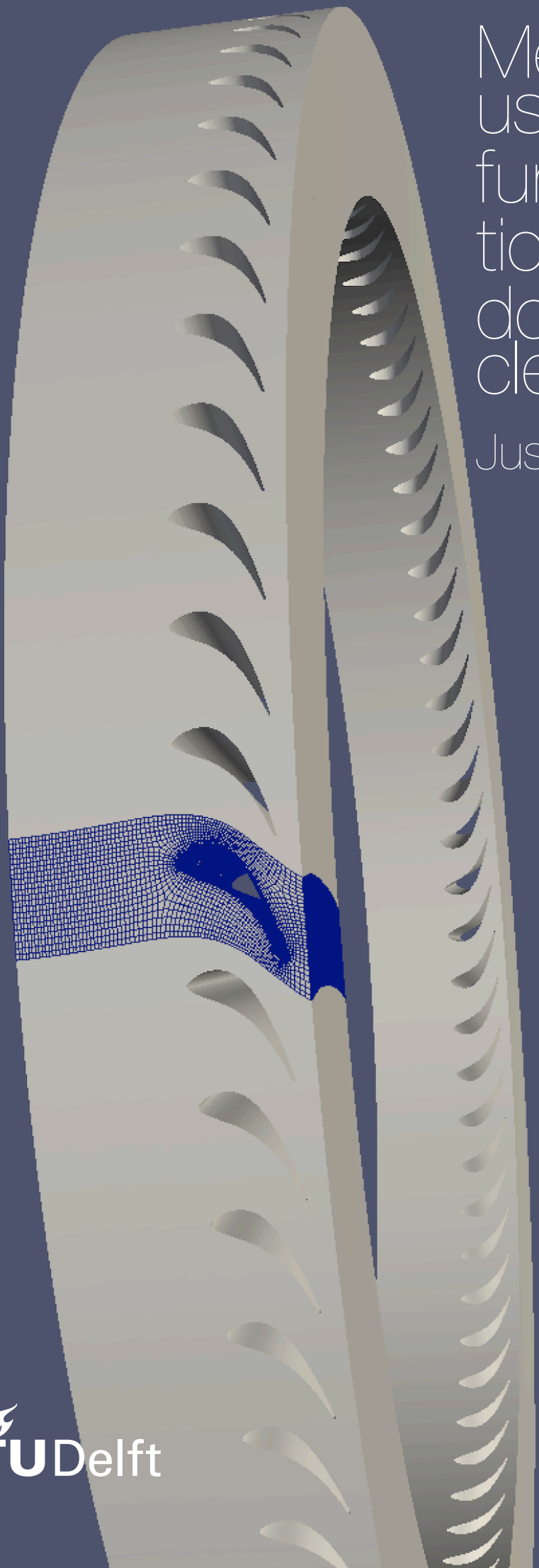


Mesh deformation using radial basis function interpolation on periodic domains with small clearance gaps

Justine De Keyser

Technische Universiteit Delft

TU Delft

Mesh deformation using radial basis function interpolation on periodic domains with small clearance gaps

by

Justine De Keyser

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, September 29, 2021 at 14:00.

Student number: 4288432
Project duration: December 14, 2020 – September 29, 2021
Thesis Committee:

Dr. Ir. B. van Oudheusden,	TU Delft, chair
Dr. Ir. A. van Zuijlen,	TU Delft, supervisor
Dr. Ir. M. Pini,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

$2D$	Two-dimensional
$3D$	Three-dimensional
CFD	Computational Fluid Dynamics
CS	Compactly Supported
$CTPS$	Compactly Supported Thin Plate Spline
FSI	Fluid-Structure Interaction
IDW	Inverse Distance Weighting
$IMQB$	Inverse Multiquadric Biharmonics
IQB	Inverse Quadric Biharmonics
MQB	Multiquadric Biharmonics
PDE	Partial Differential Equation
QB	Quadric Biharmonics
RBF	Radial Basis Function
TFI	Transfinite Interpolation
TPS	Thin Plate Spline
$TU\ Delft$	Delft University of Technology

Symbols

α_k	Determinant of the Jacobian
β	Boundary correction factor
δ	Distance function
ϵ	Error
λ	Periodic distance
$\lambda_{i,j}^{k}$	Components of metric tensor
ω	Equal displacement method variable
Φ	RBF interpolation matrix
τ	Cell volume ratio
θ_{per}	Periodic angle [rad]
$\vec{\alpha}$	Interpolation coefficient vector
$\vec{\beta}$	Polynomial coefficient vector

$\vec{d}$	Displacement vector
$\vec{n}$	Normal vector
$\vec{t}$	Tangent vector
$\vec{v}$	Vector
$\vec{x}$	Node position vector
$\vec{x}_b$	Boundary node position vector
a	Shape parameter
A_k	Jacobian matrix
$f_{size-skew}$	Size-skew mesh quality metric
f_{size}	Size mesh quality metric
f_{skew}	Skew mesh quality metric
N	Number of nodes
N_b	Number of boundary nodes
N_c	Number of control nodes
N_i	Number of internal nodes
N_m	Number of moving nodes
N_p	Number of periodic nodes
nn	Nearest neighbour
p	Polynomial
P_b	Polynomial matrix
q	Polynomial
r, θ, z	Cylindrical coordinates
r	Support radius
s	Interpolation function
x, y, z	Cartesian coordinates

Abstract

In this thesis report, a new radial basis function (RBF) interpolation mesh deformation method is investigated that is applicable on periodic domains that contain small clearance gaps. In the field of fluid structure interaction, there is a strong interaction between a structure and the fluid in which this structure is immersed. When performing simulations of this interaction, the fluid mesh has to adapt to the changing domain which occurs due to the structure deforming. The adaption of the fluid mesh can be done with an RBF interpolation method. Although this is considered to be one of the most robust methods available, there are specific cases in which the method can still yield some poor quality deformed meshes. This is for example the case when small clearance gaps are present such as the region between a blade and the shroud in turbomachinery. Then it becomes beneficial to allow boundary nodes to slide to improve the mesh quality.

In turbomachinery, periodic domains are also often encountered, for example in a gas turbine. Here, the same blade is constantly repeated, allowing to use one single blade in computations which will have a domain with periodic boundaries. In general, the larger the number of blades, the smaller the distance between the blade and the periodic mesh boundary will be. Hence, also here small clearance gaps are present. As the blade might undergo a large deformation, it is possible for the blade to even intersect the mesh boundary if this boundary is kept fixed, which would lead to a degenerate mesh.

The aim of this thesis is to avoid this scenario by making adaptions to the RBF interpolation method that allow the periodic boundaries of the mesh to be displaced with the deforming blade. Evidently, this has to be done in a periodic manner, keeping both periodic boundaries equal. This will yield a higher mesh quality of the deformed mesh as it will reduce the skewness and avoid the occurrence of degenerate cells. Two different methods for periodic boundary displacement are investigated in this report. Both methods work for both translational and rotational periodic boundaries and several two-dimensional (2D) and three-dimensional (3D) meshes are used as test cases.

The first method is the periodic distance method which is based on making the distance between the boundary nodes periodic. This method is very similar to the regular RBF method and only requires a regular Euclidean distance function to be replaced by a periodic distance function. This means that the computational cost is also similar with an interpolation matrix of size $N_m \times N_m$. This method shows good results for both 2D and 3D meshes and both types of periodic boundaries. Apart from giving the periodic boundaries an identical displacement, it also allows for a smooth transition from one mesh to another.

The second method is based on equal displacement. Hence, the RBF interpolation matrix is adapted to include conditions imposing equal displacement of the periodic boundary nodes. These extra conditions yield a larger interpolation matrix of size $(N_m + N_p) \times (N_m + N_p)$. This means the computational cost for this method is higher than for the periodic distance or the regular RBF method. This method also shows good results in terms of deformed mesh quality but does not feature the same smoothness over the interface between two meshes and creates small kinks at the interface.

Even though RBF interpolation can be relatively low in computational cost compared to other mesh deformation methods that require mesh connectivity information, it still becomes computationally expensive for large 3D problems with a high number of nodes. Therefore, a greedy algorithm is also used for larger problems which allows for a reduced amount of nodes selected to perform the interpolation.

In conclusion, the periodic displacement method used with RBF interpolation shows great potential for different types of problem sizes and periodic boundaries and can potentially make the simulation process of periodic problems much more efficient.

Preface

This report is the result of my master thesis project. The thesis project was undertaken as part of the Master of Science degree at the Aerospace Faculty of Delft University of Technology.

I would like to express my gratitude to my thesis supervisor, Alexander van Zuijlen, for his guidance and insights and patience with helping me get through the complex 3D code.
Furthermore, I'd also like to thank my friends and family for their support, not just for this thesis but throughout my full study time here in Delft.

Justine De Keyser
Delft, August 2021

1 Introduction

Fluid-structure interaction (FSI) is an interdisciplinary field that studies the interaction between a deformable structure and a fluid flow. There are countless natural and engineering applications where this interaction plays an important role. Some examples are dam-water systems [25], blood flow within arterial walls [11] and aircraft [10]. The study of these interactions is important to assess the performance of the structure once it is deformed and can also be used to predict the instability of the coupled systems. However, these interaction problems are often too complex for analytical solving, hence the importance of numerical simulations.

In the past couple of decades, computers have significantly improved in computational power, making it possible to carry out more advanced numerical computations needed for the FSI simulations. These simulations are important for shape optimization applications where typically the geometry is optimized by minimizing an objective function, such as for adjoint-based optimization methods [8]. Other commonly used applications include aeroelastic phenomena computations such as flutter [41]. The simulations combine both structural analysis and computational fluid dynamics (CFD). The fluid exerts forces on the structure, causing it to deform. This displacement of the structural boundary then leads to the fluid domain having to deform to match the deformed structural geometry, implying a deformation of the fluid mesh. The mesh can be adapted by either regenerating it, which can be very computationally expensive, or by using a mesh deformation method. When using a mesh deformation method, an appropriate choice should be made for the specific method such that it is robust and provides good quality meshes even for large displacements [31]. Another important criterion is that the computational cost is not too extensive. This could be accomplished for large problems through the use of efficiency optimization techniques.

1.1. Mesh deformation methods

In computational fluid dynamics (CFD), meshing is used where the fluid volume or domain is broken down into smaller cells, forming a two-dimensional (2D) or three-dimensional (3D) grid. During a simulation, the fluid equations are solved for each of these cells. The quality of the mesh is important for the convergence and the speed of the simulation. It also influences the accuracy and reliability of the simulation result. As the flow domain in changing, the fluid mesh has to be adapted to conform the new domain. Regenerating the mesh is a computationally expensive procedure and mapping the solution to the new mesh can introduce errors. Therefore, a method is required to dynamically deform the mesh. Several mesh deformation methods have already been presented in literature of which some are more robust and others are more efficient, depending on what is desired for the specific simulation. There are a couple of criteria that a mesh deformation method should ideally perform well for, listed here in descending order of importance:

- **Robustness:** The method is robust if the yielded mesh quality is still high after it has undergone multiple deformations and no degenerate cells are present after large deformations. This criterion is important as degenerate cells will cause errors when running the simulation and stop the simulation from continuing.

- **Deformation range:** The method should be able to handle both small and large deformations so it can be applicable for several different cases.
- **Computational cost:** The method should be computationally efficient as the simulation might consist of a large number of time steps where the method has to be performed every time. The size of the mesh can also become very large for 3D applications and therefore the method should still be efficient to keep the computational cost of the simulation reasonable even for increased mesh sizes. This efficiency is also influenced by the code its ability to be parallelized as this allows for accelerated computations with respect to performing serial computations.
- **Implementation complexity:** The method should preferably be simple to implement, making it easier for adaptions to be made to the code at a later stage and for errors to be located within the code.

Generally, it is more important to have a robust method that might be complex to implement opposed to a simple method that cannot handle large deformations and that will produce poor mesh quality.

The deformation methods can be divided in two main groups: physical analogy based and interpolation based methods. Physical analogy based methods use a physical process to describe the deformation and use the connectivity information of the mesh. These methods involve solving a system of equations that include the complete mesh and therefore result in a higher computational cost. In unstructured meshes, hanging nodes can occur when using these methods which results in extra computational cost to remove them. The need for connectivity information also makes it more complex for parallelization to be applied. The most commonly used physical analogy method is the spring analogy method [2]. Here, the mesh lines are replaced by linear springs with an inversely proportional stiffness to its length. When the springs are in equilibrium, the force in the node will be zero and this way, the displacement of the node is found. Although this method is easy to implement, it is less suitable for larger deformations and might create degenerate cells as a node can possibly cross an edge. Therefore, several adaptions of the linear spring analogy method have been created to improve some of its flaws such as the ball-vertex spring [9], the torsional spring [19], the semi-torsional spring [4, 39] and the ortho-semi-torsional spring method [14]. The latter being the most advanced method and scoring the highest for the different deformation method criteria.

Another physical analogy method is the linear elasticity method [3, 5, 17, 23, 24]. In this method, the displacement of the node is computed by modeling the mesh as an elastic body and solving linear elasticity equations. This allows for larger deformations but comes at an increased computational cost compared to the spring analogy. Furthermore there are also the Laplacian smoothing method and the biharmonic operator [15]. The Laplacian is more suitable for smaller deformations and has a computational cost similar to the spring analogy method. A disadvantage is that there is no mechanism preventing degenerate cells from forming. The biharmonic operator allows for larger deformations to be handled as it used a fourth order partial differential equation (PDE) which allows for an extra boundary condition to be applied compared to the Laplacian. This allows to have both control of the boundary node coordinates and the normal mesh spacing. This also means the computational cost is two (modified Laplacian smoothing) to four times higher.

Interpolation analogy based methods are point-by-point methods that do not need mesh connectivity information. In general, they will be more computationally efficient than physical analogy methods as they will use a smaller set of control nodes (N_c) instead of all mesh nodes to compute the displacements. They can also handle meshes with hanging nodes better. However, there is also the introduction of interpolation errors. The lack of required connectivity information makes these methods more suitable for parallelization. Transfinite interpolation (TFI) is a very common computationally efficient and simple to implement method for structured meshes [42]. Another computationally efficient method is algebraic damping which can also be used for unstructured meshes [40]. Furthermore a couple of very robust methods are the Delaunay graph [27], inverse distance weighting (IDW) [38] and radial basis function (RBF) interpolation [7]. The Delauney graph method is more efficient than the spring analogy but is also better used with smaller deformation cases since large deformations can easily result in poor meshes requiring regeneration of the graph and an increase in computational cost.

Between the IDW and RBF interpolation methods, the IDW is more computationally efficient as it is explicit and does not require a system of equations to be solved as for RBF interpolation. However, RBF interpolation can be combined with a data reduction method which can reduce the computational cost for larger problems [28]. For both methods, adaptions are available to allow for sliding in cases with small clearance gaps present.

From all the aforementioned methods, the adapted IDW and the adapted RBF interpolation score the highest for the different criteria. Although no comparison has been made between these two adapted methods, it is possible that the RBF interpolation with sliding [22] will yield a higher mesh quality than the improved IDW method [33] since it has a smoother interpolation function. Therefore, the adapted RBF interpolation method is chosen for this project and the thesis work by Mathew [22] is used as the base to develop further for periodic domains.

1.2. Periodic domains and sliding method

In some specific cases, mesh deformation might become more tricky and it can be difficult to maintain a high mesh quality. These cases are illustrated using a turbomachinery application. In the field of turbomachinery, regions are often encountered where small clearance gaps are present. Figure 1.1 shows such a small clearance gap that is located between the blade tip and the shroud of a turbine or pump.

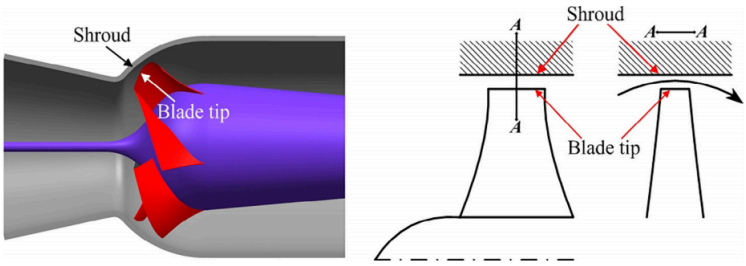


Figure 1.1: Small clearance gap between blade tip and shroud [21].

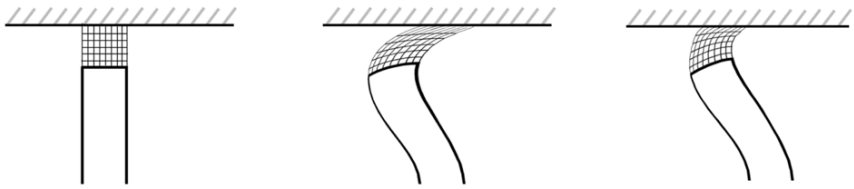
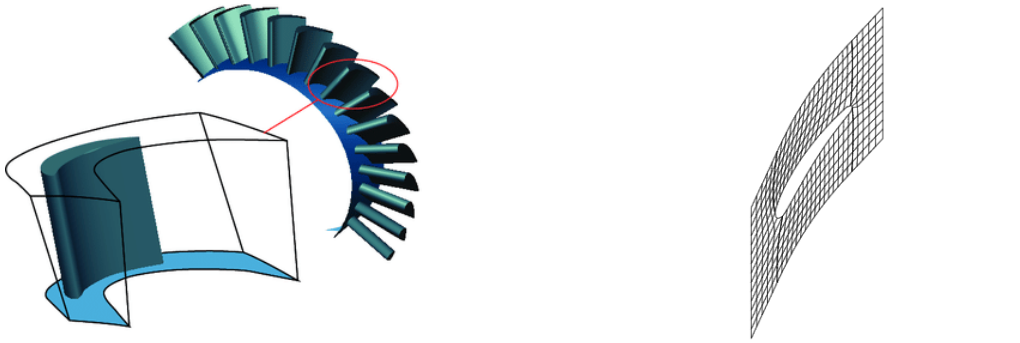


Figure 1.2: Mesh in small clearance gap between blade and shroud, before deformation, after deformation and after sliding [22].

When the blade experiences a relatively large deformation, the computational mesh between the blade and the shroud might become very distorted, as can be seen in Figure 1.2. This will heavily reduce the quality of the mesh. Therefore, it is important to make some adjustments to the mesh deformation method to ensure good mesh quality in these regions. Previous work has already been done by Mathew [22], where the radial basis function interpolation method was modified for regions with small clearance gaps using a sliding condition. When there is only a small distance between the nodes on a deforming boundary and a fixed boundary, the sliding of boundary nodes becomes essential to obtain a good quality mesh after deformation. The sliding boundary condition allows the nodes to move along their respective boundary which improves the mesh quality as can be seen on the right side in Figure 1.2. However, this sliding boundary method can still be further improved on in terms of computation time and mesh quality.

Another interesting aspect of mesh deformation is the application on periodic domains. Periodicity can also be found in the turbomachinery application. One single blade is repeated multiple times and therefore, a single blade passage can be isolated and used as the computational domain (Fig. 1.3). This computational domain uses periodic boundaries in order to serve as a representation for the full problem domain. The use of periodic boundaries thus allows for a reduction of the computational cost. However, as can be seen in Figure 1.3b, it can occur that the regions between the blade and the upper and lower periodic boundaries are small. Hence, similar as with the blade tip clearance gap, some issues might occur with deforming the mesh for larger displacements of the blade [1]. An upward movement of the blade can cause the mesh cells above the blade to get squeezed together while below the blade the mesh cells will be stretched. In those cases it might be more beneficial for the mesh quality to allow the periodic boundaries to move with the deforming blade.



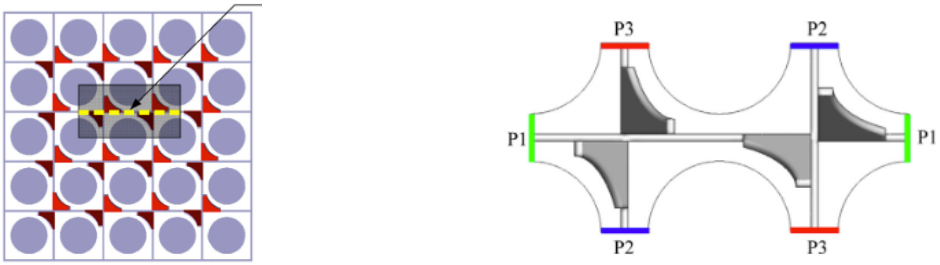
(a) Three-dimensional computational domain [26].

(b) Two-dimensional computational domain [13].

Figure 1.3: A single blade passage of a turbine as periodic computational domain.

1.3. Applications

Besides turbomachinery, the nuclear industry is also a field where small clearance gaps and periodic domains are commonly encountered. In the nuclear industry, flow induced vibration plays an important role. These occur for example in nuclear reactors where fuel assemblies are present with small spacing between the rods. The turbulent fluid flow around the rods will cause an interaction with the fuel rod structures [32]. The typical fuel assemblies consist of a large number of rods with small spacing between them. Performing a simulation for these assemblies might be impossible due to the limited computational power and memory. One way of reducing the size of the problem is by applying periodic boundary conditions [20]. Periodic boundary conditions can be used when there is both geometrical and physical periodicity present in the problem. This is expected to be the case for the central subchannels in a rod assembly. Figure 2.11a shows the cross-section of a 5x5 rod bundle where the red parts are the split-vane support grid. The shaded area that the arrow points to is the two-subchannel model which is shown in Figure 2.11b with three pairs of periodic boundary conditions. The periodic boundary pairs for these cases are dependent on the vane pattern. Small gaps also occur often between a rod and the containing wall as shown in Figure 1.5. Again, in these situations, sliding boundary nodes can be beneficial for the mesh quality in this region.



(a) Cross-section of 5x5 rod bundle model with split-vane support grid. (b) A two-subchannel model with periodic boundary conditions.

Figure 1.4: Geometry of the 5x5 and two-subchannel models [20].

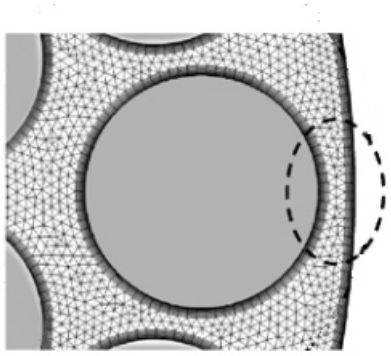


Figure 1.5: Small gap between rod and containing wall of a nuclear fuel assembly [6].

1.4. Research questions

The main research question for the thesis project is the following.

How does the implementation of periodic boundary conditions for the radial basis function mesh deformation method affect the mesh quality and computation time for regions including small clearance gaps?

The sub-questions that will help answer the main research question are as follows:

1. How do the periodic distance and equal displacement method for implementing periodic boundary conditions compare in terms of mesh quality and computation time?
    - How do the two periodic boundary methods compare using different radial basis functions?
    - How do the two periodic boundary methods compare for different periodic boundary types?
2. How do fixed periodic boundaries compare to moving periodic boundaries in terms of mesh quality?
3. How do sliding periodic boundaries compare to moving periodic boundaries in terms of mesh quality?
4. How does the use of a data reduction method influence the performance of the periodic boundary condition method in terms of mesh quality and computation time?

1.5. Outline of the thesis report

In Chapter 2, the methodology of the methods that are selected for this thesis project are elaborated on and justified. The results obtained with these methods are presented and discussed in Chapter 3 for the two-dimensional test cases and in Chapter 4 for the three-dimensional cases. Finally, the final conclusions of the thesis work and further recommendations are discussed in Chapter 5.

2

Methodology

The methodology chapter gives an overview and in depth description of the different methods and algorithms that are used in the project. The chapter starts with the description of radial basis function interpolation which is the method used for the mesh deformation. This is followed by a description of the different periodic boundary types (Sec. 2.2) and methods (Sec. 2.3) that can be used to apply mesh deformation on periodic domains. Next, the sliding method that is used in cases of small clearance gaps is explained in Section 2.4. Finally, in Section 2.6 the mesh quality metrics are described that are used to compare the different methods in terms of deformed mesh quality.

2.1. Regular radial basis function interpolation

A radial basis function (RBF) interpolation method is used for interpolating the boundary displacements to the internal nodes of a mesh through the use of radial basis functions [7]. The interpolation function s for node $\vec{x}$ can be represented as a sum of radial basis functions:

$$s(\vec{x}) = \sum_{j=1}^{N_\mathrm{b}} \alpha_j \phi\left(\left\|\vec{x} - \vec{x}_{\mathrm{b}_j}\right\|\right) + p(\vec{x}) \tag{2.1}$$

where N_b is the number of boundary nodes, α_j are the interpolation coefficients, ϕ is the selected radial basis function with respect to the Euclidean distance $\left\|\vec{x} - \vec{x}_{\mathrm{b}_j}\right\|$, $\vec{x}_b$ are the boundary nodes where the displacement is known and p is a polynomial. There will be a different interpolation function for each coordinate direction: x, y and z (3D). The vectors in the interpolation function represent 3D vectors containing the spatial coordinates of the node in x-, y- and z-direction. In Equation 2.1, the interpolation coefficients and the polynomial are unknown and thus have to be computed first before the interpolation function s can be set up. They are determined by using the known displacements of the boundary nodes, d_b, and the following conditions:

$$s(\vec{x}_{b_j}) = \vec{d}_{b_j} \tag{2.2}$$

$$\sum_{j=1}^{N_b} \alpha_j q\left(\vec{x}_{b_j}\right) = 0 \tag{2.3}$$

The interpolation condition (Eq. 2.2) states that the evaluation of the interpolation function for the boundary nodes should equal the known displacements of the boundary nodes. The homogeneous equation (Eq. 2.3) is valid for every polynomial q that has a degree equal to or lower than the degree of polynomial p. With these conditions, the following system can be set up that will be solved for $\vec{\alpha}$ and $\vec{\beta}$:

$$\begin{bmatrix} \vec{d}_\mathrm{b} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi_{b,b}} & \boldsymbol{P_b} \\ \boldsymbol{P_b^T} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \vec{\alpha} \\ \vec{\beta} \end{bmatrix} \tag{2.4}$$

where $\boldsymbol{\Phi_{b,b}}$ is the interpolation matrix containing the RBF evaluation values with respect to the Euclidean distance between the boundary nodes, $\left\|\vec{x}_{b_i} - \vec{x}_{\mathrm{b}_j}\right\|$, $\boldsymbol{P_b}$ is the polynomial matrix and $\vec{\beta}$ are the polynomial coefficients. The matrix containing the interpolation matrix and the polynomial matrices will be referred to as the *total* interpolation matrix, C. The solving of this system will be called the evaluation step throughout this report. This step will be called the update step. For this project, the additional polynomial is not useful as it typically helps to ensure exact interpolation of linear transformations [36]. Linear interpolation allows for rigid body translations and rotations, which can also be represented in a simpler way by using a moving reference frame. When omitting the polynomial, the interpolation function will be shortened to:

$$s(\vec{x}) = \sum_{j=1}^{N_\mathrm{b}} \alpha_j \phi\left(\left\|\vec{x} - \vec{x}_{\mathrm{b}_j}\right\|\right) \tag{2.5}$$

Now, using only the interpolation condition will be sufficient to find the interpolation coefficients. This results in the following system where the final *total* interpolation matrix will have the size $N_b \times N_b$ as it only contains $\boldsymbol{\Phi_{b,b}}$:

$$\vec{d}_b = \boldsymbol{\Phi_{b,b}}\vec{\alpha} \tag{2.6}$$

This system is solvable and will give a unique interpolation function when the RBF ϕ is positive definite [30]. Although most radial basis functions are always positive definite, some are only conditionally positive definite such as the thin plate spline (TPS) or the multiquadric function. The TPS function is strictly positive definite of order 2, which is the form that is given in Table 2.2. If the RBF is not positive definite, it will still be necessary to use the additional polynomial. In case of the TPS, a polynomial of the first order would be added which has the form:

$$p(\vec{x}) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 z \tag{2.7}$$

The polynomial will be different for every direction: x, y and z. For a linear polynomial as in Equation 2.7, the polynomial matrix $\boldsymbol{P_b}$ will have the form:

$$\boldsymbol{P_b} = \begin{bmatrix} 1 & x_{b_1} & y_{b_1} & z_{b_1} \\ 1 & x_{b_2} & y_{b_2} & z_{b_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{b_{N_b}} & y_{b_{N_b}} & z_{b_{N_b}} \end{bmatrix} \tag{2.8}$$

As the radial basis functions used in this project are all positive definite, the system in Equation 2.6 can be used to solve for the coefficients. Once the interpolation coefficients $\vec{\alpha}$ are computed, the displacement of internal node i, d_{i_i}, is found by evaluating the interpolation function s for every coordinate direction:

$$\begin{aligned} d_{i_{i,x}} &= s_x\left(\vec{x}_{i_i}\right) = \sum_{j=1}^{N_\mathrm{b}} \alpha_{x_j} \phi\left(\left\|\vec{x}_{i_i} - \vec{x}_{b_j}\right\|\right) \\ d_{i_{i,y}} &= s_y\left(\vec{x}_{i_i}\right) = \sum_{j=1}^{N_\mathrm{b}} \alpha_{y_j} \phi\left(\left\|\vec{x}_{i_i} - \vec{x}_{b_j}\right\|\right) \\ d_{i_{i,z}} &= s_z\left(\vec{x}_{i_i}\right) = \sum_{j=1}^{N_\mathrm{b}} \alpha_{z_j} \phi\left(\left\|\vec{x}_{i_i} - \vec{x}_{b_j}\right\|\right) \end{aligned} \tag{2.9}$$

The computation of the displacements will be referred to as the update step in this report as the computed displacements are used to update the location of the nodes.

Since the RBF interpolation method is an interpolation method, it does not require any connectivity information as was mentioned in Section 1.1. Therefore, there is no need to solve this system of equations for every single mesh node, including the internal nodes. It suffices to solve the system only for (a part of) the boundary nodes and then simply update the internal nodes. In general it could be said that the method does not require a large system of equations to be solved (w.r.t. connectivity-based methods). However, for large 3D problems it can become too costly and a data reduction method might be

required such as a greedy algorithm, which will be further elaborated on in Section 2.5. Furthermore, the performance of the method is influenced by the RBF that is used but overall it is considered a very robust method that is able to handle large deformations.

It is important to mention that relative displacements are used in this project. This means that the displacement of a node at a certain step is w.r.t. the location the node had at the previous step. This is different from absolute displacements where the displacement is w.r.t. the initial node location before the simulation. The relative displacement method is advantageous in case of rotations as a curve will be followed instead of a straight line from the initial position to the updated location [22].

2.1.1. Types of RBF

The RBF's can be split up into two groups: functions with local and with global support. The local support generally implies that the nodes within a support radius r of the center are influenced and outside of the circle with radius r, the RBF is zero. The support radius can thus be used to scale the reach that the influence of the deformation has. For global support, this influence is extended over an infinite domain and thus the matrix systems for this type of function will be dense, making these RBF's generally less efficient than compact supported functions. A list of compact support radial basis functions tested in [7], proposed by Wendland [37], is given in Table 2.1. In these functions, $\xi = \delta/r$, thus it can be said that the RBF is scaled by support radius r. In the regular RBF method, the distance function δ is the Euclidean distance between two nodes. The global support functions that were tested are listed in Table 2.2. In some of these functions, a is used as a shape parameter. The functions were tested for different deformation cases such as rotation and translation as well as rigid body rotation. The CP functions are polynomials with the lowest degree to obtain a C^n continuous basis function with $n \in \{0, 2, 4, 6\}$ and the CTPS functions are based on the thin plate spine to obtain C^n continuous basis functions with $n \in \{0, 1, 2\}$ [7].

The five best performing RBF's in terms of deformed mesh quality according to de Boer et al. [7] are CP C^2, CTPS C^1, CTPS C_a^2, CTPS C_b^2 and TPS. In terms of efficiency, the CP C^2 function was the most efficient with the TPS function in second place. Hence, the best performing RBF for the different tests was the $CP\ C^2$ function. In other research performed by Bos et al. [34], the TPS function resulted to be the most robust and highest in mesh quality. In the thesis report by Mathew [22], which is used as the base for this thesis work, both the CP C^2 and TPS functions are used based on these results and the same will be done in this thesis. In the report, the support radius that will be used for the CP C^2 RBF is 2.5 times the maximum length of the case-specific mesh as this resulted in the optimal quality and robustness [7].

Name	$f(\xi)$
CP C^0	$(1-\xi)^2$
CP C^2	$(1-\xi)^4(4\xi+1)$
CP C^4	$(1-\xi)^6\left(\frac{35}{3}\xi^2+6\xi+1\right)$
CP C^6	$(1-\xi)^8\left(32\xi^3+25\xi^2+8\xi+1\right)$
CTPS C^0	$(1-\xi)^5$
CTPS C^1	$1+\frac{80}{3}\xi^2-40\xi^3+15\xi^4-\frac{8}{3}\xi^5+20\xi^2\log(\xi)$
CTPS C_a^2	$1-30\xi^2-10\xi^3+45\xi^4-6\xi^5-60\xi^3\log(\xi)$
CTPS C_b^2	$1-20\xi^2+80\xi^3-45\xi^4-16\xi^5+60\xi^4\log(\xi)$

Table 2.1: List of radial basis functions with compact support [7].

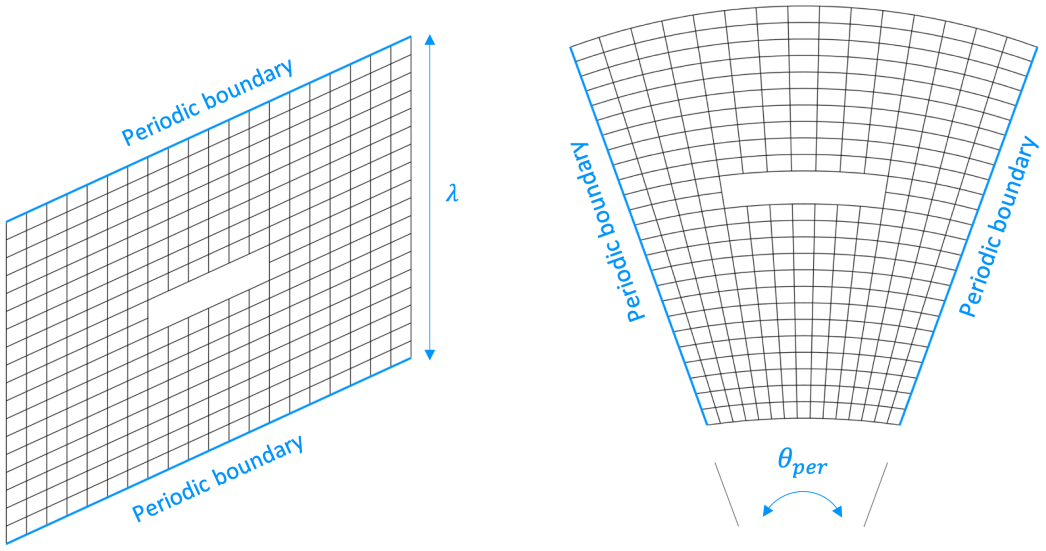
Name	$f(x)$
Thin plate spline (TPS)	$x^2 \log(x)$
Multiquadric biharmonics (MQB)	$\sqrt{a^2 + x^2}$
Inverse multiquadric biharmonics (IMQB)	$\sqrt{\frac{1}{a^2+x^2}}$
Quadric biharmonics (QB)	$1 + x^2$
Inverse quadric biharmonics (IQB)	$\frac{1}{1+x^2}$
Gaussian (Gauss)	e^{-x^2}

Table 2.2: List of radial basis functions with global support [7].

2.2. Periodic boundary types

Periodic boundary conditions are often used to simulate a small part of a bigger problem that contains a repeating pattern. This helps to reduce the computational cost. The periodic conditions can be applied to identical boundaries of the same length and shape that are located at a certain distance from one another. The fluid flow across these boundaries is identical and therefore certain variables must be the same on both boundaries. These variables can, for example, be displacement of a node on the boundary or the velocity of a particle at the boundary.

From the different applications discussed in Chapter 1, two types of periodic boundaries can be distinguished. The periodic boundaries can either be periodic in a translational manner (Fig. 2.1a) or in a rotational manner (Fig. 2.1b). With translational periodic boundaries, the distance between the boundaries has a constant value λ along the boundary. For rotational periodic boundaries, there is a periodic angle θ_{per} between the boundaries. Both types will be used in testing during this project.



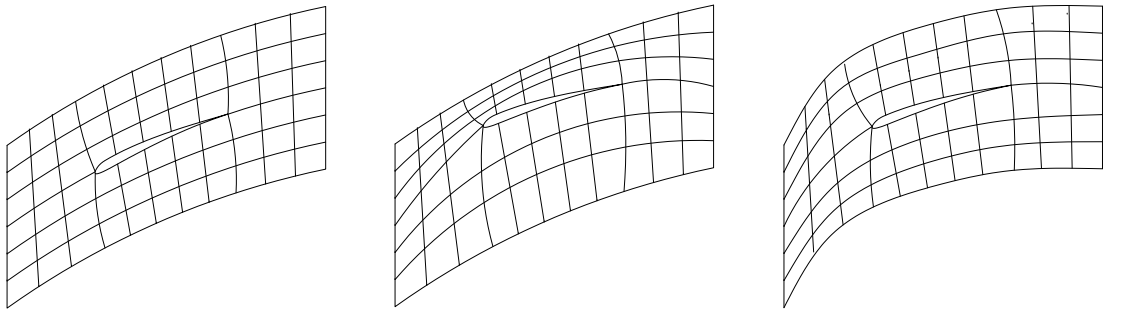
(a) Translational periodic boundaries.

(b) Rotational periodic boundaries.

Figure 2.1: Different types of periodic boundaries.

In case of large deformations, it might become inevitable to allow movement of these periodic boundaries in order to maintain a good mesh quality [1]. In Figure 2.2, an example is shown of a periodic domain around a blade that undergoes a relatively large deformation w.r.t. the size of the domain. Figure 2.2a shows the initial mesh where the blade is yet to be deformed. The periodic boundaries are the upper and lower boundaries of the mesh. In Figure 2.2b, the blade undergoes an upward deformation and the periodic boundaries are fixed in space. When the blade deforms upwards, the mesh cells in between the blade and the upper boundary might be squeezed together while on the other side of the blade the cells might be elongated. Hence, above the blade, the clearance space between the blade and the periodic boundary becomes small. By allowing the periodic boundaries to move in a similar

manner as the blade did, a better mesh quality can be maintained as is illustrated in Figure 2.2c). The displacement of the boundaries should be identical on both periodic boundaries.
Since the RBF interpolation mesh deformation method is used to deform the mesh, some adaptations will have to be made to this method to guarantee the periodic boundaries to be displaced identically. In the next section, two different methods for implementing these periodic conditions are described.



(a) Initial mesh before deformation.

(b) Deformed mesh with fixed periodic boundaries.

(c) Deformed mesh with deformed periodic boundaries.

Figure 2.2: Mesh deformation of a periodic mesh around a deforming blade, including the displacement of the periodic boundaries.

2.3. Periodic boundary displacement methods

There are two different methods for implementing periodic boundary conditions in the RBF interpolation method that are tested in this project. The first method is based on making the distance function used in the RBF periodic. The second method is based on making the displacement of the periodic nodes equal. The periodic boundary displacement methods will often be referred to as the periodic (boundary) methods in this report.
Before diving deeper into these periodic methods, a distinction has to be made between the different types of boundary nodes that are encountered when using the RBF interpolation method on periodic domains. The boundary nodes can be divided into three different groups as shown in Figure 2.3. The boundary nodes that are usually used in the evaluation step (see Sec. 2.1) are divided into N_m moving nodes, N_p periodic nodes and N_S sliding nodes. The moving nodes are those that have a known displacement. This displacement can also be equal to zero, making them fixed. The periodic nodes are those located on the periodic boundaries. Finally, the sliding nodes are those located on a boundary that is fixed but these nodes are still allowed to slide along their respective fixed boundary. Depending on which periodic boundary displacement method that is used, different node groups are used in the evaluation step.

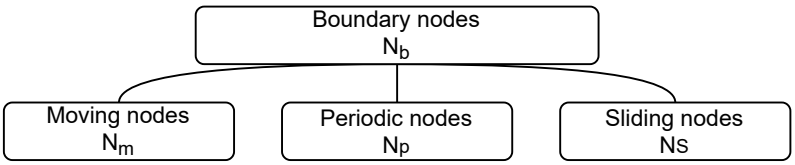


Figure 2.3: Boundary node division for periodic boundary methods.

2.3.1. Method 1: Periodic distance

The first periodic boundary displacement method is based on a periodic distance between the control nodes. The control nodes refer to the boundary nodes that are used in the evaluation step of the RBF interpolation method. The periodic distance is obtained by modifying the distance function δ, which the RBF is a function of ($\phi(\delta)$). The distance function used in the regular RBF is the Euclidean distance as was shown in Equation 2.5:

$$s(\vec{x}) = \sum_{j=1}^{N_\mathrm{b}} \alpha_j \phi\left(\left\|\vec{x} - \vec{x}_{\mathrm{b}_j}\right\|\right) \qquad \text{(2.5 revisited)}$$

In this periodic method, the distance function is made periodic by using a sine function. This periodic distance function is used in both the RBF evaluation step where the interpolation coefficients are computed as well as the update step where the displacements are computed. This gives better results than when solely using it to update the mesh.
The periodic distance function will have a different formulation depending on the type of periodic boundary used, translational or rotational.

Translational periodic boundaries
For translational periodic boundaries, the Cartesian coordinate system is used for the distance functions. For 2D cases, the implies that the euclidean distance equation will have the form:

$$\delta = \sqrt{\delta x^2 + \delta y^2} = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{2.10}$$

Here, the subscript c refers to control node. For periodic meshes this distance equation can be made periodic in a specific direction. If the periodic boundaries are periodic in the y-direction, the distance function can be made periodic over a distance λ by applying the sine function in y-direction [1]:

$$\delta y = \frac{\lambda}{\pi} \sin\left[(y - y_c)\frac{\pi}{\lambda}\right] \tag{2.11}$$

The function δy is plotted in Figure 2.4. When the distance between two nodes in y-direction, $y - y_{c,i}$, is small (relative to λ), the function δy has a value that is also small and close to the value of $y - y_{c,i}$. This implies that the distance of nodes located in the vicinity of the control nodes will be computed in the same way for all directions and the influence of the periodicity is not present [1]. This is achieved by multiplying the sine function with the term $\frac{\lambda}{\pi}$. With the adapted distance function in y-direction, the total periodic distance function δ becomes:

$$\delta = \sqrt{(x - x_c)^2 + \left(\frac{\lambda}{\pi} \sin\left[(y - y_{c_i})\frac{\pi}{\lambda}\right]\right)^2} \tag{2.12}$$

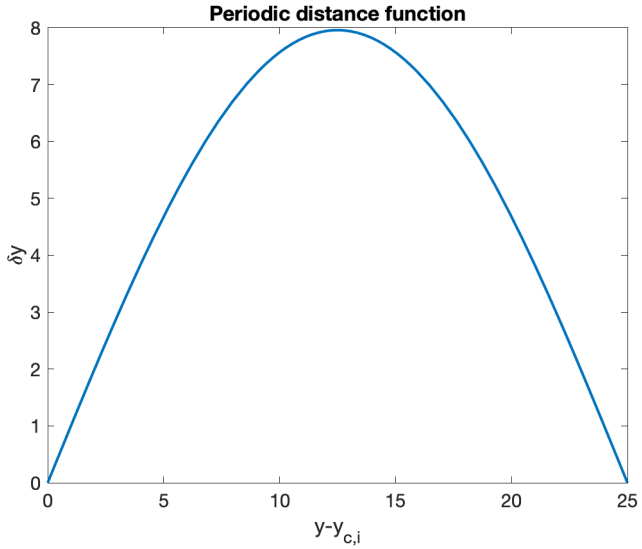


Figure 2.4: Periodic distance function for y-direction.

For 3D cases, the same principle is used. Only the Euclidean distance formulation changes to $\delta = \sqrt{\delta x^2 + \delta y^2 + \delta z^2}$. Hence, for periodic boundaries in y-direction the periodic distance function becomes:

$$\delta = \sqrt{(x - x_c)^2 + \left(\frac{\lambda}{\pi} \sin\left[(y - y_c)\frac{\pi}{\lambda}\right]\right)^2 + (z - z_c)^2} \tag{2.13}$$

Rotational periodic boundaries
When the mesh has rotational periodic boundaries, the Euclidean distance will be expressed using polar coordinates (in 2D) instead of Cartesian coordinates. This way, it is easier to implement the periodicity over a specific periodic angle θ_{per}. Therefore, the node coordinates first require a transformation from Cartesian to polar coordinates. The formulas used for these transformations can be found in Appendix B. In 2D, the Euclidean distance expression in polar coordinates is:

$$\delta = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_2 - \theta_1)} \tag{2.14}$$

In order to make the distance function periodic over angle θ_{per}, a sine function is applied to the part of the equation containing the difference between the two angles:

$$\delta = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos\left[\frac{\theta_{per}}{\pi} \sin\left((\theta_2 - \theta_1)\frac{\pi}{\theta_{per}}\right)\right]} \tag{2.15}$$

For 3D rotational periodic cases, the Cartesian coordinates are first transformed to cylindrical coordinates. The distance function then changes to:

$$\delta = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos\left[\frac{\theta_{per}}{\pi} \sin\left((\theta_2 - \theta_1)\frac{\pi}{\theta_{per}}\right)\right] + (z_2 - z_1)^2} \tag{2.16}$$

Once the evaluation and update step are done and the displacements of the nodes are computed, the polar and cylindrical coordinates are converted back to Cartesian coordinates to function with the rest of the MATLAB code. All the transformations between the different coordinate systems can be found in Appendix B. Since this method only uses a different distance function, the size of the interpolation matrix stays the same as for the regular RBF interpolation method. However, when using periodic boundaries, the evaluation step only uses the moving nodes as control nodes. Therefore, the interpolation matrix will have the size $(N_m) \times (N_m)$. Then the sliding and periodic node displacements are computed in the update step. Once these nodes have an updated location, the sliding nodes are projected back onto the boundary as they will have moved away from it. This method for sliding nodes is further elaborated on in Section 2.4.1. Finally, the internal nodes are updated where all boundary nodes are used as control nodes and also the periodic distance function is used in both steps of the RBF interpolation.

It is important to note that some extra measures are required in case of periodic control nodes. If the control nodes contain a periodic node pair, one node per periodic pair is removed from the control nodes. This is necessary because otherwise the interpolation matrix would be singular because of identical matrix rows. This would make it impossible to invert the matrix and compute the interpolation coefficients.

2.3.2. Method 2: Equal displacement

The second periodic boundary displacement method is based on equal displacement of the periodic nodes. Hence, a pair of periodic nodes will be displaced in an identical way, which requires the following conditions to be satisfied for every periodic node pair, p_1 and p_2:

$$d(\vec{p_1}) = d(\vec{p_2}) \Leftrightarrow d(\vec{p_1}) - d(\vec{p_2}) = 0 \tag{2.17}$$

This condition gives $\frac{1}{2}N_p$ equations, but there are N_p periodic nodes, each for which an interpolation coefficient has to be computed. Hence, another $\frac{1}{2}N_p$ equations are required and therefore, a second condition is used. This condition states that the periodic interpolation coefficients should have equal influence but opposite in direction:

$$\vec{\alpha}_{p_1} + \vec{\alpha}_{p_2} = 0 \tag{2.18}$$

This condition can be considered to be a local correction that guarantees that the periodic pairs will have a periodic displacement. With this additional set of $\frac{1}{2}N_p$ equations, there are sufficient equations to

compute the periodic node interpolation coefficients. Next to the periodic nodes, the moving nodes are also part of the control nodes. For the moving nodes, the interpolation condition (Eq. 2.2) can be used to find their interpolation coefficients. The interpolation condition states that the known displacements of the moving nodes, $\vec{d_m}$, have to be equal to the interpolation function evaluations of these nodes, $s(\vec{x_m})$. However, now the interpolation function will have contributions from both the moving nodes and the nodes on both periodic boundaries, p_1 and p_2. Hence, the displacements in both x- and y-direction are shown in Equations 2.19 and 2.20.

$$d_{m_x}(\vec{x}_{m_i}) = \sum_{j=1}^{N_m} \alpha_{m_x,j}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{m,j}\right\|\right) + \sum_{k=1}^{N_{p_1}} \alpha_{p_{1_x},k}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{p_1,k}\right\|\right) + \sum_{k=1}^{N_{p_2}} \alpha_{p_{2_x},k}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{p_2,k}\right\|\right) \tag{2.19}$$

$$d_{m_y}(\vec{x}_{m_i}) = \sum_{j=1}^{N_m} \alpha_{m_y,j}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{m,j}\right\|\right) + \sum_{k=1}^{N_{p_1}} \alpha_{p_{1_y},k}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{p_1,k}\right\|\right) + \sum_{k=1}^{N_{p_2}} \alpha_{p_{2_y},k}\phi\left(\left\|\vec{x}_{m,i} - \vec{x}_{p_2,k}\right\|\right) \tag{2.20}$$

With these three conditions, the *total* interpolation matrix can be set up. Although the x- and y-direction are not coupled and can be solved in separate systems, they are shown here in one matrix system.

$$\begin{bmatrix} d_{m_x} \\ d_{m_y} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{mm} & 0 & \boldsymbol{\Phi}_{m,p_1} & 0 & \boldsymbol{\Phi}_{m,p_2} & 0 \\ 0 & \boldsymbol{\Phi}_{mm} & 0 & \boldsymbol{\Phi}_{m,p_1} & 0 & \boldsymbol{\Phi}_{m,p_2} \\ \boldsymbol{\Phi}_{p_1,m} - \boldsymbol{\Phi}_{p_2,m} & 0 & \boldsymbol{\Phi}_{p_1,p_1} - \boldsymbol{\Phi}_{p_2,p_1} & 0 & \boldsymbol{\Phi}_{p_1,p_2} - \boldsymbol{\Phi}_{p_2,p_2} & 0 \\ 0 & \boldsymbol{\Phi}_{p_1,m} - \boldsymbol{\Phi}_{p_2,m} & 0 & \boldsymbol{\Phi}_{p_1,p_1} - \boldsymbol{\Phi}_{p_2,p_1} & 0 & \boldsymbol{\Phi}_{p_1,p_2} - \boldsymbol{\Phi}_{p_2,p_2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{m_x} \\ \alpha_{m_y} \\ \alpha_{p_{1_x}} \\ \alpha_{p_{1_y}} \\ \alpha_{p_{2_x}} \\ \alpha_{p_{2_y}} \end{bmatrix} \tag{2.21}$$

Studying the system of equations, it is clear that only the moving and periodic nodes are included in the evaluation step and not the sliding nodes. For the equal displacement method, the size of the interpolation matrix is therefore $(N_m + N_p) \times (N_m + N_p)$. This means that the equal displacement method has to invert a larger matrix than the periodic distance method when computing the interpolation coefficients. This should become clear when comparing computation times for both periodic methods. Next, during the update step, both the periodic an sliding node displacements are computed using the following formula (only shown for the periodic nodes) which sums up the contributions of the moving nodes and the periodic nodes:

$$d_x(p_i) = \sum_{j=1}^{N_m} \alpha_{m_x,j}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{m,j}\right\|\right) + \sum_{k=1}^{N_{p_1}} \alpha_{p_{1_x},k}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{p_1,k}\right\|\right) + \sum_{k=1}^{N_{p_2}} \alpha_{p_{2_x},k}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{p_2,k}\right\|\right) \tag{2.22}$$

$$d_y(p_i) = \sum_{j=1}^{N_m} \alpha_{m_y,j}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{m,j}\right\|\right) + \sum_{k=1}^{N_{p_1}} \alpha_{p_{1_y},k}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{p_1,k}\right\|\right) + \sum_{k=1}^{N_{p_2}} \alpha_{p_{2_y},k}\phi\left(\left\|\vec{x}_{p,i} - \vec{x}_{p_2,k}\right\|\right) \tag{2.23}$$

Once the sliding nodes are updates, they are projected back to their boundary as will be explained in Section 2.4.1. Then the regular RBF method is used with the known moving, periodic and sliding node displacements to update the internal mesh.

Deformation effect on method performance

The equal displacement method can still be improved by adapting the condition for the periodic coefficients. Where both coefficients have an equal influence in Equation 2.18, another combination of the two periodic node coefficients can be used:

$$\omega \cdot \vec{\alpha}_{p_1} + (1 - \omega) \cdot \vec{\alpha}_{p_2} = 0 \tag{2.24}$$

Here, ω is a variable ranging from zero to one. For different types of deformation, a different value of ω will yield an optimal mesh quality. This is investigated by performing a sweep for different ω values using steps of 0.01 and comparing the minimum and mean mesh quality achieved. To determine the optimal mesh quality, the ω for which the minimum quality is optimal is considered to be the optimal

ω value if the mean quality for this value is still within a 10% range from the optimal mean quality. If, however, this value yields a significant decrease in mean mesh quality, another ω value is selected where both the mean and minimum mesh quality are within 10% of there optima.

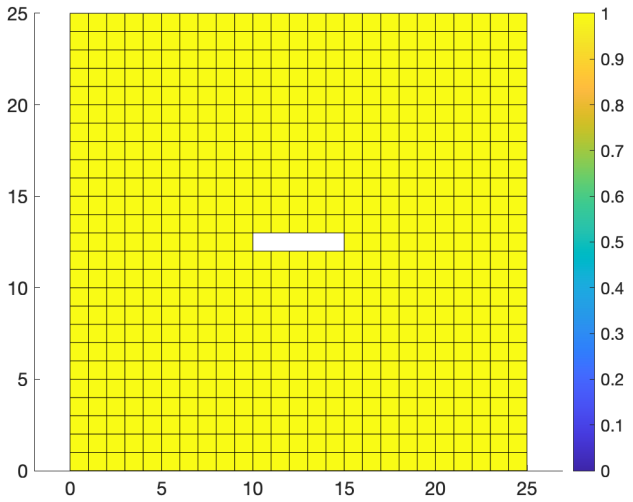


Figure 2.5: Mesh 1: structured square mesh.

The investigation is done on a square mesh as shown in Figure 2.5, which will later be used in the 2D results chapter. The nodes are periodic in the y-direction and are allowed to move. The subscript p_1 refers to the lower periodic boundary and p_2 to the upper periodic boundary. The block in the middle of the mesh will undergo displacement in vertical, horizontal and diagonal direction as well as a rotation. Although this investigation showed that the optimal combination is strongly dependent on the specific deformation applied, some generalizations can be made to facilitate the choice of combination without the need to do a sweep for every single different deformation. For every case shown below, three figures displaying the deformed mesh are included in Appendix C for $\omega = 0$, $\omega = 0.5$ and $\omega = 1$. These figures can be consulted to help visualise the results.

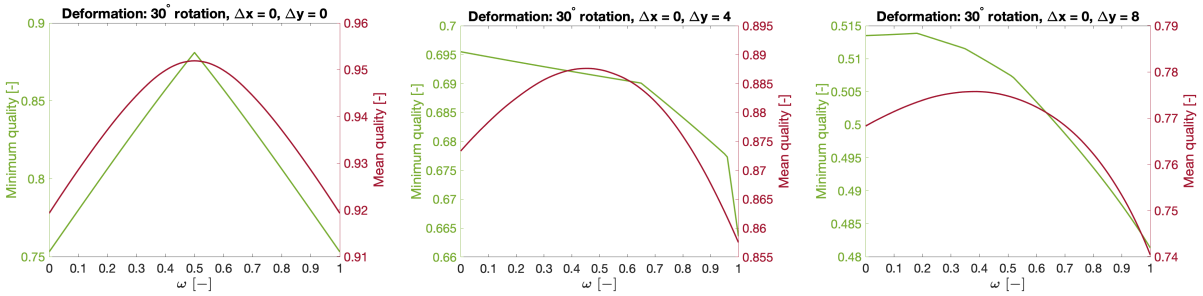
Vertical displacement

Figures 2.6 and 2.7 show the minimum and mean mesh quality for a 30° and 60° rotation with zero horizontal displacement and an increasing vertical displacement. It is clear that for pure rotation, having an equal influence from both sides yields the best mesh quality for both rotation angles. When the block starts to move upwards, the optimal quality starts to shift to the left of the graph. As the vertical displacement reaches 8 units, the optimal quality for both rotation angles is reached for $\omega = 0$. For a 30° rotation, this was already the case for the smaller vertical displacement of four units but for the 60° rotation this would imply a 10% loss in optimal minimum quality. Therefore, using $\omega = 0.5$ when the translation is smaller still allows for a good minimum and mean quality independent of the rotation angle.

In case of a downward vertical displacement, the graphs are equal to the mirrored graphs of the upward displacement meaning large downward vertical displacement with $\omega = 1$ yields the most optimal mesh quality. Furthermore, changing the sign of the rotation does not influence the graph.

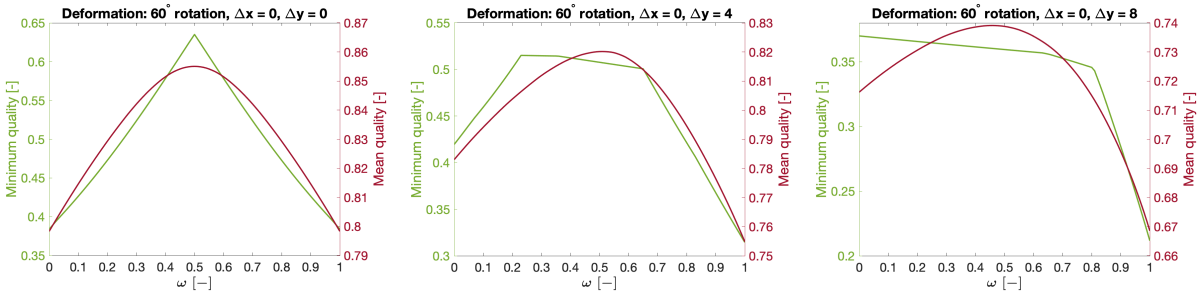
Horizontal displacement

In Figure 2.8 and 2.9, a similar investigation is done for a combination of rotation and horizontal translation. The same trend can be noticed for increasing horizontal displacement to the right as was present for increasing upward vertical displacement. Hence, when moving more to the right, the optimal mesh quality is reached for $\omega = 0$. Also for this case, the trend is already clear for a four unit displacement for



(a) Zero units vertical displacement. (b) Four units vertical displacement. (c) Eight units vertical displacement.

Figure 2.6: Mesh quality using equal displacement method with 30° rotation and upward vertical translation.



(a) Zero units vertical displacement. (b) Four units vertical displacement. (c) Eight units vertical displacement.

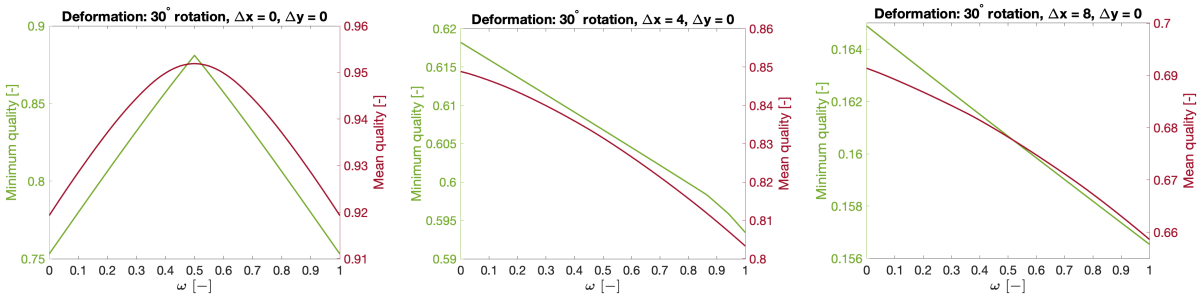
Figure 2.7: Mesh quality using equal displacement method with 60° rotation and upward vertical translation.

the 30° rotation while the minimum quality for 60° rotation significantly decreases for $\omega = 0$. Therefore, although the optimal quality already shifts to the left of the graph, using $\omega = 0.5$ for smaller translation cases might be a better choice as this will still give a decent result for both rotation angles, keeping the optimal quality loss within 5%.

When changing the sign of rotation, the graphs will be mirrored, since the periodic boundaries are in the y-direction. Also when the horizontal displacement is to the left, the graph will be mirrored.

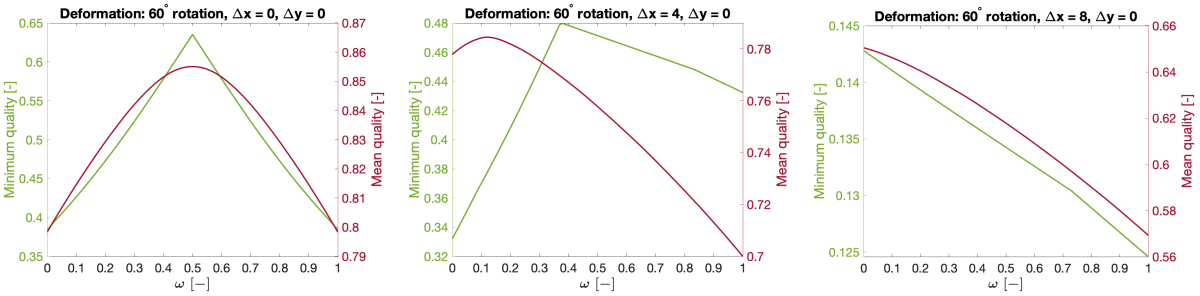
Diagonal displacement

When both a vertical and horizontal translation are applied as in Figure 2.10, things start to become more complex. In general, if the translation motion and the rotation move the block to a specific corner, the optimal value will be related to the periodic boundary the corner belongs to. Hence, if the block moves to the upper right corner and the rotation is also positive (Fig. 2.10a), rotating the right part of the block towards the corner, $\omega = 0$ will be optimal. When both the translation and rotation move the block more towards one of the bottom corners (Fig. 2.10b), $\omega = 1$ will give the most optimal result.



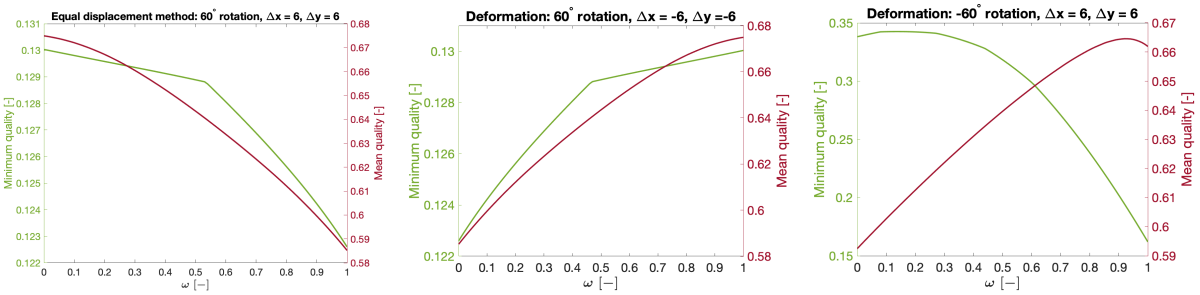
(a) Zero units horizontal displacement. (b) Four units horizontal displacement. (c) Eight units horizontal displacement.

Figure 2.8: Mesh quality using equal displacement method with 30° rotation and positive horizontal translation.



(a) Zero units horizontal displacement. (b) Four units horizontal displacement. (c) Eight units horizontal displacement.

Figure 2.9: Mesh quality using equal displacement method with 60° rotation and positive horizontal translation.



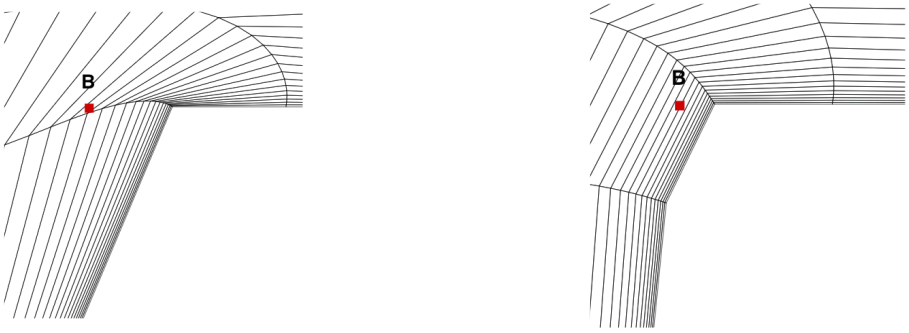
(a) Six units horizontal and vertical displacement with 60° rotation. (b) Six units negative horizontal and vertical displacement with 60° rotation. (c) Six units horizontal and vertical displacement with -60° rotation.

Figure 2.10: Mesh quality using equal displacement method with rotation and diagonal translation after 20 steps using Wendland C^2 (r=62.5).

When, however, the translation moves the block towards a corner but the rotation turns it away from the corner (Fig. 2.10c), $\omega = 0.5$ is a more optimal choice. Of course, every deformation that does not clearly move toward a corner has a specific ω value for which the optimal mesh quality is obtained but as this would take to much time to compute this for every single deformation possibility, taking $\omega = 0.5$ is a good approximation. Furthermore, when changing the sign of the displacement in x- or in y-direction, the rotation also has to change signs to obtain the same graph.

2.4. Periodic sliding

As was already mentioned in Section 1.2, it can be preferred to not keep boundary nodes fixed when deforming a mesh in the case of small clearance gaps. Instead, they are given additional degrees of freedom which reduces mesh skewness and improves the mesh quality. One way to do this is to let those boundary nodes displace, as is done with the periodic boundary displacement methods discussed in Section 2.3. However, sometimes it is not necessary to displace the full boundary but just allowing the nodes to slide along their boundary as illustrated in Figure 2.11 is sufficient to improve the mesh quality. The sliding method can be applied to periodic nodes, in which case they have to slide in a periodic manner on both boundaries. On the other hand, the sliding can also be applied to nodes on the other non-periodic boundaries of the domain. There are two different methods used to slide the nodes. For the 2D cases, this is the pseudo-sliding method and for the 3D cases this is the direct sliding method. Both methods have been fully described in the thesis report by Mathew [22]. In the following sections they are described in a slightly different way, so that they also function with periodic nodes.



(a) Without sliding method.

(b) With sliding method.

Figure 2.11: Close-up view of 2D mesh of lower boundary deformed to form a ramp [16].

2.4.1. Sliding 2D: Pseudo-sliding method

The sliding method used for the 2D cases in the thesis is the pseudo sliding method that was proposed by Mathew [22]. This method was chosen because it was simple to implement both periodic boundary methods into this method.

The first step is to displace the sliding nodes as if they are internal nodes. Hence, the interpolation coefficients are computed using a periodic displacement method RBF interpolation where the boundary nodes have a known displacement. Then the interpolation coefficients are used to compute the displacements of solely the sliding nodes in the same manner as internal node displacements are usually computed. As the sliding nodes undergo a free periodic displacement, it is possible for them to have a displacement component perpendicular to their respective boundaries. This results in the sliding nodes no longer lying on the boundary but having moved away from it. Since sliding nodes are only allowed to slide along their boundary, an extra projection step is required to project the sliding nodes back to their boundary.

The projection algorithm is illustrated in Figure 2.12. The algorithm uses geometrical information of the mesh boundaries. This information includes the midpoints of the boundary line segments and their orientation. It also takes into account the index of the boundary the node belongs to so that it is projected back onto the correct boundary. The algorithm looks for the mid-point that is closest to the displaced boundary node (red node, initial position). The normal direction of the nearest midpoint (black arrow) is used as direction and the boundary node is projected back to the boundary, parallel to this direction (red arrow). Once the nodes are projected back, it will seem as if the nodes have slid along the boundary. Since, the periodic node pairs are freely displaced in a periodic manner, the projection back onto the boundary will automatically ensure the maintenance of periodicity.

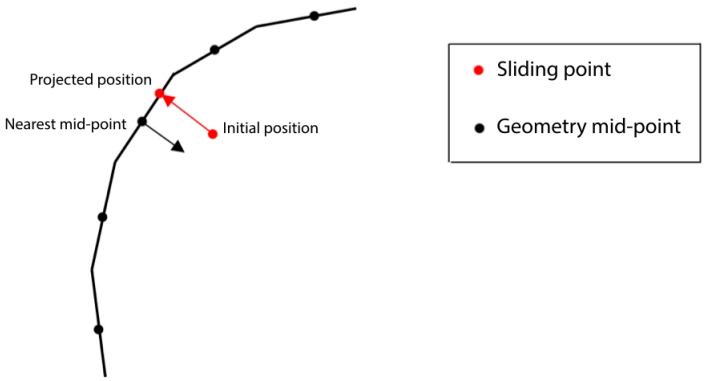


Figure 2.12: Projection algorithm used in pseudo-sliding method [22].

2.4.2. Sliding 3D: Direct sliding method

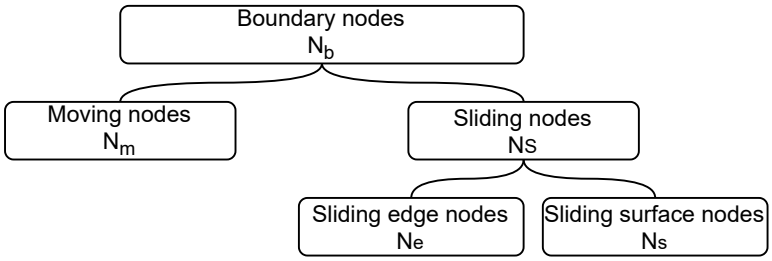


Figure 2.13: Division of the boundary nodes in 3D.

The sliding method chosen for 3D cases is referred to as the direct sliding method by Mathew [22]. The sliding of the nodes along a surface or edge is enforced by satisfying specific conditions [12]. Next to the interpolation condition, two sliding conditions are used to solve the system of equations. The different conditions apply to different groups of nodes. These groups are shown in Figure 2.13. The N_b boundary nodes are divided into two groups: N_m moving nodes (including nodes with zero displacement) and N_S sliding nodes. In 3D, the sliding nodes are further divided into N_e sliding edge nodes and N_s sliding surface nodes. The sliding nodes can either be periodic or not periodic. The indices of all control nodes are ordered starting with the moving nodes, followed by the sliding edge nodes and then the sliding surface nodes. The first condition applies to the moving nodes and is called the interpolation condition.

Interpolation condition

The interpolation condition states that the interpolation function evaluations equal the known displacements of the moving nodes. For the 3D sliding method, the control nodes contain the moving nodes, sliding edge nodes and the sliding surface nodes.

$$\forall i \in [1, N_m], \qquad \vec{s}\left(\vec{x}_{m_i}\right) = \vec{d}_{m_i}$$
$$\Leftrightarrow \quad \sum_{j=1}^{N_c} \phi\left(\left\|\vec{x}_{m_i} - \vec{x}_{c_j}\right\|\right) \vec{\alpha}_{x_j} = \vec{d}_{m_i}$$

Zero normal displacement condition

The zero normal displacement condition states that the normal component of the sliding node displacement has to be zero.

$$\forall i \in [1, N_s], \qquad s\left(\vec{x}_{s_i}\right) \cdot \vec{n} = 0$$
$$\Leftrightarrow \quad \sum_{j=1}^{N_c} \phi\left(\left\|\vec{x}_{s_i} - \vec{x}_{c_j}\right\|\right) \vec{\alpha}_{x_j} \cdot \vec{n}_x + \sum_{j=1}^{N_c} \phi\left(\left\|\vec{x}_{s_i} - \vec{x}_{c_j}\right\|\right) \vec{\alpha}_{y_j} \cdot \vec{n}_y \tag{2.25}$$
$$+ \sum_{j=1}^{N_c} \phi\left(\left\|\vec{x}_{s_i} - \vec{x}_{c_j}\right\|\right) \vec{\alpha}_{z_j} \cdot \vec{n}_z = 0$$

Zero tangential contributions condition

This condition states that the sliding nodes cannot have a contribution to the interpolation function in the tangential direction.

$$\forall i \in [N_m + 1, N_m + N_s], \qquad \vec{\alpha}_i \cdot \vec{t}_i = 0$$
$$\Leftrightarrow \quad \vec{\alpha}_{x_i} \cdot \vec{t}_{x_i} + \vec{\alpha}_{y_i} \cdot \vec{t}_{y_i} + \vec{\alpha}_{z_i} \cdot \vec{t}_{z_i} = 0 \tag{2.26}$$

The normal and tangential vectors

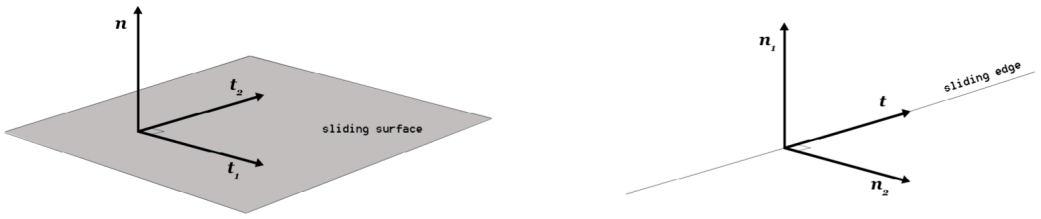
For the sliding conditions mentioned above, the normal and tangential vectors need to be known. A sliding surface node has one vector perpendicular to the surface and 2 tangent vectors (Fig. 2.14a). A sliding edge node has one tangent vector along the edge and two normal vectors to the edge (Fig. 2.14b). For the initial mesh, the sliding surface normal vector and the sliding edge tangential vector are known. From these, it is possible to compute the two other vectors representing the other orientations using the following equations which are based on the fact that all vectors are perpendicular to each other. If the initially known orientation is represented by $\vec{v}_1$, than a second orientation vector $\vec{v}_2$ is found by using the relation that $\vec{v}_1 \cdot \vec{v}_2 = 0$:

$$\vec{v}_1 = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.27}$$

$$\vec{v}_2 = \begin{bmatrix} v_x \left(v_y - v_z\right) \\ v_y \left(v_z - v_x\right) \\ v_z \left(v_x - v_y\right) \end{bmatrix} \tag{2.28}$$

A third orientation vector can be found by taking the cross-product of these two vectors as $\vec{v}_1 \times \vec{v}_2$ results in a third vector that is perpendicular to both vectors.

$$\vec{v}_3 = \begin{bmatrix} v_y \left(v_x - v_y\right) - v_z \left(v_z - v_x\right) \\ v_x \left(v_x - v_y\right) - v_z \left(v_y - v_z\right) \\ v_x \left(v_z - v_x\right) - v_y \left(v_y - v_z\right) \end{bmatrix} \tag{2.29}$$



(a) Sliding surface node. (b) Sliding edge node.

Figure 2.14: Normal and tangential vectors for sliding nodes [22].

With the three different conditions and the different orientations, the evaluation system can be set up to compute the moving and sliding interpolation coefficients:

$$\begin{bmatrix} \vec{d}_{m_x} \\ \vec{d}_{m_y} \\ \vec{d}_{m_z} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} \\ n_{e1_x}\boldsymbol{\Phi}_{em} & n_{e1_x}\boldsymbol{\Phi}_{ee} & n_{e1_x}\boldsymbol{\Phi}_{es} & n_{e1_y}\boldsymbol{\Phi}_{em} & n_{e1_y}\Phi_{ee} & n_{e1_y}\boldsymbol{\Phi}_{es} & n_{e1_z}\boldsymbol{\Phi}_{em} & n_{e1_z}\boldsymbol{\Phi}_{ee} & n_{e1_z}\boldsymbol{\Phi}_{es} \\ n_{e2_x}\boldsymbol{\Phi}_{em} & n_{e2_x}\boldsymbol{\Phi}_{ee} & n_{e2_x}\boldsymbol{\Phi}_{es} & n_{e2_y}\boldsymbol{\Phi}_{em} & n_{e2_y}\boldsymbol{\Phi}_{ee} & n_{e2_y}\boldsymbol{\Phi}_{es} & n_{e2_z}\boldsymbol{\Phi}_{em} & n_{e2_z}\boldsymbol{\Phi}_{ee} & n_{e2_z}\boldsymbol{\Phi}_{es} \\ 0 & t_{e_x} & 0 & 0 & t_{e_y} & 0 & 0 & t_{e_z} & 0 \\ n_{s_x}\boldsymbol{\Phi}_{sm} & n_{s_x}\boldsymbol{\Phi}_{se} & n_{s_x}\boldsymbol{\Phi}_{ss} & n_{s_y}\boldsymbol{\Phi}_{sm} & n_{s_y}\boldsymbol{\Phi}_{se} & n_{s_y}\boldsymbol{\Phi}_{ss} & n_{s_z}\boldsymbol{\Phi}_{sm} & n_{s_z}\boldsymbol{\Phi}_{se} & n_{s_z}\boldsymbol{\Phi}_{ss} \\ 0 & 0 & t_{s1_x} & 0 & 0 & t_{s1_y} & 0 & 0 & t_{s1_z} \\ 0 & 0 & t_{s2_x} & 0 & 0 & t_{s2_y} & 0 & 0 & t_{s2_z} \end{bmatrix} \begin{bmatrix} \vec{\alpha}_{m_x} \\ \vec{\alpha}_{e_x} \\ \vec{\alpha}_{s_x} \\ \vec{\alpha}_{m_y} \\ \vec{\alpha}_{e_y} \\ \vec{\alpha}_{s_y} \\ \vec{\alpha}_{m_z} \\ \vec{\alpha}_{e_z} \\ \vec{\alpha}_{s_z} \end{bmatrix} \tag{2.30}$$

Now, if this sliding method is used for periodic nodes, some adaptions have to be made. In case the periodic distance method is used to slide the periodic nodes in a periodic manner, all of the interpolation matrices in Equation 2.30 are computed using a periodic distance function. In case the periodic boundaries are rotational, this distance function will be transformed to cylindrical coordinates. Next to the transformed coordinates and distance function, this will also require a transformation of the orientations to cylindrical vectors using the transformation matrix given in Appendix B:

$$n_r = cos(\theta) \cdot n_x + sin(\theta) \cdot n_y \quad (2.31)$$

$$n_\theta = -sin(\theta) \cdot n_x + cos(\theta) \cdot n_y \quad (2.32)$$

$$n_z = n_z \quad (2.33)$$

$$t_r = cos(\theta) \cdot t_x + sin(\theta) \cdot t_y \quad (2.34)$$

$$t_\theta = -sin(\theta) \cdot t_x + cos(\theta) \cdot t_y \quad (2.35)$$

$$t_z = t_z \quad (2.36)$$

With all components of the *total* interpolation matrix transformed, the resulting interpolation coefficients will also be in cylindrical coordinates. Hence, the matrix system for the evaluation step will change to:

$$\begin{bmatrix} \vec{d}_{m_r} \\ \vec{d}_{m_\theta} \\ \vec{d}_{m_z} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boldsymbol{\Phi}_{mm} & \boldsymbol{\Phi}_{me} & \boldsymbol{\Phi}_{ms} \\ n_{e1_r}\boldsymbol{\Phi}_{em} & n_{e1_r}\boldsymbol{\Phi}_{ee} & n_{e1_r}\boldsymbol{\Phi}_{es} & n_{e1_\theta}\boldsymbol{\Phi}_{em} & n_{e1_\theta}\boldsymbol{\Phi}_{ee} & n_{e1_\theta}\boldsymbol{\Phi}_{es} & n_{e1_z}\boldsymbol{\Phi}_{em} & n_{e1_z}\boldsymbol{\Phi}_{ee} & n_{e1_z}\boldsymbol{\Phi}_{es} \\ n_{e2_r}\boldsymbol{\Phi}_{em} & n_{e2_r}\boldsymbol{\Phi}_{ee} & n_{e2_r}\boldsymbol{\Phi}_{es} & n_{e2_\theta}\boldsymbol{\Phi}_{em} & n_{e2_\theta}\boldsymbol{\Phi}_{ee} & n_{e2_\theta}\boldsymbol{\Phi}_{es} & n_{e2_z}\boldsymbol{\Phi}_{em} & n_{e2_z}\boldsymbol{\Phi}_{ee} & n_{e2_z}\boldsymbol{\Phi}_{es} \\ 0 & t_{e_r} & 0 & 0 & t_{e_\theta} & 0 & 0 & t_{e_z} & 0 \\ n_{s_r}\boldsymbol{\Phi}_{sm} & n_{s_r}\boldsymbol{\Phi}_{se} & n_{s_r}\boldsymbol{\Phi}_{ss} & n_{s_\theta}\boldsymbol{\Phi}_{sm} & n_{s_\theta}\boldsymbol{\Phi}_{se} & n_{s_\theta}\boldsymbol{\Phi}_{ss} & n_{s_z}\boldsymbol{\Phi}_{sm} & n_{s_z}\boldsymbol{\Phi}_{se} & n_{s_z}\boldsymbol{\Phi}_{ss} \\ 0 & 0 & t_{s1_r} & 0 & 0 & t_{s1_\theta} & 0 & 0 & t_{s1_z} \\ 0 & 0 & t_{s2_r} & 0 & 0 & t_{s2_\theta} & 0 & 0 & t_{s2_z} \end{bmatrix} \begin{bmatrix} \vec{\alpha}_{m_r} \\ \vec{\alpha}_{e_r} \\ \vec{\alpha}_{s_r} \\ \vec{\alpha}_{m_\theta} \\ \vec{\alpha}_{e_\theta} \\ \vec{\alpha}_{s_\theta} \\ \vec{\alpha}_{m_z} \\ \vec{\alpha}_{e_z} \\ \vec{\alpha}_{s_z} \end{bmatrix} \quad (2.37)$$

There is also the possibility to implement the equal distance method within the direct sliding method but that has not been done yet in this thesis and will be left as a further recommendation.

2.4.3. Curvature correction

When the nodes slide along a planar surface or a straight edge, the 3D sliding method does not require a post-process correction step. The used conditions are sufficient to enforce the nodes to slide along the surface or edge. However, in case of sliding along a curved surface or edge, it is possible that the nodes move from the surface or edge when moving along the tangent direction(s). Therefore, a curvature correction is applied to guarantee that the nodes remain on the surface or edge they are sliding along. This correction is similar to the projection algorithm mentioned in Section 2.4.1.

2.5. Data reduction: greedy algorithm

Since the RBF interpolation method can become very computationally expensive for larger problems, the method is often combined with a greedy algorithm. A greedy algorithm makes a reduced selection of control nodes from the boundary nodes that allows for an accurate representation of the interpolation function within a set tolerance. The greedy algorithm that is used in this project is referred to as 'greedy algorithm 2: full point algorithm' by Rendall and Allen [28]. This means that for every new control node added to the selection, the interpolation coefficients are recomputed.

The algorithm starts by selecting one moving node, one surface node and one edge node as the initial selection of control nodes. The first selected moving node is the node with the largest displacement. The first surface node is the node located the furthest from the selected moving node and the initial edge node is the node located furthest from both the initial moving and surface nodes. This allows for a proper distribution of the initial control nodes over the domain.

Next, in the case of rotational periodic boundaries, the known displacements, the node coordinates and the orientation vectors are all transformed to the cylindrical coordinate system. With the initial control node selection, the interpolation coefficients are computed and the other non-selected boundary nodes are updated using the obtained interpolation function. Then the error is evaluated. The error for the moving nodes ϵ_m is the difference between the interpolated distance $\vec{d}_m$ and the actual known displacement $\vec{d}_{exact}$. Several error descriptions are available [29] but the surface error is an appropriate choice if the actual boundary displacement is known and will give the most accurate results:

$$\vec{\epsilon}_m = \left\| \vec{d}_m - \vec{d}_{exact} \right\| \quad (2.38)$$

The error for the surface nodes ϵ_s is based on the requirement for the surface nodes to have zero displacement in normal direction to the surface:

$$\vec{\epsilon}_s = \vec{n}_s \cdot \vec{d}_s \quad (2.39)$$

In a similar way, the edge nodes are not allowed to displace in either normal directions to the edge. For each normal direction, n_{e_1} and n_{e_2}, a separate error is computed, ϵ_{e_1} and ϵ_{e_2}, respectively. Then, one total error for the edge nodes ϵ_e is computed by taking the norm.

$$\begin{aligned} \vec{\epsilon}_{e_1} &= \vec{n}_{e_1} \cdot \vec{d}_e \\ \vec{\epsilon}_{e_2} &= \vec{n}_{e_2} \cdot \vec{d}_e \end{aligned} \tag{2.40}$$

$$\vec{\epsilon}_e = \sqrt{\epsilon_{e_1}^2 + \epsilon_{e_2}^2} \tag{2.41}$$

The node that has the largest error over all three error vectors combined is then selected as control node and added to its specific list depending on the type of point it is: moving, sliding surface or sliding edge. In case of rotational periodic boundaries, the interpolated displacements are first transformed back to the Cartesian coordinate system before the errors are computed. This is done because in the cylindrical reference frame, r and z are dependent on the domain size but θ is not. With the way the error is computed, this independence might cause some inaccuracies.

Every time a node is added to the control nodes, the interpolation function is recomputed, hence the inverse of the interpolation matrix is computed every time. This is continued until the error is below a set tolerance value. Since typically not all moving nodes are selected and thus not all are used for the displacement of the sliding nodes (unless the tolerance was not reached before all nodes were selected), a correction is required for the interpolation error that this will cause. This correction is explained in the next section.

2.5.1. Greedy correction

The greedy correction is based on the difference between the displacements obtained with the interpolation function and the exact known displacements of the moving nodes, d_{exact}. This difference is called the error ϵ:

$$\vec{\epsilon} = \vec{d}_m - \vec{d}_{exact} \tag{2.42}$$

Each moving node has its specific error, which is zero for the nodes selected as control nodes. Then for every sliding surface and sliding edge node, the nearest moving node is selected and a correction is applied that is proportional to the error of that node. The correction is equal to the error of the nearest neighbour boundary node $\vec{\epsilon}_{nn}$ with known displacement multiplied by the correction factor β:

$$\vec{x}_{s_{new}} = \vec{x}_s + \beta \cdot \vec{\epsilon}_{nn} \tag{2.43}$$

$$\vec{x}_{e_{new}} = \vec{x}_e + \beta \cdot \vec{\epsilon}_{nn} \tag{2.44}$$

2.6. Mesh quality

Since the thesis project aims to investigate the performance of different mesh deformation methods for periodic boundaries, a mesh quality metric is essential. The algebraic expression metrics by Knupp [18] are chosen as they have advantages over other metrics, including non-dimensionality, a finite range of metric values and a specified metric domain. The metrics are geometry specific and derived from the Jacobian matrix, containing information about the orientation, size, shape and skewness of a mesh cell. They are available for both structured and unstructured meshes and in two and three dimensions.

In this project, the most important metrics are size and skew. In small clearance gaps, the cells can become heavily skewed after deformation. To check the influence of applying the sliding technique in these areas, the skew metric is required. Moreover, the sliding technique can also change the size of the cells when clustering nodes, making the size metric also important. The size metric is also important for the displacement of the periodic boundaries to check that the displacement has improved some of the compressed and stretched mesh cells after deformation.
All metrics are relative and thus determine the cell quality by comparison to a reference element which is the initial cell. A metric value of 1 represents an ideal cell and 0 represents a degenerate cell.

The Jacobian matrix A_k, listed for the different cell types in Equation 2.45, has the size $d \times d$ for

d-dimensional cells and differs for every node k of the cell. Each matrix column represents the edge vectors emerging from a cell node. In Figure 2.15, the different cell types are shown with the cell nodes numbered.

$$\begin{aligned}
Triangular: \quad & A_k = \begin{pmatrix} x_{k+1}-x_k & x_{k+2}-x_k \\ y_{k+1}-y_k & y_{k+2}-y_k \end{pmatrix} \\
Tetrahedral: \quad & A_k = (-1)^k \begin{pmatrix} x_{k+1}-x_k & x_{k+2}-x_k & x_{k+3}-x_k \\ y_{k+1}-y_k & y_{k+2}-y_k & y_{k+3}-y_k \\ z_{k+1}-z_k & z_{k+2}-z_k & z_{k+3}-z_k \end{pmatrix} \\
Quadrilateral: \quad & A_k = \begin{pmatrix} x_{k+1}-x_k & x_{k+3}-x_k \\ y_{k+1}-y_k & y_{k+3}-y_k \end{pmatrix} \\
Hexahedral: \quad & A_k = \begin{pmatrix} x_{k+1}-x_k & x_{k+3}-x_k & x_{k+4}-x_k \\ y_{k+1}-y_k & y_{k+3}-y_k & y_{k+4}-y_k \\ z_{k+1}-z_k & z_{k+3}-z_k & z_{k+4}-z_k \end{pmatrix}
\end{aligned} \tag{2.45}$$

The determinant of the Jacobian is α_k. For every node k, a symmetric metric tensor is found by forming $A_k^T A_k$ which has components λ_{ij}^k for $i,j = 1,..,n$. These components are used in the relative metric formulations. In Table 2.3, the relative skew metrics are given for different mesh cell types: triangular, tetrahedral, quadrilateral and hexahedral. There is no pure skew metric for the triangular and tethrahedral cells as skewing also changes their shape. Next to the mesh cell type it is shown for which mesh they are used in this project. Hence, the relative shape metric can be used as given by Knupp [18].

	f_{skew}
Triangular (2D, unstructured)	$\frac{\sqrt{3}\alpha}{\lambda_{11}+\lambda_{22}-\lambda_{12}}$
Tetrahedral (3D, unstructured)	$\frac{3(\alpha\sqrt{2})^{2/3}}{\frac{3}{2}(\lambda_{11}+\lambda_{22}+\lambda_{33})-(\lambda_{12}+\lambda_{23}+\lambda_{13})}$
Quadrilateral (2D, structured)	$\frac{4}{\sum_{k=0}^{3}\left(\sqrt{\lambda_{11}^k\lambda_{22}^k}\right)/(\alpha_k)}$
Hexahedral (3D, structured)	$\frac{8}{\sum_{k=0}^{7}\left(\left(\sqrt{\lambda_{11}^k\lambda_{22}^k\lambda_{33}^k}\right)/(\alpha_k)\right)^{2/3}}$

Table 2.3: Relative skew metrics [18].

The relative size metric is given by

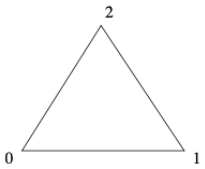
$$f_{\text{size}} = \min(\tau, 1/\tau) \tag{2.46}$$

where τ is the ratio of the deformed cell over the initial cell volume: $\tau = \sum_{k=0}^{n-1} \alpha_k/\alpha_k^0$, for a total number of nodes n.

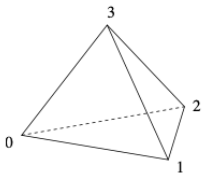
The skew and size metrics can be combined into one by using the expression:

$$f_{\text{size - skew}} = \sqrt{f_{\text{size}}}\, f_{\text{skew}} \tag{2.47}$$

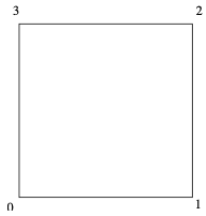
The mesh quality of the 3D meshes is assessed using the mesh quality filter in Paraview. All 3D meshes used in this report have hexahedral cells for which the skew metric is used. The skew metric determines the skewness which is a measure of the difference between the mesh cell shape and a cell of the same volume with equilateral edges. The skewness has a value between 0 and 1. The closer the skewness value gets to 1, the more skewed the mesh is and the lower the mesh quality. Hence, this is opposite of the way the quality metrics work that are used in the 2D cases.



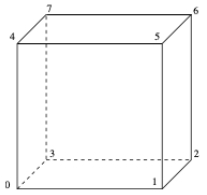
(a) Triangular cell (2D).



(b) Tetrahedral cell (3D).



(c) Quadrilateral cell (2D).



(d) Hexahedral cell (3D).

Figure 2.15: Different mesh cell types with numbered nodes [35].

3

Two-dimensional results

In this chapter, the two-dimensional results are presented. They show a comparison of four different deformation methods. These 4 methods are regular RBF, periodic sliding, periodic displacement with corners fixed and periodic displacement with corners allowed to move. The method algorithms are shown in Appendix A. For these methods, there is also a comparison between the two periodic boundary methods, the periodic distance and the equal displacement method, as mentioned in Section 2.3. The chapter starts with an overview of the different meshes used in the simulations in Section 3.1. Next, the simulation settings are listed in Section 3.2. Then, the results for the different methods are presented and analysed in Section 3.3 for the translational cases and in Section 3.4 for the rotational cases. The greedy algorithm results are discussed in Section 3.5 and finally, some conclusions are made in Section 3.6.

3.1. Two-dimensional meshes

For the two-dimensional simulations, two different meshes are used that represent the two types of periodic boundaries as discussed in Section 2.2. Mesh 1 is a structured mesh that is used for the translational periodic boundaries. Mesh 2 is an unstructured mesh used for the rotational periodic boundaries.

3.1.1. Mesh 1: Structured square mesh

Mesh 1 (Fig. 3.1) is a structured square mesh and is used to examine the performance of the methods for translational periodic boundaries. The upper and lower boundaries are considered to be periodic, the left and right boundary are not. The mesh has an inner block that undergoes a set displacement. The figure also presents the initial mesh quality before a deformation is applied to the block. The mesh is completely bright yellow, meaning an optimal quality of 1 as shown on the colorbar on the right. The details for the mesh are shown in Table 3.1.

Mesh 1: Structured square mesh	
Minimum quality	1
Mean quality	1
Length	25 units
Height	25 units
Inner block length	5 units
Inner block height	1 unit
Spacing	1 unit

Table 3.1: Mesh 1 properties.

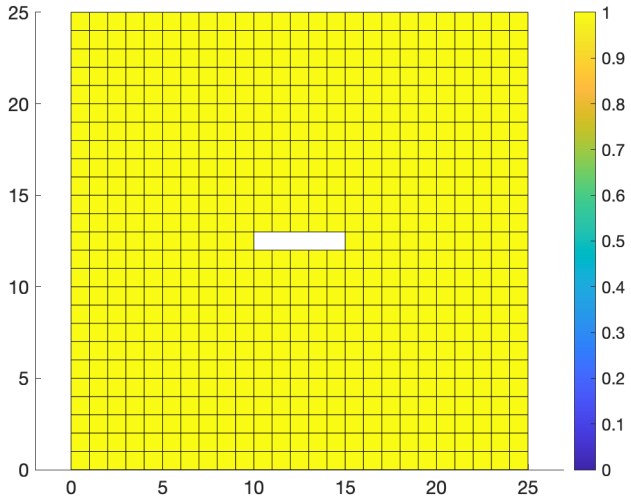


Figure 3.1: Mesh 1: structured square mesh

3.1.2. Mesh 2: Unstructured pie shape mesh

Mesh 2 (Fig. 3.2) is an unstructured pie shape mesh and is used to examine the performance of the methods for rotational periodic boundaries. The two straight boundaries are periodic and enclose an angle of 45°. The arc is a non-periodic boundary. The mesh contains an inner block that has a set displacement, different from the block displacement in mesh 1. The mesh does not have an optimal mesh quality before deformation as can be concluded from the darker yellow shades of some of the mesh cells. The specifications of mesh 2 are listed in Table 3.2.

Mesh 2: Unstructured pie shape mesh	
Minimum quality	0.8764
Mean quality	0.9823
Radius of arc	25 units
Periodic angle	45°
Inner block length	5 units
Inner block height	1 unit

Table 3.2: Mesh 2 properties.

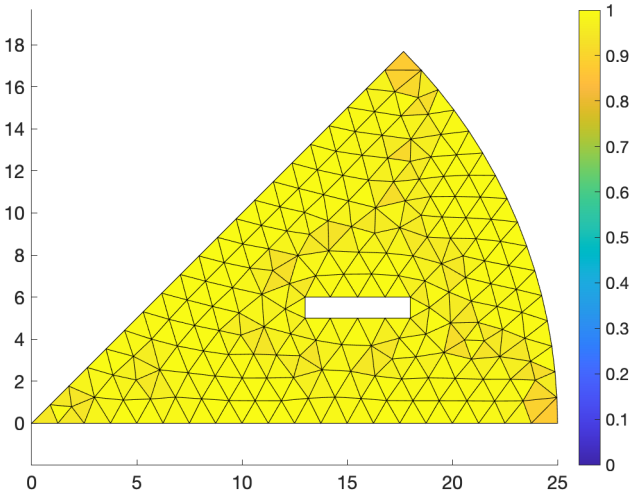


Figure 3.2: Mesh 2: unstructured pie shape mesh.

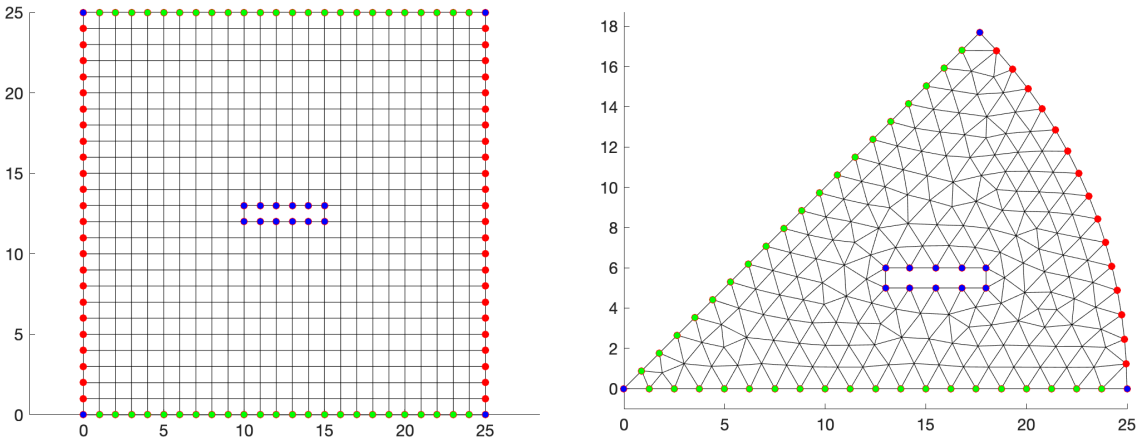
3.2. Simulation settings

The nodes that make up the different meshes can be divided into several groups. In Figure 3.3 the different node groups are indicated for both meshes. The green nodes are the periodic nodes, hence they lie on the periodic boundaries and will always have identical displacement. The red nodes are the non-periodic nodes and the blue nodes are the moving nodes for which the displacement is known. The displacement of the inner block is given in Table 4.4 for both meshes. Moving nodes can also have a displacement equal to zero, making them fixed. The rest of the nodes (not coloured) are the internal nodes. Depending on the deformation method used, different node assignments are valid for the different node groups:

- **Regular RBF:** All boundary nodes are fixed. Hence, the green, red and blue nodes on the corners are all fixed. The blue nodes on the inner block are moving nodes. The fixed nodes are also seen as moving nodes with zero displacement.
- **Periodic Sliding:** The green nodes are allowed to slide, in a periodic way. The red nodes are also sliding nodes but not periodic. The corner blue nodes are fixed and the inner block blue nodes are moving nodes.
- **Periodic displacement (fixed corners):** The green nodes are allowed to be displaced in a periodic manner. The red nodes are sliding nodes in a non-periodic manner. The blue corner nodes are fixed and the inner block blue nodes are moving nodes.
- **Periodic displacement (moving corners):** (Only for mesh 1) The green nodes are allowed to be displaced periodically. The red nodes are sliding nodes as well as the blue corner nodes. The blue sliding corner nodes slide periodically. The inner blue nodes are moving nodes.

Block deformation		
	Mesh 1	Mesh 2
Rotation	60°	30°
Displacement in x	-5 units	-2 units
Displacement in y	-8 units	-3 units

Table 3.3: Inner block deformation details for displacement of periodic boundaries.



(a) Mesh 1 for translational periodic boundaries.

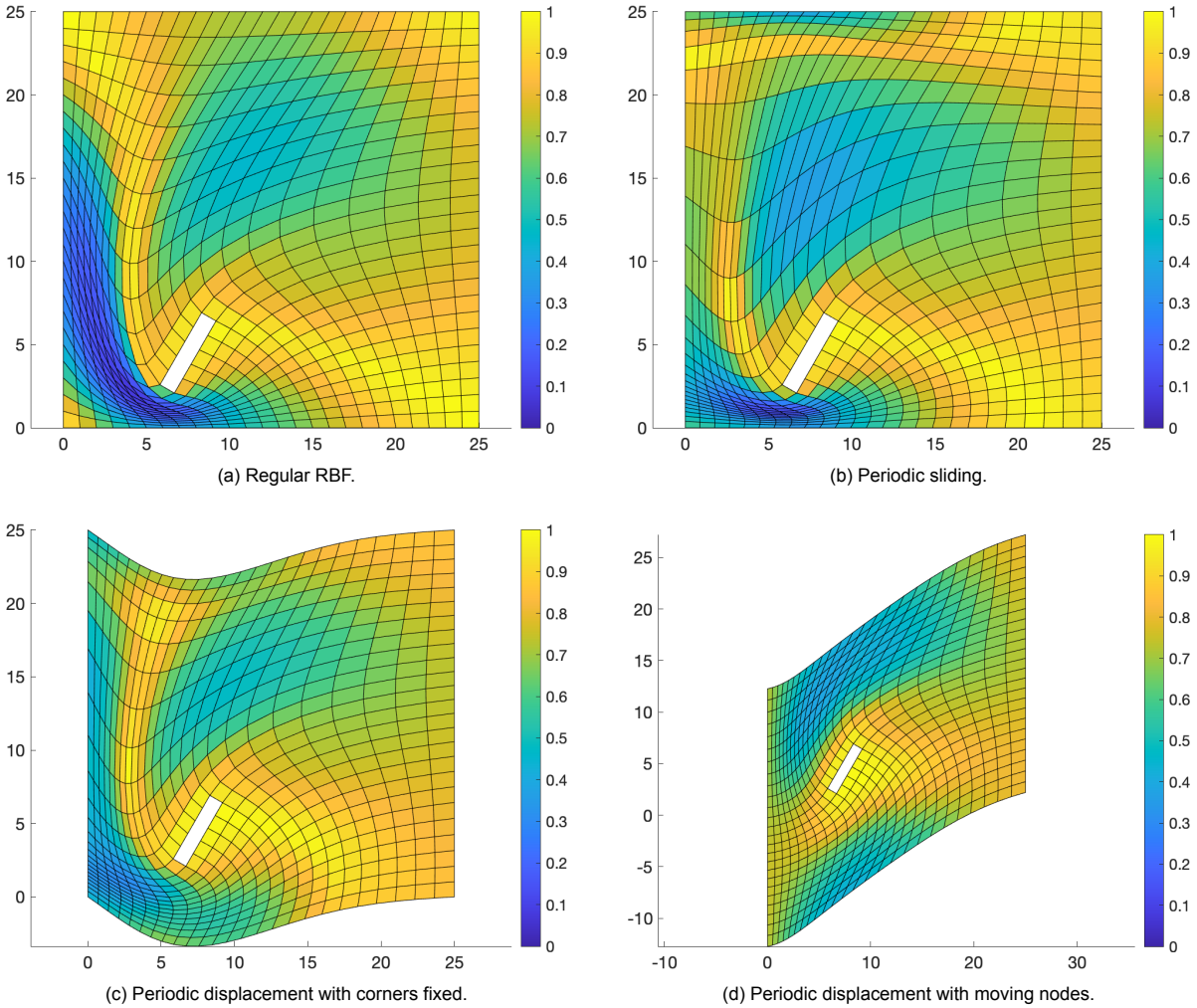
(b) Mesh 2 for rotational periodic boundaries.

Figure 3.3: Node assignment for simulations with periodic nodes (green), moving nodes (including static nodes, blue) and non-periodic boundary nodes (red).

Furthermore, the support radius r is 62.5 for both meshes and the deformation is performed in 20 steps.

3.3. Translational periodic boundaries results

In this section, the results for the translational periodic boundaries are discussed. The deformed meshes using the periodic distance method are shown in Figure 3.4 and for the equal displacement method in Figure 3.5. Using the regular RBF method causes a very poor quality region in the left bottom part of the mesh. When comparing the regular RBF (Fig. 3.4a) and the periodic sliding method (Fig. 3.4b), it can be observed that allowing the nodes to slide already reduces the size of the very poor quality cells region on the left side of the mesh. This region now seems to be concentrated at the bottom of the mesh, in the small region between the moving inner block and the lower periodic boundary. The periodic sliding method comes with a slight decrease in mesh quality at the top left corner. This is because the displacements of the non-periodic sliding nodes are also computed using the periodic distance function. Allowing the periodic nodes to move (Fig. 3.4c) further improves the minimum mesh quality of the mesh. The mesh quality at the bottom of the mesh has now improved a large amount. The lowest quality region is now located around the left bottom corner, which is a fixed node. When the corner nodes are allowed to slide vertically as well (Fig. 3.4d) this poor region disappears as well and a relatively good quality mesh is achieved. The lowest quality regions are now located close to the periodic boundaries.



(a) Regular RBF.

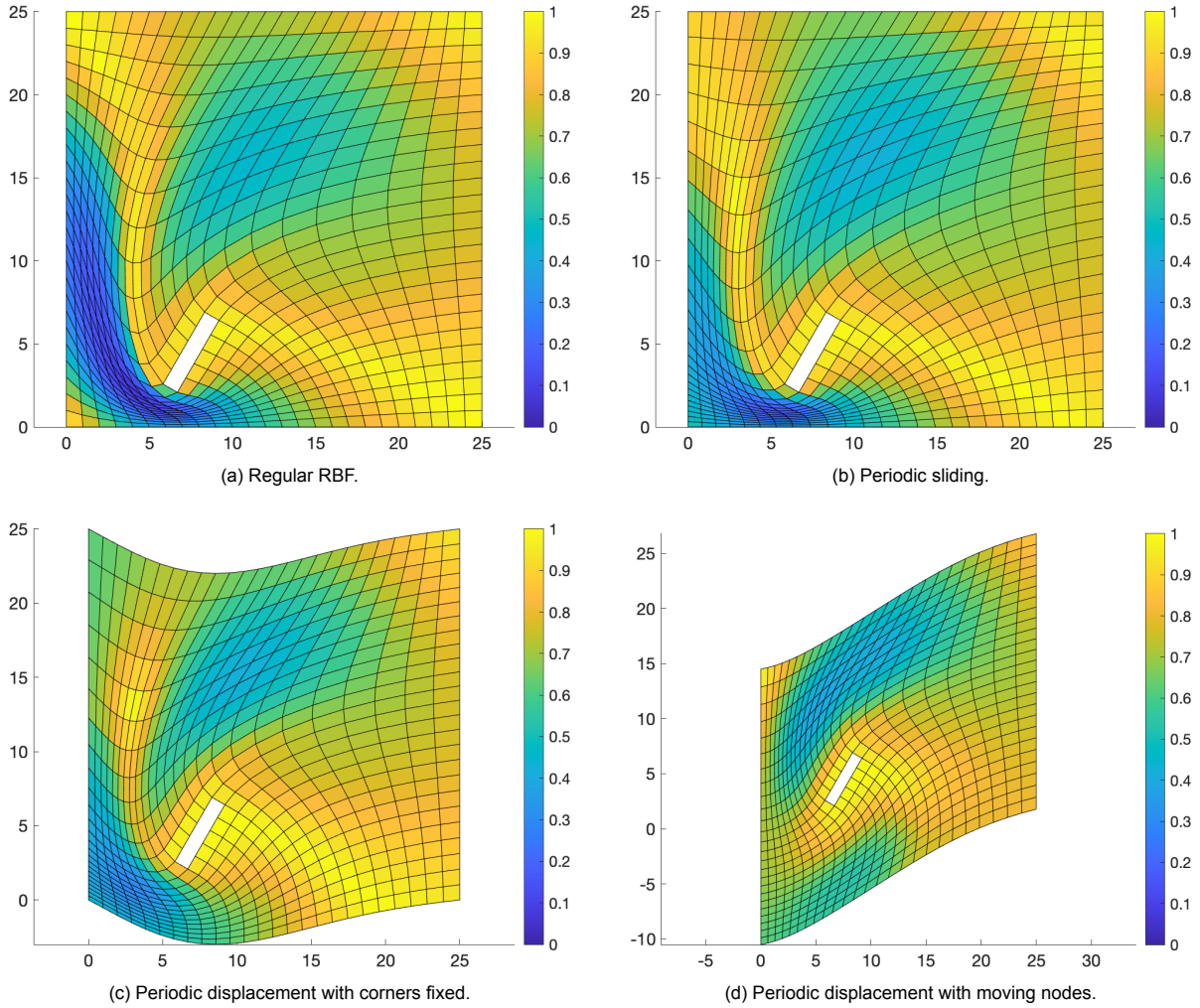
(b) Periodic sliding.

(c) Periodic displacement with corners fixed.

(d) Periodic displacement with moving nodes.

Figure 3.4: Different deformation methods performed on mesh 1 using the periodic distance method with Wendland C^2 (r=62.5) after 20 deformation steps.

Similar conclusions can be made by investigating the meshes in Figure 3.5 for the equal displacement method. However, also some differences can be noticed between the two periodic boundary meth-



(a) Regular RBF.

(b) Periodic sliding.

(c) Periodic displacement with corners fixed.

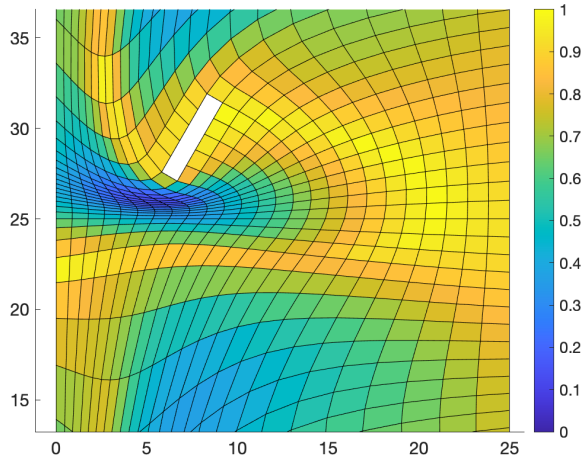
(d) Periodic displacement with moving nodes.

Figure 3.5: Different deformation methods performed on mesh 1 using the equal displacement method with Wendland C^2 (r=62.5) after 20 deformation steps.

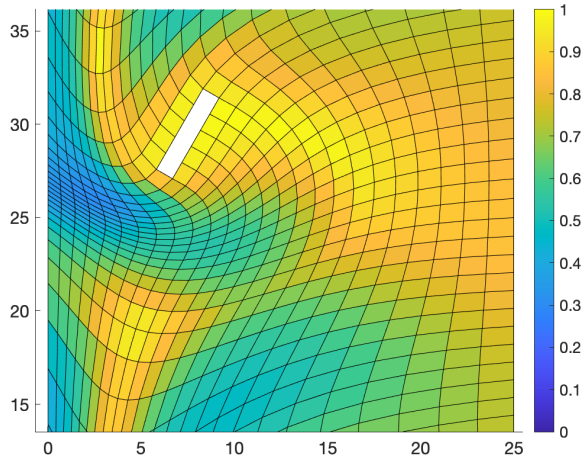
ods. For the equal displacement method, the periodic sliding in Figure 3.5b gives a much better mesh quality at the left top corner than could be seen for the periodic distance method in Figure 3.4b. This is because only the regular distance function is used in the equal displacement method. The rest of the mesh is more or less unaffected by the periodic boundary implementation. On the other hand, the method seems to yield a slightly bigger lower mesh quality region on the left and underneath the inner block. When applying the periodic displacement (Fig. 3.5c), the mesh quality is more or less similar to the mesh obtained with the periodic distance method. Finally, when the corners are allowed to move (Fig. 3.5d) the trend also seems very similar to the periodic distance method with a slightly higher overall mesh quality.

The two periodic methods can be further compared by investigating Figures 3.6 and 3.7 for the periodic distance and equal displacement method, respectively. In these figures, two identical, deformed meshes are stacked upon one another where the focus is put on the interface between the two meshes, located at 25 units in vertical direction. It is clear that both periodic methods ensure a periodic displacement of the boundaries, as the nodes on the upper and lower boundaries coincide. For the periodic distance method, it is difficult to point out where one mesh ends and another begins as this method allows for a smooth transition between the two. This is also the reason for the lower mesh quality region at the upper left corner that is clearly visible in Figure 3.4b. The mesh in this region starts to adapt to the cell sizes at the bottom of the mesh. With the equal displacement method, this phenomenon does not occur. Here, the transition from one mesh to the other is not smooth as the cell sizes and shapes

close to the interface can differ a lot. This makes the interface between the two meshes more clearly visible with some obvious kinks present.

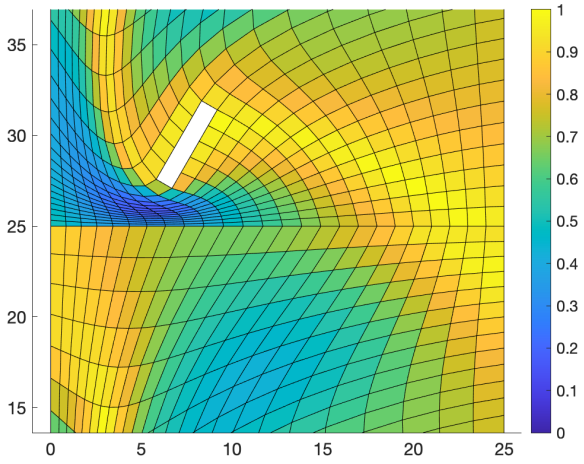


(a) Periodic sliding.

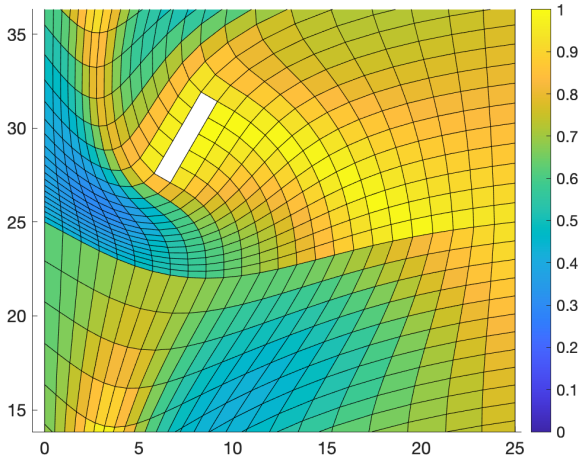


(b) Periodic displacement with fixed vertices.

Figure 3.6: Interface between two adjacent meshes after applying periodic distance method.



(a) Periodic sliding.



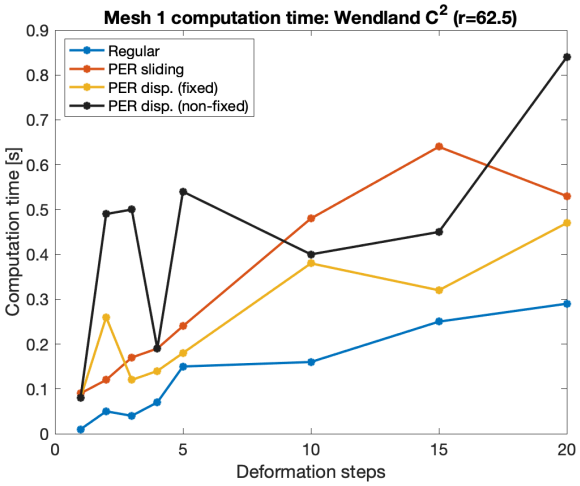
(b) Periodic displacement with fixed vertices.

Figure 3.7: Interface between two adjacent meshes after applying equal displacement method.

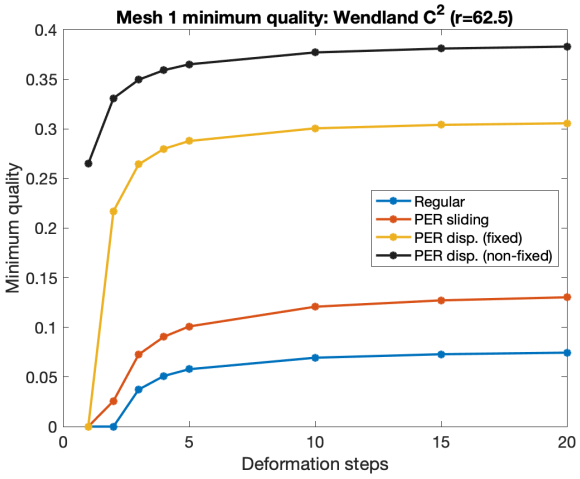
The performance of the different methods is illustrated by performance graphs of the computation time, minimum quality and mean quality for different numbers of deformation steps. These are shown in Figure 3.8 for the periodic displacement method and in Figure 3.9 for the equal displacement method. When comparing the computation times for both periodic methods, it can be observed that the periodic distance method is faster than the equal displacement method. This is because the equal displacement method has a larger interpolation matrix that needs to be inverted. Some fluctuations can also be seen in the computation time graph in Figure 3.8a. These mainly occur for lower number of deformation steps which could be attributed to overhead and sometimes are just random. Furthermore, it makes sense that the regular RBF method has the lowest computation time as all other methods contain an extra projection step for the sliding nodes.

In terms of mesh quality, the periodic displacement method with moving corners is the best performing both for minimum and mean quality. Only the mean quality for the periodic distance method seems to be most optimal when keeping the corners fixed. This can be due to the higher skewness of the cells at the top and bottom when the corners are allowed to move. Furthermore, it is clear that allowing the periodic boundaries to move, for the specific deformation used in this case, improves the mesh quality compared to keeping them fixed. This is also the case if the periodic nodes are only allowed to slide.

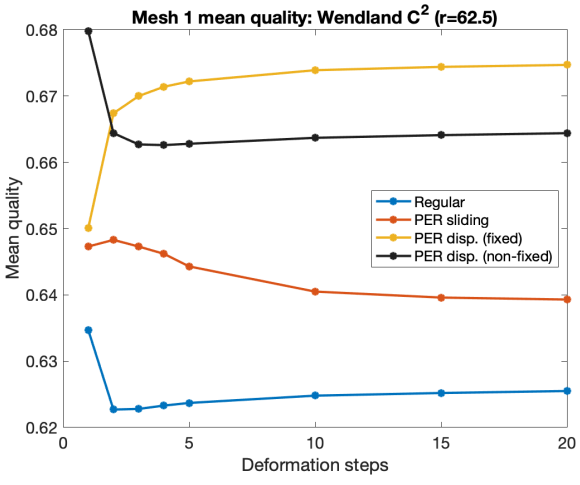
When comparing the periodic distance and the equal displacement method, it is difficult to distinguish which one performs better in terms of mesh quality as they show very similar results.



(a) Computation time.

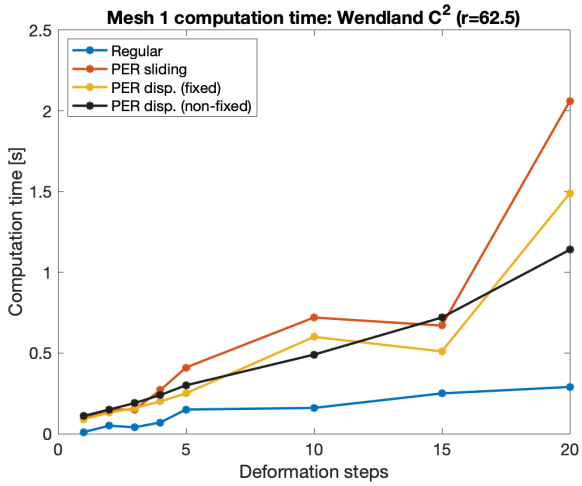


(b) Minimum mesh quality.

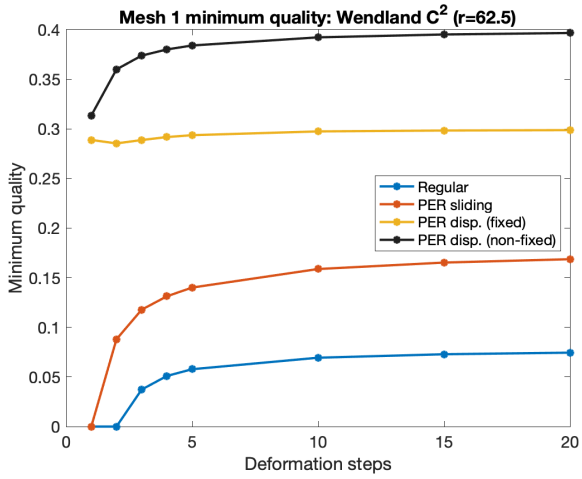


(c) Mean mesh quality.

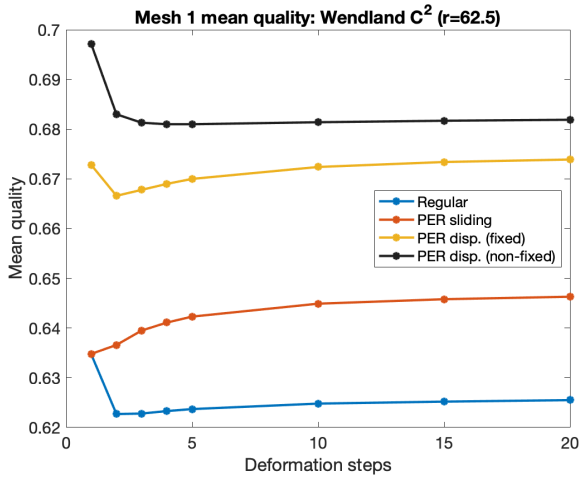
Figure 3.8: Performance of different deformation methods using the periodic distance method and the Wendland C^2 (r=62.5) RBF function on translational periodic boundaries.



(a) Computation time.



(b) Minimum mesh quality.

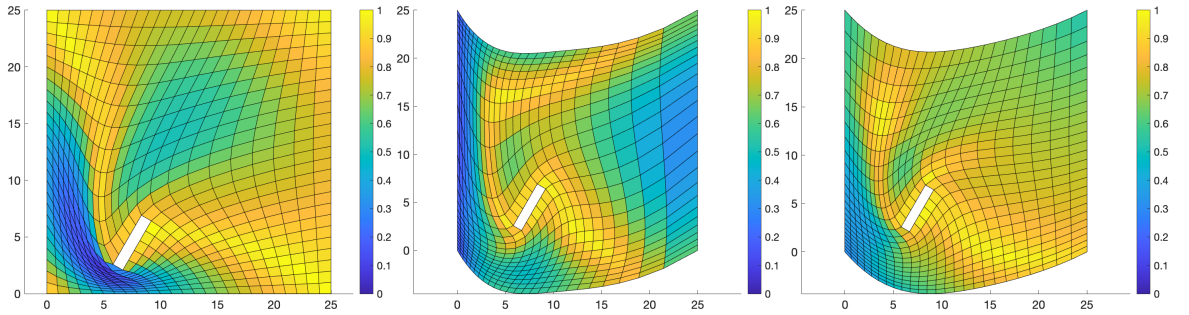


(c) Mean mesh quality.

Figure 3.9: Performance of different deformation methods using the equal displacement method and the Wendland C^2 (r=62.5) RBF function on translational periodic boundaries.

3.3.1. Influence of radial basis function

A second RBF, the unscaled TPS function, is used to examine how the periodic boundary methods perform for a different type of RBF. Only the periodic displacement method with fixed corner nodes is applied. Figure 3.10 shows the deformed meshes using the TPS function. One aspect that is clearly different from the scaled Wendland function results is the sharp edges at the corners that result from using the TPS function. They are present for both periodic methods but in a lesser extent for the equal displacement method. For the periodic distance method, the very sharp corners create a new poor cell quality region. These sharp corners are also reflected in the minimum quality graph (Fig. 3.11b) since the method performs significantly worse than the equal displacement method. When investigating the mean mesh quality in Figure 3.11c, the method even performs worse than the regular RBF method. Hence, for the TPS function it is clear that the equal distance method is the better performing one. Such clear differences in quality between the two periodic methods were not present when using the Wendland function. One thing that is similar compared to the Wendland RBF is the computation time that is still higher for the equal displacement method.
A possible solution for the very sharp corners with the periodic distance method could be to scale the distance function (Eq. 2.11) by changing its amplitude.

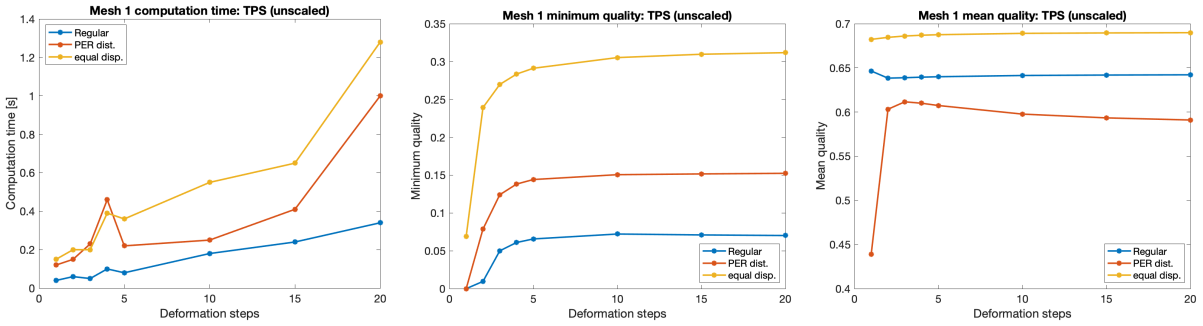


(a) Regular RBF.

(b) Periodic displacement with corners fixed using periodic distance method.

(c) Periodic displacement with corners fixed using equal displacement method.

Figure 3.10: Different deformation methods performed on mesh 1 using TPS (unscaled) in 20 deformation steps.



(a) Computation time.

(b) Minimum quality.

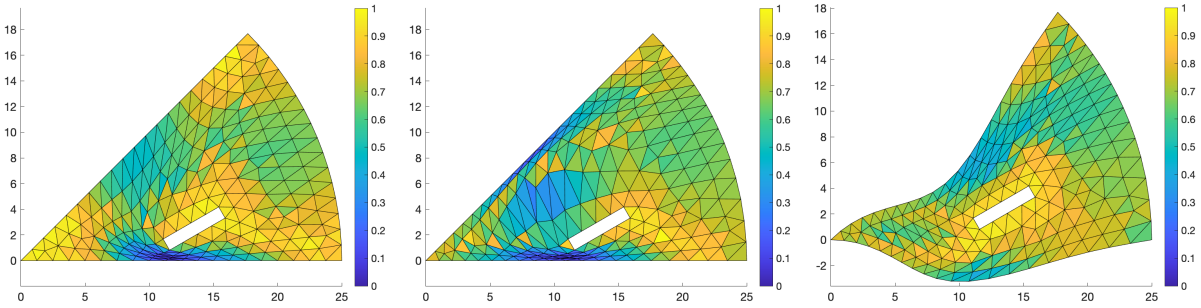
(c) Mean quality.

Figure 3.11: Performance graphs of different periodic methods applied to mesh 1 using TPS unscaled function.

3.4. Rotational periodic boundaries results

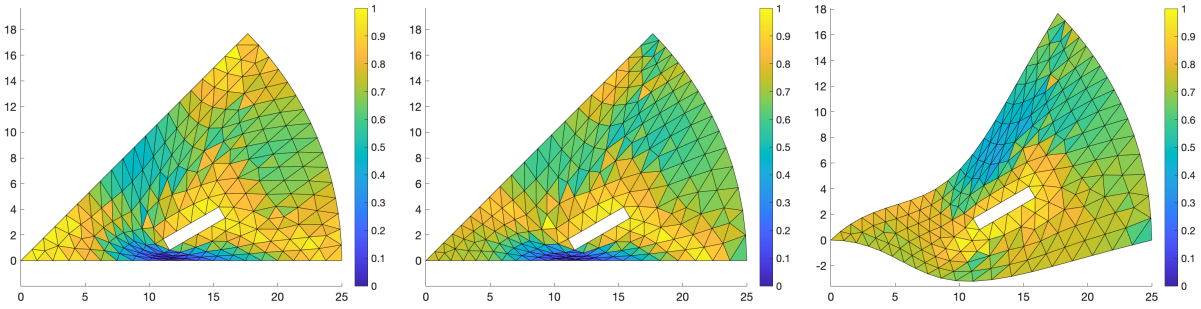
The Figures 3.12 and 3.13 show the different deformation methods applied to the rotational periodic boundaries mesh for the two different periodic boundary methods. For both methods, the regular RBF clearly produces a very poor quality region in the mesh between the inner block and the lower periodic boundary. The use of periodic sliding with the periodic distance method (Fig. 3.12b) only seems to worsen this region as well as the quality of the cells close to the upper periodic boundary. This is clearly due to the cells near the upper boundary adapting to the mesh cell size near the bottom boundary. With the equal displacement method, the periodic sliding also does not seem to offer much improvements. This is mainly because the clearance gap between the block and the lower boundary is very small. The cells are severely compressed and sliding will not help this much. Once the periodic boundaries are

allowed to move (Fig. 3.12c & 3.13c), the region below the inner block is significantly improved while the region above the block seems to reduce in quality due to the increased skewness of the cells in that location as well as the changed shape. When comparing both periodic boundary methods it also seems to be difficult to tell which one is which, purely from the mesh figures. However, when putting two identical meshes together, it becomes much more clear which periodic method is being used.



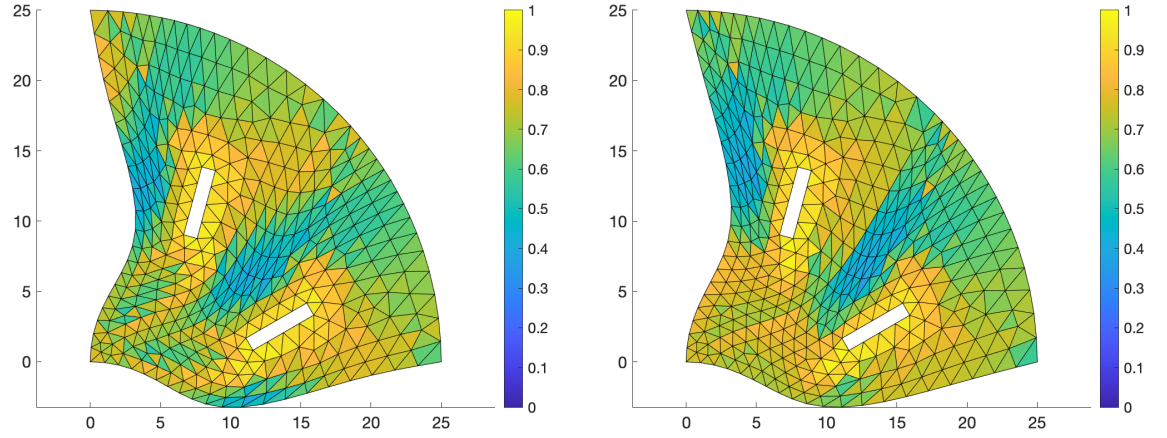
(a) Regular RBF. (b) Periodic sliding. (c) Periodic displacement with corners fixed.

Figure 3.12: Different deformation methods performed on mesh 2 using the periodic distance method with Wendland C^2 (r=62.5) after 20 deformation steps.



(a) Regular RBF. (b) Periodic sliding. (c) Periodic displacement with corners fixed.

Figure 3.13: Different deformation methods performed on mesh 2 using the equal displacement method with Wendland C^2 (r=62.5) after 20 deformation steps.

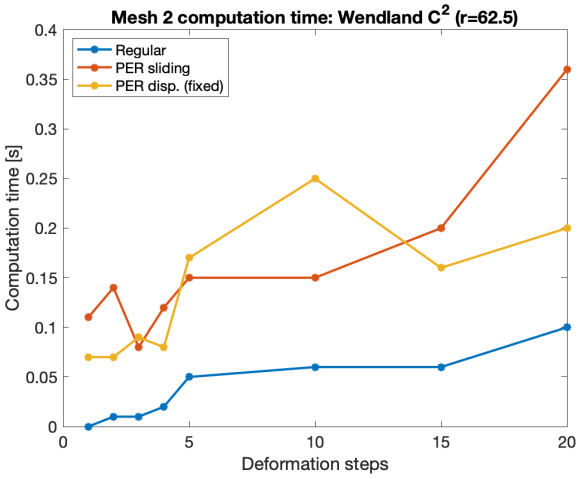


(a) Periodic displacement with fixed corners using periodic distance method. (b) Periodic displacement with fixed corners using equal displacement method.

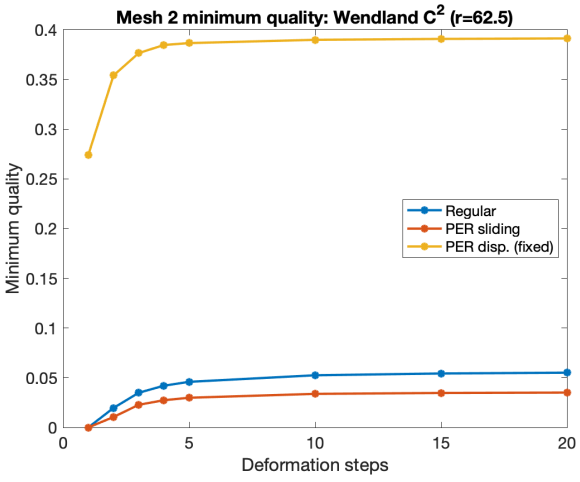
Figure 3.14: Two adjacent rotational periodic meshes using Wendland C^2 (r=62.5) RBF function after 20 deformation steps.

Figure 3.14 shows two adjacent meshes with displaced periodic boundaries. On the left, the same kind of smooth transition can be observed over the interface when using the periodic distance method, as

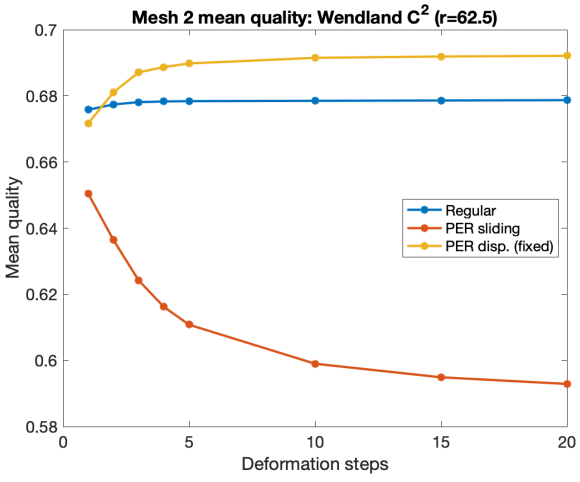
was the case for translational periodic boundaries. On the right, it is clear that there is again a difference in cell size between the upper and lower periodic boundary, prohibiting a smooth transition from one mesh to the next. Hence, it is obvious that the equal displacement has a clear drawback, irrespective of the type of periodic boundaries.



(a) Computation time.

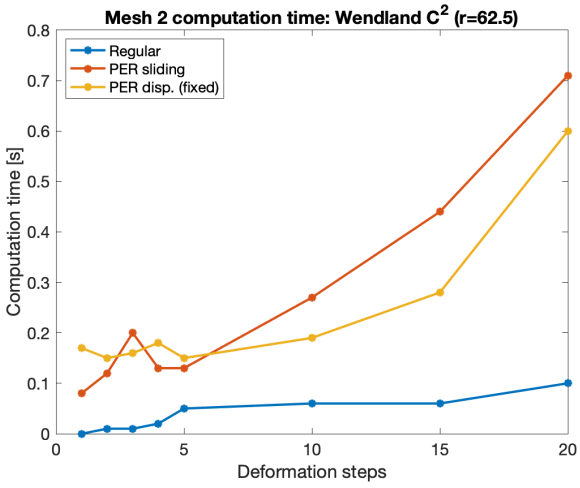


(b) Minimum mesh quality.

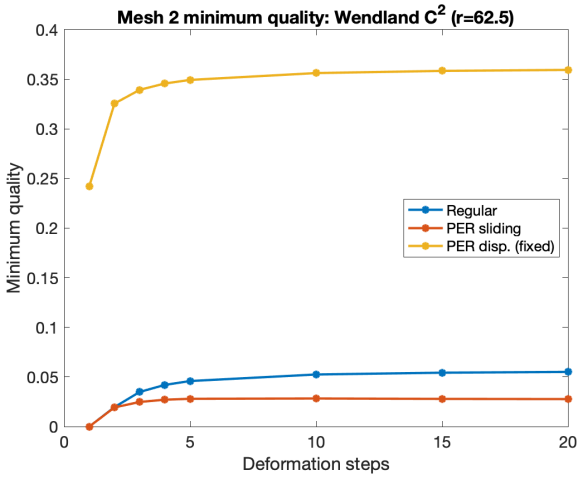


(c) Mean mesh quality.

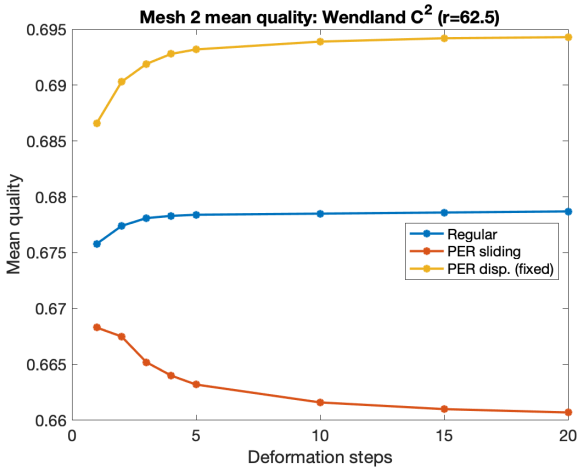
Figure 3.15: Performance of different deformation methods using the periodic distance method and the Wendland C^2 (r=62.5) RBF function on rotational periodic boundaries.



(a) Computation time.



(b) Minimum mesh quality.



(c) Mean mesh quality.

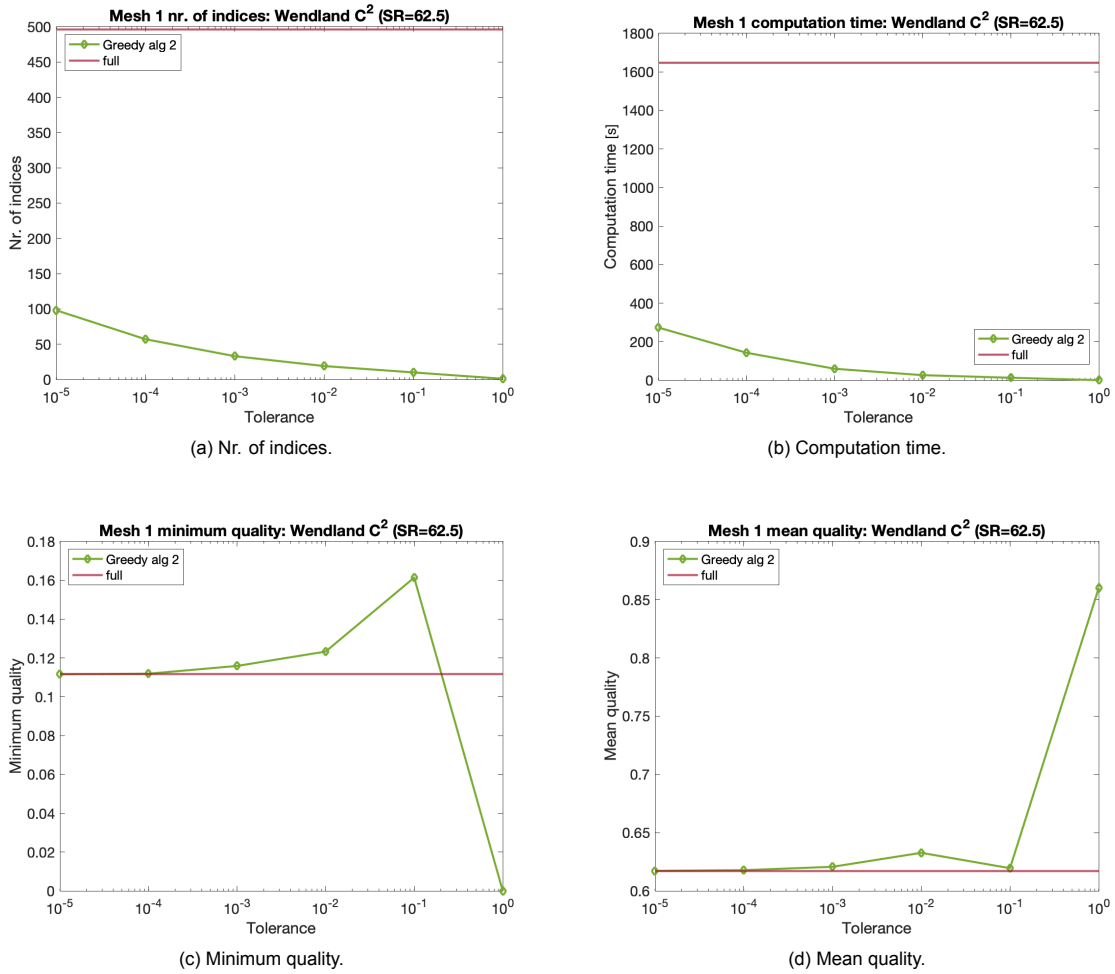
Figure 3.16: Performance of different deformation methods using the equal displacement method and the Wendland C^2 (r=62.5) RBF function on rotational periodic boundaries.

When examining the performance graphs for the different methods in Figures 3.15 and 3.16, a doubling in computation time can be noticed for the equal displacement method.
As expected, letting the periodic boundaries move shows a great improvement in the minimum quality but also improves the mean quality. However, the sliding method shows a decrease in quality for both periodic methods. Hence, the deformation for the rotational case is to severe for pure sliding to improve the mesh.
When comparing the two periodic methods, it is again difficult to distinguish which one performs better in terms of mesh quality. The minimum mesh quality seems to be slightly higher for the periodic distance method. On the other hand, the mean mesh quality is approximately the same for both periodic methods, unless periodic sliding is used in which case the equal displacement method scores better.

3.5. Greedy Results

In order to check that a periodic boundary method also performs well with the use of a greedy algorithm, a denser version of mesh 1 is used with a smaller spacing. Since the number of nodes is still relatively small, greedy algorithm 2 is used as it faster than greedy algorithm 1 for a smaller number of nodes [22].

3.5.1. Case 1: Mesh 1 with spacing 0.125, using greedy 2 algorithm



(a) Nr. of indices.

(b) Computation time.

(c) Minimum quality.

(d) Mean quality.

Figure 3.17: Greedy results on mesh 1 using periodic distance method with Wendland C^2 function (r=62.5).

The greedy case is performed on mesh 1. In order to properly investigate the performance of the periodic method using the greedy data reduction algorithm, the mesh spacing has been reduced to 0.125. The number of deformation steps is 5. The displacement that is applied to the inner block is the same

as in Table 4.4 and only the periodic nodes are allowed to move. The results for the periodic distance method are shown in Figure 3.17.

For the periodic distance method, it can be seen that for a tolerance of 10^{-5} the same minimum and mean mesh quality can be achieved using greedy as without using greedy. The greedy algorithm allows to achieve the same mesh quality with only 20% of the control nodes used and almost 6 times as fast. For the equal displacement method, a similar behavior is expected. This is something that can still be tested in the future, once the method is adapted to work with a greedy algorithm.

3.6. Discussion of results

From the periodic boundary displacement simulations it can be concluded that the TPS function does not perform well when used for the periodic distance method. When using the equal displacement method with TPS, the mesh quality was much higher but similar to the Wendland results. For that reason, only the Wendland C^2 function was used for the rest of the simulations as it performs well for both periodic methods. Although none of the periodic methods seems to be superior in terms of deformed mesh quality, the periodic displacement method does come with two advantages over the equal displacement method which are independent of the type of periodic boundaries used. Firstly, the transition between two adjacent meshes is much more smooth for the periodic distance method. The equal displacement method instead shows clear kinks at the interface. Secondly, the periodic distance method is more computationally efficient than the equal displacement method. For these reasons, the 3D test cases will only use the periodic distance method, leaving the implementation of the 3D equal displacement method as a future recommendation.
Furthermore, the most optimal mesh deformation method is when the corner nodes are allowed to move but sometimes this comes with a slight decrease in mean mesh quality.
Finally, the periodic methods can perform well when using a greedy algorithm, especially if the tolerance is 10^{-5} for both periodic boundary methods.

4

Three dimensional results

In this chapter, the 3D results are given showing a comparison between the regular RBF and the periodic displacement method. For the periodic displacement, the periodic distance method is used as is described in Section 2.3. This was shown to be computationally cheaper than the equal displacement method in the 2D test cases. It also showed a smooth transition from one mesh to another.
The chapter starts with an overview of the two types of 3D meshes the method will be tested on. Next the simulation parameters are given. Then the results are discussed for the translational case in Section 4.3 and for the rotational case in Section 4.4.

4.1. Three-dimensional meshes

The periodic boundary deformation method is tested in 3D using two different test cases. The first one is a blade within a domain that has translational periodic boundaries. The second one is a blade within a domain with rotational periodic boundaries. Both meshes consist of hexahedral cells.

4.1.1. Mesh 3: Structured mesh of blade with translational periodic domain

The third mesh is a structured 3D mesh with translational periodic boundaries in y-direction. In Figure 4.1, the initial mesh is shown before any deformation is applied. The mesh can be considered quasi 2D as the same surface is repeated multiple times along the z-axis. Therefore, only one surface is displayed in the figure. The initial mesh already starts of with some highly skewed mesh cells present.

Mesh 3: Structured mesh of blade with translational periodic domain	
Periodic distance	0.0575 units
Domain range in x	[-0.055,0.079] units
Domain range in y	[-0.076,0.027] units
Domain range in z	[0,0.1] units
Moving nodes	12248
Sliding nodes	33306
Periodic nodes	14782
Internal nodes	748524

Table 4.1: Mesh 3 properties.

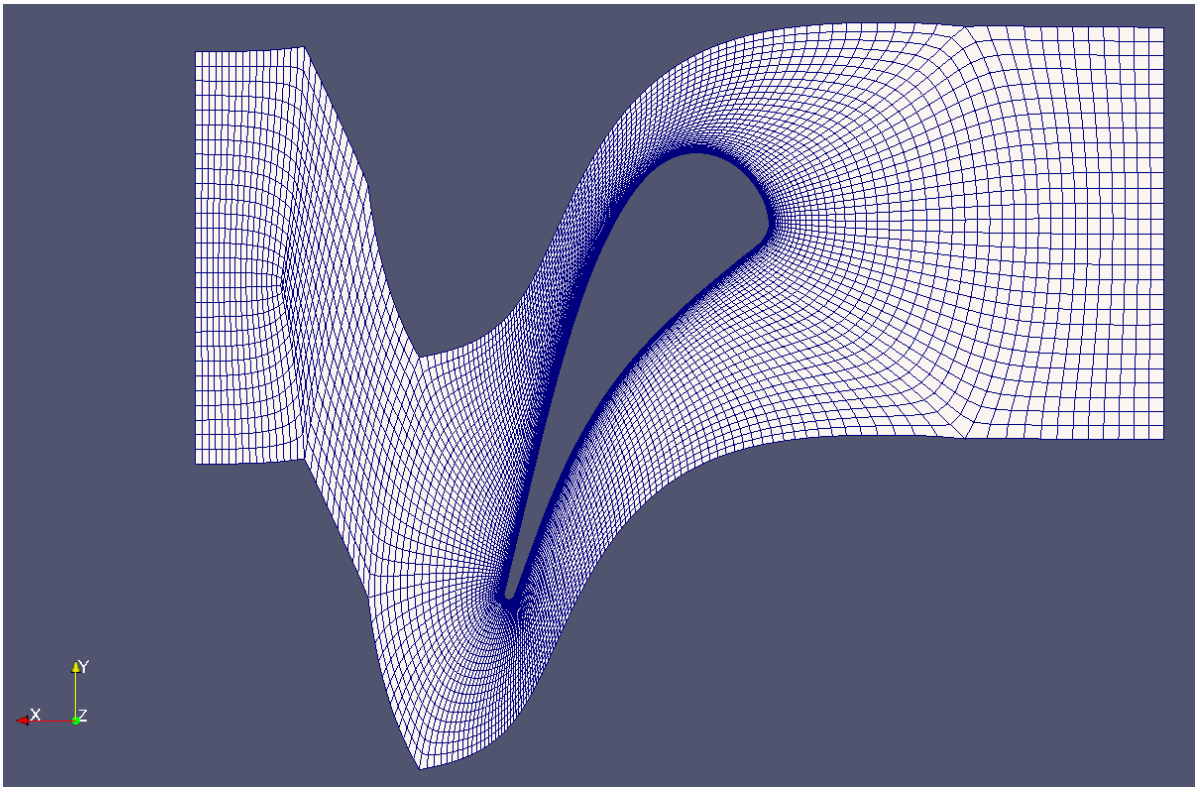


Figure 4.1: Initial mesh 3, only one surface displayed.

4.1.2. Mesh 4: Structured mesh of blade with rotational periodic domain

The fourth mesh is a structured 3D mesh with rotational periodic boundaries. The initial mesh before deformation is shown in Figure 4.2. The hub and the shroud are the outer surfaces of the mesh where the blade intersects the surface. The shroud surface, which is visible in the figure, is a type of slightly curved surface segment of a cone. The rest of the mesh is more or less a copy of the shroud surface stacked behind it but slightly decreasing in outlet edge size with the hub having the smallest outlet edge. The outlet is the surface visible on the left side of the mesh. The properties for mesh 4 can be found in Table 4.2.

Mesh 4: Structured mesh of blade with rotational periodic domain	
Periodic angle	5.0704°
Domain range in x	[0.489,0.602] units
Domain range in y	[-0.061,0.024] units
Domain range in z	[-0.075,0.054] units
Moving nodes	10808
Sliding nodes	10780
Periodic nodes	15832
Internal nodes	280820

Table 4.2: Mesh 4 properties.

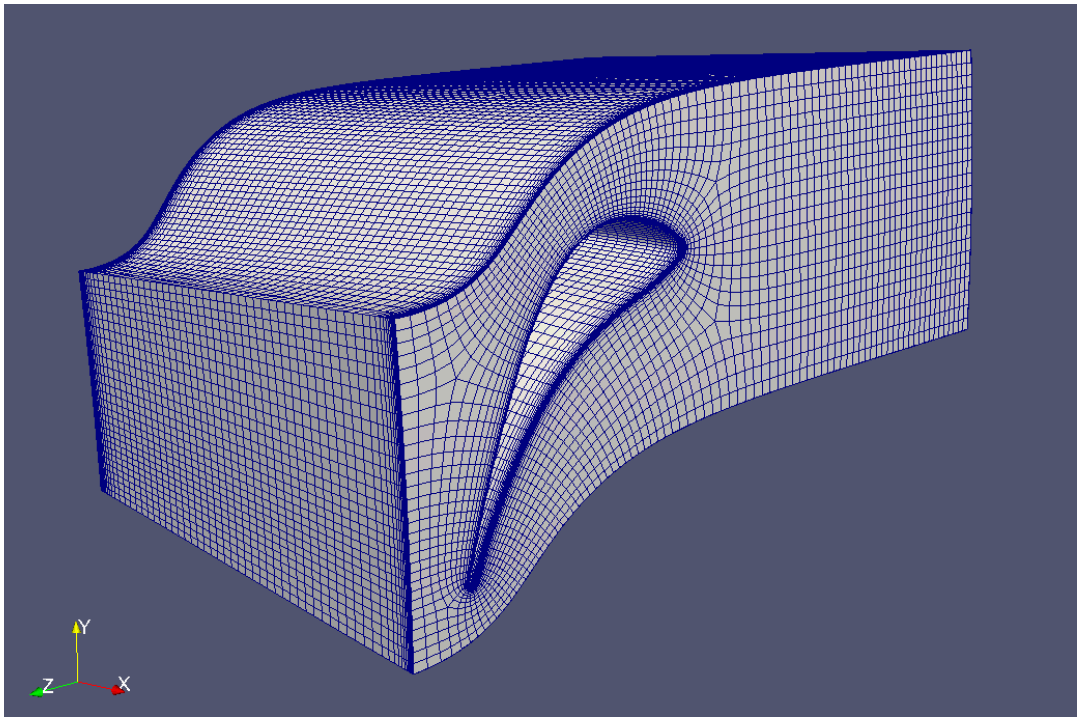
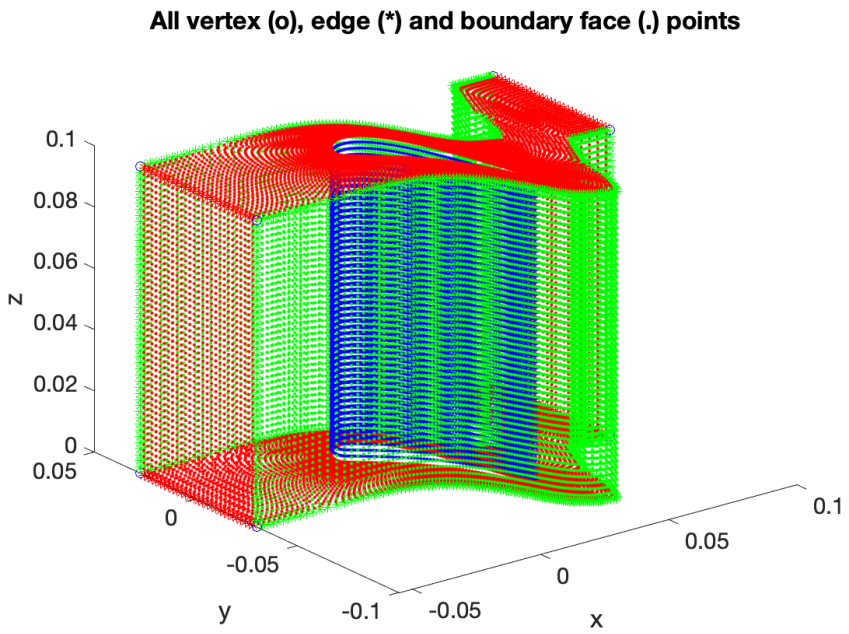


Figure 4.2: Initial mesh 4, showing decreasing size from shroud to hub.

4.2. Simulation parameters

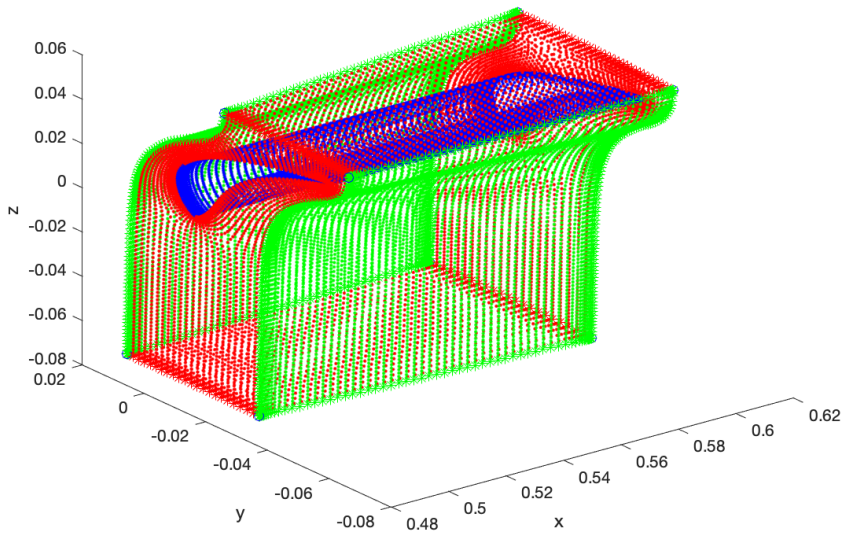
For the 3D test cases, the greedy algorithm is always used as these cases have a large number of nodes. The greedy variables and the support radius for each case are listed in Table 4.3 and the blade displacements are listed in Table 4.4. The node assignment for both meshes can be found in Figure 4.3. All nodes that are on the periodic boundaries are colored green. This includes both the surface and edge nodes. The two different deformation methods have a different node assignment for the different node groups:

- **Regular RBF:** All boundary edge and surface nodes are fixed except for the sliding surfaces in red where the blade intersects the surface. Hence, the green, the remaining red and blue nodes on the corners are all fixed. The blue nodes on the blade are moving nodes.
- **Periodic displacement (fixed corners):** The green surface nodes are allowed to be displaced in a periodic manner. The green edge nodes that are part of the outlet or inlet are fixed for mesh 3 but they are allowed to move for mesh 4. For mesh 3, the inlet and outlet are on the left and right side of the mesh, respectively. For mesh 4, they are on the bottom and top, respectively. The remaining green edge nodes are also allowed to move periodically. The red nodes are sliding nodes. The blue corner nodes are fixed (also included with moving nodes) and the blue nodes on the blade are moving nodes.



(a) Mesh 3 for translational periodic boundaries.

All vertex (o), edge (*) and boundary face (.) points



(b) Mesh 4 for rotational periodic boundaries.

Figure 4.3: Node assignment for simulations with periodic nodes (green), moving nodes (including static nodes, blue) and non-periodic surface and edge nodes (red).

Simulation variables	
Mesh 3	
Greedy tolerance	0.03
Boundary correction factor	50
Support radius, r (units)	0.335
Mesh 4	
Greedy tolerance	0.03
Boundary correction factor	50
Support radius, r (units)	0.323

Table 4.3: Greedy settings and support radii for the 3D mesh deformation cases.

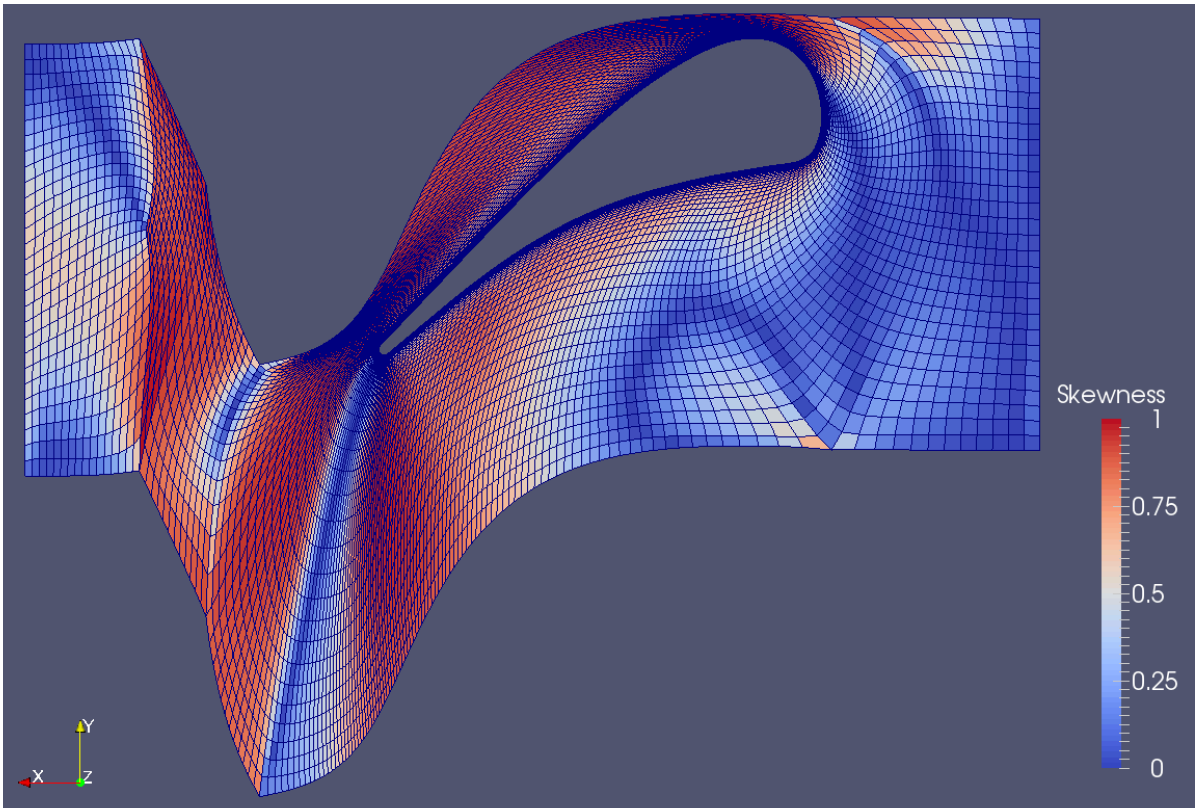
Blade displacement			
Mesh 3			
	x	y	z
Translation (units)	-0.025	0.01	0
Rotation (deg)	0	0	30
Center of rotation	0	0	0
Mesh 4			
	x	y	z
Translation (units)	0	0	0
Rotation (deg)	0	0	0.8
Center of rotation	0	0	0

Table 4.4: Blade deformation details for the 3D mesh deformation cases.

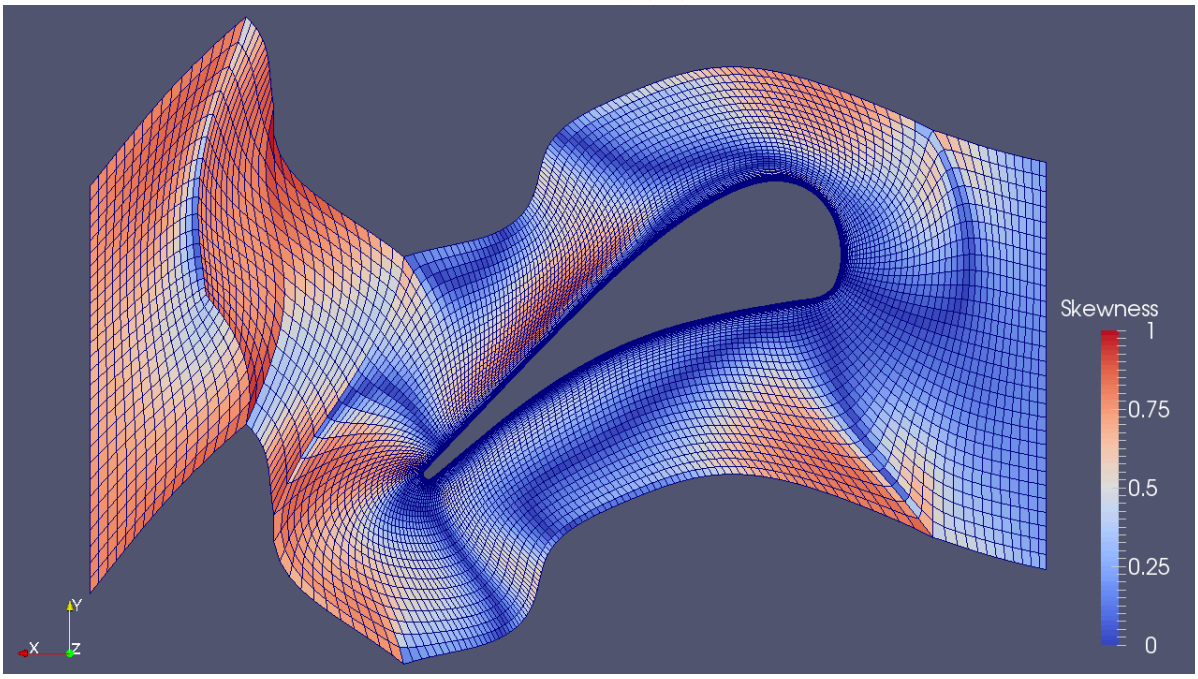
4.3. Translational periodic boundaries results

First, the regular RBF method was applied to mesh 3 (Fig. 4.4a) to be able to properly show the impact of the periodic boundary displacement method. The regular RBF method results in the creation of some very poor quality cells in the mesh as the maximum skewness is nearly 1. The mesh skewness values are listed in Table 4.5. The small clearance gap between the blade and the boundary above the blade only became smaller after the blade deformed, causing the mesh to become heavily skewed and almost degenerate. Also above and below the trailing edge some heavily skewed mesh cells are generated. The initial mesh 3 already had a high maximum skewness and somewhat high mean skewness, especially when compared to mesh 4. However, the regular RBF still managed to worsen both, especially the mean skewness.

Once the periodic boundaries are allowed to move as shown in Figure 4.4b, the mean skewness of the mesh reduces greatly and is even lower than it was for the initial mesh. When comparing the maximum skewness, only a slight improvement can be noticed for the periodic displacement method compared to the regular RBF method. However, it should be kept in mind that the initial mesh already has highly skewed mesh cells present. It can be seen that the poor quality region of skewed cells is already present initially along the upper edge of the blade but it has become much smaller with the periodic displacement method. Also the very poor regions around the trailing edge of the blade have significantly improved when compared to the regular RBF method. On the other hand, allowing the periodic boundaries to move, does create some sharp corners on the left side of the figure which were not yet present before. Hence, some deterioration of the mesh can also occur with this method, especially when the vertices of the mesh are kept fixed.



(a) Mesh 3 skewness after applying regular RBF method.



(b) Mesh 3 skewness after applying periodic displacement using the periodic distance method.

Figure 4.4: Mesh 3 skewness after deforming using different methods. The Wendland C^2 function is used with r=0.335 after 20 deformation steps.

	Initial	Regular RBF	Periodic displacement
Maximum skewness	0.972	0.999	0.977
Mean skewness	0.318	0.596	0.308

Table 4.5: Minimum mesh quality for different mesh deformation methods applied to mesh 3.

In order to analyse the mesh smoothness for the periodic distance method, two identical meshes are stacked upon one another in Figure 4.5. On first sight, the method seems to show a relatively smooth interface between the two meshes making it difficult to distinguish the actual interface. However, when zooming in on the interface (Fig. 4.6), it is clear that the method in 3D is not as smooth as is in 2D. Some small kinks can be noticed at the interface in the region between the two blades. However, it is only present for a small number of nodes at the interface, as for the rest of the interface nodes the transition from one mesh to the other is smooth. The reason for these kinks is not a flaw in the periodic distance method but rather an issue with choosing the periodic nodes to be displaced together with the internal nodes. After using greedy, a greedy correction is applied as is explained in Section 2.5. This correction for the internal nodes is based on the error of the nearest boundary point. The issue arises because the periodic nodes are also lying on a boundary. Hence, the code will not have a known error for these nodes yet which means that a zero correction is applied to the periodic nodes. This bug in the code is yet to be resolved.

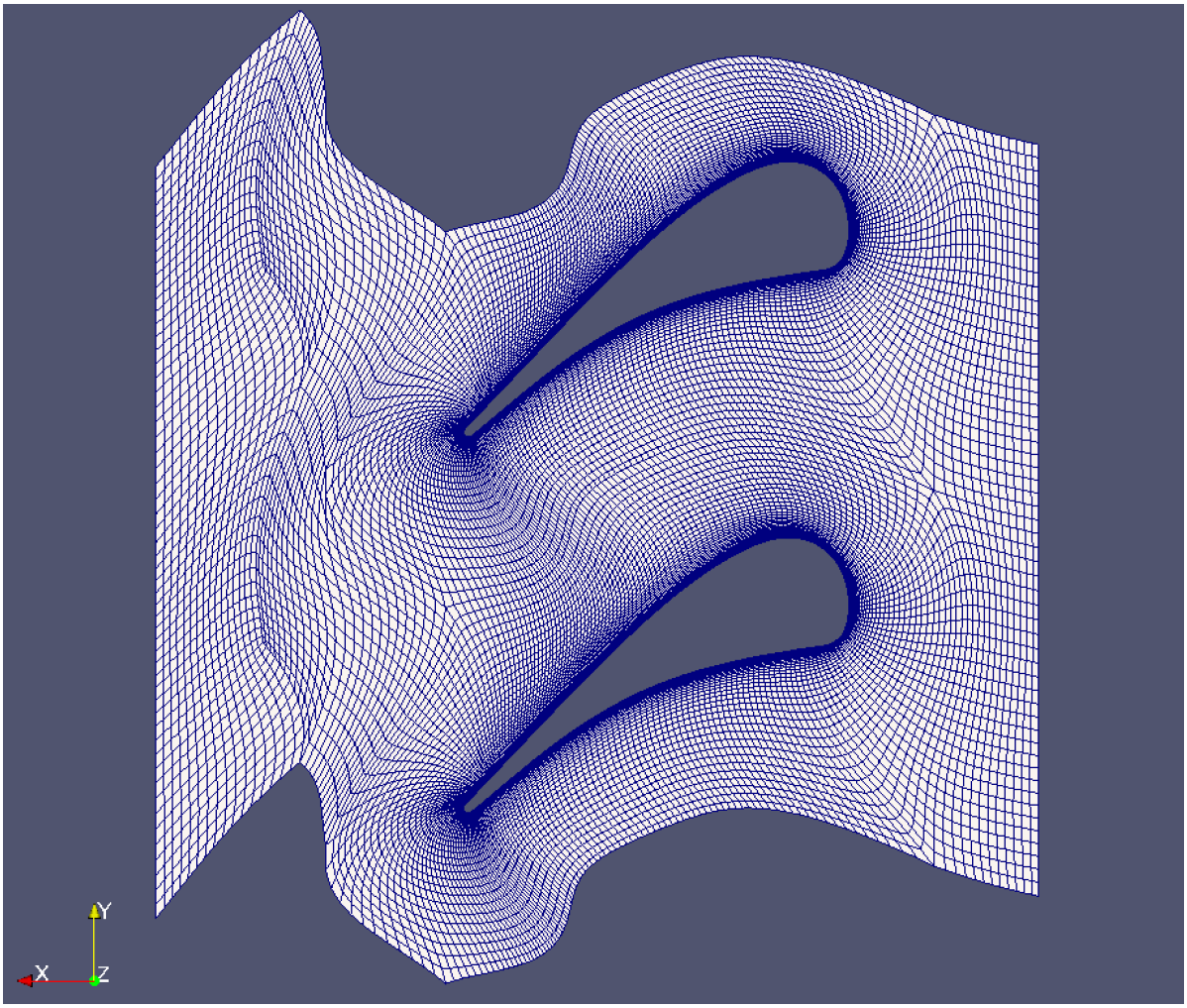


Figure 4.5: Two adjacent meshes after applying periodic displacement using periodic distance method (20 steps).

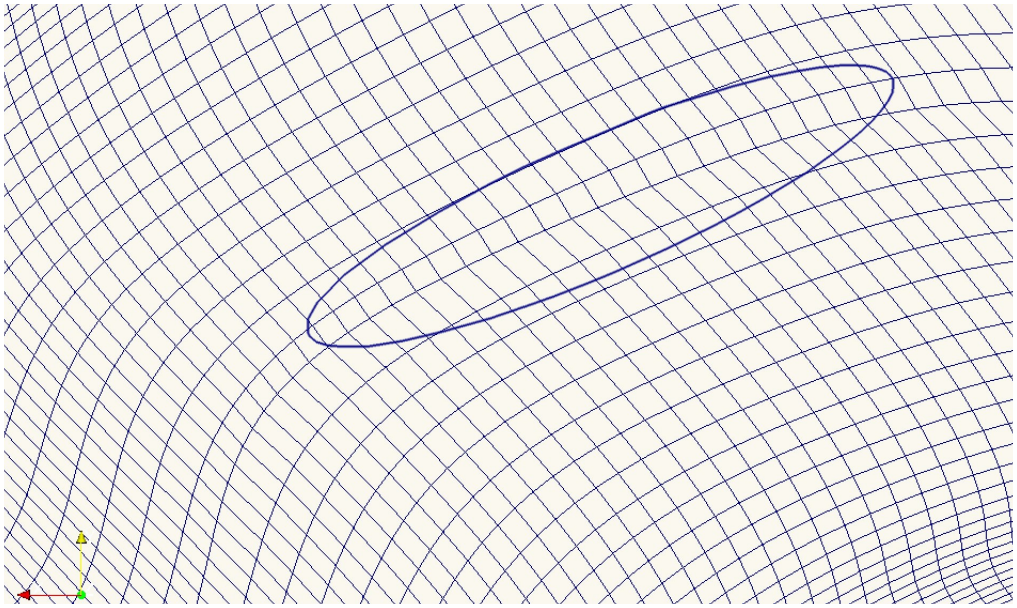
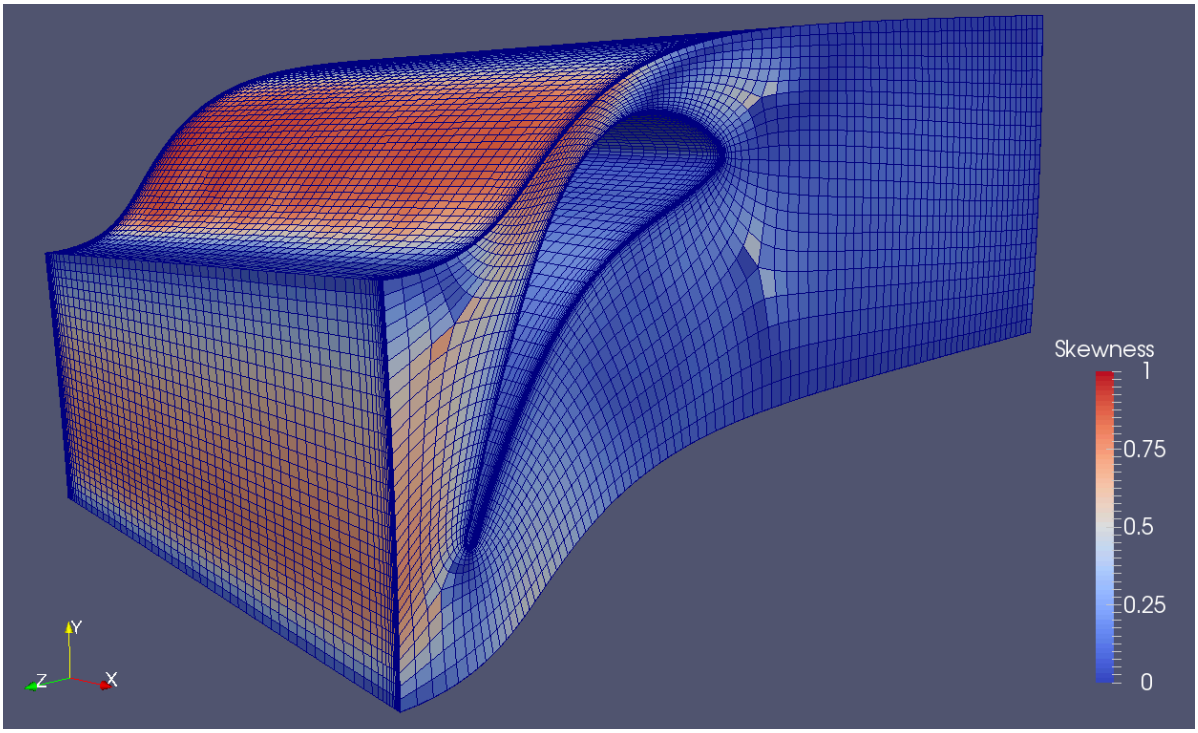


Figure 4.6: Enlarged interface between two adjacent translational periodic boundary meshes.

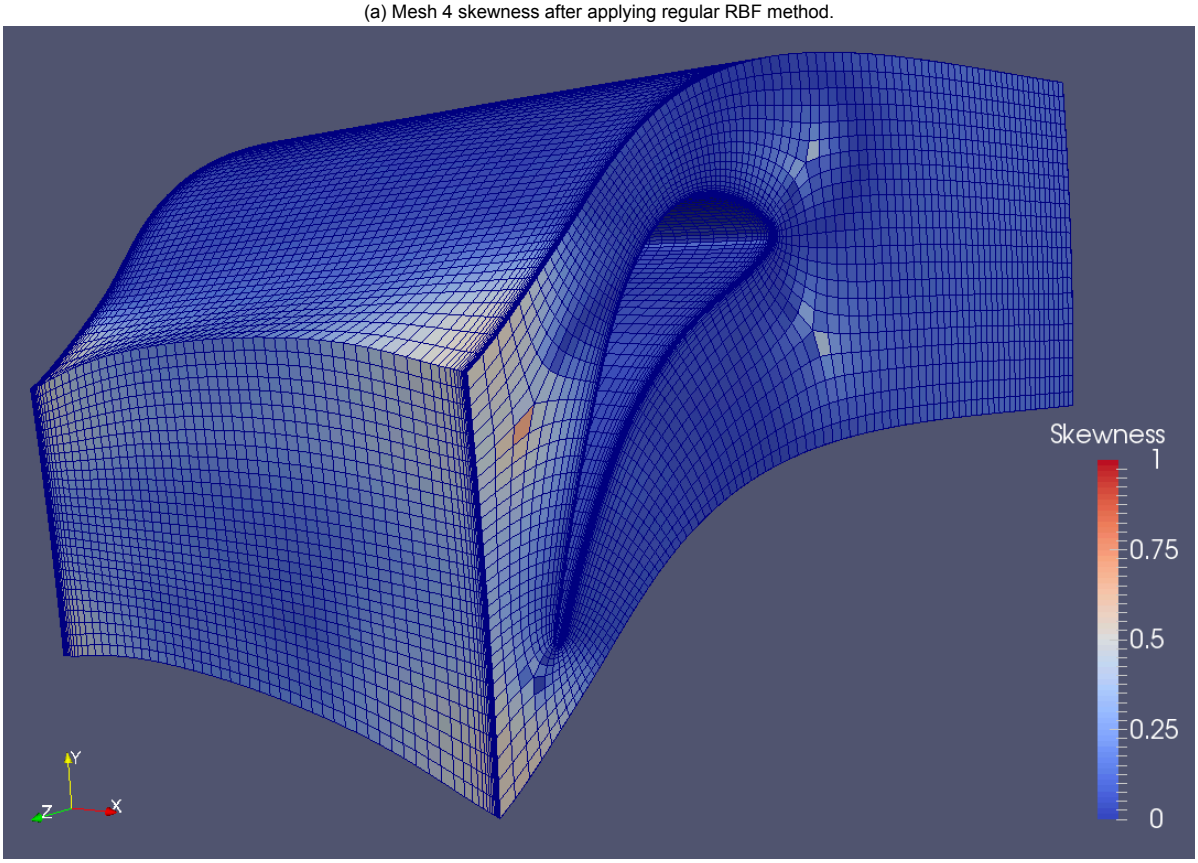
4.4. Rotational periodic boundaries results

For the rotational periodic boundaries case, also both the regular RBF and the periodic displacement method (using periodic distance) are applied to the mesh, which are shown in Figure 4.7. When using the regular RBF method and thus keeping all boundaries fixed except for the surfaces that intersect the blade, some very poor quality regions occur in the mesh as shown in Figure 4.7a. The worst region is right above the blade which is to be expected as the blade makes a positive rotation around the z-axis which results in the blade moving upwards in the domain. As the clearance gap between the blade and the fixed boundary is small here, the mesh cells become heavily skewed, causing the poor quality region. Another poorer quality region is at the outlet. Since the blade is relatively close to the outlet of the domain and the outlet is completely fixed, hence no sliding allowed, the cells become more skewed when the blade moves upward. This small clearance gap thus again results in poorer mesh quality, although not in the same amount as the region above the blade. The rest of the mesh between the blade and the inlet remains roughly the same as the blade deformation hardly affects this region. This is also reflected in the low mean skewness of the mesh, given in Table 4.6.

After applying the periodic distance method, some great improvements can be immediately noticed in Figure 4.7b. The first improvement in mesh skewness is in the region right above the blade. The displacement of the periodic boundaries allows for a better distribution of the nodes along the displaced boundary. This significantly alleviates the mesh skewness and also the size of the cells are closer to the original mesh cell size. Another region that shows great improvements is at the outlet since the nodes on the outlet surface are now allowed to slide. They will slide upwards, improving the quality of the previously more skewed mesh cells. The vertices are however fixed and with the displaced periodic edges, some sharper corners are generated at the outlet. As a result, the mesh quality is slightly lower in these corner regions than it was when using regular RBF. This could potentially be improved by allowing the fixed vertices to slide upward as well. When inspecting the maximum and mean skewness, it can be observed that the maximum skewness in the mesh is still quite high. However, this is due to only one specific cell, which is clearly visible on the shroud surface. This cell was already somewhat skewed in the initial mesh and this slightly worsened when deforming the mesh. The mean skewness however is almost as low as the initial mesh and again confirms the significant improvement of the mesh quality when allowing the periodic boundaries to move.



(a) Mesh 4 skewness after applying regular RBF method.



(b) Mesh 4 skewness after applying periodic displacement using the periodic distance method.

Figure 4.7: Mesh 4 skewness after deforming using different methods. The Wendland C^2 function is used with r=0.323 after 20 deformation steps.

When attaching two identical meshes, it can be seen in Figure 4.8 that the meshes (only shroud surface

	Initial	Regular RBF	Periodic displacement
Maximum skewness	0.433	0.964	0.720
Mean skewness	0.041	0.252	0.097

Table 4.6: Quality (skewness) for different mesh deformation methods applied to mesh 4.

displayed here) fit perfectly at the interface, displaying a smooth transition from one mesh to the other, also in terms of cell size. Even when zooming in on the interface (Fig. 4.9), the transition seems to remain smooth. It could be that the deformation of the blade is so that the problem of applying the wrong correction becomes insignificant and the kinks do not show up for this specific mesh.

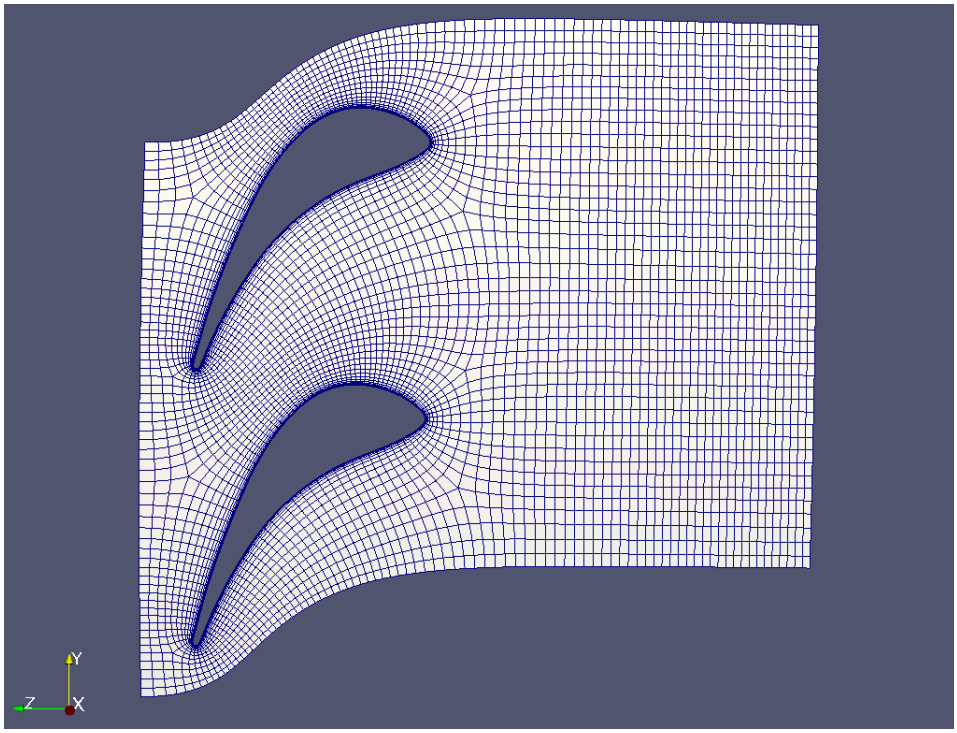


Figure 4.8: Two adjacent meshes after applying periodic distance method.

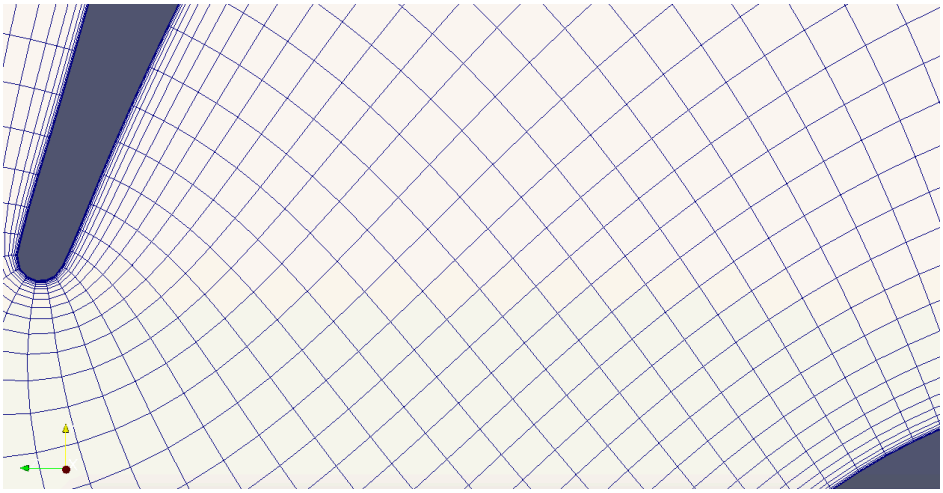


Figure 4.9: Enlarged interface between two adjacent rotational periodic boundary meshes.

5

Conclusions and recommendations

5.1. Conclusions

As expected, allowing the periodic boundaries to displace yields much higher mesh qualities than fixed boundaries for large deformations or for small clearance gap regions. This goes for both the periodic distance and the equal displacement method. However, the periodic distance clearly performs better in terms of smoothness once the periodic meshes are connected, allowing a smooth cell size transition over the interface for both the translational and rotational periodic boundary 2D cases. For the 3D translational case, some kinks occur at the mesh interface making the method less smooth. However, this was not caused by the periodic boundary displacement method but rather by a bug in the code concerning the greedy correction. Therefore, it is expected that the periodic distance method also produces smooth mesh transitions in 3D as is the case for the 2D cases. For the 3D rotational case, these kinks were not clearly visible, which allowed for a smooth transition. Apart from the smoothness, there did not seem to be a clear superior periodic boundary method (in 2D) in terms of minimum and mean mesh quality for both types of periodic boundaries. Furthermore, for the 2D cases, the mesh quality seems to be the highest when the mesh vertices are allowed to move in the form of sliding nodes, which mainly causes an improvement of the minimum mesh quality.
For the 3D cases, only the periodic distance method was used hence no comparison can be made between both periodic methods.

In terms of computation time, the periodic distance is shown to be faster than the equal displacement method since the evaluation matrix size is $N_m \times N_m$ compared to $(N_m + N_p) \times (N_m + N_p)$. When comparing translational and rotational periodic boundary cases of the same mesh size for the periodic distance method, the rotational boundary cases would likely have a higher computation time due to the several transformations required between coordinate systems.

Next, the equal displacement method showed to be highly dependent on the specific deformation applied to the mesh. However, using an ω-value of 0.5 and thus an equal influence of both periodic boundaries seems to be a reasonably good choice, independent of the deformation.

When comparing different RBF's, the Wendland C^2 function gives good results with both periodic displacement methods for both types of periodic boundaries. Especially compared to the TPS RBF function, the Wendland C^2 function gives superior mesh quality results for the periodic distance method. This is because with the TPS function, the mesh corners become very sharp with a highly skewed mesh in this region as a result.

For the 2D cases, using a greedy algorithm with a tolerance of 10^{-5} allows for the same mesh quality as when using all boundary nodes but the computation time is six times less. When the greedy algorithm is used in the 3D cases, the desired outcome was not exactly obtained because of the bug with the greedy correction as mentioned earlier.

5.2. Future recommendations

Although a small investigation has already been done into the equal displacement method, there is still room for further investigation and improvement of the method. Some way of automatizing the selection of the optimal ω value would be convenient when using the method on different meshes. It might also be interesting to look at other types of combinations instead of simply using a linear combination of the two interpolation coefficients.
The equal displacement method is yet to be implemented in 3D. This means that some adaptions have to be made to the greedy slide method in 3D to incorporate the *total* interpolation matrix for the periodic nodes. Hence, one large *total* interpolation matrix would be required that includes all of the conditions for the sliding nodes and all conditions for the periodic nodes. However, this might become a very large matrix to be inverted. Another option could be to use the pseudo sliding method as is used for the 2D cases. In that case, the greedy slide method would be replaced by a regular greedy method and an additional projection step for the sliding nodes.
Also, a more extensive comparison could be done for the computation time of the periodic distance and the equal displacement method. This was only done for small 2D cases and although a difference could already be noticed, it is expected to be much more distinct for 3D cases.

The vertices for the 3D cases have remained fixed so far, so here there is still potential to improve the mesh quality by adapting the method for sliding vertices as was used in the 2D cases. For the rotational periodic boundary case used in this project, this would imply that some extra information would be required about the geometry. The edges where the hub and the shroud intersect the outlet are located on imaginary circles of different radii‘. The radii and center points of these circles would have to be implemented in the code so that the vertices know what direction to slide along.

The test cases that are used in this thesis project are relatively simple test cases where the periodic boundaries were periodic in one specific direction and there is only one pair of periodic boundaries per case. In the example that was used for the nuclear industry application, a periodic domain was shown that included three different pairs of periodic boundaries. Two of these where periodic in vertical direction but one boundary of one pair was located below a boundary of the other pair and then the opposite was true for the second boundaries of each pair. This makes the mesh more complicated and would require some adjustments to the code to be able to move all these periodic boundaries simultaneously.

So far, the only mesh deformation method used here is the RBF interpolation method. However, it should also be possible to implement the periodic boundary displacement method with other types of interpolation mesh deformation methods such as the IDW method.

Finally, performing a CFD simulation could serve as a proper validation of the performance of the periodic boundary displacement method on periodic domains, for which the resulting data can be compared with other available data. This would allow to assess the accuracy of the solution when using the periodic displacement method.

Bibliography

[1] Y. Rozenberg, S. Aubert and G. Bénéfice. Fluid structure interaction problems in turbomachinery using rbf interpolation and greedy algorithm. In *Proceedings of ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*, 2014.

[2] J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA journal*, 28(8):1381–1388, 1990. doi: 10.2514/3.25229.

[3] R. T. Biedron and E. M. Lee-Rausch. Rotor airloads prediction using unstructured meshes and loose cfd/csd coupling. Technical report, American Institute of Aeronautics and Astronautics, NASA Langley Research Center, Hampton, VA 23681, 2008.

[4] F. J. Blom. Considerations of the spring analogy. *International Journal for Numerical Methods in Fluids*, 32:736–739, 2000. doi: 10.1002/(SICI)1097-0363(20000330)32:6<647::AID-FLD979>3.0.CO;2-K.

[5] Hsu S.-Y., Chang C.-L. and Samareh J. A simplified mesh deformationmethod using commercial structural analysis software. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference New York*, 2004. doi: 10.2514/6.2004-4409.

[6] D. Chang and S. Tavoularis. Numerical simulation of turbulent flow in a 37-rod bundle. *Nuclear Engineering and Design*, 237(6):575–590, 2007. doi: 10.1016/j.nucengdes.2006.08.001.

[7] A. de Boer, M. S. van der Schoot and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers and Structures*, 85(11–14):784–795, 2007. doi: 10.1016/j.compstruc.2007.01.013.

[8] P. Garrido de la Serna. Adjoint-based 3d shape optimization for turbomachinery applications. Technical report, 2019.

[9] C. L. Bottasso, D. Detomi and R. Serra. The ball-vertex method: A new simple spring analogy method for unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 194(39–41):4244–4264, 2005. doi: 10.1016/j.cma.2004.08.014.

[10] A. H. Memon et al. Dynamic fluid-structure interaction analysis of propeller aircraft wing. *American Scientific Research Journal for Engineering, Technology, and Sciences*, 45(1):64–74, 2018.

[11] P. Crosetto et al. Fluid structure interaction simulations of physiological blood flow in the aorta. Technical report, 2010.

[12] S. Aubert, F. Mastrippolito, Q. Rendu et al. Planar slip condition for mesh morphing using radial basis functions. *Procedia Engineering*, 203:349–361, 2017. doi: 10.1016/j.proeng.2017.09.819.

[13] M. Stelldinger, T. Giersh, F. Figaschewsky and A. Kühhorn. A semi-unstructured turbomachinery meshing library with focus on modeling of specific geometrical features. In *VII European Congress on Computational Methods in Applied Sciences and Engineering*, 2016.

[14] Z. S. Mouroutis, D. C. Charmpis, G. A. Markou, and M. Papadrakakis. The ortho-semi-torsional (OST) spring analogy method for 3d mesh moving boundary problems. *Computer Methods in Applied Mechanics and Engineering*, 196(4–6):747–765, 2007. doi: 10.1016/j.cma.2006.04.009.

[15] B. T. Helenbrook. Mesh deformation using the biharmonic operator. *International Journal for Numerical Methods in Engineering*, 56(7):1007–1021, 2003. doi: 10.1002/nme.595.

[16] D. Amirante, N. J. Hills and C. J. Barnes. A moving mesh algorithm for aero-thermo-mechanical modeling in turbomachinery. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, 00:1–24, 2017. doi: 10.1002/nme.

[17] S. L. Karman. Unstructured viscous layer insertion using linear-elastic smoothing. *AIAA Journal*, 45(1):168–180, 2007. doi: 10.2514/1.23283.

[18] P. M. Knupp. Algebraic mesh quality metrics. Technical report, SIAM Journal on Scientific Computing, 2001.

[19] C. Farhat, C. Degand, B. Koobus and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Finite Elements in Analysis and Design*, 163(1–4):231–245, 1998. doi: 10.1016/S0045-7825(98)00016-4.

[20] X. Li and Y. Gao. Methods of simulating large-scale rod bundle and application to a 17 × 17 fuel assembly with mixing vane spacer grid. *Nuclear Engineering and Design*, 267:10–22, 2014.

[21] Y. Liu and L. Tan. Tip clearance on pressure fluctuation intensity and vortex characteristic of a mixed flow pump as turbine at pump mode. *Renewable Energy*, 129:606–615, 2018. doi: 10.1016/j.renene.2018.06.032.

[22] M. T. Mathew. Mesh deformation using radial basis function interpolation with sliding boundary nodes. Technical report, 2019.

[23] Z. Yangand D.J. Mavriplis. Mesh deformation strategy optimized by the adjoint method on unstructured meshes. *AIAA Journal*, 45(12):2885–2896, 2007. doi: 10.2514/1.30592.

[24] E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. *AIAA Journal*, 40(6):1155–1163, 2002. doi: 10.2514/2.1765.

[25] R. Faria, S. Oliveira and A. L. Silvestre. A fluid-structure interaction model for dam-water systems: Analytical study and application to seismic behavior. *Advances in Mathematical Physics*, 2019(11): 1–14, 2019. doi: 10.1155/2019/8083906.

[26] A. Elsayed, F. Owis and M. M. Abdelrahman. Numerical computation and optimization of turbine blade film cooling. *Advances in Mechanical Engineering*, 2014, 2014. doi: 10.1155/2014/528031.

[27] X. Liu, N. Qin and H. Xia. Fast dynamic grid deformation based on delaunay graph mapping. *Journal of Computational Physics*, 211(2):405–423, 2006. doi: 10.1016/j.jcp.2005.05.025.

[28] T. Rendall and C. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008. doi: 10.2514/6.2008-305.

[29] T. Rendall and C. Allen. Reduced surface point selection options for efficient mesh deformation using radial basis functions. *Journal of Computational Physics*, 229(8):2810–2820, 2010. doi: 10.1016/j.jcp.2009.12.006.

[30] R. Schaback. A practical guide to radial basis functions.

[31] M. M. Selim and R. P. Koomullil. Mesh deformation approaches – a survey. *Journal of Physical Mathematics*, 7(2), 2016. doi: 10.4172/2090-0902.1000181.

[32] E. ter Hofstede, S. Kottapalli and A. Shams. Numerical prediction of flow induced vibrations in nuclear reactorapplications. *Nuclear Engineering and Design*, 319:81–90, 2017.

[33] L. Uyttersprot. Inverse distance weighting mesh deformation a robust and efficient method for unstructured meshes. Technical report, 2014.

[34] F. Bos, B. van Oudheusden and H. Bijl. Radial basis function based mesh deformation applied to simulation of flow around flapping wings. *Computers and Fluids*, 79:167–177, 2013. doi: 10.1016/j.compfluid.2013.02.004.

[35] H. Bijl, A. van Zuijlen and A. de Boer. Fluid-structure interaction lecture notes, May 2008.

[36] T. Gillebaart, D. Blom, A. van Zuijlen and H. Bijl. Adaptive radial basis function mesh deformation using data reduction. *Journal of Computational Physics*, 321:997–1025, 2016. doi: 10.1016/j.jcp.2016.05.036.

[37] H. Wendland. Konstruktion und untersuchung radialer basisfunktionen mit kompaktem trager. Technical report, 1996.

[38] J. A. S. Witteveen. Explicit and robust inverse distanceweighting mesh deformation for cfd. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010. doi: 10.2514/6.2010-165.

[39] D. Zeng and C. R. Ethier. A semi-torsional analogy method for updating the unstructured meshes in 3d moving domains. *Finite Elements in Analysis and Design*, 41(11):1118–1139, 2005. doi: 10.1016/j.finel.2005.01.003.

[40] Y. Zhao and A. Forhad. A general method for simulation of fluid flows with moving and compliant boundaries on unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 192(39-40):4439–4466, 2003. doi: 10.1016/S0045-7825(03)00424-9.

[41] J. Zhong and Z. Xu. A modal approach for coupled fluid structure computations of wing flutter. *Journal of Aerospace Engineering*, 231(1):72–81, 2017. doi: 10.1177/0954410016644630.

[42] A. S. F. Wong, H. M. Tsai, J. Cai, Y. Zhu and F. Liu. Unsteady flow calculations with a multi-block moving mesh algorithm. *AIAA Journal*, 39(6):1021–1029, 2001. doi: 10.2514/2.1442.

A Algorithms

This appendix contains some of the most important algorithms used in the code for this project.

A.1. Regular RBF

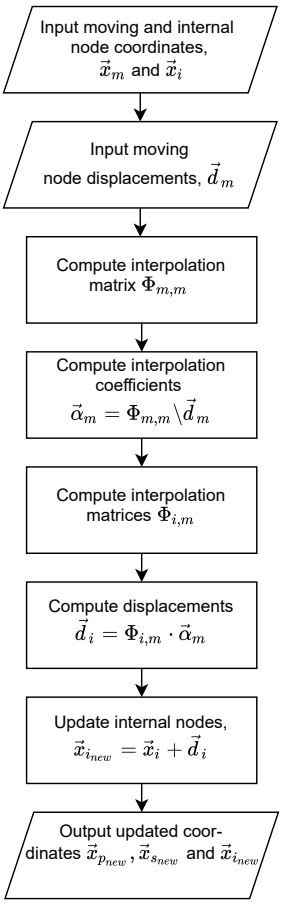


Figure A.1: Flowchart of regular RBF algorithm.

A.2. Periodic sliding (2D)

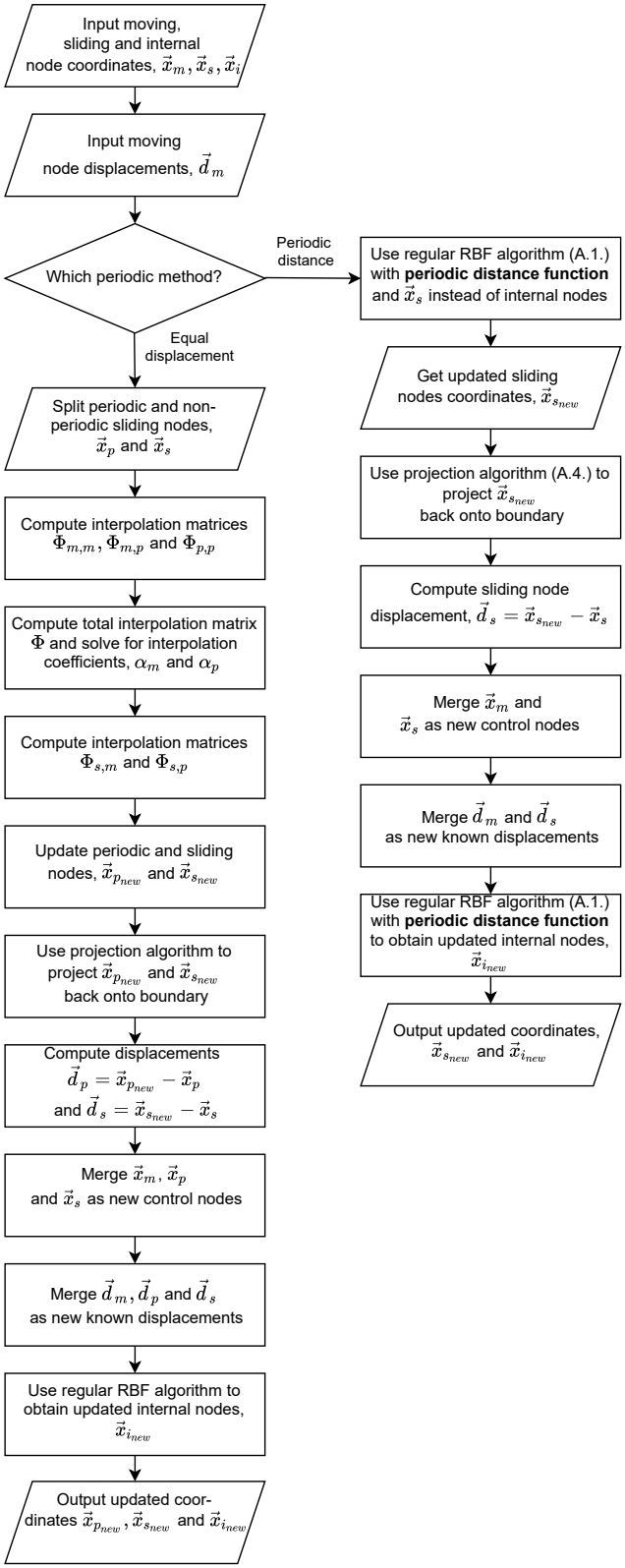


Figure A.2: Flowchart of periodic sliding algorithm in 2D.

A.3. Periodic displacement (2D)

In the 2D results chapter (Ch. 3), periodic displacement is applied to mesh 1 with the vertices of the outer block either fixed or allowed to move. When they are allowed to move, they are considered to be part of the sliding nodes and not the periodic nodes. The difference between the periodic sliding and the periodic displacement method is that the nodes on the periodic boundaries are not projected back onto their initial boundaries when using periodic displacement. When only a displacement of the periodic boundaries is desired and no sliding of the other boundary nodes, the periodic displacement is the same as the regular RBF with a different distance function for the periodic distance method.

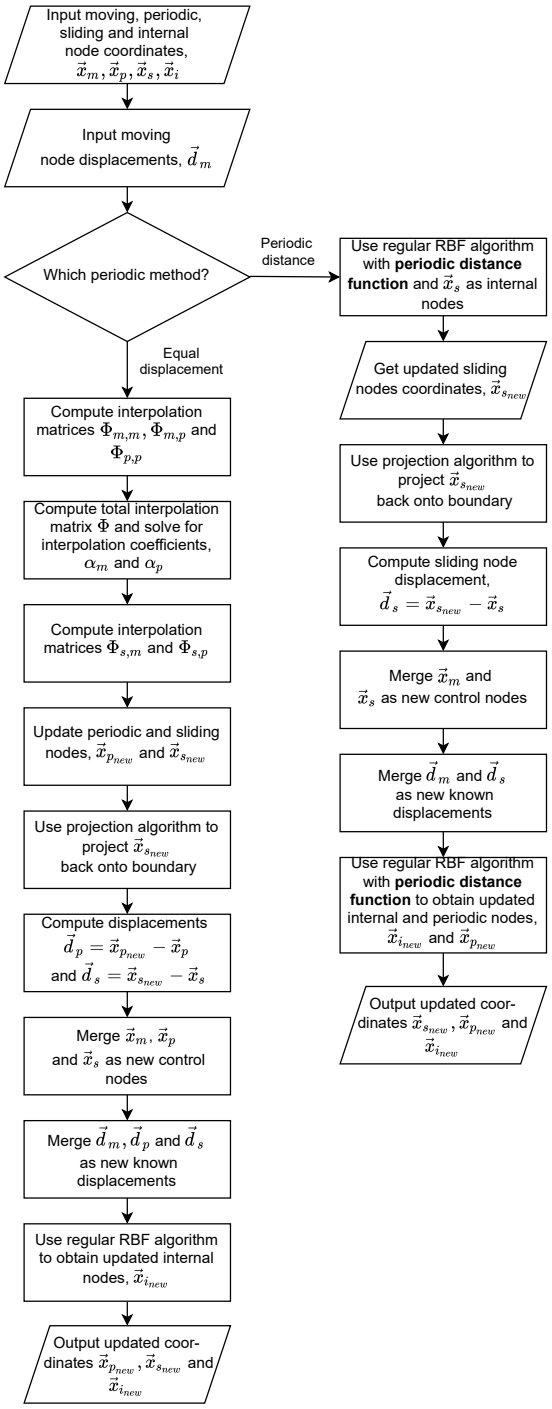


Figure A.3: Flowchart of periodic displacement algorithm in 2D.

A.4. Projection algorithm

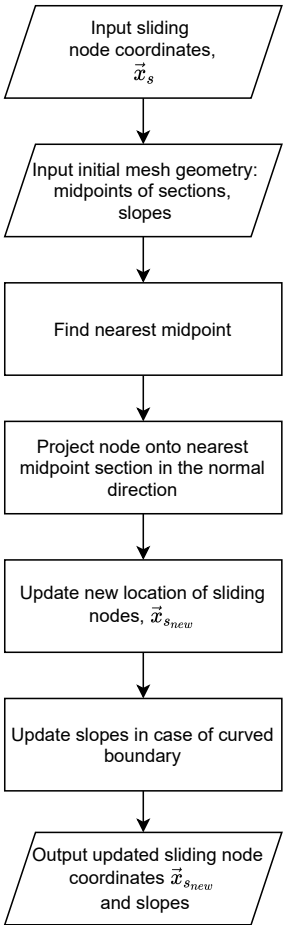


Figure A.4: Flowchart of projection algorithm.

A.5. 3D greedy Sliding

The transformations between the different coordinate systems are highlighted in blue and are only required for the periodic distance method on domains with rotational periodic boundaries.

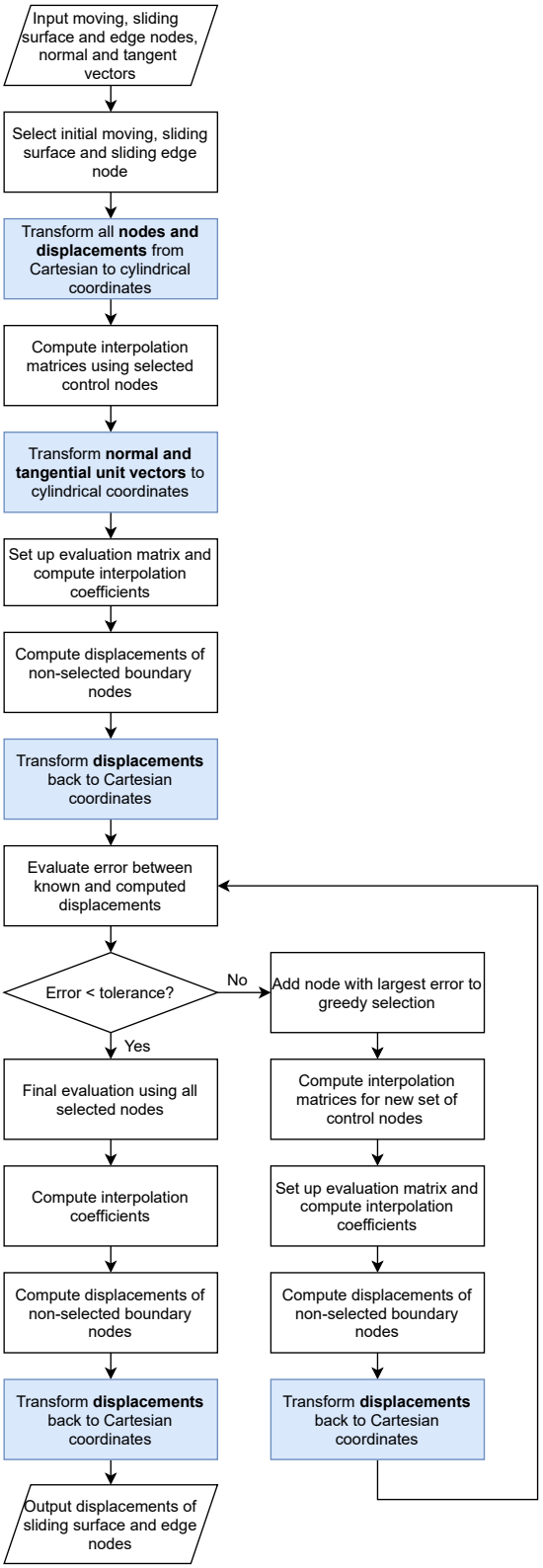


Figure A.5: Flowchart of 3D sliding greedy algorithm.

B

Frame transformations

When the domain has rotational periodic boundaries, some transformations are usually performed between the Cartesian coordinate system and the polar (2D) or cylindrical (3D) coordinate system.

B.1. Coordinates

B.1.1. From 2D Cartesian to polar

From (x,y)-coordinates to (r,θ)-coordinates.

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \arctan(\tfrac{y}{x}) \end{aligned} \tag{B.1}$$

B.1.2. From polar to 2D Cartesian

From (r,θ)-coordinates to (x,y)-coordinates.

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \end{aligned} \tag{B.2}$$

B.1.3. From 3D Cartesian to cylindrical

From (x,y,z)-coordinates to (r,θ,z)-coordinates.

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \arctan(\tfrac{y}{x}) \\ z &= z \end{aligned} \tag{B.3}$$

B.1.4. From cylindrical to 3D Cartesian

From (r,θ,z)-coordinates to (x,y,z)-coordinates.

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \\ z &= z \end{aligned} \tag{B.4}$$

B.2. Vector fields

A cylindrical vector can be expressed in terms of Cartesian vectors using the following transformation:

$$\begin{bmatrix} A_r \\ A_\theta \\ A_z \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{B.5}$$

Here, θ is computed using the expression from Equation B.1

B.3. Euclidean distance

The Euclidean distance function is used when setting up the interpolation matrices. When the mesh contains rotational periodic boundaries, the Euclidean distance expression has to be altered to work with either polar (2D) or cylindrical (3D) coordinates. The derivation of the adapted expressions are shown in the following sections.

B.3.1. From Cartesian to polar (2D)

The 2D Euclidean distance function in polar coordinates is obtained by substituting the expressions from Eq. B.2.

$$\begin{aligned}
\delta =& \sqrt{\delta x^2 + \delta y^2} \\
=& \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
=& \sqrt{(r_2 \cos\theta_2 - r_1 \cos\theta_1)^2 + (r_2 \sin\theta_2 - r_1 \sin\theta_1)^2} \\
=& \sqrt{r^2 \cos^2\theta_2 + r_1^2 \cos^2\theta_1 - 2r_1 r_2 \cos\theta_1 \cos\theta_2 + r_2^2 \sin^2\theta_2 + r_1^2 \sin^2\theta_1 - 2r_1 r_2 \sin\theta_1 \sin\theta_2} \\
=& \sqrt{r_1^2 \left(\cos^2\theta_1 + \sin^2\theta_1\right) + r_2^2 \left(\cos^2\theta_2 + \sin^2\theta_2\right) - 2r_1 r_2 (\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2)} \\
=& \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2)} \quad using \quad \cos(A - B) = \cos A \cos B + \sin A \sin B
\end{aligned} \tag{B.6}$$

B.3.2. From Cartesian to cylindrical (3D)

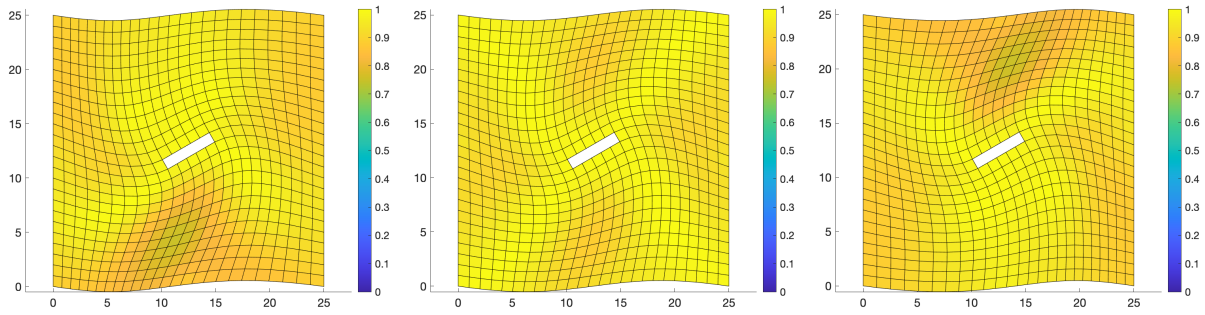
The 3D Euclidean distance function in cylindrical coordinates follows the same derivation as in Equation B.6, with an additional component in the z-direction.

$$\begin{aligned}
\delta =& \sqrt{\delta x^2 + \delta y^2 + \delta z^2} \\
=& \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\
=& \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2) + (z_2 - z_1)^2}
\end{aligned} \tag{B.7}$$

C

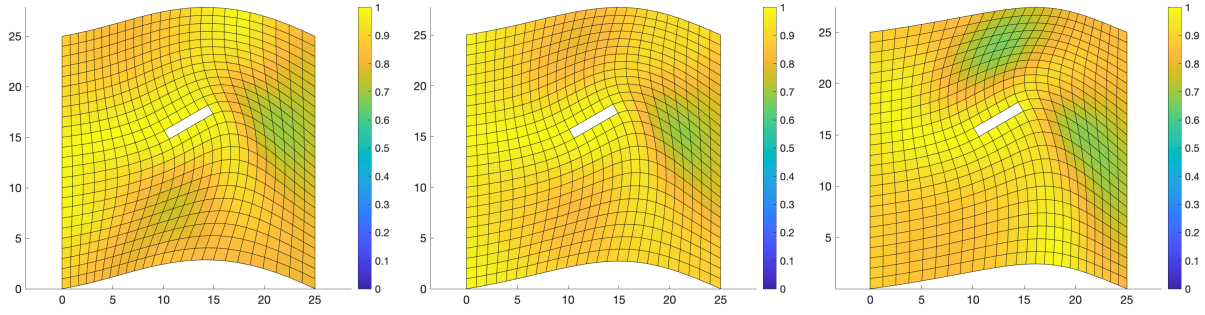
Equal displacement method

This appendix shows mesh 1 after applying the equal displacement method in 20 deformation steps using the Wendland C^2 RBF function (r=62.5) for ω-values 0, 0.5 and 1.



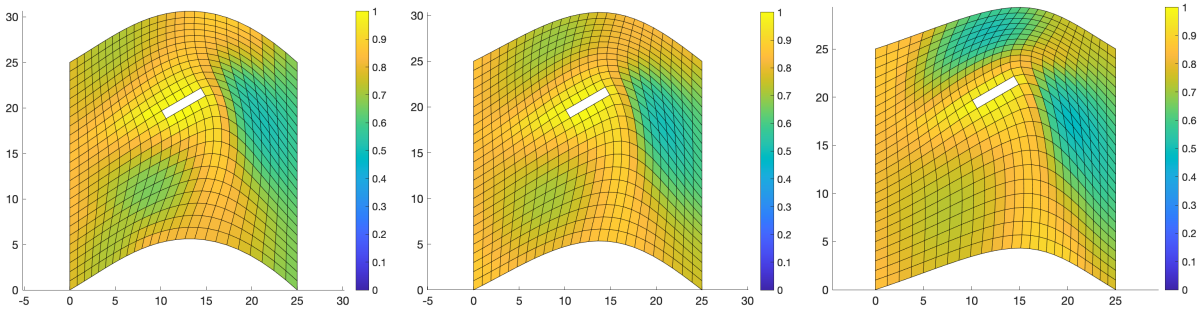
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.1: 30° rotation, 0 units vertical displacement and 0 units horizontal displacement.



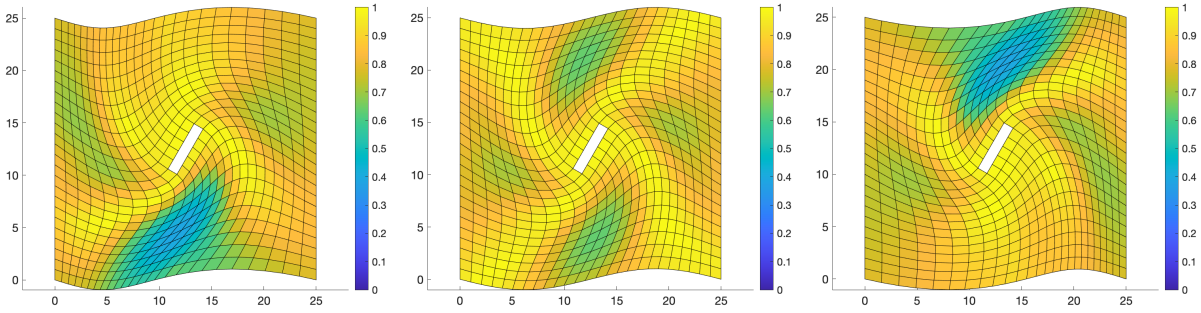
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.2: 30° rotation, 4 units vertical positive displacement and 0 units horizontal displacement.



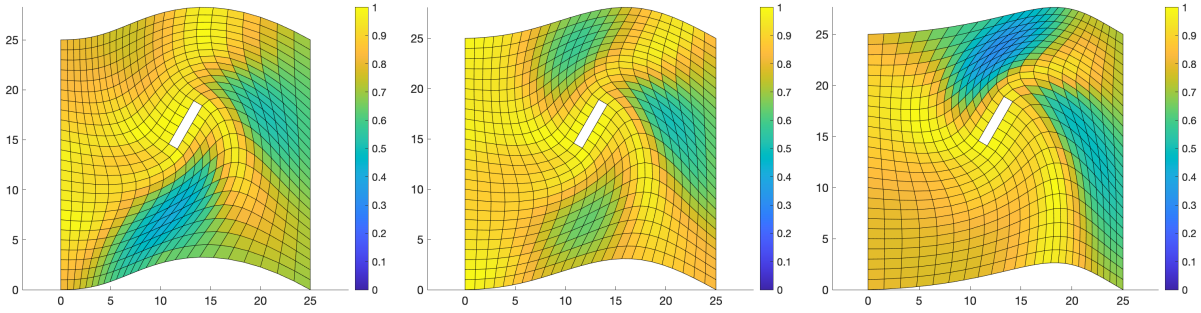
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.3: 30° rotation, 8 units vertical positive displacement and 0 units horizontal displacement.



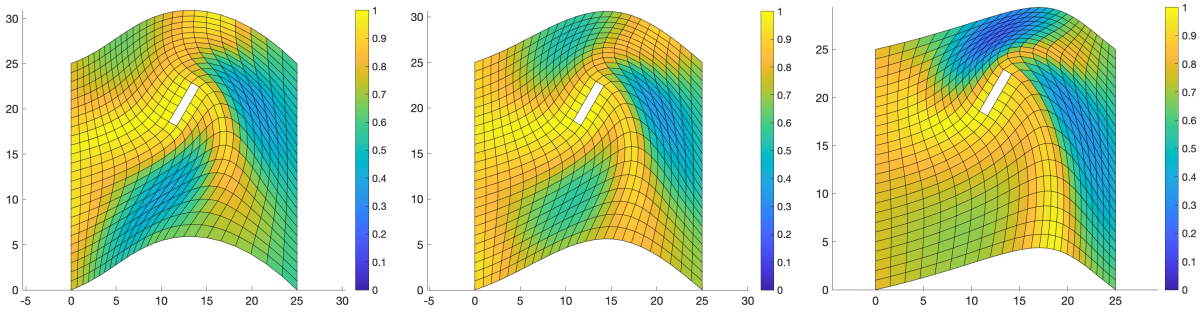
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.4: 60° rotation, 0 units vertical displacement and 0 units horizontal displacement.



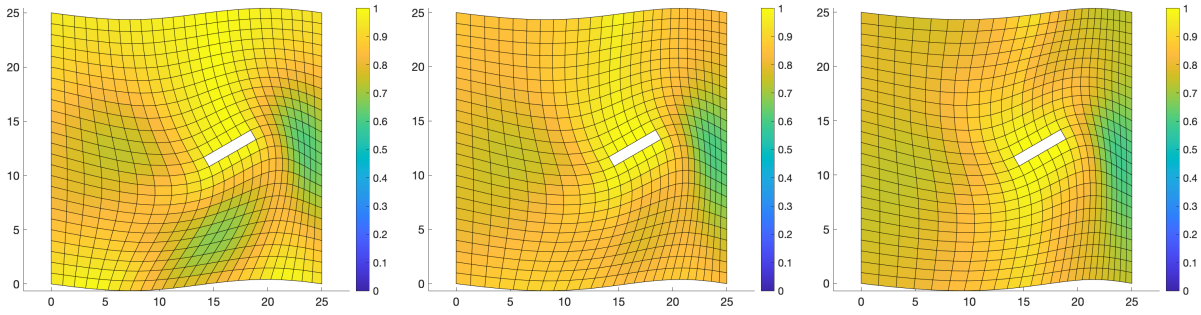
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.5: 60° rotation, 4 units vertical positive displacement and 0 units horizontal displacement.



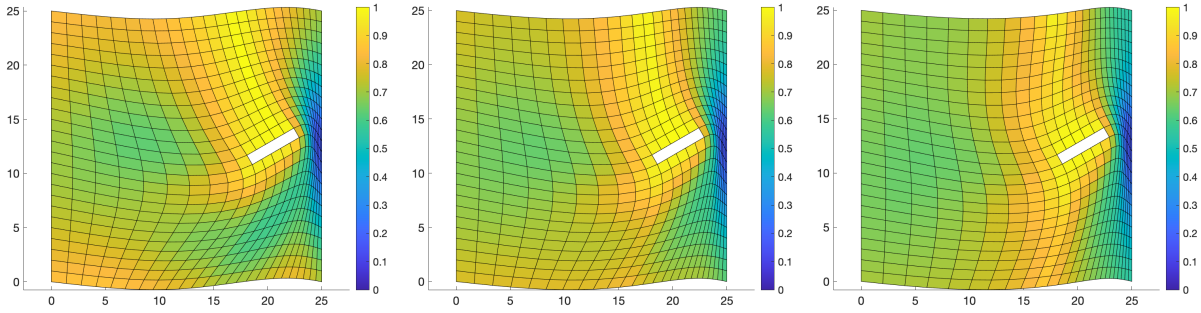
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.6: 30° rotation, 8 units vertical positive displacement and 0 units horizontal displacement.



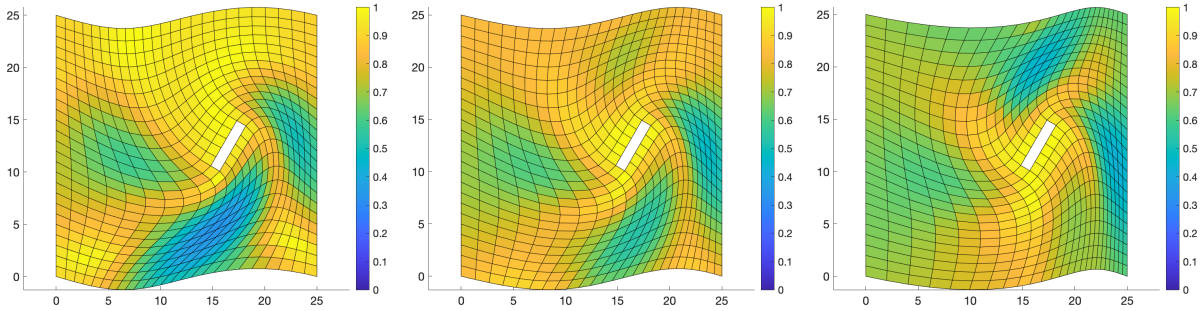
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.7: 30° rotation, 0 units vertical displacement and 4 units positive horizontal displacement.



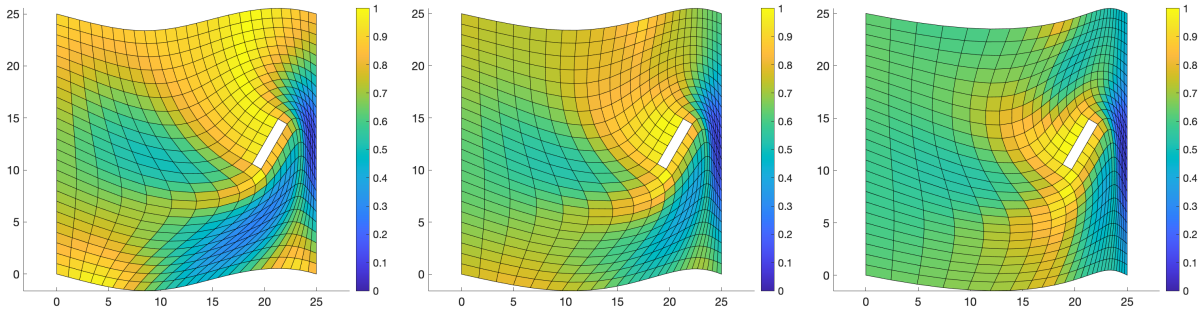
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.8: 30° rotation, 0 units vertical displacement and 8 units positive horizontal displacement.



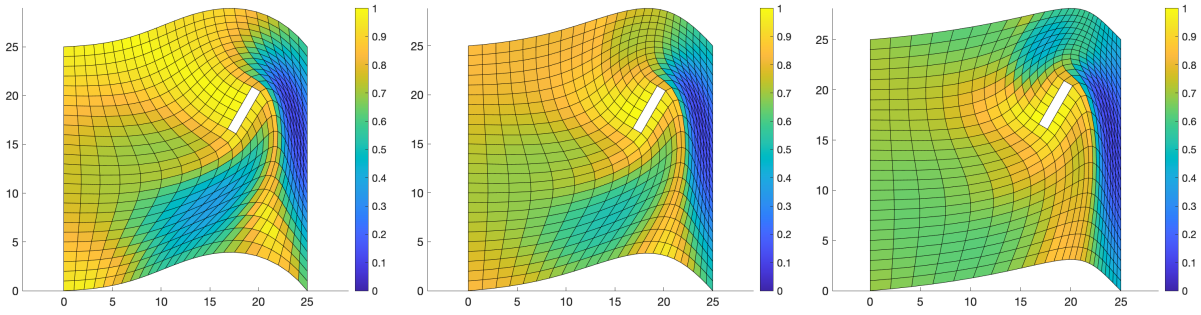
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.9: 60° rotation, 0 units vertical displacement and 4 units positive horizontal displacement.



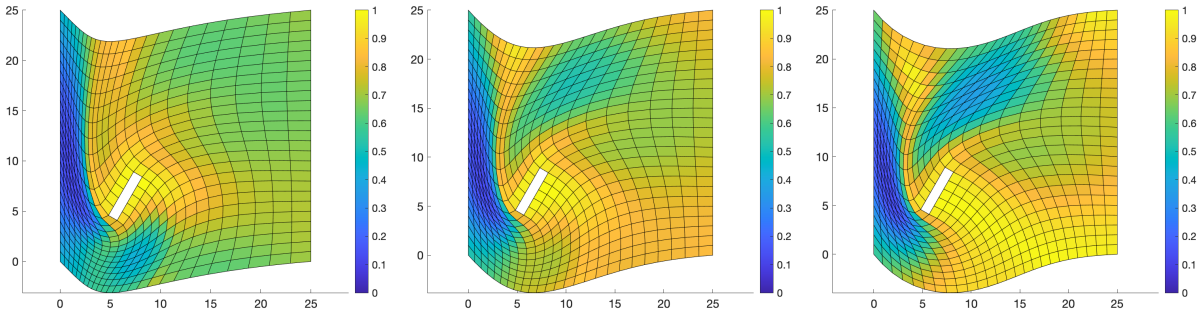
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.10: 60° rotation, 0 units vertical displacement and 8 units positive horizontal displacement.



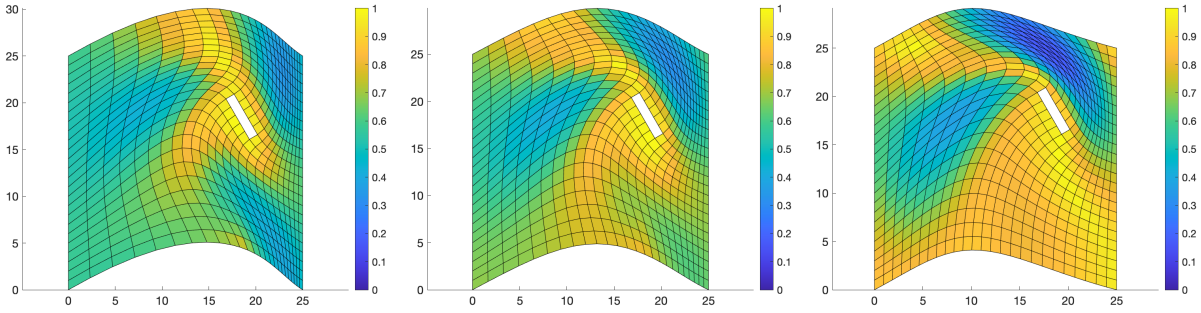
(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.11: 60° rotation, 6 units positive vertical displacement and 6 units positive horizontal displacement.



(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.12: 60° rotation, 6 units negative vertical displacement and 6 units negative horizontal displacement.



(a) $\omega = 0$. (b) $\omega = 0.5$. (c) $\omega = 1$.

Figure C.13: -60° rotation, 6 units positive vertical displacement and 6 units positive horizontal displacement.