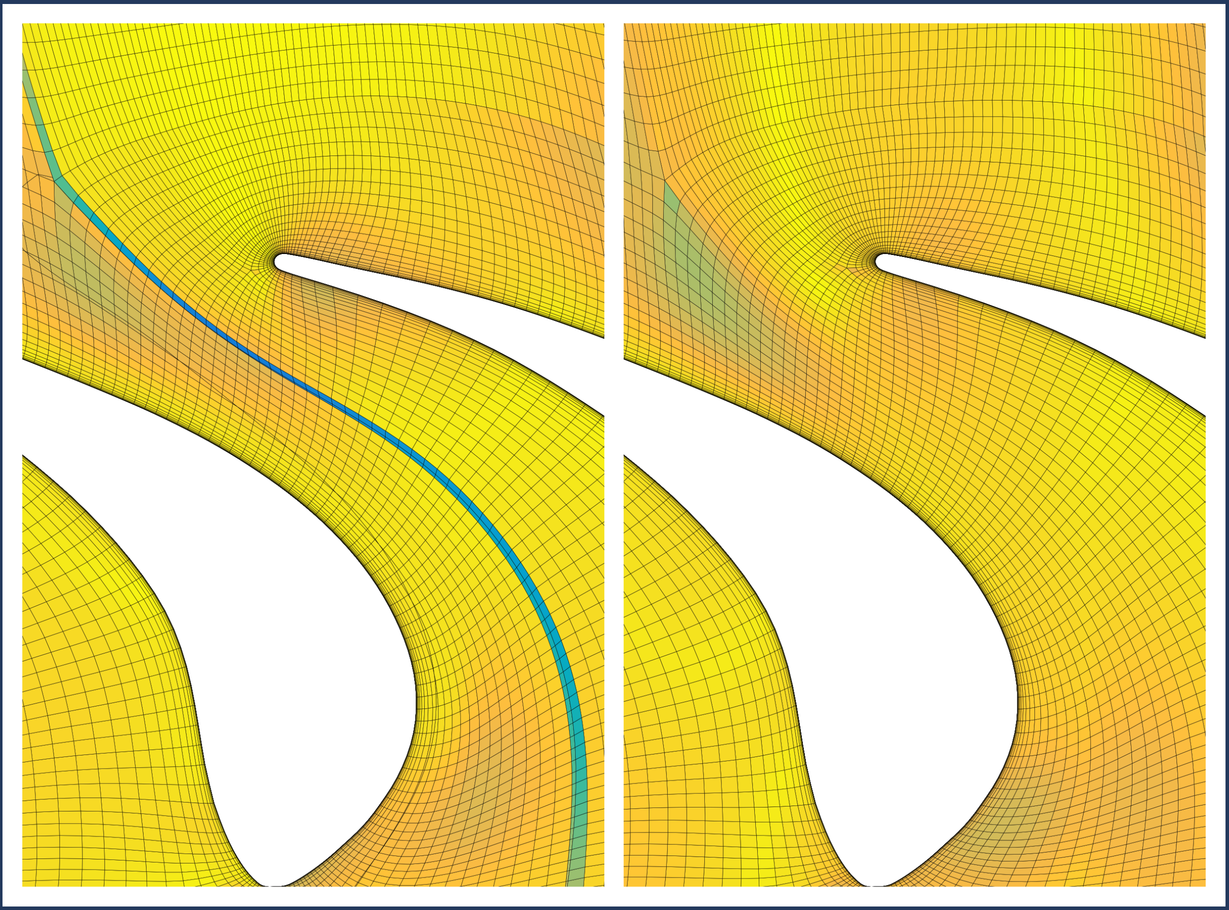
Mesh Deformation Using Radial Basis Function Interpolation for Numerical Optimisation of Internal Flow Applications

Floyd Aston van Steen



Mesh Deformation Using Radial Basis Function Interpolation for Numerical Optimisation of Internal Flow Applications

by

Floyd Aston van Steen

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on July 10, 2023 at 14:00

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This thesis report is the result of my thesis project on sliding boundary radial basis function interpolation as a mesh deformation method. This project was embarked upon as part of the Master of Science degree at the Aerospace Faculty of Delft University of Technology.

First of all, I would like to thank my project supervisors Alexander van Zuijlen and Matteo Pini for their guidance and help throughout the duration of this project.

Furthermore, a special thanks go to my family and friends for their support and friendship, not only during my thesis project but during the entirety of my studies.

Abstract

Developments in computational capabilities and the always increasing demand for higher performance of internal flow applications has meant that Computational Fluid Dynamics (CFD) has become an essential tool within the design process. A key point of interest is to couple the fluid solver with numerical optimisation techniques in order to obtain a more automated design process that can handle a vast amount of different design aspects. The open source CFD suite SU2 has emerged as an enabler for aerodynamic shape optimisation involving a large number of design variables, due to its efficient, accurate and flexible discrete adjoint solver.

For the adjoint-based aerodynamic design optimisation of internal flow applications the deformation of the volumetric mesh has to be performed in an robust and efficient manner. Often small wall clearance gaps and periodic domains are encountered in internal flow domains, which could potentially lead to the deterioration of the mesh. Sliding boundary node methods can be applied in order to maintain the mesh quality in case of small wall clearance gaps. Additionally, periodic boundaries can be displaced in a periodic manner following the applied deformation in order to prevent low quality cells near the periodic interface. Therefore, it would be of interest to implement the sliding boundary node methods and periodic conditions in a Radial Basis Function (RBF) interpolation method, one of the most robust mesh deformation methods available.

Additionally, the computational efficiency should be considered, since high computational times should be prevented for large and complex three-dimensional cases with a high number of design variables.

The aim of this thesis project is therefore to develop a robust and computationally efficient mesh deformation method suitable within the discrete adjoint optimisation framework of SU2 for internal flow applications by means of developing an implementation of the Radial Basis Function interpolation method including sliding boundary node algorithms, periodic boundary conditions and data reductions methods.

The sliding is achieved by replacing the interpolation condition for the sliding nodes with a planar slip condition. Or alternatively, by freely displacing the sliding nodes based on the known deformation and subsequently projecting the sliding nodes back onto the boundary of the domain. The periodic displacement of the boundaries is ensured by making the distance function of the RBF periodic. The periodic nodes are then treated as internal nodes in order to allow them to move.

The developed RBF-SliDe (Radial Basis Function Sliding Deformation) tool is able to generate higher minimum mesh qualities as compared with the regular RBF interpolation method. The sliding of the boundary nodes reduces the degree of skewing of the mesh elements in case of drastic deformations, resulting in a higher minimum mesh quality. Furthermore, the introduction of the periodic displacement prevents low quality skewed or compressed mesh elements, as the periodic boundaries move along with the applied deformation when the considered object approaches the periodic boundary.

The Aachen turbine stator blade is considered as a realistic three-dimensional test case. For this stator blade an optimised geometry was available, which was obtained with an adjoint-based aerodynamic optimisation performed with SU2. Therefore, the resulting minimum mesh quality is compared to the one as obtained with the more conventional linear elasticity equation method as used in SU2. The minimum mesh quality obtained with the RBF-SliDe tool is nearly three times higher compared to the minimum mesh quality of the linear elasticity equations methods. This highlights the potential of the periodic sliding RBF interpolation method in terms of preserving the mesh quality.

Contents

Nomenclature

List of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
IDW	Inverse Distance Weighting
RBF	Radial Basis Function
RBF-DS	Direct sliding radial basis function interpolation
RBF-PS	Pseudo sliding radial basis function interpolation
SST	Shear-Stress-Transport

List of Symbols

α	Interpolation coefficient	[-]
α_k	Determinant Jacobian matrix	[-]
$\boldsymbol{\alpha}$	Interpolation coefficient vector	[-]
$\boldsymbol{\beta}$	Polynomial coefficient vector	[-]
$\boldsymbol{d}$	Deformation vector	[-]
$\boldsymbol{E}$	Residual vector	[-]
$\boldsymbol{n}$	Normal vector	
$\boldsymbol{t}$	Tangent vector	
$\boldsymbol{x}$	Point coordinate vector	[-]
δ	Distance function	[-]
ϵ	Nodal error	[-]
γ	Correction factor	[-]
λ	Periodic length	[-]
$\lambda_{i,j}^k$	Metric tensor elements	[-]
Φ	Interpolation matrix	
ϕ	Radial basis function	[-]
τ	Cell size ratio	[-]
θ_p	Periodic angle	[-]
ξ	Ratio of distance function to support radius	[-]
a_ϵ	Greedy level error ratio	[-]
A_k	Jacobian matrix	
c	Control nodes	
C_D	Drag coefficient	[-]
C_L	Lift coefficient	[-]
C_p	Pressure coefficient	[-]
f_{size}	Relative size mesh quality metric	[-]
f_{skew}	Skew mesh quality metric	[-]
f_{ss}	Relative size-skew mesh quality metric	[-]
in	Internal nodes	
k	Number of dimensions	[-]
m	Moving nodes	
N	Number of nodes	
P	Polynomial matrix	
p	Polynomial	[-]
q	Polynomial	[-]
r, θ, z	Cylindrical coordinates	
R	Correction radius	[-]
r	Support radius	[-]
S	Greedy level size	[-]
s	Interpolation function	[-]
se	Sliding edge nodes	
ss	Sliding surface nodes	
x, y, z	Cartesian coordinates	

List of Figures

List of Tables

1 Introduction

With the ever-increasing demand for higher performance and efficiency of internal flow applications such as fluid machinery, Computational Fluid Dynamics (CFD) has become indispensable within the design process. Historically, the design of for instance turbomachinery devices were performed using empirical relations, trial-and-error or experimental methods where the success of the design was highly dependent on the experience of the designer [1, 2]. The advancements made in computational capabilities has meant that CFD has become an essential tool in order to obtain an understanding of the underlying flow phenomena in a time and cost efficient manner compared to traditional experimental methods.

One main point of interest is to couple the fluid solver with numerical optimisation techniques in order to obtain a more automated design process that can handle a vast amount of different aspects of the design. Optimisation techniques are generally classified into stochastic or gradient-free methods and gradient-based methods. The stochastic methods search for a global optimum and only require the computation of the objective function. Therefore, these methods are computationally very expensive as many objective function evaluations and thus many flow solutions are required [2]. As a result stochastic methods are not viable for complex design problems involving a large amount of design variables.

The open source CFD suite SU2 [3] has emerged as an enabler for aerodynamic shape optimisation involving a large number of design variables, due to its efficient, accurate and flexible discrete adjoint solver [4]. The solver employs advanced algorithmic differentiation techniques in order to automatically derive the discrete adjoint solver [5]. This discrete adjoint design framework of SU2 has been used for various external flow applications. Additionally, recent works have demonstrated the applicability of this framework on turbomachinery applications. The discrete adjoint method has been used by Vitale et al. to show the improvements in aerodynamic performacne of single-blade organic Rankine cycle turbine cascades [6]. Rubino et al. showed the unsteady optimisation of two-dimensional cascades by involving the harmonic balance methods in the discrete adjoint solver [7]. Vitale et al. have extended the discrete adjoint solver to perform fully turbulent three-dimensional design optimisations of multistage turbomachinery [8]. SU2 will be used for this project since extensive development on turbomachinery design and optimisation has been performed by the Propulsion and Power group of the Delft University of Technology. As indicated by several recent projects performed involving SU2 [9, 10, 11, 12, 13, 14, 15].

One of the aspects involved in any aerodynamic design optimisation routine is the movement of the mesh with each iteration. The volumetric mesh has to deform to represent the geometrical changes of the design in each optimisation iteration. This can either be done by using mesh regeneration or by using mesh deformation methods. Generally, the mesh deformation deteriorates the mesh quality and could potentially lead to negative volume elements. A low quality mesh negatively impacts the convergence of the numerical optimisation and therefore the mesh deformation method has to be robust.

In addition to the robustness of the mesh deformation method, also the computational cost should be considered. For complex large three-dimensional cases with a high number of design variables the mesh deformation method used has to be efficient in order to prevent high computational times.

1.1. Mesh Deformation Methods

As highlighted in the previous section, the deformation of the mesh as a result of geometrical changes in the design with each iteration is an important factor in the aerodynamic design optimisation process.

Furthermore, within the context of adjoint-based optimisation a differentiable mesh deformation method is required. This can be performed by algorithmic differentiation as done in SU2 or by means of finite differences. Employing algorithmic differentiation imposes additional challenges as the mesh deformation method must also be suitable or made suitable for this kind of differentiation. In general there are three different approaches to deform a mesh. These are mesh regeneration, connectivity based schemes and point-by-point schemes. The main criteria that a mesh deformation method should satisfy are maintaining a high mesh quality for a large range of deformations and not being computationally too expensive. Therefore, the mesh deformation method has to be robust and efficient.

Mesh regeneration is an approach where the mesh is completely regenerated for each deformation step. It is a very robust method as the mesh properties are independent of the previous iterations. However, mesh regeneration is an expensive procedure and any changes in the mesh topology also affect the gradients of the objective function within the optimisation process. Furthermore, in case of mesh regeneration no algorithmic differentiation is possible. Thus, mesh regeneration is not suitable for deforming a mesh within an optimisation framework.

Mesh deformation methods based on the grid connectivity use the connectivity information of the grid to perform the mesh deformation. All connectivity-based schemes require to solve a system of equations involving all the internal grid points, thus making this approach computationally expensive in general.

The most well-known method is the spring analogy method [16], where it is assumed that the grid lines are modelled as springs with a stiffness inversely proportional to the length of the grid lines. The spring analogy method is able to deal with moderate deformations. The method is not suited for hanging nodes which are vertices of mesh elements that are not shared with the neighbouring mesh elements. An example of a hanging node is presented in Figure 1.1, where the hanging node is not part of the vertices of the neighbouring mesh element on the right. Imagine that each line segment is a spring and the hanging node is displaced to the left. The resulting equilibrium balance is disturbed since there is no neighbouring node and spring on the right of the hanging node to keep it in equilibrium. The removal of hanging nodes will increase the computational cost. Furthermore, possible grid line crossovers resulting in degenerate cells could potentially occur in case of large deformations.

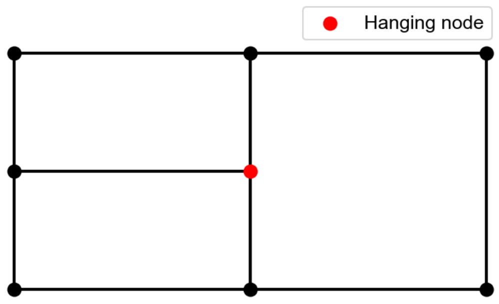


Figure 1.1: Example of a hanging node for a mesh with quadrilateral elements.

In order to prevent grid line crossovers the torsional spring [17, 18], semi-torsional spring [19, 20], ball-vertex [21] and ortho-semi-torsional spring [22] analogy methods have been introduced. These methods introduce either additional linear or torsional springs to prevent the inversion of an element. Of these methods the orto-semi-torsional spring analogy is the most computationally cost effective but more expensive than the linear spring analogy. Furthermore, hanging nodes also require manual removal for all of these methods.

A different connectivity-based method is the use of linear elastic equations [23]. The entire computational domain consisting of the volumetric grid is treated as a linear elastic element. This method allows for large deformations but comes at the cost of increased computational effort compared to the linear spring analogy.

Alternatively, the mesh deformation can be performed by applying the Laplacian smoothing method [24] or applying the biharmonic operator method [25]. For the former the Laplacian is applied to the grid nodes to smooth the information on the boundary of the domain to the interior of the domain. The Laplacian smoothing method is limited to small deformations. These limitations are due to the fact that the method results in a second order system and therefore only the mesh position or normal mesh spacing can be specified. To circumvent this limitation the biharmonic operator was applied to the grid nodes resulting in a fourth order system. This method yields similar results to the spring analogy method. However, the computational cost is roughly doubled.

In contrast to the connectivity-based schemes, a point-by-point scheme moves each point individually based on its position in space. Generally, point-by-point schemes are computationally more efficient than connectivity-based schemes since a set of control points is used to find the movement of the mesh. However, the use of a reduced set of control points does introduce interpolation errors. One point-by-point scheme is the Delauney graph method [26], which is computationally more efficient compared to the spring analogy method. The method is best suited for small deformations. Larger deformations might result in poor quality meshes which requires the regeneration of the Delauney graph which increases the computational cost.

The Radial Basis Function (RBF) interpolation [27] is a widely applied mesh deformation method. An interpolation function is used to interpolate the known displacements at the boundary of the domain to the internal nodes. There are various radial basis functions that can be used for the interpolation. These are classified in either globally or locally supported radial basis functions. Greedy algorithms have been introduced as a data reduction measure which has reduced the computational cost of the RBF interpolation significantly [28].

Alternatively, the inverse distance weighting method can be used [29]. This is an explicit method used for the multivariate interpolation of scattered data points. The inverse distance weighting method can be used as a tool to improve the cell orthogonality near surface boundaries due to its ability to treat the rotations of the boundary nodes separately. Witteveen demonstrated that for the 3D AGARD 445.6 aeroelastic wing the inverse distance weighting method is computationally more efficient compared to the RBF interpolation method [29]. On the other hand, the RBF interpolation showcased a slightly better mesh quality. It should be noted that it is not specified which form of the RBF interpolation was used.

From the aforementioned mesh deformation methods the RBF interpolation was chosen as a mesh deformation technique suitable within the aerodynamic design optimisation framework. The connectivity-based schemes are too computationally expensive since a system of equations involving all internal nodes has to be solved. The RBF interpolation is widely applied due to its robustness and in combination with data reduction techniques it is able to match the computational efficiency of the inverse distance weighting method. A final argument in favour of the RBF interpolation is that in previous works sliding boundary node methods and periodic boundary deformations have been accommodated within the RBF interpolation as will be detailed in the next section.

Some instances of radial basis function interpolation in conjunction with SU2 are found in literature. Among those is the simulation of aircraft icing using radial basis function interpolation with a separate RBF Morph tool [30] and by means of a direct implementation in SU2 [30]. A single instance was found where the RBF interpolation was used as mesh deformation method within the optimisation framework of SU2. In this case an airfoil, a three-dimensional wing and three-dimensional wing geometry with winglet were optimised using RBF interpolation [31], where a multilevel greedy data reduction algorithm was used. However, no literature exists on using the mesh deformation method for internal flow applications as per knowledge of the author.

1.2. Challenges of Mesh Deformation for Internal Flows

In the field of internal flow applications small wall clearance gaps are often encountered, consider the tip-blade clearance in turbomachines for example. These small wall clearances pose problems in maintaining high mesh quality in case of drastic deformations. These deformations can cause a large displacement of the mesh nodes near the boundary. For RBF interpolation the boundary nodes will have a zero displacement and therefore the relative motion in the mesh will introduce a high degree of skewing of the mesh elements and thus poor mesh quality in this region. To prevent this problem planar slip conditions are introduced to the RBF interpolation in order to allow the boundary nodes to slide along their respective boundary [32]. However, the planar slip conditions couple the spatial directions of the RBF interpolation increasing its computational cost significantly. Whereas, for the regular RBF interpolation each spatial direction could be solved separately. As a more computationally efficient alternative a pseudo sliding method was proposed by Mathew [33]. In this case the sliding nodes are freely displaced and projected back onto the boundary. This proved to increase only slightly the computational cost compared to a regular RBF interpolation. However, this method failed for three-dimensional problems. Therefore, it would be of interest to investigate possible adaptations for more complex cases.

A second aspect to consider in the mesh deformation is the treatment of periodic domains. Periodicity is frequently encountered in internal flow applications. For instance in turbomachinary a single isolated blade passage can represent the entire domain with the use of periodic boundary conditions. Similarly to small wall clearance gaps, drastic deformations could bring the blade or any other considered object in proximity of the periodic boundary. Although no physical wall is present at the periodic boundary, the mesh quality deteriorates due to the compression of the mesh elements in that region. Moreover, the cells at the corresponding other periodic boundary are also affected as they elongate due to deformation. Mesh quality can be maintained by allowing these boundaries to move in a periodic fashion. Research performed by De Keyser [34], indicated that significantly higher minimum mesh qualities can be achieved by allowing the periodic boundaries to displace periodically while deforming the mesh.

1.3. Research Objective

In light of the previous considerations it would be of interest to combine the RBF interpolation with the adaptations for sliding boundary nodes and moving periodic nodes along with data reduction techniques in order to create a robust and efficient mesh deformation tool to be used within the SU2 optimisation framework. Therefore, the following research objective was formulated:

"The objective of this research is to achieve a robust and computationally efficient mesh deformation method suitable within the discrete adjoint optimisation framework of SU2 for internal flow applications by means of developing an implementation of the Radial Basis Function interpolation method including sliding boundary node algorithms, periodic boundary conditions and data reductions methods."

Based on this research objective the following main research question was derived:

What improvements can be made in terms of mesh quality and computational efficiency by implementing radial basis function interpolation as mesh deformation technique, including sliding boundary nodes, periodic boundaries conditions and data reduction techniques, within the optimisation framework of SU2?

In order to answer this research question the following research sub-questions were formulated:

- How does the use of sliding boundary node methods in the RBF interpolation affect the mesh deformation performance compared to the performance of a regular RBF interpolation?
  - What differences can be observed in terms of mesh quality?
  - What differences can be observed in terms of computational costs?
- How does the use of periodicity in the RBF interpolation affect the mesh deformation performance compared to the performance of a regular RBF interpolation?
  - What differences can be observed in terms of mesh quality?
  - What differences can be observed in terms of computational costs?
- What is the effect of using periodicity on the RBF interpolation with sliding boundary node methods?
- How does the RBF interpolation with sliding boundary nodes and periodicity compare to the mesh deformation method using the linear elasticity equations as used in SU2?
  - What differences can be observed in terms of mesh quality?
  - What is the impact on the flow solution as found by the CFD solver?
  - What is the impact on the efficiency of the CFD solver?

1.4. Outline Report

The thesis report is structured as follows. The methodology used to be able to answer the formulated research questions is described in detail in Chapter 2. The numerical framework of the developed mesh deformation tool is discussed in Chapter 3. The test cases used within this project for evaluating the performance of the mesh deformation are presented in Chapter 4. The results for the two-dimensional and three-dimensional test cases are given in Chapter 5 and Chapter 6. The conclusions drawn from the results are discussed in Chapter 7, followed by recommendations for future work in Chapter 8.

2 Methodology

This chapter offers an in depth description of the methods and algorithms used in the developed RBF interpolation mesh deformation tool. The first section details the regular radial basis function interpolation, which is the main method to which alterations and adaptions are made to involve the other techniques of sliding nodes, periodicity and data reduction algorithms. Secondly, the radial basis functions are discussed in Section 2.2. The inclusion of periodicity is addressed in Section 2.3. The sliding algorithms are discussed in detail in Section 2.4. This is followed by a description on the data reduction techniques and space partitioning data structures used in the projection algorithm in Section 2.5 and Section 2.6. Lastly, the mesh quality metrics used for evaluating the quality of the mesh are presented in Section 2.7.

2.1. Regular Radial Basis Function Interpolation

The RBF interpolation is a method that can be used to determine the displacement of the internal nodes based on a known boundary point displacement [27]. The displacement of the entire domain is approximated by an interpolation function s, which is a summation of RBF's as shown in Equation 2.1.

$$s(\boldsymbol{x}) = \sum_{j=1}^{N_c} \alpha_j \phi(\|\boldsymbol{x} - \boldsymbol{x}_{c_j}\|) + p(\boldsymbol{x}) \tag{2.1}$$

Where N_c is the number of control nodes, in this case the boundary nodes, α_j are the interpolation coefficients, ϕ is the RBF with respect to the Euclidean distance $\|\boldsymbol{x} - \boldsymbol{x}_{c_j}\|$ and p is a polynomial. This polynomial is required in order to obtain a unique interpolant for conditionally positive definite RBF's [35]. In case of positive definite RBF's the polynomial can be omitted from Equation 2.1. The interpolation coefficients α_j and the polynomial p are established by the interpolation conditions stated in Equation 2.2 and Equation 2.3.

$$s(\boldsymbol{x}_{c_j}) = \boldsymbol{d}_{c_j} \tag{2.2}$$

$$\sum_{j=1}^{N_c} \alpha_j q(\boldsymbol{x}_{c_j}) = 0 \tag{2.3}$$

The first condition states that the evaluation of the interpolation function at the control nodes should be equal to the known displacement of the control nodes. The second condition has to be satisfied for all polynomials with a degree less or equal to the degree of the polynomial p. The system to be solved for the interpolation can then be expressed as given in Equation 2.4.

$$\begin{bmatrix} \boldsymbol{d}_c \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_{c,c} & P_c \\ P_c^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{2.4}$$

Here $\boldsymbol{d}_c$ is the vector containing the known displacement of the control nodes, $\Phi_{c,c}$ is an $N_c \times N_c$ matrix containing the RBF evaluations $\phi_{c_i,c_j} = \phi(\|\boldsymbol{x}_{c_i} - \boldsymbol{x}_{c_j}\|)$ and the vector $\boldsymbol{\alpha}$ contains the interpolation coefficients. P is the polynomial matrix and $\boldsymbol{\beta}$ is the polynomial coefficient vector. In order to reduce the computational cost it was chosen to use positive definite RBF's for this project. In which case the polynomial can be omitted and the resulting interpolation system reduces to:

$$\boldsymbol{d}_c = \Phi_{c,c}\boldsymbol{\alpha} \tag{2.5}$$

Once the interpolation coefficients α are determined, the displacement of the internal nodes is found by evaluating the interpolation function at these nodes as shown in Equation 2.6:

$$\boldsymbol{d}_{in_i} = s(\boldsymbol{x}_{in_i}) = \sum_{j=1}^{N_c} \alpha_j \phi(\|\boldsymbol{x}_{in_i} - \boldsymbol{x}_{c_j}\|) \tag{2.6}$$

This evaluation can be performed for each spatial coordinate separately and can be expressed as the system stated in Equation 2.7, where $\Phi_{in,c}$ has size $N_{in} \times N_c$.

$$\boldsymbol{d}_{in} = \Phi_{in,c}\boldsymbol{\alpha} \tag{2.7}$$

2.2. Radial Basis Functions

There is a variety of radial basis functions available in literature for interpolation purposes. Typically, these RBF's are categorised into two groups: functions with compact support and functions with global support. RBF's with compact support only influence the nodes that are within the support radius r of a node centre, as summarised by the following property:

$$\phi(\xi) = \begin{cases} f(\xi) & 0 \le \xi \le 1, \\ 0 & \xi > 1 \end{cases} \tag{2.8}$$

Where $\xi = \delta/r$ and δ is the distance to the node centre. As a result, compact supported RBF's can be scaled by altering the support radius. In general, higher values for the support radius will result in more accurate solutions, as more nodes will be influenced by a given displacement. However, this will result in denser matrix systems, whereas smaller support radii yield sparser matrix systems that can be solved more efficiently. The influence of RBF's with global support is not limited to a support radius but extends throughout the domain.

An extensive study on mesh quality and computational efficiency for fourteen different RBF's was performed by De Boer, Van der Schoot and Bijl [27]. The compact supported RBF's considered in this study are given in Table 2.1 and were introduced as such by Wendland [36]. The global supported RBF's considered are given in Table 2.2.

Of these function the CP C^2, CTPS C^1, CTPS C^2_a, CTPS C^2_a and the thin plate spline functions achieved high mesh qualities after deformation. However, in terms of efficiency the Wendland CP C^2 function performed best followed closely by the thin plate spline. The CP C^2 is a positive definite function, which allows the omittance of the polynomial in Equation 2.1, whereas the thin plate spline is not. Therefore, it was decided to use the Wendland CP C^2 compact support RBF for this project. This RBF is shown as a function of ξ for different support radii in Figure 2.1. The variation of the Wendland CP C^2 RBF with respect to the point $(0,0)$ and $r = 1$ is shown in Figure 2.2 for a two-dimensional x,y-plane. As support radius r a value of 2.5 times the characteristic domain length will be used as this will result in optimal robustness and mesh quality [27].

Table 2.1: Compact supported radial basis functions considered by De Boer et al. [27].

Name	$f(\xi)$
CP C^0	$(1-\xi)^2$
CP C^2	$(1-\xi)^4(4\xi+1)$
CP C^4	$(1-\xi)^6(\frac{36}{3}\xi^2+6\xi+1)$
CP C^6	$(1-\xi)^8(32\xi^3+25\xi^2+8\xi+1)$
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 2.2: Global supported radial basis functions considered by De Boer et al. [27].

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	$e^{(-x^2)}$

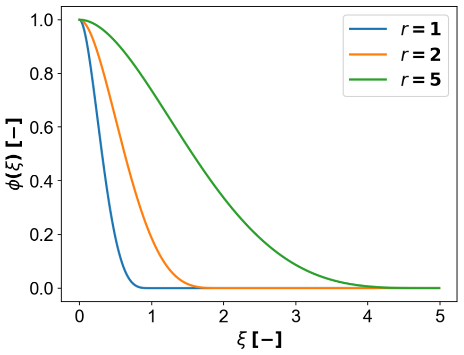


Figure 2.1: Wendland CP C^2 function as a function of $\xi = \delta/r$, for support radius values of 1, 2 and 5.

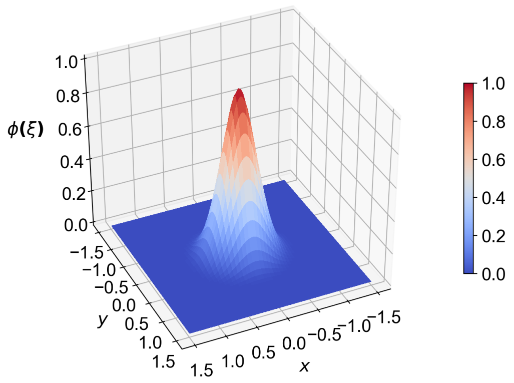


Figure 2.2: Wendland CP C^2 with $r = 1$ centered at $(0, 0)$ for a two-dimensional x,y-plane.

2.3. Periodic Boundaries

In case of problem domains that involve a repeated pattern in a given direction or angle, periodic boundary conditions can be used on the repetitive part to represent the whole domain. This significantly reduces the computational cost. The periodic boundary conditions ensure that the flow properties are identical when crossing these boundaries.

These periodic boundaries can be kept stationary, however this could potentially lead to poor mesh quality regions in case the deformation of the considered object (e.g. turbomachinery blade) nears a periodic boundary. The volumetric cells of the mesh would be compressed in this area, whereas the cells at the accompanying periodic boundary will be elongated. This phenomenon can be compared to a small wall clearance gap for non-periodic boundaries, however due to the periodic nature of the domain, no wall is present. It would therefore be beneficial to allow these periodic boundaries to be displaced in a periodic manner with the mesh deformation method to maintain mesh quality.

De Keyser investigated two methods for achieving the periodic displacement of the periodic boundaries in an RBF interpolation [34]. One method relied on altering the interpolation conditions to ensure that a periodic node pair has the same displacement. In a second approach, also known as the periodic distance method, the RBF distance function was made periodic. The latter method proved to be better in terms of computational efficiency while obtaining the same level of mesh quality. Furthermore, it yielded a smoother transition over the periodic boundary. Therefore, this approach is selected as a way to displace the periodic boundary nodes.

As mentioned, the RBF is periodic with the periodic distance method. In the interpolation function, restated in Equation 2.9 for a compact supported function, the radial basis function ϕ depends on the distance δ, which is the Euclidean distance. The distance function will be made periodic with the use of a sine function. When allowing the nodes on the periodic boundaries to displace in a periodic manner they no longer serve as control nodes within the RBF interpolation. Instead these nodes are treated as internal nodes and their displacement is found by evaluating the interpolation function as described in Section 2.1.

$$s(\boldsymbol{x}) = \sum_{j=1}^{N_c} \alpha_j \phi(\delta / r) \tag{2.9}$$

Depending on whether the domain is periodic in a translational or rotational sense different formulations of the periodic distance are used.

2.3.1. Translational Periodicity

For translational periodic boundaries the Euclidean distance is given as:

$$\delta = \sqrt{\|\boldsymbol{x} - \boldsymbol{x}_x\|} = \sqrt{(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2} \tag{2.10}$$

This distance function can be made periodic by applying a sine function in the specific periodic direction. If for instance the problem is periodic in the y-direction over a distance λ, the sine function, as given in Equation 2.11, is used to evaluate the y-distance to a given control node [37]. This periodic distance function is plotted as a function of distance $(y - y_c)$ in Figure 2.3.

$$\delta_y = \frac{\lambda}{\pi} \text{sin} \left((y - y_c) \frac{\pi}{\lambda} \right) \tag{2.11}$$

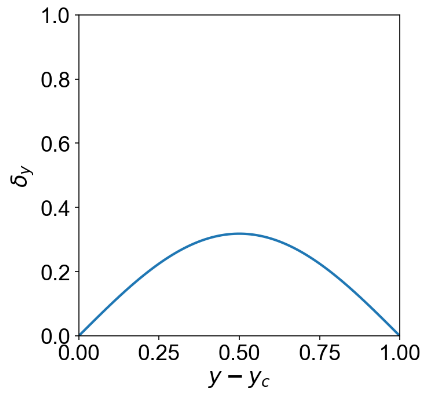


Figure 2.3: Periodic distance function in y-direction with $\lambda = 1$.

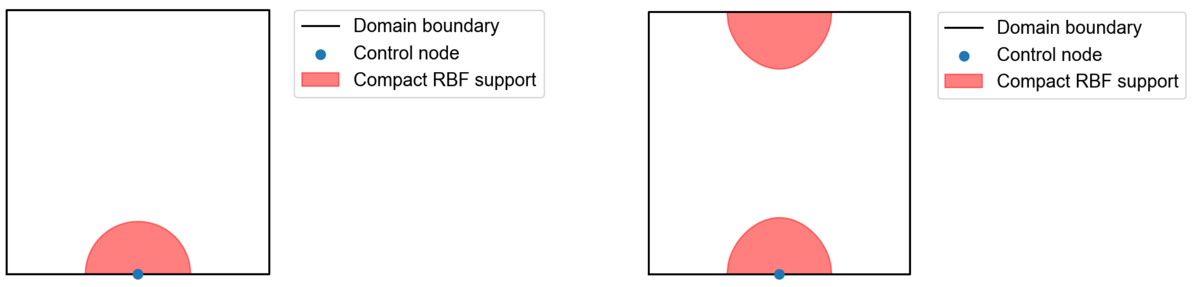
For nodes in the vicinity of the control nodes it is desirable to maintain the same properties in all directions and not to be affected by the periodicity. For this purpose the factor λ/π is included such that for small values of $(y - y_c)$, the small angle approximation will result in:

$$\delta_y = \frac{\lambda}{\pi} \sin\left[(y - y_c)\frac{\pi}{\lambda}\right] \approx \frac{\lambda}{\pi}(y - y_c)\frac{\pi}{\lambda} = y - y_c \tag{2.12}$$

Which is simply the Euclidean distance in y-direction. Due to the nature of the sine, the same behaviour will occur as $y - y_c$ approaches λ. Therefore, the nodes in the vicinity of the control nodes are not affected by the periodicity on both sides of the periodic boundary. The resulting distance function for problems involving translational periodicity in y-direction is presented in Equation 2.13, with similar functions for the other directions.

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

The effect of using the periodic distance function is demonstrated in Figure 2.4 for a square domain periodic in y-direction and a compact supported RBF. Figure 2.4a shows a single control node on the boundary of a non-periodic domain along with the region that falls within in the support radius of the RBF. Using a periodic distance function will result in the situation as presented in Figure 2.4b, where it can be seen that the compact support of the RBF extents to the corresponding periodic boundary. It can also be noted that the shape of the compact support region is oval for a periodic distance function. The reason for this is that within the distance function, the Euclidean distance is no longer considered in the periodic direction because of the inclusion of the sine function.



(a) Non-periodic **(b)** Periodic

Figure 2.4: Influence of the distance functions on the RBF interpolation for a square domain.

2.3.2. Rotational Periodicity

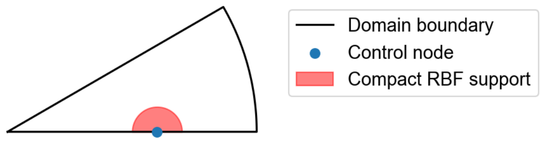
In case of rotational periodic boundaries the distance function is evaluated in either polar or cylindrical coordinates depending on the number of dimensions of the domain. The Euclidean distance in cylindrical coordinates is shown in Equation 2.14. Its derivation is given in Appendix A.

$$\delta = \sqrt{r^2 + r_c^2 - 2rr_c \cos(\theta - \theta_c) + (z - z_c)^2} \tag{2.14}$$

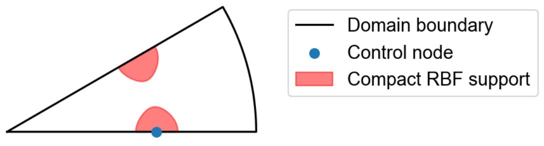
Similar to the translational periodicity, a sine function is introduced in order to make the distance function periodic over an angle θ_p as seen in Equation 2.15.

$$\delta = \sqrt{r^2 + r_c^2 - 2rr_c \cos\left(\frac{\theta_p}{\pi} \sin\left[(\theta - \theta_c)\frac{\pi}{\theta_p}\right]\right) + (z - z_c)^2} \tag{2.15}$$

Figure 2.5 shows the effect of using a periodic distance function on a compact supported RBF for a rotational periodic domain. Similarly to Figure 2.4, the influence of the RBF extends to both sides of the periodic interface for a periodic distance function. Whereas, for the non-periodic distance function the compact support region of the RBF is limited to the region within the support radius with respect to the control node.



(a) Non-periodic



(b) Periodic

Figure 2.5: Influence of the distance functions on the RBF interpolation for a wedge shaped domain.

2.4. Sliding Algorithms

Sliding boundary algorithms are introduced in an effort to reduce the skewness of the mesh elements in case of deformations occurring in close proximity of the boundaries of the domain. This section describes the two methods of boundary node sliding. With the direct sliding method a planar slip condition is used as an alternative interpolation condition [32]. For the pseudo sliding method a free displacement followed by a projection onto the boundary allows the nodes to slide along their respective boundary [33].

2.4.1. Direct Sliding Method

The direct sliding method is a way of performing the RBF interpolation while allowing a subset of the boundary nodes to slide by altering the interpolation conditions as used in the regular RBF interpolation. In a regular RBF interpolation the control nodes are the boundary nodes with known displacements. Typically part of these boundary nodes are stationary as they have zero displacement and prevent the internal points to move out of the domain. Allowing these nodes to slide along their respective boundary would be beneficial in terms of mesh quality for regions with small wall clearances. However, this would make part of the interpolation conditions as introduced for the regular RBF interpolation obsolete for these sliding nodes.Therefore, part of the interpolation conditions are altered to allow for the sliding of the stationary boundary nodes.

A subdivision of the control nodes is required for the different kinds of interpolation conditions. The control nodes are now divided into moving nodes with known displacements and sliding nodes. Depending on whether the problem dealt with is two or three-dimensional the sliding nodes are divided into sliding edge and sliding surface nodes, as there will be no boundary surfaces for two-dimensional domains. The division of the control nodes is summarised in Figure 2.6.

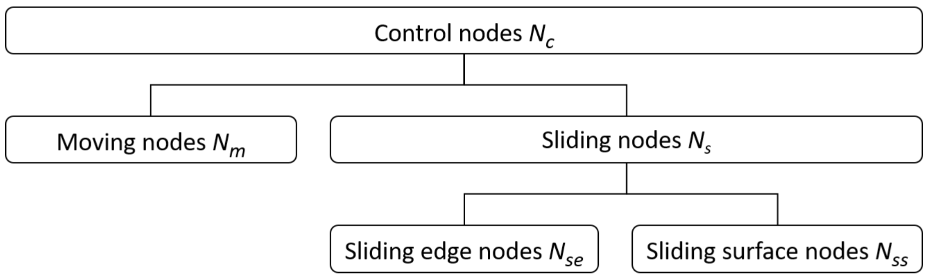


Figure 2.6: Nodal division for the direct sliding algorithm.

Moving Nodes

The interpolation conditions for the moving nodes is unchanged compared to the regular RBF interpolation and is restated in Equation 2.16 for the moving nodes.

$$\boldsymbol{d}_{m_i} = \sum_{j=1}^{N_c} \alpha_j \phi(\|\boldsymbol{x}_{m_i} - \boldsymbol{x}_{c_j}\|) \qquad \forall i \in \{1, ..., N_m\} \tag{2.16}$$

Sliding Edge Nodes

For the sliding edge nodes new interpolation conditions are introduced. Since these nodes are only allowed to slide along their respective edge there cannot be any displacement normal to this edge. Therefore, a zero normal displacement condition is defined as the dot product of the sliding edge node displacement with the a normal vector of the line segment which should equal zero:

$$s(\boldsymbol{x}_{se}) \cdot \boldsymbol{n}_{se} = 0 \tag{2.17}$$

Where inserting the interpolation function $s(\boldsymbol{x})$ results in:

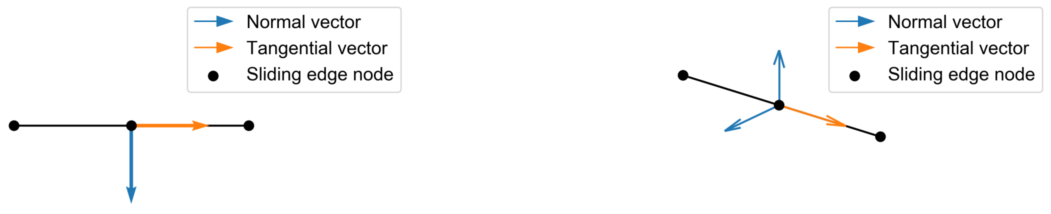
$$\begin{aligned}\sum_{j=1}^{N_c} \alpha_{j_x} \phi(\|\boldsymbol{x}_{se_i} - \boldsymbol{x}_{c_j}\|) n_{se,i_x} + \sum_{j=1}^{N_c} \alpha_{j_y} \phi(\|\boldsymbol{x}_{se_i} - \boldsymbol{x}_{c_j}\|) n_{se,i_y} \\ + \sum_{j=1}^{N_c} \alpha_{j_z} \phi(\|\boldsymbol{x}_{se_i} - \boldsymbol{x}_{c_j}\|) n_{se,i_z} = 0 \qquad \forall i \in \{1, ..., N_{se}\}\end{aligned} \tag{2.18}$$

For two-dimensional domains this condition should hold for the single normal vector of the sliding edge node, as defined in Figure 2.7a. In three dimensions two perpendicular normal vectors can be defined for each sliding edge node, as shown in Figure 2.7b, and naturally this condition should hold for both normal vectors.

A second condition is required to fully replace the regular RBF interpolation condition since the zero normal displacement condition only yields $(k-1)$ equations per single node for k dimensions. While k equations per sliding node are required to solve the interpolation system. The second condition states that there is a zero contribution to the interpolation function tangent to the line segment. This ensures that nodes with the same line segment tangent vector have no effect on each other and prevents nodes to be moved out of the domain because of contributions of neighbouring nodes. The condition can be defined as the dot product of the interpolation coefficient with the tangent vector of the sliding edge node:

$$\alpha_i \cdot \boldsymbol{t}_{se,i} = \alpha_{i_x} t_{se,i_x} + \alpha_{i_y} t_{se,i_y} + \alpha_{i_z} t_{se,i_z} = 0 \qquad \forall i \in \{1, ..., N_{se}\} \tag{2.19}$$

The tangential vectors of the sliding edge nodes are also indicated in Figure 2.7.



(a) Two-dimensional **(b)** Three-dimensional

Figure 2.7: Definitions of the normal and tangential vectors for the sliding edge nodes.

Sliding Surface Nodes

The conditions set for the sliding surface nodes are similar to those of the sliding edge nodes. The only difference being that for a boundary surface two perpendicular vectors tangent to its plane can be defined and a single normal vector as indicated in Figure 2.8, whereas a line segment has two normal vectors and a single tangent vector.

As a result there is only a single zero normal displacement condition, as given in Equation 2.20, for the sliding surface nodes.

$$\begin{aligned}\sum_{j=1}^{N_c}\alpha_{j_x}\phi(\|\boldsymbol{x}_{ss_i}-\boldsymbol{x}_{c_j}\|)n_{ss,i_x}+\sum_{j=1}^{N_c}\alpha_{j_y}\phi(\|\boldsymbol{x}_{ss_i}-\boldsymbol{x}_{c_j}\|)n_{ss,i_y}\\ +\sum_{j=1}^{N_c}\alpha_{j_z}\phi(\|\boldsymbol{x}_{ss_i}-\boldsymbol{x}_{c_j}\|)n_{ss,i_z}=0 \qquad \forall i\in\{1,...,N_{ss}\}\end{aligned} \tag{2.20}$$

The zero tangential contribution condition is presented in Equation 2.21, and should hold for both tangent vectors of the sliding surface node.

$$\alpha_{i_x}t_{ss,i_x}+\alpha_{i_y}t_{ss,i_y}+\alpha_{i_z}t_{ss,i_z}=0 \qquad \forall i\in\{1,...,N_{ss}\} \tag{2.21}$$

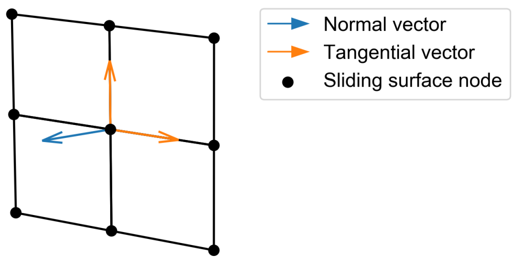


Figure 2.8: Definitions of the normal and tangential vectors for the sliding surface nodes.

Interpolation System

The interpolation system that results from the altered interpolation conditions for the direct sliding method is given in Equation 2.22. Where the indices m, e and s indicate whether moving, sliding edge or sliding surface nodes are considered. Thus, Φ_{me} is the interpolation matrix of size $N_m \times N_{se}$ containing the RBF evaluations $\phi_{m_i,se_j} = \phi(\|\boldsymbol{x}_{m_i} - \boldsymbol{x}_{se_j}\|)$ for $i = \{1, ..., N_m\}$ and $j = \{1, ..., N_{se}\}$. The diagonal matrices n and t contain the elements of the normal and tangential vectors in their respective spatial direction.

From the modified interpolation conditions it is evident that the spatial directions are coupled through the inclusion of the dot product with the normal and tangent vectors. Therefore, this system cannot be solved for each spatial direction independently with an exception for the specific case where all normal and tangent vectors are aligned with the spatial directions of the coordinate system. Which is usually only the case for simple problems. When this simplification is not applicable the size of the resulting system is $kN_c \times kN_c$, which is k^2 times larger compared to the interpolation system of the regular RBF interpolation. Thus, this method for sliding boundary nodes has a large impact on the computational efficiency.

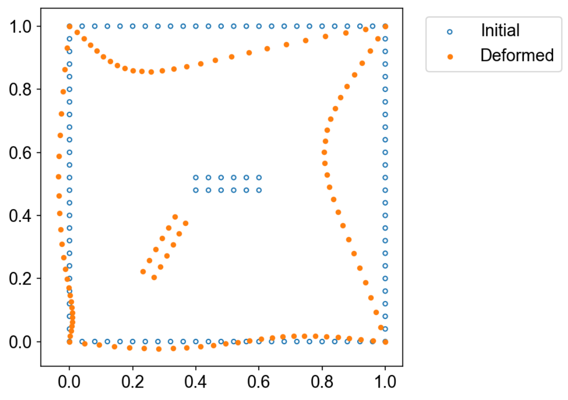
$$\begin{bmatrix}\boldsymbol{d}_{m_x}\\ \boldsymbol{d}_{m_y}\\ \boldsymbol{d}_{m_z}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\end{bmatrix}=\begin{bmatrix}\Phi_{mm} & \Phi_{me} & \Phi_{ms} & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \Phi_{mm} & \Phi_{me} & \Phi_{ms} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & \Phi_{mm} & \Phi_{me} & \Phi_{ms}\\ n_{se,1_x}\Phi_{em} & n_{se,1_x}\Phi_{ee} & n_{se,1_x}\Phi_{es} & n_{se,1_y}\Phi_{em} & n_{se,1_y}\Phi_{ee} & n_{se,1_y}\Phi_{es} & n_{se,1_z}\Phi_{em} & n_{se,1_z}\Phi_{ee} & n_{se,1_z}\Phi_{es}\\ n_{se,2_x}\Phi_{em} & n_{se,2_x}\Phi_{ee} & n_{se,2_x}\Phi_{es} & n_{se,2_y}\Phi_{em} & n_{se,2_y}\Phi_{ee} & n_{se,2_y}\Phi_{es} & n_{se,2_z}\Phi_{em} & n_{se,2_z}\Phi_{ee} & n_{se,2_z}\Phi_{es}\\ 0 & t_{se_x} & 0 & 0 & t_{se_y} & 0 & 0 & t_{se_z} & 0\\ n_{ss_x}\Phi_{sm} & n_{ss_x}\Phi_{se} & n_{ss_x}\Phi_{ss} & n_{ss_y}\Phi_{sm} & n_{ss_y}\Phi_{se} & n_{ss_y}\Phi_{ss} & n_{ss_z}\Phi_{sm} & n_{ss_z}\Phi_{se} & n_{ss_z}\Phi_{ss}\\ 0 & 0 & t_{ss,1_x} & 0 & 0 & t_{ss,1_y} & 0 & 0 & t_{ss,1_z}\\ 0 & 0 & t_{ss,2_x} & 0 & 0 & t_{ss,2_y} & 0 & 0 & t_{ss,2_z}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{m_x}\\ \boldsymbol{\alpha}_{e_x}\\ \boldsymbol{\alpha}_{s_x}\\ \boldsymbol{\alpha}_{m_y}\\ \boldsymbol{\alpha}_{e_y}\\ \boldsymbol{\alpha}_{s_y}\\ \boldsymbol{\alpha}_{m_z}\\ \boldsymbol{\alpha}_{e_z}\\ \boldsymbol{\alpha}_{s_z}\end{bmatrix} \tag{2.22}$$

One final element that should be considered is the sliding of boundary nodes over curved boundaries. It could be that due to the discritised nature of the boundary, nodes will slide off their respective edge or surface and outside of the domain. In that case a projection algorithm is employed to place these nodes back on the boundaries of the considered domain. This projection algorithm will be described in Section 2.4.3.

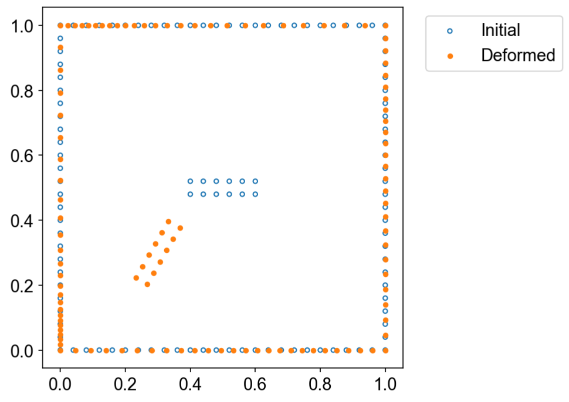
2.4.2. Pseudo Sliding Method

The pseudo sliding method was introduced by Mathew as a more computationally efficient alternative to the direct sliding method [33]. It was found that for direct sliding the spatial directions are coupled due to the interpolation conditions introduced to allow sliding. This resulted in a larger interpolation matrix compared to the regular RBF interpolation since in most cases it cannot be solved for each spatial direction separately.

The pseudo sliding method uses an approach where first the stationary nodes, i.e. the nodes with zero displacement for the regular RBF interpolation, are displaced freely as if they were internal nodes. This is followed by a projection back onto the boundary of the domain, where the new positions of the nodes makes it seem as if the nodes have slid over the boundary of the domain. The same projection algorithm as used for the treatment of curved boundaries in the direct sliding method will be used, which is described in Section 2.4.3. Finally, the change in nodal locations is used as a known displacement for a regular RBF interpolation as explained in Section 2.1. An example of the pseudo sliding method is shown in Figure 2.9 for a simple two-dimensional case involving a 25x25 mesh with a prescribed displacement to a 5x1 block in its centre. Figure 2.9a shows the initial location and freely displaced location of the boundary nodes. Figure 2.9b shows the final location of the boundary nodes after applying the projection algorithm.



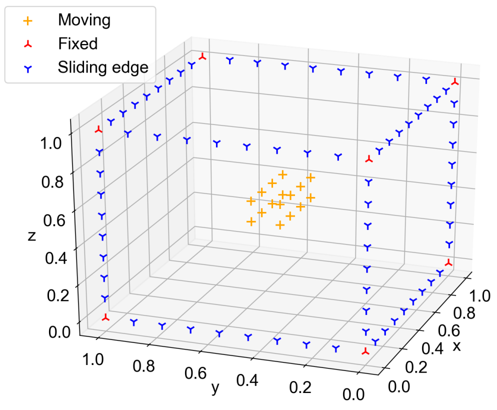
(a) Initial and freely displaced location of the boundary nodes.



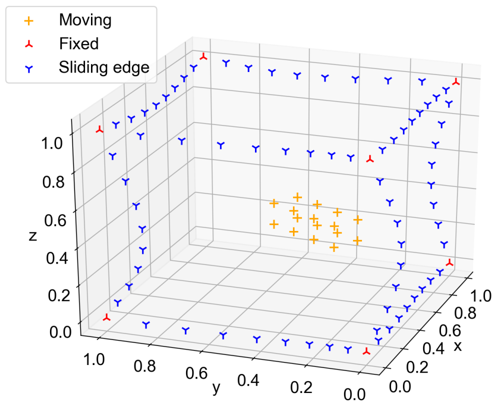
(b) Initial and projected location of the boundary nodes

Figure 2.9: 25x25 mesh example of the pseudo sliding method.

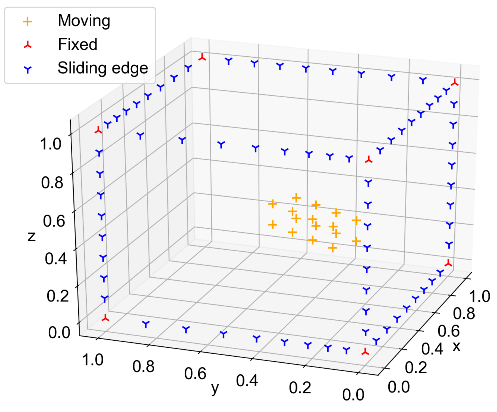
In the research performed by Mathew [33], it was found that for three-dimensional domains part of the sliding nodes moved out of the domain after performing the projection. As a result an intermediate step is proposed as a preventive measure to the pseudo sliding algorithm. Instead of projecting all the sliding edge and surface nodes in the same step, these projections are split into two steps. This process is visualised in Figure 2.10 for a hexahedral mesh with a moving block in the centre. Firstly, the sliding edge nodes are freely displaced and projected back onto their respective edges based on the displacement of the moving nodes of the block. The free displacement and projection of the edge nodes are shown in Figure 2.10b and Figure 2.10c. For the second step the sliding edge nodes are considered to be moving nodes and their projected positions are considered to be known displacements. The resulting nodal division for the second RBF interpolation is given in Figure 2.10d. The surface nodes are freely displaced based on the displacement of the moving nodes and projected back onto their respective boundary surface, as shown in Figure 2.10e and Figure 2.10f. The found projection of all sliding nodes can then be included as a known displacement in order to perform a regular RBF interpolation.



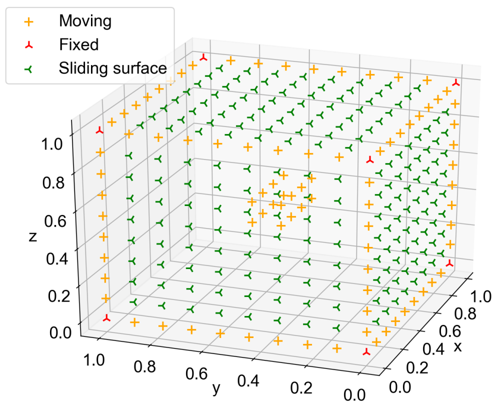
(a) Nodal division for first projection step



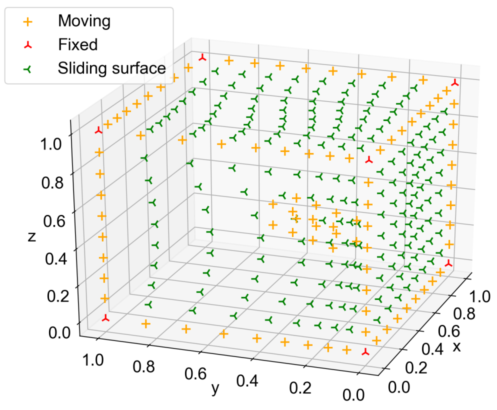
(b) Free displacement edge nodes



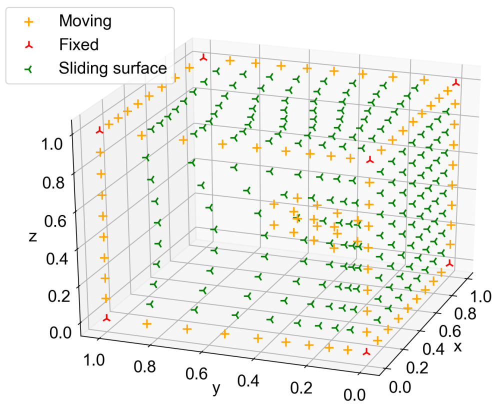
(c) Projected edge nodes



(d) Nodal division for second projection step



(e) Free displacement surface nodes



(f) Projected surface nodes

Figure 2.10: Projection steps of the pseudo sliding method for a three-dimensional RBF interpolation.

2.4.3. Projection

The projection algorithm used is an adapted version of a simple and easy to implement method as introduced by Mathew for the pseudo sliding method [33]. For a given sliding node a search for the nearest midpoint of the boundary segments is performed. For a sliding edge node this would be the nearest midpoint of the line elements that make up the boundary and for a sliding surface node this would be a midpoint of a boundary surface element. Then a projection in the normal direction(s) of the corresponding boundary element is performed in order to position the node on the boundary. This process is visualised in Figure 2.11 for a single sliding edge node.

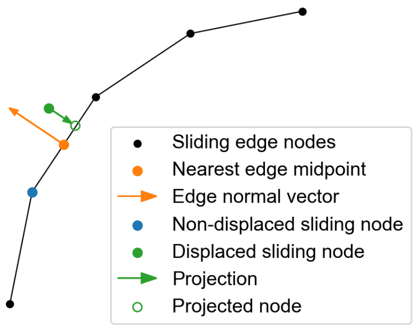
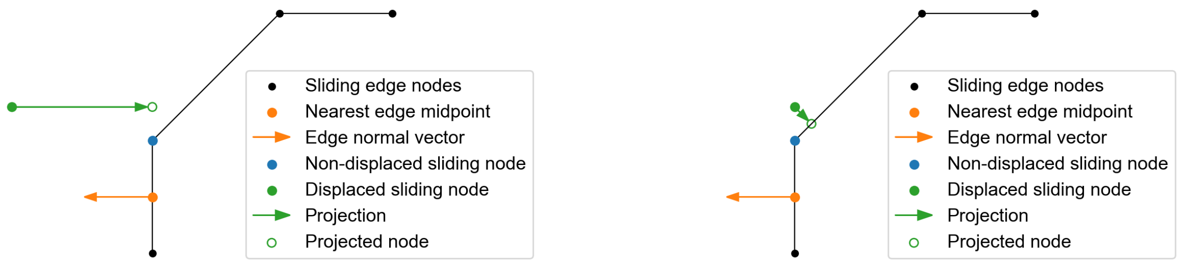


Figure 2.11: The projection algorithm applied to a displaced node.

The modification made to this projection algorithm is that in certain situations the projection algorithm should be done iteratively in order to ensure that the projected location is on the boundary of the domain. Consider a sliding edge node that in the first step of the pseudo sliding method is freely displaced to its new position shown in Figure 2.12. If the projection algorithm as proposed by Mathew would be executed then its new location, shown in Figure 2.12a, would not be located on the boundary. In this case a second projection is required in order to ensure that the point moves back onto the boundary, as seen in Figure 2.12b.



(a) First projection. **(b)** Second projection.

Figure 2.12: Displaced sliding node example requiring an iterative projection procedure.

2.4.4. Normal and Tangential Vector Determination

From the previous paragraphs on the direct sliding method and the projection algorithm it is evident that normal and tangential vectors have to be determined in order to perform these techniques. Since this project is intended for implementation within the SU2 environment, the vectors are established using information provided in an *.su2* mesh file. For the direct sliding method the normal and tangential unit vectors at the sliding nodes are required in the final interpolation system. This is different from the vectors required for the projection algorithm, where normal and tangential vectors at the midpoints of the boundary elements are required.

Sliding Node Vectors
For each sliding edge node, the two connecting nodes are determined. For the sliding surface nodes the surface elements are identified that include that node. This is done based on the information provided in the *.su2* file on the connectivity of the boundary elements. The sliding edge nodes will have a single vector tangent to its edge and two vectors normal to it, as seen in Figure 2.13. The tangent vector is determined by establishing vectors $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ as defined in Figure 2.14 and taking an Inverse Distance Weighted (IDW) average of these vectors. The resulting vector is then normalised in order to obtain a unit vector.

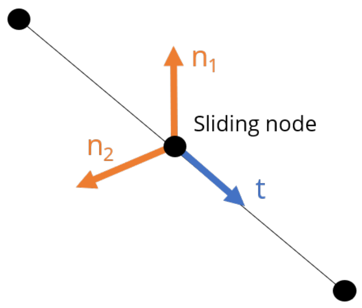


Figure 2.13: Sliding edge node tangential and normal vector definitions.

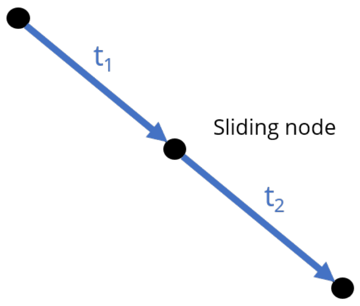


Figure 2.14: Tangential vectors of the line segments connected to the sliding edge node.

The first normal vector is found by using the fact that the dot product of two perpendicular vectors equals zero: $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0$. Let $\boldsymbol{v}_1$ be the tangent vector, it then follows that the first normal unit vector or $\boldsymbol{v}_2$ is:

$$\boldsymbol{w}_2 = \begin{bmatrix} v_{1_y} - v_{1_z} \\ v_{1_z} - v_{1_x} \\ v_{1_x} - v_{1_y} \end{bmatrix} \tag{2.23}$$

$$\boldsymbol{v}_2 = \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \tag{2.24}$$

The second normal vector is generated by the fact that the cross product of two vectors yields a vector perpendicular to both vectors. Therefore, an additional normal vector or $\boldsymbol{v}_3$ is found as follows:

$$\boldsymbol{w}_3 = \begin{bmatrix} v_{1_y}(v_{1_x} - v_{1_y}) - v_{1_z}(v_{1_z} - v_{1_x}) \\ v_{1_z}(v_{1_y} - v_{1_z}) - v_{1_x}(v_{1_x} - v_{1_y}) \\ v_{1_x}(v_{1_z} - v_{1_x}) - v_{1_y}(v_{1_y} - v_{1_z}) \end{bmatrix} \tag{2.25}$$

$$\boldsymbol{v}_3 = \frac{\boldsymbol{w}_3}{\|\boldsymbol{w}_3\|} \tag{2.26}$$

It should be noted that in case of a two-dimensional domain, there is only a single vector normal to an edge and can be determined by:

$$\boldsymbol{n} = \begin{bmatrix} t_y \\ -t_x \end{bmatrix} \tag{2.27}$$

Each sliding surface node will have a single normal vector and two tangent vectors as demonstrated for a boundary with quadrilateral elements in Figure 2.15. For each surface element that includes the sliding surface node, a surface normal vector is established based on two vectors emanating from any node and using the cross product property as stated in Equation 2.25. The normal vector at the sliding edge node is then obtained by taking an IDW of the surface normals based on the distance from the midpoint of the element to the sliding surface node. Both tangential vectors are established by using Equation 2.23 through Equation 2.26 to obtain two vectors perpendicular to the normal vector and themselves.

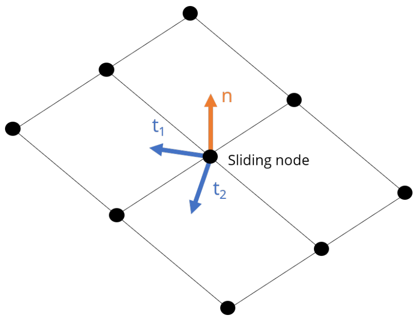


Figure 2.15: Surface edge node tangential and normal vector definitions.

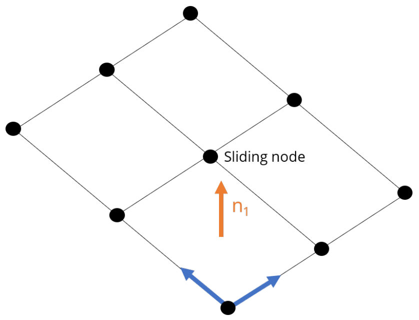


Figure 2.16: Definitions of the vectors used in determining the normal of a surface element.

Boundary Midpoints Vectors
For the projection algorithm the vectors at the midpoints of the boundary elements have to be determined. This will again be done using the boundary connectivity information provided in the *.su2* mesh file. For boundary edges two normal vectors have to be found as indicated in Figure 2.17. These are found be using the vector between the two nodes that are connected to a line segment and the relations stated in Equation 2.23 through Equation 2.26, in order to find two unit vectors perpendicular to it. For boundary surface elements two vectors emanating from any node are used in conjunction with Equation 2.25 to find the surface normal vector of that element.

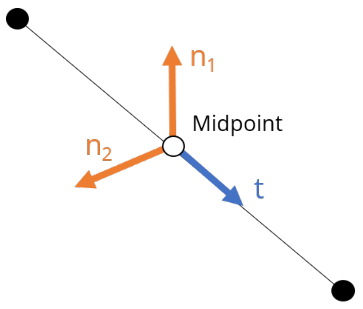


Figure 2.17: Surface edge node tangential and normal vector definitions.

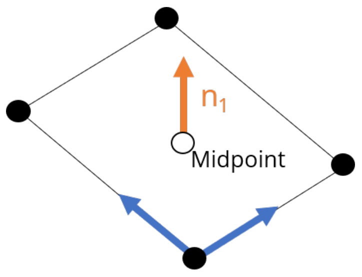


Figure 2.18: Definitions of the vectors used in determining the normal of a surface element.

2.5. Data Reduction Techniques

Although the RBF interpolation is a robust mesh deformation method, it is not the most computationally efficient among other mesh deformation methods. This is especially the case for problems involving a large number of control nodes, since the size of the interpolation matrix scales with N_c^2 for a regular RBF interpolation. The greedy algorithms, as introduced by Rendall and Allen [28], were developed in an effort to represent the interpolation with a reduced set of control nodes with an error-driven control point selection. The following sections describe in more detail the data reduction techniques used in this project.

2.5.1. Greedy Algorithms

The greedy algorithms are point-by-point based point selection methods based on the error signal of the unselected control nodes. They start with a single point with which the interpolation is performed and then points are selected with the largest error until the error falls below a set tolerance. Rendall and Allen have proposed three different versions of greedy algorithms [28].

In the greedy one point algorithm a local correction is applied to the interpolation to correct only for the node with the largest error. This process is repeated until the error falls below a certain threshold. As only a local correction is applied interpolation errors are introduced with this method. But the advantage is that it does not require any more solving of the interpolation system. The greedy full point algorithm solves the complete system each time a new maximum error node is added prior to selecting a further control node. This is slower compared to the one point algorithm due to the solving of the system each time a node is added. However, no interpolation error is introduced. The hybrid algorithm is a combination of the single point and full point algorithms, where the full point algorithm is performed every n cycles. For this project the greedy full point algorithm was selected as it was found by Mathew that the full point algorithm proved to be faster for most test cases and error tolerances [33]. Generally, due to the interpolation errors introduced with the single point algorithm a larger amount of control nodes needed to be selected compared to the full point algorithm, making it slower in comparison.

Normally, the error of the nodes would be the difference between the exact displacement and the resulting moving node displacement of the the interpolation, as shown in Equation 2.28.

$$\epsilon_m = \|\boldsymbol{d}_m - \boldsymbol{d}_{exact}\| \tag{2.28}$$

Due to the implementation of the sliding boundary node methods the way the error is computed differs for the sliding nodes. For the sliding nodes the error is found by applying the projection algorithm, as described in Section 2.4.3, to these nodes. The error is then defined as the difference between the position as found in the interpolation and the projected position:

$$\epsilon_{sliding} = \|\boldsymbol{d}_{sliding} - \boldsymbol{d}_{projected}\| \tag{2.29}$$

Each time a node is added to the reduced set of control nodes, the various interpolation matrices have to be updated. This is done by adding rows and columns to the appropriate interpolation matrices. Therefore, the interpolation matrices do not require a full regeneration each time a node is added.

Once the magnitude of the error satisfies the criterion set for the tolerance an explicit boundary correction is performed. The remaining error of these boundary nodes is the displacement that is required to position these nodes on their required location. Therefore, the error is applied as a deformation to the boundary nodes. This deformation is then dissipated into the mesh using a single point RBF interpolation with compact support as proposed by Rendall and Allen [38]. Before applying this procedure the error of the boundary nodes should be reduced sufficiently as otherwise the mesh quality is not ensured. The correction for the internal mesh nodes is found by evaluating Equation 2.30.

$$\Delta\boldsymbol{x}_{\text{corr}}(\boldsymbol{x}_{in_i}) = \boldsymbol{\epsilon}_{b_j}\phi\left(\frac{\|\boldsymbol{x}_{in_i} - \boldsymbol{x}_{b_j}\|}{R_{\text{corr}}}\right) \tag{2.30}$$

Where the correction of an internal node depends only on the error of the nearest boundary node $\boldsymbol{\epsilon}_{b_j}$ and the Euclidean distance to it $\|\boldsymbol{x}_{in_i} - \boldsymbol{x}_{b_j}\|$. Thus, a nearest neighbour search of boundary nodes is performed for each internal node and only a single RBF evaluation is performed. The influence of the correction is limited by the correction radius R_{corr}. The correction radius should be large enough to not affect the mesh quality and it should be small enough to limit the region affected by this local correction. In this case the correction radius will be linearly dependent on the maximum error on the boundary as shown in Equation 2.31 [39].

$$R_{\text{corr}} = \gamma\epsilon_b^{\text{max}} \tag{2.31}$$

Where the factor γ can be chosen to control the correction radius and should at least be larger than 1 to have a sufficiently smooth dissipation of the deformation due to the correction into the mesh. For this project it was decided to use $\gamma = 25$ for all considered test case. This should allow for a smooth dissipation of the correction into the mesh and will not cause the correction to affect a large region of the mesh.

2.5.2. Multilevel Greedy

To improve the greedy algorithm a double-edged greedy multilevel algorithm was developed [40]. Whereas the greedy methods as discussed in the previous section only consider the error magnitude for the control node selection, a double-edged method also includes the error direction in order to select an additional node. Firstly, a scan is performed to find the node with the largest error magnitude and its error direction $\boldsymbol{n}_{\epsilon_{max}}$ is determined. A second scan is performed to find the node with the largest error with the additional requirement that the angle between its error direction and $\boldsymbol{n}_{\epsilon_{max}}$ is greater than 90 degrees. Typically, this results in two nodes being selected that belong to a different edge or surface. Therefore, an additional node is selected per greedy cycle and a very concentrated distribution of selected nodes is prevented. This principle is not limited to the multilevel algorithm but can also be applied to a single level greedy algorithm.

A multilevel algorithm is designed by Wang et al. [40], that utilises this double-edged criterion in an effort to increase the efficiency of the aforementioned greedy algorithm. The RBF interpolation in matrix form can be expressed as:

$$\boldsymbol{d} = \phi\boldsymbol{\alpha} \tag{2.32}$$

Let a set of m nodes be selected from the control nodes to perform the RBF interpolation. Theoretically, the error is zero at these m nodes. When the resulting set of m interpolation coefficients is extended to all control nodes then:

$$\boldsymbol{\alpha}^{(0)} = [\alpha_{c_1}^{(0)}, ..., \alpha_{c_m}^{(0)}, \alpha_{c_{m+1}}^{(0)}, ..., \alpha_{c_n}^{(0)}]^T \tag{2.33}$$

Where the superscript 0 indicates the zeroth level and the coefficients $\alpha_{c_{m+1}}^{(0)}, ..., \alpha_{c_n}^{(0)}$ are zero because those where not included in the interpolation of the m points. The vector $\boldsymbol{\alpha}^{(0)}$ is an approximate solution of Equation 2.32 and its residual vector $\boldsymbol{E}$ can be defined as:

$$\boldsymbol{E}^{(0)} = d - \phi\boldsymbol{\alpha}^0 \tag{2.34}$$

Where $\boldsymbol{E}^{(0)}$ contains error information of all the control nodes and is nonzero apart from the m selected nodes. A new interpolation function can be defined with $\boldsymbol{E}^{(0)}$ as a deformation:

$$\boldsymbol{E}^{(0)} = \phi\boldsymbol{\alpha} \tag{2.35}$$

Using the double-edged method a new set of l nodes can be selected to approximate the interpolation function $\boldsymbol{E}^{(0)}$. If k describes the duplicate points of the set m and l then the interpolation coefficients for Equation 2.35 are expressed as:

$$\boldsymbol{\alpha}^{(1)} = [\alpha_{c_1}^{(1)}, ..., \alpha_{c_{m-k}}^{(1)}, \alpha_{c_{m-k+1}}^{(1)}, ..., \alpha_{c_{m-k+l}}^{(1)}, \alpha_{c_{m-k+l+1}}^{(1)}, ..., \alpha_{c_n}^{(0)}]^T \tag{2.36}$$

Where the interpolation coefficients $\alpha_{c_1}^{(1)}, ..., \alpha_{c_{m-k}}^{(1)}$ and $\alpha_{c_{m-k+l+1}}^{(1)}, ..., \alpha_{c_n}^{(0)}$ are zero. The residual of Equation 2.35 is:

$$\boldsymbol{E}^{(1)} = \boldsymbol{E}^{(0)} - \boldsymbol{\phi}\boldsymbol{\alpha}^{(1)} = \boldsymbol{d} - \boldsymbol{\phi}(\boldsymbol{\alpha}^{(0)} + \boldsymbol{\alpha}^{(1)}) \tag{2.37}$$

The Previous equation demonstrates that $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^{(0)} + \boldsymbol{\alpha}^{(1)}$ can be interpreted as interpolation coefficients for the RBF interpolation in Equation 2.32, with a reduced set of control nodes consisting of $m + l - k$ supporting points and a residual of $\boldsymbol{E}^{(1)}$. Or in other words, in each level an approximate interpolation is performed for the residual of the preceding level. Therefore, with each level the overall residual is reduced. This can be repeated until magnitude of $\boldsymbol{E}^{(n)}$ is below a set tolerance threshold. The final solution of the interpolation function in Equation 2.32, is given as:

$$\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^{(0)} + \boldsymbol{\alpha}^{(1)} + ... + \boldsymbol{\alpha}^{(n)} \tag{2.38}$$

This multilevel approach is computationally more efficient than a single-level greedy algorithm. If m points are used for each level, then the computational cost of constructing an RBF interpolation with $10 \times m$ control points is of the order $10 \times m^4$. As opposed to $(10 \times m)^4$ for the regular single level greedy for the same amount of control points [40]. Similarly to the single level greedy algorithm the same explicit boundary correction, as described in the previous section, is applied once the desired tolerance is reached.

2.6. Space Partitioning Data Structures

For various aforementioned methods a nearest neighbour search is required. In the projection algorithms the nearest boundary midpoint is required. This projection algorithm is used for the pseudo sliding method, for the sliding over curved boundaries with the direct sliding method and for obtaining the error in case of greedy algorithms. Furthermore, in the greedy correction also a search for the nearest boundary point is performed. One approach would be to loop over all the considered points, compute the distance and evaluate which points is closest. However, this brute force approach is not efficient in case of a high number of boundary nodes and will take up a large fraction of the overall computational time. Therefore, an alternative approach is applied using a space partitioning data structure.

A multidimensional binary search tree or k-d tree with k being the dimensionality has proven to be useful for performing nearest neighbour searches [41]. Every point in a k-d tree is a k-dimensional point and each non-leaf member of the tree produces a hyperplane that splits the domain in two half-spaces. The two subtrees of that non-leaf member represent the two half-spaces. The location of the hyperplane is determined by finding the median of the considered points. The direction of the hyperplane is dependent on one of the k-dimensions. For instance if the domain is split in y-direction then all nodes smaller than the median value of y are in one substree and the values larger than the median of y are in the other subtree. Starting from the root node the domain is split cycling over the dimensions. Thus, for three dimensions the domains is first split in x, then in y and lastly in z. This is repeated until a desired number of leaf nodes are in the respective subtrees.

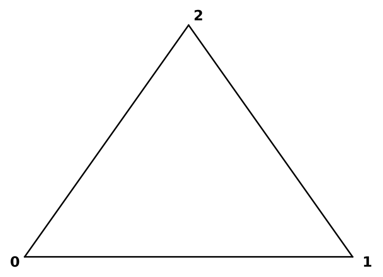
The nearest neighbour search on a query node is performed by moving down the associated subtrees while saving the closest point to the query node. Once a leaf node is reached it is checked whether it is closer than the current closest point. The algorithm then moves back up the tree to check whether there could possibly be any closer points on the other side of the splitting planes. This is a very effective way of finding a nearest neighbour since for a given domain each splitting plane reduces the considered points by a factor of two.

As SU2 and the developed mesh deformation tool in this project are developed with C++, a C++ library is used to generate the k-d trees and perform the nearest neighbour searches. Nanoflann, a fork-header only of FLANN, is selected for this purpose [42]. FLANN is a library offering fast approximate nearest neighbour searches in high dimensional spaces [43]. Typically, approximate nearest neighbour methods are used for higher dimensional spaces and therefore nanoflann was chosen as it can generate k-d trees for two and three-dimensional spaces. Furthermore, due it being a header only library, its implementation is straightforward and simple.

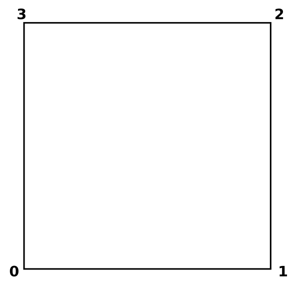
2.7. Mesh Quality Metrics

In order to be able to evaluate the quality of a mesh after applying a deformation, a set of algebraic mesh quality metrics is used as introduced by P.M. Knupp [44]. These metrics are derived from a set of Jacobian matrices, which contain information on size, orientation, shape and skew of an element. Assuming that the initial mesh is generated in an optimal way, its element shapes must be preserved as much as possible. A shape is retained when its volume and its angles are conserved. Therefore, relative size and skew metrics are used in order to evaluate the change in mesh quality. The remainder of this section describes the details of these algebraic metrics for triangular, quadrilateral, tetrahedral, and hexahedral elements.

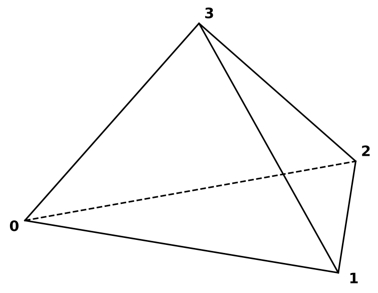
An element with n nodes has the coordinates (x_k, y_k) for planar elements or (x_k, y_k, z_k) for three-dimensional elements, where $k = 0, 1, ..., n-1$. The numbering used for triangular, quadrilateral, tetrahedral and hexahedral elements is shown in Figure 2.19. The Jacobian matrices A_k have size $d \times d$, where d is the dimension of the element. Its columns represent the edge vectors emanating from the node in consideration, ordered by the right-hand-rule to ensure a positive volume elements. The definitions of the n Jacobian matrices A_k for the different types of elements are given in Equation 2.39 through Equation 2.42, where the indices are taken modulo n for the triangular, quadrilateral and tetrahedral elements. For the hexahedral element the indices a, b and c defining the Jacobian matrices are node dependent and given in Table 2.3.



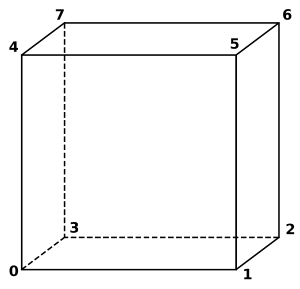
(a) Triangular element.



(b) Quadrilateral element.



(c) Tetrahedral element.



(d) Hexahedral element.

Figure 2.19: Node numbering used for determining the Jacobian Matrices.

Triangular:
$$A_k = \begin{bmatrix} x_{k+1} - x_k & x_{k+2} - x_k \\ y_{k+1} - y_k & y_{k+2} - y_k \end{bmatrix} \tag{2.39}$$

Quadrilateral:
$$A_k = \begin{bmatrix} x_{k+1} - x_k & x_{k+3} - x_k \\ y_{k+1} - y_k & y_{k+3} - y_k \end{bmatrix} \tag{2.40}$$

Tetrahedral:
$$A_k = (-1)^k \begin{bmatrix} x_{k+1} - x_k & x_{k+2} - x_k & x_{k+3} - x_k \\ y_{k+1} - y_k & y_{k+2} - y_k & y_{k+3} - y_k \\ z_{k+1} - z_k & z_{k+2} - z_k & z_{k+3} - z_k \end{bmatrix} \tag{2.41}$$

Hexahedral:
$$A_k = \begin{bmatrix} x_a - x_k & x_b - x_k & x_c - x_k \\ y_a - y_k & y_b - y_k & y_c - y_k \\ z_a - z_k & z_b - z_k & z_c - z_k \end{bmatrix} \tag{2.42}$$

Table 2.3: Node dependent indices for the definition of the hexahedral Jacobian matrices.

Node k	0	1	2	3	4	5	6	7
a	1	2	3	0	7	4	5	6
b	3	0	1	2	5	6	7	4
c	4	5	6	7	0	1	2	3

The determinant of the Jacobian matrix, $\alpha_k = det(A_k)$, gives the volume of the element. Furthermore, a symmetric metric tensors can be defined by $A_k^T A_k$, where its elements $\lambda_{i,j}^k$ for $i, j = 1, ..., d$ contain geometrical properties of the element. Using these properties the relative size and skew metrics are defined.

The relative size metric indicates the change in volume with respect to the initial volume. The ratio τ of the deformed element volume and initial element volume is defined as in Equation 2.43. Where α_k^0 are the values of α_k for the initial element.

$$\tau = \sum_{k=0}^{n-1} \frac{\alpha_k}{\alpha_k^0} \tag{2.43}$$

The relative size metric is then defined as:

$$f_{\text{size}} = \text{min}\left(\tau, \frac{1}{\tau}\right) \tag{2.44}$$

The skew metric describes the angular distortion of an element and its definitions for triangular, quadrilateral, tetrahedral and hexahedral elements are given in Equation 2.45 through Equation 2.48.

Triangular: $$f_{\text{skew}} = \frac{3\alpha}{\lambda_{11} + \lambda_{22} - \lambda_{12}} \tag{2.45}$$

Quadrilateral: $$f_{\text{skew}} = \frac{4}{\sum_{k=0}^{3} \sqrt{\lambda_{11}^k \lambda_{22}^k}/\alpha_k} \tag{2.46}$$

Tetrahedral: $$f_{\text{skew}} = \frac{3(\alpha\sqrt{2})^{2/3}}{\frac{3}{2}(\lambda_{11} + \lambda_{22} + \lambda_{33}) - (\lambda_{12} + \lambda_{23} + \lambda_{13})} \tag{2.47}$$

Hexahedral: $$f_{\text{skew}} = \frac{8}{\sum_{k=0}^{7}(\sqrt{\lambda_{11}^k \lambda_{22}^k \lambda_{33}^k}/\alpha_k)^{2/3}} \tag{2.48}$$

These two metrics can be combined in order to define the size-skew metric by means of a weighted product:

$$f_{\text{ss}} = \sqrt{f_{\text{size}}} f_{\text{skew}} \tag{2.49}$$

For this metric a value of 1 indicates that the deformed element has the same volume and the same angles as the initial element. A value of 0 indicates a degenerate element. Therefore, the value of the minimum mesh quality should be larger then zero. A higher average mesh quality is typically associated with a computation that is more stable, accurate and efficient. Having degenerate cells in the mesh will have a large negative impact on the stability and accuracy of the computation. As a result the minimum value of the mesh quality is the best indication of the quality of a mesh [27].

3 The Numerical RBF-SliDe Tool

This chapter describes the implementation of the various methods, as described in Chapter 2, in the developed RBF sliding deformation (RBF-SliDe) tool. The source code and description on how to use the tool is found at `https://github.com/FvanSteen/Mesh-Deformation-RBF-Interpolation.git`. Since SU2 was developed with C++ it was decided to use the same programming language for the development of the RBF-SliDe tool in this project. Firstly, the main software architecture and workflow of the RBF-SliDe tool is presented. This is followed by a discussion on the workflow of the data reduction techniques and periodic methods. Finally, a detailed description of the steps taken in the regular and sliding boundary RBF interpolations methods is given.

3.1. Main Software Architecture

The classes within the main program of the RBF-SliDe tool are shown in Figure 3.1. The first class encountered during the execution of the programm is the ProbParams class. Within this class a structure containing the main parameters required for the RBF interpolation are initialised. The ReadConfigFile takes this structure as input and assign the parameters based on the input read from the configuration file. The configuration file contains the following information:

- Input filename
- Output filename
- Deformation filename
- Markers that denote the different boundaries
- Markers indicating the moving, periodic boundaries
- Markers indicating boundaries that are treated as internal nodes
- Type of sliding
- Type of periodicity
- Direction of periodicity
- Number of deformation steps
- Data reduction settings

An exemplary configuration file is presented in Appendix B for a simple square mesh.

The ProbParams structure is then passed to the Mesh class, where the input mesh file is read. An example of a typical SU2 mesh file is found in Appendix B. Using the mesh file, the coordinates of all the nodes are stored and the connectivity of the boundaries are saved. Based on the provided markers and the boundary connectivity, the boundary nodes are identified as either moving nodes, edge nodes, surface nodes or as vertices of the domain. The non-identified nodes are assigned as internal nodes. Additionally, for the direct sliding method the local normal and tangential vectors have to be determined at the sliding nodes. To accomplish this the connectivity for the edge and surface nodes is determined. For each sliding edge node, the two nodes connected to it by line segments are found. The surface elements containing the considered surface node are stored for each sliding surface node. The vectors are then determined as described in Section 2.4.4. Furthermore, when reading the mesh file the domain length is computed for the support radius of the RBF and in case of periodicity the periodic length or periodic angle is established.

In case a mesh quality has to be generated, the relative size metric requires that the determinant of the Jacobian matrices of the undeformed mesh needs to be available. Therefore, this has to be done before the RBF interpolation is performed. The MeshQuality class has function related to the computation of the the mesh quality parameters. It requires as input information on the mesh and the parameters stored in the ProbParams structure.

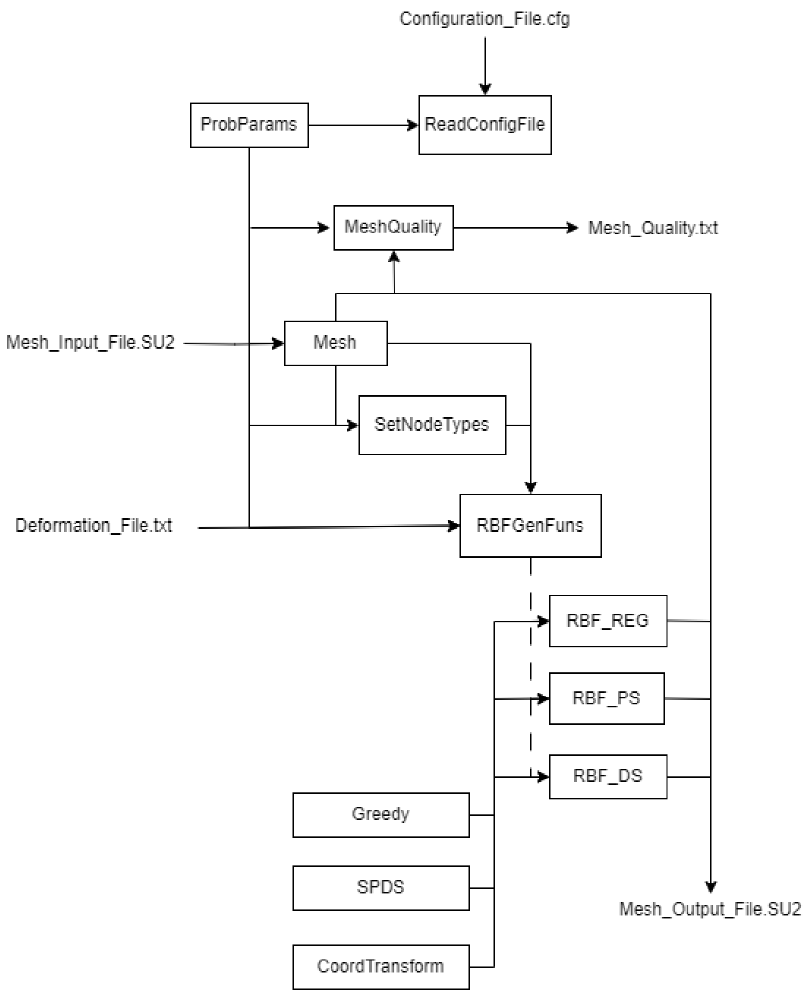


Figure 3.1: Main elements of the RBF-SliDe software architecture.

Based on the type of sliding, periodicity and depending on whether data reduction is applied, the nodes are assigned their type in the SetNodeTypes class. The possible types are moving nodes, sliding edge nodes, sliding surface nodes and internal nodes. The control nodes of the RBF interpolation consist of the moving nodes, sliding edge nodes and sliding surface nodes and are assigned in that order. In case of data reduction methods the list of control nodes is empty and all the boundary nodes are considered to be internal nodes. During the greedy cycles control nodes are added based on the error signal.

After doing this prepatory work the RBF interpolation is started by initialising the RBFGenFuns class. This class contains the general functions that are shared among the different RBF interpolation methods. These functions are inherited by the RBF_REG, RBF_PS or RBF_DS class based on whether a regular, pseudo sliding or direct sliding RBF interpolation is performed. The RBFGenFuns class contains functions for reading the deformation file and setting up the interpolation matrices and deformation vector.

The RBF interpolation itself is done in one of the three dedicated classes RBF_REG, RBF_PS or RBF_DS. If required the Greedy class can be initialised in case of data reduction methods, the SPDS

(Space Partitioning Data Structures) class contains functions required for the projection routines, and in case of rotational periodic domains the coordinate transformation are performed with the functions in the CoordTransform class.

Once the RBF interpolation is performed the output mesh file is generated and if required a file containing the mesh quality can be generated by the MeshQuality class. The general mesh deformation process as described here is summarised in the workflow diagram presented in Figure 3.2.

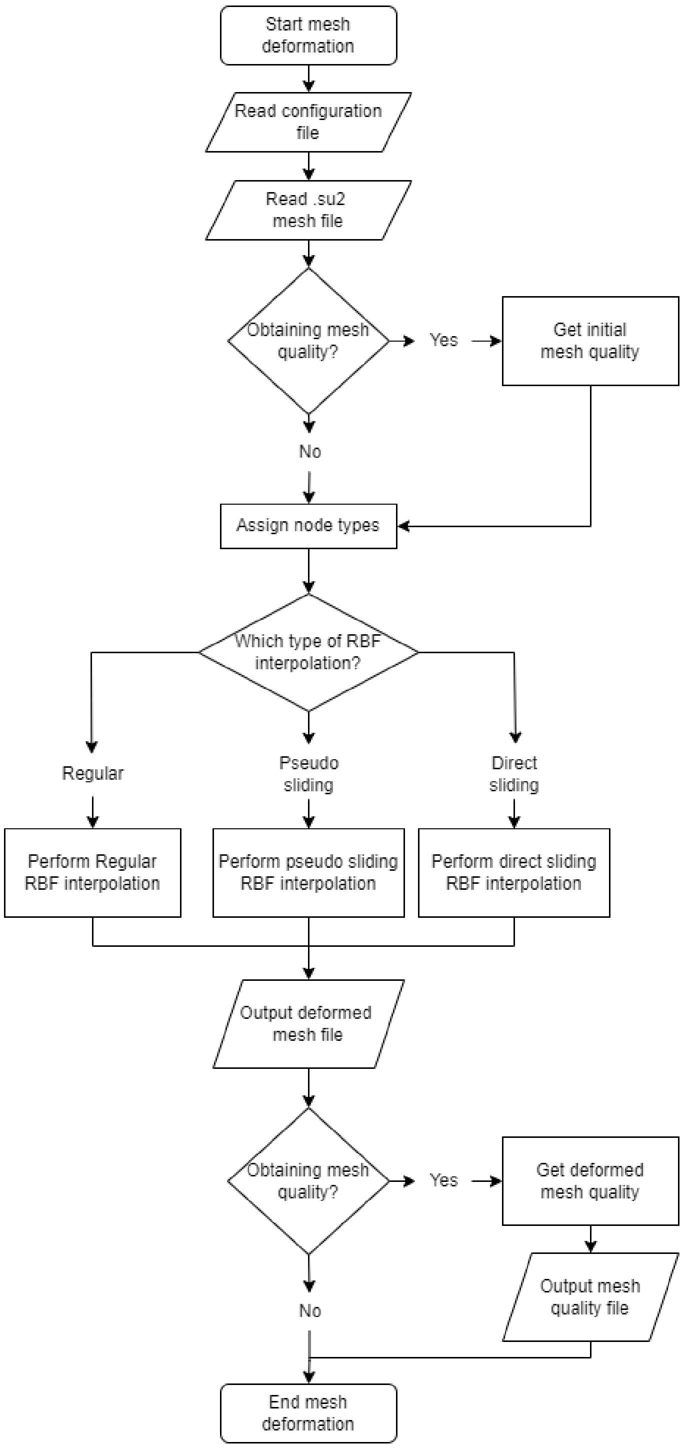


Figure 3.2: Workflow diagram of the developed RBF tool.

3.2. Data Reduction Methods and Periodicity

The flowchart in Figure 3.2 indicates the general process of the RBF mesh deformation tool, where the different forms of (non-)sliding RBF interpolation were represented by a single block. This section focuses on how the data reduction methods and periodicity are treated within these processes. Figure 3.3 shows the general workflow with the inclusion of data reduction algorithms and periodicity and is valid for the regular, pseudo sliding and direct sliding RBF interpolations. Thus, when coming across any of the three RBF interpolation blocks in Figure 3.2, the process shown in Figure 3.3 will be followed within that step.

At the start of the mesh deformation step the file containing the displacement of the moving nodes is being read and the displacement saved. Although not indicated in the diagram, if dealing with a rotational periodic problem the displacement will be transformed to polar or cylindrical coordinates. The formulas corresponding to these transformations can be found in Appendix A. In case of no data reduction the followed process is relatively simple. Firstly, the coordinates are transformed in case of a rotational periodic domain. The RBF interpolation is performed for the number of deformation steps specified in the appropriate coordinate frame. And finally, the coordinates are transformed back to a Cartesian reference system in case of a rotational periodic problem. The various steps taken within the different RBF interpolations are discussed in Section 3.3.

In case of applying a greedy algorithm, a class with functions required to perform the data reduction method is initialised. These function are required to determine the errors, performing the greedy correction and to set the level parameters in case of a multilevel algorithm. If dealing with a rotational periodic domain, the coordinates are transformed to polar or cylindrical coordinates. The difference as compared to the process without data reduction is that this step is performed each deformation step. The greedy correction at the end of each deformation step requires the k-d tree nearest neighbour search to be in Cartesian coordinates to ensure that the nearest boundary point is found in terms of the Euclidean distance. Therefore, at the start of each step a transformation to polar or cylindrical coordinates is required for rotational periodic problems.

After adding the control nodes, which for the first deformation step is based on the maximum displacement of the moving nodes, the RBF interpolation is performed with either a non-sliding or sliding RBF interpolation. After doing so, the errors of the remaining boundary nodes are computed and in case of a double-edged selection a secondary error node is determined. For a single level greedy algorithm this process is repeated until the maximum error falls below the set tolerance. In case of a multilevel algorithm, an additional check on whether the multilevel criterion is reached will be performed. If this is the case then the parameters of that level, consisting of the selected control nodes and their interpolation coefficients as well as the found displacement of the non-selected boundary nodes, are stored and a new level is initialised. Once the tolerance criterion is met, the deformation step is completed by updating the nodes and performing the greedy correction.

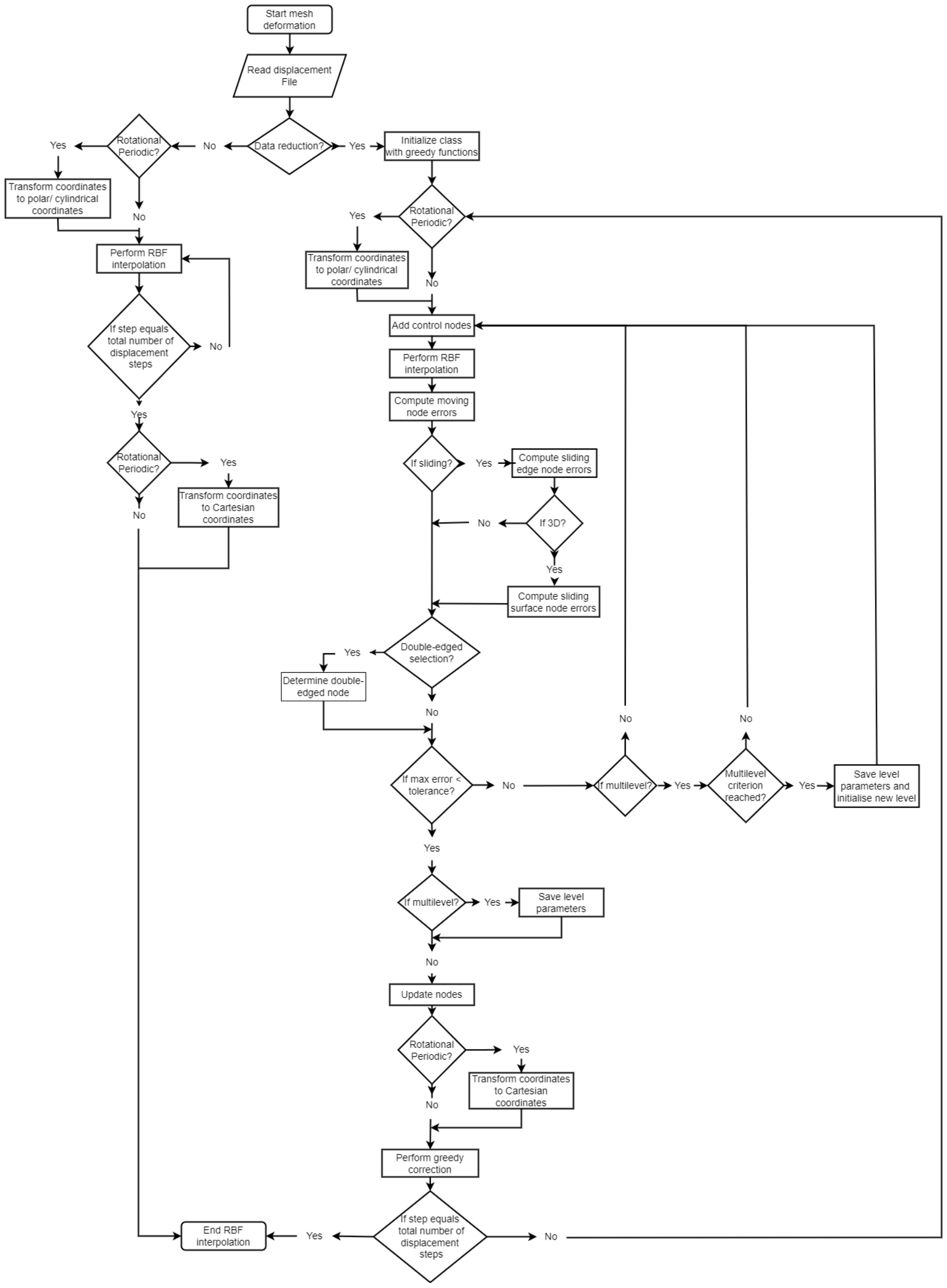

Start mesh deformation
Read displacement File
Data reduction?
Yes
No
Initialize class with greedy functions
Rotational Periodic?
Yes
No
Transform coordinates to polar/ cylindrical coordinates
Perform RBF interpolation
If step equals total number of displacement steps
Transform coordinates to Cartesian coordinates
Add control nodes
Compute moving node errors
If sliding?
Compute sliding edge node errors
If 3D?
Compute sliding surface node errors
Double-edged selection?
Determine double-edged node
If max error < tolerance?
If multilevel?
Multilevel criterion reached?
Save level parameters and initialise new level
Save level parameters
Update nodes
Perform greedy correction
End RBF interpolation

Figure 3.3: Workflow diagram of the developed RBF tool.

3.3. RBF Interpolation Process

The steps taken to perform the different forms of the RBF interpolation are shown in Figure 3.4 for the regular RBF interpolation and in Figure 3.5 for the sliding boundary node RBF interpolations. The regular RBF interpolation comprises of four steps. Firstly, the required interpolation matrices have to be computed, followed by a determination of the deformation vector based on the displacement specified in the deformation file. With this two components the interpolation system can be solved resulting in the interpolation coefficients. With these coefficients the displacement of the internal nodes is computed. For solving the interpolation system, the template C++ library for linear algebra *Eigen* is used [45]. From the range of methods for solving linear systems offered by the *Eigen* library, the *ColPivHouseholderQR* method is selected. This method is a QR decomposition with column pivoting. The *ColPivHouseholderQR* method is chosen since it has no requirements on the invertability or definiteness of the matrix and achieves accurate results. Furthermore, among the available methods with the same requirements and accurateness it is benchmarked as the fastest.

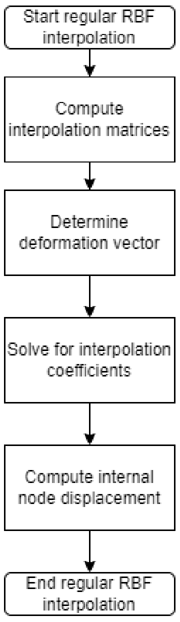
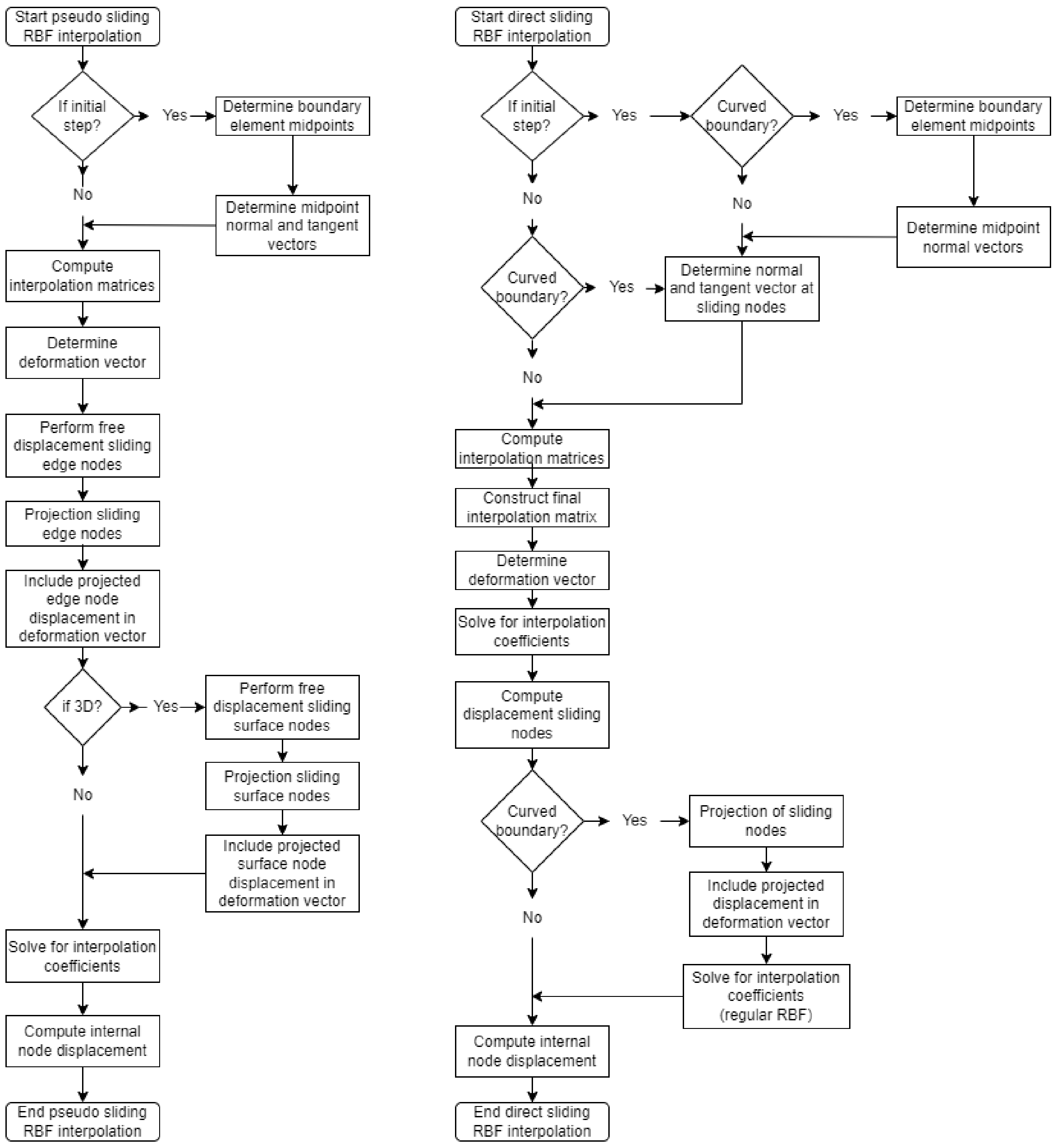

Start regular RBF interpolation
Compute interpolation matrices
Determine deformation vector
Solve for interpolation coefficients
Compute internal node displacement
End regular RBF interpolation

Figure 3.4: Flowchart of the regular RBF interpolation process

The sliding methods are more complex in their RBF interpolations as is reflected by the increased number of steps in Figure 3.5a and Figure 3.5b. In the first step of the pseudo sliding RBF interpolation the midpoints of the boundary elements and their normal vectors, as required for the projection step, are established. This is followed by computing the interpolation matrices and determining the deformation vector. The found displacement of the sliding edge nodes are included in the deformation vector after these nodes are freely displaced and projected onto the boundary. The same is done for the sliding surface nodes in case the problem is three-dimensional. The deformation vector including the projected displacements is used to solve for the interpolation coefficients, as done for the regular RBF interpolation. Finally, the displacement of the internal nodes is computed.

In case of a direct sliding RBF interpolation the normal and tangential vectors at the sliding nodes have to be established. Additionally, in case of a curved boundary these vectors have to be updated each step and the midpoints of the boundary elements and corresponding normal vectors have to be determined for the projection step. Once the interpolation matrices are computed, the final interpolation matrix is assembled and the deformation vector is determined, the system can be solved to find the interpolation coefficients. The computation of the sliding node displacement follows and depending on whether the boundary is curved, a projection of the sliding nodes is performed. If this projection is required then the resulting displacement is used to update the deformation vector and the interpolation coefficients are recomputed according to the regular RBF algorithm. Finally, the displacement of the internal nodes is computed.


Start pseudo sliding RBF interpolation
If initial step?
Yes
Determine boundary element midpoints
Determine midpoint normal and tangent vectors
No
Compute interpolation matrices
Determine deformation vector
Perform free displacement sliding edge nodes
Projection sliding edge nodes
Include projected edge node displacement in deformation vector
if 3D?
Yes
Perform free displacement sliding surface nodes
Projection sliding surface nodes
Include projected surface node displacement in deformation vector
No
Solve for interpolation coefficients
Compute internal node displacement
End pseudo sliding RBF interpolation
Start direct sliding RBF interpolation
If initial step?
Yes
Curved boundary?
Yes
Determine boundary element midpoints
Determine midpoint normal vectors
No
No
Curved boundary?
Yes
Determine normal and tangent vector at sliding nodes
No
Compute interpolation matrices
Construct final interpolation matrix
Determine deformation vector
Solve for interpolation coefficients
Compute displacement sliding nodes
Curved boundary?
Yes
Projection of sliding nodes
Include projected displacement in deformation vector
Solve for interpolation coefficients (regular RBF)
No
Compute internal node displacement
End direct sliding RBF interpolation

(a) Pseudo sliding RBF interpolation

(b) Direct sliding RBF interpolation

Figure 3.5: Flowchart of the sliding boundary node RBF interpolations.

4 Two-Dimensional Case Studies

This chapter will describe the various test cases used to evaluate the performance and efficiency of the implemented RBF interpolation with the methods as introduced in Chapter 2. Relatively simple test cases are used to test and detail the effect of introducing sliding and periodicity in the RBF interpolation. More complex test cases are used to verify to what degree these effects also translate to more realistic domains.

4.1. Two Dimensional Test Cases

Different types of meshes were employed to test the RBF interpolation for two-dimensional domains. A square mesh is used to determine the effects of the sliding algorithms. Furthermore, the square mesh serves as a translational periodic mesh where the periodic boundaries are allowed to move. The different forms of the greedy algorithm are applied to a denser version of the square mesh to evaluate their respective performance. A rotationally periodic mesh is used to verify the application of periodicity in case of rotational periodicity. The final two-dimensional case considered is a more complex mesh of a turbine row.

4.1.1. Square Mesh

The structured square mesh, as shown in Figure 4.1 with its initial mesh quality, is used to evaluate the performance of the sliding algorithms. The properties of this mesh are given in Table 4.1. The domain length and height are equal to a single unit. The spacing of the nodes is equal to 0.04 units in both directions, resulting in 25 elements along the length and height of the domain. An internal block is located in the centre of domain on which a deformation is prescribed.

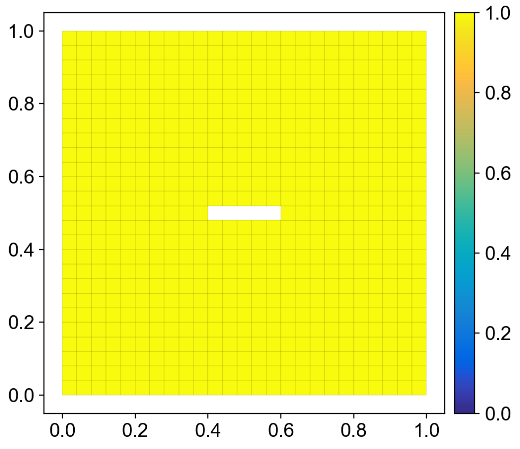


Figure 4.1: Initial mesh quality of the square 25×25 mesh with internal moving block.

Table 4.1: Properties 25×25 square mesh.

	Square mesh
Elements	620
Nodes	676
Domain length	1 unit
Domain height	1 unit
Spacing	0.04 units
Block length	0.2 units
Block height	0.04 units
Min. quality	1.0
Mean quality	1.0

For a regular RBF interpolation all the boundary nodes are moving nodes with either a given displacement for the internal block or zero displacement for the nodes on the outer boundary. In case of applying a sliding boundary node algorithm, the nodes allowed to slide are those on the outer edges of the domain. The vertices of the domain are moving nodes with a zero displacement, thus these are fixed nodes. Additionally, the nodes on the boundary of the internal block are moving nodes since these nodes will have a prescribed deformation. This division of nodes is presented in Figure 4.2.

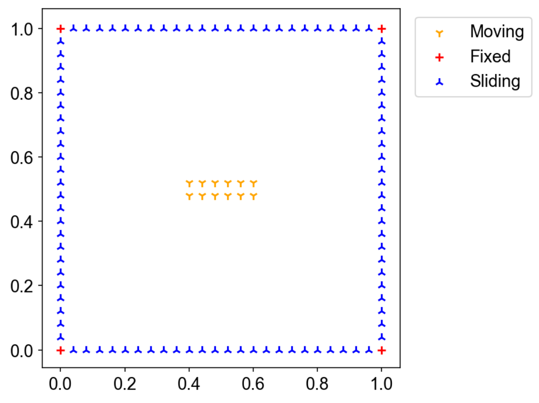


Figure 4.2: Nodal division of the square mesh for the sliding boundary node RBF interpolation.

The first test case applied to the square mesh is a severe translation and rotation applied to the internal block. The block is rotated 60 degrees in anticlockwise direction around its centre and is translated by 0.2 units to the left and 0.3 units in downward direction as shown in Figure 4.3. By applying this deformation a small wall clearance gap is simulated since the block approaches the outer boundary of the domain. A second test case is designed to evaluate the robustness of the sliding algorithms. In this second test case a translation of 0.35 units to the left and 0.43 units in the downward direction in five equal steps is applied. The final displacement will cause the block to be at 0.05 units from both the lower and left boundary of the domain. The five displaced locations *Loc* of the block are shown in Figure 4.4.

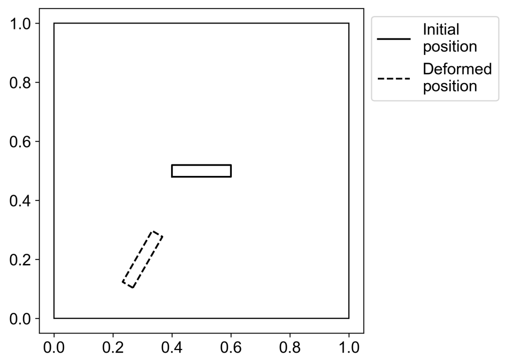


Figure 4.3: 60 degree rotation, 0.2 units translation to the left and 0.3 units translation downward applied to the internal block of the square mesh.

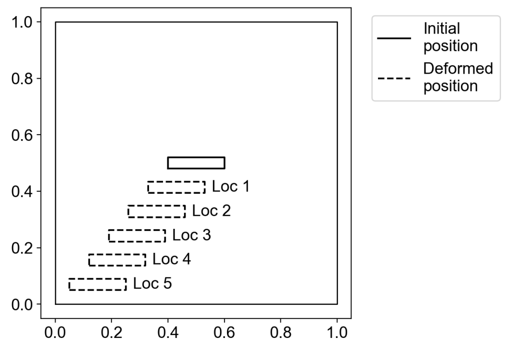


Figure 4.4: 0.35 units translation to the left and 0.43 units translation downward applied in five equal steps to the internal block of the square mesh.

The inclusion of translational periodicity in the RBF interpolation is evaluated by also applying the severe displacement as shown in Figure 4.3. Where the upper and lower boundaries are considered to be periodic, thus the displaced internal block is located in the proximity of a periodic boundary. The periodicity can be introduced in different ways. It can be done by only making the RBF periodic over the periodic distance, keeping the nodal division as depicted in Figure 4.2. Furthermore, the periodic boundaries can be allowed to displace by considering them as internal nodes, where the vertices of the domain are kept fixed as indicated by the nodal division in Figure 4.5. This will be referred to as a periodic displacement with fixed vertices. Lastly, also the vertices of the domain can be allowed to slide exclusively in the periodic direction, referred to as a periodic displacement with moving vertices. In that case the nodal division is as shown in Figure 4.6. The amount of nodes per node type is presented in Table 4.2 for the considered RBF interpolations methods.

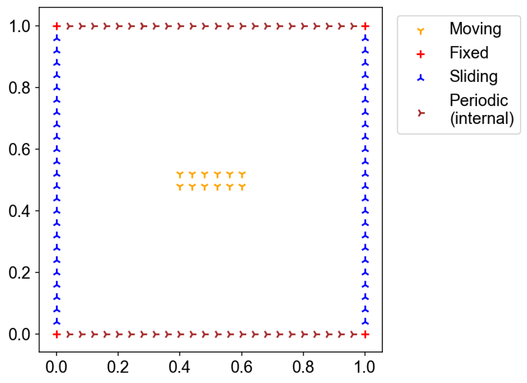


Figure 4.5: Nodal division of the periodic displacement with fixed vertices and sliding boundary nodes for the square mesh.

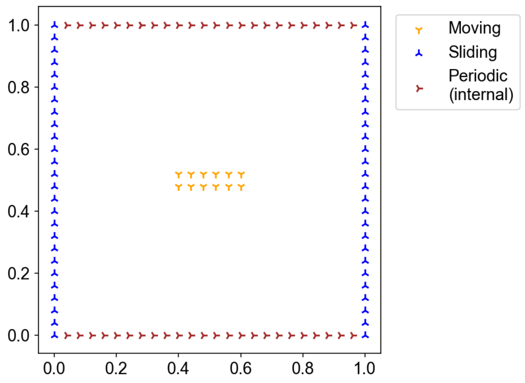


Figure 4.6: Nodal division of the periodic displacement with moving vertices and sliding boundary nodes for the square mesh.

Table 4.2: Number of nodes per node type of the 25×25 square mesh.

	Moving	Fixed	Sliding	Internal
Regular RBF, periodic & non-periodic	12	100	0	564
Regular RBF, periodic displacement with fixed vertices	12	52	0	612
Sliding RBF, periodic & non-periodic	12	4	96	564
Sliding RBF, periodic displacement with fixed vertices	12	4	48	612
Sliding RBF, periodic displacement with moving vertices	12	0	52	612

In order to evaluate the performance of the different forms of greedy algorithm as a data reduction technique, a more dense version of the square mesh is created by decreasing the spacing to 0.005. This increases the number of elements along the length and height of the domain from 25 to 200 elements. The resulting mesh and its properties are shown in Figure 4.7 and Table 4.3. The deformation applied is the same as the severe deformation of the 25×25 test case, as found in Figure 4.3. For this mesh only the non-periodic regular and sliding boundary node RBF interpolations methods are considered and the amount of nodes per node type is presented in Table 4.4.

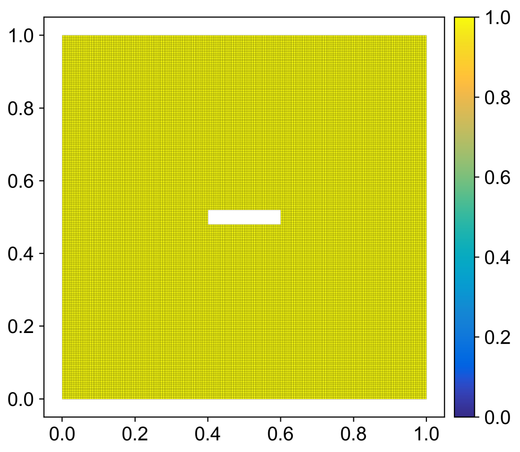


Figure 4.7: Initial mesh quality square 200×200 mesh with a moving internal block.

Table 4.3: Properties 200×200 square mesh.

	Denser square mesh
Elements	39680
Nodes	40128
Domain length	1 unit
Domain height	1 unit
Spacing	0.04 units
Block length	0.2 units
Block height	0.04 units
Min. quality	1.0
Mean quality	1.0

Table 4.4: Number of nodes per node type for the 200×200 square mesh.

	Moving	Fixed	Sliding	Internal
Regular RBF	96	800	0	39232
Sliding RBF	96	4	796	39232

4.1.2. Rotational Periodic Mesh

The rotationally periodic mesh, shown in Figure 4.8, will be used to evaluate the performance of the RBF interpolation with the application of rotational periodicity. The properties of this mesh are given in Table 4.5. The mesh is best described using polar coordinates with a radius between 0.2 and 1.0 and a polar angle between 0 and 30 degrees. The mesh is divided in 25 elements along both the r and θ axes. An internal block is located at the centre of the polar domain. The difference in inner and outer radius of the block is 0.16 and the change in polar angle over the block is 1.2 degrees.

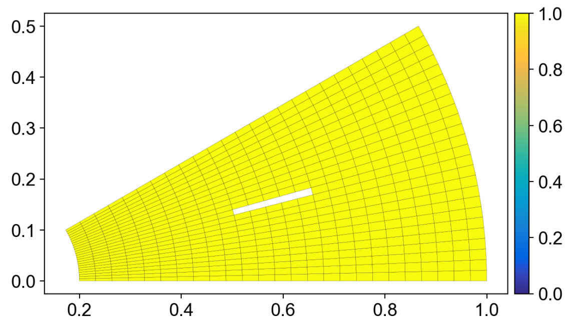


Figure 4.8: Initial mesh quality rotational periodic 25×25 mesh an internal moving block.

Table 4.5: Properties rotational periodic 25×25 mesh.

	Rotational periodic mesh
Elements	620
Nodes	676
Inner radius	0.2 units
Outer radius	1 unit
Radial spacing	0.032 units
Angular spacing	1.2 degrees
Δr block	0.16 units
$\Delta\theta$ block	1.2 degrees
Min. quality	0.99995
Mean quality	0.99995

The displacement applied to the internal block of the mesh in Figure 4.8 is shown in Figure 4.9, where the block is displaced with 0.2 units and -10 degrees in the r and θ-directions. Similar to the square mesh the periodicity can be applied in different manners. The RBF can be made periodic, the periodic boundaries can be allowed to move and the vertices can be allowed to slide in the periodic direction. Where the nodal division for these different forms of periodicity are the same as indicated in Figure 4.2, Figure 4.5 and Figure 4.6 for the square mesh. The periodic direction of the rotational mesh is along the polar angle θ. Therefore, the sliding of the moving vertices will be along a constant radius in case of a periodic displacement with moving vertices. The number of nodes for each node type is presented in Table 4.6 for the considered RBF interpolation methods.

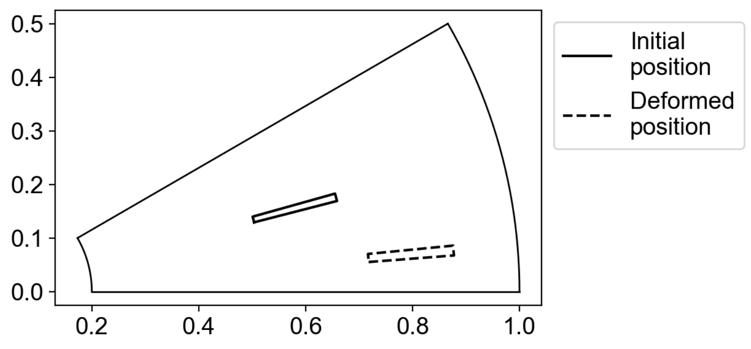


Figure 4.9: Outward radial displacement of 0.2 units and a change of -10 degrees of the polar angle applied to the rotational periodic 25×25 mesh.

Table 4.6: Number of nodes per node type of the rotational periodic 25×25 mesh.

	Moving	Fixed	Sliding	Internal
Regular RBF, periodic & non-periodic	12	100	0	564
Regular RBF, periodic displacement with fixed vertices	12	52	0	612
Sliding RBF, periodic & non-periodic	12	4	96	564
Sliding RBF, periodic displacement with fixed vertices	12	4	48	612
Sliding RBF, periodic displacement with moving vertices	12	0	52	612

4.1.3. Turbine Row

To evaluate the implemented RBF interpolation for more realistic applications, a two dimensional linear supersonic turbine row is used as a test case. Its geometry is presented in Figure 4.10 and the mesh properties are given in Table 4.7. The mesh consists of triangular elements with a structured boundary layer made out of quadrilateral elements.

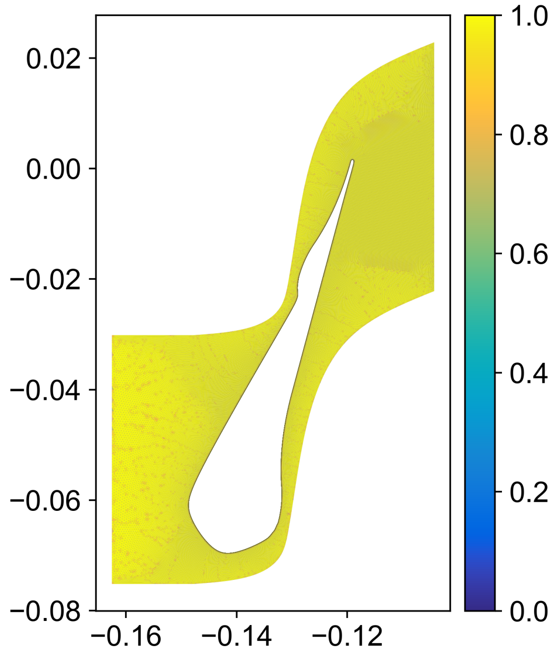


Figure 4.10: Turbine row mesh.

Table 4.7: Properties turbine row mesh.

	Rotational periodic mesh
Elements	224180
Nodes	129883
Domain length	0.058275 units
Domain height	0.097982 units
Min. quality	0.37323
Mean quality	0.99139

The deformation applied to the turbine row is shown in Figure 4.11, where the mesh is rotated 90 degrees to allow for better visualisation. The blade is enlarged with 7% with respect to the indicated point. Additionally, the blade is displaced with 0.001 units downward. The enlargement is chosen as type of deformation due to the fact that in aerodynamic design optimisation these kind of deformations are likely to occur in the design iteration steps. An enlargement percentage of 7% is a rather severe deformation of the blade, whereas for a design optimisation typically smaller deformations are to be applied. This value was selected to test the robustness of the RBF interpolation for the various methods as the blade creates a small clearance near the boundary. In terms of the sliding and periodic nodes, the same guidelines are followed as for the square and rotational periodic mesh. The number of nodes for per node type are stated in Table 4.8 for the various possible RBF interpolations.

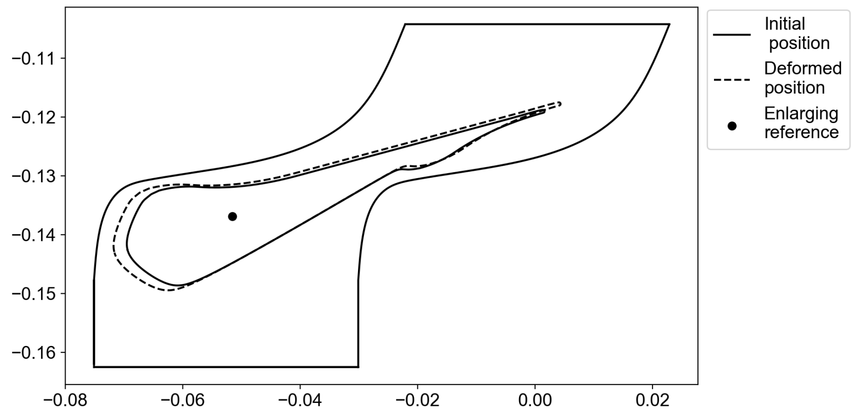


Figure 4.11: 7% enlargement and 0.001 displacement downward applied to the turbine row mesh.

Table 4.8: Number of nodes per node type for the turbine row test case.

	Moving	Fixed	Sliding	Internal
Regular RBF, periodic & non-periodic	2707	4	0	127172
Regular RBF, periodic displacement with fixed vertices	1781	4	0	128098
Sliding RBF, periodic & non-periodic	1315	4	1392	127172
Sliding RBF, periodic displacement with fixed vertices	1315	4	466	128098
Sliding RBF, periodic displacement with moving vertices	1315	0	470	128098

4.2. Three Dimensional Test Cases

The three-dimensional meshes used for the performance evaluation of the different RBF interpolation methods are a hexahedral mesh, a rotational periodic mesh and the complex mesh of a transonic Aachen axial turbine stator.

4.2.1. Hexahedron Mesh

The hexahedron mesh follows from an extension of the two dimensional square mesh to the third dimension. The length of the domain is equal to 1 unit and the spacing is 0.04 units in all directions. Similarly, to the square mesh an internal block is in the centre of the domain. The mesh consists of hexahedral elements. A three-dimensional view of the edges of the domain and the internal block is given in Figure 4.12. The properties of the hexahedron are presented in Table 4.9.

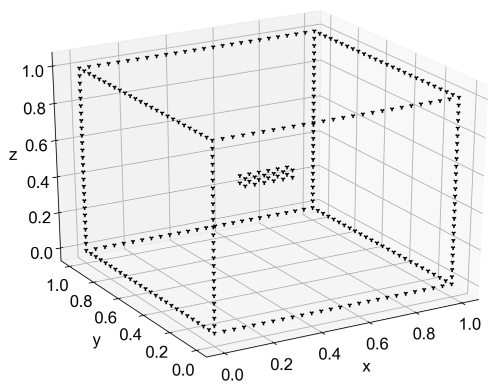


Figure 4.12: Hexahedron $25 \times 25 \times 25$ mesh containing an internal moving block.

Table 4.9: Properties $25 \times 25 \times 25$ hexahedron mesh.

	Hexahedron mesh
Elements	15620
Nodes	17576
Domain length	1 unit
Domain height	1 unit
Domain depth	1 unit
Spacing	0.04 units
Block length	0.2 units
Block height	0.04 units
Block depth	0.04 units
Min. quality	1.0
Mean quality	1.0

For a regular RBF interpolation all boundary nodes are considered to be moving nodes. The internal block has a prescribed deformation and the external boundary nodes have a zero displacement. In the application of the sliding boundary node algorithms the division among the boundary nodes is as indicated in Figure 4.13.

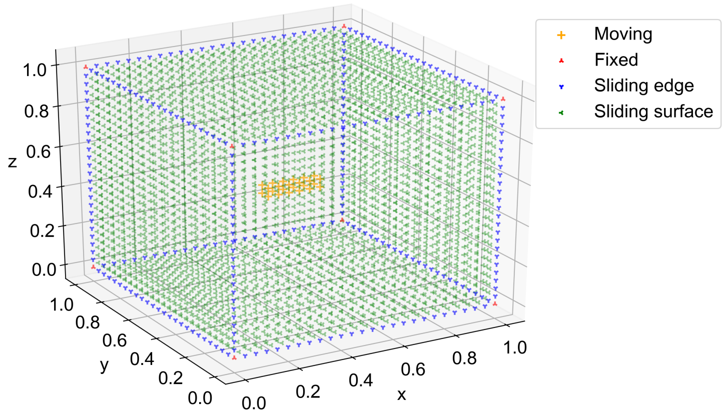


Figure 4.13: Nodal division of the hexahedron mesh in case of sliding boundary nodes.

Two test cases are considered for the hexahedron mesh. The same deformation is applied as for the two-dimensional square mesh and a deformation is considered where also a displacement in the z-direction is included for the internal block. For the two-dimensional comparison a 60 degree anticlockwise rotation around the z-axis done, in addition to a displacement of -0.2 and -0.3 units in x and y-directions. The final location of the internal block is shown in Figure 4.14. For the second test case a displacement of 0.2 units in z-direction is added to the overall displacement. The final location of the block for this displacement is given in Figure 4.15.

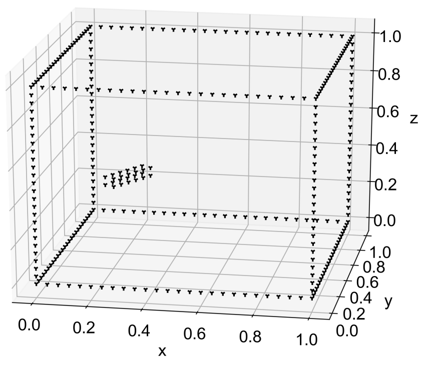


Figure 4.14: 60 degree anticlockwise rotation around the z-axis, displacements of -0.2 and -0.3 units in x and y-directions applied to the hexahedron mesh.

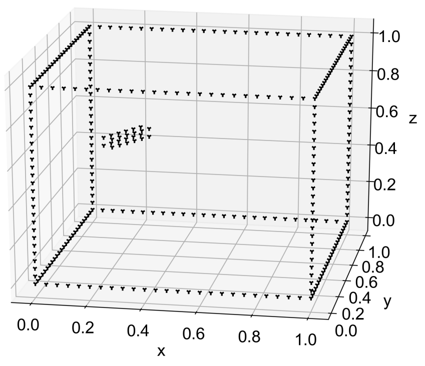


Figure 4.15: 60 degree anticlockwise rotation around the z-axis, displacements of -0.2, -0.3 and 0.2 units in x, y and z-directions applied to the hexahedron mesh.

The severe displacement of the second test case as shown in Figure 4.15, is also used to evaluate the performance of the different periodic methods of the RBF interpolations. For the hexahedron mesh the domain will be periodic in the y-direction. Several possibilities of including the periodicity are evaluated. The domain can be made periodic by simply making the RBF periodic, where the same nodal division as in Figure 4.13 is used. A periodic displacement with fixed vertices can be applied where the periodic boundaries are allowed to move. In that case the periodic surface nodes are considered as internal nodes and the periodic edge nodes are treated as sliding surface nodes that are allowed to slide in the direction of the periodicity. The division of the boundary nodes for periodic displacement with fixed vertices is given in Figure 4.16, where the periodic surface nodes treated as internal nodes are left out for clarity. Additionally, the vertices of the domain can be considered as sliding edge nodes that are allowed to slide in the periodic direction, the nodal division for the periodic displacement with moving vertices is shown in Figure 4.17. The number of nodes per node type for the considered RBF interpolations are stated in Table 4.10.

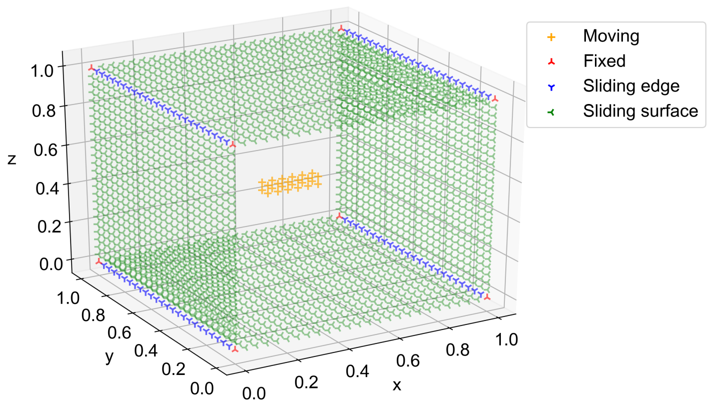


Figure 4.16: Nodal division of the periodic displacement with fixed vertices for the hexahedron mesh.

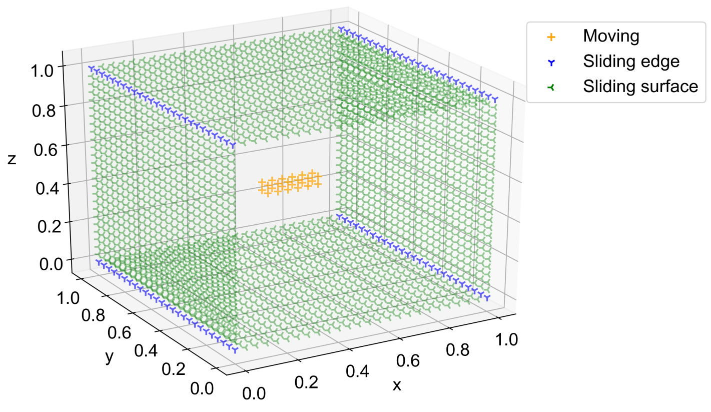


Figure 4.17: Nodal division of the periodic displacement with moving vertices for the hexahedron mesh.

Table 4.10: Number of nodes per node type for the hexahedral mesh.

	Moving	Fixed	Sliding edge	Sliding surface	Internal
Regular RBF, periodic & non-periodic	3768	8	0	0	13800
Regular RBF, periodic displacement with fixed vertices	2616	8	0	0	14952
Sliding RBF, periodic & non-periodic	24	8	288	3456	13800
Sliding RBF, periodic displacement with fixed vertices	24	8	96	2496	14952
Sliding RBF, periodic displacement with moving vertices	24	0	104	2496	14952

4.2.2. Rotational Periodic Mesh

As was the case for the hexahedron mesh, the three-dimensional rotational periodic mesh is an extension of the two-dimensional variant. The three-dimensional rotational periodic mesh is shown in Figure 4.18 and its properties are given in Table 4.11.

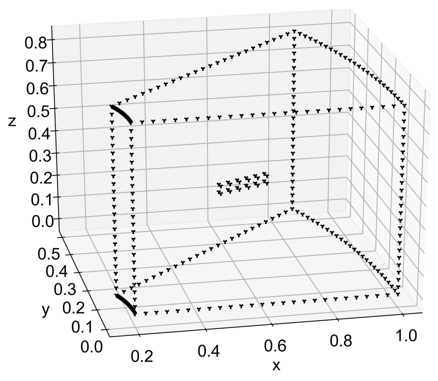


Figure 4.18: Rotational periodic $25 \times 25 \times 25$ mesh containing a block in the centre.

Table 4.11: Properties rotational $25{\times}25{\times}25$ mesh.

	Rotational periodic mesh
Elements	15620
Nodes	17576
Inner radius	0.2 units
Outer radius	1 unit
Domain depth	0.8 units
Radial spacing	0.032 units
Angular spacing	1.2 degrees
z spacing	0.032 units
Δr block	0.16 units
$\Delta\theta$ block	1.2 degrees
Depth block	0.032 units
Min. quality	0.99995
Mean quality	0.99995

The deformation applied to the internal block is similar to the two-dimensional one for the rotational periodic mesh. A displacement of 0.2 units and -10 degrees is applied in the r and θ-directions. Moreover, the block is displaced an additional 0.2 units in the z-direction. The displaced location of the block is shown in Figure 4.19. Once again, the various ways of treating the periodicity of the domain can be applied, where the same boundary node division as for the hexahedron mesh is followed. Furthermore, the number of nodes per node type are given in Table 4.12 for the RBF interpolation methods.

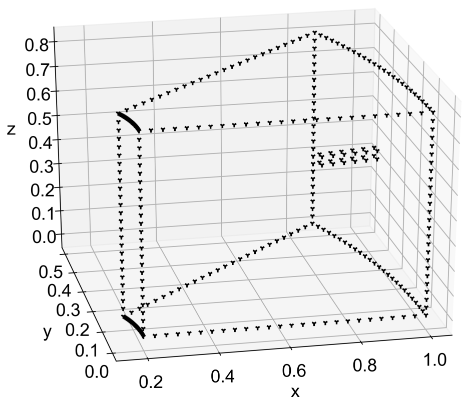


Figure 4.19: Deformation of 0.2 units, -10 degrees and 0.2 units in r,θ and z-directions applied to the three-dimensional rotational periodic mesh.

Table 4.12: Number of nodes per node type for the rotational periodic test case.

	Moving	Fixed	Sliding edge	Sliding surface	Internal
Regular RBF, periodic & non-periodic	3768	8	0	0	13800
Regular RBF, periodic displacement with fixed vertices	2616	8	0	0	14952
Sliding RBF, periodic & non-periodic	24	8	288	3456	13800
Sliding RBF, periodic displacement with fixed vertices	24	8	96	2496	14952
Sliding RBF, periodic displacement with moving vertices	24	0	104	2496	14952

4.2.3. Aachen Turbine Stator

To prove the capability of the implemented methods, a mesh of the Aachen axial turbine stator is used as a more realistic and complex test case. For this test case an aerodynamic design optimisation as performed in SU2 is available, in which the mesh deformation was achieved by using a more conventional method using the linear elasticity equations, Therefore, the resulting deformation from baseline mesh to optimised mesh of the stator blade can be extracted and the found displacement can be applied for an RBF interpolation. This allows for a comparison of mesh quality between the meshes obtained by the RBF-SliDe tool and the default linear elasticity mesh deformation method of SU2. The boundaries of the baseline stator are shown in Figure 4.20, where the periodic nodes are not displayed for clarity of the figure. The properties of the rotationally periodic Aachen turbine stator baseline mesh are given in Table 4.13.

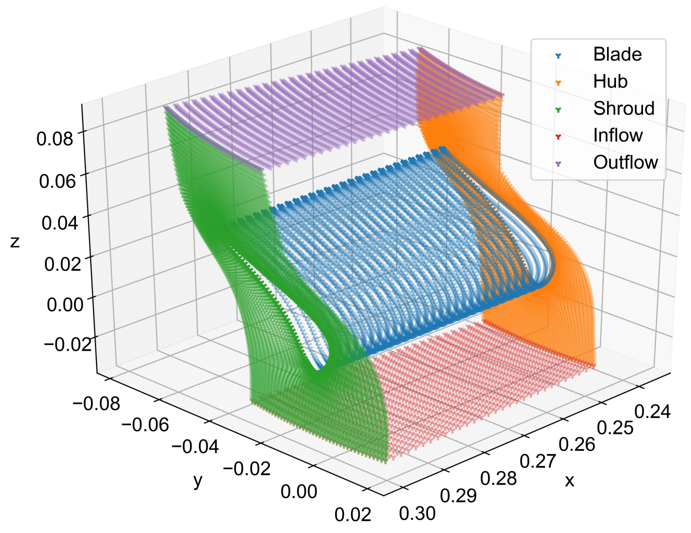


Figure 4.20: The baseline Aachen axial turbine stator mesh.

Table 4.13: Properties Aachen turbine stator mesh.

	Aachen turbine stator
Elements	240004
Nodes	256590
Inner radius	0.25 units
Outer radius	0.3 units
Domain depth	0.1143 units
Nr. of radial elements	29
Nr. of angular elements	41
Nr. of z elements	116
Min. quality	0.80565
Mean quality	0.98520

The optimised geometry of the stator blade as found by the aerodynamic optimisation is presented in Figure 4.21 through Figure 4.23. The deformation of the stator blade was not limited to boundaries of the initial domain and consequently part of the optimised stator blade is not within the boundaries of the domain of the baseline stator mesh. Therefore, a regular RBF interpolation mesh deformation is not applicable since the external boundary nodes are fixed for a regular RBF interpolation, which would lead to an intersection of the optimised stator blade and an external boundary. The regular RBF interpolation method would be possible if one also prescribes a displacement to the external boundary. However, typically the displacement of the outer boundary is a result of the mesh deformation itself and can therefore not be prescribed beforehand.

The sliding boundary node RBF interpolation methods do provide a way to displace the external boundary and as a result are applicable for this test case. Although it should be considered that within the performed aerodynamic optimisation the movement of the nodes on the hub and shroud were not restricted in the radial direction. As a result, a component of the displacement of the nodes on the hub and shroud boundaries are in a direction normal to their respective boundary surface. This is problematic for the direct sliding method since it does not allow for a displacement normal to the initial boundary and for the pseudo sliding method since it projects the sliding nodes back onto the initial boundary. In order to circumvent this problem the surface nodes corresponding to the hub and shroud are considered to be internal nodes, allowing them to displace in a radial sense. The edge nodes of the hub and shroud are included in the RBF interpolation to limit the outward and inward movement of nodes on the hub and shroud surfaces. The nodes on the edge that is shared with the inflow and outflow boundaries are considered to be sliding edge nodes and the nodes shared with the periodic boundary are considered to be sliding surface nodes.

Based on the deformation of the stator blade it is clear that the vertices of the domain should be able to slide in the periodic direction in order to maintain the mesh quality. Otherwise, significant skewing of the mesh elements is introduced by keeping the vertices of the domain fixed. Therefore, only a sliding boundary node RBF interpolation with periodic displacement with moving vertices is being considered for the Aachen turbine stator test case. The different types of nodes used for the RBF interpolation are indicated in Figure 4.24 and the number of nodes for each node type are reported in Table 4.14.

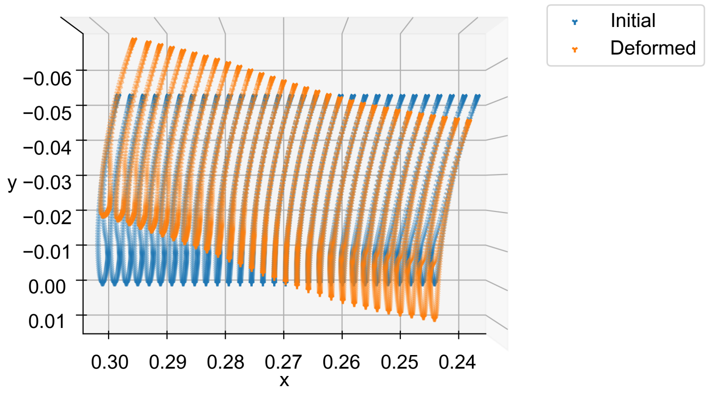


Figure 4.21: Top view of the initial and deformed Aachen stator blade.

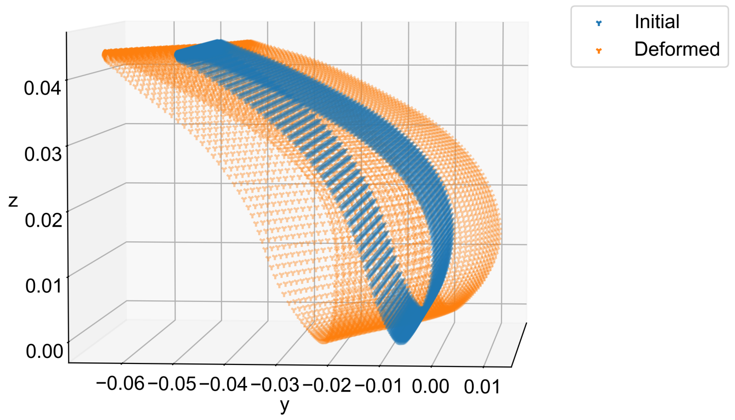


Figure 4.22: Side view of the initial and deformed Aachen stator blade.

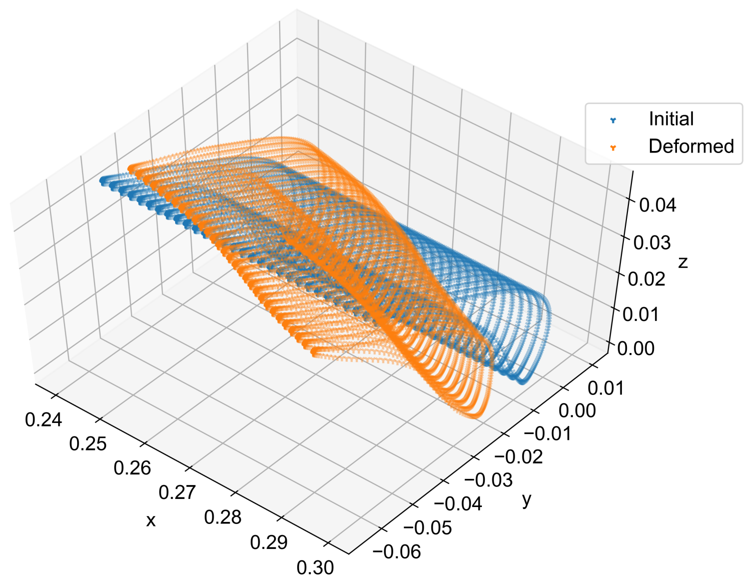


Figure 4.23: Three-dimensional view of the initial and deformed Aachen stator blade.

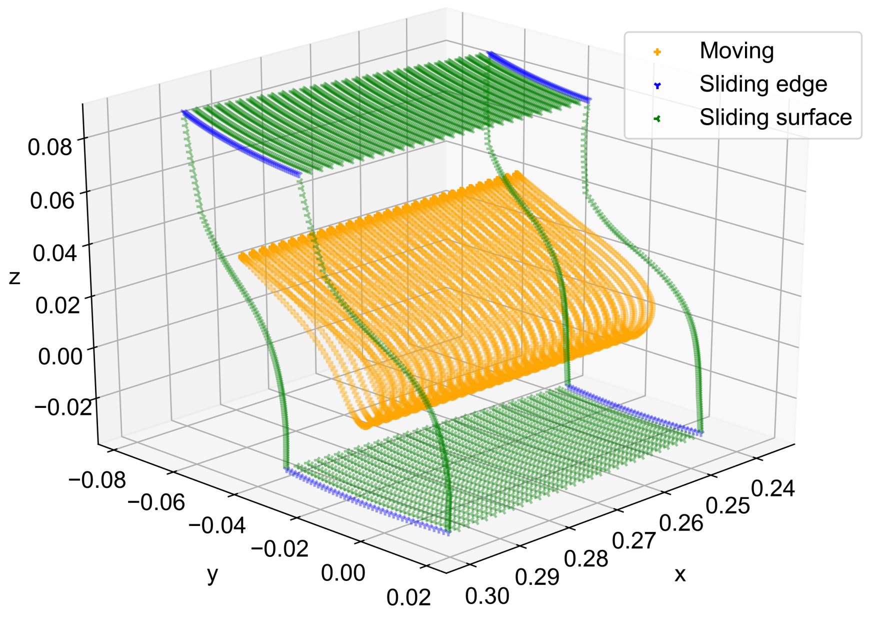


Figure 4.24: Nodal division of the Aachen turbine stator test case.

Table 4.14: Number of nodes per node type for the Aachen turbine stator test case.

	Moving	Fixed	Sliding edge	Sliding surface	Internal
Sliding RBF, periodic displacement with moving vertices	6120	0	236	3764	242960

5 Results Two-Dimensional Cases

This chapter presents the results obtained for the two-dimensional test cases as described in Chapter 4. For all meshes a comparison is made between the regular RBF interpolation and the RBF interpolation with sliding boundary node algorithms. Furthermore, the various ways of introducing periodicity in the RBF interpolation are compared for both translational and rotational periodic domains. Additionally, the performance of the greedy algorithms as control node selection is evaluated for a two-dimensional mesh.

5.1. Square Mesh

For the structured square mesh the effects of employing the sliding boundary nodes algorithms in the RBF interpolation are investigated by applying the severe deformation and increasing translational deformation as specified in Chapter 4. Subsequently, the performance of the different methods for applying periodicity in the RBF interpolation is evaluated. The results are obtained without the use of data reduction methods since the square mesh consists of a relatively small number of nodes. Moreover, it allows for a fair comparison between the RBF interpolation methods without any influence of the greedy algorithm. Additionally, since the outer boundaries are straight edges, no curved boundaries correction is required for the direct sliding method. Although the sliding occurs along the axes of the coordinate system and thus the direct sliding interpolation matrix could be decoupled, it was chosen to leave the interpolation matrix coupled as a reference for cases where sliding occurs along different directions. The RBF used to obtain the results is the compact support Wendland CP C^2 function, with a support radius of 2.5 units. To minimise the effect of possible fluctuations in CPU time, the CPU time is averaged over 5 repeated deformations.

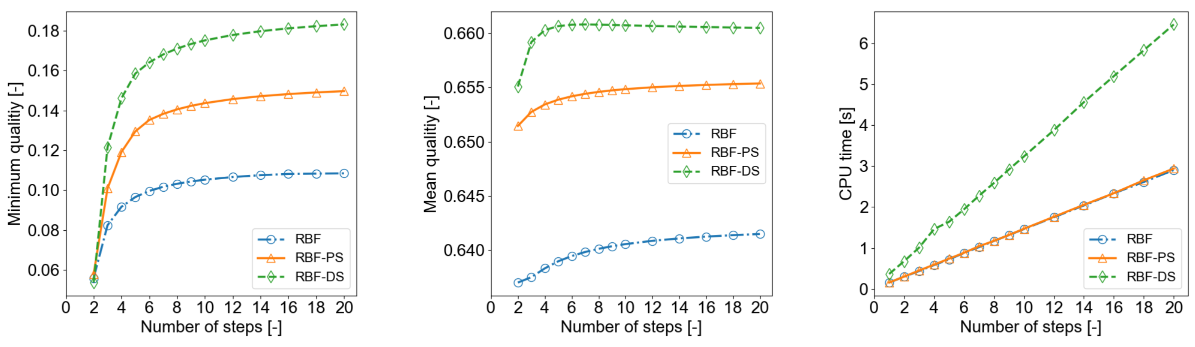
5.1.1. Effect of Sliding Boundary Methods on the RBF Interpolation

The following two sections investigate the effect of using the sliding boundary node methods on the performance of the mesh deformation method for the square mesh. The test case with the severe deformation is used to compare the deformation performance of the sliding boundary RBF interpolation to the regular RBF interpolation. The robustness of the regular and sliding RBF interpolations is investigated using the increasing translation deformation.

Mesh Deformation Performance

The impact of using the sliding boundary node RBF interpolations on the mesh deformation performance is evaluated here for the square mesh. The severe displacement considered for the internal block is a 0.2 and 0.3 units displacement in left and downward direction, in addition to a 60 degree rotation. The results of applying the severe deformation in several steps are shown in Figure 5.1.

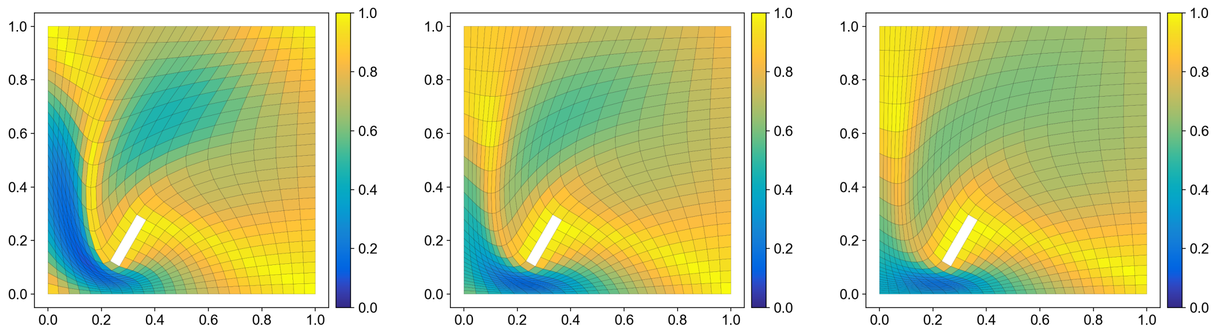
It is clear that the minimum mesh quality improves in case of allowing the boundary nodes to slide. When applying the deformation in 20 steps the regular RBF interpolation manages a minimum mesh quality of 0.108, the pseudo sliding method a minimum quality of 0.147 and the direct sliding method achieves a minimum quality of 0.183. As the minimum mesh quality increases the average quality follows the same trend, as can be seen in Figure 5.1b. In terms of computational cost it can be seen that the direct sliding method has a significantly higher computational time, which is attributed to significantly larger coupled interpolation matrix that has to be solved for the direct sliding method. The increase in CPU time due to the additional RBF interpolation step and projection for the pseudo sliding method is almost negligible compared to the regular RBF interpolation for this small number of control nodes.



(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 5.1: Performance comparison of the regular and sliding RBF interpolation methods for the square mesh.

Figure 5.2 shows the resulting mesh quality after 20 steps for the regular and sliding boundary node RBF interpolations. For the regular RBF interpolation the deformation induces a large skewing of the cells at the left side of the boundary. By allowing the nodes to slide along with the prescribed displacement the mesh quality improves significantly in this area. The nodes on the boundary have slid slightly further for the direct sliding method in comparison to the pseudo sliding method.



(a) Regular RBF **(b)** Pseudo sliding RBF **(c)** Direct sliding RBF

Figure 5.2: Mesh quality of the regular and sliding node RBF interpolation methods using 20 deformation steps for the square mesh.

Robustness

The mesh deformation performance of the regular and sliding RBF interpolations using 20 deformation steps are presented in Figure 5.3 as a function of the fraction of the final deformation. As one would expect the difference in minimum mesh quality between the regular RBF interpolation and the sliding methods diverges for increasing displacement. This is due to the fact that in case of a regular RBF interpolation the cells experience an increasingly higher skewness for an increasingly larger displacement, because the boundary nodes are fixed. The same effect is also reflected in the mean mesh quality parameter. The direct sliding achieves a higher minimum mesh quality for larger deformations compared to the pseudo sliding method. Which is a result of the increased degree of sliding of the direct sliding method. The degree of deformation has little to no effect on the computational cost of the RBF interpolation methods.

The resulting mesh quality at 80% of the total deformation is shown in Figure 5.4. In this figure the high skewness of the cells at the left side of the domain can be seen for the regular RBF interpolation. Furthermore, The higher degree of sliding can be observed for the direct sliding method compared to the pseudo sliding method. It is not only visible in the highly affected region at the lower left side of the domain but also in the top right region. Therefore, the direct sliding method can be considered to be the most robust. However, this comes at the price of an increased computational cost.

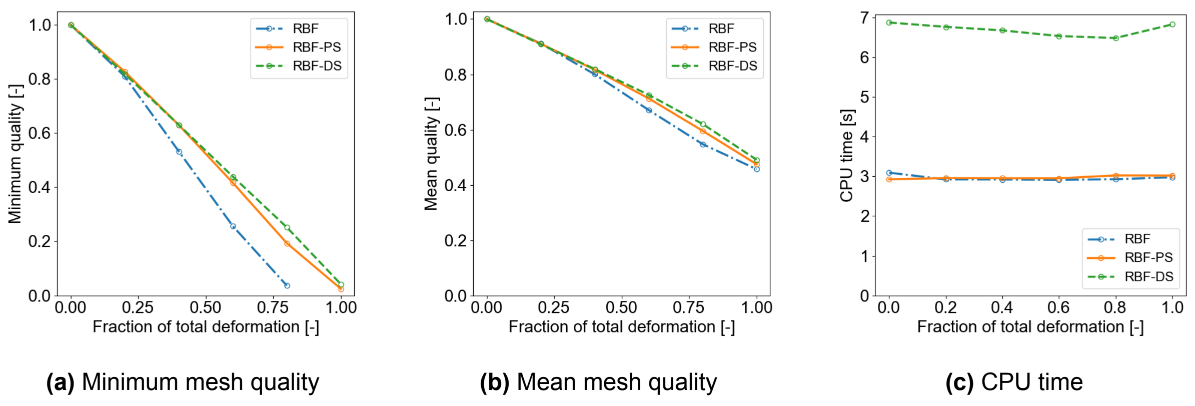


Figure 5.3: Performance comparison of the regular and sliding RBF interpolation methods with 20 deformation steps as a function of the fraction of the total deformation,

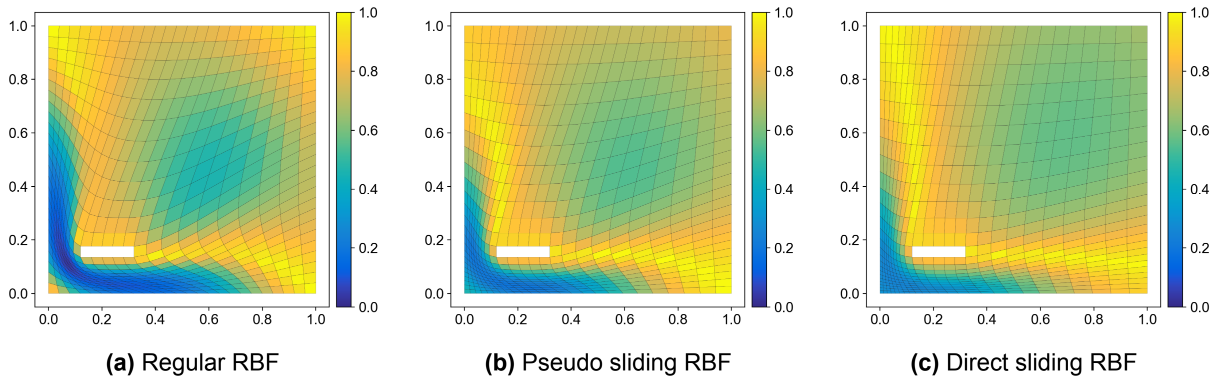


Figure 5.4: Mesh quality of the regular and sliding node RBF interpolation methods at 80% of the total deformation using 20 deformation steps.

5.1.2. Effect of Periodicity on the RBF interpolation

The effects of including translational periodicity in the various ways as described in Chapter 4 are presented here for the regular and sliding boundary node RBF interpolations.

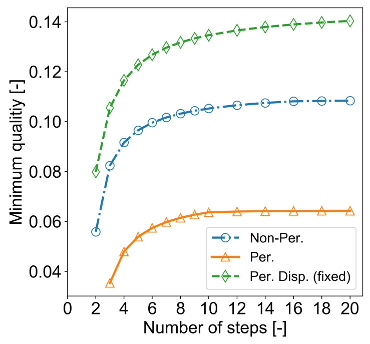
Mesh Deformation Performance Regular RBF interpolation

The CPU time, minimum and mean mesh qualities as a function of the number of deformation steps for the non-periodic, periodic and periodic displacement with fixed vertices are shown in Figure 5.5. The resulting mesh quality after 20 deformation steps are displayed in Figure 5.6. It can be seen that the minimum mesh quality decreases when a periodic RBF is applied. Figure 5.6b shows that the mesh quality at the periodic interface below the block is slightly improved, as the skewness and compression of the cells is relieved in that area compared to the non-periodic deformation. However, this induces a larger skewing and compression of the cells in the area just above the periodic interface, where the minimum mesh quality is found.

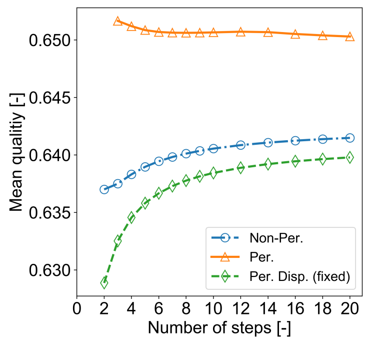
Allowing the periodic boundaries to displace by treating as them internal nodes and keeping the vertices of the domain fixed does improve the minimum mesh quality. In that case the minimum mesh quality equals 0.1403 after 20 deformation steps in comparison to 0.1084 of the non-periodic RBF. The reason for this is that the periodic boundary moves according to the displacement of the internal block, as shown in Figure 5.6c. Which prevents the deterioration of mesh quality in case the block approaches the periodic interface.

In terms of mean mesh quality small differences can be noticed, with the periodic RBF interpolation producing the highest mean mesh quality. However, the minimum mesh quality is the better indicator for the quality of the mesh and a small reduction in mean mesh quality can therefore be taken for granted.

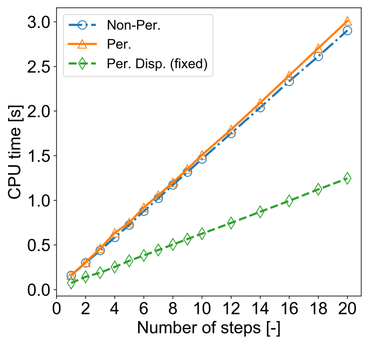
The computational cost of the periodic and non-periodic RBF interpolations are nearly identical, as seen in Figure 5.5c. This is due to the fact that only the distance function of the RBF is altered by introducing the sine function to make it periodic, which has no significant impact on the computational cost. The periodic displacement with fixed vertices is computationally more efficient. The reason for this is that the nodes on the periodic boundaries no longer serve as control nodes and thus the interpolation system that has to be solved reduces significantly in size.



(a) Minimum mesh quality

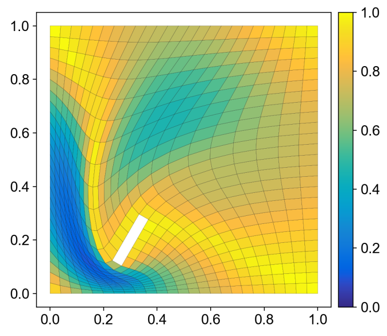


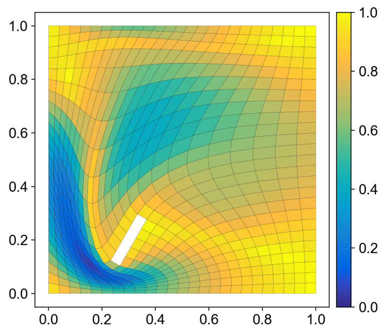
(b) Mean mesh quality



(c) CPU time

Figure 5.5: Performance comparison of the periodic regular RBF interpolation methods for the square mesh.


(a) Non-periodic RBF


(b) Periodic RBF

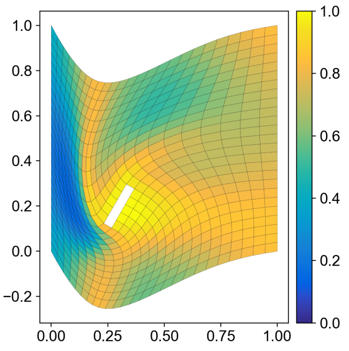

(c) Periodic displacement with fixed vertices.

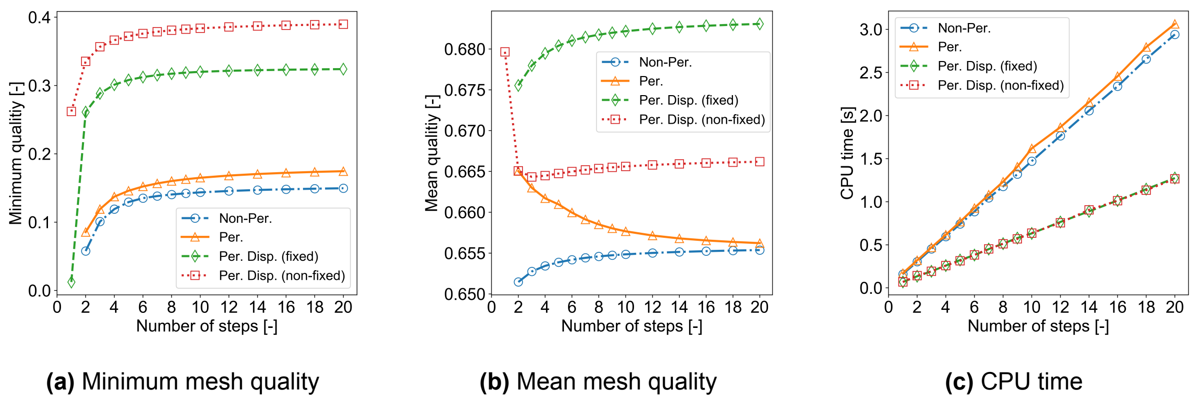
Figure 5.6: Mesh quality of the periodic regular RBF interpolation methods using 20 deformation steps for the square mesh.

Mesh Deformation Performance Sliding RBF interpolation

The results of including periodicity in combination with the pseudo sliding and direct sliding methods are shown in Figure 5.7 and Figure 5.8. The resulting mesh qualities of both sliding methods are presented in Figure 5.9 and Figure 5.10. In Figure 5.7a and Figure 5.8a it can be noticed that the minimum mesh quality increases when comparing a non-periodic to a periodic sliding RBF interpolation. This is likely due to the reduces skewness of the cells as a result of the sliding. The mean mesh quality on the other hand is reduced, this is reflected in Figure 5.9b and Figure 5.10b. Where in addition to the affected region below the block, also a region at upper part of the domain and above block have a reduced mesh quality due to the periodicity.

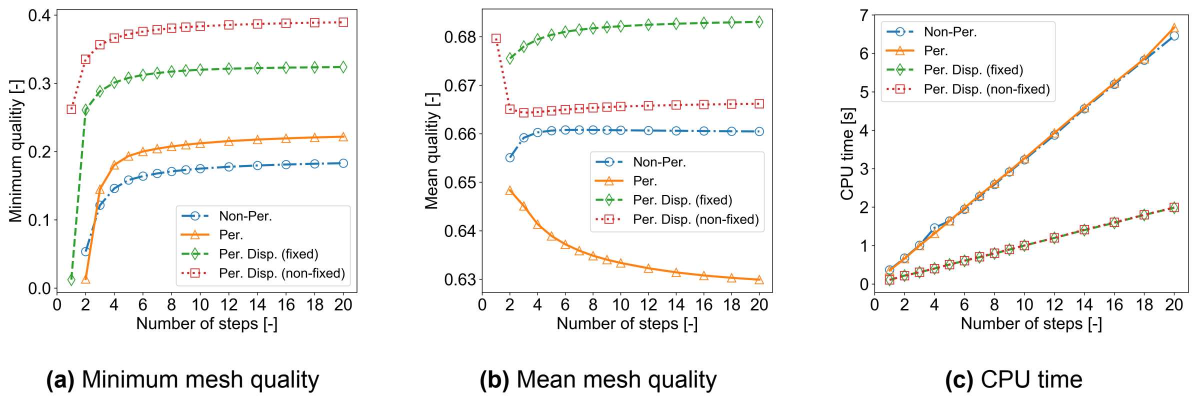
Allowing a periodic displacement of the periodic boundaries produces a significant increase in minimum mesh quality for both sliding methods, as shown in Figure 5.7a and Figure 5.8a. With the periodic displacement with moving vertices achieving the highest minimum quality of 0.3896 for both sliding methods, in that case the sliding along the left and right boundaries are no not limited by a zero displacement at the ending vertices. This results in less skewing of the cells at the sliding boundaries as can be seen when comparing Figure 5.9d with Figure 5.9c for the pseudo sliding and Figure 5.10d with Figure 5.10c for the direct sliding.

In terms of computational efficiency the same trends can be found for both sliding methods, as seen in Figure 5.7c and Figure 5.8c. There is hardly any difference between the non-periodic and periodic methods, since changing the RBF distance function does not significantly increase the computational cost of the interpolation. The periodic displacement with fixed and moving vertices require less computational effort due to the reduction in control nodes. This effect is more pronounced for the direct sliding method since its interpolation matrix has size $3N_c \times 3N_c$ as compared to $N_c \times N_c$ for the pseudo sliding method.



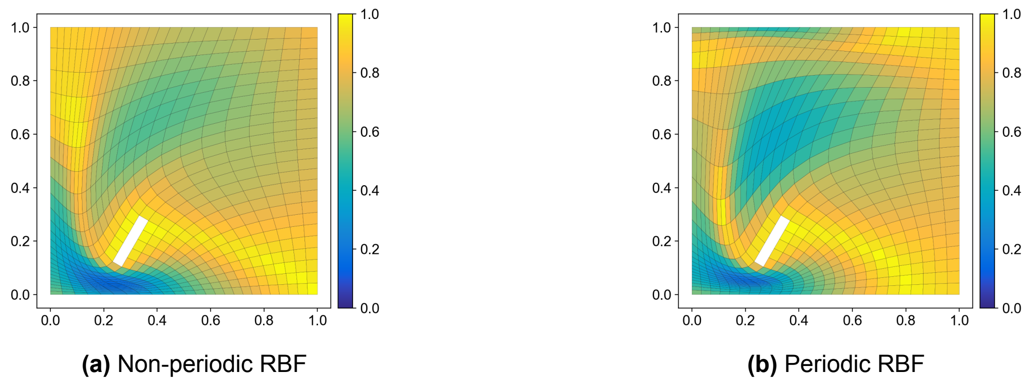
(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 5.7: Performance comparison of the periodic pseudo sliding RBF interpolation methods for the square mesh.

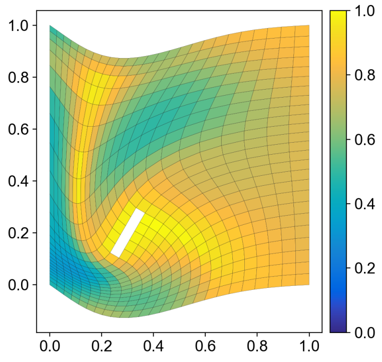


(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

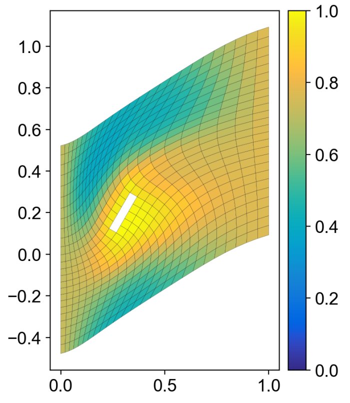
Figure 5.8: Performance comparison of the periodic direct sliding RBF interpolation methods for the square mesh.



(a) Non-periodic RBF **(b)** Periodic RBF

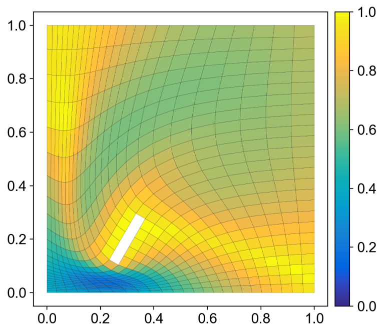


(c) Periodic displacement with fixed vertices

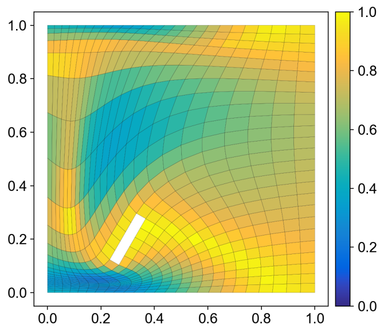


(d) Periodic displacement with non-fixed vertices

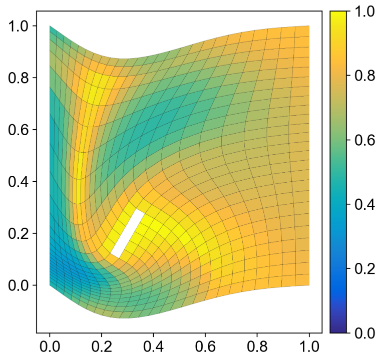
Figure 5.9: Mesh quality of the periodic pseudo sliding RBF interpolation methods using 20 deformation steps for the square mesh.



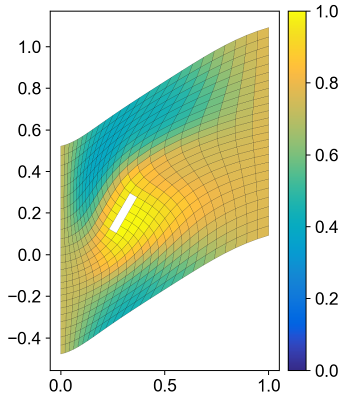
(a) Non-periodic RBF



(b) Periodic RBF



(c) Periodic displacement with fixed vertices

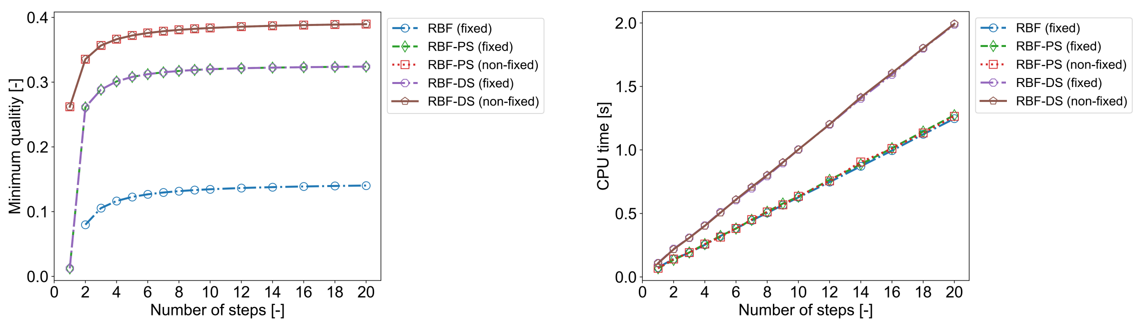


(d) Periodic displacement with non-fixed vertices

Figure 5.10: Mesh quality of the periodic direct sliding RBF interpolation methods using 20 deformation steps for the square mesh.

Comparison of the Best Deformation Methods

Figure 5.11 shows a comparison in terms of minimum mesh quality and computational time of the best performing methods of the RBF interpolations as considered in the previous sections. All results shown in these figures are for periodically displaced boundaries with either fixed or moving vertices. Notably, both sliding methods achieve approximately the same minimum mesh quality for the whole range of deformation steps. Where the pseudo sliding method has the advantage of being computationally more efficient, due to its smaller uncoupled interpolation system. The potential of the sliding methods is highlighted by the significant increase in minimum mesh quality. The regular RBF interpolation with fixed vertices achieves a minimum mesh quality of 0.14 and the sliding methods with periodic displacement with moving vertices are able to obtain a minimum mesh quality of 0.39 after 20 deformation steps.



(a) Minimum mesh quality **(b)** CPU time

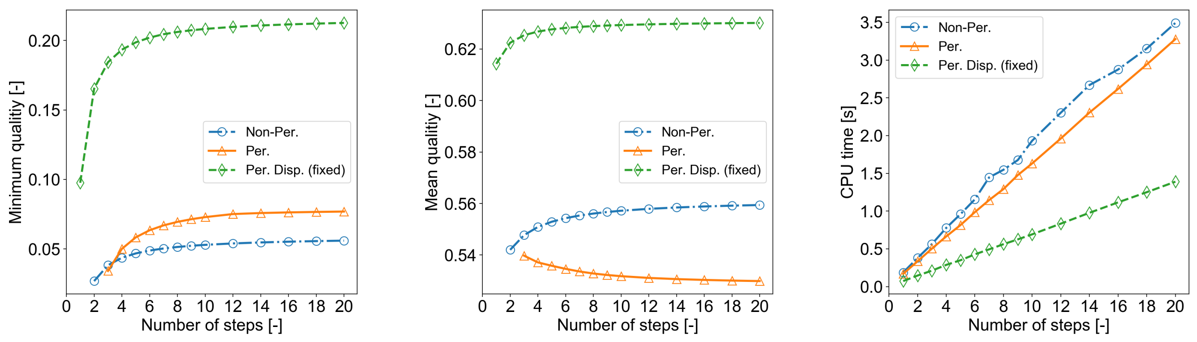
Figure 5.11: Performance comparison of the best achieving regular and sliding RBF interpolation methods for the square mesh.

5.2. Rotational Periodic Mesh

This section shows the effect of applying the periodicity to a rotationally periodic mesh. Similar to the results in the previous section, no data reduction methods are used and the CPU time is averaged over 5 interpolations. Additionally, no projection for curved surfaces is required for the rotationally periodic mesh, since in polar coordinates either the radius or polar angle varies on the outer boundaries of the domain. The internal block is displaced by 0.2 units and -10 degrees in the r and θ-directions, as described in Chapter 4. The RBF used is the compact support Wendland CP C^2 function, with a support radius of 2.07 units. which is 2.5 times the maximum domain length.

5.2.1. Effect of Periodicity on the Regular RBF interpolation Performance

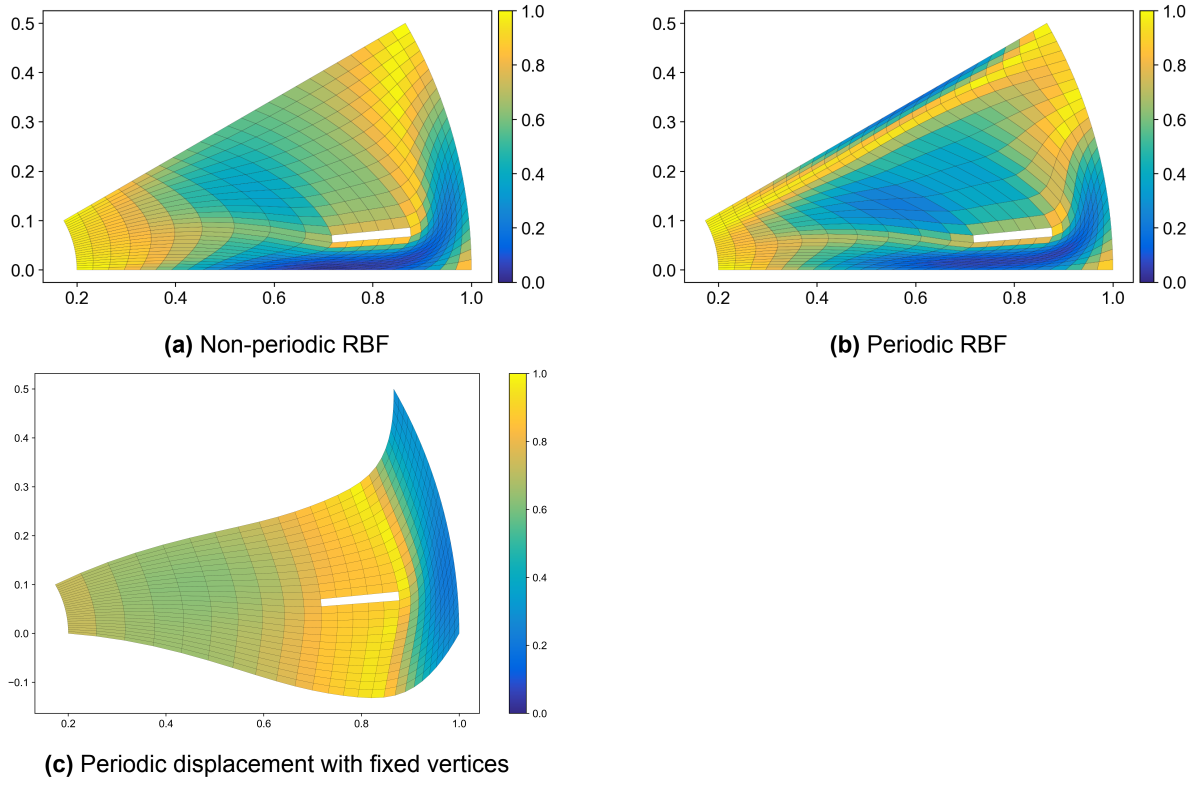
The graphs of Figure 5.12 show the resulting mesh deformation performance of applying the methods of periodicity on the rotational periodic mesh. Figure 5.13 displays the mesh quality after 20 deformation steps. For this test case it can be seen that minimum mesh quality does increase slightly for the regular RBF interpolation when the distance function of the RBF is made periodic. However, the mean mesh quality is lower compared to the non-periodic one. As can also be seen when comparing Figure 5.13b to Figure 5.13a.



(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 5.12: Performance comparison of the periodic regular RBF interpolation methods for the two-dimensional rotational periodic mesh.

The minimum mesh quality significantly increases in case the periodic boundaries are allowed to displace. With a comparison of Figure 5.13a and Figure 5.13c, it can be seen that the mesh quality under the block is improved as the compression of cells in the vicinity of the periodic boundary is prevented due to the displacement of the boundary. The trends in computational effort as shown in Figure 5.12c are similar to those of the translational periodic domain of the previous sections. There is a slight increase in computational effort due to the transformations between coordinate frames, but the general trends are similar. The non-periodic and periodic are comparable in terms of computational cost and the periodic displacement with fixed vertices is more efficient due to the reduction in control nodes.



(a) Non-periodic RBF

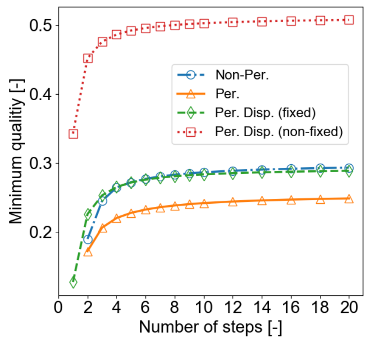
(b) Periodic RBF

(c) Periodic displacement with fixed vertices

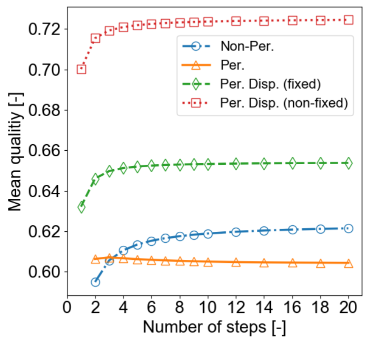
Figure 5.13: Mesh quality of the periodic regular RBF interpolation methods using 20 deformation steps for the two-dimensional rotational periodic mesh.

5.2.2. Effect of Periodicity on the Sliding RBF Interpolation Performance

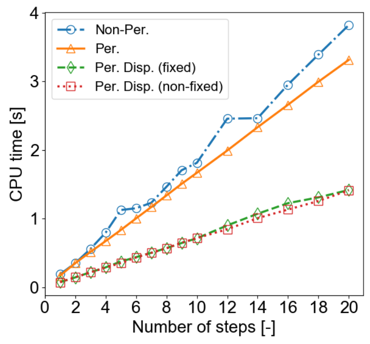
The results obtained for the rotational periodic mesh in case of employing the pseudo and direct sliding methods are shown in Figure 5.14 and Figure 5.15. When applying the sliding boundary node algorithms it is noticed that the minimum mesh quality does not improve by making the distance function periodic or allowing a periodic displacement with fixed vertices. The periodic RBF reduces the mesh quality in the lower right corner of the domain as can be seen in Figure 5.16b and Figure 5.17b for the pseudo and direct sliding methods. The use of the periodic displacement with fixed vertices method induces skewing of the cells along the entire outer radius of the domain, as shown in Figure 5.16c and Figure 5.17c. Using the periodic displacement with moving vertices method alleviates this issue as the inner and outer boundaries are able to slide along with the prescribed deformation of the block. This is reflected in the higher mesh qualities displayed in Figure 5.16d and Figure 5.17d for both sliding methods. The minimum mesh quality is almost doubled with respect to the other periodicity methods. Due to the significant rise in minimum quality also the mean quality is higher compared to the other methods. The same trends can be observed for the CPU times as for the square mesh, with the periodic displacement method being more efficient due to the reduced set of control nodes.



(a) Minimum mesh quality

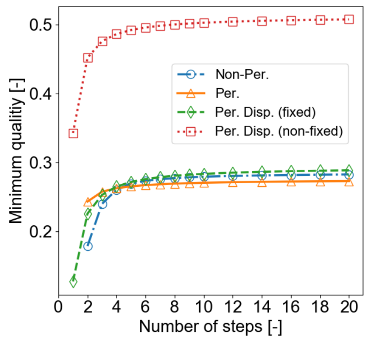


(b) Mean mesh quality

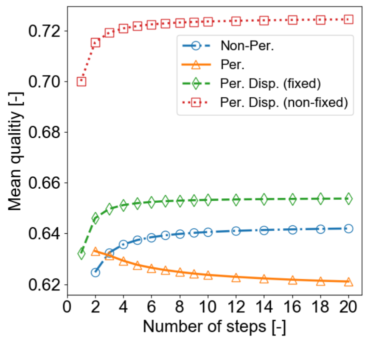


(c) CPU time

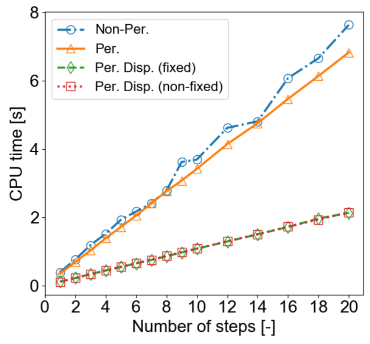
Figure 5.14: Performance comparison of the periodic pseudo sliding RBF interpolation methods for the two-dimensional rotational periodic mesh.



(a) Minimum mesh quality



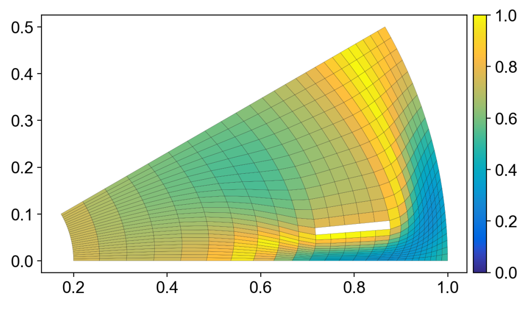
(b) Mean mesh quality



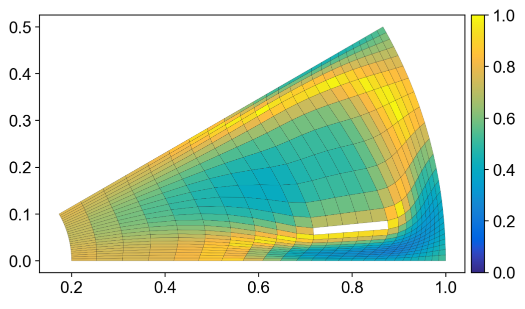
(c) CPU time

Figure 5.15: Performance comparison of the periodic direct sliding RBF interpolation methods for the two-dimensional rotational periodic mesh.

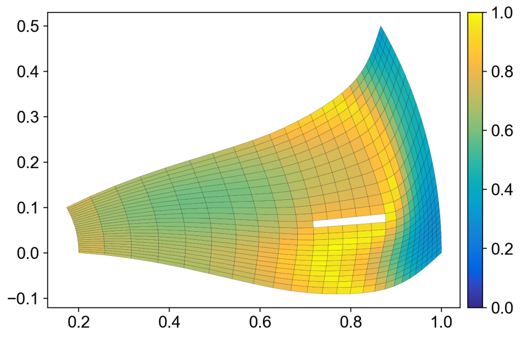
The resulting mesh qualities after 20 deformation steps are presented in Figure 5.16 and Figure 5.17 for the pseudo and direct sliding methods. The direct sliding displays a slightly larger magnitude of sliding compared to the pseudo sliding method for the non-periodic and periodic mesh deformations, since the lower right region of the domain is affected to a lesser extent by the deformation. In case of periodic displacement of the boundary, both sliding methods produce very similar mesh qualities.



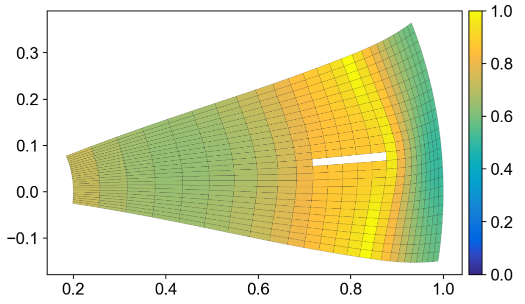
(a) Non-periodic RBF



(b) Periodic RBF

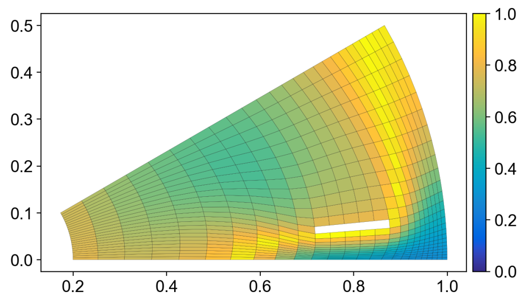


(c) Periodic displacement with fixed vertices

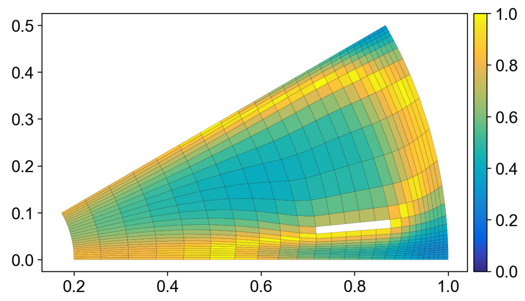


(d) Periodic displacement with non-fixed vertices

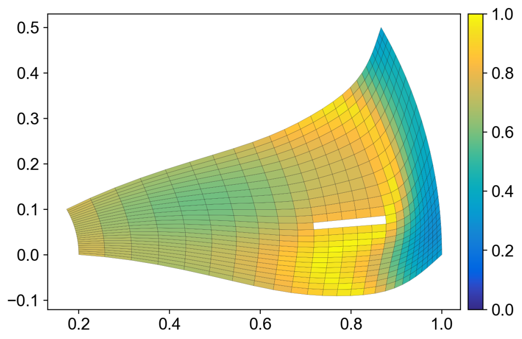
Figure 5.16: Mesh quality of the periodic pseudo sliding RBF interpolation methods using 20 deformation steps for the two-dimensional rotational periodic mesh.



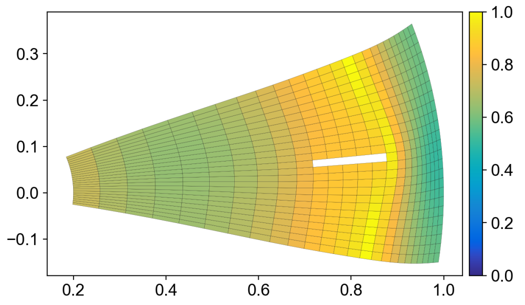
(a) Non-periodic RBF



(b) Periodic RBF



(c) Periodic displacement with fixed vertices

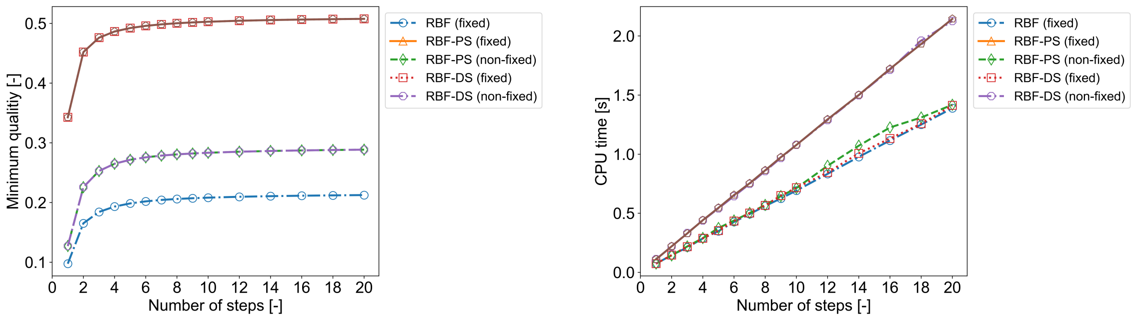


(d) Periodic displacement with non-fixed vertices

Figure 5.17: Mesh quality of the periodic direct sliding RBF interpolation methods using 20 deformation steps for the two-dimensional rotational periodic mesh.

5.2.3. Comparison of the Best Deformation Methods

A comparison is done between the best performing methods of the regular and sliding RBF interpolations in Figure 5.18 in terms of minimum quality and computational effort. An increase of approximately 2.5 times the minimum mesh quality is found when using periodically displaced boundaries with moving vertices in combination with a sliding method compared to a regular RBF interpolation with periodically displaced boundaries. The moving vertices of the domain allow the boundary to slide along with the displacement of the block allowing for a better preservation of the mesh quality. In terms of computational efficiency the pseudo sliding only has a slightly increased computational effort compared to the regular RBF interpolation, due to the additional RBF interpolation and projection steps. The direct sliding is the least computationally efficient method, due to the larger interpolation system. Thus, for the methods considered the pseudo sliding with periodic displacement with moving vertices performs best when considering both minimum mesh quality and computational effort.



(a) Minimum mesh quality **(b)** CPU time

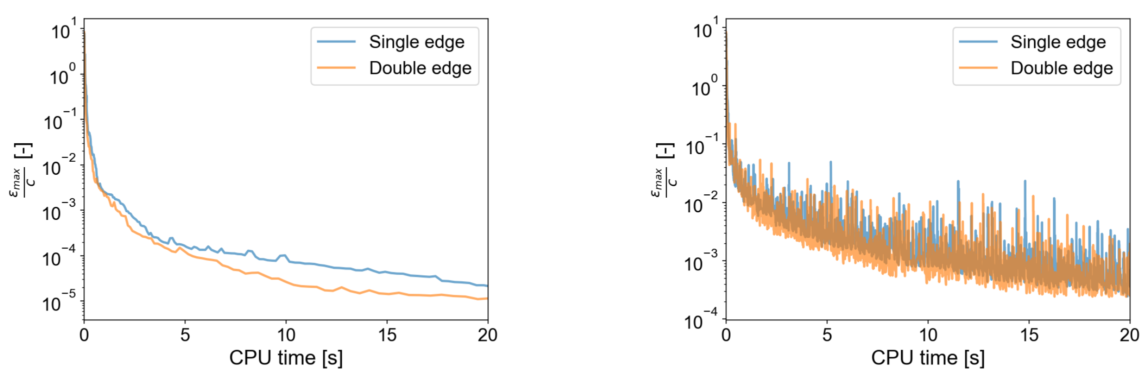
Figure 5.18: Performance comparison of the best achieving regular and sliding RBF interpolation methods for the two-dimensional rotational periodic mesh.

5.3. Impact of Data Reduction Techniques

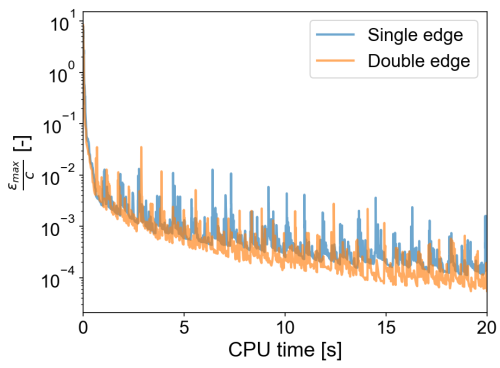
This section shows the results of applying greedy algorithms. This is done with the more dense version of the two-dimensional square mesh. Firstly, a comparison is performed between a single and double-edged selection for the full point greedy algorithm as described in Section 2.5. This is followed by an evaluation of the performance of the multilevel greedy algorithm as control node selection method.

5.3.1. Double-Edged Control Node Selection

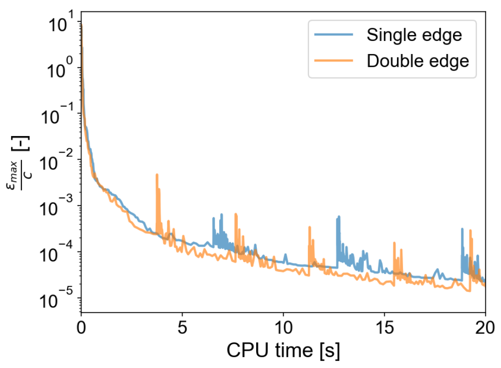
In order to evaluate the performance of the double-edged selection criterion, the error convergence rate of the single and double-edged selection of the control nodes is considered for the single level full point greedy algorithm and the multilevel algorithm with a fixed level sizes of S= 32, 64 and 128. The error convergence rate is a quantity indicating how fast the error is reducing over time. A regular RBF interpolation with a single deformation step is in this case used for obtaining the error convergence rate. The resulting error convergence are given in Figure 5.19 for the considered greedy algorithms. As one would expect using the double-edged criterion results in a faster error convergence. This is seen for both the single as well as the multilevel greedy algorithms. The double-edged greedy selection results in the selection of two control nodes per greedy cycle and therefore the set of control nodes grows faster in size and increase the error convergence rate.



(a) Single level full point **(b)** Multilevel S=32



(c) Multilevel S=64



(d) Multilevel S=128

Figure 5.19: Error convergence of the single and multilevel greedy algorithms with a regular RBF interpolation for the dense square mesh.

5.3.2. Multilevel Greedy Algorithm

The performance of the multilevel greedy algorithm is assessed by considering the error convergence of a regular RBF interpolation for different level sizes and comparing it to a reference of the single level full point algorithm. For both the single level and multilevel greedy algorithms the double-edged selection is used. The error convergence for different level sizes is given in Figure 5.20. The found error convergences of the multilevel greedy algorithms display large variations, where the peaks are coinciding with the initialisation of a new level. Due to this behaviour the single level algorithm performs better for error tolerance up to $\epsilon/c = 2 \cdot 10^{-6}$. Under that error tolerance only a large level size of 256 yields a faster convergence. However, this level size is so large that it negates the principle of using a multilevel algorithm in the first place and only is beneficial for very low error tolerances. In literature this algorithm was applied for external flow applications [40]. Internal flow domains consist of multiple boundaries which often are in close proximity to each other. Due to the nature of the multilevel algorithm, an error can still be induced at the control nodes selected in previous levels. Thus, it is likely that due to the relatively small distance between the boundaries errors are introduced on one boundary due to a control node being selected on a diferent boundary.

In an effort to force a continuously decreasing trend in the error convergence for the multilevel algorithms a dynamic level size was introduced. In that case the algorithm only proceeds to a further level when the error is below a certain fraction of the error at the previous level. This error ratio is defined as: $a_\epsilon = \frac{\epsilon_i}{\epsilon_{i-1}}$. The resulting error convergence for a variety of different a_ϵ is given in Figure 5.21. Even with this additional measure the single level algorithm still has a faster error convergence compared to the multilevel algorithm.

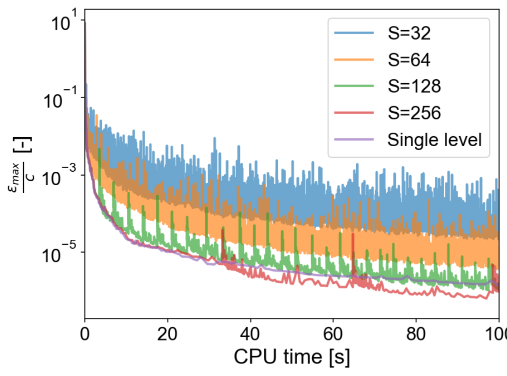


Figure 5.20: Error convergence of the multilevel greedy algorithm with different fixed level sizes for the dense square mesh.

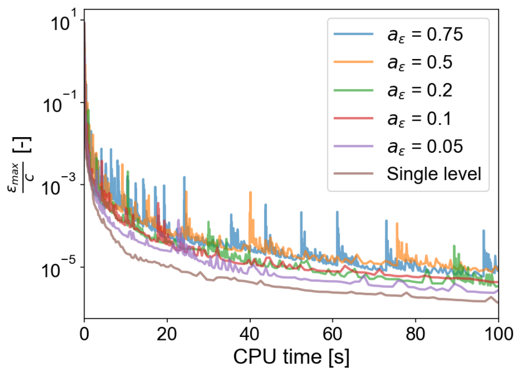
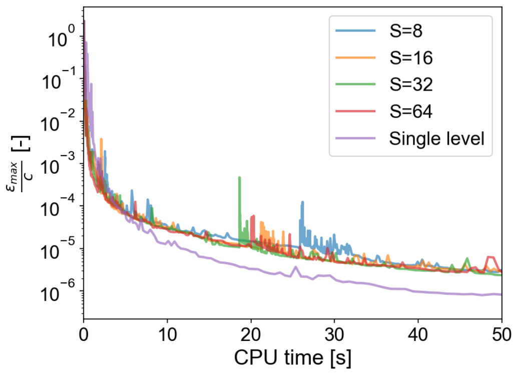
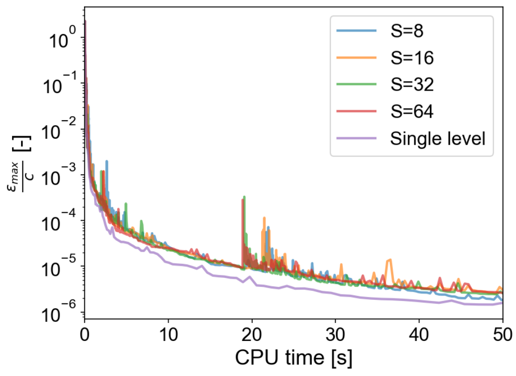


Figure 5.21: Error convergence of the multilevel greedy algorithm with different α_ϵ for the dense square mesh.

The same analysis is performed for the sliding boundary node algorithms to verify whether these produce similar results to the regular RBF interpolation. Figure 5.22 shows the error convergence for different fixed level sizes for the sliding methods. Figure 5.23 on the other hand shows the error convergence for different a_ϵ for both sliding methods. These results indicate that for the sliding RBF interpolation the single level algorithm tends to perform better or at least as good as any of the multilevel greedy algorithms.

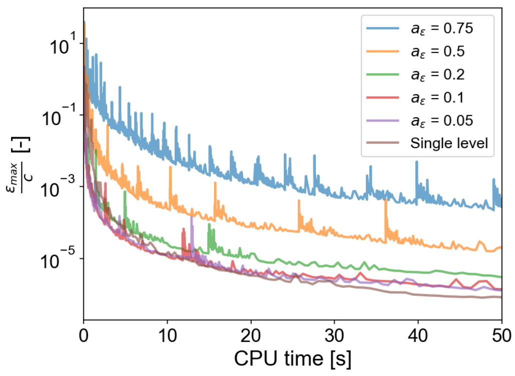


(a) Pseudo sliding

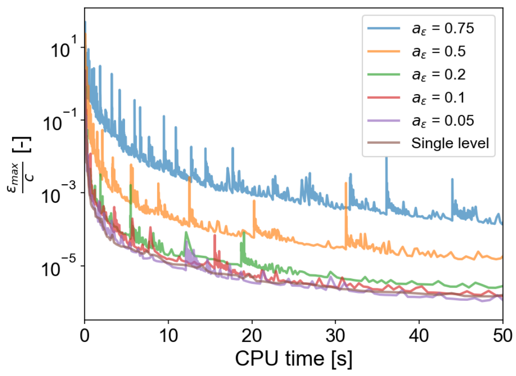


(b) Direct sliding

Figure 5.22: Error convergence for the sliding node RBF interpolations of the dense square mesh with a multilevel greedy algorithm for different fixed level sizes.



(a) Pseudo sliding



(b) Direct sliding

Figure 5.23: Error convergence for the sliding node RBF interpolations of the dense square mesh with a multilevel greedy algorithm for different α_ϵ.

5.4. Turbine Row

The influence of the sliding boundary node algorithms and methods of applying periodicity is evaluated for the more realistic turbine row test case as detailed in Chapter 4. Firstly, the performance of the sliding methods is investigated, followed by an evaluation of the periodic RBF interpolations methods. The deformation applied to the turbine blade is an enlargement of 7% and a translation of 0.001 units in downward direction. Similar to the previous test cases the compact support Wendland CP C^2 function is used as RBF with a support radius of 0.245 units. Unlike the previously considered test cases, the periodic boundaries of the domain are curved. Thus, the curvature correction is required in case of the direct sliding method, to ensure that the sliding nodes remain on their respective boundary.

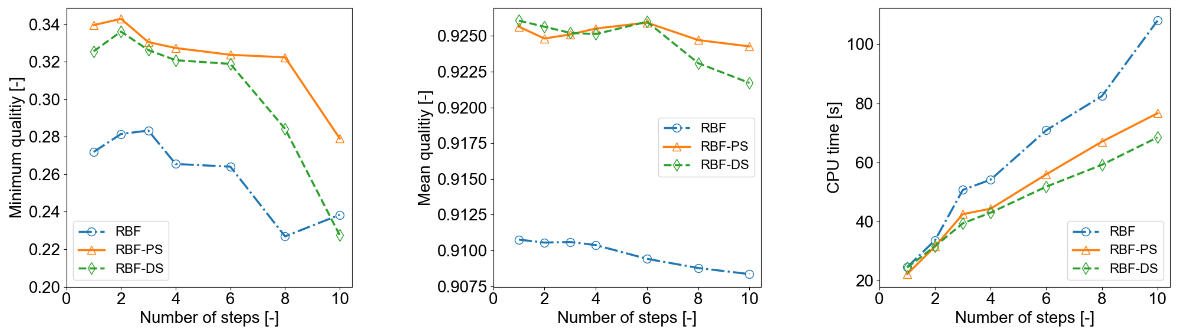
Due to the large number of nodes of the turbine mesh, a greedy algorithm was used in order to reduce the computational time required for the RBF interpolation. In the previous section it was determined that the single level double-edged greedy algorithm yielded the best performance and is therefore selected. As a tolerance criterion it was decided to use a percentage of the dimension of the object in consideration.

Other possibilities for a tolerance criterion could be a percentage of the maximum displacement or mesh size. However, using a percentage of the dimension of the object allows for setting an accuracy that is independent of the deformation applied and the coarseness of the mesh. In this case the tolerance is set at 0.01% of the chord length. This seems a rather strict tolerance, but the distance of the blade to the boundary is relatively small compared to the chord length, which justifies using such a tolerance.

5.4.1. Effect of Sliding Boundary Methods on the RBF Interpolation

The resulting minimum mesh quality, mean mesh quality and CPU times are shown in Figure 5.24, for the regular and sliding RBF interpolations. It can be seen that also for a more complex test case the minimum mesh quality increases when applying the sliding boundary node methods, although this effect can only be seen to a lesser extent. Only small differences in mean mesh quality are found, but the sliding methods perform slightly better compared to the regular RBF interpolation. Furthermore, the minimum mesh quality decreases for all methods of RBF interpolation as the number of deformation steps is increased. This is due to the fact that the deformation performed in each step is smaller for a larger number of steps. As a result the greedy tolerance will be attained for a smaller number of control nodes, which is likely to reduce the accuracy of the mesh deformation and therefore the minimum mesh quality as well.

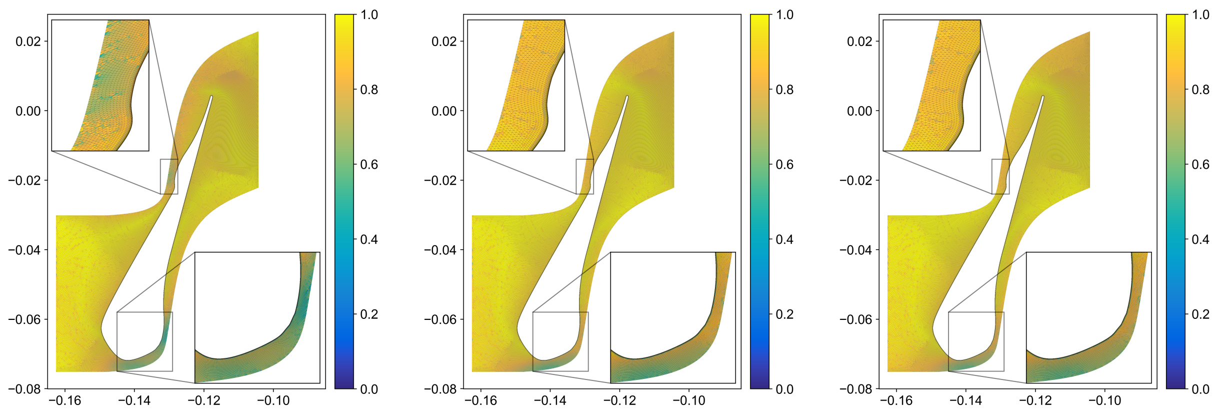
In terms of CPU time the direct sliding algorithms was the most computationally efficient. This indicates that the direct sliding method requires less control nodes to obtain the same error tolerance, as for the same number of control nodes the direct sliding method was proven to be less efficient in previous sections for test cases without data reduction techniques. The reduced amount of control nodes for the direct sliding method can also explain why the minimum quality is lower compared to the pseudo sliding method. Which was not the case for the more simpler two-dimensional cases. Along the same line of reasoning, it can be concluded that the regular RBF interpolation requires larger sets of control nodes as the CPU time proved to be the highest of the considered methods.



(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 5.24: Performance comparison of the regular and sliding RBF interpolations methods for the turbine row mesh.

The resulting mesh quality after two steps is shown in Figure 5.25 for the different methods. It can be seen that the region most affected by the applied displacement is at the leading edge of the blade in the vicinity of the outer boundary. Additionally, a small region near the bump on the top part of the blade is affected. The mesh quality improves in these regions when allowing the boundary nodes to slide within the RBF interpolation, as can be seen in the zoomed-in views of Figure 5.25.



(a) Regular RBF **(b)** Pseudo sliding **(c)** Direct sliding

Figure 5.25: Mesh quality of the regular and sliding RBF interpolations methods using two deformation steps for the turbine row mesh.

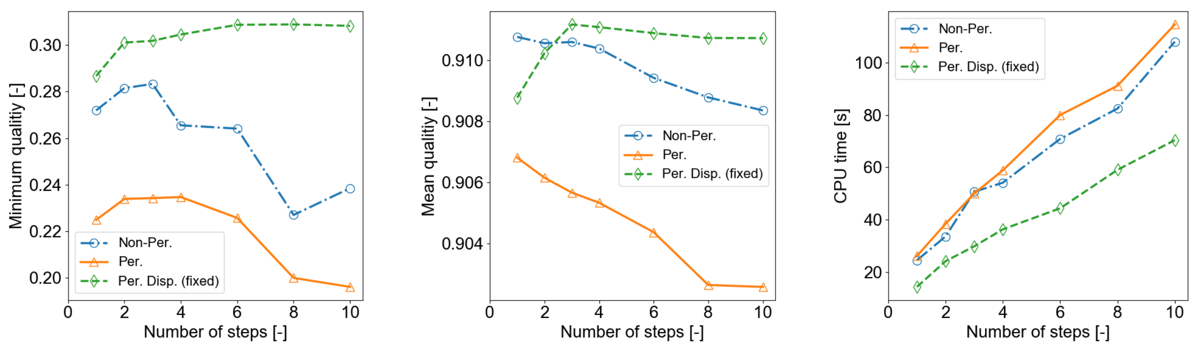
5.4.2. Effect of Periodicity on the RBF interpolation

This section present the results of the RBF interpolation with the inclusion of periodic boundaries for the turbine mesh.

Performance Regular RBF interpolation

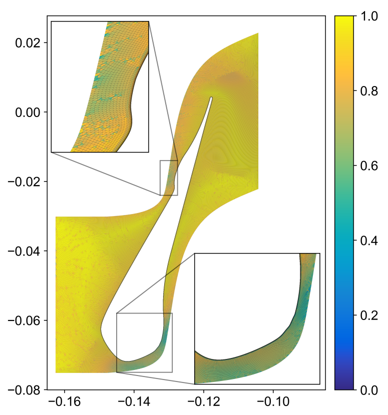
The effects of applying the various periodic methods on the CPU time, minimum and mean mesh qualities are shown in Figure 5.26. The resulting mesh qualities after two steps are given in Figure 5.27. From these results it can be seen that the minimum mesh quality does not improve when using a periodic distance function for the RBF interpolation. At the leading edge it can be seen that, similarly to the square mesh test case, the mesh quality improves at the periodic boundary, but this induces a larger skewness and compression in the region just above it.

Using the periodic displacement with fixed vertices leads to an increase in minimal quality of the mesh. This is best noticed in the region at the lower side of the blade near the leading edge in Figure 5.27c. The treatment of the periodic boundary nodes as if they were internal nodes allows them displace in a similar fashion to the deformation of the turbine blade. Therefore, the skewness and compression at the leading edge is prevented. Similar to the simpler two-dimensional test cases, the periodic displacement of the boundary leads to a reduction of the required CPU time due to the reduction of control nodes.

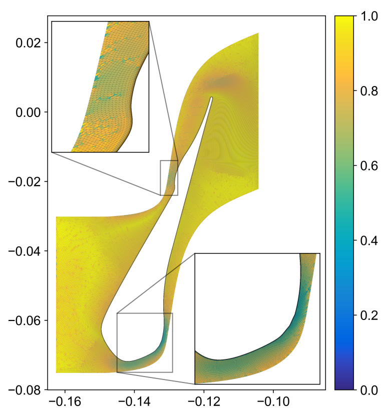


(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

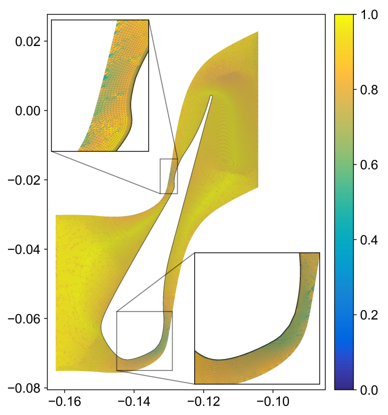
Figure 5.26: Performance comparison of the periodic regular RBF interpolation methods for the turbine row mesh.



(a) Non-periodic RBF



(b) Periodic RBF

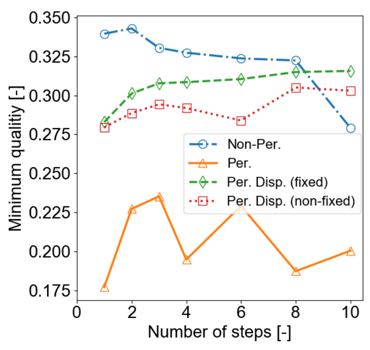


(c) Periodic displacement with fixed vertices

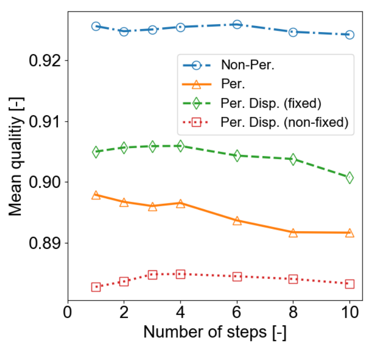
Figure 5.27: Mesh quality of the periodic regular RBF interpolation methods using two deformation steps for the turbine row mesh.

Performance Sliding RBF interpolation

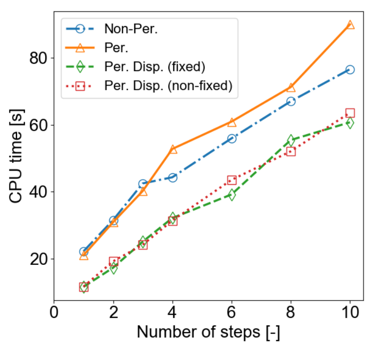
The resulting performance of both sliding boundary node algorithms are shown in Figure 5.28 and Figure 5.29 for the turbine row test case. From these graphs it becomes apparent that for the smaller number of deformations steps the non-periodic RBF interpolation performs best for the sliding methods. For the periodic distance function a decrease in minimum mesh quality was found, as was also the case for the regular RBF interpolation. Furthermore, when applying the periodic displacement methods with fixed or moving vertices the minimum mesh quality does not improve as was the case for the simpler test cases considered previously. The higher minimum mesh quality of the non-periodic sliding RBF interpolation is not directly evident when comparing the non-periodic sliding deformations of Figure 5.30a with the periodic displacement with fixed and moving vertices as shown in Figure 5.30c and Figure 5.30d for the pseudo sliding method. The same can be said when comparing Figure 5.31a with Figure 5.31c and Figure 5.31d for the direct sliding method. What can be noticed is that the region most affected in terms of mesh quality changes from the non-periodic to the periodic RBF interpolations. For the non-periodic RBF interpolations the region most affected is near the periodic boundary at the leading edge of the blade. Whereas for the periodic RBF interpolations the region most affected is at the boundary of the blade itself near the leading edge, which typically is a more critical area. The periodic movement in case of the periodic displacement methods is very limited due to the small clearance between the blade and both periodic boundaries. This likely limits the effect of using a periodic displacement method and hence the expected improvement in minimum mesh quality is not replicated for the turbine row mesh.



(a) Minimum mesh quality



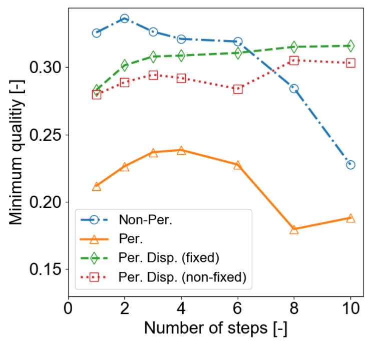
(b) Mean mesh quality



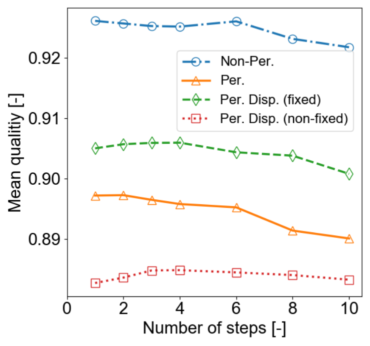
(c) CPU time

Figure 5.28: Periodicity comparisons of minimum mesh quality, mean mesh quality and CPU times for the pseudo sliding RBF as a function of the number of deformation steps for the turbine row mesh.

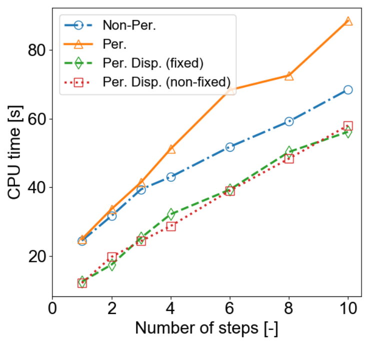
In terms of computational efficiency, the periodic displacement methods clearly are more efficient due to the reduction in the number of control nodes used in the RBF interpolation. The RBF interpolation with a periodic RBF was found to be the least efficient. Due to the periodicity introduced in the domain more control nodes are required to achieve the set greedy tolerance throughout the domain.



(a) Minimum mesh quality

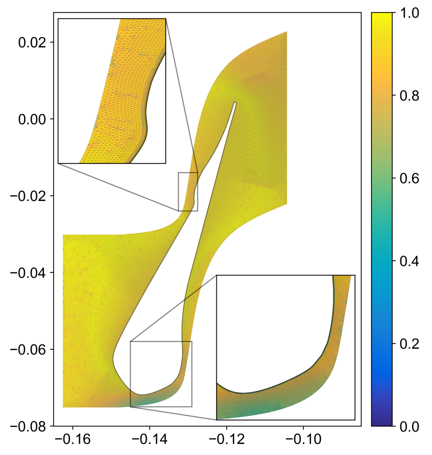


(b) Mean mesh quality

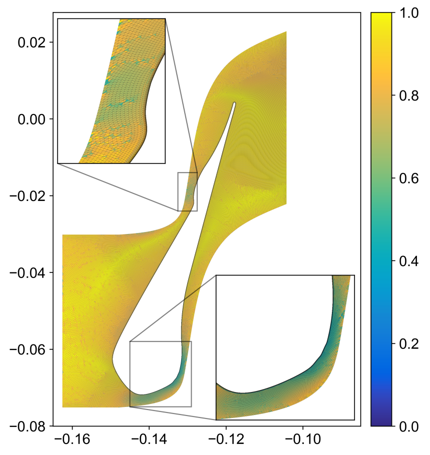


(c) CPU time

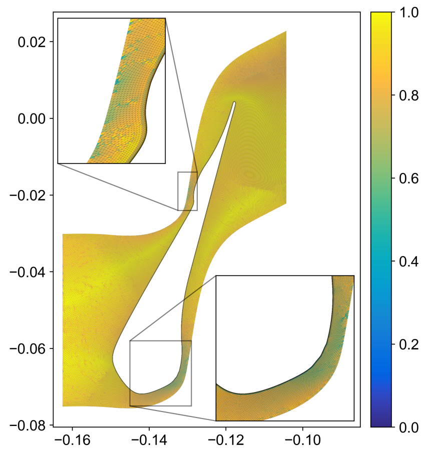
Figure 5.29: Periodicity comparisons of minimum mesh quality, mean mesh quality and CPU times for the direct sliding RBF as a function of the number of deformation steps for the turbine row mesh.



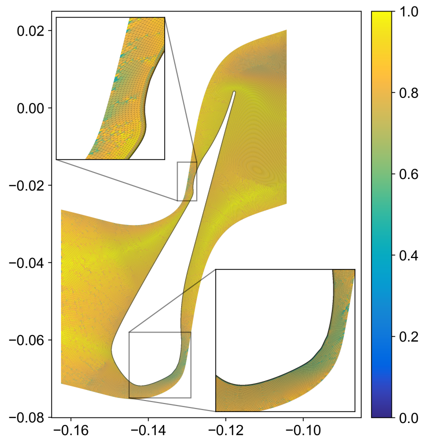
(a) Non-periodic RBF



(b) Periodic RBF

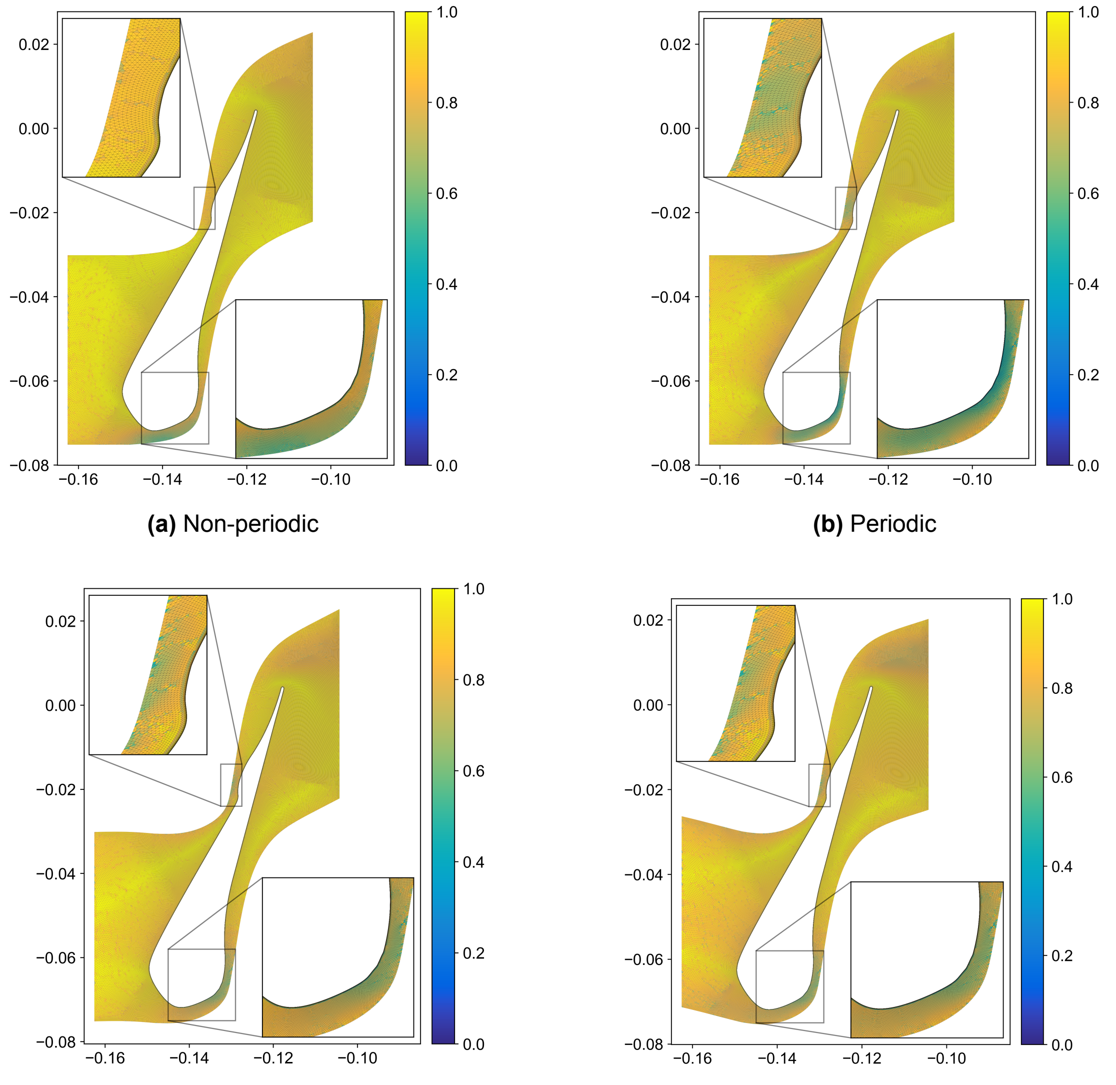


(c) Periodic displacement with fixed vertices



(d) Periodic displacement with non-fixed vertices

Figure 5.30: Mesh quality of the periodic pseudo sliding RBF interpolation using two deformation steps for the turbine row.



(a) Non-periodic

(b) Periodic

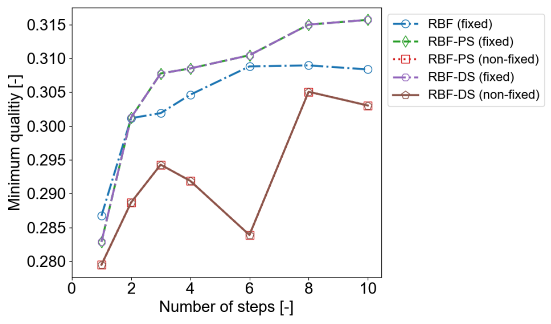
(c) Periodic displacement with fixed vertices

(d) Periodic displacement with non-fixed vertices

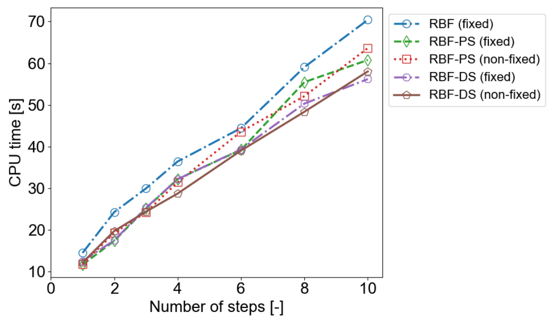
Figure 5.31: Mesh quality of the periodic direct sliding RBF interpolation using two deformation steps for the turbine row.

5.4.3. Comparison of the Best Deformation Methods

For the turbine row the best performing periodic RBF interpolation methods have been selected from the previous sections. It was chosen to only consider periodic methods, since in reality a periodic mesh deformation method would be used for this periodic mesh. A comparison of these methods is made in Figure 5.32 in terms of minimum mesh quality and CPU time. Figure 5.32a shows that the sliding methods with periodic displacement and fixed vertices generate the highest minimum mesh quality of the considered methods. In this case the periodic displacement with moving vertices did not replicate the increase in minimum mesh quality that was seen for the simpler two-dimensional meshes. This is due to the limited periodic displacement because of the small distances between the turbine blade and both periodic boundaries. The minimum mesh quality of the regular RBF interpolation with periodic displacement is only slightly lower compared to the sliding methods. This indicates that the degree of sliding is limited for the applied deformation. Only the left and right boundaries are allowed to slide in case of the periodic displacement methods and since these boundaries are relatively far from the turbine blade the degree of sliding is limited. In terms of computational effort the sliding methods yield very similar results and are more efficient than the regular RBF interpolation.



(a) Minimum mesh quality



(b) CPU times

Figure 5.32: Comparison of the best performing methods for the turbine row mesh.

5.5. Summary Two-Dimensional Results

The sliding RBF interpolation methods proved to produce higher minimum mesh quality compared to the regular variant of the RBF interpolation. For the square mesh the regular RBF interpolation achieved a minimum quality of 0.108 for 20 deformation steps, while the pseudo and direct sliding methods obtained minimum qualities of 0.147 and 0.183 respectively. Similar, behaviour was found for the turbine row mesh, however the difference between the two sliding methods was less significant. Furthermore, the sliding methods proved to be more robust as proven with the test case involving the increasing translation displacement. In terms of computational effort without data reduction techniques the regular and pseudo sliding RBF interpolations proved to be similar in their efficiency. The direct sliding method required more computational effort due to the larger interpolation matrix that has to be solved for. The turbine row test case indicated that when data reduction methods are applied both sliding methods are more similar in terms of efficiency and outperform the regular RBF interpolation method.

The results indicated that the introduction of periodicity in the RBF interpolation can best be done by using the periodic displacement methods. For the simpler test cases these methods produced superior minimum mesh qualities. Allowing the vertices the slide in the periodic displacement proved to yield better minimum mesh qualities than the periodic displacement the fixed vertices. Moving vertices allows the boundaries to move along with the prescribed displacement. This reduces the skewing and compression of the mesh elements as the considered object approaches a periodic boundary. Additionally, the periodic displacement methods are advantageous for the computational efficiency since the set of control nodes is reduced. For the turbine row mesh the periodic displacement method was less successful since the displacement of the periodic nodes was limited by the small clearances to the turbine blade.

The double-edged control node selection proved to be more efficient for the greedy algorithms. Higher convergence rate were found for the double-edged selection for all considered greedy algorithms. The investigation in the single and multilevel greedy algorithms showed that the multilevel greedy algorithm did not provide the expected improvement in error convergence rate. This was the case for both a fixed level size and a set error reduction factor as criteria. It is thought that the close proximity of the boundaries in internal flow problems is the cause for this. As control nodes are selected on one boundary it might introduce an error on a different boundary. As a result the single level greedy algorithm was selected as the optimal control node selection algorithm.

6 Results Three-Dimensional Cases

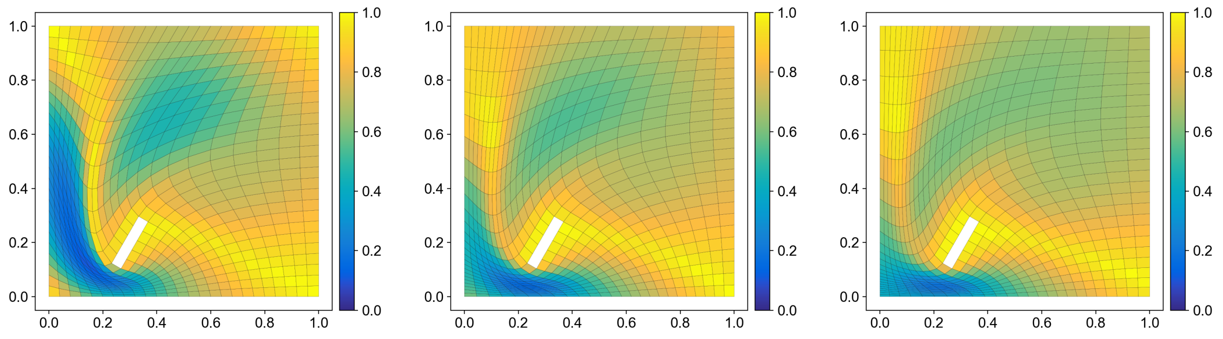
The results obtained for the three-dimensional test cases as described in Chapter 4 are presented in this chapter. Firstly, a comparison is made with between the three-dimensional hexahedron mesh and the two-dimensional square mesh with a similar displacement. Then the effects of applying the sliding boundary node algorithms and different types of periodicity for three-dimensional meshes are discussed. Finally, results of the the more realistic Aachen turbine stator test case are presented. For the Aachen turbine stator mesh, also a CFD simulation is performed with SU2 to investigate the impact of using the periodic sliding RBF interpolation deformation methods on the fluid solver.

6.1. Verification Three-Dimensional Implementation of the Sliding RBF Interpolation

In order to verify a correct implementation of the sliding algorithms for more complex three-dimensional problems a comparison is made between the two-dimensional square mesh and the three-dimensional hexahedron mesh where a similar displacement is applied. The displacement is only in the x and y-direction of the mesh to replicate the two-dimensional displacement of the square mesh. By taking the cross section at $z = 0.5$ of the hexahedral mesh it can be compared to the results obtained by the two-dimensional square mesh. A similar mesh deformation and magnitude of sliding should be found in order to verify the correct implementation of the sliding algorithms in three-dimensional domains.

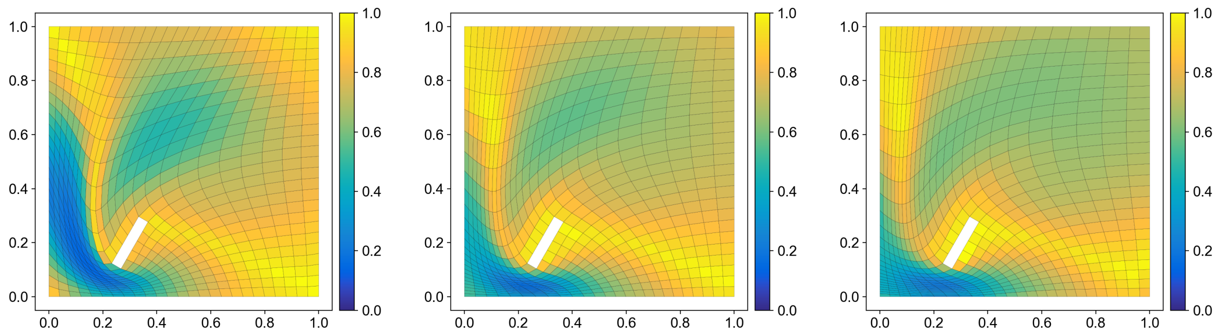
To limit the computational requirements of evaluating the three-dimensional mesh, it was chosen to use the single level double-edged greedy algorithm that was proven to be the most efficient in Section 5.3. An error tolerance of 0.5% of the length of the internal block, equalling 0.001 units, is used to obtain the results as shown in this section. The Wendland CP C^2 function is used as RBF with a support radius of 2.5, which was also the case for the two-dimensional test case. The displacement applied to the internal block to replicate the two-dimensional displacement is a 60 degree anticlockwise rotation around the z-axis and a displacement of -0.3 and -0.2 units in x and y-directions.

The mesh quality as obtained for the square mesh in 20 deformation steps for the regular and sliding RBF interpolation are shown in Figure 6.1. The resulting mesh quality for the hexahedral mesh are presented in Figure 6.2. Based on a visual inspection it is found that the mesh quality for the regular and sliding RBF interpolation methods are nearly identical and that implementation of the sliding algorithms is deemed verified for three-dimensional domains. A small difference that can be noticed is that the minimum mesh quality is slightly higher for the hexahedral mesh in the region below the internal block. This difference is very subtle and is likely caused due to the three-dimensional nature of the hexahedral mesh.



(a) Regular RBF **(b)** Pseudo sliding **(c)** Direct sliding

Figure 6.1: Mesh quality of the regular and sliding node RBF interpolation methods with 20 deformation steps for the tw-dimensional square mesh.



(a) Regular RBF **(b)** Pseudo sliding **(c)** Direct sliding

Figure 6.2: Mesh quality at cross section $z = 0.5$ of the regular and sliding node RBF interpolation methods with 20 deformation steps for the three-dimensional hexahedron mesh.

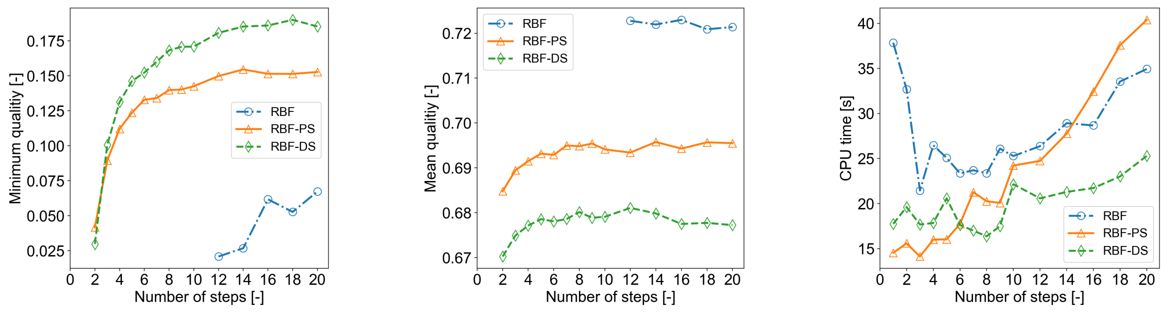
6.2. Hexahedron Mesh

In this section the effects of the sliding boundary node algorithms and translational periodicity are evaluated for the hexahedron mesh with the severe deformation as described in Chapter 4. Again to limit the computational impact of the size of the three-dimensional domain a single level double edged greedy algorithm is employed. The error tolerance is set to 0.5% of the block length, which amounts to 0.001 units. As was the case previously, the Wendland CP C^2 RBF is used in the interpolation with a support radius of 2.5 units. An additional displacement of 0.2 units in z-direction is added to the displacement used for verifying the three-dimensional implementation of the sliding methods. Thus, the total displacement applied is -0.3, -0.2 and 0.2 units in the x,y and z-directions and a 60 degree anticlockwise rotation around the z-axis, as was described in Chapter 4.

6.2.1. Effect of Sliding Boundary Methods on the RBF Interpolation

The minimum mesh quality, mean mesh quality and CPU times as a function the number of deformation steps are found in Figure 6.3. In terms of minimum quality the sliding boundary node algorithms provide significantly higher minimum qualities compared to the regular RBF interpolation, as seen in Figure 6.3a. The direct sliding method yields a better minimum quality with respect to the pseudo sliding method, in particular for a higher number of deformation steps. For 20 deformation steps the regular RBF interpolation achieves a minimum quality of 0.0673, the pseudo sliding RBF interpolation a minimum quality of 0.1530 and the direct sliding RBF interpolation a minimum quality of 0.1852. Furthermore. it was found that the sliding methods are more robust since they can obtain valid meshes for at least two deformation steps. While the regular RBF interpolation obtained non degenerate meshes from twelve deformation steps onwards.

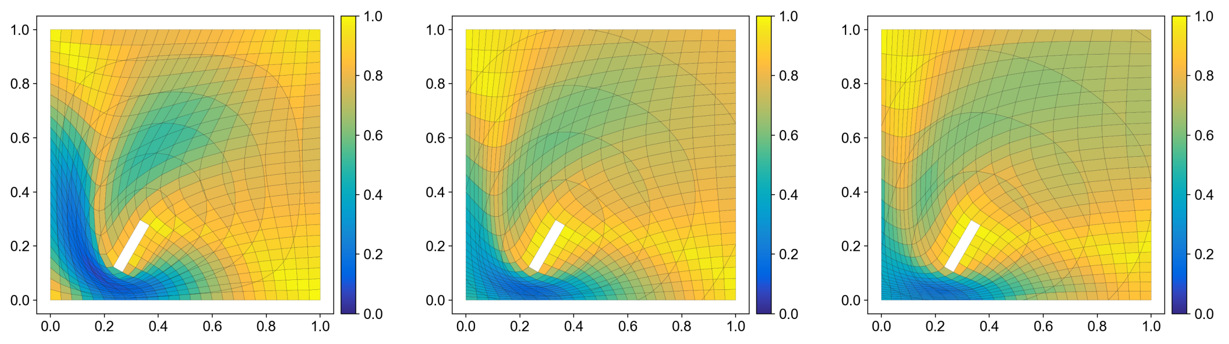
The mean mesh qualities do not follow the same trend as the minimum mesh quality. The regular RBF interpolation achieves the highest mean mesh quality, followed by the pseudo sliding RBF interpolation. However, these variations are considered to be minimal and this reduction in mean quality is insignificant given the increase in minimum mesh quality for the sliding methods. In terms of computationally efficiency the direct sliding algorithm proved to be the best for 6 deformation steps or higher. The pseudo sliding method proved most efficient at a low number of deformation steps but its efficiency reduces drastically for a higher number of steps. Whereas, for the direct sliding the computational time only slightly increases for an increased amount of deformation steps.



(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 6.3: Performance comparison of the regular and sliding RBF interpolations methods for the hexahedron mesh.

Figure 6.4 show the resulting mesh quality at the cross section $z = 0.7$ for the different methods. It should be noted that the circular lines on the plot are a result of the intersection of the elements that are one the cross-sectional plane. In general, similar results are achieved as for the square two-dimensional mesh. The sliding methods provide better mesh quality at the lower left region due to the reduces skewness of the mesh elements as a result of the sliding. Due to the increased magnitude of sliding for the direct sliding method its mesh quality is better in that area compared to the pseudo sliding method, as was also found for the two-dimensional test case.



(a) Regular RBF **(b)** Pseudo sliding **(c)** Direct sliding

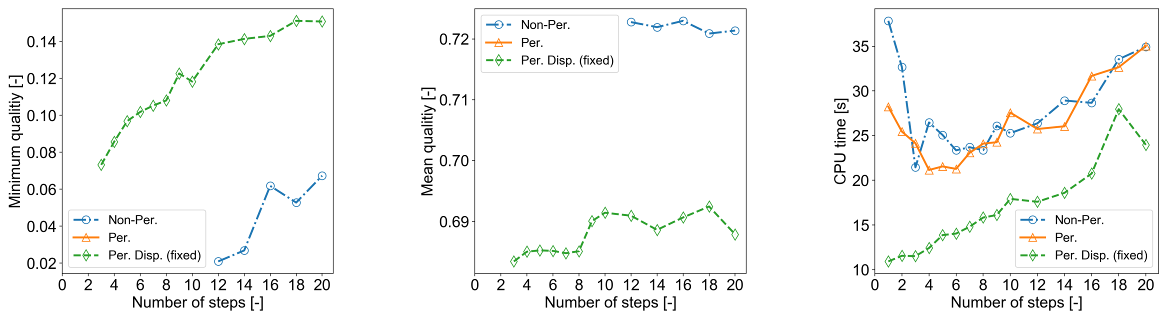
Figure 6.4: Mesh quality at cross section $z = 0.7$ of the regular and sliding node RBF interpolation methods with 20 deformation steps for the hexahedron mesh.

6.2.2. Effect of Periodicity on the RBF interpolation

The results of the inclusion of the translation periodicity for the hexahedron mesh are presented in the following section. The effects on both the regular RBF interpolation as well as on the sliding boundary node algorithms are evaluated.

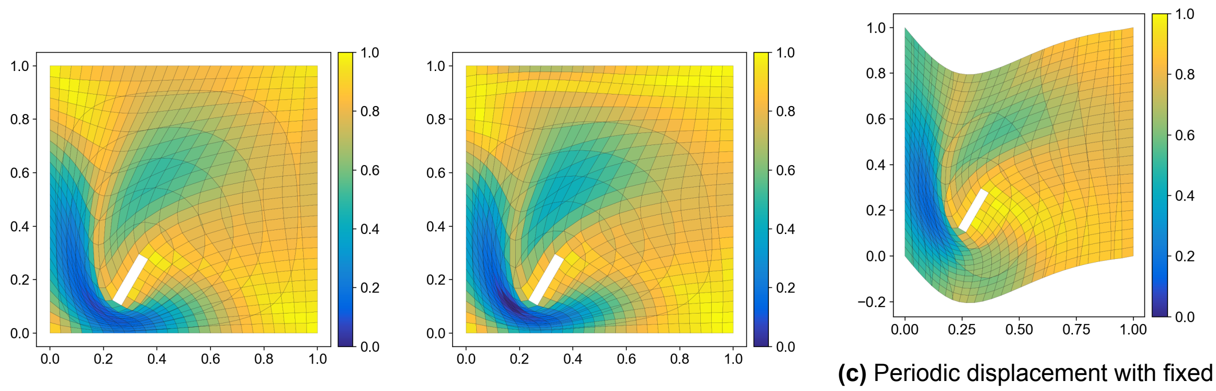
Performance Regular RBF interpolation

The resulting CPU times and minimum and mean mesh qualities as a function of the number of deformation steps for the non-periodic, periodic and periodic displacement with fixed vertices periodic methods are shown in Figure 6.5 for the regular RBF interpolation. From these graphs it can be seen that the use of a periodic distance function does not generate any valid meshes. As was the case for the two-dimensional square mesh a significant improvement in minimum quality results from using the periodic displacement with fixed vertices. For 20 deformation steps the minimum quality is 0.067 for the non-periodic RBF interpolation and 0.151 for the RBF interpolation with periodic displacement and fixed vertices. Furthermore, the periodic displacement proves to be more robust as it can obtain valid meshes for a larger range of deformation steps. Due to the treatment of the periodic boundaries as internal nodes the computational time is reduced compared to the non-periodic RBF interpolation. The resulting mesh quality at the cross section $z = 0.7$ for 20 deformation steps is given in Figure 6.6. For the periodic displacement the mesh quality is improved significantly in the region below the internal block due to the periodic movement of the boundary, leading to the increase in minimum mesh quality.



(a) Minimum mesh quality **(b)** Mean mesh quality **(c)** CPU time

Figure 6.5: Performance comparison of the periodic regular RBF interpolation methods for the hexahedron mesh.

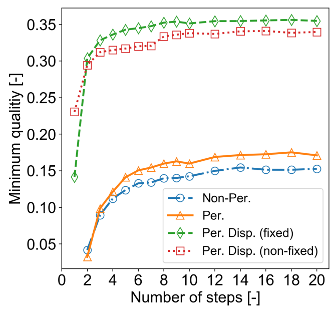


(a) Non-periodic RBF **(b)** Periodic RBF **(c)** Periodic displacement with fixed vertices

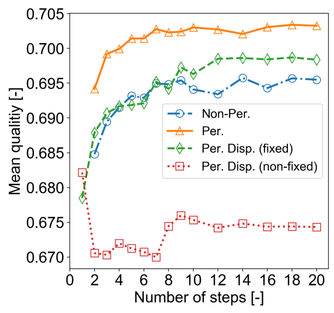
Figure 6.6: Mesh quality at cross section $z = 0.7$ of the periodic regular RBF interpolation using 20 deformation steps for the hexahedron mesh.

Performance Sliding RBF interpolation

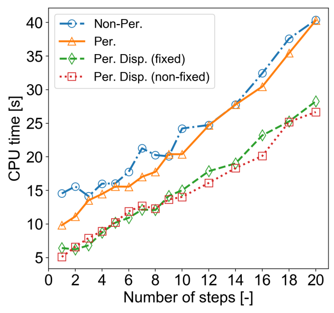
The effects of using periodicity with the sliding methods are presented in this section. The resulting performance of the pseudo and direct sliding methods are given in Figure 6.7 and Figure 6.8. An improvement in minimum mesh quality can be seen when using a periodic distance in the RBF. A minimum mesh quality of 0.1711 is achieved compared to the 0.1527 of the non-periodic pseudo sliding method. A larger and more significant improvement in minimum mesh quality is found for the periodic displacement methods. For the pseudo sliding the fixed vertex form results in a slightly better minimum quality of 0.3548, where the moving vertex variant achieved a minimum quality of 0.3397, While for the direct sliding the non-fixed vertex form generates a better minimum quality of 0.3421, compared to 0.3046 of the periodic displacement with fixed vertices. No large variations in mean mesh quality can be determined in comparison to the change in minimum mesh quality. For the pseudo sliding the periodic displacement methods are clearly more computationally efficient, due to the reduced set of overall control nodes that can be selected by the greedy algorithm. For the direct sliding this is not so evident, where only for a small number of deformation steps it can be said that the non periodic RBF interpolation is the least efficient.



(a) Minimum mesh quality

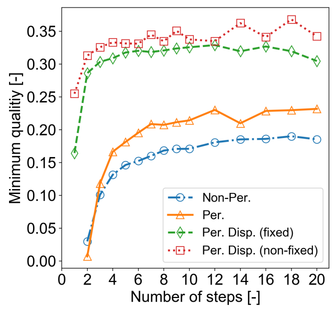


(b) Mean mesh quality

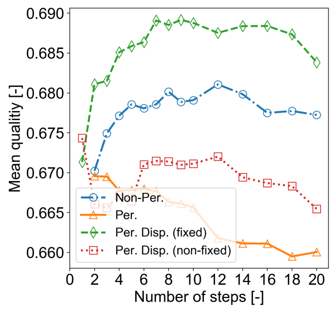


(c) CPU time

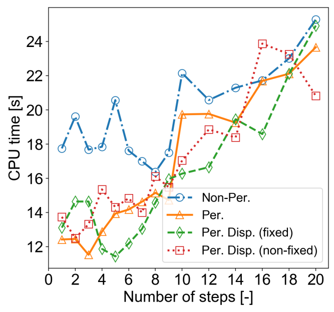
Figure 6.7: Performance comparison of the periodic pseudo sliding RBF interpolation methods for the hexahedron mesh.



(a) Minimum mesh quality



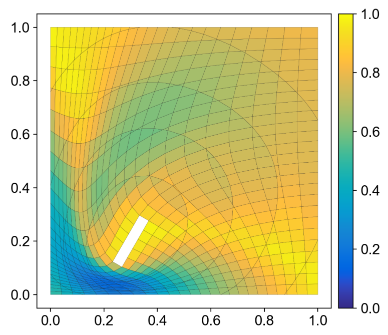
(b) Mean mesh quality

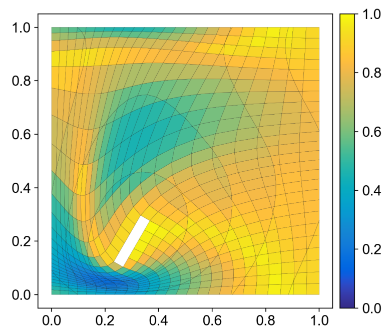


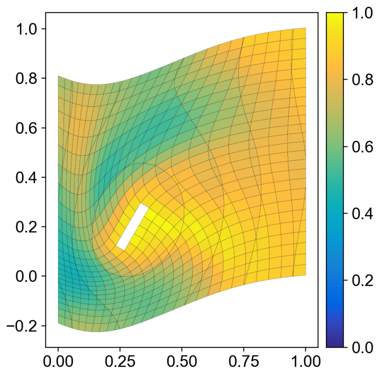
(c) CPU time

Figure 6.8: Performance comparison of the periodic direct sliding RBF interpolation methods for the hexahedron mesh.

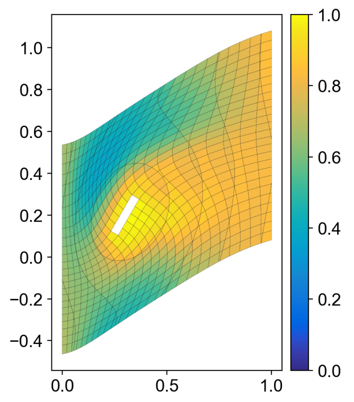
The corresponding mesh quality for the cross section at $z = 0.7$ units and 20 deformation steps for the pseudo and direct sliding methods are shown in Figure 6.9 and Figure 6.10. Visually, no significant differences are found between the pseudo and direct sliding methods. The improvement in using the periodic displacement methods is evident from these figures, the internal block does not come in the vicinity of the periodic boundary. Therefore, the skewing and compression of the mesh elements is prevented in that region.


(a) Non-periodic RBF


(b) Periodic RBF

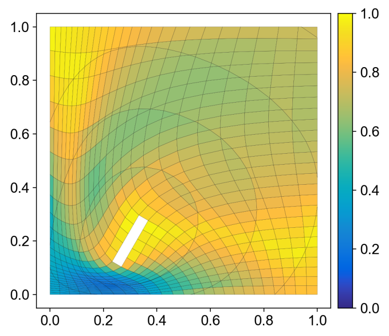


(c) Periodic displacement with fixed vertices

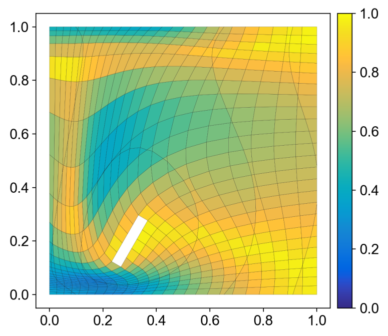


(d) Periodic displacement with non-fixed vertices

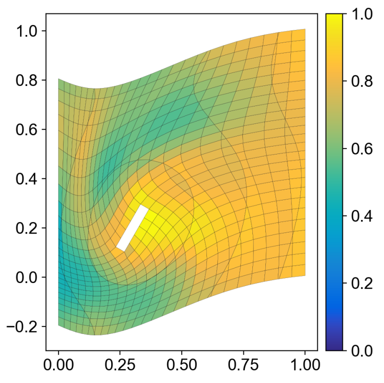
Figure 6.9: Mesh quality at cross section $z = 0.7$ of the periodic pseudo sliding RBF interpolation using 20 deformation steps for the hexahedron mesh.



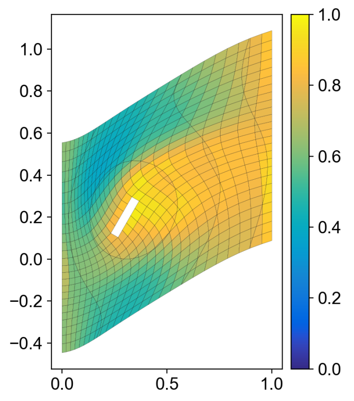
(a) Non-periodic RBF



(b) Periodic RBF



(c) Periodic displacement with fixed vertices

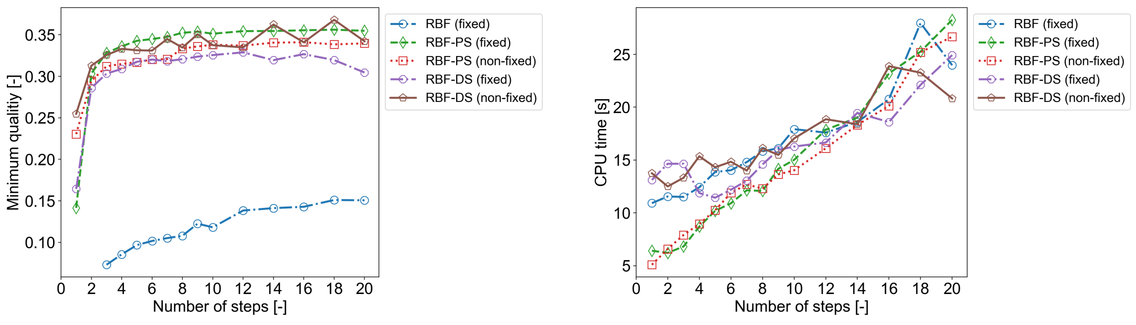


(d) Periodic displacement with non-fixed vertices

Figure 6.10: Mesh quality at cross section $z = 0.7$ of the periodic direct sliding RBF interpolation using 20 deformation steps for the hexahedron mesh.

Comparison of the Best Deformation Methods

From the previous sections a selection of the best performing methods are compared. In this case the RBF interpolations with periodic displacement of the boundaries resulted in the highest minimum mesh qualities. Figure 6.11 gives a comparison of these methods. It is clear that the sliding methods generate a far better minimum quality as compared to the regular RBF interpolation. For 20 deformation steps the regular RBF with fixed vertices periodic displacement achieves a minimum mesh quality of 0,151. The pseudo sliding RBF interpolation with fixed and moving vertices obtained a minimum quality of 0.355 and 0.34. The direct sliding RBF interpolation on the other hand obtained minimum qualities of 0.305 and 0.342 for the periodic displacement with fixed and moving vertices. The highest minimum quality over the range of deformation steps is obtained by the pseudo sliding in combination with a periodic displacement with fixed vertices. In terms of computational effort no clear best performing method can be determined for the higher range of deformation steps. For the smaller number of steps it can be said that pseudo sliding method is the most efficient.



(a) Minimum mesh quality **(b)** CPU time

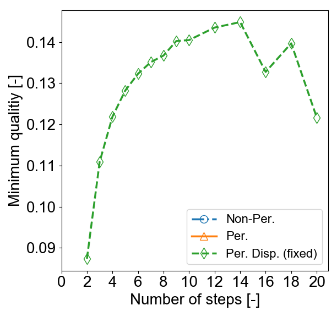
Figure 6.11: Performance comparison of the best achieving regular and sliding RBF interpolation methods for the hexahedron mesh.

6.3. Rotational Periodic Mesh

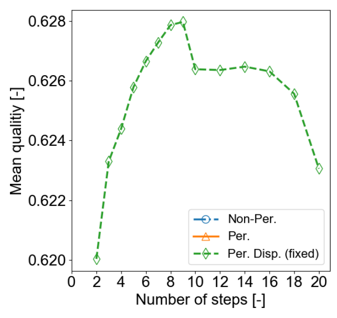
This section details the results obtained for the rotationally periodic mesh for the test case as described in Chapter 4. Similar to the hexahedron mesh a single level double-edged greedy algorithm was used. In order to make somewhat of a comparison with the hexahedron mesh the greedy tolerance was selected to be 0.001 units as well. This amounts to 0.625% of the initial block length. Furthermore, the Wendland CP C^2 RBF was used with a support radius of 2.07 units, which is 2.5 times the maximum domain length. The displacement applied to the internal block is a displacement of 0.2 units is applied in both the r and z-direction in addition to a change in θ of -10 degrees. The effects of the periodicity on the regular RBF interpolation and the sliding boundary node algorithms are discussed in the following sections.

6.3.1. Effect of Periodicity on the Regular RBF interpolation Performance

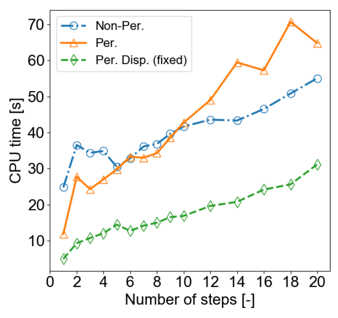
The resulting minimum mesh quality, mean mesh quality and CPU time for the different methods of applying periodicity are given in Figure 6.12 for a range of deformation steps. The mesh quality for a cross section at $z = 0.6$ corresponding to the different forms of the regular RBF interpolation are given in Figure 6.13. For this test case only the regular RBF interpolation with a periodic displacement using fixed vertices was able to produce a valid mesh. Therefore, it is more robust compared to a non-periodic regular RBF interpolation. The information provided by the CPU times once again confirms that in case of using a periodic displacement the computational effort reduces. Considering the mesh quality for the cross section it is clear why only the periodic displacement method attained a valid mesh. The severe translation of the block causes a high compression and skewness of the cells on the lower boundary in the region near the block. For the mesh with periodically displaced boundaries it can be seen that this region has a significantly improved mesh quality. The most affected region is to the right of the block and is caused by the radial outward displacement of the block.



(a) Minimum mesh quality

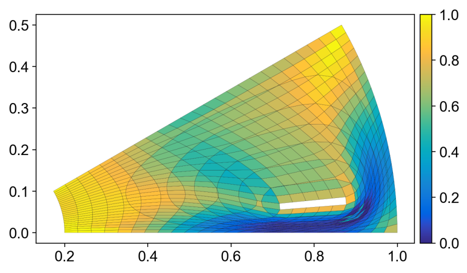


(b) Mean mesh quality

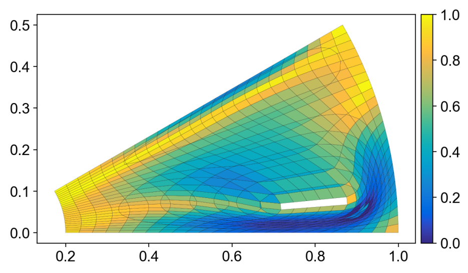


(c) CPU time

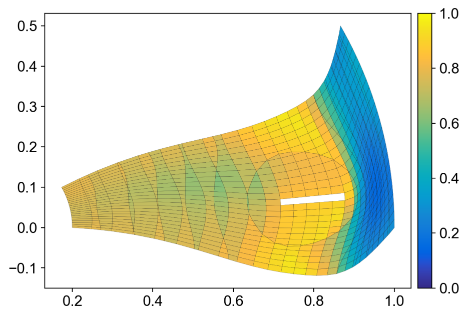
Figure 6.12: Performance comparison of the periodic regular RBF interpolation methods for the 3D rotational periodic mesh.



(a) Non-periodic RBF



(b) Periodic RBF

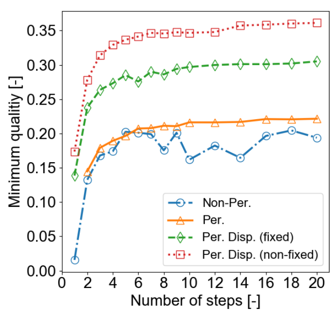


(c) Periodic displacement with fixed vertices

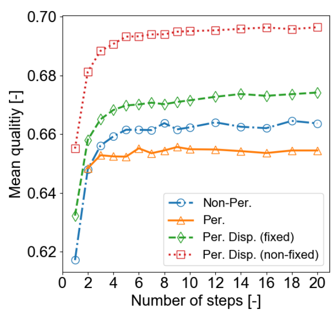
Figure 6.13: Mesh quality at cross section $z = 0.6$ of the periodic regular RBF interpolation method using 20 deformation steps for the 3D rotational periodic mesh.

6.3.2. Effect of Periodicity on the Sliding RBF interpolation Performance

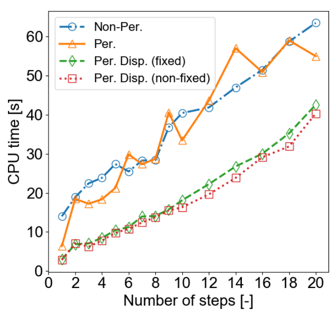
The influence of using the different methods of periodicity on the sliding boundary node methods for the rotationally periodic three-dimensional mesh is discussed in this section. The performance of the sliding RBF interpolation methods with periodicity are shown in Figure 6.14 and Figure 6.15 for the pseudo and direct sliding methods. It can be seen that for both sliding methods the minimum mesh quality slightly increases when only making the distance function of the RBF periodic. However, in case of the periodic RBF with the direct sliding method, no valid meshes could be generated. A significant increase in minimum mesh quality can be observed for the periodic displacement methods. Where the periodic displacement method with moving vertices generates the highest minimum quality. Minimum mesh qualities of 0.3611 and 0.2460 are found for the pseudo and direct sliding methods with periodic displacement and moving vertices. These observations are reflected in the mean mesh qualities, where the same trends are found. In terms of computational time it can be seen that the periodic displacement methods are generally faster compared to the other methods for the pseudo sliding. This is a result of treating the periodic boundaries as internal nodes, the available set of control nodes is smaller and the computational effort will be reduced. This distinction is less clear for the direct sliding method but still noticeable when comparing the periodic displacement method with moving vertices to the non-periodic RBF interpolation.



(a) Minimum mesh quality

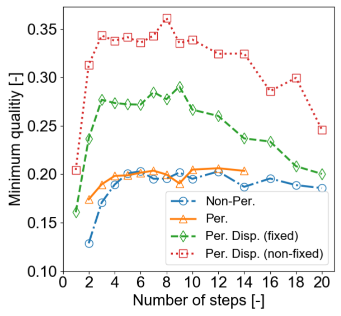


(b) Mean mesh quality

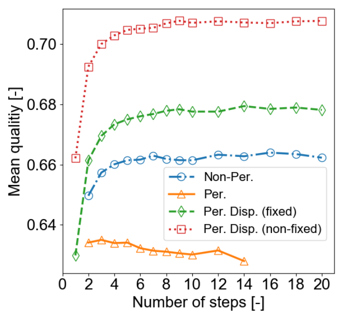


(c) CPU time

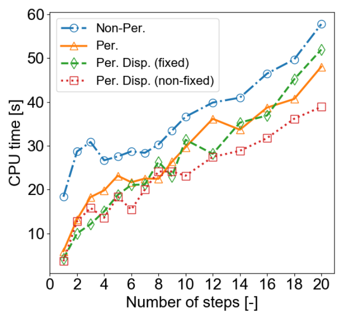
Figure 6.14: Performance comparison of the periodic pseudo sliding RBF interpolation methods for the 3D rotational periodic mesh.



(a) Minimum mesh quality



(b) Mean mesh quality



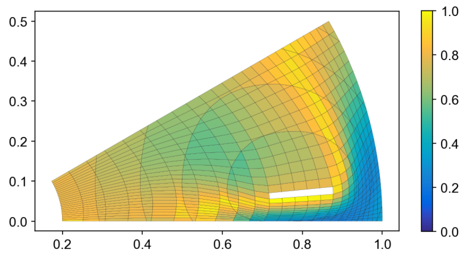
(c) CPU time

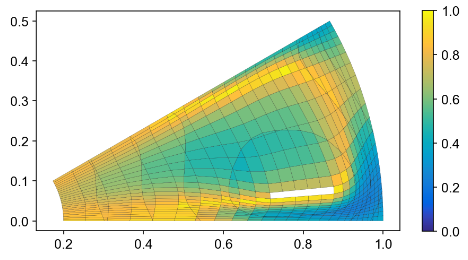
Figure 6.15: Performance comparison of the periodic direct sliding RBF interpolation methods for the 3D rotational periodic mesh.

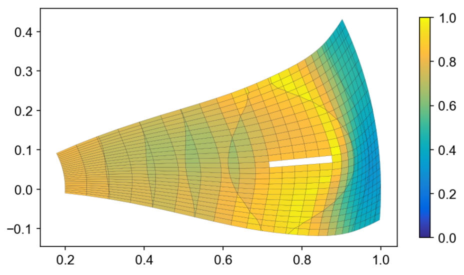
The mesh quality of the cross sections at $z = 0.6$ for 20 deformation steps are presented in Figure 6.16 and Figure 6.17 for the pseudo and direct sliding methods. For the non-periodic and periodic RBF interpolations it can be seen that the direct sliding method generates better mesh qualities compared to the pseudo sliding method in the lower right region of the domain. This is a result of the higher sliding magnitude that was also observed for the two-dimensional square test case, the two-dimensional rotationally periodic test cases and three-dimensional hexahedron test cases. The periodic displacement of the boundaries result in similar mesh qualities at the cross section $z = 0.6$ and visually no significant differences can be detected between the pseudo and direct sliding methods. It is evident that periodic displacement is beneficial in reducing the skewness of the mesh elements at the outer boundary. The degree of sliding is in that case not limited by any fixed vertices of the domain. For the both sliding methods the minimum mesh quality is increased when the vertices are allowed to move compared to the periodic displacement with fixed vertices.

Comparison of the Best Deformation Methods

From the previous sections on the results of the three-dimensional rotational periodic mesh, the best performing methods are selected for a direct comparison. It was evident that for this test case the periodic displacement methods generated meshes with higher qualities compared to the other methods. Figure 6.18 presents a comparison of the periodic displacement methods in terms of minimum mesh quality and CPU times. The sliding methods are able generate a superior mesh quality compared to the non-sliding regular variant. The periodic displacement with non-fixed vertices is the best performing method in terms


(a) Non-periodic RBF


(b) Periodic RBF


(c) Periodic displacement with fixed vertices

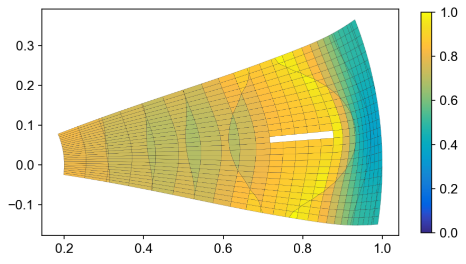
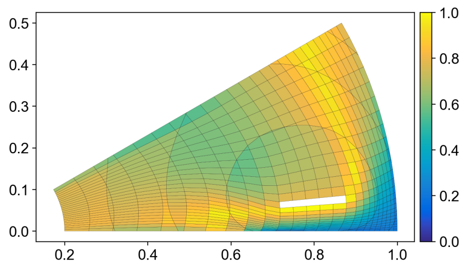
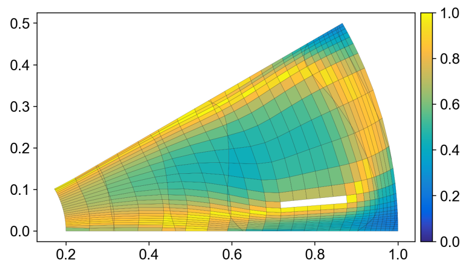
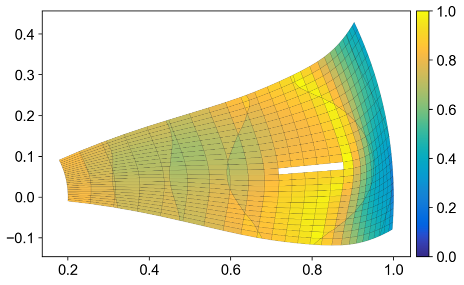

(d) Periodic displacement with non-fixed vertices

Figure 6.16: Mesh quality at cross section $z = 0.6$ of the periodic pseudo sliding RBF interpolation using 20 deformation steps for the 3D rotational periodic mesh.


(a) Non-periodic RBF


(b) Periodic RBF


(c) Periodic displacement with fixed vertices

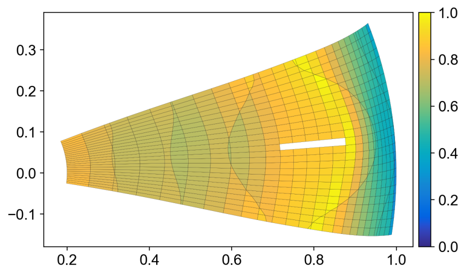
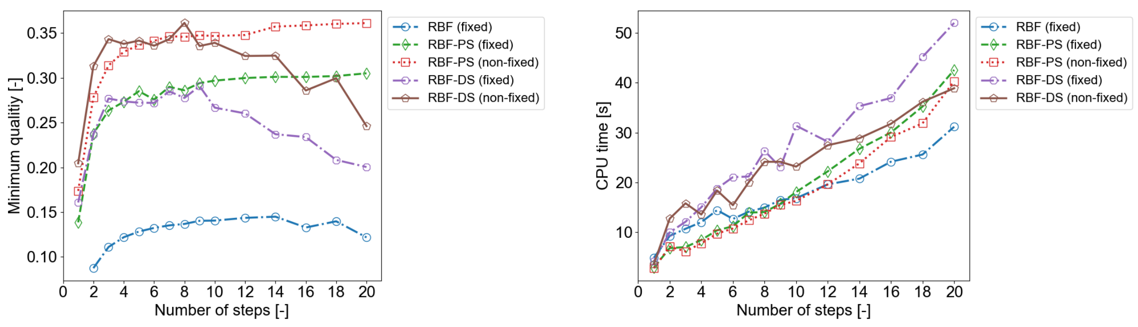

(d) Periodic displacement with non-fixed vertices

Figure 6.17: Mesh quality at cross section $z = 0.6$ of the periodic direct sliding RBF interpolation using 20 deformation steps for the 3D rotational periodic mesh.

of minimum quality. Where the pseudo sliding method seems to be most consistent for the range of deformation steps and thus more robust. Considering the entire range of deformation steps the pseudo sliding method is more efficient compared to the direct sliding method.



(a) Minimum mesh quality **(b)** CPU time

Figure 6.18: Performance comparison of the best achieving regular and sliding RBF interpolation methods for the hexahedron mesh.

6.4. Aachen Turbine Stator

The influence of using the sliding boundary node algorithms as mesh deformation method for the Aachen turbine stator test case will be presented in this section. As was explained in Chapter 4, only the sliding boundary RBF interpolation in conjunction with the periodic displacement method with moving vertices can be applied to the prescribed displacement. Similar to the other test cases, the Wendland CP C^2 RBF is used. The support radius in this case is 0.285806 units, which equals to 2,5 times the maximum domain length. A single level double-edged greedy algorithm is applied and a tolerance of 0.1% of the mean chord length is used. The displacement is obtained by extracting the deformation of the turbine blade from an aerodynamically optimised geometry, as was described in Chapter 4. The resulting displacement is applied in a single deformation step.

6.4.1. Performance Sliding RBF interpolation

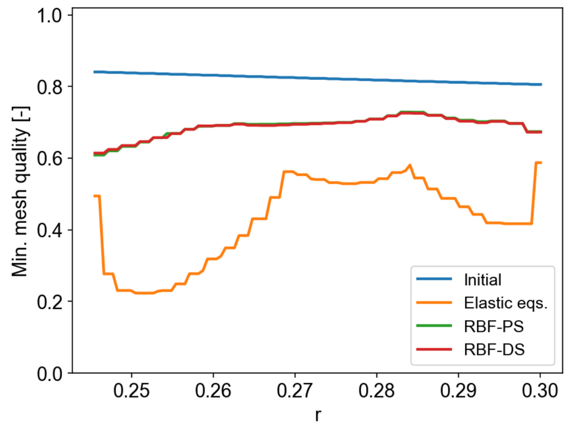
The resulting minimum and mean mesh quality achieved by the sliding boundary node RBF interpolations is reported in Table 6.1, along with the initial mesh quality and the mesh quality obtained by the linear elasticity mesh deformation method.

Table 6.1: Aachen turbine stator mesh qualities.

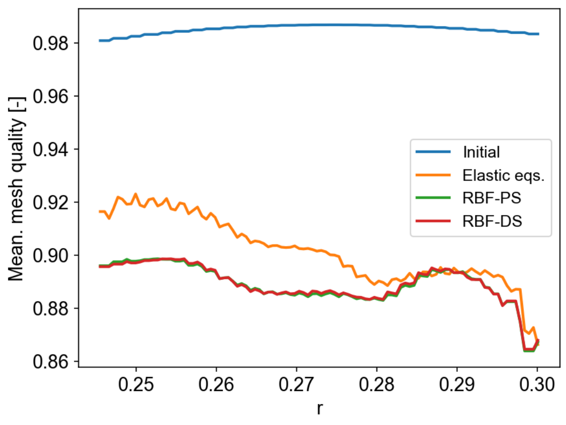
Mesh	Minimum Quality	Mean Quality
Initial	0.80565	0.98520
Deformed linear elasticity equations	0.22330	0.90442
Deformed RBF-PS	0.60859	0.88922
Deformed RBF-DS	0.61407	0.88942

A significant increase in minimum mesh quality can be achieved when using the sliding boundary node algorithms as mesh deformation method instead of the linear elasticity equations method. The minimum mesh quality decreases to 27.7% of the initial value when deforming the mesh with the linear elastic equations. For the pseudo sliding and direct sliding RBF interpolations this decrease is limited to 75.5% and 76.2%. The mean mesh quality is slightly worse for the sliding boundary node methods compared to the linear elastic deformation. However, the minimum quality is the better indicator for the quality of the mesh and therefore the sliding boundary node methods are considered to generate higher quality meshes after deformation. Furthermore, it can be noticed that both sliding boundary node methods produce very similar results in terms of quality. The pseudo sliding RBF interpolation required a CPU time of 306.8 seconds to apply the deformation and the direct sliding RBF interpolation required 767.8 seconds of CPU time. Thus, the pseudo sliding method proves to be the most computational efficient between the two sliding methods for this particular test case,

The radial variation of minimum and mean mesh quality are shown in Figure 6.19. At $r = 0.2512$, an overall minimum mesh quality of 0.2233 is obtained for the mesh deformation using the linear elasticity equations. For the sliding boundary node methods the minimum quality at that radial position is nearly tripled in value, as the minimum quality is approximately 0.61 in those cases. It can be seen that the sliding boundary methods generate larger minimum mesh quality for every spanwise position of the stator blade. Moreover, the variation of minimum mesh quality along the span is more consistent than the one obtained by the linear elasticity equations method. Additionally, a lower mean mesh quality of the sliding boundary node methods can be observed, which is in agreement with the values stated in Table 6.1. However, the minimum mesh quality is typically the best indicator for the quality of a mesh.



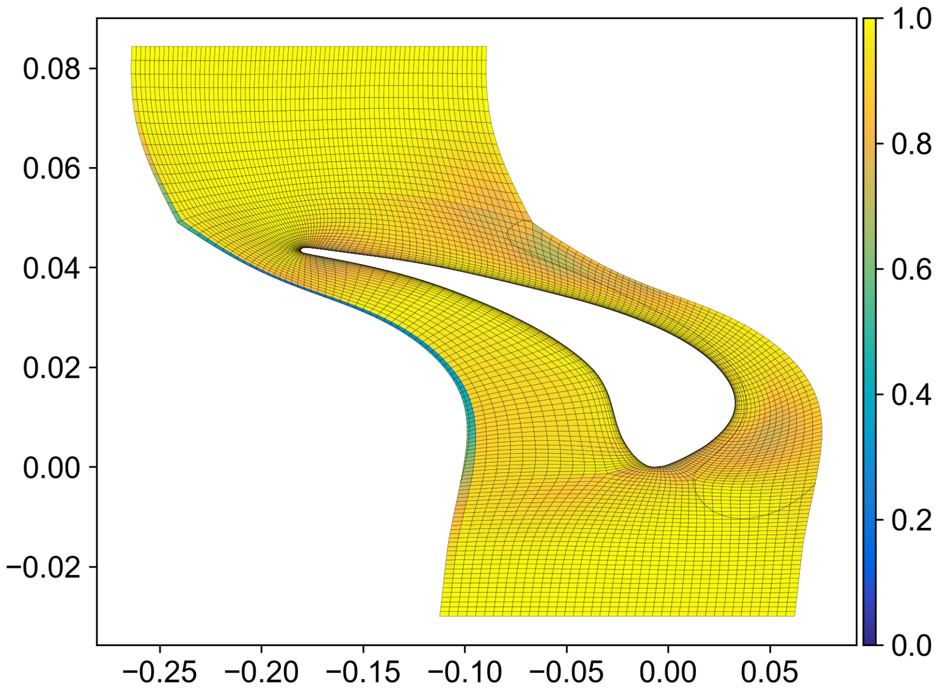
(a) Minimum quality



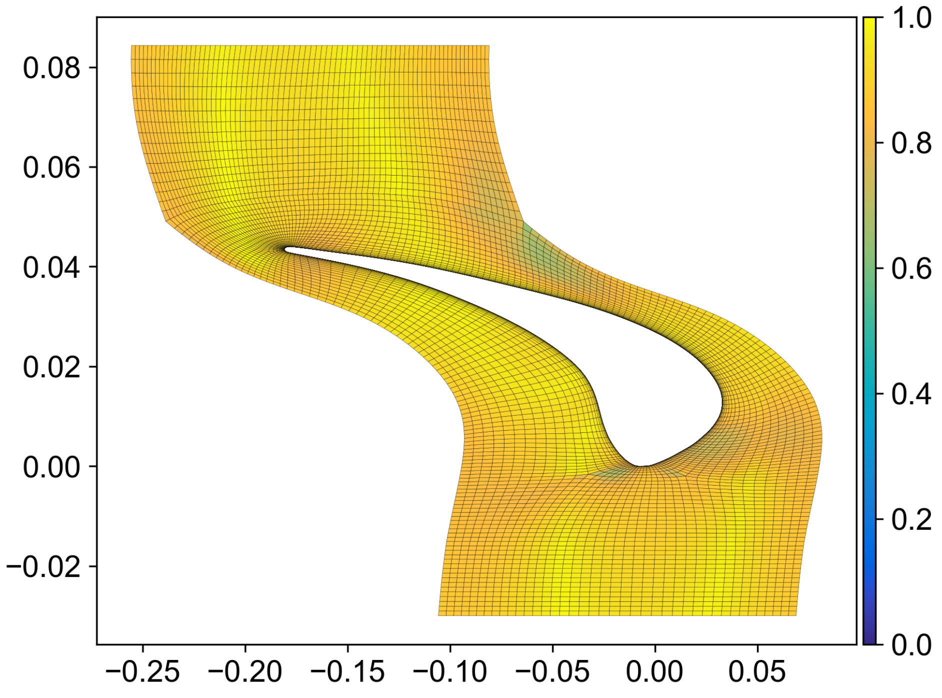
(b) Mean quality

Figure 6.19: Spanwise variation of the minimum and mean mesh quality of the Aachen turbine stator.

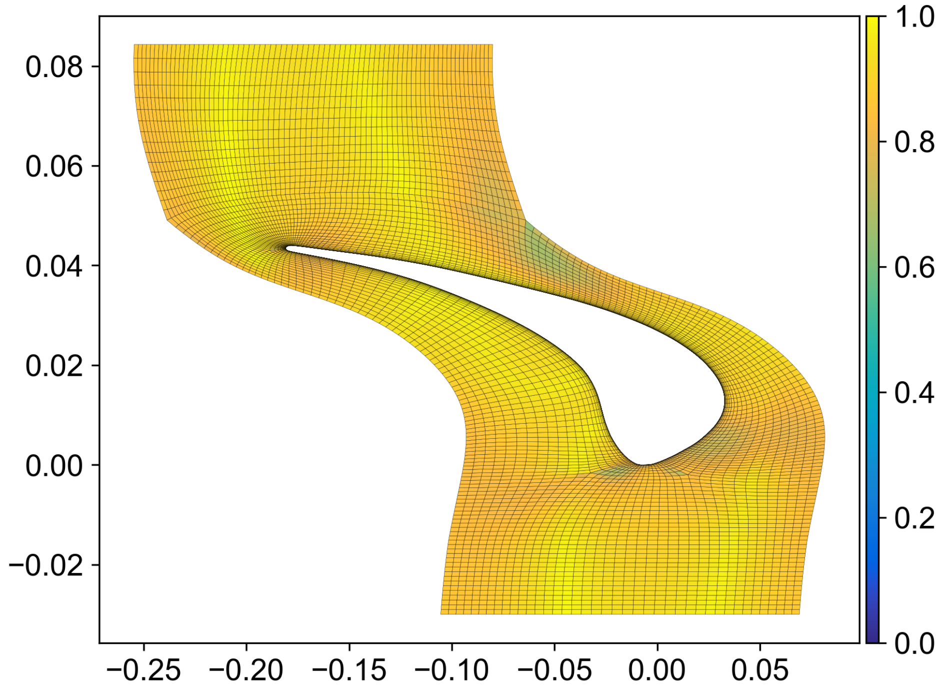
Figure 6.20 shows the mesh quality at the cross section $r = 0.2512$ for the linear elasticity and the sliding RBF interpolation methods. This cross section is selected as at this radial position the minimum mesh quality was found for the linear elasticity method. Figure 6.20 clearly shows that the way the periodicity is dealt with in case of the linear elasticity equations methods is not optimal. A high degree of skewness is introduced for a single row of cells at the periodic boundary, which is not the case for the sliding RBF interpolation methods. This phenomenon can be better seen in the zoomed in views of the periodic interface at the boundary of Figure 6.21, where two periodic domains are placed adjacent to each other. The use of the periodic displacement method in the sliding boundary node RBF interpolations prevents the skewing of the mesh elements and as a result the mesh orthogonality is preserved to a higher degree, as seen in Figure 6.22 and Figure 6.23 for the pseudo and direct sliding methods. In Figure 6.22 and Figure 6.23 the periodic boundary can no longer be distinguished due to the smooth transition over the periodic interface due to the periodic RBF.



(a) Linear elastic equations



(b) Pseudo sliding



(c) Direct sliding

Figure 6.20: Mesh quality of cross section at $r = 0.2512$.

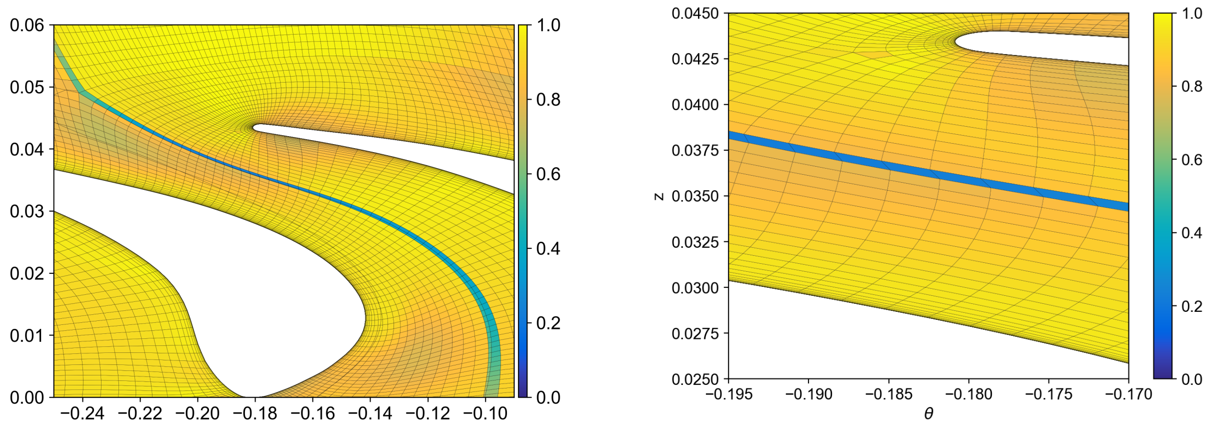


Figure 6.21: Zoomed in view of mesh quality at cross section $r = 0.2512$ for the linear elasticity equation method.

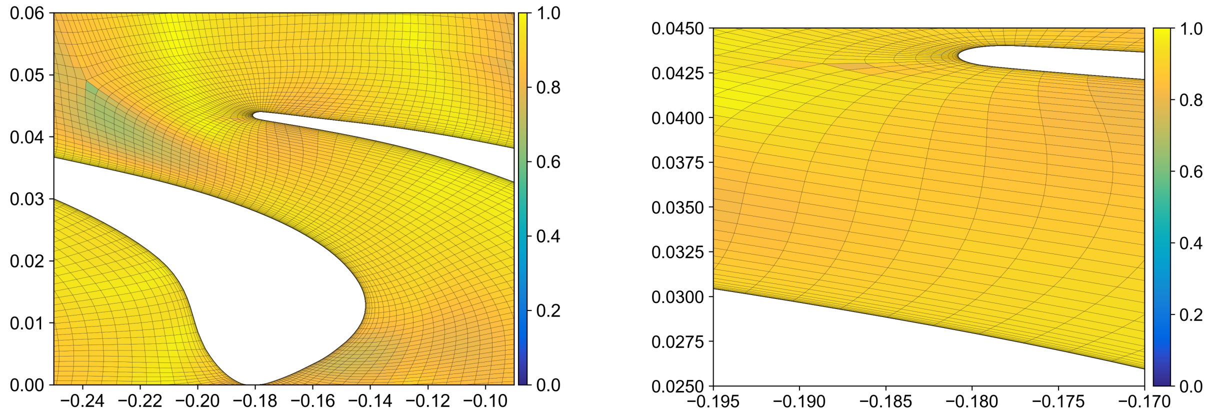


Figure 6.22: Zoomed in view of mesh quality at cross section $r = 0.2512$ for the pseudo sliding method.

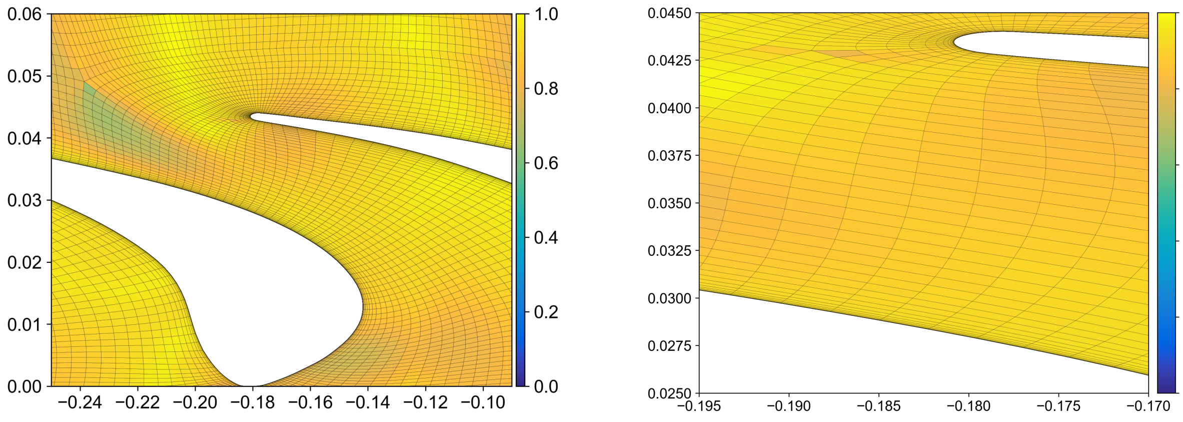


Figure 6.23: Zoomed in view of mesh quality at cross section $r = 0.2512$ for the direct sliding method.

6.4.2. Effect of the Periodic Sliding RBF Interpolation on the Fluid Solver

In the previous section it was found that the minimum mesh quality was significantly higher for the sliding RBF interpolations. However, this improvement in minimum mesh quality was mainly at the periodic boundary and not in the close vicinity of the stator blade. In this section the impact of using the sliding RBF interpolation with the periodic displacement including moving vertices on the fluid solution as computed with SU2 is investigated. A simulation will be performed with the RANS solver of SU2 for the mesh of the optimised stator blade geometry and the meshes obtained with the sliding RBF interpolation methods. The turbulence is modeled using the k-ω Shear-Stress-Transport (SST) turbulence model [46]. At the inlet the boundary condition applied is a total pressure of 1.696$\cdot 10^5$ Pa, a total temperature of 305.76 K and the flow direction is normal to the inlet. At the outlet a static pressure of 1.2$\cdot 10^5$ Pa is prescribed as boundary condition. A Roe upwind scheme is used for the discretisation of the convective terms. Furthermore, a Courant–Friedrichs–Lewy (CFL) number of 50 is used in conjunction with an implicit Euler time marching scheme.

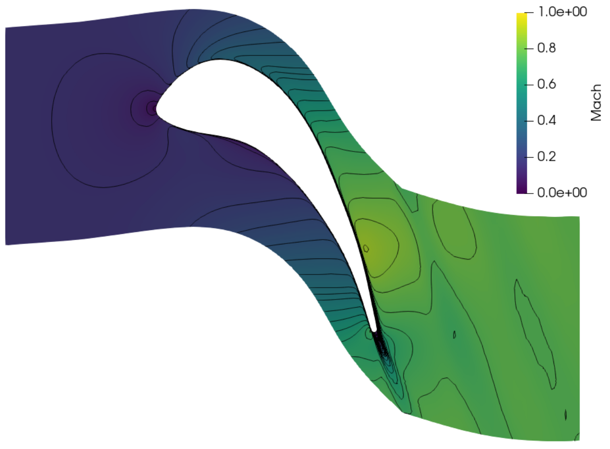
Table 6.2 shows the percentages of total pressure loss and generated entropy for the different mesh deformation method. It can be noticed that both sliding methods achieve nearly identical results. Which is what was expected since their mesh quality parameters were very similar. Comparing the sliding methods with the elasticity equation methods, it can be observed that the total pressure loss and entropy generated is slightly lower for the elasticity equations method. However, it is hard to pinpoint whether this small difference is actually related to the mesh quality or not.

Table 6.2: Percentage of total pressure loss and entropy generated for the Aachen turbine stator as found by the SU2 RANS simulation for deformed meshes using the elasticity equation or sliding RBF methods.

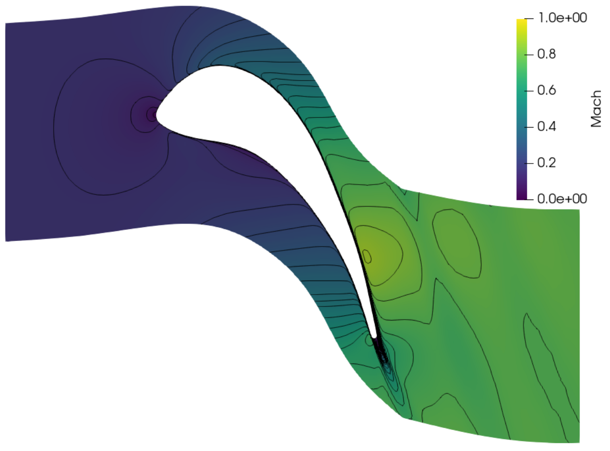
	Elasticity equations	Pseudo	Direct
Percentage of total pressure loss	3.28011%	3.30446%	3.30669%
Percentage entropy generated	0.07022 %	0.07089%	0.07092 &

Figure 6.24 show the resulting Mach number distribution for the linear elasticity method, the pseudo sliding method and the direct sliding method. The cross section considered is the one where the linear elasticity method obtained its lowest mesh quality at x=0.248. It can be observed that in general the resulting Mach number distributions look nearly identical. Looking at the contour lines some small changes can be noticed downstream of the blade, however no significant change in flow solution can be found.

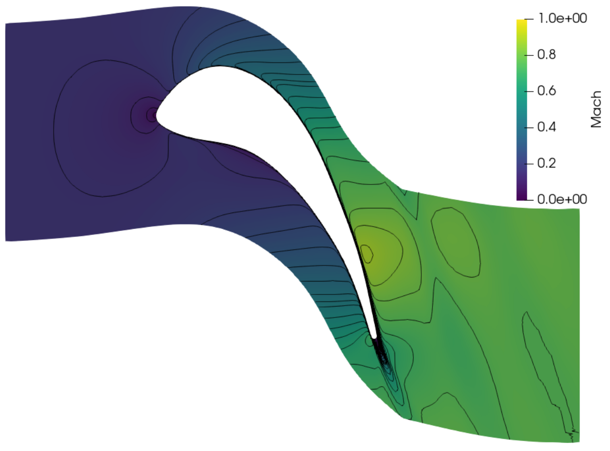
The resulting pressure distribution at the same cross section for the different deformation methods is presented in Figure 6.25. The distributions of the pressure coefficient are nearly identical as well and therefore it can be concluded that the increase of minimum mesh quality of the sliding RBF interpolation method had very little effect on the flow solution as obtained by SU2.



(a) Linear elasticity



(b) Pseudo sliding



(c) Direct sliding

Figure 6.24: Mach number distribution obtained by an SU2 RANS simulation for the Aachen stator mesh deformed with the linear elasticity. pseudo sliding RBF and direct sliding RBF methods.

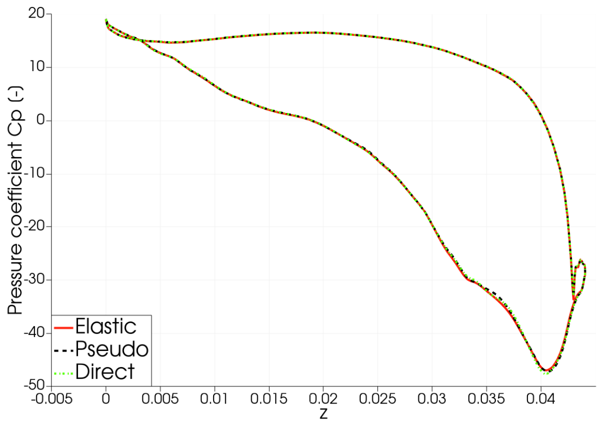
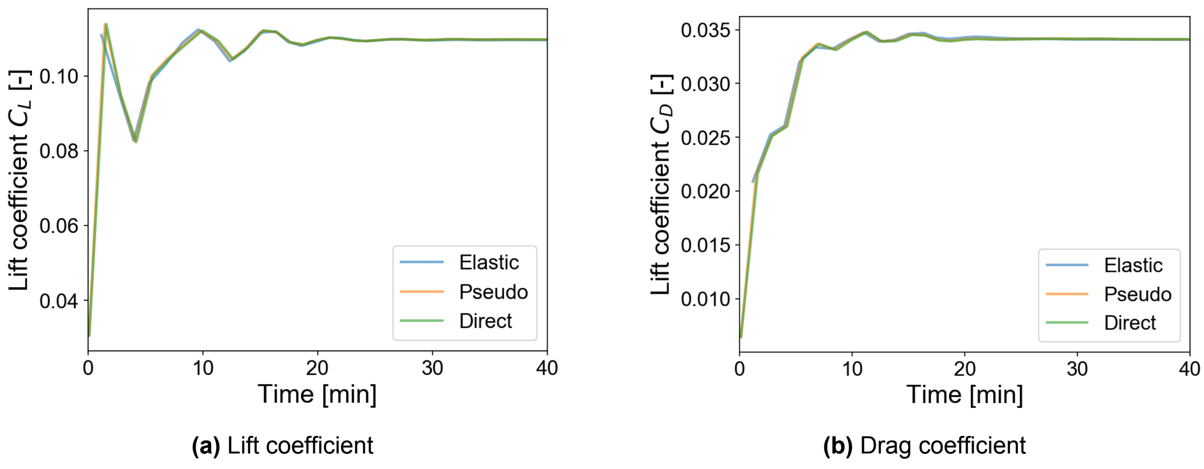


Figure 6.25: Pressure distribution of the Aachen stator blade as found by the SU2 RANS simulations.

Lastly, the convergence of the lift and drag coefficient at the cross section $x = 0.248$ is given in Figure 6.26. Once again, no significant difference are noticed between the linear elasticity deformation method and the sliding RBF interpolation methods. This suggest that the improved mesh quality of the sliding methods have very little effect on the convergence of the fluid solver for this particular case. The reason for this is likely the fact that the region in which the mesh quality is improved is not close enough to the considered object to influence the fluid solution.



(a) Lift coefficient

(b) Drag coefficient

Figure 6.26: Aerodynamic coefficient convergence of the SU2 RANS simulation for the elasticity equations, pseudo sliding and direct sliding deformation methods.

6.5. Summary Three-Dimensional Results

The effect of using sliding boundary nodes in the RBF interpolation on the mesh quality are similar to those found in the two-dimensional results. The minimum mesh quality increases significantly from 0.067 for the regular RBF interpolation to 0.1530 and 0.1852 for the pseudo and direct sliding methods after 20 deformation steps. Additionally, the sliding methods proved to be more robust since valid meshes were generated for fewer deformation steps compared to the regular RBF interpolation. Due to the use of data reduction methods, no single method could be identified as most efficient. The pseudo sliding performed best at fewer deformation steps and the direct sliding method for a higher number of deformation steps.

In agreement with the results of two-dimensional test cases it was observed that the periodic displacement methods yielded superior minimum mesh qualities compared to the non-periodic and periodic RBF methods. For the regular RBF interpolation the minimum mesh quality increased from 0.0673 to 0.1507, when going from the non-periodic to a periodic displacement with fixed vertices method. Allowing the boundary nodes to displace in a periodic manner prevents the deterioration of the mesh quality in case the considered object nears a periodic boundary. For the translational periodicity both periodic displacement methods proved to yield similar results, where minimum qualities of 0.3548 and 0.3421 are obtained for the pseudo and direct sliding methods. In case of a rotational periodic domain the periodic displacement with moving vertices outperformed the one with fixed vertices, since the sliding on the outer radius of the considered domain was no longer limited by the fixed vertices. Which reduced the skewing of the mesh elements. In terms of computational efficiency the periodic displacement methods are performing best. This is due to the smaller set of control nodes which are considered, which is in agreement with the results of the two-dimensional domains.

The use of a sliding and periodic RBF interpolation as mesh deformation method yielded improved minimum mesh quality for the Aachen turbine test case, when compared with the linear elasticity mesh deformation method. For the linear elasticity method a minimum mesh quality of 27.7% of the initial minimum mesh quality was found. While for the pseudo and direct sliding methods minimum qualities that where 75.5 and 76.2% of the initial minimum quality are achieved. The linear elasticity method induced large skewing near a periodic boundary which was prevented with the periodic RBF as used in the sliding methods. The sliding methods yielded very similar results, and showed an improvement in terms of minimum mesh quality over the entire span of the stator blade compared to the linear elasticity method. The pseudo sliding method proved to be computationally more efficient than the direct sliding method, as indicated by their CPU times of 306.8 seconds and 767.8 seconds.

The simulation performed by SU2 with the meshes deformed with linear elasticity and the sliding methods resulted in only small differences identical results in terms percentage of total pressure loss, percentage of entropy generation, Mach number distribution, pressure distribution of the stator blade and solver convergence of the lift and drag coefficients. Thus, the improved minimum mesh qualities of the sliding method did not have a significant impact on the fluid solver. The region where the mesh quality was improved was near the periodic boundary. It is likely that this was not sufficiently close to the stator blade to have an impact on the resulting fluid solution.

7 Conclusion

The aim of this work was to develop a robust and computationally efficient mesh deformation method suitable within the discrete adjoint optimisation framework of SU2 for internal flow applications by means of developing an implementation of the Radial Basis Function interpolation method including sliding boundary node algorithms, periodic boundary conditions and data reductions methods. The developed RBF-SliDe proved to be more robust as higher mesh qualities were found and more severe deformations could be applied compared to a regular RBF interpolation. The skewing of the mesh elements was reduced through the adaptation of the sliding boundary node methods, which allows the boundary nodes to slide along with the prescribed deformation. The inclusion of the periodic displacement method with moving vertices allowed the periodic boundary to be displaced according to the displacement of the considered object. This prevented the deterioration of the mesh quality due to skewing and compression of the mesh elements as an object approached an periodic boundary. Based on the obtained results for the various test cases presented in Chapter 5 and Chapter 6. The following conclusions are drawn.

- **The inclusion of sliding boundary node methods in the RBF interpolation resulted in an improvement of minimum mesh quality compared to the regular RBF interpolation.**

Due to the sliding motion of the boundary nodes the skewing of the cells near the boundary is reduced and the minimum mesh quality is increased, The degree of sliding is higher for the direct sliding method and as a result it generated the highest minimum qualities. In case of the two-dimensional square mesh the minimum mesh quality was improved from 0.108 for the regular RBF interpolation, to values of 0.147 and 0.183 for the pseudo and direct sliding methods. This effect also translated to the three-dimensional hexahedron test case where the minimum mesh quality increased from 0.0673 for the non-sliding RBF to 0.1530 and 0.1852 for the pseudo and direct sliding RBF interpolation methods. Furthermore, the sliding RBF interpolation proved to be more robust than the regular RBF interpolation. The sliding methods were able to obtain valid meshes larger deformations due to the sliding. In the three-dimensional hexahedron test case the sliding boundary node methods obtained valid meshes for fewer deformation steps compared to the regular RBF interpolation method.

- **The inclusion of sliding boundary node methods increase the efficiency of the RBF interpolation in case data reduction methods are applied.**

The sliding methods proved to be less efficient compared to the regular RBF interpolation in the test cases where no data reduction methods were applied. This is as expected since for the pseudo sliding an additional RBF interpolation and projection is performed. The coupling of the spatial directions in the direct sliding method resulted in a larger interpolation matrix which significantly increases the computational cost. For the pseudo sliding method only a slight increase in computational cost was observed compared to the regular RBF interpolation. However, when a greedy algorithm is applied as data reduction method the sliding RBF methods proved to be more efficient compared to the regular RBF. That indicates that the error convergence of the sliding RBF methods is better than the one for the regular RBF interpolation. From the sliding methods no general best performing method could be identified as this varied from case to case.

- **Allowing the periodic boundaries to displace in a periodic manner results in an increase of minimum mesh quality and reduces the required computational effort for the regular RBF interpolations method.**

When considering the inclusion of the periodicity for a non-sliding RBF interpolation it was found that allowing a periodic movement of the boundaries improved the mesh quality significantly. For the two-dimensional square mesh the minimum mesh quality increases from 0.108 to 0.140, when going from a non-periodic regular RBF interpolation to one with periodic displacement. Due to the movement of the periodic domain a small clearance between the considered object and the periodic boundary is prevented. Therefore, the compression and skewing of the mesh elements is reduced in that region. Furthermore, the computational effort is reduced as the set of control nodes is smaller due to treatment of the periodic nodes as internal nodes. Similar improvements in terms of mesh quality and computational effort were found for the three-dimensional hexahedron mesh.

- **Using a pseudo sliding RBF interpolation in conjunction with the periodic displacement with moving vertices method provides the best performing RBF interpolation method in terms of minimum quality and computational efficiency.**

For all test cases considered the use of the periodic displacement methods with the sliding methods resulting in the highest minimum mesh quality. Of the fixed and moving vertices forms of the periodic displacement methods, the moving vertices variant produces the highest minimum mesh quality. When the vertices of the domain are allowed to slide, the periodic movement is less limited and the mesh quality is preserved to a higher degree. Therefore, when considering the sliding and the periodic displacement the RBF interpolation has the most freedom to deform, which results in a higher minimum mesh quality. Particularly, for the rotationally periodic test cases considered the moving vertices resulted in a higher minimum mesh quality. The pseudo sliding method proved to be more computational efficient compared to the direct sliding method. For the simple test cases without data reduction the direct sliding is computationally more expensive due to the larger resulting interpolation matrices. Additionally, when applying data reduction the pseudo sliding method is more efficient. For the Aachen stator test case the direct sliding required more than double the CPU time of the pseudo sliding method.

- **The double-edged single level greedy algorithm resulted in the highest error convergence rate of the RBF interpolation.**

The performance of the different forms of the greedy algorithm was evaluated for the square mesh with reduced nodal spacing creating a finer mesh. Using a double-edged selection within control node selection proved to enhance the error convergence for both single and multilevel greedy algorithm. The multilevel algorithm proved to be unsuitable for the considered test case. As the algorithm moved to a new level, peaks in the error convergence were observed. This behaviour suggests that the close proximity of multiple boundaries in internal flow problems possibly have a negative impact on the error convergence. As a result the double-edged single level greedy algorithm provided the fastest error convergence.

- **The periodic sliding boundary RBF interpolation deformation resulted in a higher minimum mesh quality compared to the linear elasticity method.**

The Aachen stator turbine test case showcased that the sliding RBF interpolation generated a higher minimum mesh quality over the entire span of the stator blade compared to the linear elasticity method. The minimum mesh quality was reduced to 27.7% of the initial minimum quality for the linear elasticity methods. While the minimum mesh quality for the pseudo and direct sliding methods was only reduced to 75.5 and 76.2% of the initial minimum quality. The pseudo and direct sliding methods provided nearly identical results in terms of mesh quality. However, the pseudo sliding methods proved to be more efficient than the direct sliding method. The CPU time of the pseudo sliding method was 306.8 seconds and the direct sliding method required a CPU time of 767.8 seconds for the same deformation.

- **The impact of the sliding RBF interpolation on the flow solver proved to be very limited.**

The values for the total pressure loss and entropy generation only had very small difference between the linear elasticity method and the sliding RBF methods. Furthermore, the Mach distribution and pressure distribution of the stator blade did not indicate a significant change between the mesh deformation methods. Moreover, the convergence of the lift and drag coefficients seemed nearly identical for the linear elasticity method and both sliding methods. Thus, the improved region of the mesh quality at the periodic boundary did not result in any significant changes in the flow solution or convergence of the fluid solver.

8 Recommendations

This section discusses a few recommendations that could be considered as further steps in future works.

- **The influence of the sliding RBF interpolation should be investigated on different types of boundary geometries.**

In order to gain more insight in the effectiveness of the sliding methods other boundary geometries should be considered. In this project domains with predominantly straight boundary edges and planar boundary surfaces are considered as simple test cases. It would be of interest to evaluate the performance of the sliding boundary node methods in conjunction with the periodic measures for domains containing boundaries with (large) curvature.

- **Further investigation is required on the applicability of the multilevel greedy algorithm on internal flow domains.**

Furthermore, the multilevel greedy algorithm was proven in literature to be more efficient for external flow applications, but did not improve the error convergence rate for the considered test case in this research. Therefore, additional investigation in this area is required to find out why this is exactly the case. Perhaps some adaptations can be made in order to make it more suitable for internal flow problems.

- **The influence of adding more control nodes per greedy cycle on the computational efficiency of the RBF interpolation should be studied.**

Moreover, for each greedy cycle a single or two nodes were selected before performing a new RBF interpolation and determining the resulting error. A study could be performed on the influence of adding more control nodes per cycle. This could make the greedy algorithms more effective as less RBF interpolations have to be performed as the set of control nodes grows.

- **The RBF-SliDe tool should be made suitable for algorithmic differentiation in order to be employed within an adjoint-based optimisation framework.**

Now that a sliding RBF interpolation tool is developed. A further step to an implementation within an adjoint-based optimisation framework would be to make the deformation method suitable for algorithmic differentiation, such that it can function in an adjoint-based optimisation.

- **The influence of using the periodic sliding RBF interpolation method in an aerodynamic design optimisation should be evaluated.**

To evaluate the effects of the RBF interpolation with sliding boundary nodes and the inclusion of periodicity on the aerodynamic design optimisation, an optimisation should be performed within the framework of SU2. The resulting optimised design can be compared to one obtained by using the default SU2 mesh deformation method based on linear elastic equations. This will give an understanding on the influence of the RBF interpolation as mesh deformation method in the optimisation process.

References

[1] A. Jameson. “Aerodynamic Design Through Control Theory”. In: *Journal of Scientific Computing* 3 (1988), pp. 233–260.

[2] Z. Li et al. “Review of design optimization methods for turbomachinery aerodynamics”. In: *Progress in Aerospace Sciences* 93 (2017), pp. 1–23.

[3] T.D. Economon et al. “SU2: An Open-Source Suite for Multiphysics Simulation and Design”. In: *AIAA Journal* 54, No.3 (2015), pp. 828–846.

[4] T.A. Albring et al. “Efficient Aerodynamic Design Using the Discrete Adjoint Method in SU2”. In: *AIAA 2016-3518* (). 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2016.

[5] M. Sagebaum et al. “Expression Templates for Primal Value Taping in the Reverse Mode of Algorithmic Differentiation”. In: *Optimization Methods and Software* 33(4-6) (2018), pp. 1–25.

[6] S. Vitale et al. “Fully Turbulent Discrete Adjoint Solver for Non-Ideal Compressible Flow Applications”. In: *Journal of the Global Power and Propulsion Society* 1 (2017), pp. 252–270.

[7] A. Rubino et al. “Adjoint-Based Fluid Dynamic Design Optimization in Quasi-Periodic Unsteady Flow Problems Using a Harmonic Balance Method”. In: *Journal of Computational Physics* 372 (2018), pp. 220–235.

[8] S. Vitale et al. “Multistage Turbomachinery Design Using the Discrete Adjoint Method Within the Open-Source Software SU2”. In: *Journal of Propulsion and Power* 36 (2020), pp. 465–478.

[9] R. Vello. *A Validation Infrastructure for Non-Ideal Compressible Fluid Dynamics: with applications to ORC Turbines*. Tech. rep. University of Technology Delft, 2021.

[10] K. Johri. *Data-driven boundary layer loss model for organic rankine cycle turbomachines*. Tech. rep. University of Technology Delft, 2020.

[11] L. Bills. *Validation of the SU2 Flow Solver for Classical Non Ideal Compressible Fluid Dynamics*. Tech. rep. University of Technology Delft, 2020.

[12] B.S. Sanghera. *Adjoint Shape Optimisation of Rocket Engine Turbine Blades*. Tech. rep. University of Technology Delft, 2020.

[13] P. Garrido de la Serna. *Adjoint-based 3D Shape Optimization for Turbomachinery Applications*. Tech. rep. Delft University of Technology, 2019.

[14] I. Castro de Castro. *Assessment of SU2 for radial compressor performance prediction*. Tech. rep. University of Technology Delft, 2019.

[15] P.P. Natarajan. *Blade shape optimization of an axial turbine using the adjoint method*. Tech. rep. University of Technology Delft, 2018.

[16] J.T. Batina. “Unsteady Euler airfoil solutions using unstructured dynamic meshes”. In: *AIAA Journal* 28(8) (1990), pp. 1381–1390.

[17] C. Farhat et al. “Torsional springs for two-dimensional dynamic unstructered meshes”. In: *Computer Methods in Applied Mechanics Engineering* 163 (1998), pp. 231–245.

[18] C. Degand et al. “A three-dimensional torsional spring analogy method for unstructured dynamic meshes”. In: *Computers and Structures* 80 (2002), pp. 305–316.

[19] F.J. Blom. “Considerations on the spring analogy”. In: *International Journal for Numerical Methods in Fluids* 32 (2000), pp. 647–668.

[20] D. Zeng et al. “A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains”. In: *Finite Elements in Analysis and Design* 41(11-12) (2005), pp. 1118–1139.

[21] C.L. Bottasso et al. “The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes”. In: *Computer Methods in Applied Mechanics and Engineering* 194(39-41) (2005), pp. 4244–4264.

[22] G.A. Markou et al. “The ortho-semi-torsional (OST) spring analogy method for 3D mesh moving boundary problems”. In: *Computer Methods in Applied Mechanics and Engineering* 196(4-6) (2007), pp. 747–765.

[23] R. P. Dwight. “Robust Mesh Deformation using the Linear Elasticity Equations”. In: *Computational Fluid Dynamics 2006*. Ed. by H. Deconinck et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 401–406.

[24] R. Löhner et al. “Improved ALE mesh velocities for moving boundaries”. In: *Communications in Numerical Methods in Engineering* 12 (1996), pp. 599–608.

[25] B.T. Helenbrook. “Mesh deformation using the biharmonic operator”. In: *Iinternational Journal for Numerical Methods in Engineering* 56 (2003), pp. 1007–1021.

[26] X. Liu et al. “Fast dynamic grid deformation based on Delaunay graph mapping”. In: *Journal of Computational Physics* 211(2) (2006), pp. 405–423.

[27] A. de boer et al. “Mesh deformation based on radial basis function interpolation”. In: *Computers and Structures* 85 (2007), pp. 784–795.

[28] T.C.S. Rendall et al. “Efficient mesh motion using radial basis functions with data reduction algorithms”. In: *Journal of Computational Physics* 228(17) (2009), pp. 6231–6249.

[29] J.A.S. Witteveen. “Explicit and Robust Inverse Distance Weighting Mesh Deformation for CFD”. In: *8th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. 2010.

[30] M. Morelli et al. “Efficient radial basis function mesh deformation methods for aircraft icing”. In: *Journal of Computational and Applied Mathematics* 392 (2021), p. 113492.

[31] L. Abergo et al. “Aerodynamic shape optimization based on discrete adjoint and RBF”. In: *Journal of Computational Physics* 477 (2023), p. 111951.

[32] S. Aubert et al. “Planar Slip Condition For Mesh Morphing Using Radial Basis Functions”. In: *Procedia Engineering* 203 (2017), pp. 349–361.

[33] M.T. Mathew. *Mesh Deformation Using Radial Basis Function Interpolation With Sliding Boundary Nodes*. Tech. rep. Delft University of Technolog, 2019.

[34] J. de Keyser. *Mesh deformation using radial basis function interpolation on periodic domains with small clearance gaps*. Tech. rep. Delft University of Technology, 2021.

[35] M.D. Buhmann. “Radial basis functions”. In: *Acta Numerica* 9 (2000), pp. 1–38.

[36] H. Wendland. *Konstruktion und untersuchung radialer basisfunktionen mit kompaktem träger*. Tech. rep. 1996.

[37] Y. Rozenberg et al. “Fluid Structure Interaction Problems in Turbomachinery Using RBF Interpolation and Greedy Algorithm”. In: *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*. 2014.

[38] T.C.S. Rendall et al. “Parallel Efficient Mesh Motion using Radial Basis Functions with Application to Multi-bladed Rotors”. In: *Int. J. Numer. Meth. Engng.* 81 (2010), pp. 89–105.

[39] T. Gillebaart et al. “Adaptive radial basis function mesh deformation using data reduction”. In: *Journal of Computational Physics* 321 (2016), pp. 997–1025.

[40] G. Wang et al. “Improved Point Selection Method for Hybrid-Unstructured Mesh Deformation Using Radial Basis Functions”. In: *AIAA Journal* 53(4) (2015), pp. 1016–1025.

[41] J.L. Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Communications of the ACM* 18(9) (1975), pp. 509–517.

[42] Jose Luis Blanco et al. *nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees*. `https://github.com/jlblancoc/nanoflann`. 2014.

[43] Mm Muja et al. “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”. In: *International Conference on Computer Vision Theory and Applications (VISAPP’09)* (2009).

[44] P.M. Knupp. “Algebraic mesh quality metrics for unstructured initial meshes”. In: *Finite Elements in Analysis and Design* 39 (2003), pp. 217–241.

[45] Gaël Guennebaud et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[46] F. Menter. “Zonal Two Equation k–ω Turbulence Models for Aerodynamic Flows”. In: *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference* AIAA Paper 93.2906 (1993).

A Frame Transformations

When the considered domain is rotationally periodic, coordinate transformations are required between the Cartesian and polar or cylindrical coordinate systems. The transformations of coordinates and Euclidean distances are summarised in this appendix.

A.1. Coordinates

Cartesian to polar

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \text{arctan}\left(\frac{y}{x}\right) \end{aligned}$$

Cartesian to cylindrical

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \text{arctan}\left(\frac{y}{x}\right) \\ z &= z \end{aligned}$$

Polar to Cartesian

$$\begin{aligned} x &= r\,\text{cos}(\theta) \\ y &= r\,\text{sin}(\theta) \end{aligned}$$

Cylindrical to Cartesian

$$\begin{aligned} x &= r\,\text{cos}(\theta) \\ y &= r\,\text{sin}(\theta) \\ z &= z \end{aligned}$$

A.2. Euclidean Distance

From Cartesian to polar

The Euclidean distance in terms of polar coordinates is found by substituting the expressions of x and y of the polar coordinate system into the formula for Euclidean distance as shown in Equation A.1.

$$\begin{aligned}\delta &= \sqrt{d_x^2 + d_y^2} \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ &= \sqrt{(r_2 \cos\theta_2 - r_1 \cos\theta_1)^2 + (r_2 \sin\theta_2 - r_1 \sin\theta_1)^2} \\ &= \sqrt{r_2^2 \cos^2\theta_2 + r_1^2 \cos^2\theta_1 - 2r_2r_1 \cos\theta_2 \cos\theta_1 + r_2^2 \sin^2\theta_2 + r_1^2 \sin^2\theta_1 - 2r_2r_1 \sin\theta_2 \sin\theta_1} \\ &= \sqrt{r_1^2(\cos^2\theta_1 + \sin^2\theta_1) + r_2^2(\cos^2\theta_2 + \sin^2\theta_2) - 2r_1r_2(\cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2)} \\ &= \sqrt{r_1^2 + r_2^2 - 2r_1r_2(\cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2)} \qquad \text{With: } \cos^2\theta + \sin^2\theta = 1 \\ &= \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_2 - \theta_1)} \qquad \text{With: } \cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta\end{aligned} \tag{A.1}$$

From Cartesian to cylindrical

The Euclidean distance within the cylindrical coordinate system is found by following the same derivation as performed in Equation A.1 with the addition of a term in the z-direction:

$$\begin{aligned}\delta &= \sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2} \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)} \\ &= \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_2 - \theta_1) + (z_2 - z_1)^2}\end{aligned} \tag{A.2}$$

B Exemplary RBF-SliDe Files

B.1. Configuration File

```
% Example configuration file for a square 25x25 mesh for the RBF-SliDe tool


% ----------------------- INPUT/ OUTPUT FILENAMES -----------------------
% Input mesh filename
MESH_FILENAME = 25x25.su2

% Output mesh filename
MESH_OUT_FILENAME = 25x25_def.su2

% Filename of the file containing the prescribed nonzero deformation
DEFORMATION_FILENAME = 25x25_deformation.txt

% Generate quality file of output mesh file (NO, YES)
GENERATE_QUALITY = YES

% ----------------------- MESH PARAMETERS -----------------------

% Markers denoting the domain boundaries
MARKER_BDRY = (LEFT, RIGHT, UPPER, LOWER, BLOCK)

% Marker of the boundaries with the prescribed nonzero deformations
MARKER_MOVING = (BLOCK)

% Marker of boundaries that are treated as internal nodes
MARKER_INTERNAL = ( )

% Type of periodicity. Translational = 0, rotational = 1
PERIODIC_TYPE = 0

% Direction of periodicity.
% Translational: x = 0, y = 1, z = 2, or Rotational: r = 0, theta = 1, z = 2
PERIODIC_DIRECTION = 1

% Apply curvature correction (NO, YES)
CURVED = NO

% ----------------------- RBF INTERPOLATION PARAMETERS -----------------------

% Method of sliding
% none = no sliding, ps = pseudo sliding, ds = direct sliding
SLIDING_MODE = ps

% Method of applying periodicity (none, periodic, fixed, moving)
% none = non-periodic, periodic = periodic,
% fixed = periodic displacement with fixed vertices,
% moving = periodic displacement with moving vertices
PERIODIC_MODE = none

% Number of steps in which the deformation is applied
STEPS = 1

% RBF support radius = INFLUENCE_FACTOR * largest domain length
INFLUENCE_FACTOR = 2.5


% ----------------------- DATA REDUCTION PARAMETERS -----------------------

% Apply data reduction methods (NO, YES)
DATA_REDUCTION = YES

% Data reduction tolerance
DATA_RED_TOLERANCE = 1e-3

% Apply multilevel greedy algorithm (NO, YES).
% only in case DATA_REDUCTION = YES
MULTILEVEL = NO

% Multilevel greedy criterium (size, tol)
% Proceed to next level when fixed level size, or a given error reduction
% factor is reached
MULTI_CRIT = size

% multilevel greedy tolerance criterium
% Error reduction factor
LVL_TOL_CRIT = 0.5

% Multilevel greedy size criterium
% Level size
LVL_SIZE = 16

$ Apply double edge control node selection (NO, Yes)
DOUBLE_EDGE = YES

$ Greedy correction factor, Correction radius = GAMMA * max error
GAMMA = 25
```

B.2. 5x5 Square SU2 Mesh file

```
%
% Problem dimension
NDIME= 2
%
% Number of elements
NELEM= 24
%
% Connectivity inner elements
% First integer is the element identifier following the VTK format
% Last integer is the element number
% Remaining integers are the vertices of the element and correspond to the indices
% of the grid points
%
9 0 1 7 6 0
9 1 2 8 7 1
9 2 3 9 8 2
9 3 4 10 9 3
9 4 5 11 10 4
9 6 7 13 12 5
9 7 8 14 13 6
9 8 9 15 14 7
9 9 10 16 15 8
9 10 11 17 16 9
9 12 13 19 18 10
9 13 14 20 19 11
9 15 16 22 21 13
9 16 17 23 22 14
9 18 19 25 24 15
9 19 20 26 25 16
9 20 21 27 26 17
9 21 22 28 27 18
9 22 23 29 28 19
9 24 25 31 30 20
9 25 26 32 31 21
9 26 27 33 32 22
9 27 28 34 33 23
9 28 29 35 34 24
%
% Number of grid points
NPOIN= 36
%
% Node coordinates
% Last integer is coordinate number
0 0 0
0.2 0 1
0.4 0 2
0.6 0 3
0.8 0 4
1 0 5
0 0.2 6
0.2 0.2 7
0.4 0.2 8
0.6 0.2 9
0.8 0.2 10
1 0.2 11
```

```
0 0.4 12
0.2 0.4 13
0.4 0.4 14
0.6 0.4 15
0.8 0.4 16
1 0.4 17
0 0.6 18
0.2 0.6 19
0.4 0.6 20
0.6 0.6 21
0.8 0.6 22
1 0.6 23
0 0.8 24
0.2 0.8 25
0.4 0.8 26
0.6 0.8 27
0.8 0.8 28
1 0.8 29
0 1 30
0.2 1 31
0.4 1 32
0.6 1 33
0.8 1 34
1 1 35
%
% Number of boundaries
NMARK= 5
%
% Boundary tag name
MARKER_TAG= lower
%
% Number of elements in the boundary
MARKER_ELEMS= 5
%
% Boundary element connectivity follows same format as inner element connectivity
3 0 1
3 1 2
3 2 3
3 3 4
3 4 5
MARKER_TAG= right
MARKER_ELEMS= 5
3 5 11
3 11 17
3 17 23
3 23 29
3 29 35
MARKER_TAG= upper
MARKER_ELEMS= 5
3 35 34
3 34 33
3 33 32
3 32 31
3 31 30
MARKER_TAG= left
MARKER_ELEMS= 5
```

```
3 30 24
3 24 18
3 18 12
3 12 6
3 6 0
MARKER_TAG= block
MARKER_ELEMS= 4
3 14 15
3 15 21
3 21 20
3 20 14
```