European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

NEW METHOD FOR MESH MOVING BASED ON RADIAL BASIS FUNCTION INTERPOLATION

Aukje de Boer, Martijn S. van der Schoot and Hester Bijl

Delft University of Technology, Faculty L&R,
Kluyverweg 1, 2629 HS Delft, The Netherlands
e-mail: A.deBoer@tudelft.nl

Key words: Mesh deformation, Unstructured meshes, Radial basis function interpolation, Fluid-structure interaction

Abstract. *A new point-by-point mesh movement algorithm is developed for the deformation of unstructured grids. The method is based on using radial basis functions (RBF's) to interpolate the displacements of the boundary nodes to the whole flow mesh. A small system of equations has to be solved, only involving the nodes on the boundary of the flow domain. Because no grid-connectivity information is needed, this method is very easy to implement, even for 3D applications. There are various RBF's available literature that can be used for the new method. Therefore, the new algorithm is tested with several RBF's for a variety of problems to investigate which RBF produces the best meshes and which one is the most efficient. The method can handle large mesh deformations caused by translations, rotations and deformations of the boudary of the domain. However, the performance depends on the used RBF. The best accuracy and robustness are obtained with the thin plate spline. When efficiency is more important, a C^2 continuous polynomial RBF with compact support is the best choice.*

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[2,3,4] or as solid body elasticity[5]. Special instances of this continuous approach include moving grids based on Laplacian and Biharmonic operators[6]. All the methods based on grid connectivity involve solving a system of equations involving all the flow points and are therefore very 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, occuring in parallel flow computations, is straightforward. However, until now point-by-point schemes are only applied to the boundary nodes of multi-grid blocks[7]. 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[8,9]. 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[7,10]. 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 aplying the method to several severe test cases.

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

displacement, d, can be approximated by a sum of basis functions both on the interface (denoted by subscript in) and in the interior of the mesh (denoted by subscript m)

$$d_i(\mathbf{x}) = \sum_{j=1}^{n_{\text{in}}} \alpha_j \phi(||\mathbf{x} - \mathbf{x}_{\text{in}_j}||) + p(\mathbf{x}) \qquad i = \{\text{in}, \text{m}\}, \tag{1}$$

where $\mathbf{x}_{\text{in}_j} = [x_{\text{in}_j}, y_{\text{in}_j}, z_{\text{in}_j}]$ are the centres in which the values are known, in this case the nodes on the interface, p a polynomial, n_{in} the number of points on the interface 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

$$d_{\text{in}}(\mathbf{x}_{\text{in}_j}) = \mathbf{d}_{\text{in}_j}, \tag{2}$$

with $\mathbf{d}_{\text{in}}$ the vector containing the discrete known values of d at the interface, and the additional requirements

$$\sum_{j=1}^{n_{\text{in}}} \alpha_j q(\mathbf{x}_{\text{in}_j}) = 0, \tag{3}$$

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 $m \leq 2$, a linear polynomial can be used[8]. 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.

Equation (1) can be written in matrix form for the discrete displacement on the interface as follows

$$\begin{bmatrix} \mathbf{d}_{\text{in}} \\ 0 \end{bmatrix} = \begin{bmatrix} M_{\text{in}} & P_{\text{in}} \\ P_{\text{in}}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{4}$$

with $\boldsymbol{\alpha}$ the vector containing the coefficients α_j, $\boldsymbol{\beta}$ the coefficients of the linear polynomial p, M_{in} an $n_{\text{in}} \times n_{\text{in}}$ matrix containing the evaluation of the basisfunction $\phi_{\text{in}_i\text{in}_j} = \phi(||\mathbf{x}_{\text{in}_i} - \mathbf{x}_{\text{in}_j}||)$ and P_{in} an $n_{\text{in}} \times 4$ matrix with row j given by $\begin{bmatrix} 1 & x_{\text{in}_j} & y_{\text{in}_j} & z_{\text{in}_j} \end{bmatrix}$.

For the unknown values in the interior of the flow mesh we can write in a similar way

$$\mathbf{d}_{\text{m}} = \begin{bmatrix} M_{\text{m}} & P_{\text{m}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}. \tag{5}$$

The $n_{\text{m}} \times n_{\text{in}}$ matrix M_{m} contains the evaluation of the basisfunction $\phi_{\text{m}_i\text{in}_j} = \phi(||\mathbf{x}_{\text{m}_i} - \mathbf{x}_{\text{in}_j}||)$ and P_{m} is an $n_{\text{m}} \times 4$ matrix with row j given by $\begin{bmatrix} 1 & x_{\text{m}_j} & y_{\text{m}_j} & z_{\text{m}_j} \end{bmatrix}$. The values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are obtained by solving system (4). This can be done by fast iterative techniques[11]. The size of the system that has to be solved in (4) is equal to $(n_{\text{b}} + 4) \times$

$(n_{\rm b}+4)$ which is usually very small compared to the systems that have to be solved in mesh-connectivity schemes. The systems encounterd there are approximately as large as $n_{\rm in} \times n_{\rm in}$, with $n_{\rm in}$ 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, occuring 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:

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

where $f(x) \geq 0$. 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, only the mesh nodes inside a circle (2D) or sphere (3D) with radius r around a centre x_j are influenced by the movement of this centre. 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, wheras low values of r result in sparse matrix systems which can be solved more efficiently.

nr.	**name**	$f(\xi)$
1	CP C^0	$(1-\xi)^2$
2	CP C^2	$(1-\xi)^4(4\xi+1)$
3	CP C^4	$(1-\xi)^6(\frac{35}{3}\xi^2+6\xi+1)$
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	$1-20\xi^2+80\xi^3-45\xi^4-16\xi^5+60\xi^4\log(\xi)$

Table 1: Radial basis functions with compact support[12].

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[12]. 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 $n \in \{0, 1, 2\}$[12]. There are two possible 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[13] and for data transfer in fluid-structure interaction computations[9]. The MQB and IMQB methods use a parameter a, that controls the shape

nr.	**name**	**abbrev.**	$f(x)$
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}

Table 2: Radial basis functions with global support.

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 $a = 10^{-3}$. 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 introduced 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[14]. The mesh quality metrics are based on a set of Jacobian matrices. These matrices contain information on basic element qualities such as size, orientation, shape and skew. Because we are mainly interested in quadrilateral and hexahedral meshes only the metrics for these elements are introduced. A Jacobian matrix A_k can be determined for each node of the quadrilateral or hexahedral element, where the nodes are numbered according to the right-hand-rule to ensure positive volume elements, see Figures 1 and 2. The columns of the Jacobian matrices are formed from the edge

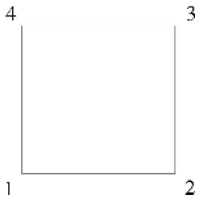

Figure 1: Numbering of a quadrilateral element.

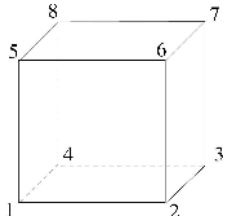

Figure 2: Numbering of a hexahedral element.

vectors emanating from element node k:

$$\begin{array}{cc} \text{Quadrilateral (2D):} & \text{Hexahedral (3D):} \\ A_k = \begin{bmatrix} x_{k+1}-x_k & x_{k+3}-x_k \\ y_{k+1}-y_k & y_{k+3}-y_k \end{bmatrix}, & A_k = \begin{bmatrix} x_{k+1}-x_k & x_{k+3}-x_k & x_{k+4}-x_k \\ y_{k+1}-y_k & y_{k+3}-y_k & y_{k+4}-y_k \\ z_{k+1}-z_k & z_{k+3}-z_k & z_{k+4}-z_k \end{bmatrix}. \end{array} \quad (7)$$

Note that the indices are taken modulo four for quadrilateral elements, and modulo eight for hexahedral elements. The determinant of A_k, denoted by γ_k, determines the size of the element. The entries of the associated symmetric 'metric tensor' $A_k^T A_k$ are denoted by λ_{ij} and contain information about side lengths and the angle between two sides.

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 $\tau = \sum_{k=1}^{2^d} \gamma_k/(2^d w)$ be the ratio between the current and initial element volume, where w is the total volume of the element in the initial mesh and d the dimension ($d = 2$ for quadrilaterals and $d = 3$ for hexahedrals). The *relative size metric*[14] is then given by $f_{size} = \min(\tau, 1/\tau)$. Essential properties of the relative size metric are: $f_{size} = 1$ if and only if the element has the same total area as the initial element and $f_{size} = 0$ 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 ($\gamma_k < 0$ for some k), this metric value is set to zero. The *skew metric*[14] is defined by

$$f_{skew} = \frac{2^d}{\sum_{k=1}^{2^d} \left[\frac{1}{\gamma_k^2} \prod_{i=1}^{d} \lambda_{ii}^k \right]^{\frac{1}{d}}}. \quad (8)$$

Essential properties of the skew metric are: $f_{skew} = 1$ if and only if the element has only right angles and $f_{skew} = 0$ 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*[14] is introduced which is defined as the product of the relative size and skew metrics: $f_{ss} = f_{size} f_{skew}$. Essential properties of the quadrilateral size-skew metric are:

- $f_{ss} = 1 \Leftrightarrow$ element has right angles and same size as the initial element.
- $f_{ss} = 0 \Leftrightarrow$ 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, four 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, deformation, and flutter of a retangle block. After that the efficiency of the most promising RBF's is investigated.

The quality and robustness of the new method depend 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, $r = 100$, is used in the first four 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 4.5.

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 a Cartesian mesh with cells of size $1D \times 1D$, see Figure 5. The block is translated $10D$ down and to the left and is rotated 67.5 degrees 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.

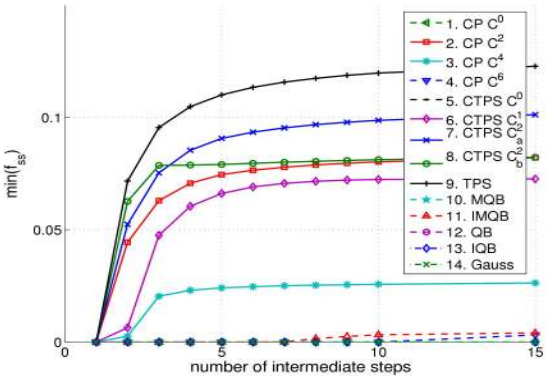

Figure 3: Quality of the worst cell of the mesh for the different RBF's (test case 1).

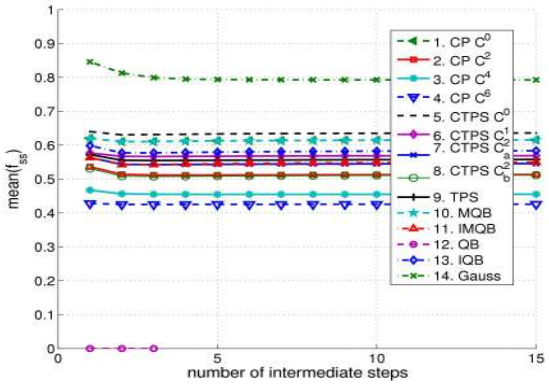

Figure 4: Average quality of the mesh for the different RBF's (test case 1).

The minimum value of f_{ss} after mesh movement with the different RBF's is shown in Figure 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. However, only the RBF's 9, 7, 8, 2, 6 and 3 fulfill the robustness requirement $\min(f_{ss}) > 0$ and therefore only the ranking for these functions is given in Table 3. Figure 4 shows the average value of the mesh quality metric. The Gaussian basis function (nr. 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 Figure 6 where the final mesh generated with the Gaussian basis function with 15 intermediate steps is shown. Only

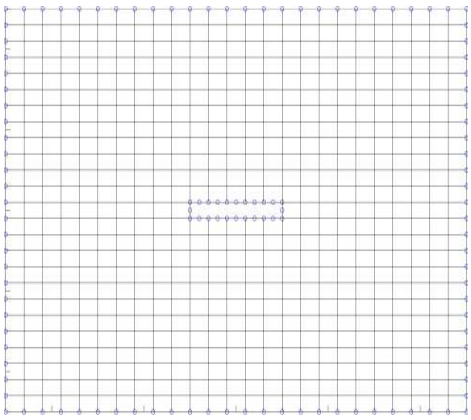

Figure 5: Initial mesh.

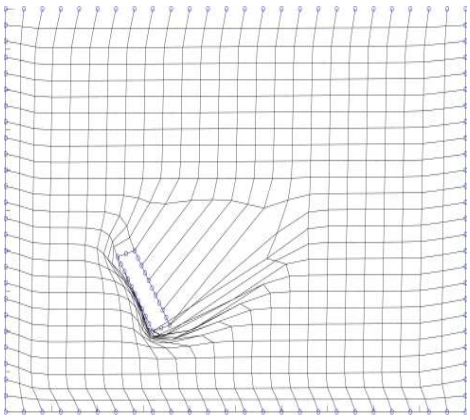

Figure 6: Final mesh using Gaussian basis function with 15 intermediate steps (test case 1).

cells very close to the moving block are heavily deformed, resulting in a high average mesh

quality, but the flow solver will probably crash due to the degenerate cells close to the block. The mesh with the highest quality according to Figure 3 is generated with TPS (nr. 9) and is shown in Figure 7. The average value of the mesh quality metric is lower

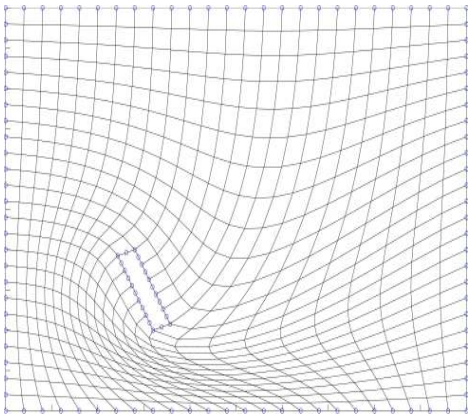

Figure 7: Final mesh using TPS with 15 intermediate steps (test case 1).

$\min(f_{ss})$	$\text{mean}(f_{ss})$
9. TPS	6. CTPS C^1
7. CTPS C_a^2	9. TPS
8. CTPS C_b^2	7. CTPS C_a^2
2. CP C^2	2. CP C^2
6. CTPS C^1	8. CTPS C_b^2
3. CP C^4	3. CP C^4

Table 3: Ranking for test case 1.

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 the unsteady flow computation. In the next two test cases we will only consider the RBF's 9, 7, 8, 2 and 6, because they are the most promising, according to Table 3.

4.2 Test case 2: Rigid body Rotation

It is not guaranteed that the initial mesh is recovered when the domain returns to its initial form. Therefore we investigate the mesh quality when the block is severly 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 (Figure 5). First the block is rotated 180^o counterclockwise, then 360^o clockwise and back to the starting position by rotating it again 180^o counterclockwise. The rotation is performed with a variable number of intermediate steps. In Figure 8 again the minimal value and in Figure 9 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 4. As can be seen from Figure 8, more than 40 intermediate steps are needed with CTPS C^1 to obtain a positive value of f_{ss} for all cells. The robustness of the mesh using CTPS C^1 function is not sufficient enough and this function is therefore excluded for the rest of the test cases. Figures 10 and 11 show the final meshes with 40 intermediate steps using the best, CTPS C_b^2 (nr. 8) and the worst RBF, CTPS C^1 (nr. 6), respectively, to clarify the insufficient properties of CTPS C^1 compared to the other functions. Figure 11 clearly shows the heavily deformed cells close to the block. With the CTPS C_b^2 function the mesh is also distorted compared to the initial mesh. However, this distortion is rather small

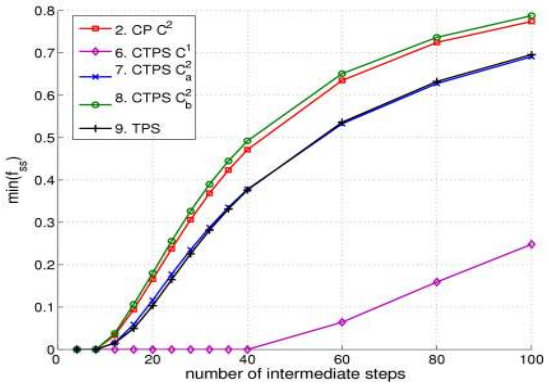

Figure 8: Quality of the worst cell of the mesh for the different RBF's (test case 2).

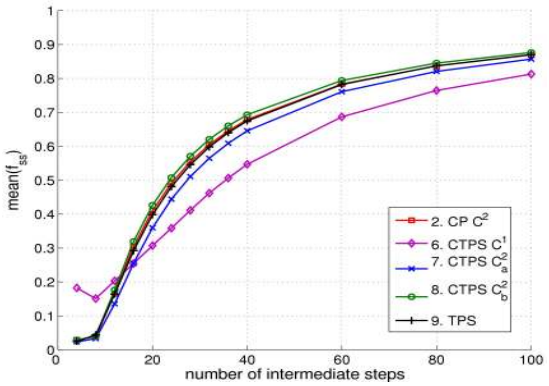

Figure 9: Average quality of the mesh for the different RBF's (test case 2).

$\min(f_{ss})$	$\text{mean}(f_{ss})$
8. CTPS C_b^2	8. CTPS C_b^2
2. CP C^2	2. CP C^2
9. TPS	9. TPS
7. CTPS C_a^2	7. CTPS C_a^2
6. CTPS C^1	6. CTPS C^1

Table 4: Ranking for test case 2.

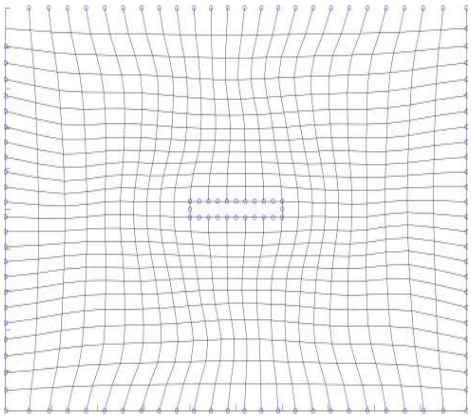

Figure 10: Final mesh using CTPS C_b^2 after 40 intermediate steps (test case 2).

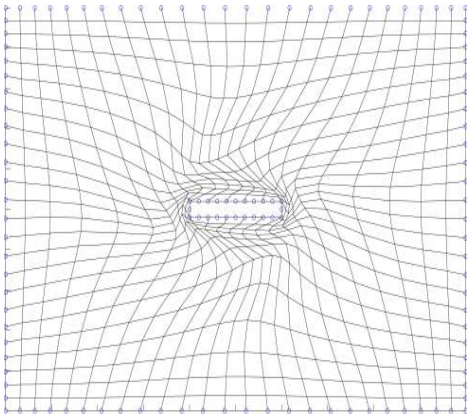

Figure 11: Final mesh using CTPS C^1 after 40 intermediate steps (test case 2).

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: Deformation

Untill now we only studied rigid body rotation and translation of the block. In the third test case we investigate the effect on the mesh quality when the block deforms by bending it in the middle. The test case starts again with the same initial mesh as in test case 1 (Figure 5). We only consider the functions 2, 7, 8 and 9, because they performed best in the previous test cases. In Figure 12 the minimal value and in Figure 13 the

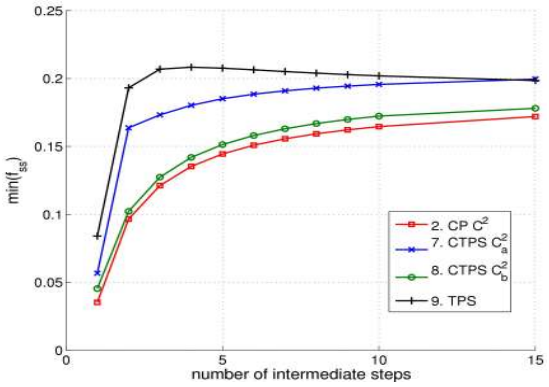

Figure 12: Quality of the worst cell of the mesh for the different RBF's (test case 3).

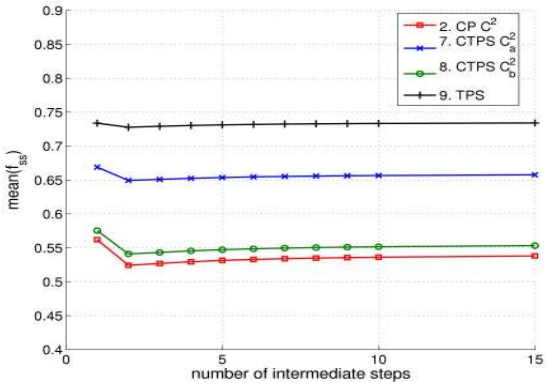

Figure 13: Average quality of the mesh for the different RBF's (test case 3).

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, however there are some differences in the results. The TPS gives the best quality and robustness, closely followed by CTPS C_a^2. The CTPS C_b^2 and CP C^2 functions give approximately the same results, which are little worse than those of the other two functions. The ranking is given in Table 5. Figure 14 displays the

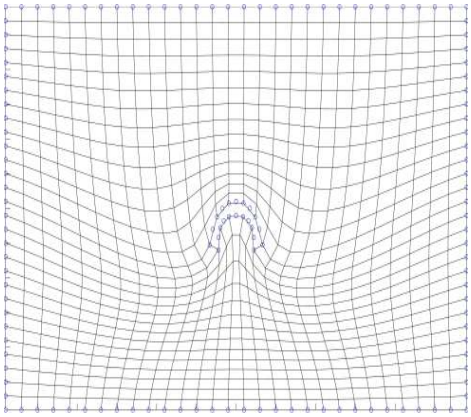

Figure 14: Final mesh using TPS with 5 intermediate steps (test case 3).

$\min(f_{ss})$	$\text{mean}(f_{ss})$
9. TPS	9. TPS
7. CTPS C_a^2	7. CTPS C_a^2
8. CTPS C_b^2	8. CTPS C_b^2
2. CP C^2	2. CP C^2

Table 5: Ranking for test case 3.

final mesh with 5 intermediate steps for the TPS and shows that the mesh quality is very

good. The final meshes of the other functions look very similar.

4.4 Test case 4: Flutter

In this test case the quality of the mesh after repeated movements is investigated. The movement consists of a translational and rotational part. The initial mesh is the same as in test case 1 (Figure 5). The center of the block is moved upwards, with a maximum deflection of $2.5D$. This is a tenth of the distance between the center and the upper boundary. The block also rotates during the movement. A rotation of 67.5 degrees is applied around the center of the block. The location of the block at maximum deflection is shown in Figure 17. After reaching its maximum deflection the block moves back to its initial position. In this test case, 50 oscillations are performed with a varying number of intermediate steps within an oscillation.

Figure 15 shows that all the remaining functions are equally robust. There are some

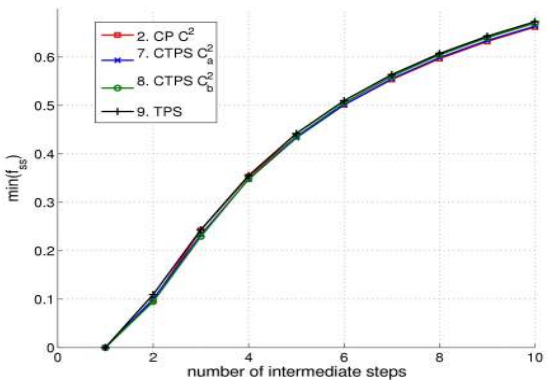

Figure 15: Quality of the worst cell of the mesh for the different RBF's (test case 4).

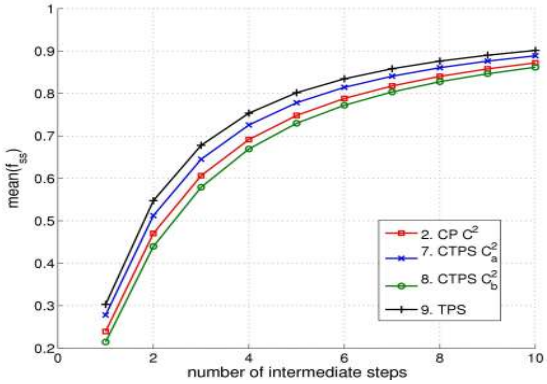

Figure 16: Average quality of the mesh for the different RBF's (test case 4).

$\min(f_{ss})$	$\text{mean}(f_{ss})$
All same value	9. TPS
	7. CTPS C_a^2
	2. CP C^2
	8. CTPS C_b^2

Table 6: Ranking for test case 4.

slight differences in the average value of f_{ss}, shown in Figure 16. Between the best and worst results there is a difference of 10%-25% and the ranking is given in Table 4.4.

Figure 18 displays the final mesh using only 1 step within an oscillation to reach the maximum deflection obtained with TPS. The meshes obtained with the other functions are almost the same. All the functions have difficulties with this test case when only 1

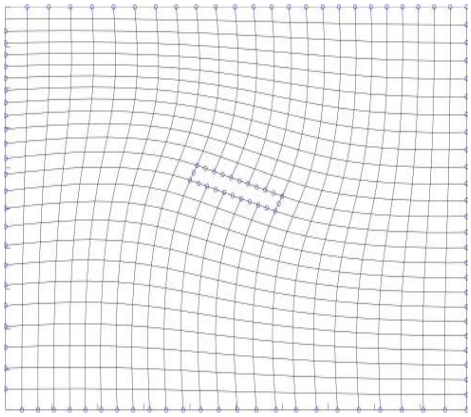

Figure 17: Block at maximum deflection (test case 4).

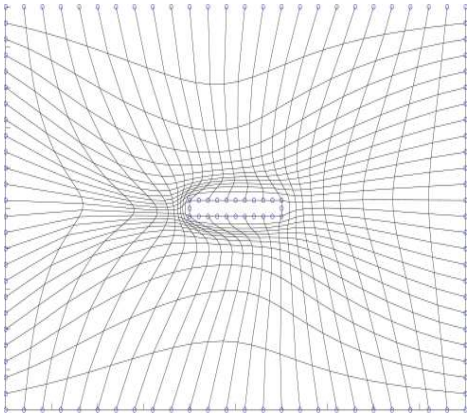

Figure 18: Final mesh using TPS with 1 step to reach maximum deflection (test case 4).

step is used. When the number of steps is increased to 10, the obtained quality of the meshes is much better.

The overall conclusion is that TPS gives the best accuracy and robustness closely followed by CP C^2, CTPS C_a^2 and CTPS C_b^2 which all produce meshes of similar accuracy and robustness.

4.5 Efficiency

The first test case with a rotating and translating block is also used to investigate the efficiency of the remaining RBF's. The number of boundary nodes, the total number of nodes and the support radius is varied to investigate 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[11].

In Figure 19 the computation time needed for the mesh movement with an increasing number of structure nodes is shown for the remaining RBF's. 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_a^2 and CTPS C_b^2 require exactly the same computation time, but more than the other two functions. The difference in computation 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, wheras in TPS, CTPS C_a^2 and CTPS C_b^2 an evaluation of a log-function has to be carried out. On top of that, CTPS C_a^2 and CTPS C_b^2, require the evaluation of a fifth order polynomial and are therefore more computational expensive than TPS.

Figure 20 shows again the computation time needed for the mesh movement but this

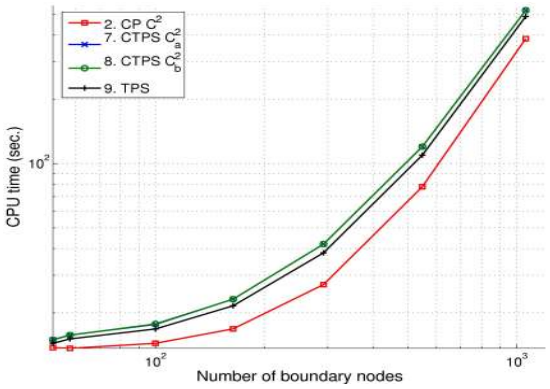

Figure 19: Influence of the number of boundary nodes on CPU time.

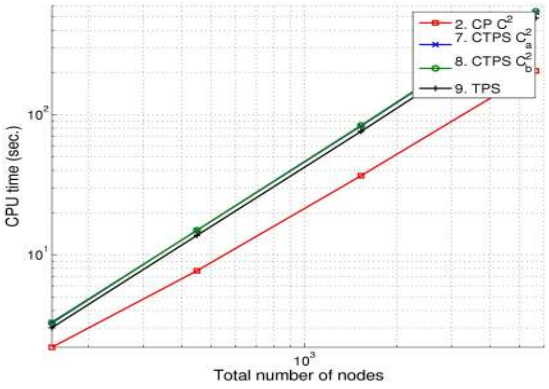

Figure 20: Influence of the total number of nodes on CPU time.

time the total number of nodes is varied. Here both the number of nodes on the outer boundary and the number of nodes in the inner domain is increased. The relative results between the RBF's are the same as when only the number of boundary nodes is varied. Again CTPS C^2_a and CTPS C^2_b require exactly the same computation time. 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, wheras 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 the system can be solved more efficiently. In Figure 21 the computation time is plotted against the support radius. The CPU-time needed by TPS does not change with r, because this is a global radial

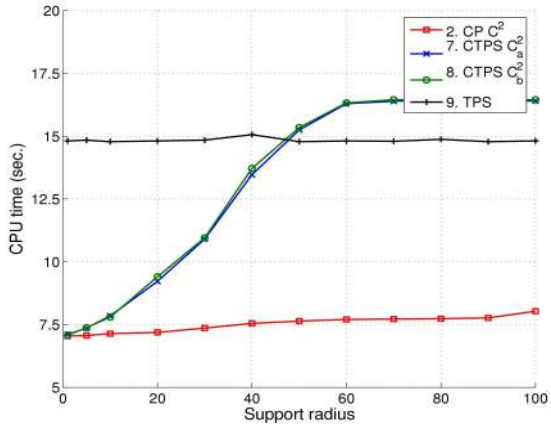

Figure 21: Influence of support radius on CPU time.

basis function. The CPU time needed by CP C^2 only mildly increases with r, whereas the computation time needed for mesh movement with the remaining two functions CTPS C_a^2 and CTPS C_b^2, 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. Overall it can be concluded that CP C^2 requires the least computation time.

5 CONCLUSIONS

In this paper a new 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, occuring in parallel flow computations, is straightforward.

The new algorithm is tested with fourteen RBF's for a variety of problems. The method can handle large deformations of a mesh caused by translating, rotating and deforming a solid block. The performance of the method is not the same for all RBF's. The thin plate spline generates the most accurate and robust meshes after deformation. However, when efficiency is more important, the CP C^2 RBF with compact support is the best choice.

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

REFERENCES

[1] Z. J. Wang and A. J. Przekwas, Unsteady flow computation using moving grid with mesh enrichment, AIAA Paper 94-0285, 1994.

[2] J. T. Batina, Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aeroelastic analysis, Tech. Rep. AIAA-89-1189, 1989.

[3] C. Farhat, C. Degrand, B. Koobus and M. Lesoinne, Torsional springs for two-dimensional dynamic unstructured fluid meshes, *Computer Methods in Applied Mechanics and Engineering*, **163**, 231–245, 1998.

[4] C. Degand and C. Farhat, A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes, *Computers and Structures*, **80**, 305–316, 2002.

[5] D. Lynch, K. ONeill, Elastic grid deformation for moving boundary problems in two space dimensions, in S. Wang (ed.), *Finite elements in water resources*, 1980.

[6] B. T. Helenbrook, Mesh deformation using the biharmonic operator, *International journal for numerical methods in engineering*, **56**, 1007–1021, 2003.

[7] M. A. Potsdam, G. P. Guruswamy, A parallel multiblock mesh movement scheme for complex aeroelastic applications, Tech. Rep. AIAA-2001-0716, 2001.

[8] A. Beckert and H. Wendland, Multivariate interpolation for fluid-structure-interaction problems using radial basis functions, *Aerospace Science and Technology*, **0**, 1–11, 2001.

[9] M. J. Smith, C. E. S. Cesnik, D. H. Hodges, Evaluation of some data transfer algorithms for noncontiguous meshes, *Journal of Aerospace Engineering*, **13** (2), 52–58, 2000.

[10] S. P. Spekreijse and B. B. Prananta and J. C. Kok, A simple, robust and fast algorithm to compute deformations of multi-block structured grids, 8th International Conference on Numerical Grid Generation in Computational Field Simulations, Honolulu, Hawaii, USA, June 3-6, 2002.

[11] A. C. Faul and M. J. D. Powell, Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions, *Advances in Computational Mathematics*, **11**, 183–192, 1999.

[12] H. Wendland, Konstruktion und Untersuchung radialer Basisfunktionen mit kompaktem Träger, tech. report, Georg-August-Universität, Göttingen, 1996.

[13] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan and T. J. Mitchell, Smooth surface reconstruction from noisy range data, First international conference on Computer graphics and interactive techniques, Melbourne, Australia, 2003.

[14] D. A. Field, Quality measures for initial meshes, *International journal for numerical methods in engineering*, **47**, 887–906, 2000.