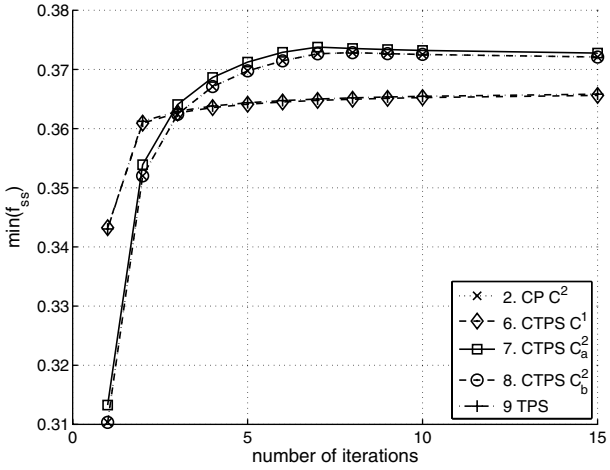

Fig. 15. Quality of the worst cell of the mesh for the different RBF's.

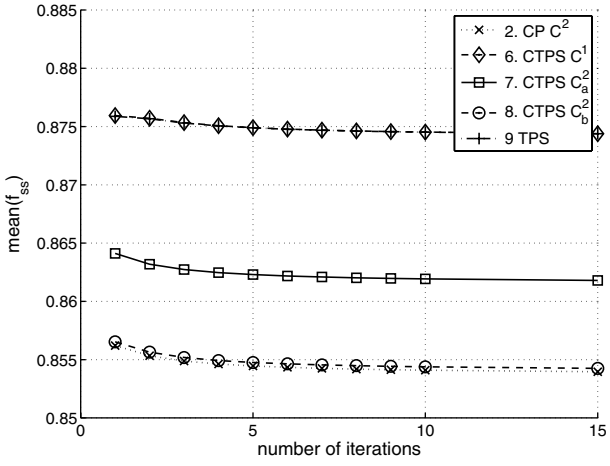

Fig. 16. Average quality of the mesh for the different RBF's.

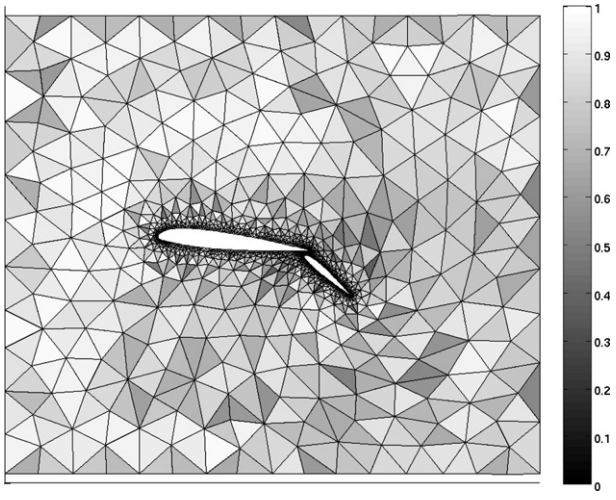

Fig. 17. Final mesh using CTPS C_a^2 with 10 intermediate steps.

RBF's. The number of boundary nodes, the total number of nodes and the support radius is varied to investigate

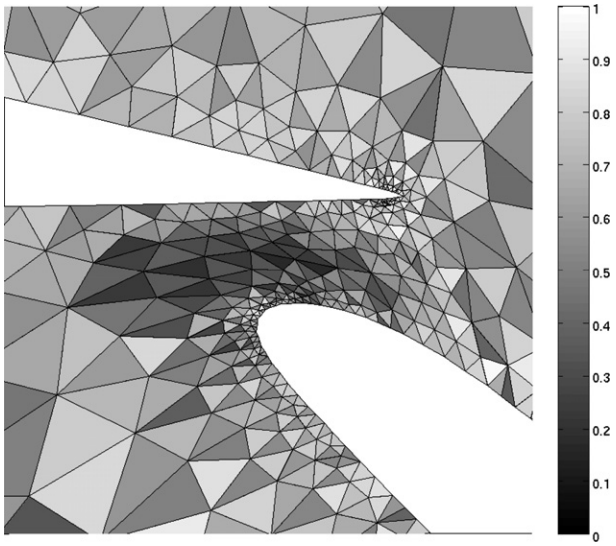

Fig. 18. Zoom of final mesh using CTPS C_a^2 with 10 intermediate steps.

their effect on the computation time needed for the mesh movement with the different RBF's. At this time we are not interested in the most efficient way to implement the new method, but only in the effect of the different RBF's on the computation time. For the comparison the same implementation of the new method is used, only the RBF is changed. The system is solved directly by LU-decomposition. In a later stage this can be improved by implementing a fast iterative method [13,14] or using partition of unity [15].

In Fig. 19 the computation time needed for the mesh movement with an increasing number of structure nodes is shown. The number of nodes in the inner domain of the mesh is kept at a constant value of 100. It can be seen that CP C^2 requires the least computation time, followed by TPS. The functions CTPS C^1, CTPS C_a^2 and CTPS C_b^2 require exactly the same computation time, but more than the other two functions. The difference in computation

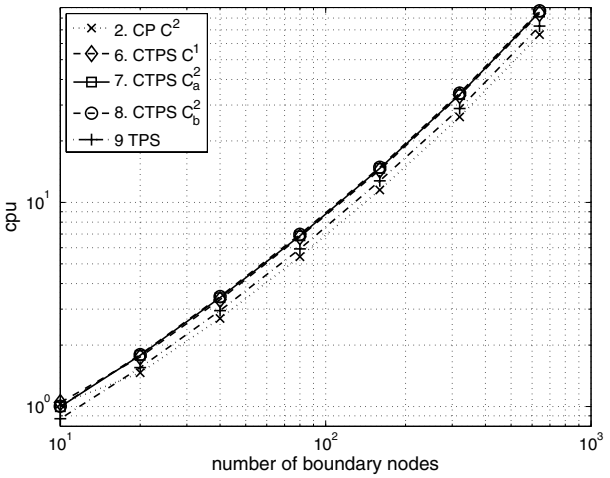

Fig. 19. Influence of the number of boundary nodes on CPU time.

time can be explained by the difficulty of the evaluation of the RBF. CP C^2 only involves the evaluation of a fifth order polynomial, whereas in the other functions an evaluation of a log-function has to be carried out. On top of that, CTPS C^1, CTPS C^2_a and CTPS C^2_b, require the evaluation of a higher order polynomial and are therefore more computational expensive than TPS.

Fig. 20 shows again the computation time needed for the mesh movement but this time the number of internal nodes is varied. The relative results between the RBF's are the same as when only the number of boundary nodes is varied. The difference between the two figures is caused by the fact that the number of boundary nodes determines the size of the system to be solved, whereas increasing the number of internal nodes only results in more function evaluations. Therefore the computation time increases much faster with an increasing number of boundary nodes than with an increasing number of internal nodes.

Finally the effect of the support radius r on the computation time is investigated. For radial basis functions with compact support the matrix system to be solved becomes less dense, when the support radius is decreased. This means that less function evaluations are needed and the system can be solved more efficiently. In Fig. 21 the computation time is plotted against the support radius. Since TPS is a global radial basis function the CPU time does not change with r. The CPU time needed by CP C^2 only mildly increases with r, whereas the computation time needed for mesh movement with the remaining three functions CTPS C^1, CTPS C^2_a and CTPS C^2_b, is very sensitive to the support radius. Only for very small values of the support radius they are approximately as fast as CP C^2. However, for these small values of r the quality of the resulting meshes is very poor as is shown in Fig. 22 where the minimum value of the mesh quality is plotted against the support radius. Especially for the CTPS C^1 function a very high support radius is needed to avoid degenerate cells. Overall

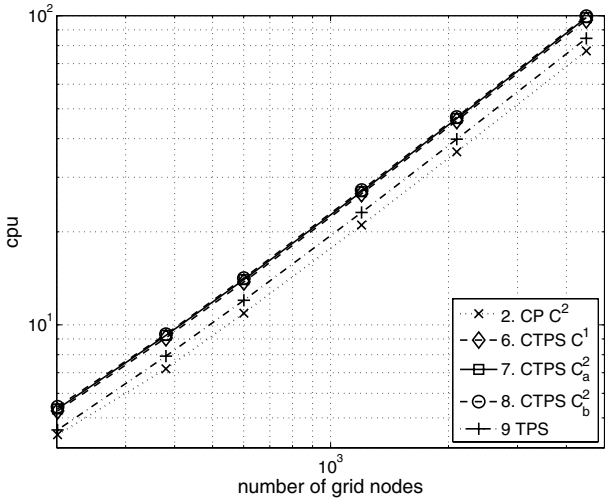

Fig. 20. Influence of the total number of nodes on CPU time.

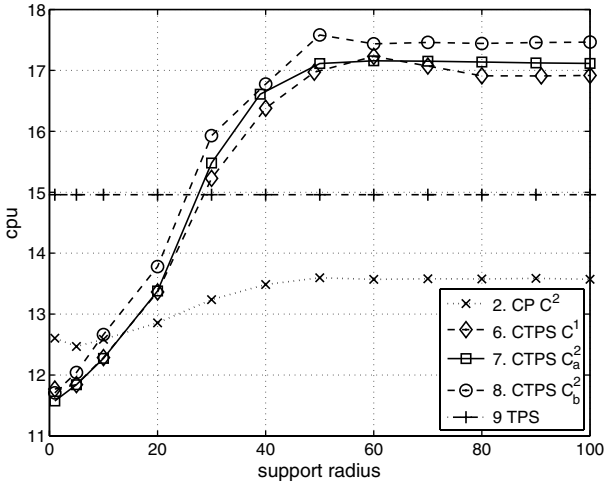

Fig. 21. Influence of support radius on CPU time.

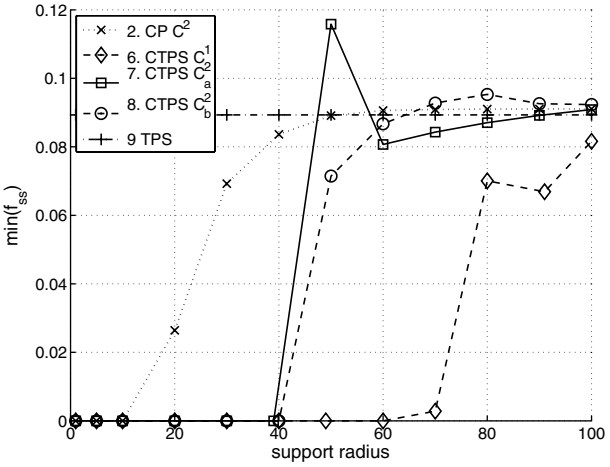

Fig. 22. Influence of support radius on mesh quality.

it can be concluded that CP C^2 requires the least computation time, followed by TPS.

4.5. Flow around airfoil

Until now only the mesh quality of the deformed meshes is investigated without solving a single physical problem. In this section more realistic flow results on a distorted mesh are presented. We perform two calculations of viscid flow around a NACA-0012 airfoil. The airfoil is rotated 8° and moved 5 cords downstream and 2 cords upwards and a steady state solution of Mach 0.3 with $Re = 1000$ is computed around the wing. In the first computation a new unstructured hexahedral mesh is generated around the moved airfoil and a close up of the mesh together with the pressure field is shown in Fig. 23. In the second computation the mesh is deformed in one step with the thin plate spline from its original position. The result is shown in Fig. 24. The resulting pressure distributions over the wing are identical as can be seen in Fig. 25. The difference in lift computed on the two different meshes is only 0.8%.

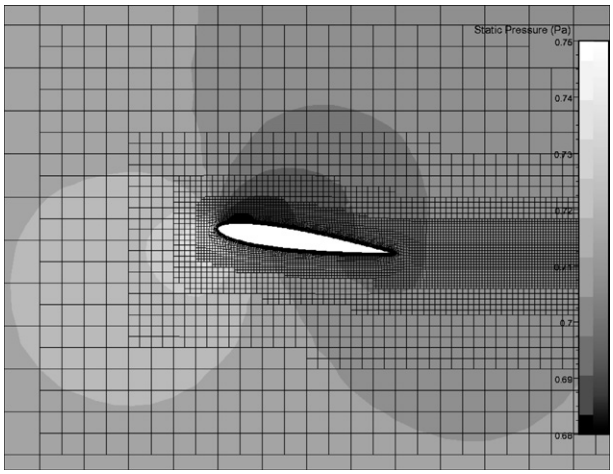

Fig. 23. Pressure field around wing on a new generated mesh.

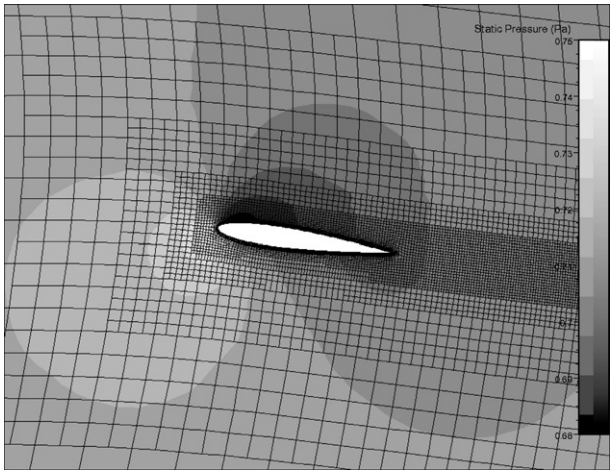

Fig. 24. Pressure field around wing on a deformed mesh.

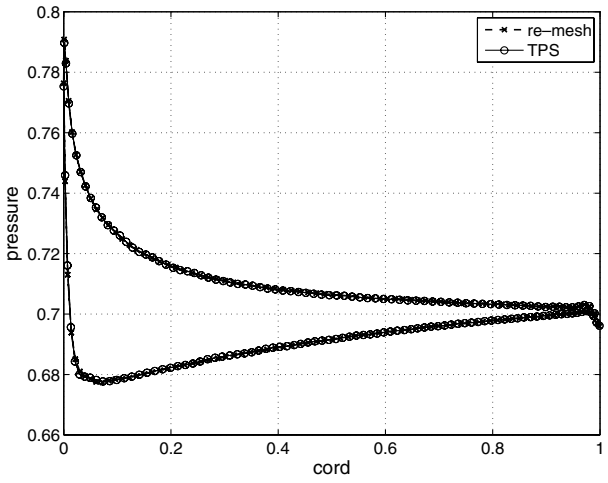

Fig. 25. Pressure distribution over the wing.

4.6. 3D mesh movement

To show the capability of the new method for 3D applications we consider a test case similar to test case 1, only this time in a 3D domain. A cross-section showing the initial mesh and location of the block is given in Fig. 26. The block is translated 2.5 times the thickness of the block and rotated 15° in all three directions. A cross-section of the final mesh and location of the block is shown in Fig. 27. Visually it is quite hard to judge the quality of the mesh and therefore the values of the mesh quality metric in the

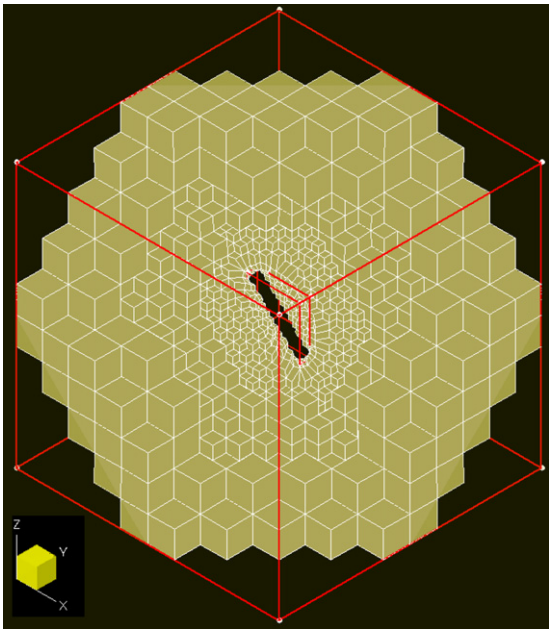

Fig. 26. Cross-section of initial mesh.

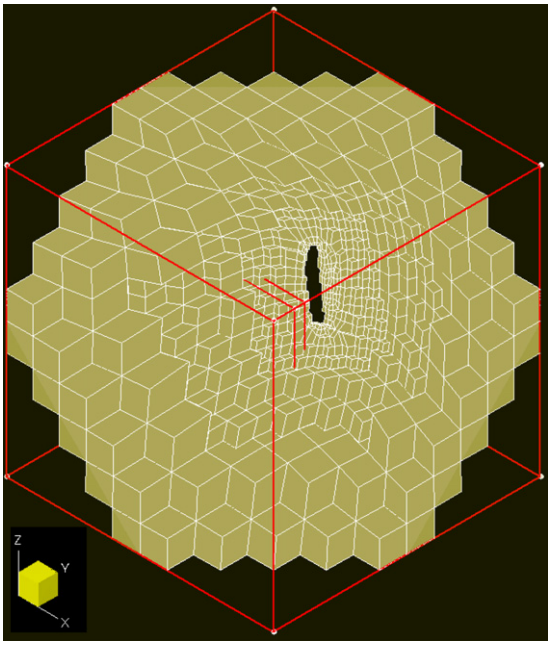

Fig. 27. Cross-section of final mesh.

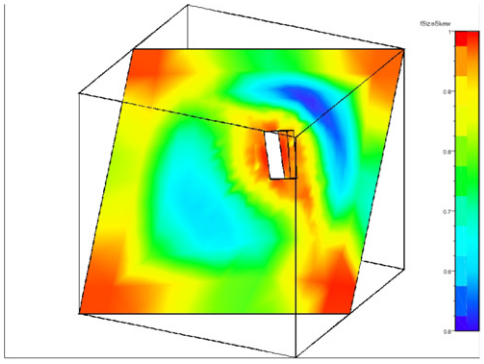

Fig. 28. Mesh quality in cross-section using CP C^2.

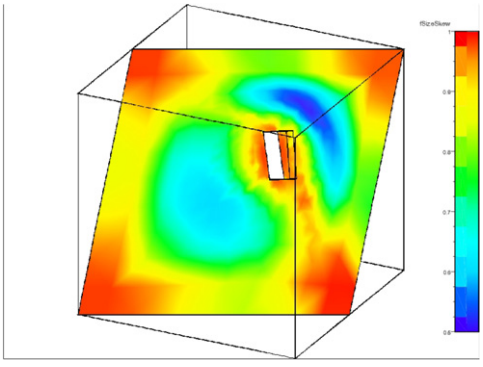

Fig. 29. Mesh quality in cross-section using TPS.

cross-sections is displayed in Figs. 28 and 29 for the CP C^2 and TPS function, respectively. It can be seen that the mesh quality is everywhere larger than 0.5 and therefore the meshes are suitable for computational analysis. The main difference between the two figures is that with CP C^2 the mesh quality close to the moving structure remains a little higher than with TPS.

5. Conclusions

In this paper a point-by-point mesh movement algorithm is developed for the deformation of unstructured grids. Radial basis functions (RBF's) are used to interpolate the displacements of the boundary nodes of the mesh to the inner domain. The method requires solving a small system of equations, only involving the nodes on the boundary of the flow domain. The implementation of the method is relatively simple, even for 3D applications, because no grid-connectivity information is needed. Also the implementation for partitioned meshes, occurring in parallel flow computations, is straightforward.

The new algorithm is tested with 14 RBF's for a variety of problems. The method can handle large deformations of a mesh caused by translation, rotation and deformation of a structure both on 2D and 3D meshes. The performance of the method is not the same for all RBF's. Five RBF's generate meshes of high quality after deformation. However, when efficiency is more important, the CP C^2 RBF with compact support is the best choice, closely followed by the thin plate spline.

In a first comparison the RBF-method produces meshes of higher quality than the popular semi-torsional spring analogy for very large deformations. The quality of the meshes after deformation is high enough to perform accurate flow calculations.

Further research includes a better comparison between the new method and existing methods on accuracy and efficiency and applying it to a real fluid–structure interaction problem.

References

[1] Wang ZJ., Przekwas AJ. Unsteady flow computation using moving grid with mesh enrichment. Tech Rep AIAA-94-0285, 1994.
[2] Batina JT. Unsteady euler algorithm with unstructured dynamic mesh for complex-aircraft aeroelastic analysis. Tech Rep AIAA-89-1189, 1989.
[3] Farhat C, Degrand C, Koobus B, Lesoinne M. Torsional springs for two-dimensional dynamic unstructured fluid meshes. Comput Methods Appl Mech Eng 1998;163:231–45.
[4] Degand C, Farhat C. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. Comput Struct 2002;80:305–16.
[5] Lynch D, ONeill K. Elastic grid deformation for moving boundary problems in two space dimensions. In: Wang S, editor. Finite elements in water resources, 1980.
[6] Helenbrook BT. Mesh deformation using the biharmonic operator. Int J Numer Methods Eng 2003;56:1007–21.
[7] Bathe KJ, Zang H, Ji S. Finite element analysis of fluid flows coupled with structural interactions. Comput Struct 1999;72:1–16.
[8] Bathe KJ, Zang H. Finite element developments for general fluid flows with structural interactions. Int J Numer Methods Eng 2004;60:213–32.
[9] Potsdam MA, Guruswamy GP. A parallel multiblock mesh movement scheme for complex aeroelastic applications. Tech Rep AIAA-2001-0716, 2001.
[10] Beckert A, Wendland H. Multivariate interpolation for fluid–structure-interaction problems using radial basis functions. Aerospace Sci Technol 2001;5(2):125–34.
[11] Smith MJ, Cesnik CES, Hodges DH. Evaluation of some data transfer algorithms for noncontiguous meshes. J Aerospace Eng 2000;13(2):52–8.
[12] Spekreijse S, Prananta B, Kok J. A simple, robust and fast algorithm to compute deformations of multi-block structured grids. Tech Rep, 2002.
[13] Buhmann MD. Radial basis functions. Acta Numer 2000;9:1–38.
[14] Faul AC, Powell MJD. Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions. Adv Comput Math 1999;11:183–92.
[15] Wendland H. Fast evaluation of radial basis functions: methods based on partition of unity. In: Chui CK, Schumaker LL, Stöckler J, editors. Approximation theory X: wavelets, splines, and applications. Vanderbilt University Press; 2002. p. 473–83.
[16] Wendland H. Konstruktion und untersuchung radialer basisfunktionen mit kompaktem träger. Tech Rep, 1996.
[17] Carr JC, Beatson RK, McCallum BC, Fright WR, McLennan TJ, Mitchell TJ. Smooth surface reconstruction from noisy range data. In: First international conference on computer graphics and interactive techniques, 2003.
[18] Wendland H. On the smoothness of positive definite and radial functions. J Comput Appl Math 1999;101:177–88.

[19] Wendland H. Error estimates for interpolation by compactly supported radial basis functions of minimal degree. J Approx Theory 1998;93:258–72.
[20] Knupp PM. Algebraic mesh quality metrics for unstructured initial meshes. Finite Elem Anal Des 2003;39:217–41.
[21] Zeng D, Ethier CR. A semi-torsional spring analogy model for updating unstructured meshes in 3d moving domains. Finite Elem Anal Des 2005;41:1118–11139.
[22] Blom FJ. Considerations on the spring analogy. Int J Numer Methods Fluids 2000;32:647–68.