Master of Science Thesis

Inverse Distance Weighting Mesh Deformation

A Robust and Efficient Method for Unstructured Meshes

Laura Uyttersprot

February 12, 2014

Faculty of Aerospace Engineering | NUMECA International | Delft University of Technology

Inverse Distance Weighting Mesh Deformation

A Robust and Efficient Method for Unstructured Meshes

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Laura Uyttersprot

February 12, 2014

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"Inverse Distance Weighting Mesh Deformation"** by **Laura Uyttersprot** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: February 12, 2014

Supervisors:

_____
Reader 1

_____
Reader 2

_____
Reader 3

_____
Reader 4

Preface

This thesis marks the end of a very challenging but enriching time as an aerospace engineering student in Delft. It has been an exciting journey which has taken me across five different countries in three different continents. However I could never have accomplished it without the support of many others.

First of all I would like to thank my supervisors Sander van Zuijlen and Benoit Leonard for creating the opportunity for me to do this interesting thesis project at NUMECA international. In particular a huge thank you goes out to both Sander van Zuijlen and Liesbeth Florentie for many hours of phone call meetings, providing me with valuable insights and feedback. Their dedication and enthusiasm has made it possible for me to perform at the best of my abilities.

I would also like to express my gratitude to Mohamed Mezine, for always making time to answer my many questions, and Sergio Gonzalez Horcas for the many brainstorming sessions and for welcoming me so warmly in the mesh deformation team at NUMECA. I would also like to thank all my other colleagues at NUMECA for the many breaks, lunches and evenings that we spent together. Without them this experience would not have been the same.

To all my family and friends, in particular my parents and Anthony, many thanks for your support.

Laura Uyttersprot
Brussels, February 2014

Summary

The computational modelling of fluid-structure interaction phenomena is currently gaining importance thanks to engineering design trends towards lighter and more flexible structures. FSI simulations are important because they can predict potentially dangerous interaction instabilities and the performance of designs in their deformed shapes. However, the cost of such FSI simulations is currently still very high due to the fact that two coupled solvers have to be applied. Moreover, interaction problems usually require a large amount of small time steps to ensure computational stability.

One of the expensive parts of the FSI simulation is the deformation of the fluid mesh. The fluid mesh has to be deformed in order to be conformal to the displaced structure. During an FSI simulation large displacements can occur, which leads to the requirement that a very robust mesh deformation method is needed. On the other hand, the method has to be highly efficient since it has to be executed at every time step. Therefore the goal of this thesis project is to design, implement and test a robust, efficient and user-friendly mesh deformation algorithm for application to a wide variety of industrial FSI problems using unstructured polyhedral meshes.

A literature review revealed that many mesh deformation methods have been developed already. These can be divided in three main groups: mesh-connectivity based schemes, point-by-point schemes and hybrid schemes. However, so far none of the methods has come forward as the best technique to be applied in FSI, as they all have their specific drawbacks. One of the most recently developed methods did stand out thanks to its high robustness in combination with a low computational cost and complexity. This method is the inverse distance weighting (IDW) interpolation method. IDW is an explicit interpolation technique, which computes the interpolation function as a weighted average of the known boundary node displacements. The weights are inversely proportional to the distances between the inner mesh nodes and the boundary nodes. Thanks to its explicit nature, it does not require the solution of a system, which is one of the main drawbacks of the robust radial basis function (RBF) interpolation method. The basic IDW method has been investigated and improved with several additions, in order to arrive at a promising mesh deformation method, which can be applied in a commercial FSI software package.

Firstly, the effect of including the boundary node rotations has been investigated. To this end, the displacement of each boundary node is split into a translation and rotation part.

The rotation part is represented with quaternions, which are hyper-complex numbers that represent a rotation axis and a rotation angle. Four ways to interpolate the quaternions into the volume mesh have been identified: spherical linear interpolation (SLERP) of quaternions, linear interpolation (LERP) of quaternions, linear interpolation of the logarithm of quaternions and interpolation of the displacement due to applying the rotation. Of these methods the last one appeared to be most robust for general deformation cases. On the other hand, LERP of quaternions is the preferred method when the deformation consists of a large rotation. By including the rotations in the IDW method, the orthogonality of the deformed mesh is greatly improved in the near boundary region.

Secondly, a new method to include sliding boundary nodes has been implemented. A sliding boundary condition allows the nodes to slide, while remaining on the original surface. The implemented method consists of several steps. First the nodes on the sliding edges are interpolated as if they were inner mesh nodes. This means that they might not have remained on their respective edges. Therefore the nodes are moved to the closest point on their original edge, which is called snapping. This process is then repeated for the nodes on sliding faces. Once the new position for all sliding boundary nodes is known they become data points in the interpolation to the volume mesh nodes. As such the sliding boundary nodes are actually treated as a mix between inner and boundary nodes.

The adapted method is found to be faster than RBF, however, it still becomes costly for large-scale meshes. For example, a mesh with approximately 3 million nodes, of which 130,000 are boundary nodes, has a deformation time of 2 hours in serial mode. Therefore several efficiency improvement methods have been investigated. Thanks to the proven efficiency of boundary node coarsening for the RBF method, this method has been applied in this thesis. The boundary node coarsening is done with a greedy algorithm, which starts from a small initial set of boundary nodes and adds the node with the largest surface displacement interpolation error to the the list of selected nodes, at each iteration. The nodes which are selected by the greedy algorithm will become data points for the interpolation, whereas for the other nodes the IDW interpolation error is computed. This interpolation error is corrected for by moving the non-selected boundary nodes to their actual positions in a local secondary mesh deformation step, which uses a nearest neighbour RBF correction. By applying boundary node coarsening, the same test problem can be done in six minutes, while the mesh quality is maintained at the same level after the deformation. If a small reduction in mesh quality is acceptable, it is even possible to reduce the deformation time further to only 37 s, which is negligible compared to the time to compute the CFD results. This means that boundary node coarsening can reduce the computation time by a factor of nearly 200.

Finally, the IDW method is compared to the radial basis function and elastic analogy mesh deformation (EAMD) method, for five different test problems. For all these tests the IDW method results in a higher or equal minimum mesh quality than the other two methods. Especially for test cases with large rotations the IDW method has proven to be superior. It has also been shown that for several test cases, such as a rotor within a turbo-machinery casing, sliding nodes are essential to obtain good results. Therefore, the RBF method, which currently does not have this functionality, cannot compete with IDW in terms of mesh quality for such problems. Moreover the IDW method also maintains a higher mesh quality in the near-boundary mesh layers, compared to both RBF and EAMD. This is particularly important

to ensure a high accuracy CFD solution. In terms of computation time, the IDW method (without coarsening) has also proven to be significantly more efficient than both RBF and EAMD. When comparing the time required to evaluate the interpolation function for all the volume mesh nodes, IDW is up to three times faster than RBF for 3D test cases. Additionally the time to solve the RBF system at the start of the simulation, which can take hours to days, is not present in IDW mesh deformation. Furthermore the IDW method requires significantly less memory than RBF for which the matrix inversion quickly requires several gigabytes. Also compared to EAMD the IDW method has proven to be at least 7 times faster for all test problems.

In conclusion it can be stated that a versatile, robust and efficient IDW mesh deformation method has been developed, which can easily be applied to any type of mesh, for any type of FSI problem, with a large amount of cells and time steps.

Table of Contents

List of Figures

List of Tables

Nomenclature

Greek Symbols

ν	Poisson's ratio
$\boldsymbol{\epsilon}$	Strain tensor
$\boldsymbol{\sigma}$	Stress tensor
Γ_I	Boundary between fluid and structure domain
λ	First Lamé constant
μ	Second Lamé constant or shear modulus
Ω	Computational domain
Φ_{AR}	Aspect ratio quality metric of a mesh cell
Φ_{ortho}	Orthogonality quality metric of a mesh cell
Φ_{skew}	Equi-angular skewness quality metric of a mesh cell

Latin Symbols

n_b	Number of boundary mesh nodes
n_i	Number of inner mesh nodes
$\boldsymbol{u}$	Displacement vector
$\boldsymbol{n}$	Normal vector
E	Young's modulus or elastic modulus
p	Pressure

Abbreviations

ALE	Arbitrary Lagrangian-Eulerian
CFD	Computational Fluid Dynamics
CSM	Computational Structure Mechanics

DGM	Delaunay Graph Mapping
EAMD	Elastic Analogy Mesh Deformation
FMLE	Fast Multi-Level Evaluation
FMM	Fast Multipole Method
FSI	Fluid-Structure Interaction
IDW	Inverse Distance Weighting
LERP	Linear IntERPolation
MpCCI	Mesh-based Parallel Code Coupling Interface
MSA	Moving Submesh Approach
QP	Quad points, i.e. four pseudo nodes
RBF	Radial Basis Function
SLERP	Spherical Linear Interpolation
TPS	Thin Plate Spline

Chapter 1

Introduction

1.1 Motivation

Fluid-structure interaction (FSI) is a multidisciplinary field that comprises all phenomena where a flow and a structure interact with each other. The computational modelling of such phenomena is currently gaining importance due to engineering design trends towards lighter and more flexible structures. Examples of this can be found in modern wind turbine [67], aircraft [21] and bridge designs [22]. Furthermore also medical research requires modelling of fluid-structure interaction phenoma such as arterial blood flows [5]. Simulations of fluid-structure phenomena are important because they can predict the potentially dangerous instability of coupled systems. Furthermore simulations allow to predict the performance of designs in their deformed shapes.

Thanks to the significant increase of computation power over the past couple of decades, complex fluid-structure interaction simulations have now become feasible. Computational fluid-structure interaction combines computational fluid dynamics (CFD) and computational structure mechanics (CSM) by means of an interaction interface where the forces and displacements are coupled. The dynamics of the flow results into forces on the structure, which causes the structure to move or deform. This displacement of the structure in turn causes the fluid domain to move. Besides the coupling of the two solvers another concern in FSI is the deformation of the fluid mesh to conform to the deformation of the flexible structure as illustrated in figure 1.1. In some cases the structure is so flexible that very large deformations can occur. Therefore an automatic mesh deformation algorithm that can handle large deformations is a must. The development of such a mesh deformation algorithm is the main topic of this research project.

The current research project takes place in a collaboration framework between the Faculty of Aerospace Engineering at the Delft University of Technology and the CFD software developer NUMECA International. NUMECA International has developed an FSI software package,

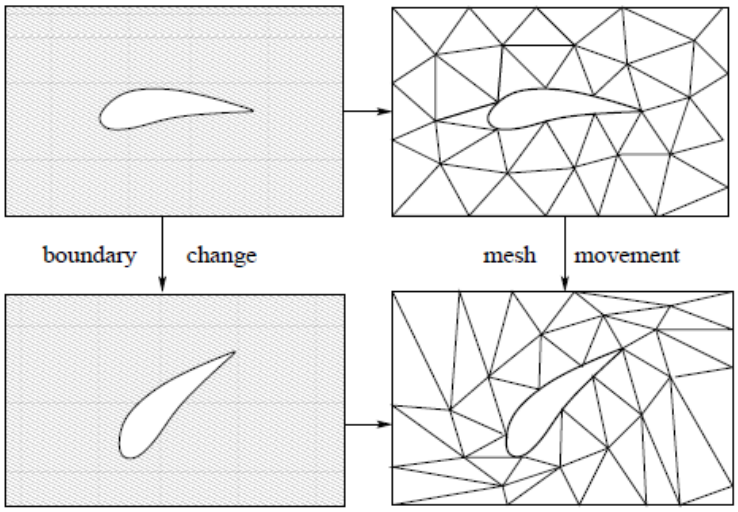


Figure 1.1: Schematic representation of the mesh deformation due to the deformation of the structure [7].

together with Open Engineering. The software package combines NUMECA International's CFD solver FINE^{TM}/Open [47] with Open Engineering's CSM solver Oofelie [51] and is therefore called FINE^{TM}/FSI-OOFELIE. The main goal of the current project is to design, implement and test a mesh deformation method within the context of this software package. The meshes in the fluid domain are unstructured hexahedral meshes generated with NUMECA International's mesh generator HEXPRESS^{TM} [48].

1.2 Background

This section provides the reader with a general background of computational fluid-structure interaction. Several of the main issues in fluid-structure interaction, such as temporal and spatial coupling and mesh deformation, are discussed here.

In general an FSI problem can be divided in a fluid domain Ω_f, a structure domain Ω_s and a boundary Γ_I which forms an interface between the two domains. At this interface the forces and displacements should match, which can be expressed by the kinematic and a dynamic boundary conditions

$$\boldsymbol{u}_f = \boldsymbol{u}_s \quad \text{on } \Gamma_I, \tag{1.1}$$

$$p_s \boldsymbol{n}_s = p_f \boldsymbol{n}_f \quad \text{on } \Gamma_I, \tag{1.2}$$

where the subscripts f and s denote the fluid and the structure respectively, $\boldsymbol{u}$ is the displacement vector, p is the pressure and $\boldsymbol{n}$ is the outward pointing normal.

When simulating an FSI problem there are two main options. The first option is to develop a dedicated solver which solves the fluid and structure problem in one monolithic framework.

The second is to make use of separate fluid and structure solvers which are coupled via a coupling shell. Due to the fact that monolithic solvers are problem specific, not much research has been dedicated to this subject. On the other hand many dedicated CFD and CSM solvers have been developed, which can be used as part of partitioned FSI solvers. One of the main difficulties with partitioned solvers is the temporal coupling. If the temporal coupling is not done properly the accuracy of the simulation will decrease significantly.

Temporal coupling can either be done with a loosely coupled or a strongly coupled algorithm. In a loosely coupled scheme the solution of the flow and the structure is computed only once per time step. Such a scheme introduces a partitioning error due to the time lag between the two solvers. In strongly coupled schemes sub-iterations are used for each time step such that the flow and structure boundary conditions are converging before going to the next time step. For FSI problems where the structure deforms significantly, strongly coupled schemes are advised.

Besides temporal coupling between the solvers, also spatial coupling is needed. The forces have to be transferred from the fluid to the structure and the displacements have to be transferred from the structure to the fluid. This would be an easy task if the fluid and structure mesh would match up at the boundary. However this is usually not the case as the flow simulation requires a much finer mesh than the structure simulation. Therefore interpolation methods have to be used to transfer the variables across the boundaries. Several transfer algorithms are described in the work of de Boer [13].

Fluid equations are usually discretised using the Eulerian framework, meaning that the fluid particles are moving through the mesh and the equations are expressed in spatial coordinates. The structure on the other hand is usually discretised using the Lagrangian framework where the mesh is fixed to the material points and hence material coordinates are used. In fluid-structure interaction the domain of the fluid changes due to the deformation of the structure. Therefore it is inconvenient to have a fixed fluid mesh. This is solved by using the Arbitrary Lagrangian-Eulerian (ALE) formulation [16], which can also be referred to as the dynamic mesh formulation. In the ALE formulation the fluid mesh moves according to the boundary change, therefore it is neither Eulerian (fixed) nor Lagrangian (moving with material particles), but somewhere in between.

Due to the use of the ALE method, the fluid mesh has to move conforming to the displacement of the structure. One might first think that the mesh generator can be used at each time step to generate a new mesh. However this is a complex, time consuming task which also leads to interpolation errors from the old to the new mesh [28]. The second option is to deform the mesh based on structural analogies or interpolation methods. A drawback of mesh deformation is that usually the mesh quality deteriorates in time, which affects the accuracy of the simulation. Furthermore the fact that the mesh deformation has to be executed at each time step requires high efficiency.

1.3 Research Question, Aims and Objectives

This section gives a detailed overwiew of the direction and goals of the research project. First, the main research question in this project can be stated as

> *Which mesh deformation method is the optimal choice in terms of robustness, efficiency and user-friendliness for implementation in commercial FSI software applied to a wide variety of problems using unstructured polyhedral meshes?*

The goal of the project is to implement a mesh deformation module into the commercial FSI software FINE^{TM}/FSI-Oofelie, developed by a cooperation between NUMECA International and Open-Engineering. This mesh deformation method has to fulfil four main requirements, in order of importance:

- *The mesh deformation method has to be robust.* This means that the quality of the mesh has to remain as high as possible after several consecutive deformations. Even for very large deformations the mesh deformation should generate meshes that are valid, i.e. no negative cells are present. This is very important because the simulation has to be aborted in case negative cells are present. Furthermore the quality of the mesh is important to maintain the computational accuracy of the simulation [68].

- *The mesh deformation method has to be efficient.* Due to the fact that the mesh deformation has to be executed at each time step of the simulation, it is important that the computational cost remains low. Especially in the case of very large industrial meshes, containing millions of cells, the cost of mesh deformation can become prohibitively high. Therefore the method should scale well for large meshes and has to be applicable in parallel simulations.

- *The mesh deformation method has to be user-friendly.* This requirement is important because the method is implemented in commercial software. The customers have to be able to use the mesh deformation module without much difficulty or user input. For example the use of complex input parameters that need tweaking should be avoided.

- *The implementation of the mesh deformation method should be as simple and concise as possible.* This is particularly important for commercial software, because the software is under continuous development by a large team of people. Therefore a simple implementation will limit code maintenance and debugging cost. This requirement is however inferior to the previous requirements, as a complex robust method will be preferred over a simple method which cannot do large deformations. However the complexity of the method can drive the decision between two methods that otherwise perform equally well.

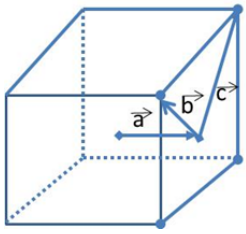


Figure 1.2: Computation of volume of hexadron. [50]

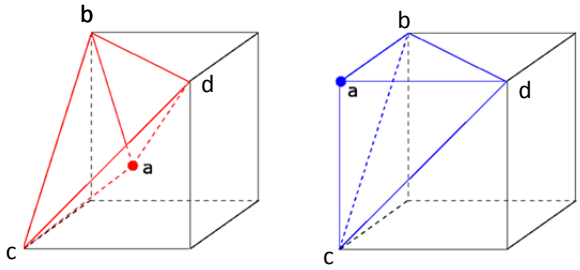


Figure 1.3: The left cell is concave because the volume of the tetrahedron $abcd$ is negative. The right cell is convex. [50]

1.4 Mesh Quality Measures

The properties of the mesh have an important influence on the accuracy of the solution. In order to assess whether the quality of the mesh after a deformation is sufficient there is a need for mesh quality measures. Ultimately, one would assess the quality of the mesh by computing the error in the solution. However this would be time consuming and also requires knowledge of the exact solution, which is often not at hand. Therefore mesh quality measures, that are based on the geometric properties of the mesh, have been introduced in HEXPRESS^{TM} [50].

The minimum quality requirement for any mesh is that no inverted cells should be present. A cell is inverted when the signed volume is negative, therefore inverted cells are also called *negative cells*. The volume of a hexahedron is computed by splitting it up into 24 tetrahedra as illustrated in figure 1.2. The equation for the signed volume V_{hexa} is then given by

$$V_{hexa} = \frac{\sum_1^6 \sum_1^4 [-\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c})]}{6} \tag{1.3}$$

A second form of a degenerate cell is the concave cell, as shown in the left of figure 1.3. To check whether a cell is concave, a tetrahedron is constructed for each vertex as illustrated for vertex a in figure 1.3. If at least one of these tetrahedra has a negative volume, the cell is concave.

The last form of a degenerate cell is a twisted cell. This is a cell where a face is rotated in its

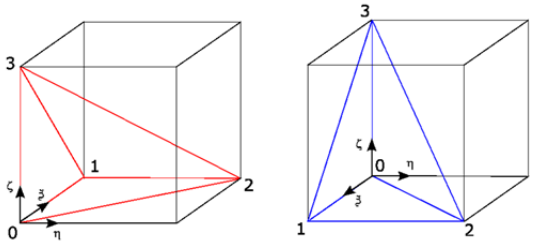


Figure 1.4: Twisted cell. [50]

own plane by an angle of 180° or more. This can be further explained by means of figure 1.4 where either the red or the blue tetrahedron has a negative volume.

The occurrence of degenerate cells should always be avoided when possible. In case negative cells are generated the simulation is automatically stopped. For twisted or concave cells the simulation will continue, however their occurrence might still cause stability or robustness issues for the solver. Furthermore concave and twisted cells can result into negative cells after cell subdivision, for example during refinement of the mesh.

Besides the occurrence of degenerate cells the quality of the mesh can also be assessed with metrics that result in a value ranging from good to bad. The first mesh quality metric is orthogonality. The cell orthogonality is computed by means of the angle between three vectors. These three vectors connect the centres of opposite faces of the cell. After normalisation of these vectors, for each vector the angle between the vector and the plane formed by the other two vectors is computed as

$$\Gamma_{ijk} = \frac{\pi}{2} - \cos^{-1}\left(\boldsymbol{h}_i \cdot (\boldsymbol{h}_j \times \boldsymbol{h}_k)\right). \tag{1.4}$$

The orthogonality metric Φ_{ortho} is then computed by taking the minimum value of these angles in degrees

$$\Phi_{ortho} = \min\left(\Gamma_{ijk}\frac{180}{\pi}\right). \tag{1.5}$$

It can be seen that an orthogonality measure of 90° indicates a perfectly orthogonal cell, whereas a value equal to or below 0° means the cell is degenerate. In general it is advised that the minimum orthogonality should be higher than 5° [49].

A second important metric is the equi-angular skewness of a cell. This is calculated by computing the 12 dihedral angles θ corresponding to the 12 edges of a cell. After finding the maximum θ_{max} and minimum θ_{min} dihedral angles of the cell, the equi-angular skewness Φ_{skew} of the cell can be computed as follows

$$\Phi_{skew} = max\left(\frac{\theta_{max} - 90}{90}, \frac{90 - \theta_{min}}{90}\right) \tag{1.6}$$

A skewness of 0 indicates a perfect cell, and 1 indicates that the cell is degenerate.

The last mesh quality metric discussed in this section is the aspect ratio criterion. The aspect ratio of a hexahedral cell is given by the ratio of longest to the shortest edge.

$$\Phi_{AR} = \frac{h_{max}}{h_{min}} \tag{1.7}$$

Usually a mesh will have large aspect ratio in viscous boundary layers, and low aspect ratio cells in the far field.

1.5 Outline

The outline of the remainder of this thesis report is as follows. First, chapter 2 gives a brief overview of the software in which the mesh deformation method is implemented. Next chapter 3 gives an overview of the current state of the art mesh deformation methods. In chapter 4, the test cases that are used throughout the thesis, are presented. The implemented inverse distance weighting (IDW) mesh deformation method is introduced with great detail in chapter 5. This is followed in chapter 6 by a presentation of possible efficiency improvements, of which one, the boundary coarsening method, is implemented and explained in further detail. Finally, the performance of the implemented IDW mesh deformation method is compared to the radial basis funcation (RBF) and elastic analogy mesh deformation (EAMD) method in chapter 7, by means of five different test cases. The thesis is concluded by several recommendations and conclusions in chapter 8.

Chapter 2

NUMECA International Software

In this project a mesh deformation method will be designed specifically for the CFD software FINE^{TM}/Open from NUMECA International. FINE^{TM}/Open is NUMECA's CFD **F**low **IN**tegrated **E**nvironment for complex external and internal flows. This software package is composed of three separate systems [47]

- HEXPRESS^{TM},
- FINE^{TM}/Open flow solver,
- CFView^{TM}.

The first system, HEXPRESS^{TM}, is the mesh generator [48]. The generated meshes are unstructured and fully hexahedral, i.e. no prisms, tetrahedra or pyramids are present. HEXPRESS^{TM} meshes are generated in five main steps

- Initial mesh
- Adapt to geometry
- Snap to geometry
- Optimize
- Viscous layers

The first step generates an initial mesh with all perfectly orthogonal cells. This mesh is then adapted to match the characteristic geometry lengths by means of refinement. The snapping phase projects the mesh on the geometry and recovers lower dimensional features. Then the mesh is optimised by means of smoothing algorithms. Finally viscous layers are inserted along the solid boundaries when necessary. [49]

The second system, FINE^{TM}/Open flow solver, is the core of the software. It is responsible for all aspects of the flow simulation. The solver is designed to be able to handle a wide range of simulation cases, both with external and internal flow. It can handle all types of fluids (incompressible, low-compressible, condensible and fully compressible) and speed regimes (low speed to hypersonic). The solver uses the finite volume discretisation of the equations and is capable of doing multigrid and parallel computations to accelerate the process. [47]

The FINE^{TM}/Open flow solver has two integrated coupling modules that allow for fluid-structure interaction. The first module is FINE^{TM}/FSI-OOFELIE. It is a fully integrated solution for strongly coupled fluid-structure interaction simulations. As the name indicates it combines FINE^{TM}/Open with the CSM software Oofelie, developed by Open Engineering. The communication between the two solvers is done with the Message Passing Interface (MPI) protocol. The temporal coupling scheme between the two solvers is a strongly coupled staggered approach. Hence at each time step several iterations are done until synchronisation between the two solutions is obtained. [46]

The second coupling module is the mesh-based parallel code coupling interface (MpCCI). This is an interface to perform multi-physics simulations, developed by the Fraunhofer Institute (SCAI). The interface consists of a server that can couple two or more solvers such as a CSM and a CFD solver. The MpCCI allows for several global and cell face values to be communicated between the solvers. In the case of FSI, MpCCI would be used to pass the displacement of the structure to the fluid and the pressure of the fluid to the structure. Note that MpCCI can only be used for loosely coupled systems as no sub-iterations are present. [46]

The last system, CFView^{TM}, is a graphical user interface that allows to visualise all the results.

Chapter 3

Mesh Deformation Algorithms

The interaction between fluids and moving structures has been a topic of research for many years. Furthermore mesh deformation methods are also applied extensively in aerodynamic shape optimisation. As such many mesh deformation methods have been designed and investigated in the past. This chapter aims to provide a brief overview of these mesh deformations methods. The first three sections describe three different groups of mesh deformation algorithms: mesh connectivity, point-by-point and hybrid. Next, in section 3.4 an overview is given of the mesh deformation methods that are already available in the FINE^{TM}/Open software. Finally, section 3.5 gives a synthesis of the described methods.

3.1 Mesh Connectivity Schemes

In mesh connectivity based schemes the mesh topology is needed to move cells based on the displacement of its neighbouring cells. These methods are usually based on some physical analogy, such as elasticity or diffusion. The most popular of the connectivity based schemes is the spring analogy [4], where the mesh is represented by a network of springs, whose spring stiffness is related to some geometric quantity such as the length of the edge. The basic spring analogy scheme cannot prevent negative cells due to edge cross-overs. Therefore several improvements, such as the torsional [17], semi-torsional [8][73], ball-vertex [9] and ortho-semi-torsional [40] spring analogy method have been introduced. All these methods significantly increase the robustness of the spring analogy method, but usually also introduce an extra computational cost.

Another structural analogy scheme is based on elasticity, and models the mesh as a linear elastic solid [39]. The main challenge when using the elastic analogy is how to model the elastic properties of the mesh. Several criteria can be used to stiffen the cells that are most prone to inversion. These criteria can be geometry [28], distance [11], distortion [3] or strain field [26] based. The elastic analogy is significantly more robust than the basic spring analogy,

but also introduces an extra computational overhead.

The last group of mesh connectivity based schemes are elliptic smoothing techniques, which model the mesh deformation system as a diffusion problem. Here the Laplacian equation is the most used smoother [36]. Similar to the the elastic and spring analogy, a variable diffusion coefficient is necessary to increase the robustness of the method. However, even with this variable diffusivity, the Laplace equation is often only able to handle small deformations and does not maintain an orthogonal mesh. Helenbrook [25] provided a solution to this problem by using the biharmonic operator which allows to impose a second set of boundary conditions, such that the near-boundary cell quality is maintained. However, the biharmonic operator is also significantly more expensive than the Laplacian equation.

In order to increase the quality of the meshes that are deformed with structural analogy schemes, Samareh [58] introduced the use of quaternion algebra. These quaternions are used to represent the rotations of the boundary. Just like the displacement vectors, the quaternions can be interpolated through the mesh by means of mesh connectivity schemes such as the spring analogy or the Laplacian method. Finally the position of the mesh nodes can be updated by combining the influence of the rotation and translation component. The usage of quaternions increases the orthogonality compared to the original spring analogy method [58].

A disadvantage of all mesh connectivity based methods, is that for different mesh topologies, the implementation will be different. Also irregularities, such as hanging nodes, where only one of the neighbouring elements is subdivided, require special treatment. Furthermore mesh connectivity based methods often involve solving a system of partial differential equations with computational techniques such as finite elements or finite volumes and are therefore usually quite expensive. Due to the connectivity information involved in the scheme, it is often not straightforward to implement them in parallel. Alternatively the efficiency can be increased with a multigrid approach [71], which introduces extra complexity. Moreover non-linear terms can cause slow convergence, resulting in a large amount of iterations. Structural analogies also often lead to stiff matrix systems, especially in the case of viscous meshes with high aspect ratio cells. This means that only small displacements can be treated, or larger displacements have to be divided in smaller sub-displacements. As such structural analogy methods are usually not ideal for viscous flow FSI simulations.

3.2 Point-by-Point Schemes

Point-by-point schemes typically solve the mesh deformation problem by interpolating the displacement of the boundary nodes to the volume nodes. Usually these schemes do not require any connectivity information. This has the advantage that arbitrary mesh types with hanging nodes can be treated in a uniform way. Additionally point-by-point schemes can easily be implemented in parallel. One of the most popular point-by-point schemes is based on radial basis function (RBF) interpolation as introduced by de Boer et al [14]. This method uses the displacement of the boundary nodes to construct an interpolation function that is a sum of radial basis functions. One of the advantages of the RBF method is that

it can handle large deformations, with sufficient quality. Furthermore its implementation is straightforward and easily done in parallel. Also the method does not require different coefficients for different test cases, such as is often the cases with structural analogy methods. Despite all the advantages, the RBF in its most basic form is not applicable to large industrial FSI cases, because the cost of solving the system scales with $\mathcal{O}(n_b^3)$, and of the evaluation scales with $\mathcal{O}(n_b n_i)$, where n_i is the number of inner mesh nodes and n_b is the number of boundary nodes. Rendall and Allen [54] proposed to use a greedy algorithm in order to select a reduced set of surface nodes, while minimising the surface error. This method showed good results, both in terms of efficiency and robustness. However an unwanted surface error is introduced at the unused boundary nodes. This was resolved by adding a very efficient correction step based on a simple decaying nearest neighbour interpolation [55].

Witteveen [70] showed that Inverse Distance Weighting (IDW) interpolation can be used instead of RBF interpolation to deform the mesh. Thanks to the fact that IDW is an explicit interpolation technique, it does not require the expensive matrix inversion to solve the system. As such, the IDW technique can be significantly faster that RBF, with only a moderate reduction in robustness. However the cost of the evaluation (just like for RBF) scales with $\mathcal{O}(n_b n_i)$ and therefore the method becomes expensive for large 3D cases. To resolve this issue Luke et al [38] introduced an efficient kd-tree optimisation which uses approximations for the far-field nodes.

Another point-by-point scheme is the Delaunay Graph Mapping (DGM) method as introduced by Liu et al [33]. In this method the Delaunay graph of the boundary nodes is used in order to interpolate the boundary displacement to the volume mesh by means of barycentric coordinates. The DGM method has proven to be very efficient compared to many other mesh deformation schemes, such as the spring analogy method. However, it does not maintain orthogonality in case of rotations, and larger displacements might have to be split into smaller sub-displacements.

Allen [1] introduced a surface influence technique, where each node is updated based on the deformation of the nearest nodes on both the moving and the far-field surface. Several other novel point-by-point schemes can be found in the literature. These include barycentric coordinate interpolation [64], disk-relaxation [74] and neural networks [62]. These novel methods will not be discussed further here, as they are still in the earlier stages of their development.

3.3 Hybrid Schemes

Recent mesh deformation research often proposes the use of hybrid mesh deformation strategies. These hybrid methods apply more than one mesh deformation strategy to deform the mesh. The reason to apply more than one technique is that each technique can be applied based on its own strengths to appropriate problems. Often a robust, but expensive, mesh deformation method is combined with a less robust but efficient method. One example is to apply a robust method to a coarse level grid and to interpolate this deformation to the fine grid with a cheaper method, as done in the moving submesh approach of Lefrançois [32].

Another approach is to introduce several deformation steps as in Martineau's and Georgala's [41] predictor-corrector scheme. The first step is a fast rigid body initialisation, followed by a more robust structural analogy scheme in the second step. Hybrid approaches have also been proven to be efficient in moving viscous meshes, by adapting a different method to the viscous layers than to the rest of the mesh. The method for the viscous layers makes specific use of the high aspect ratio characteristic of these cells [29][43]. Finally also the surface correction, when applying boundary node coarsening in RBF mesh deformation, can be seen as a hybrid method. For example Kowollik [31] proposed to use the DGM method for the correction step.

3.4 Review of Mesh Deformation Available in FINE^{TM}/Open

Two different mesh deformation methods are already present in NUMECA's CFD software FINE^{TM}/Open, namely the quaternion method and the radial basis function interpolation method. Furthermore, the implementation of the elastic analogy mesh deformation (EAMD) has started, however the development has not been finalised yet due to limited performance so far. The three methods are explained in more detail in the following sections.

3.4.1 Mesh Deformation with Quaternions

The implementation of the quaternion method is based on the quaternion method of Samereh [58] as described in section 3.1. This means that the deformation is split up in a rotation and translation component and then interpolated through the mesh. The interpolation into the mesh is done with the Laplacian smoothing method. In order to increase the efficiency of this method, the rigid body initialisation of Martineau and Georgala [41] as described in section 3.3 was added as an initial guess for the mesh node positions. This method performs well for small deformation cases, however in case of large deformations the method fails, especially when viscous layers are present. For further details of the combined method the reader is referred to [30].

3.4.2 Mesh Deformation with RBF

Deformation with RBF is the default deformation module in FINE^{TM}/Open. The implemented method is based on the work of de Boer [14]. RBF mesh deformation uses the displacement of the boundaries to construct an interpolation function $u(\boldsymbol{x})$ that is a sum of radial basis functions

$$u(\boldsymbol{x}) = \sum_{b=1}^{n_b} \alpha_b \phi(\|\boldsymbol{x} - \boldsymbol{x}_b\|) + p(\boldsymbol{x}), \tag{3.1}$$

where $\boldsymbol{x}_b$ are the boundary nodes in which the displacement is known, p is a polynomial, n_b is the number of boundary nodes, α_b are coefficients and ϕ is a basis function with respect to

the Euclidean distance $\|\cdot\|$. The coefficients α_b and the polynomial have to be chosen in such a way that the known displacements at the boundary nodes are interpolated exactly

$$u(\boldsymbol{x}_b) = u_b, \tag{3.2}$$

where u_b are the known displacements at the boundary in one specific direction, because each direction (x, y, z) is interpolated separately. Furthermore there is the additional requirement for the coefficients α_b that

$$\sum_{b=1}^{n_b} \alpha_b q(\boldsymbol{x}_b) = 0, \tag{3.3}$$

where q can be any polynomial with a degree less or equal than the degree of polynomial p. In order to find the coefficients α_b and the linear polynomial, equations 3.2 and 3.3 can be transformed into the following matrix system

$$\begin{bmatrix} \boldsymbol{u}_b \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{b,b} & \boldsymbol{P}_b \\ \boldsymbol{P}_b^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{3.4}$$

where $\boldsymbol{\alpha}$ contains the coefficients α_b, $\boldsymbol{\beta}$ contains the coefficients of the linear polynomial p, $\boldsymbol{M}_{b,b}$ is the $n_b \times n_b$ interpolation matrix given by

$$M_{jk} = \phi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|), \tag{3.5}$$

and $\boldsymbol{P}_b$ is a matrix which is determined by the choice of polynomial. In the case of a first degree polynomial, $\boldsymbol{P}_b$ is given by an $n_b \times 4$ matrix

$$\boldsymbol{P}_b = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n_b} & y_{n_b} & z_{n_b} \end{bmatrix}. \tag{3.6}$$

Finally the displacement values of the internal grid points u_i at location $\boldsymbol{x}_i$ can be found by applying the interpolation function

$$u_i = u(\boldsymbol{x}_i), \tag{3.7}$$

for each spatial direction separately.

Note that the size of the system in equation 3.4 is $(n_b + 4) \times (n_b + 4)$. This is a lot smaller than the size of the systems that have to be solved in mesh connectivity schemes, which is approximately $n_i \times n_i$. However, in mesh connectivity schemes the matrix system is usually sparse, such that it can be solved iteratively. The RBF system on the other is a dense system, which requires a direct matrix inversion or more complicated solvers.

Several RBFs could be used for mesh deformation. In FINE™/Open, the user can choose between two radial basis functions, namely the thin plate spline (TPS) with global support or the Wendland function with compact support (radius defined by the user). The user also has the option to turn on boundary node coarsening by using the multigrid levels that are

used by the CFD solver. The user decides which multigrid level should be used for boundary node coarsening. A drawback of using a coarse grid is that the displacement at the boundary is not exact, and no correction has been included.

Finally to increase the robustness for specific cases it is possible to set an outer boundary as floating. This means that all the nodes on this boundary will be considered as inner nodes, such that they can move. The downside of floating nodes is that there is no guarantee that they will remain on the boundary face.

3.4.3 Elastic Analogy Mesh Deformation

The elastic analogy mesh deformation (EAMD) method implemented in FINE^{TM}/Open is largely based on the work of Denayer [44]. This method is a variation on the standard elastic analogy, using an orthotropic basis transformation. In this section first the basic elastic analogy method is explained, followed by a description of the specific method used in FINE^{TM}/Open .

In the elastic analogy method, the displacement field of the mesh is determined by the equilibrium equation for elasticity

$$\nabla \cdot \boldsymbol{\sigma} = 0 \text{ on } \Omega, \tag{3.8}$$

where $\boldsymbol{\sigma}$ is the stress tensor and Ω is the computational domain. For an isotropic material the constitutive linear elastic relation between stress and strain may be written

$$\boldsymbol{\sigma} = \lambda \mathrm{Tr}(\boldsymbol{\epsilon})\mathrm{I} + 2\mu\boldsymbol{\epsilon}, \tag{3.9}$$

where $\boldsymbol{\epsilon}$ is the strain tensor, Tr the trace and λ and μ are the Lamé elastic constants. The Lamé constants are a material property, alternatively they can be written in terms of the Young's modulus E and the Poisson's ratio ν as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{3.10}$$

Furthermore a linear relation between the stress tensor and the displacement is used

$$\boldsymbol{\epsilon} = \frac{1}{2}\left[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right]. \tag{3.11}$$

In case of a three dimensional problem in Cartesian coordinates, the six components of the stress tensor of equation 3.9 can be written as

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} = \begin{bmatrix} (2\mu+\lambda) & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & (2\mu+\lambda) & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & (2\mu+\lambda) & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{xy} \\ \epsilon_{xz} \\ \epsilon_{yz} \end{bmatrix}, \tag{3.12}$$

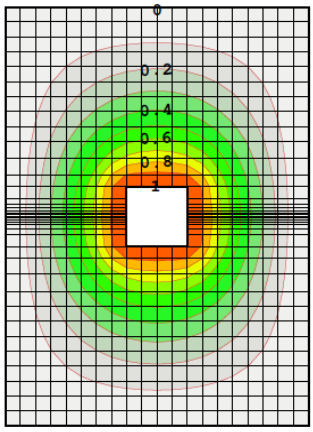


Figure 3.1: Definition of the weighting coefficient ϕ based on the Laplacian solution [44].

and the six components of the strain tensor from equation 3.11 are given by

$$\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{xy} \\ \epsilon_{xz} \\ \epsilon_{yz} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \end{bmatrix}. \tag{3.13}$$

In the implemented EAMD method the Young's modulus is varied throughout the mesh according to

$$E = 8^{a\phi^b}, \tag{3.14}$$

where a and b are empirical coefficients and ϕ is a weighting coefficient that is obtained from the Laplacian solution ($\nabla^2\phi = 0$) where the boundary condition is $\phi = 1$ at the moving boundary and $\phi = 0$ at the fixed boundary as illustrated in figure 3.1. The Poisson's ratio ν is set to 0.3 throughout the whole mesh.

The same weighting coefficient ϕ is also used in order to solve the linear elastic deformation using orthotropic direction in the coordinate system in order to keep the orthogonality of the deformed mesh. For each cell a local orthogonal basis is created, where one vector $\boldsymbol{n}$ is orthogonal to the isometric weighting coefficient line and the other two vectors $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ are tangential to $\boldsymbol{n}$, as illustrated in figure 3.2. By using this approach the equations of linear elasticity are solved for an orthotropic, or more specifically a transverse isotropic material, instead of an isotropic material. The orthotropic solution is obtained by applying a change of basis to the original equations.

$$\boldsymbol{\sigma}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} = \boldsymbol{P}^T \boldsymbol{\sigma}_{(\boldsymbol{n},\boldsymbol{t_1},\boldsymbol{t_2})} \boldsymbol{P}, \tag{3.15}$$

$$\boldsymbol{\epsilon}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} = \boldsymbol{P}^T \boldsymbol{\epsilon}_{(\boldsymbol{n},\boldsymbol{t_1},\boldsymbol{t_2})} \boldsymbol{P}, \tag{3.16}$$

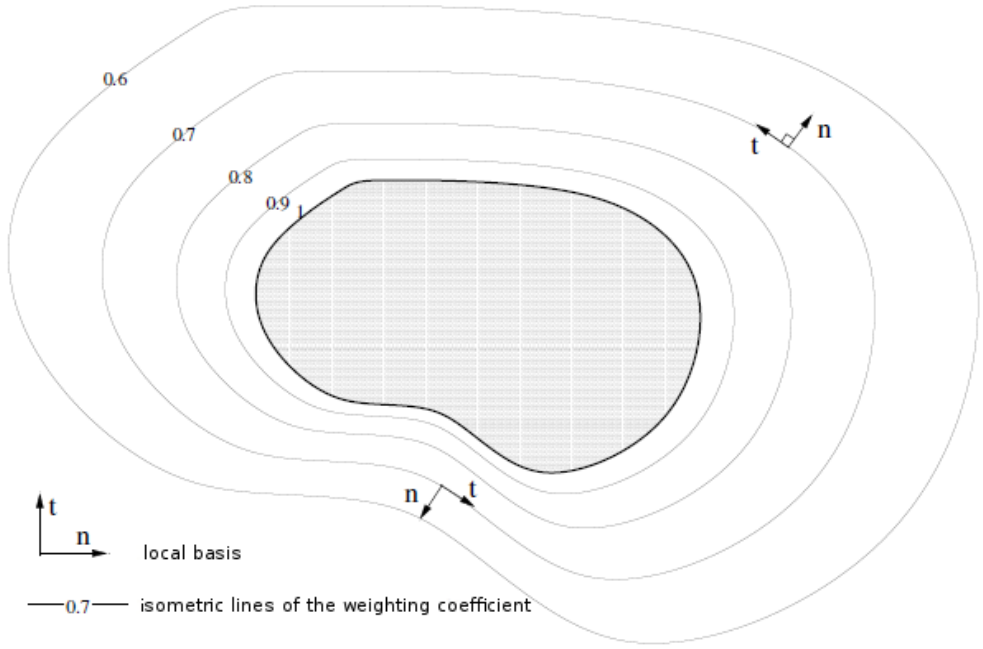


Figure 3.2: Local basis created on the isometric lines of the weighting coefficient [44].

where

$$\boldsymbol{P} = \begin{bmatrix} n_x & n_y & n_z \\ t_{1x} & t_{1y} & t_{1z} \\ t_{2x} & t_{2y} & t_{2z} \end{bmatrix}, \tag{3.17}$$

and $\boldsymbol{\sigma}_{(\boldsymbol{n},\boldsymbol{t_1},\boldsymbol{t_2})}$ and $\boldsymbol{\epsilon}_{(\boldsymbol{n},\boldsymbol{t_1},\boldsymbol{t_2})}$ are obtained analogous to equation 3.12 and 3.13 respectively, only the subscripts x, y and z have to be replaced by n, t_1 and t_2 respectively. The above system results into the following relation

$$\boldsymbol{\sigma}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46} \\ C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56} \\ C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{xy} \\ \epsilon_{xz} \\ \epsilon_{yz} \end{bmatrix}, \tag{3.18}$$

where C_{ij} are coefficients as given in the appendix of [53]. For conciseness the subscript $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is omitted from now, since all the next equations are in the Cartesian system. In order to facilitate the numerical solution of the above system, it is split up into an isotropic part and an orthotropic part. If the matrix in equation 3.18 is named C_{ortho} and the matrix in equation 3.12 is named C_{iso} then one can write

$$\boldsymbol{\sigma} = C_{iso}\boldsymbol{\epsilon} + (C_{ortho} - C_{iso})\boldsymbol{\epsilon}. \tag{3.19}$$

When solving this system the second term is treated explicitly as a source term. The first term is split further into explicit and implicit terms. Let us first recall that the first term in

equation 3.19 can be written in non-matrix notation as in equation 3.9. By combining this with equation 3.11 the first term can be rewritten as

$$C_{iso}\boldsymbol{\epsilon} = \mu\nabla\boldsymbol{u} + \mu(\nabla\boldsymbol{u})^T + \lambda\mathrm{tr}(\nabla\boldsymbol{u}), \tag{3.20}$$

which is split into explicit and implicit terms as follows

$$C_{iso}\boldsymbol{\epsilon} = \underbrace{(2\mu+\lambda)\nabla\boldsymbol{u}}_{implicit} + \underbrace{\mu(\nabla\boldsymbol{u})^T + \lambda\boldsymbol{I}\mathrm{tr}(\nabla\boldsymbol{u}) - (\mu+\lambda)\nabla\boldsymbol{u}}_{explicit}. \tag{3.21}$$

Finally the equilibrium for elasticity can be computed by solving $\nabla\cdot\sigma = 0$ and by moving the explicit terms to the right hand side.

$$\nabla\cdot[(2\mu+\lambda)\nabla\boldsymbol{u}] = -\nabla\cdot\left[\mu(\nabla\boldsymbol{u})^T + \lambda\boldsymbol{I}\mathrm{tr}(\nabla\boldsymbol{u}) - (\mu+\lambda)\nabla\boldsymbol{u}\right] - \nabla\cdot[(C_{ortho} - C_{iso})\boldsymbol{\epsilon}]. \tag{3.22}$$

The right hand side is solved explicitly based on the displacement of the previous iteration. This system is solved iteratively by means of the preconditioned conjugate gradient method, where the inverse of the diagonal matrix is used as pre-conditioner. When the solution changes less than a predefined tolerance, the iterations are stopped.

The elastic analogy method allows to have two types of boundary conditions: fixed or sliding. At a fixed boundary the displacement is zero throughout the iterations. At a sliding boundary the displacement is defined tangential to the boundary, or in other words the displacement normal to the boundary is zero.

3.5 Overview

A broad structured overview of the existing mesh deformation methods is given in figure 3.3. Based on this overview and the discussion in the previous paragraphs, a few general conclusions can be drawn. Firstly it can be stated that there is not one method that is the best for all applications. This is due to the fact that robustness and efficiency seem to be contradicting terms in the world of mesh deformations. A method is either very robust but not efficient or very efficient but not robust or somewhere in the middle. Therefore in applications where deformations are small other methods will be preferred than in applications with very large deformations. This also leads to the fact that in FSI software that is used in the industry, usually a range of mesh deformation algorithms is present. The most common methods in industry practice still are the mesh connectivity based methods (mainly the spring analogy), followed by radial basis function interpolation.

Secondly the most common methods can be ordered in terms of robustness and efficiency as illustrated in figure 3.4. Note that this schematic is only a rough estimation based on a compilation of the literature. The actual robustness and efficiency is highly dependent on the specific implementation details, the type of deformation and the scale of the problem. Mesh-connectivity based solutions, will converge fast in case of small deformations, however for

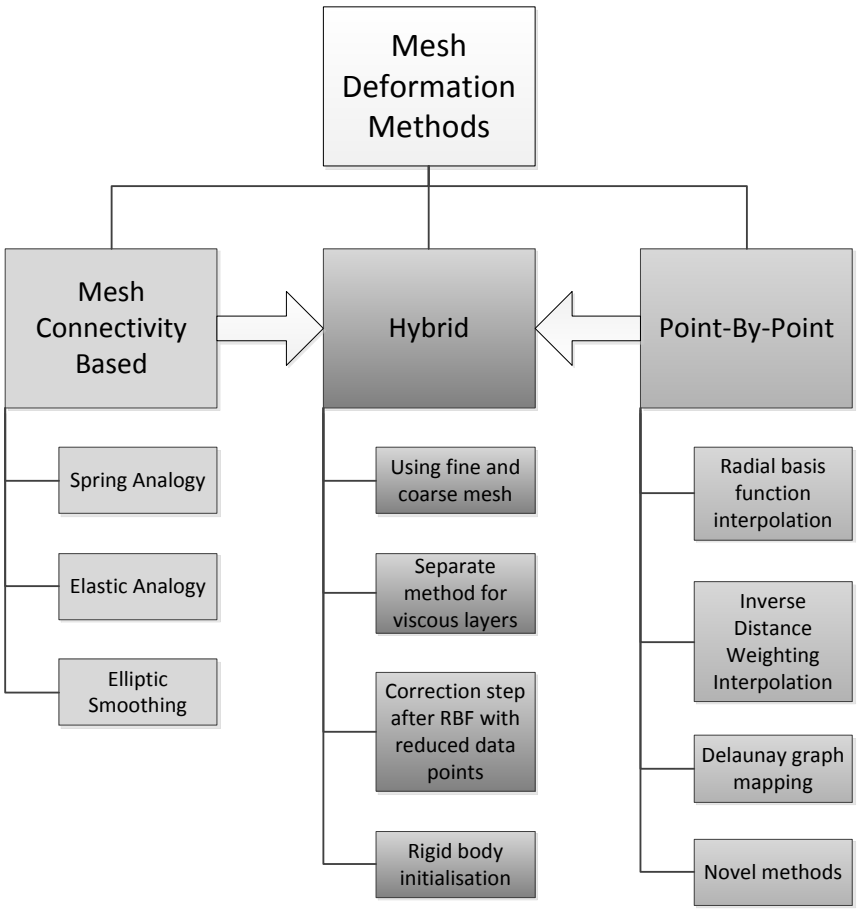


Figure 3.3: Overview of existing mesh deformation methods.

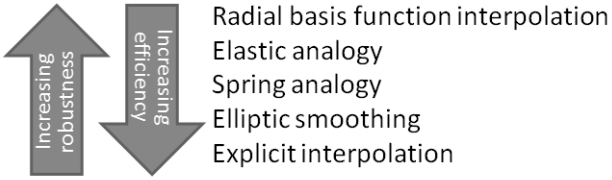


Figure 3.4: Most common mesh deformation methods ordered based on robustness and efficiency.

complex or large deformations the CPU time can increase considerably. On the other hand, the CPU time of point-by-point schemes is not influenced by the type of deformation. In many point-by-point schemes, such as RBF and IDW, the number of boundary points is the determining factor of the CPU time. These methods are very fast in 2D cases, however can become prohibitively expensive for 3D cases. For mesh-connectivity based methods, the total number of nodes plays an important role in CPU time. In terms of robustness, the structural analogies can suffer from poor diffusion of the displacements to the far field, especially when the variation in structural properties is not chosen carefully. RBF usually delivers the highest quality meshes, however due to the fact that the displacements in different directions are not coupled to each other, the orthogonality close to the deforming surface might deteriorate for certain cases such as the bending of a beam.

In this thesis the inverse distance weighting method is further developed. This choice is based on the fact that it combines a good robustness, which is of comparable quality to RBF, with a much higher efficiency as the interpolation function is formed explicitly. Moreover, the method has a low complexity level, which makes it easily maintainable. Finally the method is also believed to be user-friendly, as it does not have many user-parameters. The following chapters will aim to provide a detailed overview of the developed IDW method.

Chapter 4

Test Cases

In this chapter the five test cases that will be used throughout the report are presented. A variety of test cases is chosen to illustrate different performance properties of the mesh deformation method. Both 2D and 3D applications, with smaller and larger scale meshes, are used. For all test cases a large deformation is imposed, which is based on the resulting deformation of previously done simulations. In this chapter each section will present one test case.

An important property of a test case, in regards to the IDW mesh deformation CPU time, is the amount nodes in the mesh. The mesh is divided into boundary nodes, which lie on the domain surfaces, and inner nodes, which are located in the volume mesh. The boundary nodes can be divided further into moving, sliding and fixed nodes. The moving nodes have a displacement which is imposed by the structure solver. They are located on the moving object, such as a wing, flap or airfoil. The fixed nodes are nodes that remain fixed throughout the simulation. Usually these can be found at the exterior boundaries. Finally the sliding boundary nodes are nodes which are allowed to slide along the surface they are located on.

4.1 Rotation of 2D NACA 0012 Airfoil

Table 4.1: Number of nodes: NACA 0012.

	Without sliding nodes	With sliding nodes
Inner Nodes	30,336	30,336
Boundary Nodes	768	768
- Moving	384	384
- Fixed	384	0
- Sliding	0	384
Total	**31,104**	**31,104**

The first test case is a 2D rigid body rotation of a NACA 0012 airfoil. This test case is meant to illustrate the capability of the mesh deformation method to withstand significant rotations. The mesh, depicted in figure 4.1 consists of 31104 nodes, of which 768 are boundary nodes, as indicated in table 4.1. This mesh has a minimum orthogonality of 19.76° and a maximum skewness of 0.81. In order to make this a particularly challenging test case, the mesh has viscous layers around the airfoil, with a cell height of the order of 10^{-6} of the chord length and an aspect ratio up to 2096. The diameter of the computational domain is 80 chord lengths. In the NACA 0012 test case a rigid rotation is imposed of 90° around the leading edge of the airfoil. This deformation is applied in one time step. Depending on the situation, a sliding boundary condition might be used for the exterior boundary. What such a sliding boundary condition entails exactly is explained further in section 5.4.

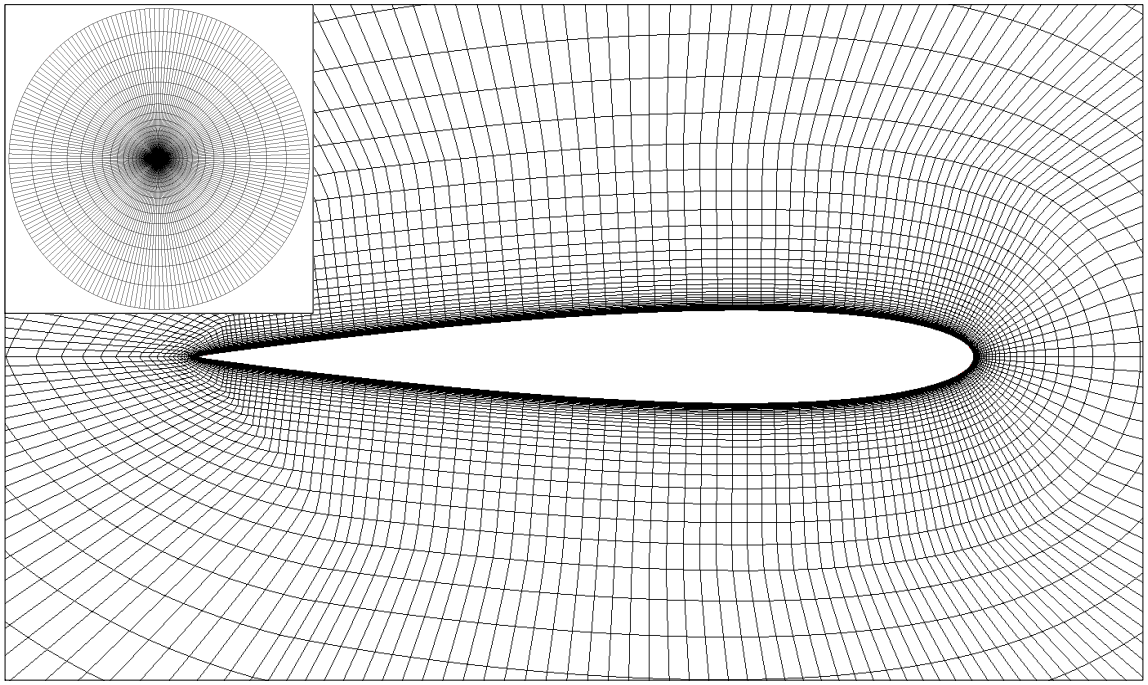


Figure 4.1: Undeformed mesh for the NACA 0012 test case.

4.2 2D Vortex Induced Beam Vibration

The second test case is a 2D vortex induced beam vibration. The set-up of this test case is illustrated in figure 4.2. The beam is flexible and will deform due to the load exerted by the von Karman vortices shed behind the square. This displacement has been computed by means of a time-dependent CFD computation, using the structure modes as an input. The structure mode shapes were computed by means of a FEM simulation and are displayed in figure 4.3(a). The settings for the structure and fluid simulation are given in appendix C.1. During the simulation the beam undergoes a periodic motion, which grows during the initial transient phase and then remains at a constant amplitude. For the tests throughout the report, the maximum displacement which occurred during the CFD computation is imposed

Table 4.2: Number of nodes: Vortex induced beam vibration.

	Without sliding nodes	With sliding nodes
Inner Nodes	28,708	28,708
Boundary Nodes	948	948
- Moving	354	354
- Fixed	594	0
- Sliding	0	594
Total	**29,656**	**29,656**

directly, in one deformation step. This maximum displacement, with a tip motion amplitude of 2.4 cm, is shown in figure 4.3(b).

The mesh for the vortex vibration test case is displayed in figure 4.4. It consists of 29,656 nodes, of which 948 are boundary nodes, as summarised in table 4.2. The quality of the initial mesh is characterised by a minimum orthogonality of 49.11° and a maximum skewness of 0.54. Hereafter the vortex induced beam vibration test case will be referred to as vortex vibration. For more information about this test case the reader is referred to reference [19].

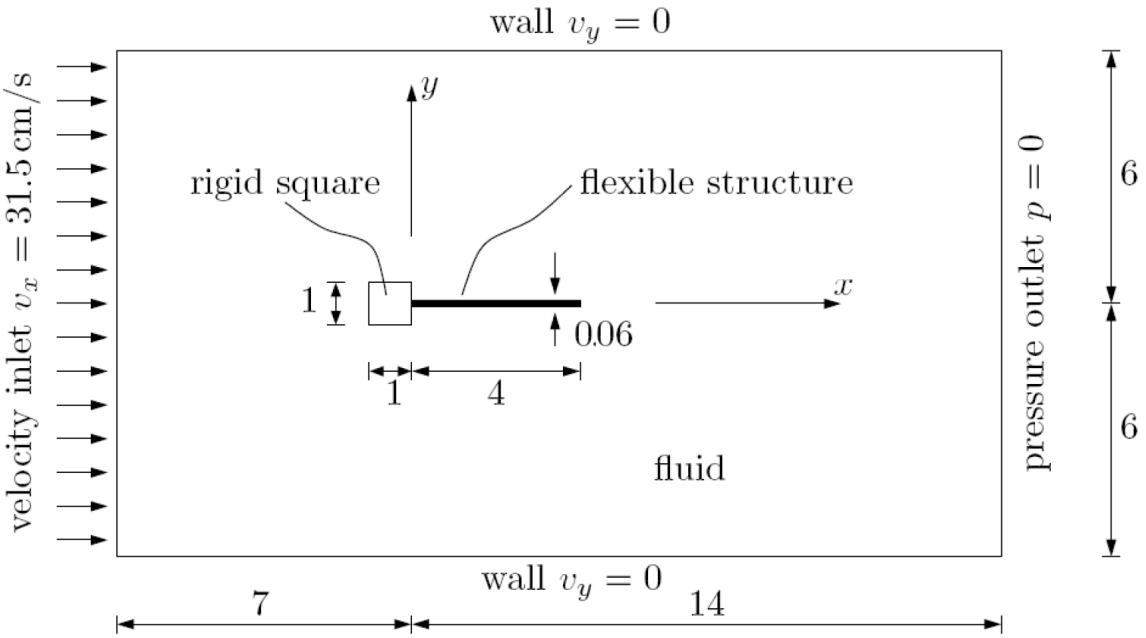
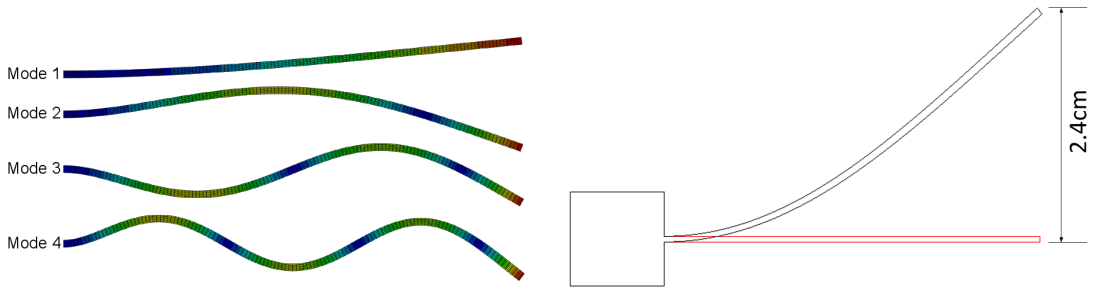


Figure 4.2: Set up of the vortex induced beam vibration test case [19] (dimensions are in centimetres).



(a) Structural modes [15]. (b) Imposed deformation.

Figure 4.3: Deformation of the vortex induced beam vibration test case.

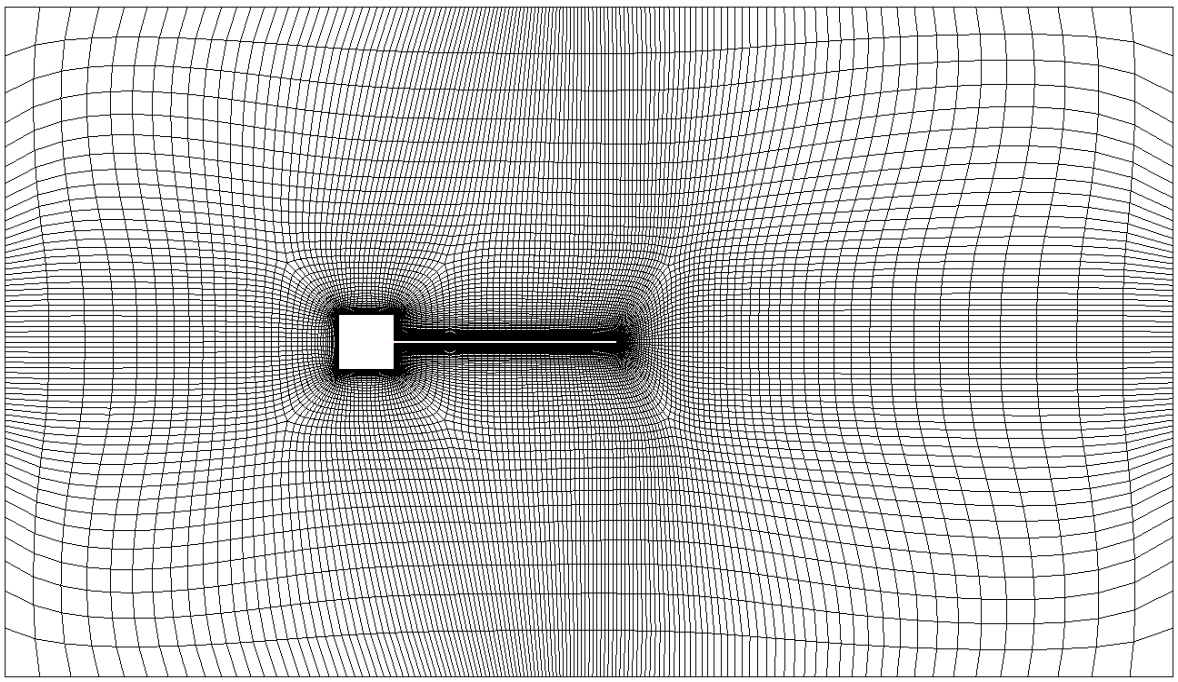


Figure 4.4: Mesh for the vortex induced beam vibration test case.

4.3 Elastic Flap in a Duct

Table 4.3: Number of nodes: Elastic flap.

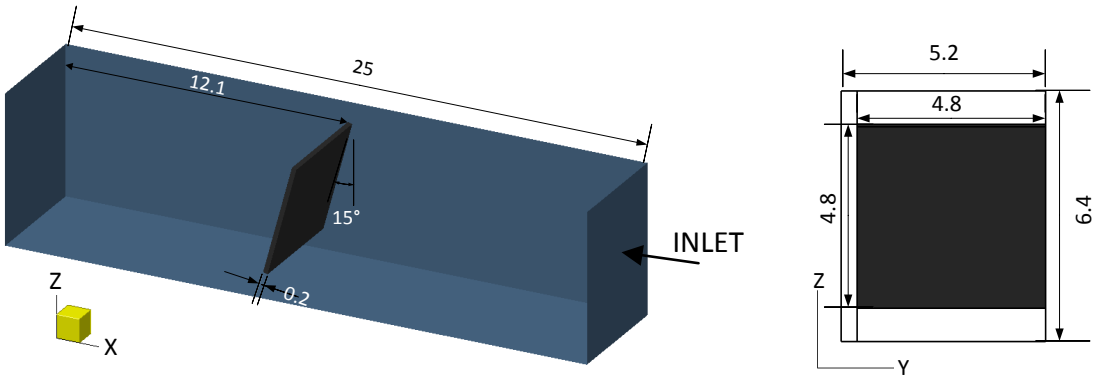
Inner Nodes	110,775
Boundary Nodes	17,282
- Moving	2,273
- Fixed	7,584
- Sliding	7,425
Total	**128,057**

This test case considers an elastic flap inside a 3-dimensional duct, as illustrated in figure 4.5. The flap is attached at the middle of the back plate of the duct and is placed at an angle of 15°. The flow enters the duct on the right and exits at the left. Thanks to the gaps between the flap and three walls of the duct, it is possible for the flow to move past the flap. However, the flap forms an obstacle to the flow, and hence the flow exerts a force on the flap, which causes the flap to deform. More information about this test case can be found in [19].

The deformation of the flap can be computed in an unsteady CFD simulation which uses the mode shapes of the elastic flap as an input. The flap undergoes a damped oscillation, before reaching a steady deformed position. For the purpose of testing the mesh deformation algorithm, the maximum displacement that occurred during the CFD simulation is imposed directly. The simulation settings to achieve this result are summarised in appendix C.2.

The maximum displacement of the flap is illustrated in figure 4.6. The amplitude of the motion is 0.01m at the tip of the flap. This displacement is especially tricky for the mesh motion algorithm, since it is a relatively large displacement in relatively close proximity of three surrounding walls. As such the upper, lower and front wall receive a sliding boundary condition, such that the surface mesh can follow the displacement of the flap.

The mesh of the elastic flap case consists of 128,057 nodes, of which 17,282 are boundary nodes as listed in table 4.3. The initial mesh, visualised in figure 4.7, has a maximum skewness of 0.19, while the minimum orthogonality is 67.49°.



(a) Set-up of elastic flap test case. (b) Inlet.

Figure 4.5: Elastic flap test case (dimensions in cm).

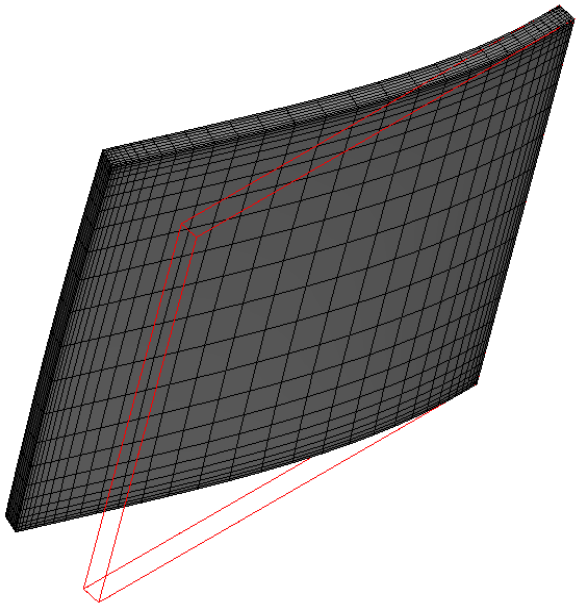
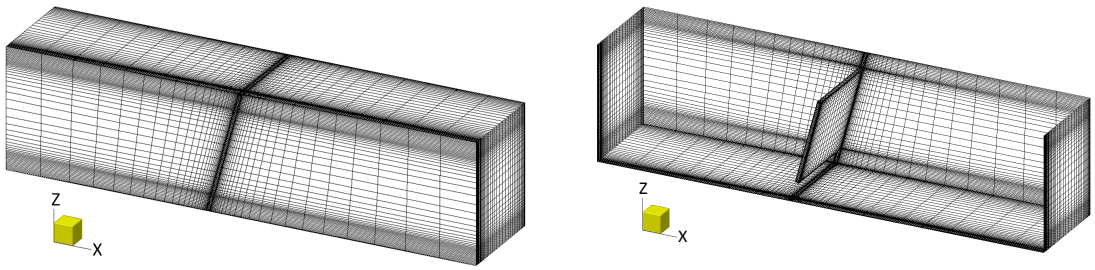


Figure 4.6: Deformation of the flap.



(a) External view. (b) Internal view.

Figure 4.7: Initial mesh of the elastic flap test case.

4.4 Rotor 67 Blade Deflection

This test case considers the deformation of a blade of the NASA rotor 67, which is a transonic axial-flow fan, as defined by Strazisar et al [63]. The geometry of the rotor is depicted in figure 4.8. It consists of 22 blades with a tip gap of 0.1016 cm. The axial width of the rotor is 23.2 cm. The radius of the rotor at the blade tip is 25.7 cm at the inlet and 24.3 cm at the outlet. At the hub, these radii are 9.64 cm and 11.6 cm respectively.

Due to the fact that all the blades of the rotor are identical, the computational domain of this test case only consists of one blade in a channel with periodic boundaries. Two meshes are considered, a coarser and a finer one, as shown in figures 4.9(a) and 4.9(b) respectively. The number of nodes in the fine and coarse mesh are listed in table 4.4. The minimum orthogonality for the initial fine mesh is 32.01° and the maximum skewness is 0.68. For the coarse mesh the values are similar with a minimum orthogonality of 31.99° and a maximum skewness of 0.72.

The blade of the rotor undergoes a steady deformation, due to two types of loads: centrifugal and aerodynamic. The centrifugal deformation can be computed with a CSM simulation. Whereas for the aerodynamic loads, the deformation can be computed with a modal approach CFD simulation. Such a computation has been performed by Debrabandere in [15]. However, since in this thesis the interest lies in the mesh deformation, it has been decided to simply impose a modal displacement that is larger than the resulting displacement as reported in [15]. Therefore the resulting modal displacement of the CFD simulation (see appendix C.3), which uses 30 mode shapes, is multiplied with a factor of 5. This means that the shape of the blade deformation does not represent a realistic one as the modal shapes are exaggerated and the non-linear terms are not represented. The deformed blade is shown in figure 4.10. The tip displacement is 0.53 cm at the leading edge and 0.22 cm at the trailing edge. This deformation is not large when compared to the size of the blade, however the challenge for this test case lies in the fact that there is only a 0.1 cm gap between the blade and the shroud. This gap is shown in figures 4.11 for both the fine and the coarse mesh. To maintain a good mesh quality after the deformation, it is important to impose a sliding boundary condition at the shroud.

Table 4.4: Number of nodes: Rotor 67.

	Coarse	Fine
Inner Nodes	82,887	697,935
Boundary Nodes	17,730	70,914
- Moving	4,941	19,641
- Fixed	9,156	36,552
- Sliding	3,633	14,721
Total	**100,617**	**768,849**

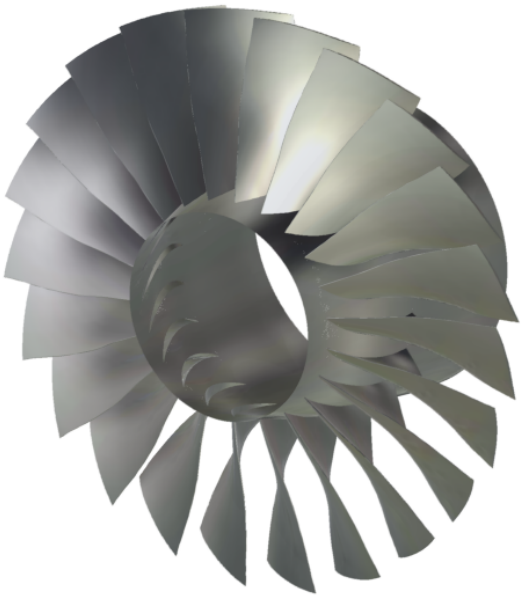
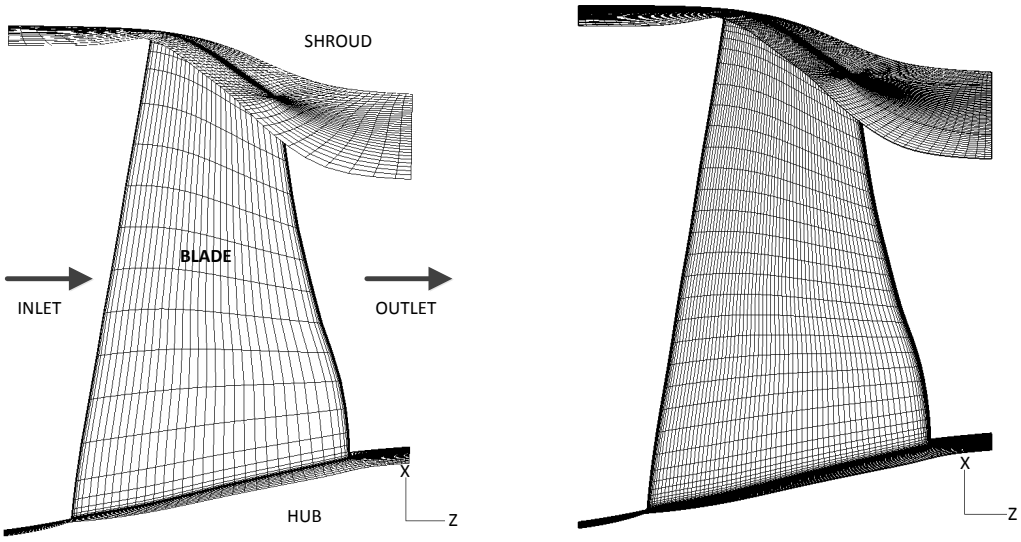


Figure 4.8: Geometry of rotor 67.



(a) Coarse mesh. (b) Fine mesh.

Figure 4.9: Initial mesh of the rotor 67 test case.

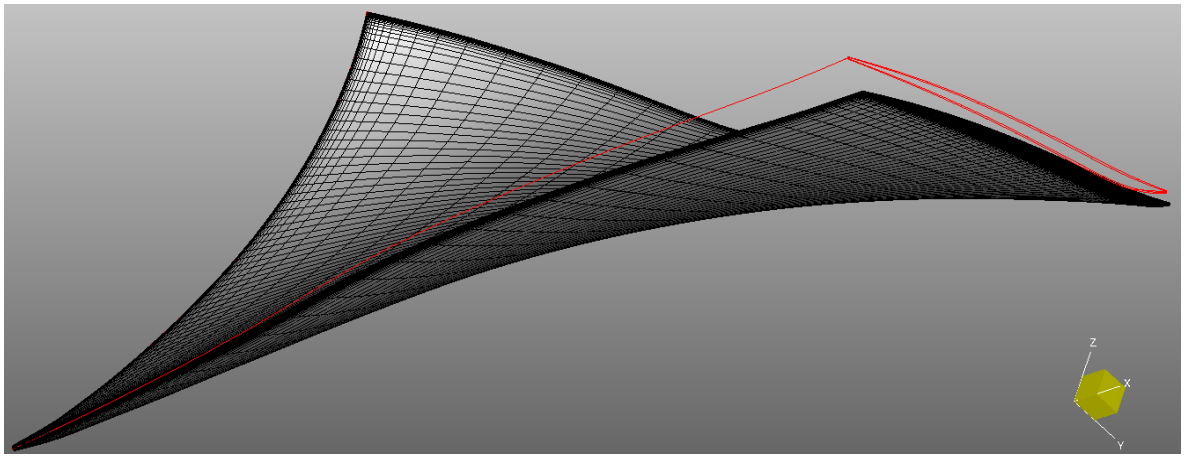
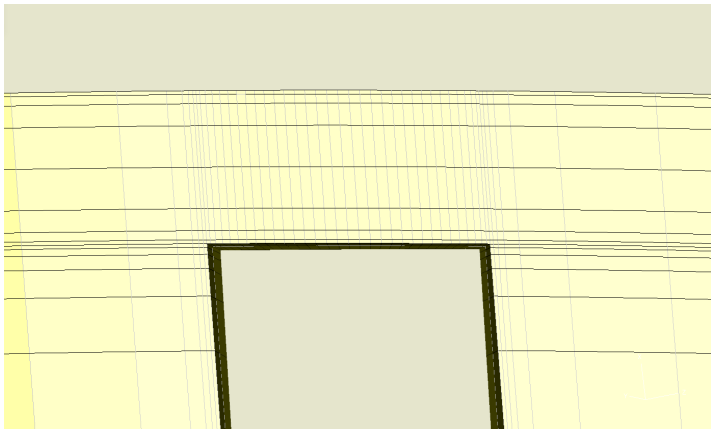
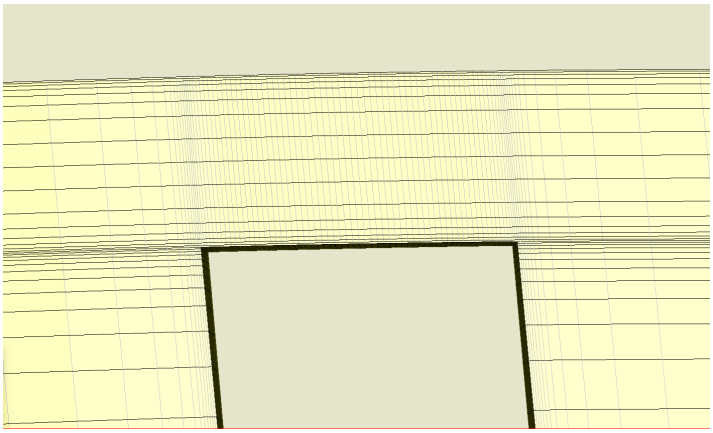


Figure 4.10: Deformed shape of the rotor 67 blade.



(a) Coarse mesh.



(b) Fine mesh.

Figure 4.11: Cut through the mesh, showing the small gap between the blade and the shroud.

4.5 AGARD 445.6 Wing Deflection.

Table 4.5: Number of nodes: AGARD 445.6.

	Coarse	Fine
Inner Nodes	339,967	2,785,983
Boundary Nodes	33,602	134,402
- Moving	5,205	20,713
- Fixed	28,397	113,689
- Sliding	0	0
Total	**373,569**	**2,920,385**

This test case considers the deflection of the AGARD 445.6 wing, as experimentally studied in [72]. The wing has a NACA 65A004 airfoil, a root chord of 0.5587 m, a quarter-chord sweep angle of 45°, a half wing span of 0.762 m and a taper ratio of 0.6, as illustrated in figure 4.12. The domain extends to 5 chord lengths in front of the wing and 4.5 chord lengths behind the wing and it is 10 chord lengths high, with the wing placed in the middle.

For this test case, once again, a coarse and a fine mesh are used (see figure 4.13). The minimum orthogonality of the initial fine mesh is 26.26° and the maximum skewness is 0.60. The coarse mesh has a slightly lower initial quality with a minimum orthogonality of 24.48° and maximum skewness of 0.75. Moreover the fine mesh is quite large, with nearly 3 million grid points, whereas the coarse mesh has less than 400,000 grid points.

The deflection of the AGARD 445.6 wing can be computed with a modal approach CFD simulation. For this simulation the first six modes are taken from the experimental data of Yates [72]. A velocity perturbation in the vertical direction is imposed during the first 0.05 s. This perturbation induces a change in the lift, which causes the wing to deflect. If the flow conditions are above the flutter limit, this motion will be amplified. The settings for the simulation and the imposed perturbation are presented in appendix C.4. The largest modal displacement that occurred during the flutter simulation, is directly imposed throughout this

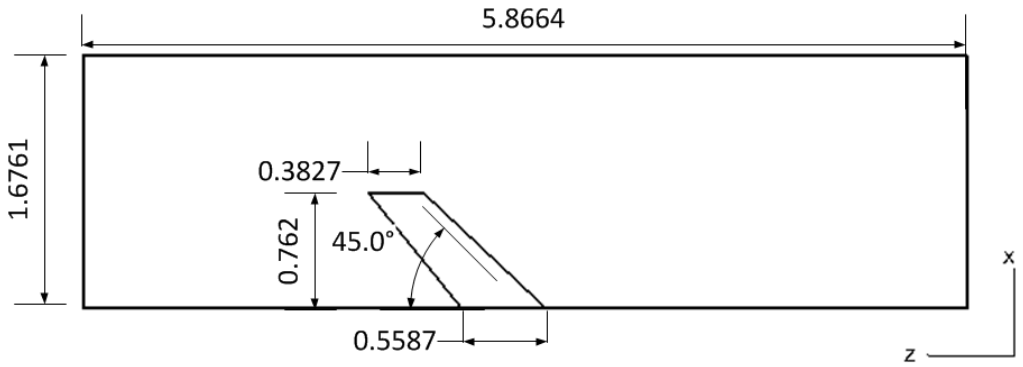
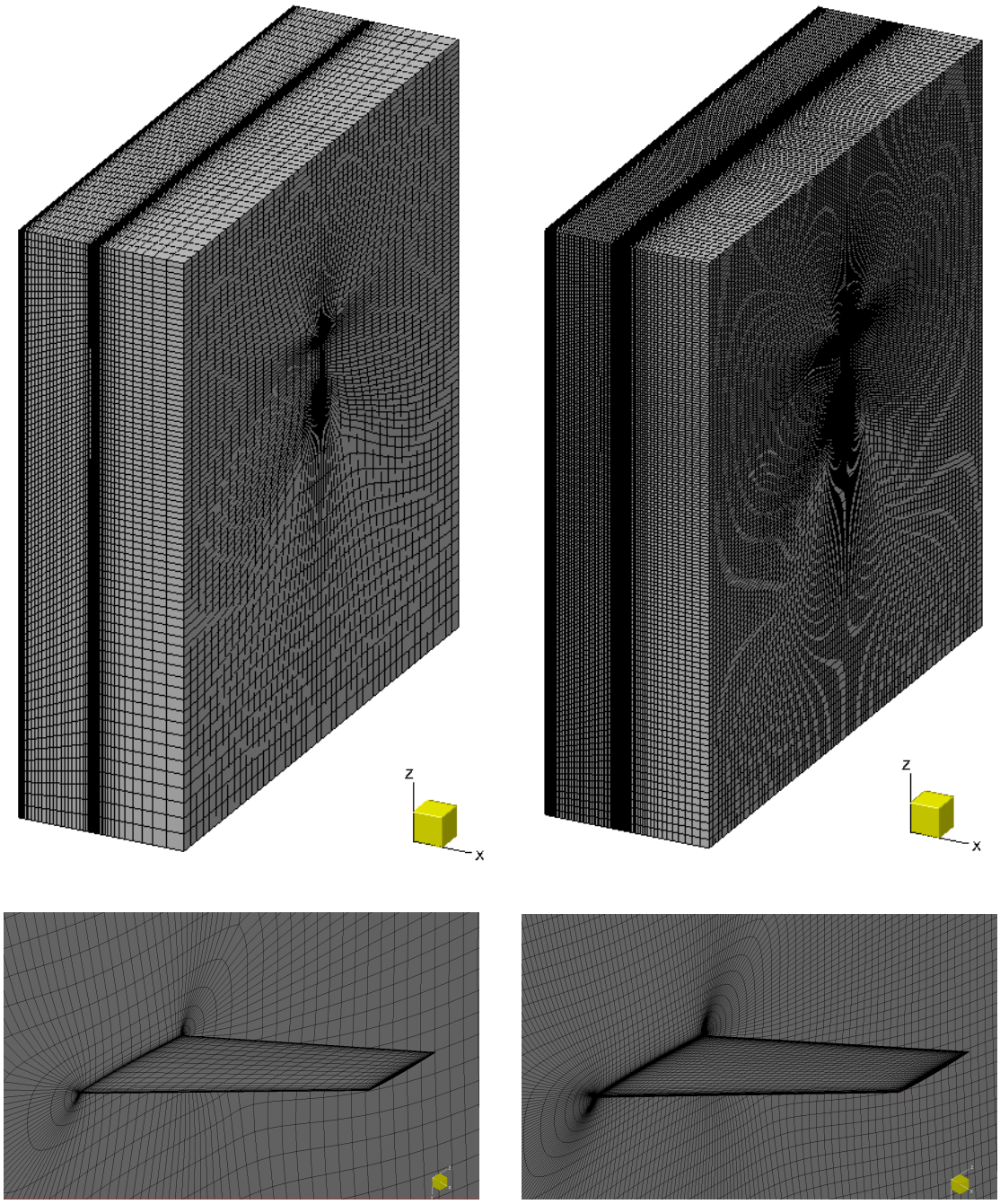


Figure 4.12: Dimensions of the AGARD 445.6 wing test case (meters).



(a) Coarse mesh.

(b) Fine mesh.

Figure 4.13: Mesh of the AGARD 445.6 wing test case.

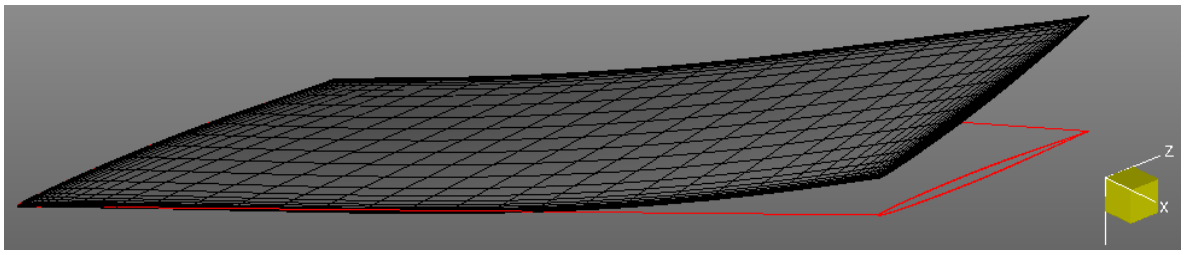


Figure 4.14: Deformation of the AGARD 445.6 wing.

report. This modal displacement is illustrated in figure 4.14. The magnitude of the motion at the wing tip is $0.1\,c$ at the leading edge and $0.3\,c$ at trailing edge, where c is the wing root chord. The motion consists of both a bending and twist mode. Thus this test case can be used to illustrate the ability of the mesh deformation algorithm to handle complex motions, with a rotation component around different axes.

Chapter 5

Inverse Distance Weighting Mesh Deformation

In this chapter inverse distance weighting mesh deformation is explained in detail. The first section provides a detailed background of inverse distance weighting interpolation. Next, in section 5.2, the basic implementation in FINE^{TM}/Open and the choice of the weighting function with its different input parameters are explained. This is followed by a detailed description of how rotations are included in the method, in section 5.3. An important addition to this IDW mesh deformation method, namely the option for sliding nodes on the boundary, is detailed in section 5.4. Finally, the difference between absolute and relative deformations is highlighted in section 5.5.

5.1 Background of Inverse Distance Weighting Interpolation

This section provides the reader with a detailed background regarding IDW interpolation. The first part gives a more general background, whereas the second part focusses on IDW applied to mesh deformation.

5.1.1 General Principle of Inverse Distance Weighting Interpolation

Inverse Distance Weighting (IDW) interpolation is an explicit technique for multivariate interpolation of scattered data points. It was first introduced in 1968 by D. Shepard in the frame work of geographical information systems [59]. In its basic form, IDW interpolation computes the interpolated value u at a given point $\boldsymbol{x}$ as a weighted average of the known values at the data points $u_j = u(\boldsymbol{x}_j)$. The equation of the IDW interpolation function is

given by

$$u(\boldsymbol{x}) = \frac{\sum_{j=0}^{N} w_j(\boldsymbol{x}) u_j}{\sum_{j=0}^{N} w_j(\boldsymbol{x})}, \tag{5.1}$$

where N is the number of data points and $w_j(\boldsymbol{x})$ is the weighting function. This weighting function is proportional to the inverse of the distance. Its standard form is given by

$$w_j(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_j\|^p}, \tag{5.2}$$

where $\|\cdot\|$ is the Eucledian distance operator and p is a power parameter. The power parameter p has to be larger than one in order for the function to be differentiable [60]. Large values of p result in an interpolation that is influenced mainly by the closest data points, leading to steep gradients over low intervals. On the other hand lower values of p result in smooth interpolations where the solution at a point is also influenced by more distant data points. Often the parameter $p = 2$ is chosen because it yields satisfactory results for the least computational effort (since the Eucledian distance contains a square root).

Inverse distance weighting is intensively used in many geographical information systems to create surfaces through spatial data points. Examples can be found in for example hydrology [65], geography [57] and climatology [45]. As such many different forms of IDW exist in order to improve its performance for specific applications. One example is the use of only a certain amount of nearby points or points within a specified radius, in order to make the method more efficient for large spatial applications [59]. Another option is to include slope data in order to solve the fact that local peaks should not always be located at the data points [59], which is the case in the standard method. Another example is the use of spatially varying power parameters, according to varying spatial patterns of the data [37].

5.1.2 Inverse Distance Weighting in Mesh Deformation

Inverse distance weighting was first introduced as a mesh deformation method by Witteveen [70] in 2009. His IDW mesh deformation method uses the standard inverse distance weighting interpolation as introduced in equation 5.1 and 5.2. The value to be interpolated for mesh deformation is the known displacement of all boundary nodes. However for a good mesh quality it is preferable to also interpolate the rotation. Witteveen proposes to use different values for the power parameter p for moving and fixed boundary nodes and for rotations and translations: $p_{\text{mov,r}}$, $p_{\text{mov,t}}$, $p_{\text{fix,r}}$ and $p_{\text{fix,t}}$.

A beneficial property of the IDW for mesh deformation is, that it is an extremum conserving interpolation method. However when also including rotations, this property is lost, meaning that internal nodes could move outside of the domain. In order to prevent this situation, it is checked for each of the N contributions in equation 5.1 whether this contribution would move the point out of the domain. If this is the case, then this contribution is modified such that it places the node onto the boundary of the domain. As such the inner mesh nodes will never be able to cross the outer boundary.

A major advantage of this explicit interpolation technique, compared to RBF, is that the time consuming step to directly solve the system which requires $\mathcal{O}(n_b^3)$ operation, where n_b is the number of boundary nodes, is no longer present. Additionally the implementation of the IDW technique is fairly straightforward since no mesh connectivity has to be taken into account. This makes it possible to deform arbitrary mesh topologies, even with hanging nodes, without any modifications. Finally the IDW method can easily be parallelised.

Luke et al [38] further modified the IDW mesh deformation method to increase its performance. The first modification is to assign a rigid body displacement field to each boundary node. The rotation component is estimated by finding the rotation that best matches the rotation of edges and normals to faces that reference the given node. For each boundary node b a displacement field can be defined as follows

$$\boldsymbol{s}_b(\boldsymbol{x}) = \boldsymbol{u}_b + M_b\boldsymbol{x} - \boldsymbol{x}, \tag{5.3}$$

where M_b is the rotation matrix associated with boundary node b. More information about rotation matrices can be found in section 5.3.1.

The interpolation function is now a vector function given by

$$\boldsymbol{u}(\boldsymbol{x}) = \frac{\sum_{b=0}^{n_b} w_b(\boldsymbol{x})\boldsymbol{s}_b(\boldsymbol{x})}{\sum_{b=0}^{n_b} w_b(\boldsymbol{x})}. \tag{5.4}$$

The second modification to the method is that an improved weighting function is chosen

$$w_b(\boldsymbol{x}) = A_b \left[\left(\frac{L_{def}}{\|\boldsymbol{x} - \boldsymbol{x}_b\|} \right)^a + \left(\frac{\alpha L_{def}}{\|\boldsymbol{x} - \boldsymbol{x}_b\|} \right)^b \right], \tag{5.5}$$

where A_b is the area weight assigned to boundary node b, L_{def} is an estimated length of the deformation region, α is an estimated size of the near body influence region and a and b are the power parameters. Luke et al suggest the values $a = 3$ and $b = 5$ for the power parameters, according to their numerical tests. The parameter L_{def} is usually computed automatically as the furthest distance from any mesh node to the center of the mesh. Parameter α determines the weight of the near boundary nodes over the more distant ones. This should usually be higher when the displacement is more variable. Therefore parameter α can be computed automatically as the maximum difference between a node displacement and the average displacement field as a fraction of L_{def}

$$\alpha = \frac{\eta}{L_{def}} \max_{b=1}^{n_b} \|\boldsymbol{s}_b(\boldsymbol{x}_b) - \boldsymbol{s}_{mean}\|, \tag{5.6}$$

where

$$\boldsymbol{s}_{mean} = \sum_{b=1}^{n_b} a_b \cdot \boldsymbol{s}_b(\boldsymbol{x}_b), \tag{5.7}$$

where $a_b = A_b / \sum_{j=1}^{n_b} A_j$ is the normalised node weight. The symbol η represents a user parameter which determines whether the deformation should be more local (small η) or spread throughout the domain (large η). Generally $\eta = 5$ results in good results. In order to guarantee good near-boundary mesh quality it is important to set $\alpha \geq 0.1$.

Compared to Witteveen's method, Luke's method results in better preservation of the orthogonality of the viscous layers in the mesh, even better than for RBF. This is thanks to the addition of the second term in the weighting factor with a higher power parameter. Furthermore Luke et al propose an efficiency improvement technique using a parallel tree-code optimisation, which allows for a high efficiency and good scaling for large industrial meshes. For more details about this tree-code optimisation, the reader is referred to section 6.1.1.

5.2 Basic Implementation in FINE^{TM}/Open

Based on the background provided in the previous section, IDW can be implemented in FINE^{TM}/Open . This section will describe the most basic implementation and its parameters. Step-by-step improvements to this basic implementation will be described in the following sections.

5.2.1 The Interpolation Function

For the basic implementation it has been chosen to interpolate the given boundary displacements $\boldsymbol{u}_b$, without taken into account rotations. The interpolation function $\boldsymbol{u}(\boldsymbol{x})$ is then given by

$$\boldsymbol{u}(\boldsymbol{x}) = \frac{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})\boldsymbol{u}_b}{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})}, \tag{5.8}$$

where n_b is the number of boundary nodes and $w_b(\boldsymbol{x})$ is a weighting function proportional to the inverse of the distance $d = \|\boldsymbol{x} - \boldsymbol{x}_b\|$. The exact form of the chosen weighting function will be explained in the next section. The interpolation function has to be evaluated for all inner mesh nodes. Therefore it is easy to see that the cost of this method scales with $\mathcal{O}(n_i n_b)$. Furthermore the implementation is very straightforward as it only involves two simple *for*-loops, as illustrated in the pseudo code below.

for *all inner nodes* **do**
  for *all boundary nodes* **do**
    evaluate weighting function $w_b(\boldsymbol{x}_i)$;
    numerator += $w_b(\boldsymbol{x}_i)\boldsymbol{u}_b$;
    denominator += $w_b(\boldsymbol{x}_i)$;
  end
  $\boldsymbol{u}(\boldsymbol{x}_i)$ = numerator / denominator;
end

Note that for the fixed boundary nodes $\boldsymbol{u}_b$ is the zero-vector, which means that the step to update the numerator can be skipped to increase the efficiency of the method. To avoid the cost of doing an if-statement for each boundary node, the most efficient implementation is

done by splitting the boundary node for-loop in two loops, one for the moving nodes and one for the fixed nodes. This is illustrated in the following piece of pseudo-code.

for *all inner nodes* **do**
  for *all moving boundary nodes* **do**
    evaluate weighting function $w_b(\boldsymbol{x}_i)$;
    numerator += $w_b(\boldsymbol{x}_i)\boldsymbol{u}_b$;
    denominator += $w_b(\boldsymbol{x}_i)$;
  end
  for *all fixed boundary nodes* **do**
    evaluate weighting function $w_b(\boldsymbol{x}_i)$;
    denominator += $w_b(\boldsymbol{x}_i)$;
  end
  $\boldsymbol{u}(\boldsymbol{x}_i)$ = numerator / denominator;
end

In the above pseudo-code the evaluation of the weighting function is now also separated for the moving and fixed boundary nodes. As a consequence it is easy to apply different parameters in the weighting function for either moving or fixed boundary nodes.

5.2.2 The Weighting Function

The chosen weighting function in IDW has an important influence on the resulting mesh quality. Therefore it is attempted in this section to choose the optimal weighting function for mesh deformation. As a starting point the weighting function of Luke et al [38], as given in equation 5.5 is used. This weighting function has five different parameters, namely

1. A_b: area weight assigned to boundary node b,
2. L_{def}: estimated length of the deformation region,
3. α: estimated size of the near body influence region,
4. a: 1st power parameter,
5. b: 2nd power parameter.

In order to understand the influence of these parameters, a basic 2D test case is set up. The test case consists of 1×1 moving square inside a 10×10 fixed square, as illustrated in figure 5.1. The inner square will be moved 1.5 units up and 1.5 units to the right in one deformation step.

First, the following values for the default parameters are chosen

- $A_b = 1$,

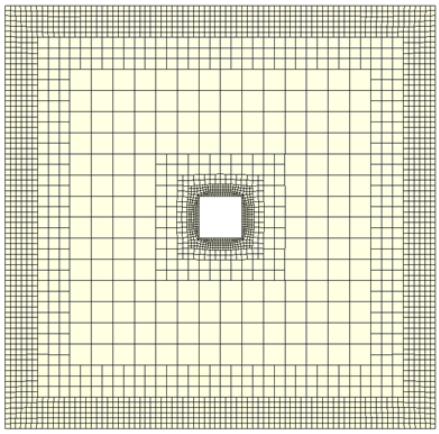


Figure 5.1: Initial 2D mesh of the test case.

- $L_{def} = 5\sqrt{2}$,
- $\alpha_{mov} = 0.1$,
- $\alpha_{fix} = 0.0$,
- $a = 3$,
- $b = 5$.

Most of these parameters, have been chosen in accordance to the proposed values of Luke et al [38]. The first exception is $A_b = 1$, which is equivalent to ignoring this value. The second exception is that two values for α are chosen, one for the moving boundary $\alpha_{mov} = 0.1$ and one for the fixed boundary $\alpha_{fix} = 0.0$. The value $\alpha_{mov} = 0.1$, is based on the maximum difference between the displacement of a boundary node and the average boundary displacement, as in equation 5.6. For a rigid body translation, all nodes have the same displacement and this parameter would have been zero. However Luke et al proposed a minimum value of $\alpha = 0.1$. Since the fixed boundary is not moving, there is no reason to strive for a better near-boundary quality preservation. Therefore α_{fix} is set equal to zero. Finally note that $L_{def} = 5\sqrt{2}$ is based on the computation of the furthest distance from any mesh node to the centre.

In figure 5.2 the resulting mesh is shown for a series of parameter variations. First, the default result is shown in figure 5.2(a). In all other figures, the changed parameters are indicated. In figure 5.2(b) the effect of including the area of the boundary nodes A_b in the weighting function, is shown. Compared to the default case, where A_b has been omitted, the largest distortion of the mesh occurs closer to the moving boundary. This is due to the fact that the fixed boundary has a larger surface area, and hence the weight of the fixed outer boundary is increased, relative to the moving boundary. Similar results were found in a series of other test cases, because the fixed boundary is almost always a lot larger than the moving boundary. When omitting this A_b from the weighting function, more weight is given to regions where the mesh is finer. These regions are usually the regions where the mesh quality is most critical. Therefore it is a desired property to give more weight to these regions, such that they will

deform less. Note, that this also means that a refinement or coarsening of the boundary mesh, will lead to a different interpolation function.

In figure 5.2(c) the parameter L_{def} is multiplied by 1000. In this case this leads to a distorted mesh. It certainly does not make sense to play with this parameter as it is used to scale the weighting function, such that it becomes unitless. Moreover small variations in this parameter did not seem to have any noticeable influence on the final result. Therefore the choice of L_{def} equal to the maximum distance between any mesh node and the mesh centre, seems to be appropriate.

The influence of parameter α is clear in figures 5.2(d) and 5.2(e). When α is increased, the near-boundary region, where the mesh remains more or less rigid, grows. Increasing α_{mov} certainly has a positive effect on the quality of the boundary layer mesh. It is clear that in case the moving object is small compared to the total mesh, α_{mov} can be set larger, in order to maintain a large region of good quality around the moving object. However, it also has the negative side effect that the deformation, then has to be absorbed over a smaller outer region, leading to steep gradients. Therefore it is usually best to use $\alpha_{mov} = 0.1$ seeing as this already maintains a good boundary layer quality, and leaves room for dissipation of the deformation. Similarly α_{fix} should be chosen low, such that the region around the fixed boundary can be used to absorb the deformation. In figure 5.2(a) it is clear that even for $\alpha_{fix} = 0.0$, the quality of the fixed wall boundary layer remains high.

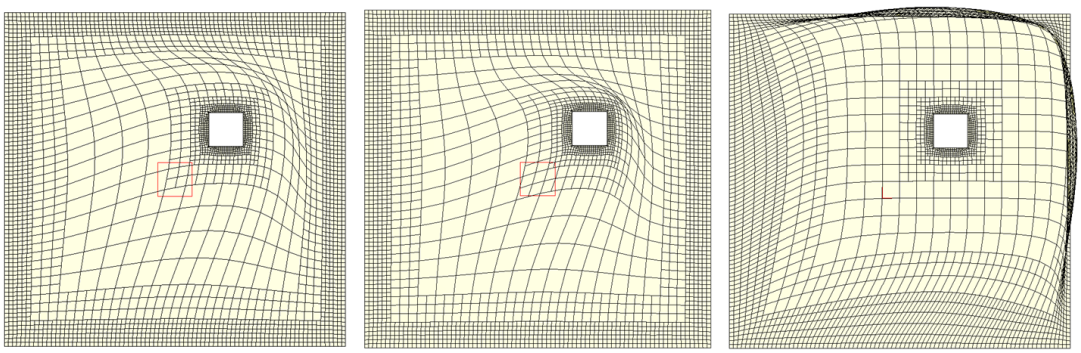
The influence of the power parameters a and b becomes clear in figures 5.2(f) through 5.2(i). A lower power parameter, results in smoother results, meaning that also the near-boundary region will be affected. On the other hand, higher power parameters, lead to steep gradients, where the deformation is absorbed over a small interval in between the two boundaries. When looking carefully at figure 5.2(f) it can be seen that for $a = 2$, the cells close to the moving boundary decrease in quality. This situation should usually be avoided, as the near-boundary quality, is the most critical for obtaining good CFD results. On the other hand, increasing the power parameter to $a = 4$ (see figure 5.2(g)), is clearly not desirable. The mesh remains rigid at both boundaries, and all the motion is absorbed in a small interval, leading to negative cells. Additionally, changing parameter b does not have a large influence on the solution as long as α is low, as can be seen in figure 5.2(h), where b is increased to 7. When α_{mov} is increased to 0.8 at the same time, negative cells are created because the gradient of the deformation is too steep over a very small interval.

In conclusion, the following weighting function has been chosen

$$w_b(\boldsymbol{x}) = \left[\left(\frac{L_{def}}{\|\boldsymbol{x} - \boldsymbol{x}_b\|} \right)^a + \left(\frac{\alpha L_{def}}{\|\boldsymbol{x} - \boldsymbol{x}_b\|} \right)^b \right], \tag{5.9}$$

where

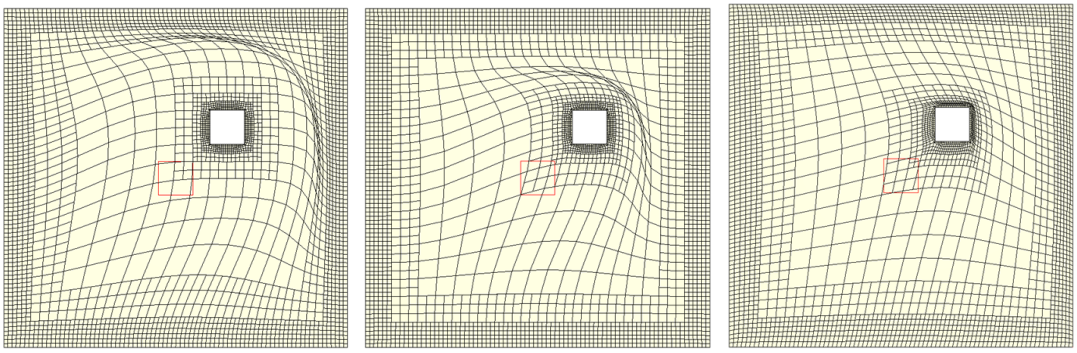
- $L_{def} = \max_{b=1}^{n_b}(\boldsymbol{c} - \boldsymbol{x}_b)$, where $\boldsymbol{c} = 1/n_b \sum_{b=1}^{n_b} \boldsymbol{x}_b$,
- $\alpha_{mov} = 0.1$,
- $\alpha_{fix} = 0.0$,



(a) Default parameters

(b) A_b = area of boundary node b

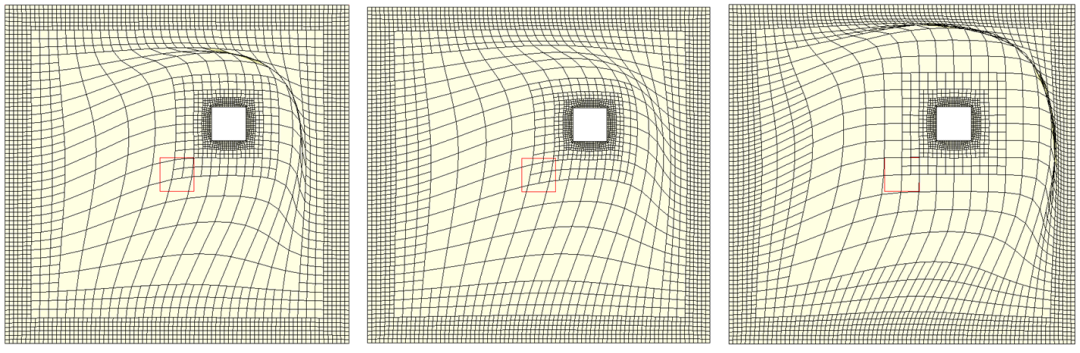
(c) $L_{def} = 2\sqrt{5} \times 1000$



(d) $\alpha_{mov} = 0.8$

(e) $\alpha_{fix} = 0.8$

(f) $a = 2$



(g) $a = 4$

(h) $b = 7$

(i) $b = 7$ and $\alpha_{mov} = 0.8$

Figure 5.2: The influence of the different parameters.

- $a = 3$,
- $b = 5$.

The values for L_{def}, a and b will be fixed values which cannot be changed by the user. Keeping a and b fixed, has the additional advantage that one can avoid using the *C++ pow*-function, which seems to be a rather expensive operation, by simply writing $c^3 = c \cdot c \cdot c$. The values for α are default parameters, which usually do not have to be changed. However, when necessary, they can be adjusted by the user. Tweaking these parameters is usually straightforward. In case the mesh is too distorted close to a boundary, its α value has to be increased. On the other hand, when the mesh distortion is in between the two boundaries, the α values have to be lowered.

5.3 Boundary Node Rotations

As mentioned before, it is important to include boundary node rotations in the IDW interpolation method, in order to maintain an orthogonal mesh close to the moving surface. However, including rotations causes additional difficulties which do not occur when only translations are considered. First of all, at each boundary node only the displacement is given, hence the rotations will somehow have to be determined. Secondly a decision has to be made about how these rotations have to be interpolated to the volume mesh. The following paragraphs will explain how these issues have been tackled.

5.3.1 Rotation Methods

In general two ways can be found to represent rotations in mesh deformation: matrix and quaternion. The following two sections give a brief description of both methods.

Rotation Matrix

The matrix representation is mainly used in combination with Euler angles. Euler angles represent the rotation around three Euclidean axes, x, y and z, with angles θ_x, θ_y and θ_z respectively. Each Euler angle rotation can be expressed by one rotation matrix as follows:

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \; M_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \; M_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

The final rotation matrix representing the rotation is a multiplication of the three matrices. Here it is important to notice that the order of the rotations matters, because rotations do not commute. This representation by Euler angles has historically been the most popular.

However, Euler angles can cause *Gimbal lock*, the loss of one rotational degree of freedom. This is caused by ignoring the interactions between the different rotations [61].

Alternatively rotation matrices can also use the axis angle method, where a rotation angle θ and a unit rotation axis $\boldsymbol{n} = [n_x\ n_y\ n_z]^T$ are defined. The rotation matrix is then given by

$$M = \begin{bmatrix} \cos\theta + n_x^2(1-\cos\theta) & n_x n_y(1-\cos\theta) - n_z\sin\theta & n_x n_z(1-\cos\theta) + n_y\sin\theta \\ n_y n_x(1-\cos\theta) + n_z\sin\theta & \cos\theta + n_y^2(1-\cos\theta) & n_y n_z(1-\cos\theta) - n_x\sin\theta \\ n_z n_x(1-\cos\theta) - n_y\sin\theta & n_z n_y(1-\cos\theta) + n_x\sin\theta & \cos\theta + n_z^2(1-\cos\theta) \end{bmatrix} \tag{5.11}$$

Rotation Quaternion

For the purpose of interpolation of rotations, quaternions form a better alternative, both in terms of computational resources and the proper treatment of rotation information (safe from Gimbal lock). Thanks to their powerful properties, quaternions are often used in computer graphics and spacecraft navigation. From a mathematical perspective, quaternions are hypercomplex numbers, with one real and three imaginary terms:

$$Q = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k, \quad q_\alpha \in \mathbb{R} \tag{5.12}$$

where

$$\begin{aligned} ii = jj = kk = -1, \\ ij = -ji = k, \\ jk = -kj = i, \\ ki = -ik = j. \end{aligned}$$

The real number represents the angle of rotation and the three imaginary numbers represent the axis of rotation. As such a quaternion Q can be represented by one scalar and one Cartesian vector:

$$Q = [r, \boldsymbol{v}],\ r = q_0,\ \boldsymbol{v} = [q_1\ q_2\ q_3]. \tag{5.13}$$

The scalar and vector component are given by

$$r = \cos\frac{\alpha}{2}, \tag{5.14}$$

$$\boldsymbol{v} = \boldsymbol{a} \cdot \sin\frac{\alpha}{2}, \tag{5.15}$$

where α is the rotation angle and $\boldsymbol{a}$ is the unit axis of rotation. For clarity, the most relevant properties of quaternions are listed here. However for a more complete description of quaternions the reader is referred to [61] and [2].

- Associative: $(Q_1 \cdot Q_2) \cdot Q_3 = Q_1 \cdot (Q_2 \cdot Q_3)$,
- Distributive: $Q_1 \cdot (Q_2 + Q_3) = Q_1 \cdot Q_2 + Q_1 \cdot Q_3$,

- Not commutative: $Q_1 \cdot Q_2 \neq Q_2 \cdot Q_1$,
- Conjugate of a quaternion: $Q^* = q_0 - q_1 \cdot i - q_2 \cdot j - q_3 \cdot k$,
- Magnitude of a quaternion: $\|Q\| = \sqrt{Q \cdot Q^*} = \sqrt{q_0^2 + q_1^2 + q_3^2 + q_4^2}$,
- Inverse of a quaternion: $Q^{-1} = Q^*/(Q \cdot Q^*)$,
- Unit quaternion: $\|Q\| = 1$,
- For unit quaternion: $Q^{-1} = Q^*$.

In order to find the new position of point $\boldsymbol{p}$ after rotation, with quaternion algebra, first the point's position vector $\boldsymbol{p}$ is converted into a quaternion $P = [0, \boldsymbol{p}]$. Then the rotated position P_r is found with

$$P_r = Q \cdot P \cdot Q^{-1}. \tag{5.16}$$

Since a rotation quaternion is always a unit quaternion, the property $Q^{-1} = Q^*$ can be used

$$P_r = Q \cdot P \cdot Q^*. \tag{5.17}$$

Here the multiplication of quaternions is given by

$$Q_1 \cdot Q_2 = [r_1,\ \boldsymbol{v}_1] \cdot [r_2,\ \boldsymbol{v}_2] = [r_1 r_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2,\ r_1 \boldsymbol{v}_2 + r_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2] \tag{5.18}$$

In case of multiple consecutive rotations, the rotations can be combined into one rotation quaternion

$$Q = Q_n \cdot Q_{n-1} \cdots Q_2 \cdot Q_1, \tag{5.19}$$

where the number also indicates the order of rotation.

5.3.2 Determining the Rotation of a Boundary Node

For each boundary node, only the position vector $\boldsymbol{x}$ and the displacement vector $\boldsymbol{u}$ are known. In order to include rotations in IDW mesh deformation, the rotation for each boundary node will first have to be determined. Based on the description in the previous section, it is clear that quaternions have advantages over rotations matrices. Therefore it is chosen to represent the boundary rotations with quaternions. The goal is to find a translation vector $\boldsymbol{t}$ and a rotation quaternion R for each boundary node, such that

$$X_d = R \cdot X_u \cdot R^* + T, \tag{5.20}$$

where the subscripts u and d indicate the undeformed and deformed states respectively and X_u, X_d and T are quaternions representing position and translation vectors as follows

$$\begin{aligned} X_u &= [0, \boldsymbol{x}], \\ X_d &= [0, \boldsymbol{x} + \boldsymbol{u}], \\ T &= [0, \boldsymbol{t}]. \end{aligned}$$

In order to determine the boundary node rotation quaternions, a set of neighbouring nodes of each boundary nodes will have to be considered. One could opt to use the nodes that are directly connected to a certain boundary node [58]. Alternatively one can use the cell centres of the neighbouring faces [42]. However in this thesis, it was chosen to first determine the quaternions of the faces, according to the work of Kovalev [30], and then interpolate to the face's nodes. Which method one chooses mainly depends on the specific code architecture of the mesh at hand. Since the method of Kovalev had already been used in FINE$^{\text{TM}}$/Open (see section 3.4.1), it was a straightforward choice.

The rotation of a boundary face is determined in three steps, as shown in figure 5.3. Let's consider the deformed and undeformed position of the face, as the input of the method. In the first step the face centres of both the undeformed and deformed face, are moved to the origin. Here, the face centre $\boldsymbol{c}$ is computed as the average of the position vectors $\boldsymbol{x}_j$ of all the corners of the face

$$\boldsymbol{c} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{x}_j, \tag{5.21}$$

where n is the number of face corners. The updated positions $\boldsymbol{x}'$ of the vertices, after the first step are computed as

$$\boldsymbol{x}'_u = \boldsymbol{x}_u - \boldsymbol{c}_u \tag{5.22}$$
$$\boldsymbol{x}'_d = \boldsymbol{x}_d - \boldsymbol{c}_d \tag{5.23}$$

In the second step the undeformed face is rotated such that its normal aligns with the normal of the deformed face. Therefore, first the unit normal $\boldsymbol{n}$ to a face is computed as the average of the unit normals to two consecutive face corner position vectors

$$\boldsymbol{n} = \frac{1}{n}\sum_{j=1}^{n} \frac{\boldsymbol{x}_j \times \boldsymbol{x}_{j+1}}{\|\boldsymbol{x}_j \times \boldsymbol{x}_{j+1}\|}. \tag{5.24}$$

Note that $\boldsymbol{x}_{n+1} = \boldsymbol{x}_1$, because the nodes are numbered in a consecutive manner, as shown in figure 5.3. Now the first axis of rotation is computed as the unit vector which is perpendicular to both the deformed and undeformed normal

$$\boldsymbol{a}_1 = \frac{\boldsymbol{n}_u \times \boldsymbol{n}_d}{\|\boldsymbol{n}_u \times \boldsymbol{n}_d\|}. \tag{5.25}$$

The angle of the first rotation is equal to the angle between the deformed and undeformed normal, computed as

$$\alpha_1 = \arccos\left(\frac{\boldsymbol{n}_u \cdot \boldsymbol{n}_d}{\|\boldsymbol{n}_u\|\,\|\boldsymbol{n}_d\|}\right) \tag{5.26}$$

This unit axis of rotation $\boldsymbol{a}_1$ and angle of rotation α_1, are used to find the first rotation quaternion R_1 with equations 5.13 to 5.15. Now the updated position vectors $\boldsymbol{x}''_u$ of the undeformed face can be found by applying this rotation to all the face corners

$$X''_u = R_1 \cdot X'_u \cdot R_1^*, \tag{5.27}$$

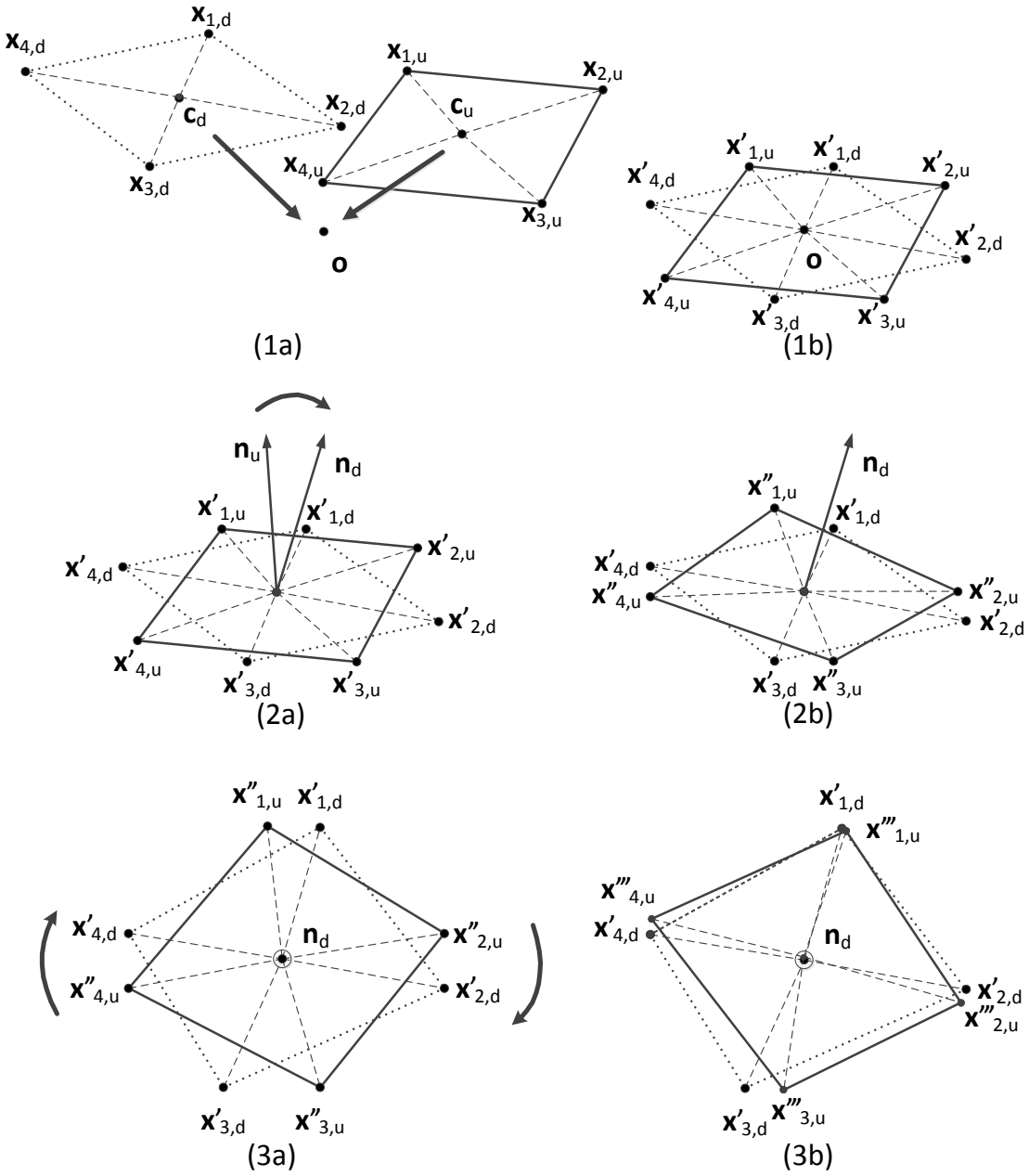


Figure 5.3: Three steps to find the rotation of a boundary face.

where $X''_u = [0,\ \boldsymbol{x}''_u]$ and $X'_u = [0,\ \boldsymbol{x}'_u]$.

The third and last step, rotates the undeformed face around the normal vector, such that the corners of the faces are aligned as well as possible. The axis of this rotation, is simply the unit normal to the deformed face

$$\boldsymbol{a}_2 = \frac{\boldsymbol{n}_d}{\|\boldsymbol{n}_d\|} \tag{5.28}$$

The angle of the second rotation, is given by the average angle between deformed and undeformed position vectors of the face corners

$$\alpha_2 = \frac{1}{n}\sum_{j=1}^{n} \arccos\left(\frac{\boldsymbol{x}_{j,u}\cdot\boldsymbol{x}_{j,d}}{\|\boldsymbol{x}_{j,u}\|\cdot\|\boldsymbol{x}_{j,d}\|}\right). \tag{5.29}$$

This unit axis of rotation $\boldsymbol{a}_2$ and rotation angle α_2 determine the second rotation quaternion R_2, by equations 5.13 to 5.15. The final rotation quaternion for the face is then given by R

$$R = R_2 R_1 \tag{5.30}$$

Once the rotation quaternions of the boundary mesh faces are known, the rotation quaternions of the boundary mesh nodes can be computed via interpolation. This interpolation is done with spherical linear interpolation or SLERP. The spherical linear interpolation of two quaternions is computed as [61]

$$Q_{\text{SLERP}} = \frac{\sin[(1-w)\omega]}{\sin\omega}Q_1 + \frac{\sin[w\omega]}{\sin\omega}Q_2, \tag{5.31}$$

where $0 \leq w \leq 1$ is the weight for the second quaternion, and ω is the angle between the two quaternions computed as

$$\omega = \arccos(Q_1 \cdot Q_2) \tag{5.32}$$

To compute the spherical linear interpolation Q_{SLERP_n} of multiple quaternions $Q_1, Q_2, ..., Q_n$, the following procedure can be applied [30]

$$Q_{\text{SLERP}_2} = \text{SLERP}\left(Q_1, Q_2, \frac{w_2}{w_1+w_2}\right) \tag{5.33}$$

$$Q_{\text{SLERP}_3} = \text{SLERP}\left(Q_{\text{SLERP}_2}, Q_3, \frac{w_3}{w_1+w_2+w_3}\right) \tag{5.34}$$

$$\vdots$$

$$Q_{\text{SLERP}_n} = \text{SLERP}\left(Q_{\text{SLERP}_{n-1}}, Q_n, \frac{w_n}{\sum_{j=1}^{n} w_j}\right) \tag{5.35}$$

For the purpose of computing the rotation quaternion of a node, the above procedure is applied using the rotation quaternions of all n faces connected to this node, with equal weights $w_j = \frac{1}{n}$.

Once the rotation quaternion R for all boundary nodes is known, the translation vector $\boldsymbol{t}$ can be computed with equation 5.20.

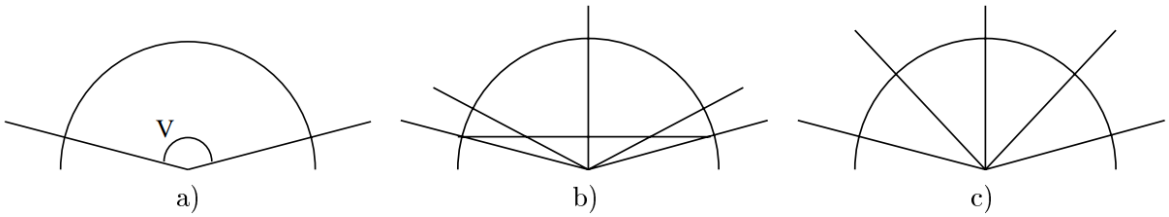


Figure 5.4: Difference between LERP and SLERP. a) The angle v is interpolated in three steps. b) LERP: the secant is divided in equal pieces, resulting in small angles at the sides and large angles in the middle. c) SLERP: interpolation with equal angles. [12]

5.3.3 Interpolation of Rotations to Volume Mesh

IDW will be used to interpolate both the translation and rotation component into the volume mesh. For the translation component this is straightforward, following the general IDW method

$$\boldsymbol{t}(\boldsymbol{x}) = \frac{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})\boldsymbol{t}_b}{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})}, \tag{5.36}$$

where $\boldsymbol{t}(\boldsymbol{x})$ is the translation vector of the volume mesh node at position $\boldsymbol{x}$, n_b is the number of boundary nodes, $w_b(\boldsymbol{x})$ is the weighting function and $\boldsymbol{t}_b$ is the translation vector of boundary node b. For the rotation component however, several methods can be applied. These methods are introduced in the following paragraphs.

Linear Interpolation of Quaternions

The first option is to to apply the same procedure to the rotation component as the one used for the translation component

$$R(\boldsymbol{x}) = \frac{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})R_b}{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})}, \tag{5.37}$$

where R_b is the rotation quaternion of boundary node b. Note that the resulting rotation quaternion is not necessarily a unit quaternion, and therefore it has to be normalised. The displacement $\boldsymbol{u}$ of a volume mesh node at position $\boldsymbol{x}$ can then be computed as

$$U(\boldsymbol{x}) = \hat{R}(\boldsymbol{x})X\hat{R}^*(\boldsymbol{x}) - X + T(\boldsymbol{x}), \tag{5.38}$$

where $U = [0,\ \boldsymbol{u}]$, $X = [0,\ \boldsymbol{x}]$, $T = [0,\ \boldsymbol{t}]$ and $\hat{R}(\boldsymbol{x})$ is the normalized quaternion $R(\boldsymbol{x})$.

The above described method, which is called linear interpolation or LERP, has one main limitation. This limitation is the fact that the interpolation has a varying velocity, which is larger in the middle than at the sides. This is due to the fact that a simple linear interpolation yields a secant between two quaternions, as illustrated in figure 5.4 (b).

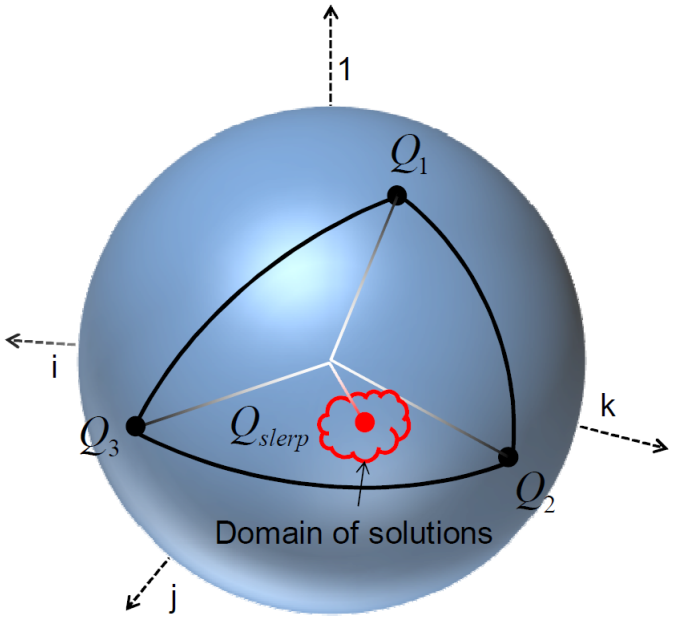


Figure 5.5: Uncertainty of SLERP for interpolation of multiple quaternions. [42].

Spherical Linear Interpolation of Quaternions

To resolve the varying velocity problem of LERP, the interpolation of the rotation quaternions can be done with SLERP, as described in equations 5.31 to 5.35. The used weights for SLERP are equal to the IDW weights $w(\boldsymbol{x})$ and n is the total number of boundary nodes. SLERP of two quaternions yields the shortest route along a great arc on the unit sphere. Furthermore the angular velocity of the interpolation is constant, as illustrated in figure 5.4 (c) [12]. The disadvantage of SLERP, is that for multiple quaternion interpolation SLERP is not uniquely defined. When the interpolation order changes, the results will also change. This uncertainty in the result is illustrated in figure 5.5. Once the translation component has been computed with equation 5.36, and the rotation component has been computed with SLERP, the displacement can again be computed with equation 5.38.

Linear Interpolation of Logarithm of Quaternion

The uncertainty in the interpolation result of multiple quaternions is due to the non-commutative property of quaternions in multiplication [42]. This can be eliminated by calculating in Lie algebra space, using the exponential map [23]. If a quaternion is given by equations 5.13 to 5.15, then the logarithm of this quaternion is given by [42]

$$ln(R) = [0, \frac{\alpha}{2}\boldsymbol{a}], \tag{5.39}$$

where $\boldsymbol{a}$ is the unit axis of rotation and α is the angle of rotation. It can be seen that the logarithm of the quaternion is actually a three-dimensional vector. In [42] it is shown that the exponential map of SLERP, can be approximated with a LERP of the logarithms of the

quaternions. Therefore the interpolation of rotations could be done as follows

$$\ln(R(\boldsymbol{x})) = \frac{\sum_{b=1}^{n_b} w_b(\boldsymbol{x}) \ln(R_b)}{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})}, \tag{5.40}$$

This interpolated exponential map can then be converted back to a quaternion as follows

$$R = [\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}\boldsymbol{a}], \tag{5.41}$$

$$\frac{\alpha}{2} = \|ln(R)\|, \tag{5.42}$$

$$\boldsymbol{a} = \frac{ln(R)}{\|ln(R)\|}. \tag{5.43}$$

Since the summation of vectors is commutative, this interpolated quaternion is uniquely determined.

Linear Interpolation of Displacement Field

Finally it is also possible to avoid interpolating rotations, by first determining the displacement at a certain position due to the rotation, and then interpolating this displacement. Let us first consider the displacement field $\boldsymbol{s}(\boldsymbol{x})$ of a boundary node b

$$[0, \boldsymbol{s}_b(\boldsymbol{x})] = R_b[0, \boldsymbol{x}]R_b^* - [0, \boldsymbol{x}] + [0, \boldsymbol{t}_b]. \tag{5.44}$$

As proposed in the IDW method of Luke et al [38], this displacement field can be interpolated with IDW

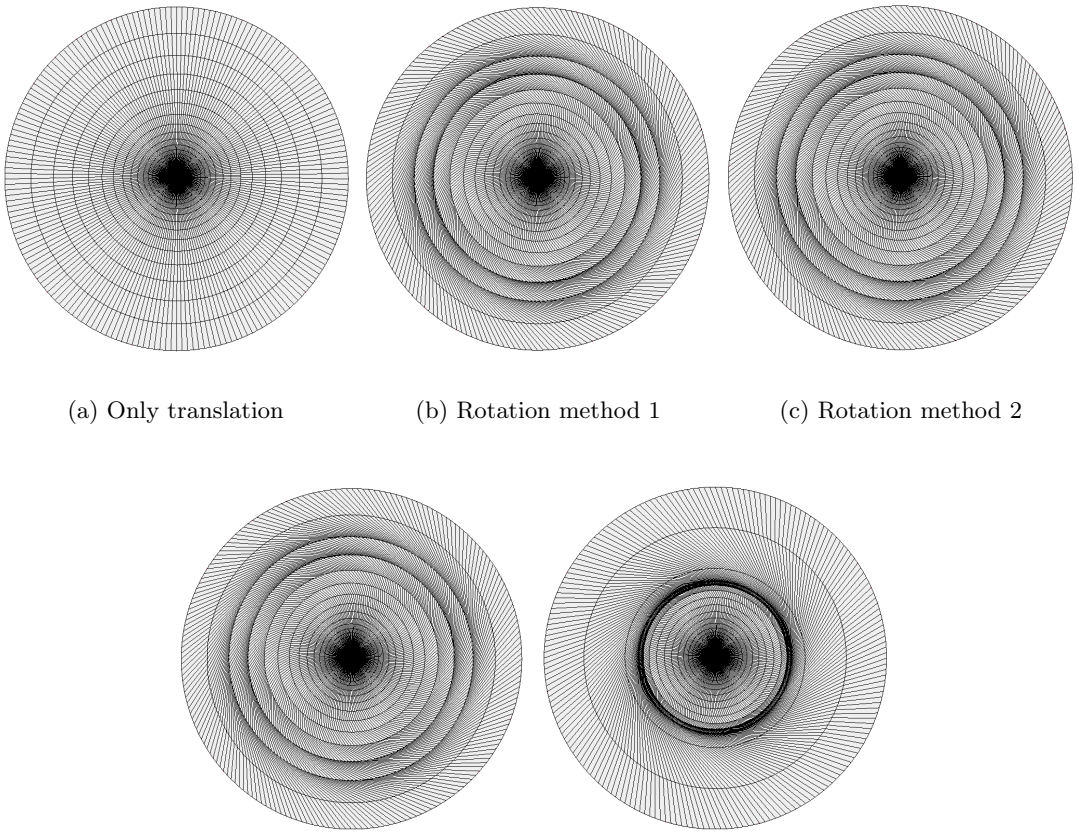
$$\boldsymbol{u}(\boldsymbol{x}) = \frac{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})\boldsymbol{s}_b(\boldsymbol{x})}{\sum_{b=1}^{n_b} w_b(\boldsymbol{x})}. \tag{5.45}$$

5.3.4 Results of Mesh Deformation with Rotations

In total four different methods for the interpolation have been introduced

1. LERP of quaternions,
2. SLERP of quaternions,
3. LERP of logarithm of quaternions,
4. LERP of displacement field.

In this section the performance of these four rotation interpolation methods is compared to each other. This is done by using five different test cases, as described in chapter 4. In all these test cases relatively large deformations are applied in one step.



(a) Only translation (b) Rotation method 1 (c) Rotation method 2

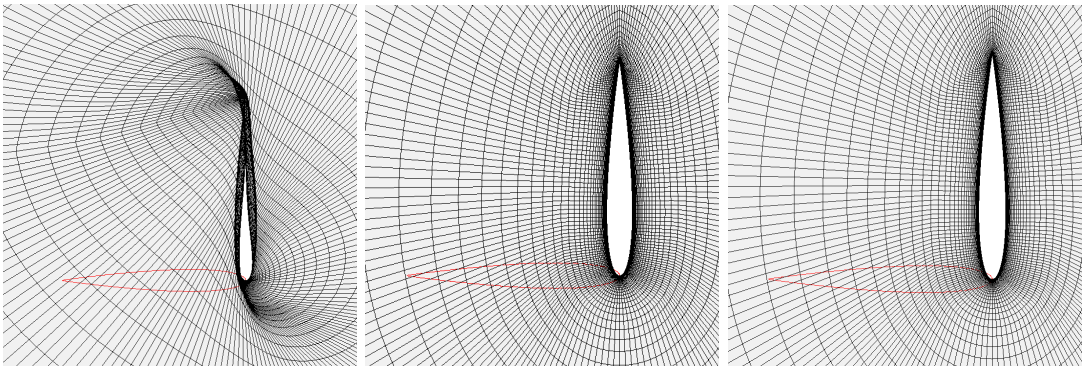
(d) Rotation method 3 (e) Rotation method 4

Figure 5.6: View of the mesh of NACA 0012 airfoil after 90° rotation.

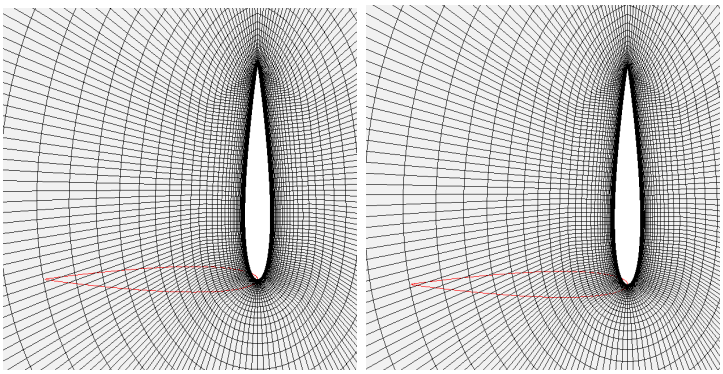
First let us have a look at the resulting mesh qualities. The orthogonality and skewness are shown in figures 5.10 and 5.11 respectively. Remember that a perfect orthogonality value is 90°, whereas a perfect skewness value is zero. Then it becomes clear that method 4 yields the highest quality for all test cases except for the NACA airfoil rotation. The three other methods yield results that are very similar to each other in terms of quality. Except for the rotor 67 test case, where method 3 performs significantly worse than method 1 and 2. This might indicate that the logarithm approximation used in method 3 is not always accurate.

For the 2D test cases the mesh quality can also easily be assessed by looking at the mesh itself, as shown in figures 5.6 to 5.8. Figure 5.6(a), 5.7(a) and 5.8(a) show the resulting mesh, when the basic implementation, without rotations, is used. It is clear that when rotations are not included in the interpolation, the mesh is not orthogonal at the moving boundary.

Visually, most of the resulting meshes using rotation methods 1-4, look very similar. The one exception is rotation method 4 in the NACA test case, as shown in figure 5.6(e). This difference is due to the large rotation angle of 90°. When interpolating the displacement due to rotation, instead of the rotation itself, the resulting position will be along a straight

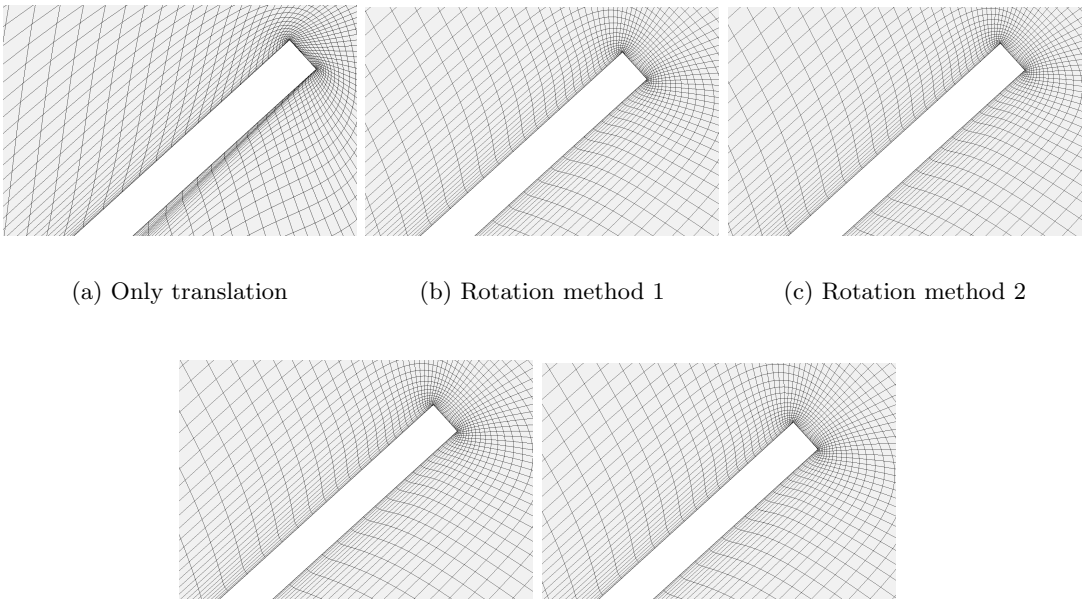


(a) Only translation (b) Rotation method 1 (c) Rotation method 2



(d) Rotation method 3 (e) Rotation method 4

Figure 5.7: Detailed view of the mesh of NACA 0012 airfoil after 90° rotation.



(a) Only translation (b) Rotation method 1 (c) Rotation method 2

(d) Rotation method 3 (e) Rotation method 4

Figure 5.8: Detailed view of the mesh of the vortex induced beam vibration test case after deformation.

line, instead of along an arc. This is illustrated in figure 5.9, where an interpolation is done between a 0° and a 90° rotation with equal weights $w = 0.5$. In case of interpolation with method 4, the resulting position is in in the middle of the straight line connecting the two positions. This will have the resulting effect that the mesh seems to be shrinking. For all other interpolation methods, the resulting position is always on the arc connecting the two positions, which yields a better result for large rotations.

When looking at the CPU times for the different methods in figure 5.12, method 2 jumps out as the slowest method. This is due to the fact that SLERP requires several trigonometric function evaluations which are expensive operations. Method 1 and 3 are the fastest, because

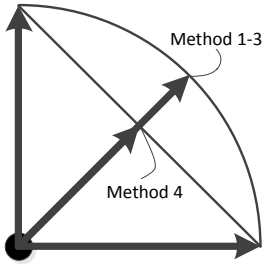


Figure 5.9: Comparison of the different methods for the interpolation between a 0° and 90° rotation

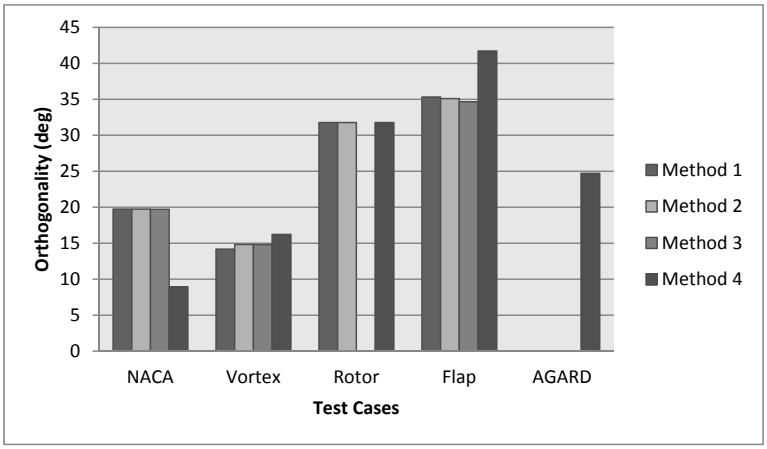


Figure 5.10: Comparison of orthogonality for the four rotation interpolation methods.

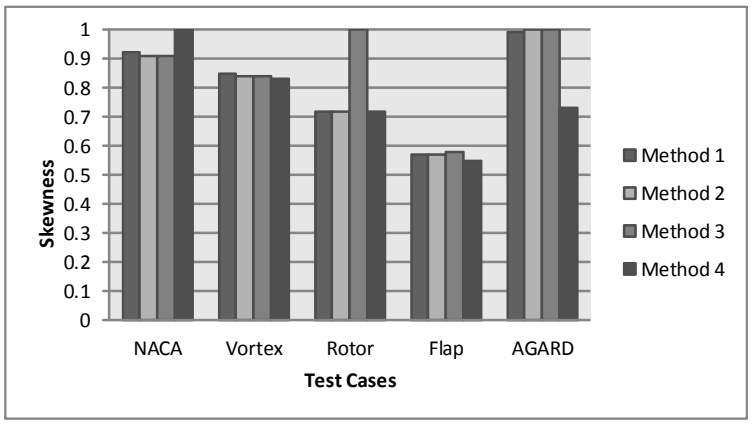


Figure 5.11: Comparison of skewness for the four rotation interpolation methods.

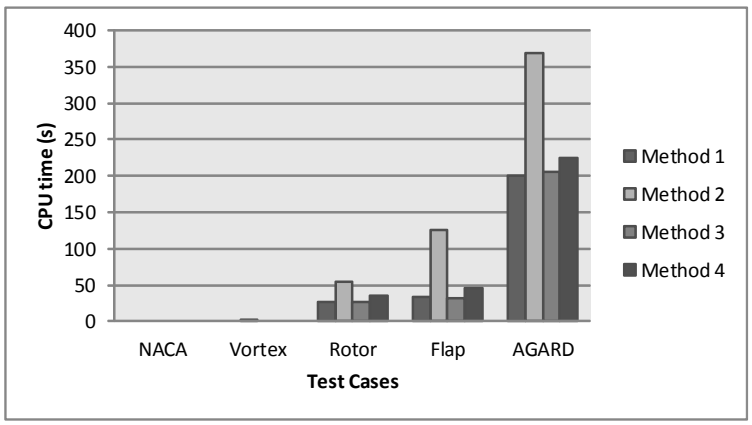


Figure 5.12: Comparison of CPU time for the four rotation interpolation methods.

only summations are required. Finally method 4 is slightly slower than 1 and 3, due to the fact that first the rotation has to be applied to a vector, which requires some extra multiplications.

In conclusion it can be stated that for large pure rotations, such as the NACA test case, method 1 or 3 should be used, since they result in high quality meshes for low computational cost. For more general deformation test cases, such as the bending of a wing, it would be advised to use method 4 for the best possible mesh quality. However if the deformations are small and CPU time is more important, method 1 could be used. In general method 2 should be avoided, due to the high computational cost.

5.4 Sliding Boundary Nodes

So far, two types of boundary conditions, moving and fixed, have been used. In this section a third type of boundary condition is introduced, namely the sliding boundary condition. This boundary condition allows for nodes to move, while remaining on the specified boundary. This is particularly interesting for deformation cases where the gap between the moving and fixed boundary is small. The fixed boundary condition can then be replaced with a sliding boundary condition, such that the sliding boundary can follow the displacement of the moving boundary. One example of a fluid-structure interaction problem where this condition occurs, are turbo-machinery cases, such as the Rotor 67 test case from chapter 4.

5.4.1 Sliding Boundary Strategy

In structural analogy mesh deformation methods, such as the elastic and spring analogy, sliding boundary nodes are often incorporated by means of a Neumann boundary condition [28][25], where the motion is restrained to be tangential to the undeformed domain. This is fairly straightforward to implement in all mesh strategies where discrete mesh connectivity methods are used to solve a system of equations. However, since IDW is a point-by-point method, a different strategy was chosen for the sliding boundaries. This section first explains the foundation of the method and then introduces further improvements and adaptations to increase the robustness.

The chosen sliding boundary strategy can easily be explained by means of figure 5.13. The figure shows a schematic deformation case, where a rectangular block is rotated. The top and bottom boundaries are fixed boundaries, whereas the left and right boundaries are sliding boundaries. Three main steps can be distinguished in the sliding boundary strategy

1. Compute the displacement of the sliding boundary nodes using IDW interpolation. Here, the data points for the interpolation are the fixed and moving boundary nodes, i.e. $n_b = n_{fix} + n_{mov}$. (See figure 5.13 (3))
2. Snap the sliding boundary nodes to the closest point on the domain boundary. (See figure 5.13 (4))

3. Compute the displacement of the inner mesh nodes using IDW interpolation. Here, all boundary nodes, including the sliding ones, are used as data points in the interpolation, i.e. $n_b = n_{mov} + n_{slide} + n_{fix}$. (See figure 5.13 (5))

This strategy treats the sliding boundary nodes as a mix between inner mesh nodes and boundary nodes. First they are inner nodes, for which the displacement has to be computed. Then, once the displacement is known, they become boundary nodes. As boundary nodes, they will be included as data points in the interpolation, just like the fixed and moving boundary nodes.

This strategy is simple and does not add much complexity to the IDW mesh deformation method. However, a couple of considerations have to be made. First of all, an extra parameter has to be accounted for. Just like α_{mov} and α_{fix}, as introduced in section 5.2.2, a parameter α_{slide}, will be needed. Usually the value of α_{slide} can be set equal to 0.1, just as for the moving boundary. However, for specific test cases the user might choose to adjust this when necessary. These adjustments can be done in the same way as described in section 5.2.2 for α_{mov} and α_{fix}. When the user wishes a higher mesh quality close the sliding boundary α_{slide} can be increased. On the other hand when the user wants to make sure that the area where the deformation is absorbed is as large as possible, α_{slide} can be decreased such that the mesh close to the sliding boundary will also deform.

The most difficult part of the sliding strategy, however, would be to implement the snapping method, to make sure the sliding nodes remain on the domain boundary surface. Luckily, in Hexpress^{TM}, such a function was already available. Normally this function is used in the process to generate unstructured meshes. However, it can just as easily be used during the mesh deformation process. The used function finds the closest point on the domain boundary, using a triangulation to define the domain geometry [30].

Next, it is important to consider the domain topology when snapping nodes to the domain. The domain topology consists of three types of elements, namely vertices, edges and faces. Each boundary node in the mesh is linked to one of these elements, i.e. a boundary node is either a node on a topological vertex, a node on a topological edge or a node on a topological face. Since it is important to preserve the domain topology, all mesh nodes should always be snapped onto the element that they are connected to. This means, for example, that a boundary node that was originally on a domain edge, will remain on that specific edge. However, for certain CFD domains, this gives rise to problems. Look for example at the domain of the Rotor 67 test case in figure 5.14(a). Here you can see that the shroud of the domain actually consists of seven topological faces. The same can be seen in the domain of the elastic flap test case in figure 5.15(a), where both the top and bottom of the channel consist of two topological faces and the left face is composed of three topological faces. Of course, when using the sliding boundary node condition, it is preferable to treat these examples as one face, seeing as the division is merely a fictional one, that was introduced for meshing purposes.

To resolve these topological issues, the concept of topological sliding face groups, is introduced. All faces in such a sliding face group will consequently be treated as if they were just one

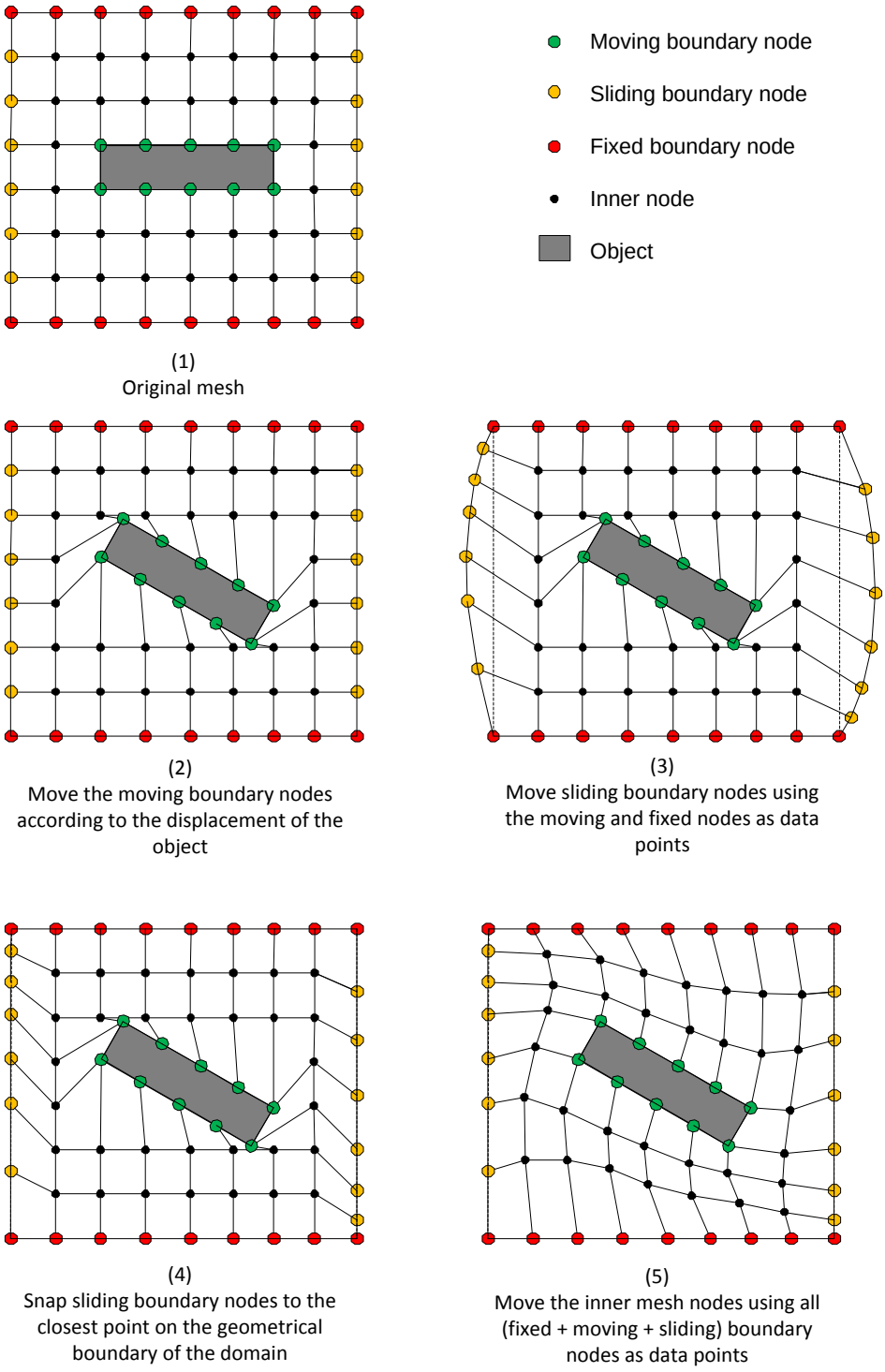
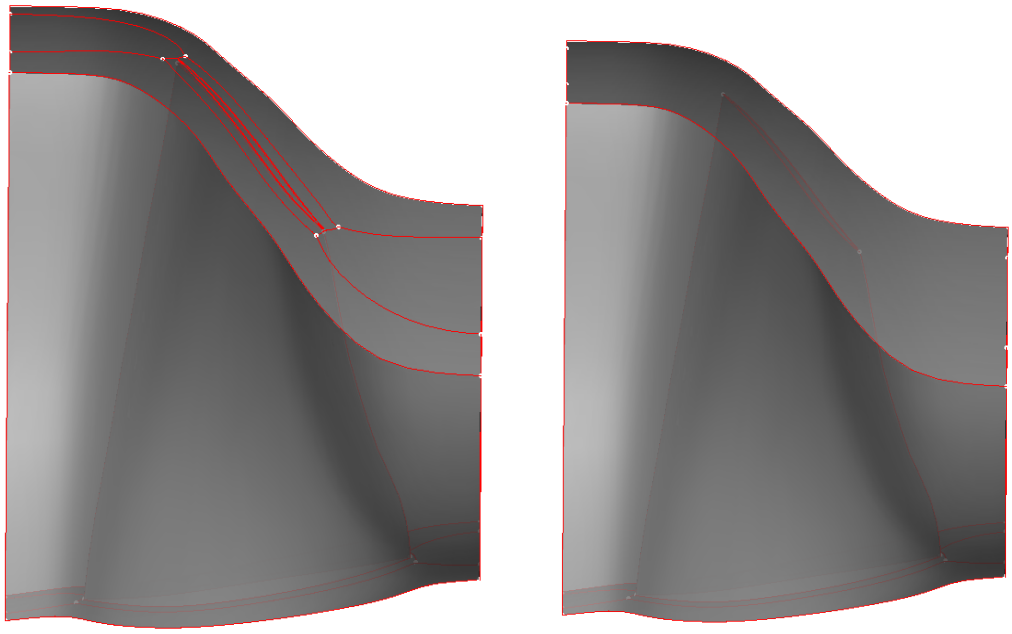


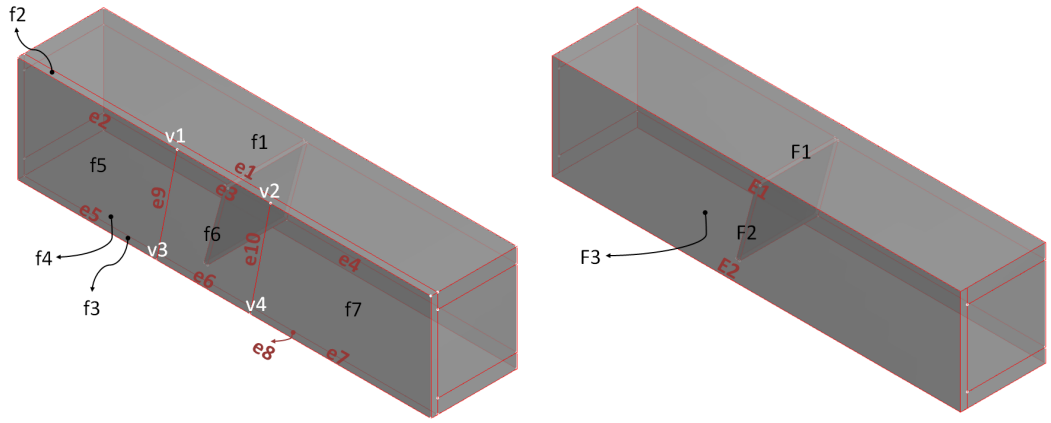
Figure 5.13: Schematic overview of sliding boundary strategy



(a) Domain definition of the rotor 67 test case.

(b) Domain definition of the rotor 67 test case after the definition of sliding groups.

Figure 5.14: Domain definition of rotor 67 test cases before and after the introduction of topological sliding face groups.



(a) Domain definition of the elastic flap test case.

(b) Domain definition of the elastic flap test case after the definition of sliding groups.

Figure 5.15: Domain definition of the elastic flap test case before and after the introduction of topological sliding face groups.

face. For example, in the rotor 67 test case, the user will appoint one sliding face group, consisting of the seven faces in the shroud. For the elastic flap test case, the user will assign three sliding face groups:

- Topological sliding face group 1 = Top: faces 1 and 2.
- Topological sliding face group 2 = Bottom: faces 3 and 4.
- Topological sliding face group 3 = Left: faces 5 to 7.

Inside the mesh deformation algorithm the topological sliding face groups are then used to redefine the domain topology. Four changes are made

1. All faces in one group are treated as if they were one face.
2. All edges that are embedded in one sliding face group are ignored.
3. All edges that form a boundary between the same sliding face groups are treated as if they were one edge, i.e. they form one sliding edge group.
4. All topological vertices that are embedded within one sliding face group or within one sliding edge group are ignored.

Concretely, this can be illustrated by means of the elastic flap domain. In figure 5.15(a) the original domain definition is shown, whereas in figure 5.15(b) the redefined domain, after the introduction of sliding groups, is illustrated. Note that lower-case symbols are used for the original domain and upper-case symbols are used for the new domain. In the first step the grouped faces are redefined as one face, i.e. $F1 = f1 + f2$, $F2 = f5 + f6 + f7$ and $F3 = f3 + f4$. Then in the second step, edges $e1$, $e8$, $e9$ and $e10$ are removed, because they are embedded in one face group. Next, in step 3, the remaining sliding edges are grouped if they form a boundary between two sliding face groups. This means that $E1 = e2 + e3 + e4$ (boundary between $F1$ and $F2$) and $E2 = e5 + e6 + e7$ (boundary between $F2$ and $F3$). Finally, in step 4 the topological vertices within one group are ignored. This means that $v1$ and $v2$ can be removed, because they lie within edge $E1$ and $v3$ and $v4$ can be removed because they lie within edge $E2$. The same process can be applied to the domain of the rotor 67 test case, to obtain the new domain as in figure 5.14(b).

The final consideration that has to be made when using sliding boundaries, is the fact that multiple sliding boundaries can be used, as is the case for the elastic flap. This is particularly a problem when these sliding faces border to each other. The reason being, that only the moving and fixed nodes serve as data points in the IDW interpolation to update the positions of the sliding nodes. This means that the nodes on one sliding face receive no displacement information from the nodes on other sliding faces. As such, it is as if the boundaries between the different sliding faces are not there. This can lead to problems, because the IDW deformation method with rotations is not extremum conserving. Consequently sliding nodes can easily move past the boundaries of their plane. When these nodes are then snapped back

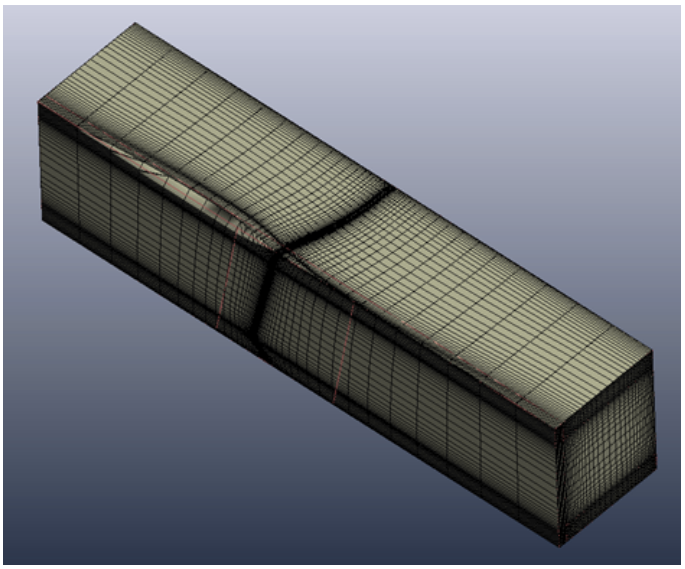


Figure 5.16: Poor quality mes, resulting from improper treatment of sliding boundary.

to their topological surfaces, a bunch of them will be snapped on top of each other at the boundary of the topological surface. On the other hand the sliding boundary nodes, could also move away from the boundaries of their topological surfaces, such that large spacings in the mesh are created. An example of a mesh where multiple sliding face were used is shown in figure 5.16.

The problem, caused by using multiple sliding nodes, can most clearly be illustrated, by using the schematic example from figure 5.13 again. This time, it is chosen to use a sliding boundary condition for the whole outer boundary, as indicated in figure 5.17. The block receives the same pure rotation as before. When applying IDW interpolation (with rotations) to the sliding boundary, the whole outer boundary will actually rotate with the same angle as the block. This is due to the fact that all the data points for the interpolation are moving boundary nodes, with the same rotation component. Now, when the sliding boundary nodes are snapped back to their respective topological elements, a poor distribution of the sliding boundary nodes results. Several corner nodes are even snapped on top of each other in the corners. This is obviously not an acceptable result. Therefore it is essential that when IDW is used to update the sliding boundary, the appropriate data points are selected. These data points should also include sliding boundary nodes, such that the boundaries of the domain can be preserved.

In order to resolve the issues in test cases with multiple sliding faces, the original sliding strategy is adjusted. The new strategy splits the computation of the sliding boundary nodes in two phases. First the sliding edges are updated and then the sliding faces are updated. When updating the sliding edge nodes all boundary nodes, except for the sliding nodes on its neighbouring faces, are included as data points. Due to the fact that the sliding node displacements are not known yet, the included sliding data points have a fictional zero-displacement. The reason why the sliding nodes on neighbouring faces are left out as data points is because they would restrain the motion of the sliding edge.

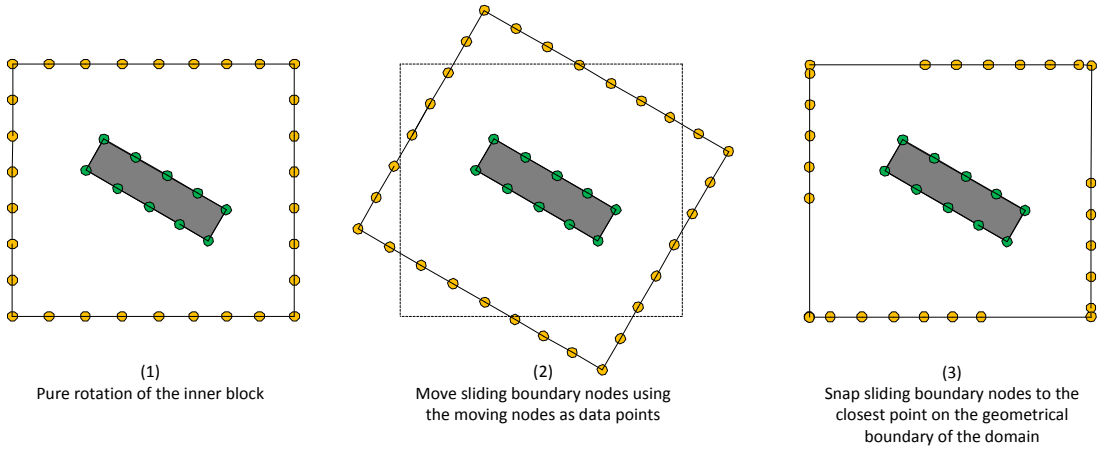


Figure 5.17: Schematic example of a poor quality mesh deformation when choosing the sliding boundary condition for the whole outer boundary.

In the next step, when the nodes on topological sliding faces are updated, the data points for the IDW interpolation function include all moving and fixed boundary nodes and the sliding nodes on edges. However since the amount of nodes on an edge is remarkably lower than the amount of nodes on a face, a compensation in the weighting function has to be done. A simple adjustment is made, where the weighting function for the nodes on sliding edges is simply multiplied by a factor. This factor is equal to the amount of nodes on sliding faces divided by the amount of nodes on sliding edges. A final structured overview of the sliding strategy is given below

1. Adjust the domain topology.
2. Compute the displacement of nodes on topological sliding edges with IDW interpolation. The included data points are
   (a) all moving boundary nodes,
   (b) all fixed boundary nodes,
   (c) all sliding boundary nodes, except for the nodes on neighbouring sliding faces. All sliding boundary data points have a zero-displacement.
3. Snap the displaced sliding nodes to the closest point on their respective topological edges.
4. Compute the displacement of nodes on sliding topological faces with IDW interpolation. The included data points are
   (a) all moving boundary nodes,
   (b) all fixed boundary nodes,
   (c) all sliding nodes on topological edges, with weighting function multiplied by factor $f = \frac{\text{number of nodes on sliding faces}}{\text{number of nodes on sliding edges}}$

5. Snap the displaced sliding nodes to the closest point on their respective topological faces.

6. Compute the displacement of inner mesh node displacements with IDW interpolation. The included data points are

   (a) all moving boundary nodes,

   (b) all fixed boundary nodes,

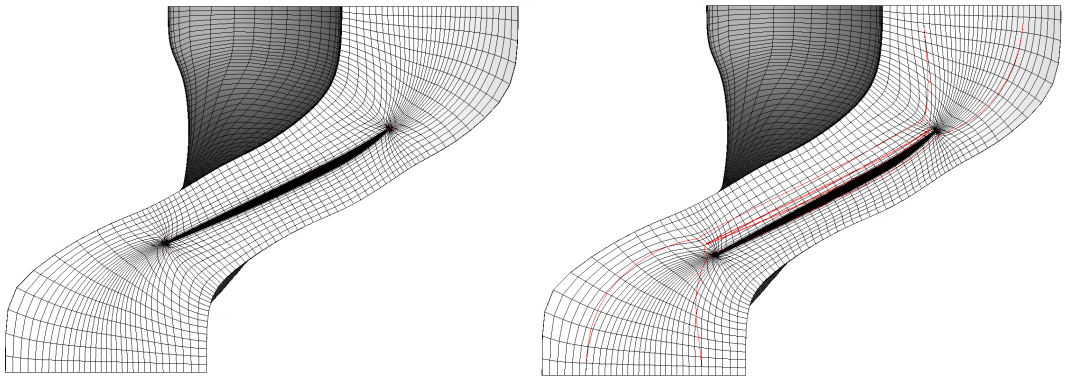
   (c) all sliding boundary nodes.

Note that when using IDW interpolation, it is very easy to adapt which nodes are included as data points. Since IDW interpolation consists of a simple summation over all data points, it is easy to leave certain data points out of the sum. On the other hand, for RBF interpolation this would be more difficult because it is not an explicit interpolation method. Depending on which data points have to be included, a new interpolation function will have to be created, which could be a costly procedure.

5.4.2 Sliding Boundary Results

In this section two different test cases for which sliding nodes are necessary, are discussed. The first one is the rotor 67 test case. Due to the very small gap between the rotor blade and the shroud of the turbo-engine, a sliding boundary condition has to be imposed at the shroud to maintain a good mesh quality. The second test case is the elastic flap. This test case has a relatively large deformation, within a confined space. Therefore, both the top, bottom and left panel of the channel have a sliding boundary condition.

Firstly, the rotor 67 test case is examined. In figure 5.18 the shroud of rotor 67 is shown. The left figure shows the resulting surface mesh without sliding nodes (i.e. the shroud remains fixed), and the right figure shows the resulting surface mesh with sliding nodes. The red lines indicate the original positions of the surfaces. When the sliding boundary condition is used, the shroud mesh has moved slightly to the right and to the bottom of the image. Even though this might seem to affect the quality of the shroud mesh in a negative sense, it is actually very beneficial for the volume mesh quality, as the shroud mesh has followed the movement of the blade (if you could see through the shroud in figure 5.18(b) you would see the blade tip right underneath it). The fact that the shroud mesh has followed the blade movement leads to a better orthogonality of the mesh between the shroud and the blade. This can clearly be seen when looking at a cross-section through the length of the blade, as in figure 5.19. Without sliding nodes (left), the cells between the shroud and the blade have become skewed. The blade even seems to cut through the mesh, as the mesh cannot follow the motion of the blade. When using the sliding boundary condition at the shroud (right), the cells between the blade and the shroud remain orthogonal. The good quality of the mesh can also be assessed quantitatively. The resulting mesh with sliding nodes has a minimal orthogonality of 33.77° and a maximal skewness of 0.72. This is only a small reduction compared to the original mesh quality with a minimal orthogonality of 31.99° and a maximal skewness of 0.72.



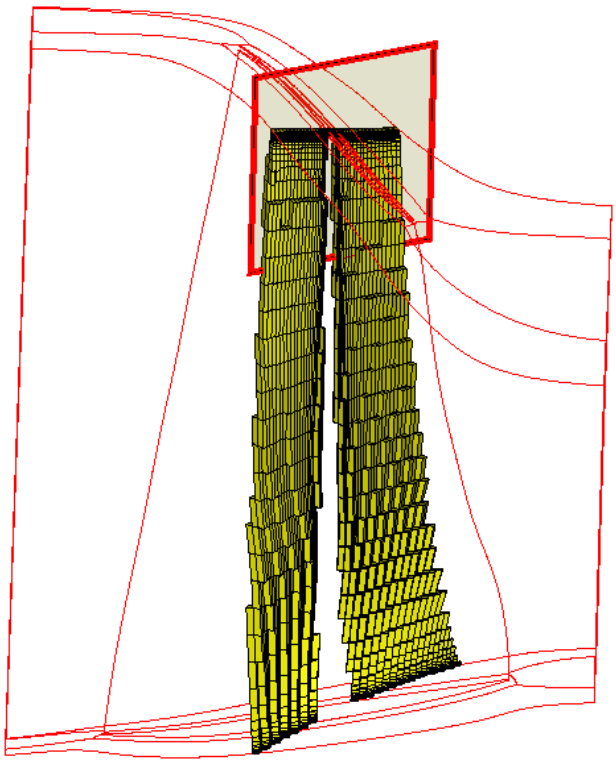
(a) Without sliding nodes

(b) With sliding nodes

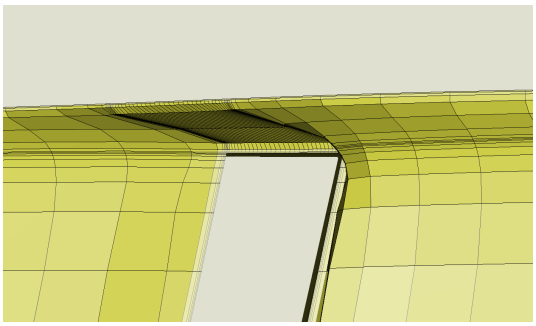
Figure 5.18: View of the shroud of rotor 67.

However, when no sliding nodes are used the mesh has negative cells, which means a minimal orthogonality of 0° and a maximum skewness of 1.

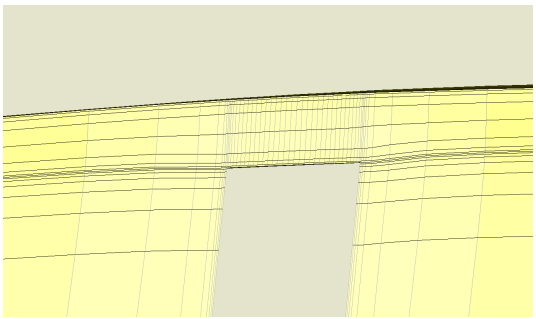
For the elastic flap test case the surface mesh is shown in figure 5.21. On the left, the surface mesh remains fixed (i.e. no sliding boundary), whereas on the right the surface mesh follows the deformation of the flap (i.e. with sliding nodes). Once again this leads to significant differences in deformed mesh quality, as can be seen from a horizontal cross-section through the middle of the channel in figure 5.20. Without sliding nodes, the mesh cannot follow the deformation of the flap, leading to negative cells. On the other hand, when sliding nodes are used, the mesh shows a smooth transition between the flap and the wall of the channel. Moreover, when sliding nodes are used a minimum orthogonality of 41.74° and a maximum skewness of 0.55 indicate that the mesh remained of high quality.



(a) View of the cutting plane position

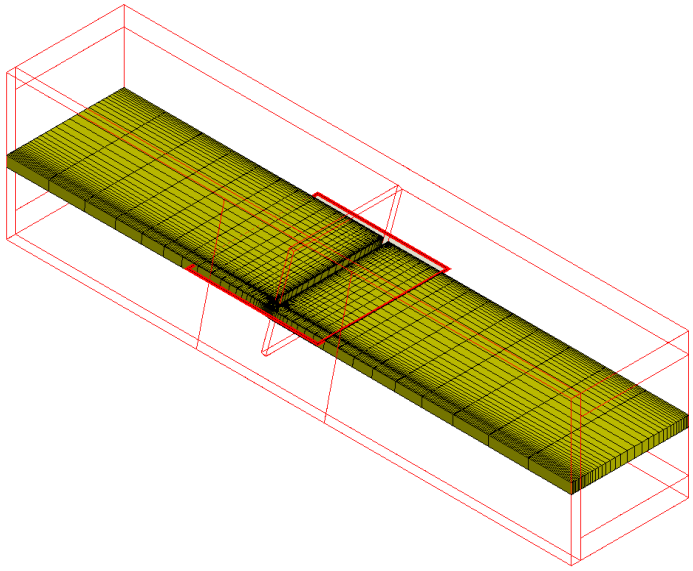


(b) Without sliding nodes

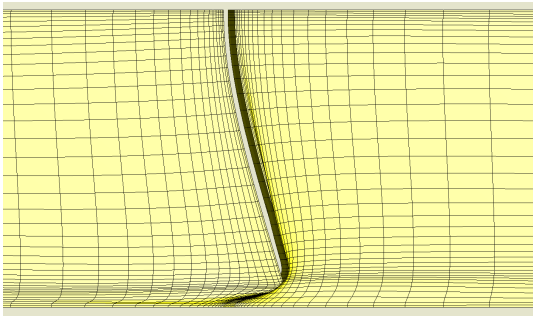


(c) With sliding nodes

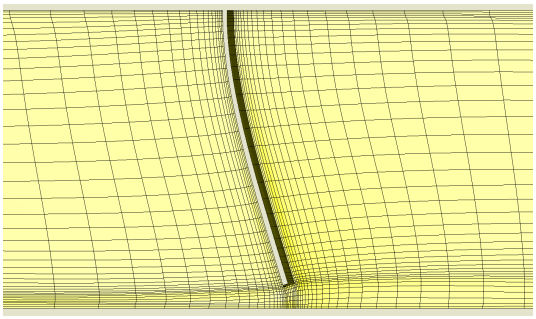
Figure 5.19: Cutting plane through the blade, showing the small gap between the rotor blade and the shroud.



(a) View of the cutting plane position

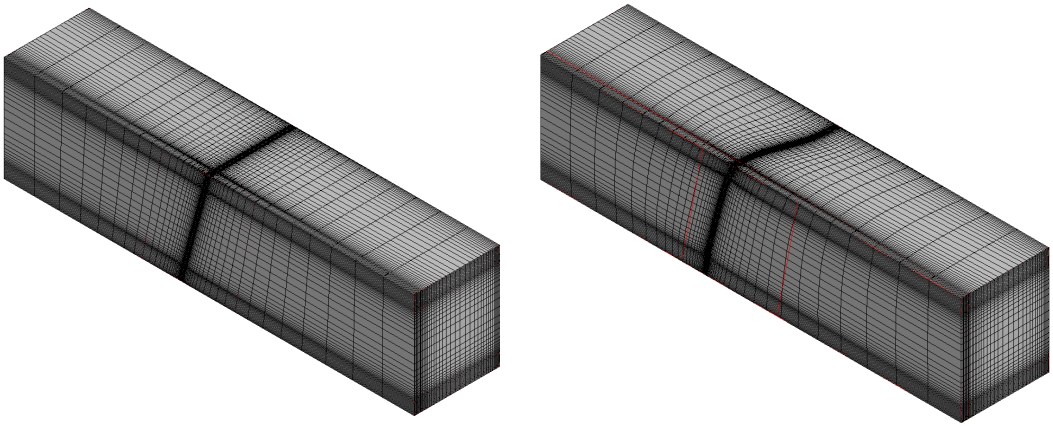


(b) Without sliding nodes



(c) With sliding nodes

Figure 5.20: Horizontal cutting plane through the middle of the channel, showing the gap between the flap and the channel wall.



(a) Without sliding nodes (b) With sliding nodes

Figure 5.21: 3D view of the deformed elastic flap mesh.

5.5 Absolute and Relative Displacements

Displacements can be defined with respect to either the initial mesh or the mesh of the previous time step. These two methods are referred to as absolute and relative displacements respectively. Both methods give different results and could moreover be implemented differently. This section will briefly discuss these issues.

In general it can be stated that the largest deformations can be performed when applying this deformation in several steps, using relative displacements. This is demonstrated by means of the orthogonality metric in figure 5.22 for the vortex induced beam vibration test case. Note that the skewness results are not displayed here, as they show the same trends. One tenth of the modal displacement as displayed in figure 4.3(b) is added to the motion at each deformation step. The relative method only breaks down after the tip displacement has reached 5 cm. On the other hand, the absolute displacement method manages to perform equally well as the relative method until a tip displacement of 3 cm is obtained. Beyond the 3 cm amplitude, the quality of the mesh drops quickly and negative cells occur at around 3.5 cm.

Despite the good performance of the relative method for large deformations, it has a main drawback which becomes clear when a periodic motion is applied. Namely, the mesh is not guaranteed to return to its initial quality when passing through the initial position. This situation is demonstrated in figure 5.23 for the vortex induced beam vibration test case, where the following oscillating motion is applied

$$\boldsymbol{u}(t) = \sin(0.5t) \cdot \boldsymbol{u}_{modal}, \tag{5.46}$$

where t is the displacement step and $\boldsymbol{u}_{modal}$ is the original modal displacement from figure 4.3(b). It is clear that the relative deformation method breaks down quickly after only a few

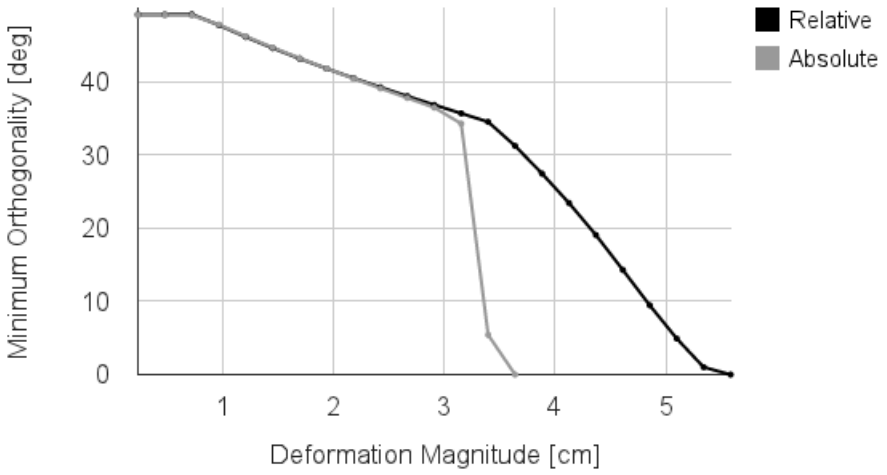


Figure 5.22: Comparison of the minimum orthogonality reached with absolute and relative displacements for the beam vibration test case.

oscillations, whereas the quality of the absolute method always oscillates around the initial mesh quality.

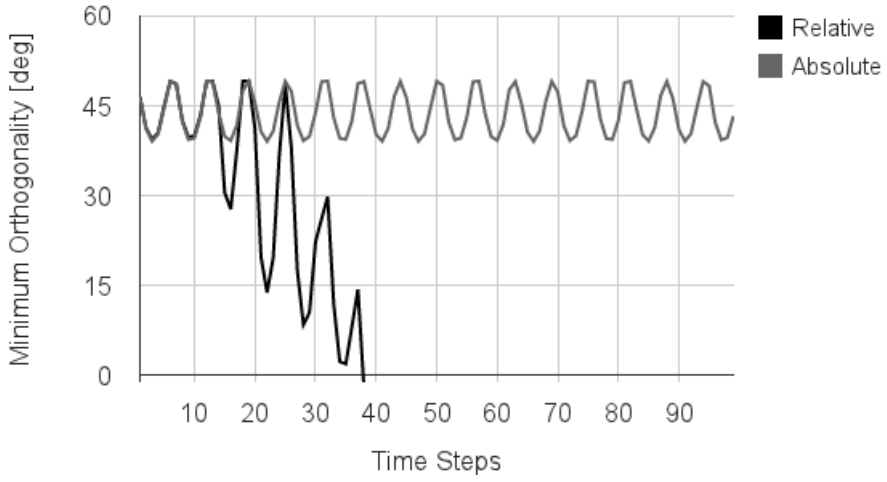


Figure 5.23: Comparison of the minimum orthogonality during an oscillating motion of the vortex beam when using absolute and relative displacements.

Therefore, a different method might be chosen for different deformation types. A large steady deformation should be done with relative displacements and smaller oscillating motions should be done with absolute displacements. The question remains how to deal with oscillating motions that have very large amplitudes which cannot be deformed with the absolute method. In such a case a hybrid method might be recommended, where relative displacements are used to reach the maximum amplitude, and once the object is back at its original position, the quality of the mesh is reset by using one absolute displacement from the initial mesh. However, due to time constraints, this avenue has not been investigated here and remains a topic for future research. One alternative to allow for larger deformation is to increase the size of the domain, such that there is more room to dissipate the displacements. However, this solution

can only be applied to external flows, where the user has the freedom to choose the size of the domain. The drawback of increasing the domain size is that the CFD time will increase proportionally.

In addition to the type of motion, the choice between the absolute and the relative method also depends on the mesh deformation method used. Structural analogies are best in combination with relative displacements, as this allows the iterations to start from a good initial condition. On the other hand, when the RBF method is chosen it is customary to use absolute displacements, as this allows to only invert the system matrix once, at the start of the computation. The inversion of the matrix is so costly that for a mesh of less than 1 million points it can take up to one day to complete. Once the matrix has been inverted, it can be re-used for several computations.

Also for the IDW method it is possible to store the interpolation weights $w_b(\boldsymbol{x})$ from equation 5.8, such that they can be re-used at every time step. There are however a very large amount of weights, namely $n_i \times n_b$. This can quickly grow to an unacceptable amount of data of several gigabytes. Even if it is taken into account that the weights only have to be stored for the moving and sliding boundary nodes, as for the fixed nodes simply the sum of all these weights could be stored, the size remains unacceptable. For example for the coarse mesh AGARD 445.7 test case, $n_i \times n_{mov} = 1,769,528,235$. If the distances have to be stored in double precision (1 double = 8 bytes), the storage required is equal to 13.184 GB. For the AGARD 445.7 fine mesh, this even increases to 430 GB. Moreover, it should not be forgotten that reading and writing data from a file also requires some time. This means that not much time can be saved by storing and re-using the weights when the cost to compute the weights is relatively low. Especially when storing data at network locations, which is often the case in industry environments, the reading time becomes large. Therefore this path is not further investigated here.

In FINE^{TM}/Open the user has the choice between absolute and relative displacements by means of an expert input parameter. The recommendation is to use absolute displacements for oscillating motions and relative displacements otherwise. For most of the test cases which are presented here, only one (large) deformation step is done. This means that a relative and an absolute displacement is the same.

Chapter 6

Efficiency Improvements

The IDW mesh deformation method as presented in this thesis, is a point-by-point method and hence has the advantage that it can easily be implemented in parallel. Despite this advantage, the cost of this deformation method can quickly become dominant for large 3D simulations. This is due to the fact that the cost of the method is of the order $\mathcal{O}(n_i n_b)$, where n_i is the amount of inner mesh nodes and n_b is the amount of boundary nodes. It is important to note that in three-dimensional cases, the number of surface nodes grows at a rate of $\mathcal{O}(n^{\frac{2}{3}})$, where n is the total number of mesh nodes. Finally this leads to a total cost of $\mathcal{O}(n^{\frac{5}{3}})$ [38]. In this chapter, a possible solution to this problem is presented. First a range of efficiency improvement methods are reviewed based on available literature, in section 6.1. Next, the implementation of the chosen method is explained in detail in section 6.2. Finally the results of this efficiency improvement are presented in section 6.3.

6.1 Efficiency Improvement Methods

The aim of this section is to present a brief literature overview of possible methods to increase the efficiency of inverse distance weighting interpolation, specifically for the application in mesh deformation. These methods can generally be sub-divided in two philosophies. The first is to find an approximate solution to the exact problem, whereas the second tries to find an exact solution to an approximate problem. Each subsection presents one possible method to increase the efficiency of IDW mesh deformation. These methods come from different backgrounds and tackle the problem in different ways. It has to be noted that most of the methods have not been proven effective for IDW mesh deformation, however they have the potential to give good results. Finally the last paragraph gives conclusions and recommendations concerning the presented methods.

6.1.1 Tree-Code Optimisation

In [38] Luke et al present a tree-code optimisation to increase the efficiency of IDW for large scale meshes. The method is inspired by tree-code optimisations that are frequently used in N-body simulations, such as the Barnes-Hut method and the Fast Multipole Method (FMM). The introduced method is based on the idea that nearby nodes should be treated exactly, whereas distant nodes can be approximated as one group.

Firstly the boundary nodes are organised in a kd-tree, such that clusters in the boundary nodes are identified. Next it is necessary to approximate the effect of a group of nodes by means of a set of four pseudo-nodes, called quad points. From this approximation a fast kd-tree evaluator can be build. For each volume node the kd-tree is descended recursively. At a vertex of the tree it is checked whether the error of the approximation meets a certain tolerance. If this is the case, the quad points are used, otherwise the tree has to be descended further.

The advantage of using such a kd-tree is that it automatically reduces the error near the boundaries, because in this region nearby leaves of the tree will be evaluated as an exact expansion. Typically a target error of 1% is sufficient for mesh deformation.

It is also possible to parallelise the kd-tree method, by dividing the vertices and leaves of the tree over several processors. A drawback is that the evaluation of the IDW interpolation leads to load imbalance because the nodes close to the boundaries tend to visit more leaves than the nodes further away from the boundaries. However this can be solved by redistributing the nodes based on the measured load imbalance of the previous evaluation.

Although no information is given about how the efficiency of the tree code evaluation compares to exact evaluation, it is demonstrated that its evaluation phase can be more than ten times faster compared to the RBF evaluation phase. Moreover it is demonstrated that the method can deform a 100 million node mesh distributed over 192 processors in less than 20s. [38]

For a more detailed description of the method, the reader is referred to appendix A.

6.1.2 Boundary Node Coarsening

Boundary node coarsening or data point reduction is a method that is popular to increase the efficiency of RBF mesh deformation. Since also in IDW mesh deformation the number of boundary nodes is one of the driving factors in the cost of the method, it is a good idea to investigate coarsening of the boundary nodes as an IDW efficiency improvement strategy.

The reduction of data points was first applied to RBF mesh deformation by Jakobsson and Amoignon [27]. They proposed to select boundary nodes by means of a minimum distance between data points, according to the following procedure:

1. Choose a minimal distance d between data points.
2. Organise all the boundary nodes into one long array. Select the first node as the first data point to initialise the procedure.
3. The next data point is the next point in the array whose distance to all previously selected nodes is at least d.
4. Continue the procedure until the end of the array is reached.

For the boundary nodes, which are not selected as data points, the displacement is not imposed, but will be computed by the interpolation just like for the volume mesh nodes. This means that an error is introduced at the boundary, i.e. the difference between the actual boundary node displacement and the computed interpolated displacement. However, even when selecting a relatively small amount of data points, this error is usually not very large [27].

Later, Rendall and Allen [54] noted that Jakobsson's method to select a reduced amount of data points does not try to minimise the error at the boundary. Therefore they introduced a greedy algorithm that selects data points to achieve a certain error bound. A greedy selection algorithm is an algorithm that at each iteration appends point(s) based on a local error assessment. It is greedy because at each iteration the algorithm corrects only for the largest local error, not taking into account the global consequences of this choice. Note that the greedy algorithm starts from a random initial point, or small set of points, and adds points per iteration, until the stopping criterion is met. This means that the greedy algorithm will require less time when less nodes are selected. Or in other words, the coarser the final surface mesh is, the more efficient this method becomes.

In order to accelerate the greedy-algorithm it is possible to select more than one point in each iteration. For example one could use the direction of the error to select two points with an angle of at least 90 degrees between their respective errors [69]. This can be done by first selecting the point with the largest error, and secondly selecting the point with the largest error that has at least an angle of 90 degrees with respect to the first selected node.

Experiments have shown that the error in surface displacement is approximately inversely proportional to the number of greedy iterations. Moreover, the final number of selected surface nodes is usually independent of the initial selection of surface nodes [54]. This leads to a mesh deformation algorithm cost that only depends on the number of volume mesh nodes and not on the number of boundary nodes. One particular result showed that when using only 3% of the boundary nodes, the error was only 0.022% of the maximum deformation [56]. Moreover when the number of boundary nodes is reduced with a certain factor, the evaluation cost is reduced with this same factor.

Finally, Kowollik et al [31], have proposed to coarsen the boundary mesh by using a kd-tree method. Their method consists of the following five steps

1. Construct a kd-tree with specified maximum and minimal region bounds.

2. Create bounding boxes for each leaf cell.
3. For each bounding box, generate search points. This can be one central point, 8 corner points or 8 gauss points, depending on the test case.
4. For each search point, find the closest surface mesh node.
5. Drop the surface nodes that have been selected more than once.

The advantage of this method, over the greedy method, is that it is not dependent on the deformation of the structure. This means that a black box approach can be used, where the coarse boundary nodes are selected beforehand.

The main drawback of coarsening the boundary is that the boundary nodes which are not included as data points are moved through the interpolation, rather than with their exact imposed displacement. This surface error can however be corrected for by means of a secondary motion scheme. Rendall and Allen [55] proposed to use a simple nearest point correction. The advantage of using this nearest point correction is that the method remains meshless. Another method that could be used for the correction step is the Delaunay Graph Mapping method of Liu et al [33]. In [31] it was demonstrated that using this method gives results which have a comparable mesh quality to the full RBF method, even with a large reduction (up till 98%) of control points. However, also other correction methods such as the linear spring analogy or Laplacian smoothing could be used. Moreover also less conventional methods such as the advancing front technique of Gerhold and Neumann [20] would be appropriate.

The quality of this hybrid scheme, which combines the coarsened RBF with a correction step, lies in the fact that two methods are used, where each method is used based on its own strengths. The RBF maintains orthogonality very well, but is prohibitively expensive for large applications. The correction method does not have to maintain orthogonality, but has superior efficiency. As such the resulting technique is both robust and efficient, without the drawbacks.

6.1.3 Local Inverse Distance Weighting

A frequently applied method to reduce the computational cost of RBF mesh deformation is the use of a local support radius. This means that the interpolation at a certain volume mesh node is only dependent on the surface mesh nodes that are within a certain radius. Also in IDW interpolation a local support radius is one of the oldest and most used methods to increase its efficiency [60][52]. One drawback to using a local support radius is that it is possible that some nodes have no data points within their radius and other nodes might have too many. Therefore it is possible to use the set of N closest surface nodes to a volume node in order to compute the interpolation, instead of using a fixed support radius. However, this method is more involved as it requires an efficient method to find nearest neighbours, which can be expensive. In order to minimise these drawbacks the two approaches can be combined. This is done by first selecting a maximum and a minimum amount of nodes that should be included for the interpolation at each volume node. Then an initial search radius is chosen

based on the average source point density and the required amount of nodes. During the interpolation it is checked for each point how many nodes there are within the initial search radius. If the amount is within the set minimum and maximum bounds, the initial radius is kept. Otherwise it will be set larger or smaller in case there were not enough or too many nodes within the initial search radius respectively. When using a local IDW interpolation the weighting function has to be adjusted such that it goes to zero at the radius R. One option is the following function as proposed in [60]

$$w_b(\boldsymbol{x}) = \begin{cases} \frac{1}{\|\boldsymbol{x}-\boldsymbol{x}_b\|} & \text{if } 0 < \|\boldsymbol{x}-\boldsymbol{x}_b\| < \frac{R}{3}, \\ \frac{27}{4R}\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_b\|}{R}-1\right)^2 & \text{if } \frac{R}{3} < \|\boldsymbol{x}-\boldsymbol{x}_b\| \leq R, , \\ 0 & \text{if } R < \|\boldsymbol{x}-\boldsymbol{x}_b\| \end{cases} \tag{6.1}$$

however this function can be adapted in different ways. For example Franke proposed a simpler scheme [18]

$$w_b(\boldsymbol{x}) = \left(\frac{(R-\|\boldsymbol{x}-\boldsymbol{x}_b\|)_+}{R\,\|\boldsymbol{x}-\boldsymbol{x}_b\|}\right)^2, \tag{6.2}$$

where

$$(R-\|\boldsymbol{x}-\boldsymbol{x}_b\|)_+ = \begin{cases} R-\|\boldsymbol{x}-\boldsymbol{x}_b\| & \text{if } \|\boldsymbol{x}-\boldsymbol{x}_b\| < R \\ 0 & \text{if } \|\boldsymbol{x}-\boldsymbol{x}_b\| > R \end{cases} \tag{6.3}$$

Using a local support radius has the difficulty to select the proper radius. When the selected radius is too small, the resulting interpolation might show bad properties. In the case of mesh deformation the support radius is dependent on the size of the deformation.

6.1.4 Fast Multi-Level Evaluation

Livne and Wright [35] introduced a fast multi-level evaluation (FMLE) to speed up the evaluation phase in RBF interpolation. The idea of the method is to evaluate the sum at a coarser level grid, which is then interpolated back to the fine grid. The method consists of three main steps

1. *Anterpolation:* compute the interpolation coefficients for the coarse Cartesian mesh
2. *Coarse Level Summation:* evaluate the interpolation sum on the coarse level grid.
3. *Interpolation:* interpolate the coarse level solution back to the fine level mesh, using a centered pth-order interpolation.

This method specifically applies to smooth radial basis functions. It decreases the cost of the RBF evaluation part from $\mathcal{O}(n_i n_b)$ to $\mathcal{O}((n_b+n_i)(ln(1/\delta))^d)$ where δ is the desired accuracy, d is the dimension, n_i is the number of evaluation points (or volume mesh nodes) and n_b is the number of data points (or boundary mesh nodes). When using radial basis functions that

have a singularity, such as for example the Thin Plate Spline (TPS) at $r = 0$, the interpolation error cannot be controlled in the region around $r = 0$. In such a case the function has to be split into a smooth and a local part. The local part can then be evaluated exactly, because it only has a local support, and the smooth part can be evaluated with the FMLE method. This consequently also requires working with a hierarchy of coarser level grids.

De Boer [14] showed that using FMLE for a smooth RBF indeed gives a significant efficiency increase: 35 times faster for an accuracy of 10^{-13} and 2000 times faster for an accuracy of 10^{-3} compared to the exact evaluation. However when using a hierarchy of grids applied to the piecewise smooth TPS function the efficiency drops drastically, which can be accounted to the fact that the density of the data points is not uniform.

Due to the fact that IDW also has a singularity at $r = 0$, the question remains whether FMLE is a suitable method to increase the efficiency. Furthermore the division of the function into a local and a smooth part also complicates the implementation.

6.1.5 Fast Multipole Method

The fast multipole method (FMM) [24] is often applied in N-body simulations. Its strategy is similar to the kd-tree optimisation in the sense that a tree data structure is used to separate the far-field and near-field contributions in the interpolation. As such the far-field contributions are evaluated with an approximation, whereas the near-field contributions are evaluated exactly. The difference with the method presented in section 6.1.1 lies in how the far-field is approximated. In FMM multipole expansions are used, which allow to group the far-field contributions. Using FMM can bring back the order of the method to $\mathcal{O}(n)$.

The FMM has already been proven to be applicable to RBF interpolation [6], however not yet in the context of mesh deformation. This can be attributed to the fact that the implementation of the method is quite involved [14]. Furthermore Luke et al [38] noted that it is not clear at what point the FMM would actually perform better than their kd-tree method. This combined with its complexity leads to the conclusion that FMM applied to IDW mesh deformation will not be further investigated here.

6.1.6 Moving Submesh Approach

Lefrançois [32] introduced a new method, which uses a combination of a coarse and a fine grid, called the Moving Submesh Approach (MSA). In this method first a coarse level grid is generated with a random meshing algorithm, by specifying a large cell size. This coarse mesh is then updated using a robust mesh deformation method, for which Lefrançois opted to use the elastic analogy. Finally the deformation of the coarse mesh is interpolated to the original fine mesh by using an interpolation method, similar to the finite element approach. One limitation of this method is that the background mesh needs to consist of triangular elements in 2D or tetrahedral elements in 3D. The resulting method is very efficient and

robust. However it is possible that the topology of the background mesh is visible in the fine mesh. In that case an additional post-smoothing step might be required.

Liu et al [34] demonstrated that also the RBF method is suitable to move the background mesh. Moreover they introduced another interpolation technique from the coarse to the fine mesh, using the area or volume ratios, determined by the position of the fine level node in the coarse level cell. The method is proven to be able to deal with large rotations, translations and deformations, with a quality comparable to radial basis function interpolation and an efficiency of two orders of magnitude higher.

It is clear that the MSA method could also be using IDW interpolation to update the coarse mesh. The remaining issue is how to generate the background mesh. In both of the above references the background mesh is generated separately at the start of the computation. This would require an extra input mesh from the user, plus additional knowledge to decide upon the size of the coarse level cells. Moreover in both examples the background mesh is required to be triangular or tetrahedral. With the application into FINE^{TM}/Open in mind, only a hexahedral or hybrid mesh will be available to work with. Therefore the interpolation method from fine to coarse mesh might have to be adapted. However, since each hexahedral element can easily be subdivided into imaginary tetrahedral cells, this should not impose a significant problem. The application in FINE^{TM}/Open would also allow to use the existing coarse level agglomerations from the multigrid approach as the background mesh.

6.1.7 Conclusions and Recommendations

In total six methods have been discussed which might be used to increase the efficiency of IDW mesh deformation. Each of these methods has its own advantages and drawbacks. Based on the background provided in the previous sections, some conclusions can be drawn.

- The FMM is the most complex method. Moreover, it has no clear advantages over the tree-code optimisation method. Therefore it is not recommended to use this method.
- For the FMLE method it is not clear whether good results can be obtained, due to the singularity of the IDW interpolation function. As such, this method is also not recommended.
- For the MSA method, several considerations have to be taken. One will need to have a coarse background mesh available in order to apply this method. In FINE^{TM}/Open , the agglomerated multigrid levels could be used for this purpose. However, due to the object-oriented architecture of the code, the multigrid levels are only available in the CFD solver part and not in the mesh deformation part. Therefore, even-though this method might be a good method to apply to IDW, it will not be considered further here.
- The locally supported IDW method could be very beneficial for mesh deformation cases where the deformation is small compared to the size of the mesh. However, in our particular case of FSI, this would usually not be the case. Additionally, the use of a

local IDW interpolation could also affect the quality of the mesh, as the motion cannot be propagated far into the volume mesh.

- The kd-tree method seems to be particularly useful to increase the efficiency of IDW mesh deformation. However this method is also fairly complex and limited information about this method is available as it is fairly new.
- The method of boundary node coarsening is particularly appealing because of its simplicity and the fact that it can be applied to both RBF and IDW, which is not the case for the kd-tree method. Additionally, the boundary node coarsening method has been proven highly effective to increase the efficiency of RBF mesh deformation and similar results should be expected for IDW.

Based on this list of conclusions, the boundary node coarsening method is chosen to be implemented here. The exact details of the implementation are given in the following section.

6.2 Boundary Node Coarsening

As explained in section 6.1.2, there are several methods to coarsen the boundary mesh. However so far only one method tries to select an optimal set of boundary nodes, which reduces the introduced boundary error. This method is the greedy algorithm as presented in [54]. Moreover, thanks to the explicit nature of the IDW interpolation function, it is easy to update the interpolation function at each iteration of the greedy method. Therefore, the greedy method is chosen to coarsen the boundary mesh in this thesis project. The following sections will elaborate on the details of the boundary node coarsening implementation.

6.2.1 The Greedy Algorithm

The greedy algorithm can be explained in five steps:

1. Select an initial active point or set of active points to start the iteration. This can simply be the first point of the list of boundary nodes.
2. Use the active list of boundary nodes to construct a surface interpolation.
3. At each boundary node, compute the norm of the error between the interpolated displacement and the actual displacement at the boundary node.
4. Append the point with the largest error to the active list. Now this point is included in the construction of the surface interpolation function, as such the error at that point is per definition equal to zero.
5. Iterate until the target error bound is met at all the inactive nodes or until a set number of nodes has been selected.

In this thesis it was chosen to use all the corners of the boundaries to start the greedy iteration. This choice makes sure that these crucial points of the domain are represented exactly.

At each iteration of the greedy algorithm a new interpolation function has to be computed, including the one extra contribution of the new active node. For RBF interpolation, this requires either solving the system at each greedy iteration or adding a local correction to the function from the previous step [54]. For IDW, however, this is easy because it simply requires adding the contribution of the active node to the sums of the IDW interpolation function. Even when rotations are included this can easily be done. It simply requires storing the numerator and denominator for each inactive node for both the translation and rotation component. There is one rotation method however, which is difficult to combine with the greedy method. This is the SLERP rotation method. As mentioned in section 5.3.3, the SLERP interpolation is dependent on the order of interpolation. This means that when the greedy method is used, the order in which the active nodes are added to the list, is also the order in which the interpolation for the inner nodes will have to be done. This is in itself not problematic, however the code had already been structured in a way that the interpolation for the inner nodes always applied an order of moving, sliding and then fixed. The reason why the moving, sliding and fixed nodes are grouped is that it allows to avoid if-statements for each boundary node, which makes it more efficient. Therefore, the current implementation does not allow to combine boundary node coarsening with the SLERP rotation interpolation method. This is however not a problem as several other methods, such as LERP and LERP-log are available, which give very similar results, as explained in section 5.3.

One additional difficulty, when using the greedy method for boundary node coarsening, is the treatment of sliding boundary nodes. In order to apply the greedy method, the displacement of all nodes has to be known. However, the displacement of the sliding nodes is not known beforehand. This problem can be tackled in several ways. First of all it is possible to assume a displacement of zero for all sliding nodes during the greedy algorithm. This option has the advantage that it is very simple, although it also leads to the selection of more nodes on the sliding boundary as necessary, as the assumption of a zero displacement leads to bigger errors at the sliding boundary. Additionally this method gives no guarantees about the actual maximum interpolation error after coarsening, as it is not based on the actual displacements. The second option is to first coarsen the fixed and moving boundary, then update the position of the sliding boundary, and then coarsen the whole boundary (including the sliding boundary), now taking the active nodes of the moving and fixed boundary as a starting point. This method is more involved, as it requires several steps, and several considerations have to be made when multiple sliding faces are used. In both of the above methods, the new position of the sliding boundary is based on the coarse mesh. This means that there might be some errors at the boundaries with other surfaces. In order not to jeopardise the robustness of the IDW mesh deformation method, it is chosen to compute the position of the sliding boundary nodes before coarsening is done. This means that the IDW interpolation function for the sliding nodes will be based on the original surface mesh, without coarsening, which might make the computation of the sliding boundary displacement expensive. However, since the amount of sliding boundary nodes is usually limited to a couple of surfaces, this is not a limiting factor.

6.2.2 Greedy Error Functions and Stopping Criteria

In this section a detailed description will be given of the choice of error function and stopping criteria for the greedy algorithm. Several options are proposed and compared to each other in a series of test problems.

The greedy algorithm needs an error signal to select the surface nodes. This error function can either be based on geometrical arguments or interpolation errors. For the purpose of mesh deformation Rendall and Allen [56] have demonstrated that the actual surface error, i.e. the norm of the error between the actual surface displacement and the interpolated surface displacement, is the most effective error signal. One disadvantage of this error choice is that the displacement of the boundary has to be known, and hence this requires selecting a new set of points for each new deformation step. Alternatively the unit error function can be used to select a reduced data set only once at the start of the computation. This is identical to imposing a unit displacement to the boundary, instead of the actual displacement. However using the unit function gives no guarantees that a certain target error bound is met at each deformation step. Therefore, when using the unit function, usually more boundary nodes will have to be selected, compared to when the surface error function is used. This in turn results in a higher cost of the interpolation phase for the inner mesh nodes. Additionally some preliminary test cases showed that the greedy method has a low cost compared to the total cost of the mesh deformation. As such, it is chosen to use the surface error function and perform the greedy algorithm at each iteration. Section 6.4 will discuss how this can be applied more efficiently during time-dependent simulations.

For the greedy coarsening method to be useful, it has to be combined with a good choice of stopping criteria. When the surface error function is used, the greedy algorithm can be stopped as soon as a pre-defined error bound is met. This error bound will, however, be problem dependent. Since it is preferable to have a method that requires the least possible problem dependent inputs, it is attempted here to find several criteria that can be applied more universally. The performance of each criterion is demonstrated by means of both the coarse and fine mesh AGARD 445.6 test case. The test results for the elastic flap and rotor 67 (fine and coarse mesh) demonstrate the same characteristics and can be reviewed in appendix B.

Criterion 1

The first, logical choice is to scale the errors ϵ with a fixed reference length l_{ref}, such that a relative, dimensionless error $\hat{\epsilon}$ is achieved

$$\hat{\epsilon} = \frac{\epsilon}{l_{ref}}. \tag{6.4}$$

The reference length, should represent the scale of the problem. In this case it was chosen to be equal to the aerodynamic reference length, as this is already available during the simulation. The greedy algorithm can then be stopped when all errors are smaller than the desired

maximum relative error, which will usually range from 10^{-3} to 10^{-2}

$$\frac{\epsilon_{max}}{l_{ref}} < k, \quad \text{where } k \approx 10^{-3} \rightarrow 10^{-2}. \tag{6.5}$$

The maximum relative error achieved, with respect to the number of selected nodes, is shown in figure 6.1(a) and 6.2(a) for the coarse and fine mesh AGARD test case respectively when $k = 0.01$. For both meshes, the selected number of active boundary nodes is around 340 out of an original of 33,602 and 134,402 boundary nodes for the coarse and fine mesh respectively. This indicates that the selected number of nodes is independent of the mesh, and rather depends on the geometry and deformation of the problem. Moreover it is remarkable that by only selecting 340 nodes, the error is already smaller than 1% of the root wing chord. For the fine mesh this means that only 0.26% of the entire boundary is used. The selected nodes are visualised in figures 6.3(b) and 6.4(b) for the coarse and fine mesh respectively. Here it is clear that the amount of nodes that has been selected on the moving boundary is very small compared to the number of nodes selected on the outer boundary. This might be due to the fact that in the AGARD test case the parameter $\alpha_{mov} = 0.3$, whereas $\alpha_{fix} = 0.1$, which means that the area around the moving boundary will be deformed in a more rigid fashion.

Criterion 2

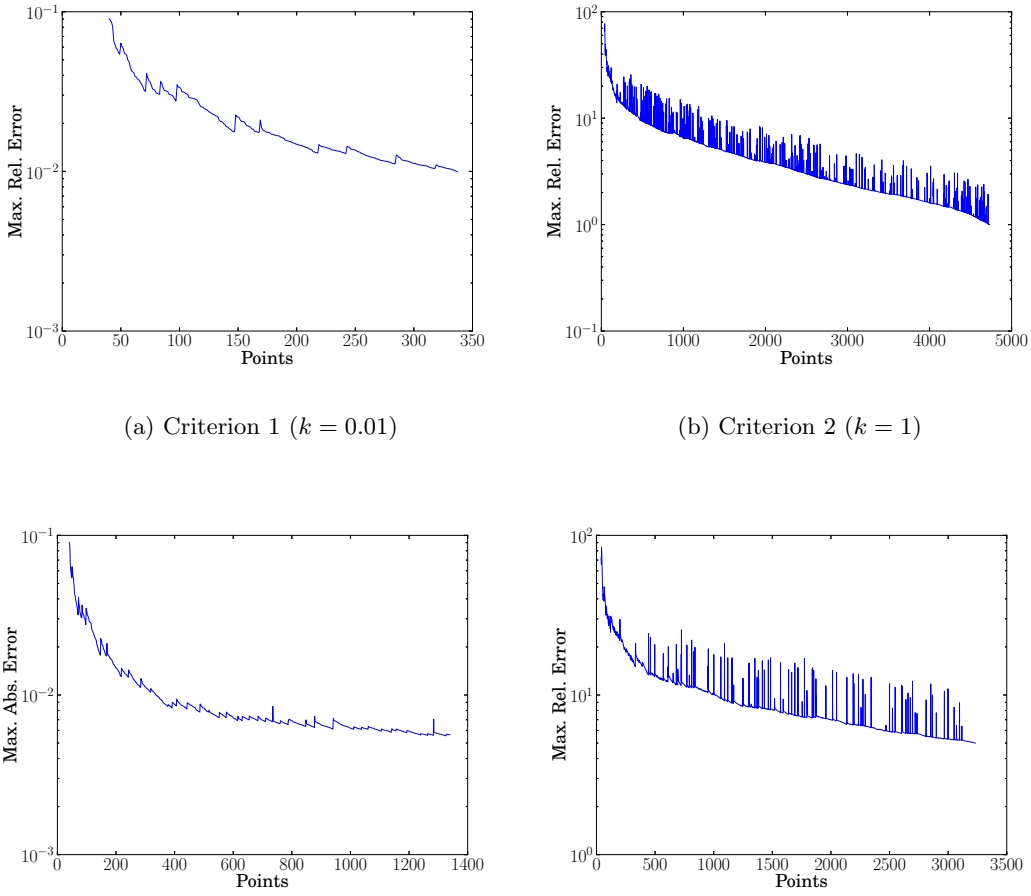
The second stopping criterion is based on the idea that not all boundary nodes require the same maximum error. It is reasonable to assume that the boundary nodes in viscous layers might require a smaller error than the boundary nodes at the exterior boundaries where the cells are larger. Therefore, a second choice of error function is to scale the surface error ϵ with the boundary cell height h corresponding to a boundary node

$$\hat{\epsilon} = \frac{\epsilon}{h}. \tag{6.6}$$

This will automatically lead to a higher selection of nodes where the cells are smaller. For this method a good stopping criterion would be to stop when the relative error is smaller than 1, since this means that the error at the boundary is smaller than its cell height

$$\frac{\epsilon_{max}}{h} < k, \quad \text{where } k \approx 1. \tag{6.7}$$

The maximum relative error function for criterion 2 is plotted in figures 6.1(b) and 6.2(b), for the coarse and fine mesh respectively. From these results it is clear that a lot more active nodes (18959) are selected for the fine mesh compared to the coarse mesh (4733). This is due to the fact that the boundary height in the fine mesh is twice as small, which means that many extra nodes have to be selected to reach such a low error. The noise in the displayed signals is caused by the fact that each time an extra node is selected, this reduces the error around the node, but at the same time it can also introduce an error further away from the node. When looking at the selected nodes in figures 6.3(c) and 6.4(c), it can be seen that the nodes are mainly selected in the areas where the boundary height is very small. It is remarkable that for the right hand exterior boundary nearly no nodes are selected, as the cells in this far field boundary are relatively high. This can be problematic as it means that the absolute error at this boundary is quite large, and such a large error cannot always be corrected properly with the correction step as described in section 6.2.3. On the other hand,



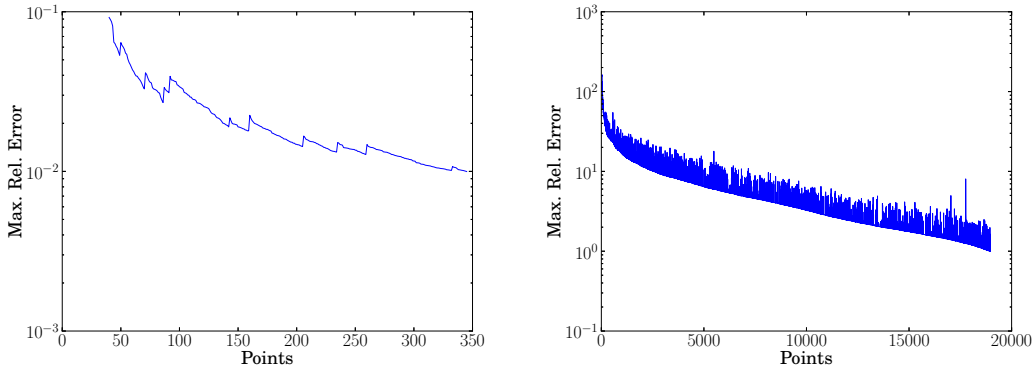
(a) Criterion 1 ($k = 0.01$)

(b) Criterion 2 ($k = 1$)

(c) Criterion 3 ($k = -10^{-6}$)

(d) Criterion 4 ($k_1 = 5$, $k_2 = 0.01$)

Figure 6.1: Maximum relative error with respect to the number of active boundary nodes for the different greedy criteria for the coarse mesh AGARD 445.6 test case.



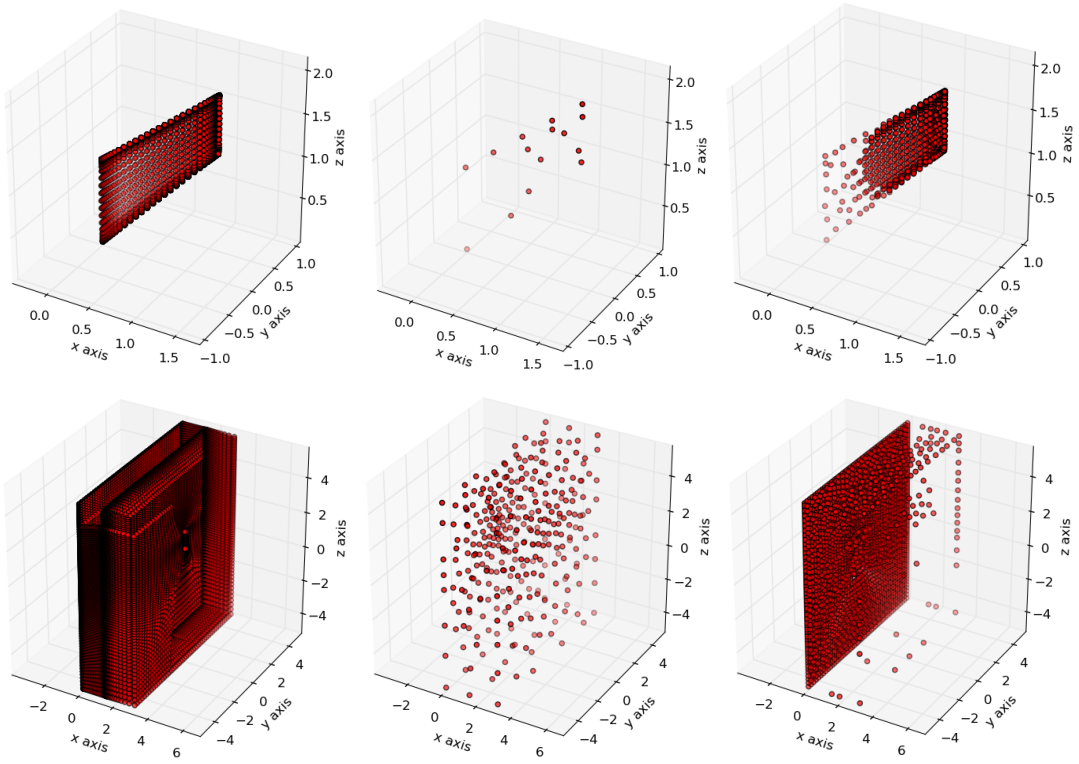
(a) Criterion 1 ($k = 0.01$)

(b) Criterion 2 ($k = 1$)

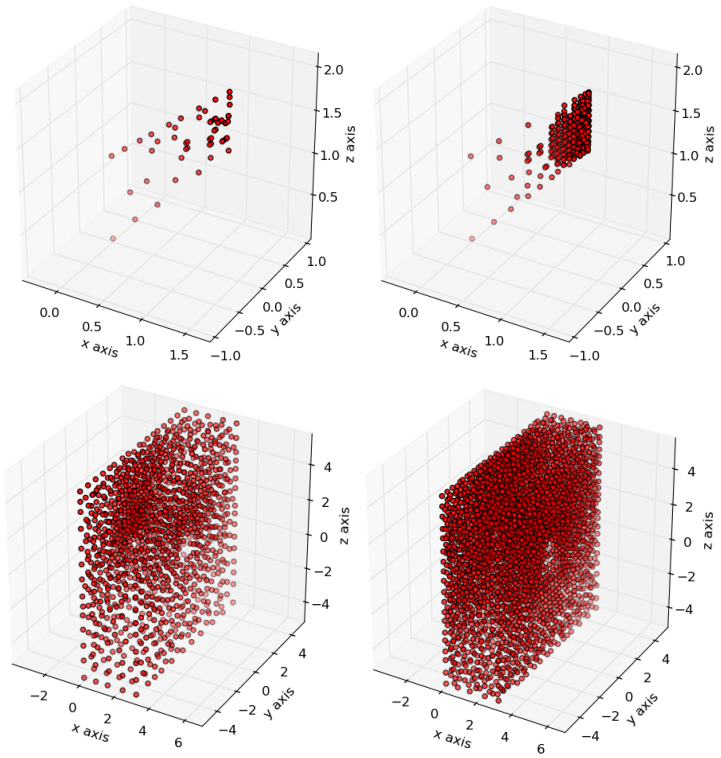
(c) Criterion 3 ($k = -10^{-6}$)

(d) Criterion 4 ($k_1 = 10$, $k_2 = 0.01$)

Figure 6.2: Maximum relative error with respect to the number of active boundary nodes for the different greedy criteria for the fine mesh AGARD 445.6 test case.

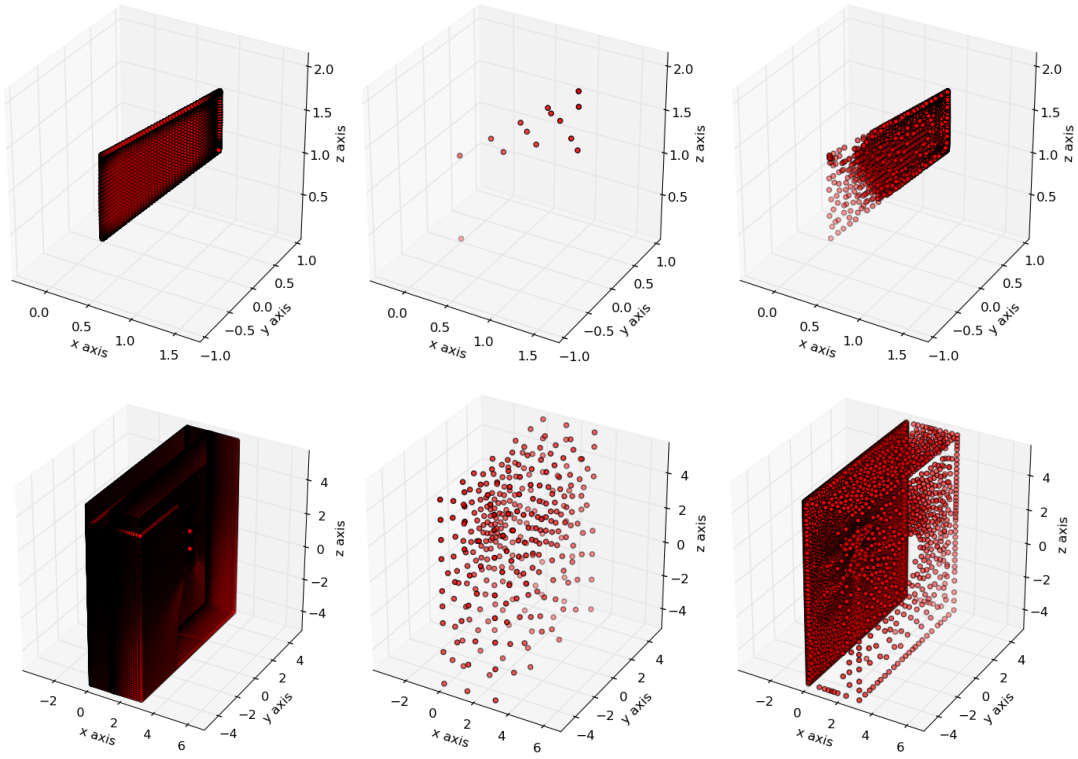


(a) Without coarsening (b) Criterion 1 ($k = 0.01$) (c) Criterion 2 ($k = 1$)

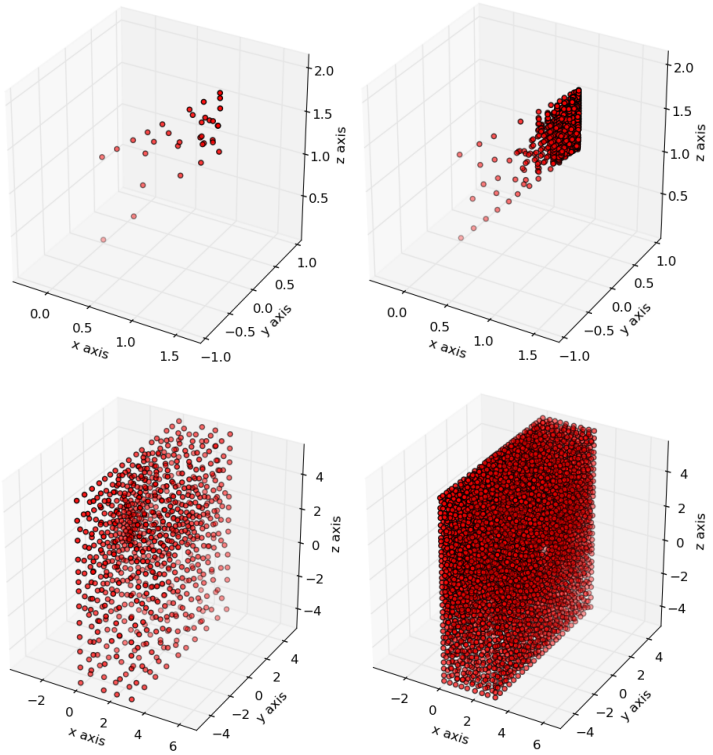


(d) Criterion 3 ($k = -10^{-6}$) (e) Criterion 4 ($k_1 = 5$, $k_2 = 0.01$)

Figure 6.3: Comparison of the selected nodes for the coarse AGARD 445.6 test case between the different greedy criteria (top = wing, bottom = exterior boundary).



(a) Without coarsening (b) Criterion 1 ($k = 0.01$) (c) Criterion 2 ($k = 1$)



(d) Criterion 3 ($k = -10^{-6}$) (e) Criterion 4 ($k_1 = 10$, $k_2 = 0.01$)

Figure 6.4: Comparison of the selected nodes for the fine AGARD 445.6 test case between the different greedy criteria (top = wing, bottom = exterior boundary).

the mirror face at the root of the wing, has a lot of active boundary nodes. This is due to the fact that the mesh originates from a structured mesh, where the cells along this wall are of equal size as the cells around the wing.

Criterion 3

For the next error criterion, the fact that the maximum error achieved varies logarithmically with the number of active nodes, is used. This can be seen in figures 6.1(a) and 6.2(a) (note the logarithmic scale for the vertical axis). This means that, at the start, adding nodes is very effective, resulting in a rapid decrease in the error. However after adding a certain amount of nodes to the active list, this decrease will stagnate, and the error will only continue to drop very slowly. Therefore it makes sense to use the slope of the maximum absolute error function as a stopping criterion. This slope can be approximately computed by doing a least-squares fit to the list of maximum errors of each greedy iteration. The linear least-squares fit has the following form

$$\epsilon_{max}(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3, \tag{6.8}$$

where $x = \frac{1}{N}$, c_i are the polynomial coefficients, ϵ_{max} is the maximum error and N is the number of active boundary nodes. To accurately represent the slope at the last node of the maximum error function, it suffices to do the least-squares fit to the last added active nodes. Here it was chosen to do a fit to the last 500 nodes. Moreover this fit does not have to be done at every greedy iteration, but can for example be done every 100 iterations. The slope of the absolute error function is negative, and it is desirable for the greedy algorithm to stop when the slope approximates zero. Experiments have proven that a good stopping criteria would be around -10^{-6}

$$\frac{d\epsilon_{max}}{dN} < k, \quad \text{where } k \approx -10^{-6}. \tag{6.9}$$

The error function for the third error criterion is shown in figures 6.1(c) (coarse mesh) and 6.2(c) (fine mesh). Due to the fact that for this criterion the error is not scaled, the absolute errors are displayed. The function itself is actually the same as for criterion 1, only the scale is different. It is surprising that for this criterion significantly more nodes are selected for the coarse (1340) than for the fine mesh (840), even when the same maximum value for the slope of -10^{-6} is used. This might be due to the fact that the computed slope is only an approximation and the noise in the signal can reduce the accuracy of the computed slope. Moreover it can be noted that more nodes are selected than for criterion 1. This indicates that the chosen error bound for criterion 1 might not have been sufficiently small. However, if an error bound of 0.001 instead of 0.01 would have been selected, a large amount of nodes would be added to the list of active nodes, as it is clear that the error is only dropping slowly. Therefore using criterion 3, allows to use the greedy criterion in an efficient manner, without imposing a specific error bound.

Criterion 4

Finally, the last choice of error function to be introduced in this thesis is a combination of the first and the second criterion. When using the surface error relative to the cell height,

this means that the surface error will be extremely small for viscous layers. However, since a correction step is also being used, it might not be necessary to reach such a small error bound as it will lead to a large amount of active boundary nodes. This could be solved by setting a larger stopping criterion, such as five for example. On the other hand, this will result in large errors at the exterior boundaries, which might be problematic for the correction step, as it is only designed for small corrections. By combining the first and second criterion, both the error relative to the cell height and the error relative to a certain reference length can be kept small. This will make sure that enough nodes are selected for both the viscous boundary and the exterior boundary. The combination of the two criteria is done as follows. First for the small cells, the following criteria should hold

$$\frac{\epsilon_{max}}{h} < k_1, \quad \text{where } k_1 \geq 1, \tag{6.10}$$

where h is the height of the boundary cell and k_1 is the error bound, which for this case will be equal to or larger than 1. Secondly for all boundary nodes, the maximum error relative to the aerodynamic reference length l_{ref} is required to remain below a certain upper bound k_2

$$\frac{\epsilon_{max}}{l_{ref}} < k_2, \quad \text{where } k_2 \approx 10^{-3} \rightarrow 10^{-2}. \tag{6.11}$$

Both criteria should hold at the same time. From equations 6.10 and 6.11 it can be seen that the first criterion violates the second one if

$$h > \frac{k_2 l_{ref}}{k_1}. \tag{6.12}$$

This can be prevented by creating a pseudo height h' as follows

$$h' = \begin{cases} h & \text{if } h \leq \frac{k_2 l_{ref}}{k_1}, \\ \frac{k_2 l_{ref}}{k_1} & \text{if } h > \frac{k_2 l_{ref}}{k_1}. \end{cases} \tag{6.13}$$

The stopping criterion for the greedy loop then becomes

$$\frac{\epsilon_{max}}{h'} < k_1. \tag{6.14}$$

For this criterion, $k_1 = 10$ for the fine mesh and $k_1 = 5$ for the coarse mesh test case. These values are chosen differently because the cell height of the fine mesh is twice as small as the cell height of the coarse mesh. For the parameter k_2 the same value of 0.01 is chosen as in criterion 1. Note that when using criterion 2 for the AGARD test case, it would not have been possible to choose a value of 5 or 10 for the error bound, because this would lead to prohibitively large errors at the exterior boundary. The relative error functions for both the coarse and the fine mesh are shown in figures 6.1(d) and 6.2(d) respectively. In this case more nodes are selected for the fine mesh (5609) than for the coarse mesh (3233). When comparing this to the original amount of boundary nodes, only 4% of the total fine mesh boundary nodes are selected, whereas for the coarse mesh 10% of the boundary nodes are selected. Finally when looking at the distribution of the selected nodes in figures 6.3(e) and 6.4(e), it is clear that, compared to the second criterion more nodes are selected on the right hand outer boundary, whereas the amount of nodes on the wing has slightly decreased due

to the fact that $k_1 > 1$ for criterion 4. On the other hand, when comparing the results to the selected nodes for criterion 1, more nodes are selected on the wing, and surprisingly, also more nodes are selected on the outer boundary. This is probably due to the fact that including more nodes on the wing, introduces additional errors in the outer boundary. Therefore, also more nodes on the exterior boundary are necessary to compensate for the additional moving boundary nodes.

Comparison of the Absolute Errors

Finally a comparison can be made between the different greedy criteria by looking at the maximum absolute errors that are achieved, as indicated in figures 6.5 and 6.6 for the coarse and fine mesh respectively. The most peculiar result is that when using criterion 2, the absolute error actually increases first. Once again this can be explained by the fact that adding nodes at one place, introduces an error at another place. In this case, initially many nodes are selected at the moving boundary, which introduces larger errors at the outer boundary. Eventually this absolute error decreases again, by adding more nodes at the exterior boundary, such that the error will be below the cell height for all nodes. For criteria 1 and 3 the absolute error is the same, only the iteration at which the greedy algorithm is stopped is different. This means that the order in which the nodes are added during the greedy iterations are the same. This is as expected since the node where the absolute error is the largest is also the node where the error divided by l_{ref} (which is a constant) is the largest. It can be stated that method 1 and 3 have the same selection criteria with a different stopping criteria. For the other methods both the selection and stopping criteria are different. When using criterion 1 or 3 the absolute error drops the fastest, which is due to the fact that, at each iteration the node with the largest absolute error is added to the active list. On the other hand criterion 4 is somewhere in the middle, as both the error with respect to cell height and with respect to the reference length are decreased simultaneously. The influence of the absolute and relative errors on the mesh quality will become clear in section 6.3.

Summary of the Criteria

The four error criteria for the greedy iteration are summarised in table 6.1. The second column gives indicative values that can be chosen by the user. Except for the value for criterion 3, these values are quite intuitive, and it should not form a problem for the inexperienced CFD user to set-up these parameters. However, it is also possible to use the default values, as indicated in the third column.

Table 6.1: Error criteria for greedy method

	Criterion	Indicative Value	Default Value
1.	$\frac{\epsilon_{max}}{l_{ref}} < k$	$k \approx 10^{-3} \rightarrow 10^{-2}$	$k = 10^{-2}$
2.	$\frac{\epsilon_{max}}{h} < k$	$k \approx 1$	$k = 1$
3.	$\frac{d\epsilon_{max}}{dN} > k$	$k \approx -10^{-6} \rightarrow -10^{-8}$	$k = -10^{-6}$
4.	$\frac{\epsilon_{max}}{h} < k_1$ and $\frac{\epsilon_{max}}{l_{ref}} < k_2$	$k_1 \approx 1 \rightarrow 10$ and $k_2 \approx 10^{-3} \rightarrow 10^{-2}$	$k_1 = 3$ and $k_2 = 10^{-2}$

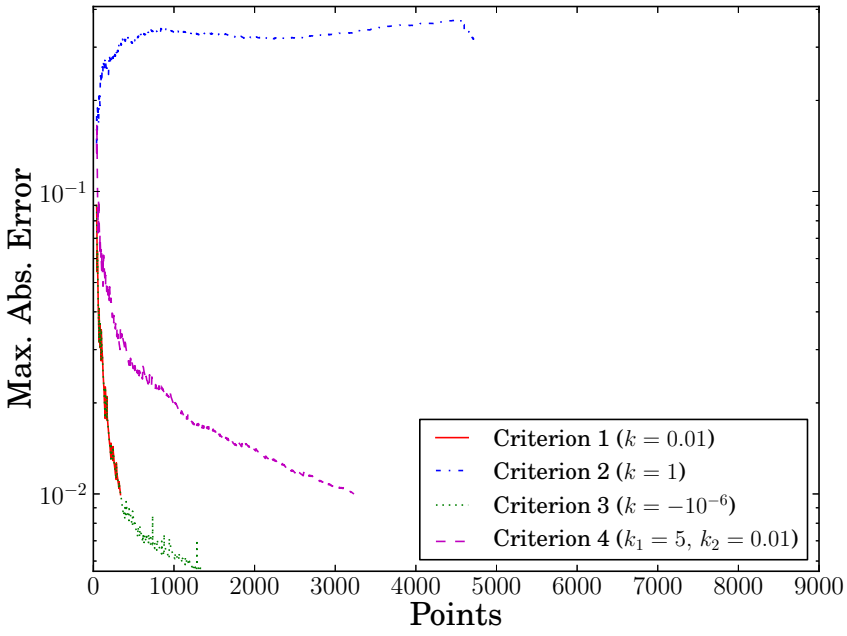


Figure 6.5: Comparison of the absolute errors of the different greedy criteria for the coarse mesh AGARD 445.6 test case.

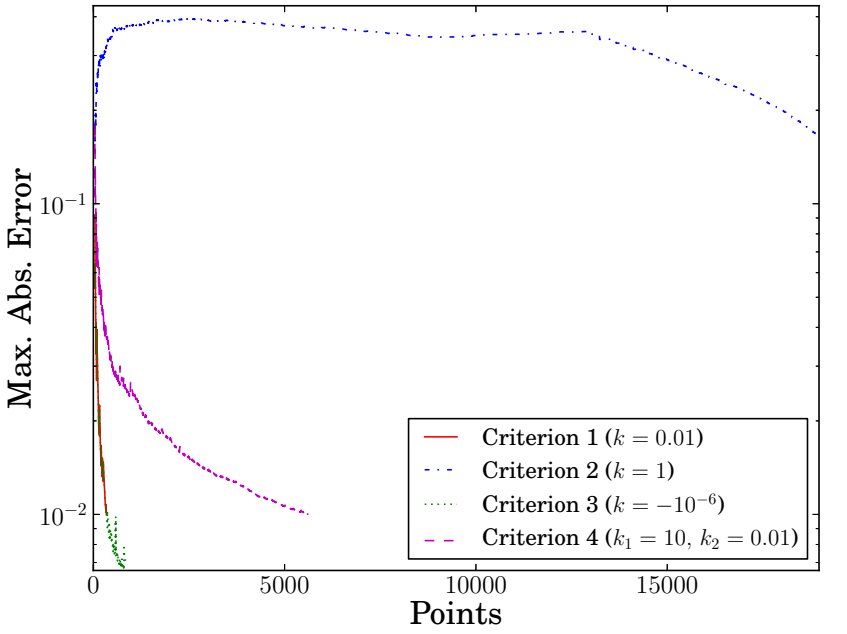


Figure 6.6: Comparison of the absolute errors of the different greedy criteria for the fine mesh AGARD 445.6 test case.

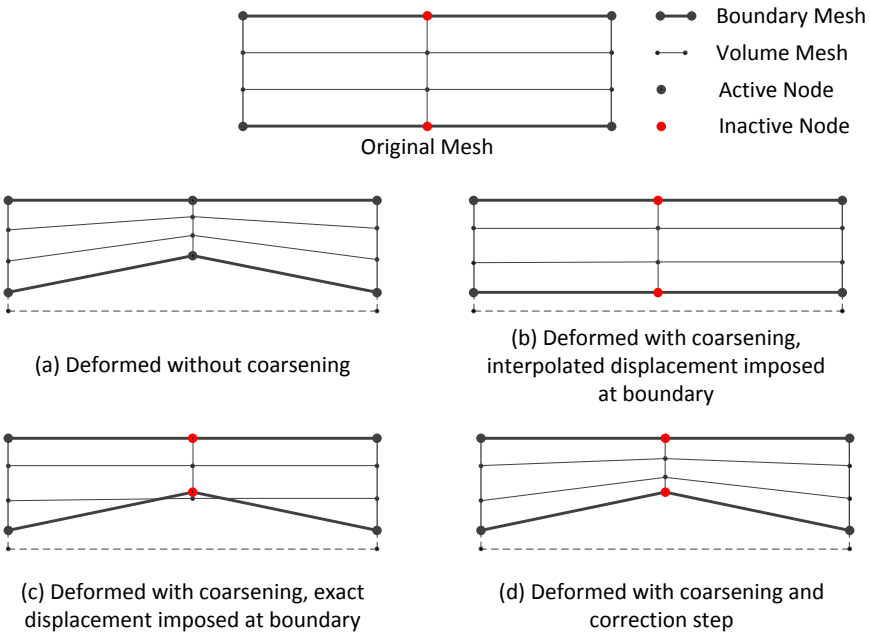


Figure 6.7: Comparison between the different options to correct for the boundary displacement error due to coarsening.

6.2.3 Correction Step

As explained in the previous section, the use of only a limited set of boundary nodes, means that the imposed boundary geometry is only satisfied exactly at the active boundary nodes. For all the inactive boundary nodes there will be an error between the interpolated displacement and the actual imposed displacement. This is illustrated in figure 6.7(b), where the middle boundary nodes are inactive nodes. If one wants to make sure that the boundary geometry is preserved exactly, there are two options. The first option, is to still use the imposed boundary displacements to move the boundary mesh. However, this will only result in a valid mesh, in case the error at the boundary is smaller than the first cell layer. Otherwise, moving the boundary to their exact positions might intrude in this layer, causing negative cells, as shown in figure 6.7(c).

The second option corrects for the error in the boundary layer displacement by means of a correction step. This means that in figure 6.7, the mesh with erroneous boundary in image (b), is corrected to the situation in figure (d). This correction step can be seen as a local secondary mesh deformation step. The main requirement for the secondary mesh deformation method is that it has to be a highly efficient method. On the other hand, it does not need to be very robust, as it only has to deal with a small deformation. Several options for the correction step were discussed in section 6.1.2.

In this thesis it is chosen to satisfy the boundary motion exactly by means of a correction step. The proposed method of Rendall and Allen in [55] is used, because of its efficiency and

simplicity. This method is a nearest neighbour correction, given by

$$\boldsymbol{u}_c(\boldsymbol{x}) = \boldsymbol{\delta}_{\mathrm{n}} \phi_n(\boldsymbol{x}), \tag{6.15}$$

where the subscript n indicates the nearest boundary node neighbour, $\boldsymbol{u}_c(\boldsymbol{x})$ is the correction displacement of an inner node at position $\boldsymbol{x}$, $\boldsymbol{\delta}_n$ is the displacement of the boundary node from its interpolated to its exact position and $\phi_n(\boldsymbol{x})$ is the used RBF. An RBF with compact support should be chosen, such that the correction step can be limited within a confined region. As in [55] Wendland's C^2 function will be used here. It is given by

$$\phi_n(\boldsymbol{x}) = \begin{cases} (1-d)^4(4d+1) & \text{if } d \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{6.16}$$

where $d = \|\boldsymbol{x} - \boldsymbol{x}_n\| / s$ and s is the support radius. Since the correction displacement $\boldsymbol{\delta}_n$ is small, the support radius can be chosen relatively small too. Experiments showed that using a variable support radius, depending on the magnitude of the corrected displacement, gives the best results. Concretely, the support radius is scaled with $\boldsymbol{\delta}_n$ for each boundary node. A factor of ten seemed to be sufficient here, such that the support radius for boundary node n is equal to

$$s_n = 10 \|\boldsymbol{\delta}_n\| . \tag{6.17}$$

The only issue concerning this correction step is that finding the nearest neighbour of all volume nodes is an expensive operation if it is done by a 'brute force' method. This means that for each volume node the distance to each boundary node is computed and the minimum distance is found by comparison, which has a total cost of $\mathcal{O}(n_i n_b)$. Luckily, other methods exist to find a volume node's nearest neighbour. Specifically in FINE$^{\mathrm{TM}}$/Open a wave-front vector distance transform method, has already been implemented in the turbulence model, which is based on the work in [10]. Since this method has been proven to be fast and accurate, the same method is pursued here. The idea behind this method is that the closest boundary node to a certain volume node will usually be close to the closest boundary node of its neighbours in the volume grid. Thus, if the closest boundary nodes are known for the neighbours of a volume mesh node, the approximate closest boundary node of this volume mesh node can be found. This algorithm can be started at the boundary itself, and from there propagated further into the mesh. For the boundary correction step the distance only needs to be computed within a certain radius of the boundary. Therefore the wave propagation can simply be stopped when the maximum defined support radius is reached. To improve the accuracy of this wave-front method, not only the neighbours of a node, but also the neighbours of its neighbours are taken into account for both the boundary and volume mesh nodes. For more details regarding the implementation of this method the reader is referred to [10].

The addition of the boundary node coarsening method with correction step concludes the implementation of the IDW mesh deformation method. A high-level flowchart of the complete IDW mesh deformation algorithm can be found in appendix D.

Table 6.2: Efficiency and quality of the different coarsening criteria for the coarse mesh AGARD test case.

Criterion	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
1 ($k = 0.01$)	338	0.00s	2.10s	1.60s	5.12s	24.17°	0.75
2 ($k = 1$)	4733	9.92s	32.75s	3.04s	46.24s	0.94°	0.99
3 ($k = -10^{-6}$)	1340	2.95s	7.96s	1.37s	12.78s	24.99°	0.75
4 ($k_1 = 5$ and $k_2 = 0.01$)	3233	6.42s	20.00s	1.48s	28.62s	25.19°	0.75
No coarsening	33602	/	225.16s	/	225.66s	24.71°	0.73

Table 6.3: Efficiency and quality of the different coarsening criteria for the fine mesh AGARD test case.

Criterion	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
1 ($k = 0.01$)	346	4.5s	16.98s	12.15s	38.25s	23.83°	0.73
2 ($k = 1$)	18959	150.5s	1031.00s	19.64s	1205.80s	0.00°	1.00
3 ($k = -10^{-6}$)	840	8.63s	40.81s	12.24s	66.29s	23.71°	0.73
4 ($k_1 = 10$ and $k_2 = 0.01$)	5609	47.75s	287.44s	12.46s	352.3s	25.06°	0.73
No coarsening	134402	/	7279.84s	/	7284.44s	24.48°	0.69

6.3 Results of IDW Mesh Deformation with Boundary Coarsening

In this section the resulting mesh qualities and the efficiency of the different greedy coarsening criteria will be compared to each other for five different test cases. The test cases that are considered here are the coarse and fine mesh AGARD 445.6 wing, the coarse and fine mesh rotor 67 and the elastic flap.

The computation time and resulting mesh quality for the five test cases for the different greedy criteria are listed in tables 6.2 to 6.6. Here, n_b is the number of active boundary nodes, t_{coarse} is the computation time to coarsen the mesh, t_{inner} is the computation time to compute the interpolated displacements for the volume mesh nodes, t_{corr} is the computation time to perform the correction step, t_{tot} is the total time required to deform mesh, *Ortho* is the *minimum* orthogonality and *Skew* is the *maximum* skewness. It is important to remember that for the orthogonality a high value indicates a high quality mesh, whereas for the skewness a low value indicates a high quality mesh as explained in section 1.4. Note also that the total time for the mesh deformation is not exactly equal to the sum of t_{coarse}, t_{inner} and t_{corr}. This is because also other steps are performed within the mesh deformation module, such as the computation of the rotation component.

The first conclusion that can be drawn from the tables is that the time required to coarsen

Table 6.4: Efficiency and quality of the different coarsening criteria for the coarse mesh rotor 67 test case.

Criterion	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
1 ($k = 0.001$)	2907	3.25s	5.61s	0.62s	11.35s	24.39°	0.79
2 ($k = 1$)	2254	2.81s	5.24s	0.70s	10.69s	30.45°	0.74
3 ($k = -10^{-6}$)	1728	1.94s	3.33s	0.68s	7.83s	0.00°	1.00
4 ($k_1 = 1$ and $k_2 = 0.001$)	3958	4.42s	7.96s	0.60s	14.84s	26.04°	0.78
No coarsening	17730	/	34.15s	/	36.04s	31.77°	0.72

Table 6.5: Efficiency and quality of the different coarsening criteria for the fine mesh rotor 67 test case.

Criterion	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
1 ($k = 0.001$)	8433	40.95s	145.27s	5.11s	219.63s	22.19°	0.79
2 ($k = 1$)	9597	47.35s	189.68s	6.07s	270.81s	2.05°	1.00
3 ($k = -10^{-6}$)	1228	6.78s	20.21s	6.33s	66.15s	0.00°	1.00
4 ($k_1 = 1$ and $k_2 = 0.001$)	14515	67.37s	254.12s	5.12s	355.11s	25.70°	0.74
No coarsening	70914	/	1159.90s	/	1187.96s	6.51°	0.93

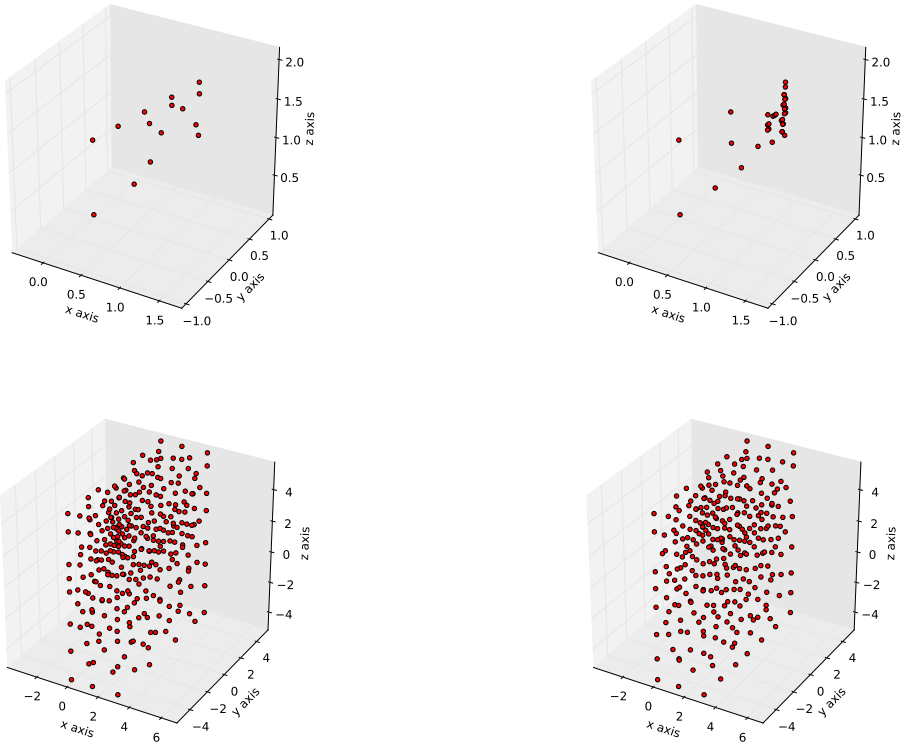
Table 6.6: Efficiency and quality of the different coarsening criteria for the elastic flap test case.

Criterion	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
1 ($k = 0.01$)	578	0.81s	1.73s	0.93s	6.29s	29.94°	0.69
2 ($k = 1$)	1626	2.08s	4.92s	0.89s	10.57s	31.44°	0.74
3 ($k = -10^{-6}$)	432	0.67s	1.25s	0.89s	5.42s	28.10°	0.69
4 ($k_1 = 1$ and $k_2 = 0.01$)	1657	2.16s	5.07s	0.97s	10.84s	31.65°	0.73
No coarsening	17282	/	46.89s	/	49.63s	41.72°	0.55

and to correct the mesh displacement is significantly lower than the time required to compute the displacement of the inner nodes. Moreover, it can be seen that t_{coarse} scales with $\mathcal{O}(n_b)$, t_{inner} scales with $\mathcal{O}(n_i n_b)$ and t_{corr} scales with the maximum absolute error achieved and the number of active boundary nodes. This means that t_{corr} is usually the largest for criterion 2, where the number of boundary nodes and the maximum error are relatively large. On the other hand, the coarsening time t_{coarse} is usually larger for criteria 4, as more nodes are added due to the fact that two criteria are satisfied at the same time, leading to a higher amount of nodes. The maximum time spend in coarsening is 67 s for criterion 4 in the fine mesh rotor 67 test case. In this case 14,515 active nodes are selected out of a total of 70,914 boundary nodes. This coarsening time is, however, still only 5% of the total deformation time without coarsening.

Secondly, in terms of total deformation time, it is clear that, no matter which greedy criterion is used, a significant reduction in computation time is achieved. In general the increase in efficiency is higher for the finer meshes. The best efficiency increase is reached for the fine mesh AGARD 445.6 test case when using criterion 1. For this case the computation time is 38.25 s, or 0.5% of the computation without coarsening which is 7284.44 s (2 h 1 m 24.44 s). This is a remarkably good result, as it means that a mesh with approximately 3×10^6 nodes can be moved in just over a half a minute, without even using multiple processors. This time is negligible compared to the time to perform the CFD computations. The lowest efficiency increase is obtained in the coarse mesh rotor 67 test case when using criterion 4. In this case the computation time is 14.84 s, or 41% of the computation without coarsening which is 36.04 s. These numbers clearly indicate that the efficiency improvements that can be reached with coarsening both depend on the test case and the criterion that is used. Some experience from the user will be required to choose the desired outcome. For very fine meshes, it is recommended to always apply coarsening. Moreover it is important to note that the efficiency increase would also be higher if smaller deformations were applied. This is due to the fact that a lower deformation leads to lower errors, which means that less nodes would be added to the active list during coarsening.

However, such a reduction in computation time would mean nothing, if it meant that the mesh quality was very poor. Therefore, it is important to compare this increase in efficiency to the change in mesh quality. Once again the change in mesh quality is highly dependent on the test case and the coarsening method used. For most test cases the best results are achieved when using criterion 4, which is probably due to the fact that it combines the advantages of criteria 1 and 2. It is remarkable that for the AGARD test cases and the fine rotor 67 case, the minimum orthogonality even increases when criterion 4 is used, compared to the case without coarsening. This might be due to the fact that when coarsening is done, the weight distribution between the moving, fixed and sliding boundary is automatically optimised. Another remark that can be made, is that when criterion 2 is used for the AGARD test cases and the fine rotor 67 test case, the quality drops dramatically to the point where negative, concave and twisted cells occur. This can be explained by the fact that the errors at the outer boundary are too large for the correction method to correct them. Finally, also criterion 3 seems to have problems to create good quality meshes for certain test cases. For both rotor 67 test cases negative cells occur when this criterion is used. In these cases the negative cells are due to the fact that the errors in the shroud gap are too large. As such, it seems that the slope criterion is not always capable to define a good point to stop, as there is no guarantee that



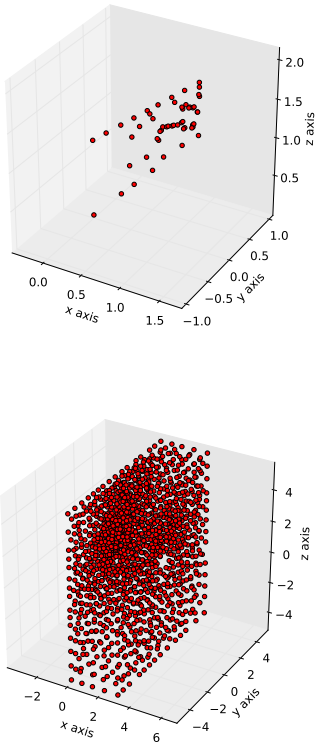
(a) Selection method 1 (n_b = 15 moving +321 fixed = 336)

(b) Selection method 4 (n_b = 35 moving +301 fixed = 336)

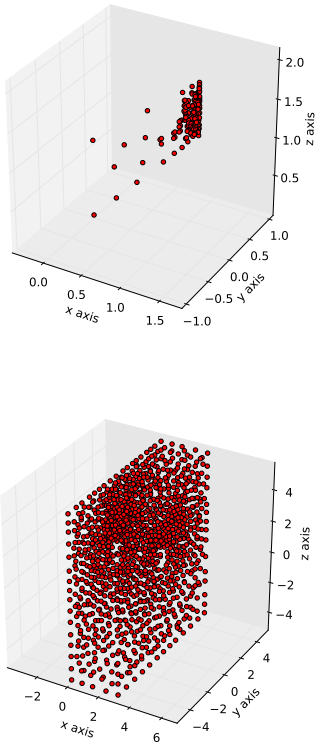
Figure 6.8: Comparison of the selected nodes for the coarse AGARD 445.6 test case for selection method 1 and 4, when the greedy iteration is stopped as soon as 1% of the boundary nodes are selected (top = wing, bottom = exterior boundary).

sufficiently small errors are achieved.

In order to find the method which is the most effective in selecting boundary nodes it is necessary to compare the different selection methods while keeping the number of active boundary nodes at a fixed value. As stated before the order in which the nodes for criterion 1 and 3 are selected is the same. Hence this will be referred to as selection method 1. Criterion 2 will not be considered further since it is already clear from the previous paragraphs that it has a poor performance. Finally, the order in which nodes are selected for criterion 4 will be referred to as selection method 4. A test has been done for both the fine and coarse mesh AGARD 445.6 case, where 1% of the boundary nodes is selected. Note that for selection method 1 it is not necessary any more to choose a k-value. On the other hand, for selection method 4, the values of $k_1 = 10$ for the fine mesh, $k_1 = 5$ for the coarse mesh and $k_2 = 0.01$ for both meshes are chosen. These values are important because they are used to scale the errors. The resulting coarsened meshes for the coarse and fine mesh are displayed in figures 6.8 and 6.9 respectively. For the same amount of active nodes, selection method 4 selects at least twice as many moving boundary nodes compared to selection method 1. This is reflected



(a) Selection method 1 ($n_b = 53$ moving $+1291$ fixed $= 1344$).



(b) Selection method 4 ($n_b = 147$ moving $+1197$ fixed $= 1344$).

Figure 6.9: Comparison of the selected nodes for the fine AGARD 445.6 test case for selection method 1 and 4, when the greedy iteration is stopped as soon as 1% of the boundary nodes are selected (top = wing, bottom = exterior boundary).

Table 6.7: Comparison between selection method 1 and 4 when selecting 1% of the total amount of boundary nodes for the coarse and fine mesh AGARD 445.6 test case.

Method	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	**Ortho**	**Skew**
Coarse mesh AGARD 445.6							
1	336	0.95s	2.10s	1.69s	5.57s	24.17°	0.75
4 ($k_1 = 5$ and $k_2 = 0.01$)	336	0.94s	2.16s	1.85s	5.67s	24.85°	0.75
Fine mesh AGARD 445.6							
1	1344	12.41s	65.21s	11.08s	95.59s	23.78°	0.73
4 ($k_1 = 10$ and $k_2 = 0.01$)	1344	12.83s	69.29s	13.82s	102.87s	24.97°	0.73

Table 6.8: Comparison between selection method 1 and 4 when selecting 10% of the total amount of boundary nodes for the coarse and fine mesh rotor 67 test case.

Method	n_b	t_{coarse}	t_{inner}	t_{corr}	t_{tot}	Ortho	Skew
Coarse mesh rotor 67							
1	1773	2.16s	3.52s	0.66s	8.34s	0°	1.00
4 ($k_1 = 1$ and $k_2 = 0.001$)	1773	2.22s	3.81s	0.69s	8.74s	26.34°	0.78
Fine mesh rotor 67							
1	7091	34.78s	121.06s	5.45s	191.15s	22.34°	0.85
4 ($k_1 = 2$ and $k_2 = 0.001$)	7091	35.77s	131.68s	5.62s	202.84s	18.52°	0.89

in the resulting mesh qualities presented in table 6.7, where selection method 4 always has a slightly higher mesh quality. On the other hand selection method 1 still performs moderately faster than selection method 4. The reason is twofold. First, due to the fact that the absolute errors that are reached with selection method 1 are smaller, the correction step requires less time as the local support radii become smaller. Second, the fact that more moving boundary nodes are selected by method 4, results in a higher computation time as moving nodes require more computations for rotations and translations (for all fixed nodes the translation vector is the zero-vector and the rotation quaternion is the unit quaternion, which allows to compute the effect of all fixed nodes at once). For the rotor 67 test case the same tests have been done. This time with a fixed maximum of 10% of the nodes. The results are summarised in table 6.8, whereas the meshes are visualised in appendix B.4. These tests largely show the same trends as discussed for AGARD 445.6. One exception is the fact that selection method 1 results in a better quality than selection method 4 for the fine mesh rotor 67 test. An explanation might be that selection method 4 chooses too many nodes at the blade, which means that too much weight is assigned to the blade causing the quality of the cells around the blade to increase, at the cost of a decreasing quality between the blade and the exterior. It should be noted that for the rotor 67 test case the geometry is more complex and as such coarsening becomes more difficult.

In general it can be concluded that criterion 4 provides the most stable coarsening method in terms of resulting mesh quality. This criterion usually also has the highest cost, however, with an appropriate choice of the criterion boundaries k_1 and k_2, a good efficiency increase can be reached. Moreover the previous paragraph has illustrated that this criterion can also be stopped earlier, by setting a maximum allowed percentage of selected boundary nodes. Criterion 1 also gives good quality results for a high efficiency, however usually the quality is slightly lower than criterion 4. On the other hand, criterion 2 has too many shortcomings with respect to both the quality and the effectiveness, and should therefore not be used. Finally criterion 3, seems to be slightly less stable, as there are no guarantees about the resulting errors. Also the fact that the slope is difficult to approximate because of the noise in the signal, makes this method less reliable. It could potentially be used to get a quick idea of how many nodes should be selected.

6.4 Coarsening During a Time-Dependent Simulation

In this section it will be investigated whether it is possible to apply the greedy coarsening method in a more efficient way when a time dependent simulation is done. Such a time-dependent simulation is usually characterised by a large series of deformation steps ($\mathcal{O}(10^3)$), where the difference between two consecutive deformation steps is small. As such it is fair to assume that the coarsening of the boundary for two consecutive deformation steps will result in the selection of a similar group of active nodes.

The idea is to use the set of boundary nodes from the previous time step as the initial set to start the greedy algorithm for the current time step. In figure 6.10 a result is shown where this strategy is applied to the fine mesh AGARD 445.6 test case. The original method, where the full coarsening algorithm is repeated for each deformation step, is referred to as "full coarsening" and "incremental coarsening" indicates that the set of coarse nodes from the previous step is used as the initial set for the current step. The deformation at each time step is increased with 0.1 of the total modal displacement. From the graph it is clear that a larger displacement indicates a higher computation time. This is due to the fact that for a large displacement more active nodes are selected, which induces a higher cost for both the greedy algorithm and the interpolation step. Secondly, it is clear that the *incremental coarsening* method is only slightly faster than the *full coarsening* method. The reason why incremental coarsening is almost as costly as full coarsening, is that for incremental coarsening the interpolated values for all inactive nodes, based on interpolation from the active nodes, still have to be computed. This is necessary to start the greedy algorithm and to be able to apply the correction step at the end. The computation of the interpolated values for the inactive nodes is nearly as expensive as the greedy algorithm itself, as in essence all the greedy algorithm is doing is progressively computing this value by adding one active node per iteration. The difference in cost between the greedy algorithm and the computation of the interpolated values of the inactive nodes is due to the fact that the greedy algorithm has to compute and compare errors along the way.

Another important note to make, is that because the greedy method is based on errors, which are zero for the active nodes, it is not possible to use the same greedy algorithm to remove nodes. This means that if a decreasing motion is applied, the incremental coarsening method is stuck with the large set of active nodes from the initial deformation step. This is illustrated in figure 6.11, where the deformation steps of figure 6.10 are applied in reversed order. From this figure it is clear that applying incremental coarsening is not efficient when used for decreasing motions. As such figure 6.10, represents the case where the incremental coarsening method is at its most efficient behaviour.

One way to eliminate the drawbacks of incremental coarsening is by setting up a fixed number of iterations after which the coarsening is restarted. However, in the case of oscillating or variable deformations, it is questionable whether the incremental coarsening method will actually outperform the full coarsening method. As such it is advised to always apply the full coarsening method in such cases. Incremental coarsening should only be applied in the case where one is certain that the motion is only in an increasing fashion.

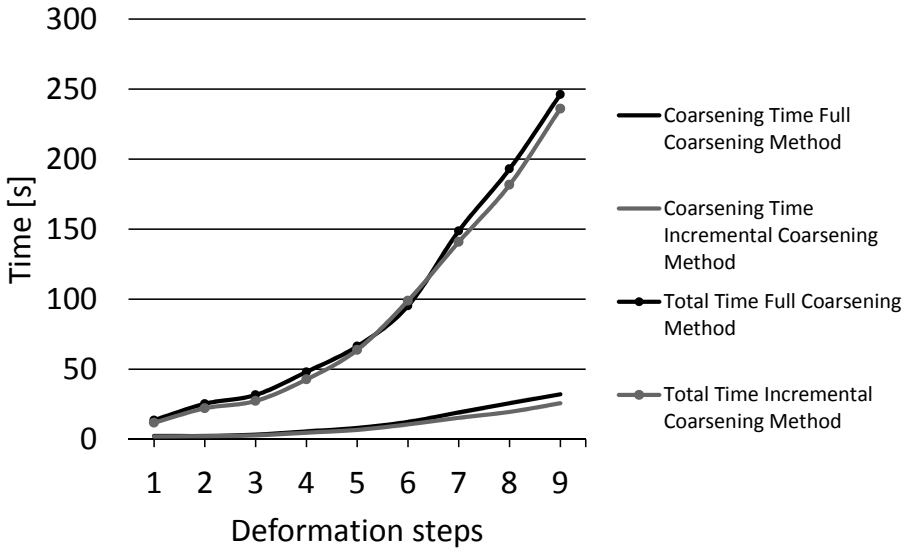


Figure 6.10: Coarsening and total deformation time for 9 deformation steps with growing amplitude for the AGARD 445.6 fine mesh test case.

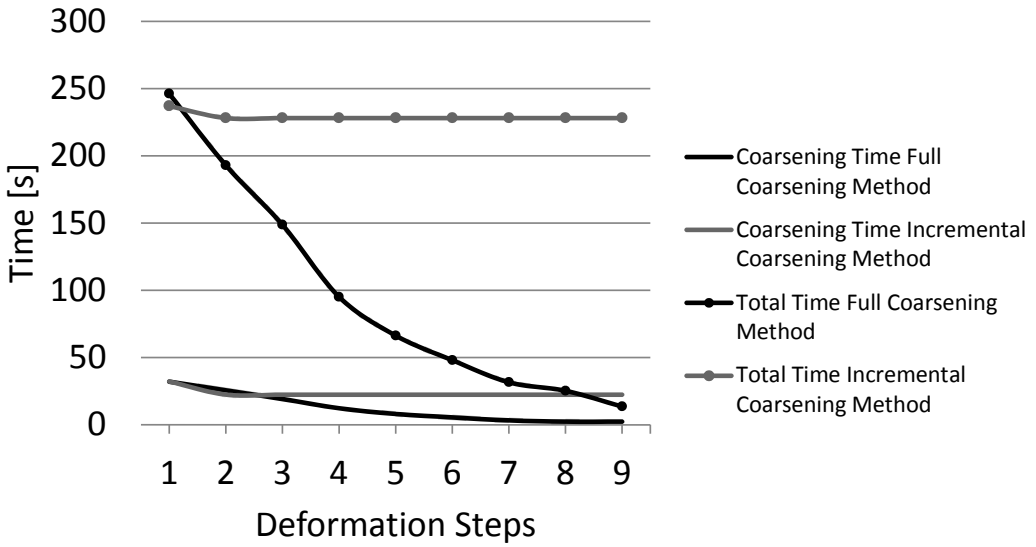


Figure 6.11: Coarsening and total deformation time for 9 deformation steps with decreasing amplitude for the AGARD 445.6 fine mesh test case.

6.5 Conclusion

In conclusion it can be stated that boundary node coarsening is a valuable method to improve the efficiency of the IDW mesh deformation method. In particular a method where the relative error with respect to the boundary height is kept small for the smallest cells and the relative error with respect to the aerodynamic reference length is kept small for the larger cells, results in high quality meshes. For this method two parameters have to be specified. One error bound with respect to the cell height (typically $1 \rightarrow 10$) and one error bound with respect to the aerodynamic reference length (typically $0.01 \rightarrow 0.001$). Alternatively, the easiest way to do coarsening is by simply setting an error bound for the relative error compared to the reference length. This usually results only in a slightly lower mesh quality as compared to when also the error relative to the cell height is minimised and has a higher efficiency. It is also possible to stop the coarsening before the error bounds are reached by simply setting a maximum percentage of nodes that is allowed to be selected. The application of boundary node coarsening can increase the efficiency of the IDW method up to 200 times. Moreover, the detrimental effect coarsening has on the mesh quality is fairly limited. For some test cases the resulting mesh quality is even higher when coarsening is applied compared to the full IDW method without coarsening.

Chapter 7

Results

In this chapter the results of the IDW mesh deformation method are compared to the RBF and elastic analogy mesh deformation methods as detailed in section 3.4.2 and 3.4.3. All test cases as presented in chapter 4 are performed for the three methods.

7.1 Test Case Settings

Firstly, the settings used for all the test cases are summarised in table 7.1. Note that in this table only the parameters that vary between the test cases are given. The scaling factor for the RBF method is a factor by which the distances are multiplied such that they remain between 0 and 1. For most cases this was left at the default value of 1. Only for the AGARD 445.6 test it was necessary to change it to 0.0675, which is equal to the inverse of the longest diagonal of the domain in grid units (= 0.5587 m in AGARD 445.6 case). Furthermore the Thin Plate Spline function is used for the RBF.

Next, the IDW method parameters include the α-values and the rotation interpolation method. As explained in section 5.3, it is necessary to choose method 1 for large rotations, such as in the NACA 0012 test case, and method 4 for all other cases. Additionally, the default values for the α-parameters are: $\alpha_{mov} = 0.1$, $\alpha_{slide} = 0.1$ and $\alpha_{fix} = 0.0$. The α_{mov} and α_{fix} values are increased for the AGARD 445.6 test case, to ensure high near boundary quality. This is possible since the mesh has more than enough room to dissipate the displacements. On the other hand the α_{slide} value is decreased for the vortex vibration case to leave more area to absorb the deformation. For more information about changing the α-values the reader is referred to section 5.2.2. No coarsening is used for neither IDW nor RBF. This is done because at the moment different coarsening strategies are implemented for both methods, which would make a good comparison impossible.

The default settings for the EAMD tests are indicated in table 7.2. The elastic coefficients

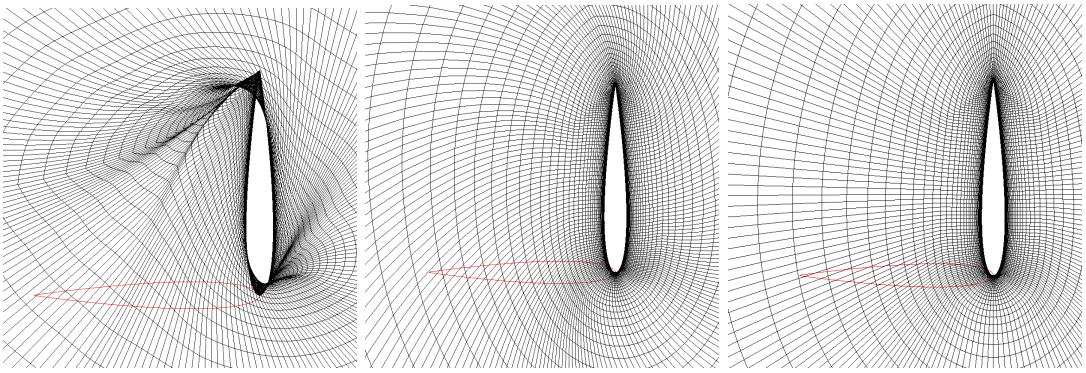
1 and 2 are the values of a and b in equation 3.14. Note also that the values for the sliding faces are not always equal between the IDW and EAMD tests (see table 7.1). This is due to the different nature of both methods, where IDW spreads the motions much further into the mesh, which leads to the fact that it is more useful to slide the exterior boundaries.

Table 7.1: Settings for the IDW tests.

Test case	**IDW**					**EAMD**	**RBF**
	α_{mov}	α_{fix}	α_{slide}	Rotation	Sliding	Sliding	Scaling Factor
NACA 0012	0.1	0.0	0.1	1	Exterior	Exterior	1
Vortex vibration	0.1	0.0	0.0	4	Exterior	-	1
Elastic flap	0.1	0.0	0.1	4	Top, bottom, front	Top, bottom, front	1
Rotor 67	0.1	0.0	0.1	4	Shroud	Shroud	1
AGARD 445.6	0.3	0.1	-	4	-	-	0.0675

Table 7.2: Settings for the EAMD tests.

Poisson Coefficient	0.3
Elastic Coefficient 1	5.0 for 2D and 3.0 for 3D
Elastic Coefficient 2	3.0 for 2D and 1.0 for 3D
Tolerance	10^{-5}

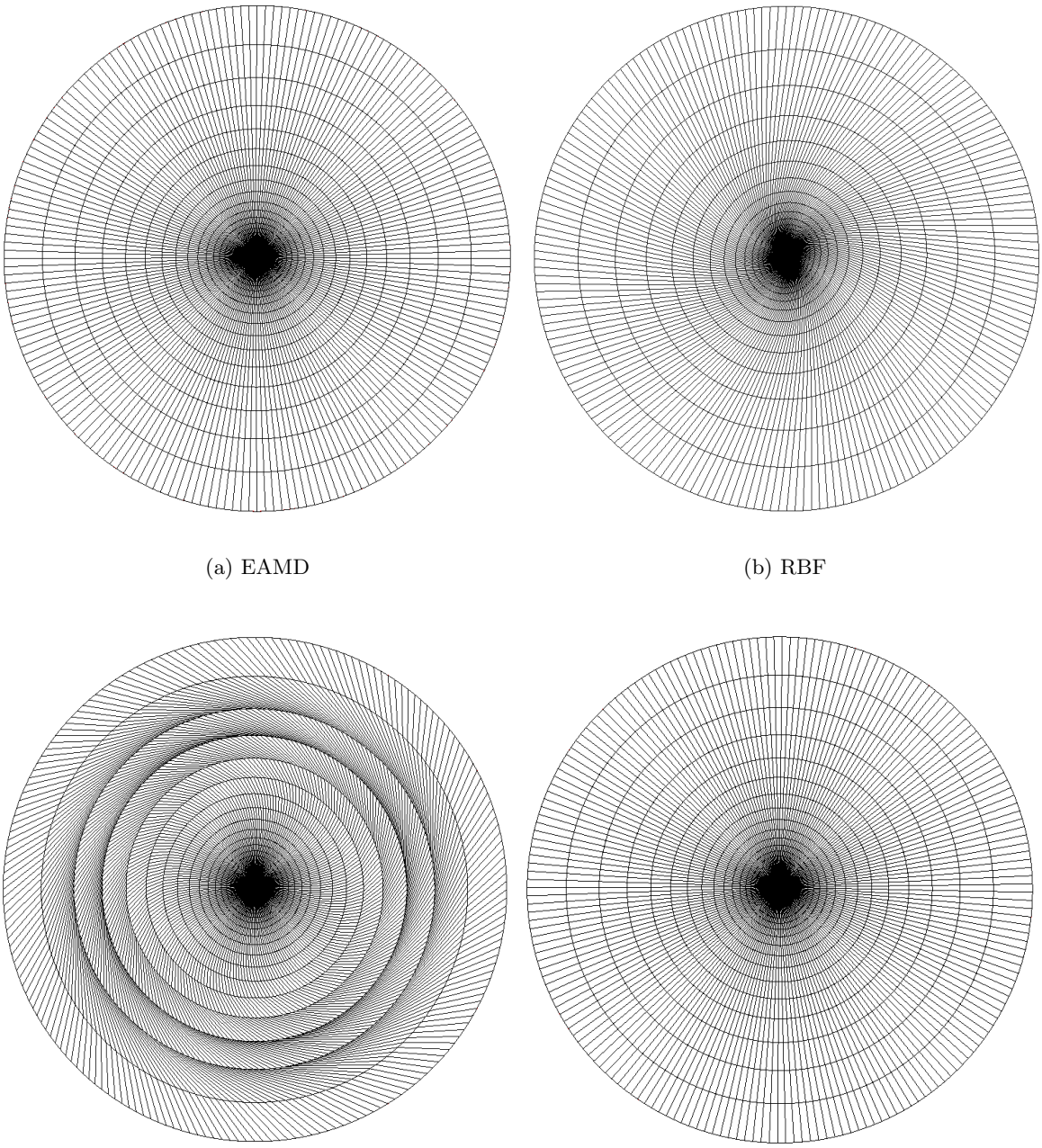


(a) EAMD (b) RBF (c) IDW (exterior fixed)

Figure 7.1: Zoom of the NACA 0012 mesh after deformation.

7.2 Quality

7.2.1 Rotation of 2D NACA 0012 Airfoil



(a) EAMD (b) RBF

(c) IDW (exterior fixed) (d) IDW (exterior sliding)

Figure 7.2: Mesh of the NACA 0012 test case after deformation.

The resulting meshes for the NACA 0012 test case are shown in figures 7.1 (zoomed in at the

airfoil) and 7.2 (complete mesh). It is clear that the elastic analogy method is not able to withstand such a large rotation in one step. The trailing edge of the airfoil pierces through the mesh and 3178 negative cells are formed. This result highlights EAMD's poor performance in capturing rotations. Also in figure 7.2(a) it can be seen that the EAMD does not spread the deformation through the domain, but only tries to deform the mesh locally. This is especially a poor result, because the outer boundary has a sliding boundary condition, but still does not move.

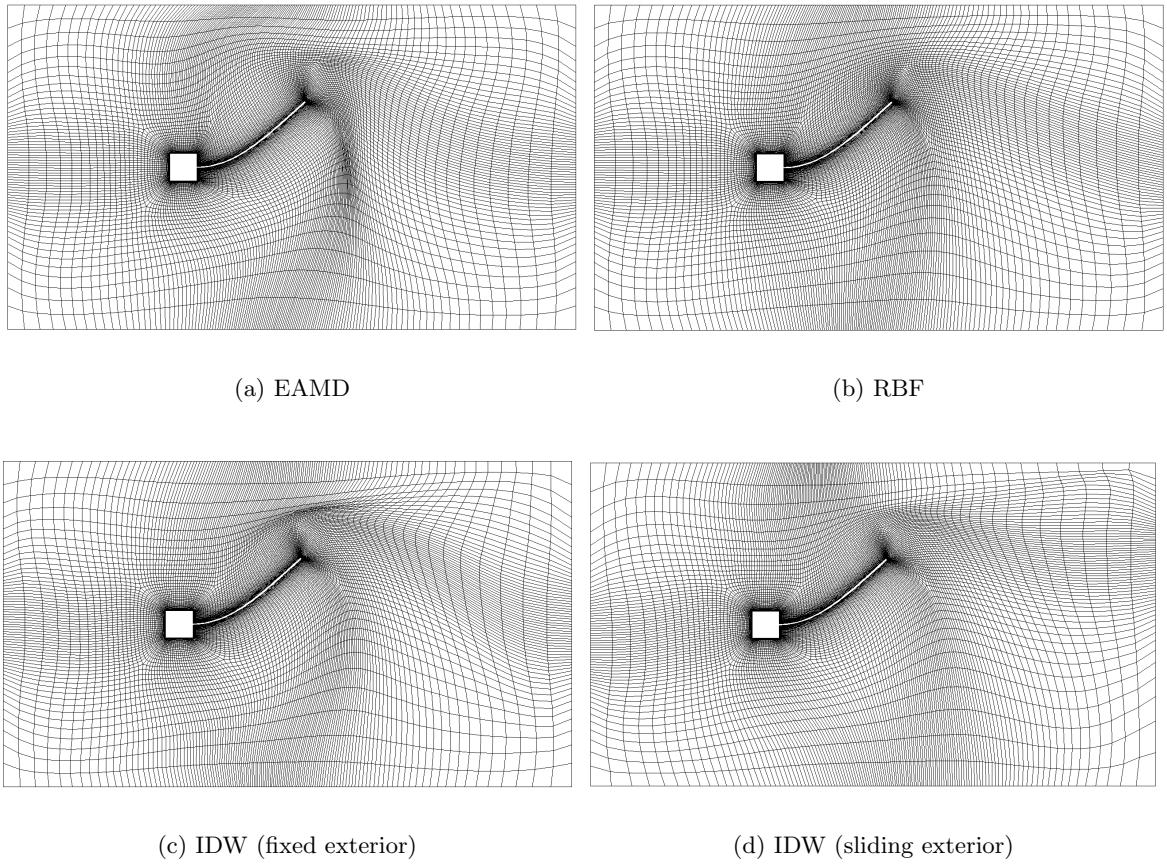
Secondly, when looking at figures 7.1(b) and 7.2(b), it seems that RBF mesh deformation provides much better results. However, due to the presence of very high aspect ratio boundary layer cells, RBF deformation leads to four negative cells. This illustrates that also RBF has difficulty in dealing with very large rotations, especially due to a low robustness in the near-boundary region. This can once more be attributed to the fact that the motion is absorbed locally around the airfoil, and not spread to the exterior boundary.

On the other hand, the IDW mesh deformation leads to a good mesh quality. When a sliding boundary condition is imposed at the exterior boundary (see figure 7.2(d)), the result is even such that the whole mesh is simply rotated by an angle of 90°, which means that the mesh quality remains exactly the same. If the sliding boundary condition would have been implemented for RBF, the same result would have been obtained. However, as mentioned before, implementing the same sliding strategy for RBF, would be more expensive and complex, and has thus not been done so far.

Even without using the sliding boundary feature (figure 7.2(c)), the IDW method produces a high quality mesh, where the deformation is absorbed mainly in the region in the middle between the inner and the outer boundary. The zoom of the airfoil even looks identical for both IDW with or without sliding, which is why figure 7.1(c) is only presented once. In the case with a fixed exterior boundary, the minimum orthogonality remains the same as the initial one (19.74°), whereas the maximum skewness increases from 0.81 to 0.92. Overall, this test cases clearly illustrates IDW's superiority to capture rotations compared to RBF and EAMD. Also the sliding boundary feature clearly shows its advantages.

7.2.2 2D Vortex Induced Beam Vibration

The vortex vibration test case results in valid meshes for all mesh deformation methods. However, there are still important differences between the three methods. First of all, this test case reveals one of the drawbacks of the IDW method. Namely, if all outer boundaries are kept fixed, the inner mesh is quite distorted in the middle between the top boundary and the beam, as shown in figure 7.3(c). The minimum orthogonality in this mesh is 16.23° and the maximum skewness is 0.83. These values are probably too low to ensure a good CFD computation. The problem of the squeezed cells is caused by the explicit nature of the IDW interpolation equation, which propagates the displacement information based on a distance criterion. At some point between the two boundaries, the displacement information coming from one boundary might contradict the information coming from the other boundary. This can cause different nodes over a short interval to move in opposite directions, which leads



(a) EAMD

(b) RBF

(c) IDW (fixed exterior)

(d) IDW (sliding exterior)

Figure 7.3: Deformed mesh of the vortex vibration test case.

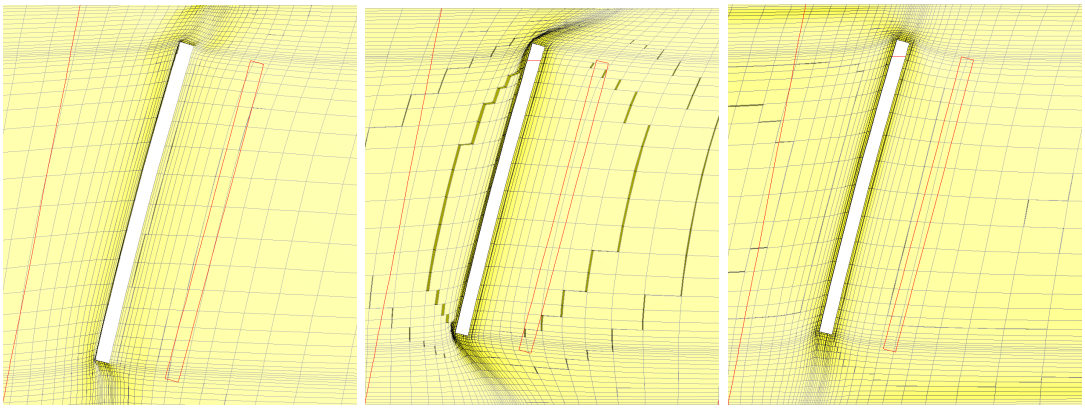
to skewed cells. However, the problem is easily solved by imposing sliding boundaries where necessary. This is shown in figure 7.3(d), where all the outer boundaries are allowed to slide. Especially the right and the top boundary follow the beam movement, leading to a high mesh quality with a minimum orthogonality of 39.07° and a maximum skewness of 0.60.

Next, RBF deformation has no problem to deform the mesh for the vortex vibration test case, as it interpolates the motion very smoothly. The resulting mesh quality is only slightly lower than when IDW with a sliding exterior is used. On the other hand, the EAMD method causes a region of poor quality cells at the bottom right of the beam. Therefore the EAMD has the lowest quality with a minimum orthogonality of 23.88° and a maximum skewness of 0.94, as indicated in figure 7.9.

7.2.3 Elastic Flap in a Duct

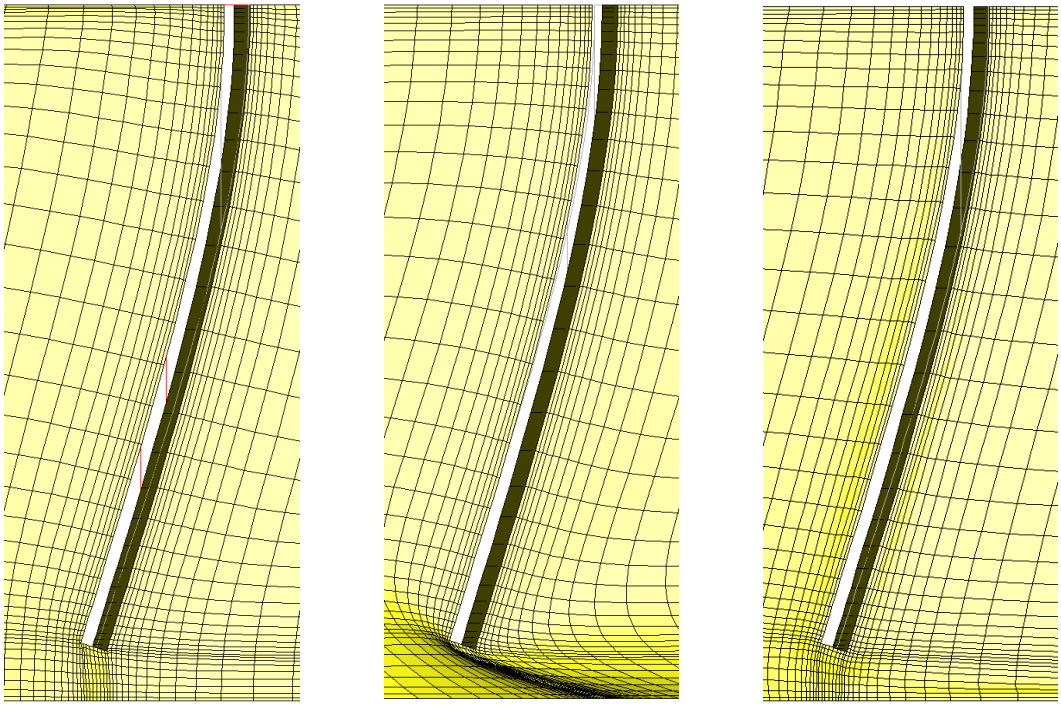
The results of the elastic flap test case are depicted in figures 7.4 and 7.5. These results clearly illustrate the need for sliding boundaries. The RBF method does not have this sliding capability, which means that bottom, top and front face cannot follow the motion of the flap.

As a result, the deformed RBF mesh has 158 negative cells, which occur around the bottom and top of the tip of the elastic flap as can be seen in figure 7.4(b).



(a) EAMD (b) RBF (c) IDW

Figure 7.4: Vertical cut through the tip of the deformed elastic flap.



(a) EAMD (b) RBF (c) IDW

Figure 7.5: Horizontal cut through the middle of the deformed elastic flap.

Thanks to the three sliding faces (top, bottom and front) in the EAMD and IDW test, meshes without negative cells are produced. However, EAMD does produce 21 concave and 24 twisted

cells. This can be attributed to the inability of EAMD to deal with sharp edged objects. For such cases the elasticity coefficients should be extremely high close to the object, relative to the rest of the mesh. It might be that these bad quality cells could have been avoided by using the optimal distribution of elasticity coefficients. However, the search for the optimal distribution of these coefficients is a study of its own, which will not be elaborated upon here.

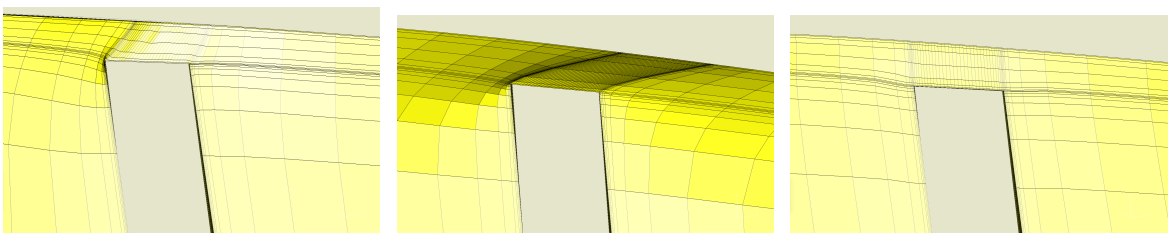
Finally, IDW mesh deformation has no problem to deform the mesh, as can be seen in figures 7.4(c) and 7.5(c). The minimum mesh quality remains high with values of 41.72° and 0.55 for the minimum orthogonality and maximum skewness respectively. This high quality is due to both the sliding boundary and IDW's ability to maintain a high near-boundary mesh quality.

7.2.4 Rotor 67 Blade Deflection

Similar to the elastic flap, the rotor 67 test case highlights the need for sliding boundaries. The resulting mesh in the gap between the blade and the shroud is illustrated in figure 7.6 for the three methods. When looking at the RBF results (figure 7.6(b)), one can see that the deformation of the blade causes the cells to stretch quite heavily. Therefore, 158 negative cells are formed between the blade and shroud, as shown in figure 7.7(a).

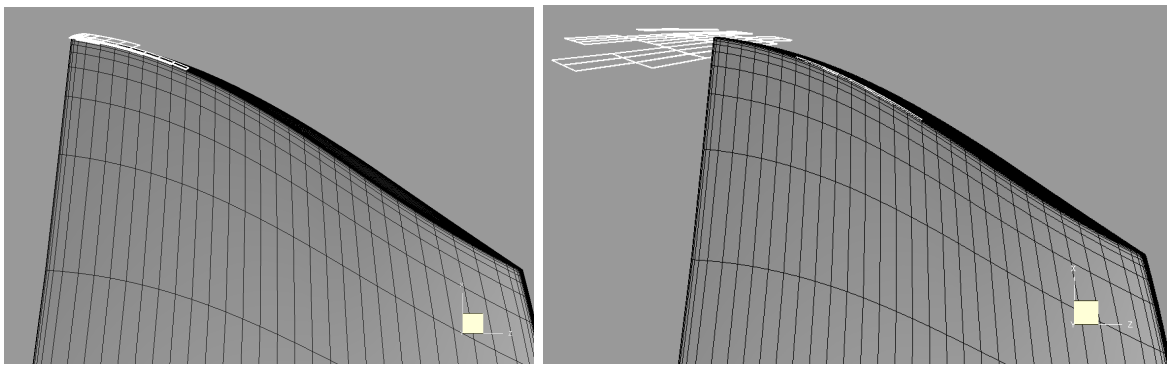
At the same time this test case also shows that a sliding boundary alone in not sufficient, as the EAMD method with sliding shroud still results in a mesh with 54 negative cells (see figure 7.7(b)). These negative cells are caused by two factors. Firstly, there is the limitation of EAMD that the sliding boundary cells are moved according to the definition of a surface of revolution for the shroud. This definition comes from an external file that specifies the shroud radius at various axial locations. However, if the interpolation between these points is different compared to the exact definition of the shroud, negative cells can occur. On the other hand, some negative cells are once again caused by EAMD's lack of smoothness around sharp corners, as illustrated in figure 7.6(a).

Finally, the IDW method results in a good quality mesh, with an orthogonality and skewness values that are nearly identical to the initial mesh as indicated in figure 7.9. This good result is entirely thanks to the fact that the shroud follows the movement of the blade exactly (see figure 7.6(c)).



(a) EAMD (b) RBF (c) IDW

Figure 7.6: Cut through the blade tip of deformed rotor 67 mesh.



(a) RBF (b) EAMD

Figure 7.7: Location of negative cells in the deformed rotor 67 mesh.

7.2.5 AGARD 445.6 Wing Deflection

Finally the AGARD 445.6 test case illustrates the performance of the three methods for a common wing deflection problem. In this problem, the domain is large compared to the wing, meaning that there is space for the deformation to be spread through the domain. Therefore this is a typical test problem where a sliding boundary is not necessary. In this case, the RBF and IDW method have a similar performance, with only a negligible difference in minimum mesh quality, as depicted in figure 7.9. More surprisingly, this minimum mesh quality is even slightly better than the initial mesh quality. The EAMD method, on the other hand, struggles with the large deformation in combination with high aspect ratio boundary layer cells. Therefore causing 161 negative cells close to the wing tip, as can be seen in figure 7.8. As such this test case illustrates the RBF's and IDW's superiority when it comes to maintaining high boundary layer quality and spreading the deformation through the domain.

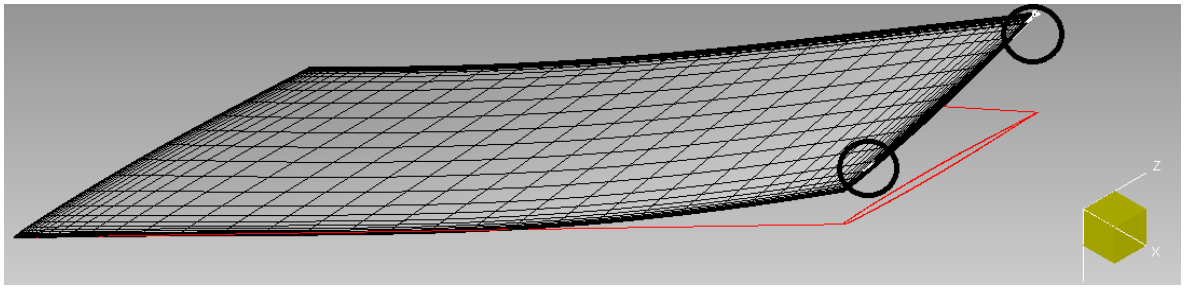


Figure 7.8: Negative cells (white) in the AGARD 445.6 EAMD deformed mesh.

7.2.6 Quality Overview

Finally, a complete overview of the quality for all test cases is given in figure 7.9 (note that for the cases for which no bar is shown the value is equal to zero). From this figure it can be concluded that IDW always results in a higher than or equal quality with respect to RBF and EAMD. Especially IDW's ability to maintain a high near-boundary quality and to spread the deformation far into the volume mesh, has revealed to be important properties of IDW. For cases where sliding boundaries are not essential, such as the AGARD 445.6 and the vortex vibration problem, the RBF also results in a high quality. If a comparison would have been made for all the test cases, without using sliding boundaries, RBF might have come out the strongest, since its interpolation function is smoother than the one for IDW. However, the fact that RBF is not an explicit interpolation makes it more difficult to use the same sliding strategy as introduced in this thesis. The current sliding strategy requires several steps where interpolations are based on different sets of data points. Applying the same method to the RBF deformation would mean that for each set of data points, the system for the interpolation function has to be solved, which is a costly operation. Finally, the EAMD method has proven to be of inferior quality compared to the other two methods, mainly caused by poor deformation of small cells.

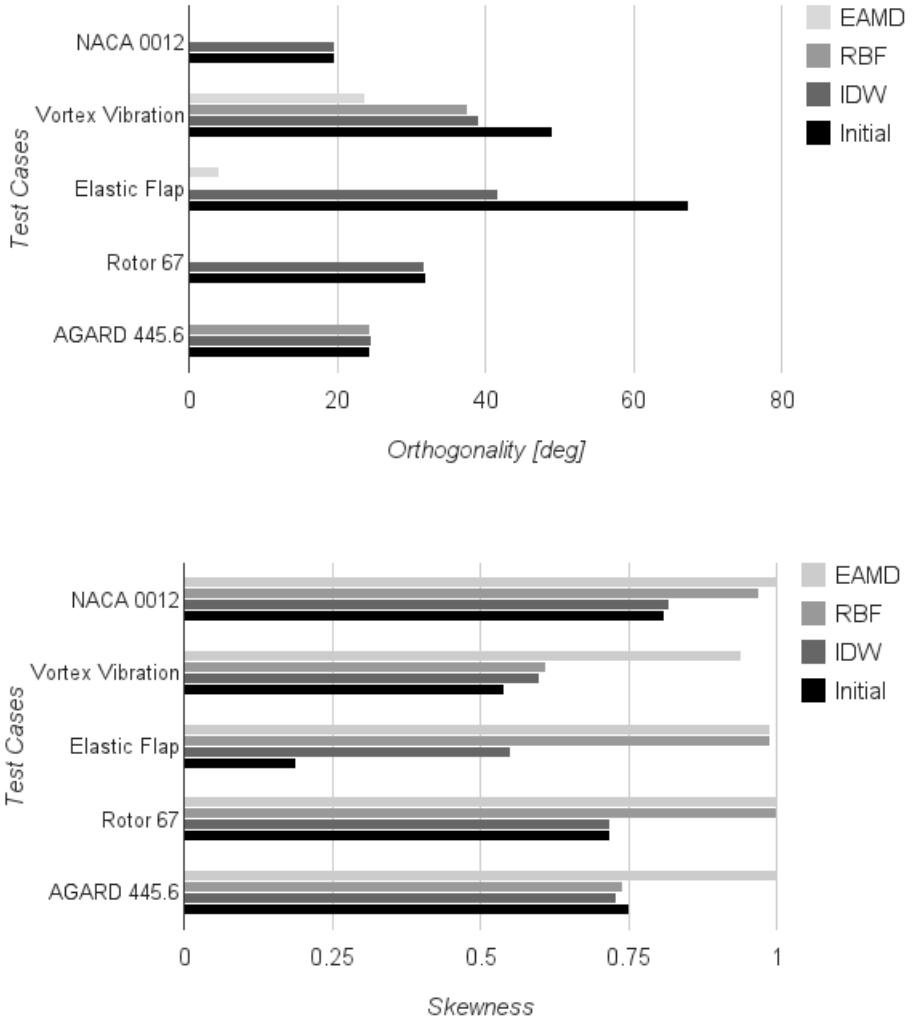


Figure 7.9: Quality comparison of the mesh deformation methods.

7.3 Computational Cost

Next to the quality of the deformed mesh, attention should be paid to the computational cost of the method. These costs are summarized for all tests in table 7.3. A first note that should be made, is that in order to have a fair comparison, no coarsening has been applied to either RBF nor IDW. The reader interested in the performance of IDW with coarsening is referred to chapter 6.2. Secondly, it should be noted that the cost of RBF is split into the matrix inversion phase to solve the system and the evaluation phase of the interpolation function. This is split up, because usually the matrix inversion is only done at the start of the computation.

Firstly, it is clear that both RBF and IDW are much faster than EAMD for 2D test cases. This is mainly caused by the iterative nature of EAMD which requires to solve the system several times until convergence is reached. RBF also appears to be faster than IDW in 2D. This is probably caused by an implementation difference, where the 2D case in IDW is treated as a 3D case with a zero z-component. For RBF a 2D case only has two coordinates, making the evaluation faster.

When looking at the 3D cases, however, the IDW method is clearly the fastest one. Even when the time to solve the RBF system would not be taken into account, IDW is still about three times faster than RBF. Although the time difference between IDW and EAMD has become proportionally smaller, IDW is still more than six times faster than EAMD for all test cases. The largest mesh presented in this table is the AGARD 445.6 one, with 373,569 nodes. This is still smaller than most industrial test cases, and for extremely large cases with several millions of cells, EAMD might be faster than the full IDW method. However taking into account that for such a large case IDW should be done with coarsening, IDW will still be the most efficient method.

Finally it is important to realise that for RBF, not only the large matrix inversion time of more than one hour, but also the memory required to do this inversion, is a limiting factor. The used memory during the AGARD 445.6 RBF mesh deformation computation is 5.0 GB, whereas this is only 818 MB for the IDW method. Also the size of the matrix stored on disk is 4.2 GB for the AGARD 445.6 test case. Moreover the true advantage of IDW will become clear when parallel computations are considered. The evaluation phase for both IDW and RBF are easily implemented in parallel. However, for the RBF it is difficult to implement the matrix inversion in parallel. This means that in a parallel computation, the IDW will perform even better compared to RBF.

Table 7.3: Time measurement comparison for RBF, EAMD and IDW.

Test case	RBF solution	RBF evaluation	EAMD	IDW
NACA0012	0.08s	0.30s	3m 25.73s	0.45s
Vortex vibration	0.43s	0.44s	1m 04.63s	0.88s
Elastic flap	1h 49m 44.64s	1m 39.40s	7m 6.6s	37.67s
Rotor 67	1h 57m 17.78s	1m 14.87s	40m 1.67s	28.14s
AGARD 445.6	13h 32m 39.16s	10m 15.25s	18m 34.10s	2m 44.30s

7.4 Modal CFD Simulation of Vortex Induced Beam Vibration

This test illustrates the effect of the chosen mesh deformation method on the CFD computation. Both the RBF and IDW mesh deformation method are applied to the unsteady modal computation of the vortex induced beam vibration, with parameters as indicated in appendix C.1. For both the IDW and RBF method absolute displacements are used. This means that the quality of the mesh will not deteriorate during the simulation as explained in section 5.5. As a result the minimum mesh quality is reached at the maximum tip displacement. These results have already been presented in section 7.2.2. The minimum quality of the IDW mesh at maximum displacement is just slightly higher than for the RBF, as illustrated before in figures 7.9(a) and 7.9(b).

The tip displacement of the beam during the simulation is shown in figure 7.10. It can be seen that the two plots are nearly identical. The only difference being that the amplitude of the motion is slightly higher for the IDW result. The difference between the maximum tip displacements is equal to 0.05 cm. Moreover when comparing the total pressure in figures 7.11, it can be seen that the pressure fields are nearly identical. Therefore it can be concluded that the mesh deformation method only has a small influence on the aerodynamic results for this test case, thanks to the fact that both RBF and IDW maintain a high mesh quality during the complete simulation.

In terms of efficiency it has already been mentioned before that RBF is faster than IDW for a 2D simulation due to an implementation difference. The RBF treats a 2D case by only using two coordinates, whereas the IDW treats a 2D case as a 3D case with three coordinates, where the last coordinate is always zero. The total time spent in the IDW mesh deformation module is equal to 19m 26.33s and for RBF this is 11m 22.80s. This leads to an average time per deformation step of 1.17 s for IDW and 0.68 s for RBF. With respect to the total CFD computation time, IDW is still very efficient as the total time to deform the mesh is only 3% of the total time to perform the CFD simulation which is 10h 29m 6.59s.

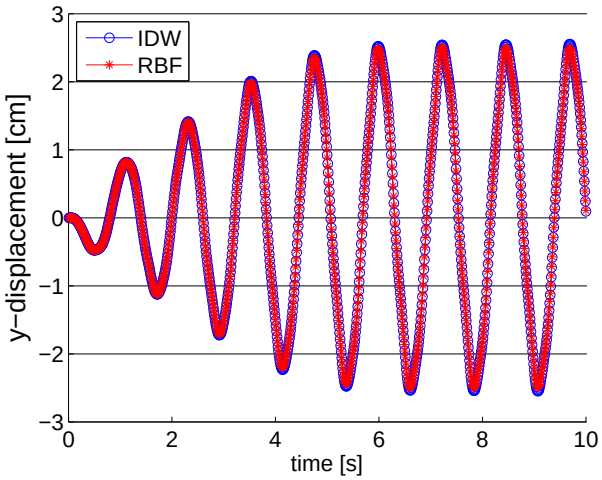
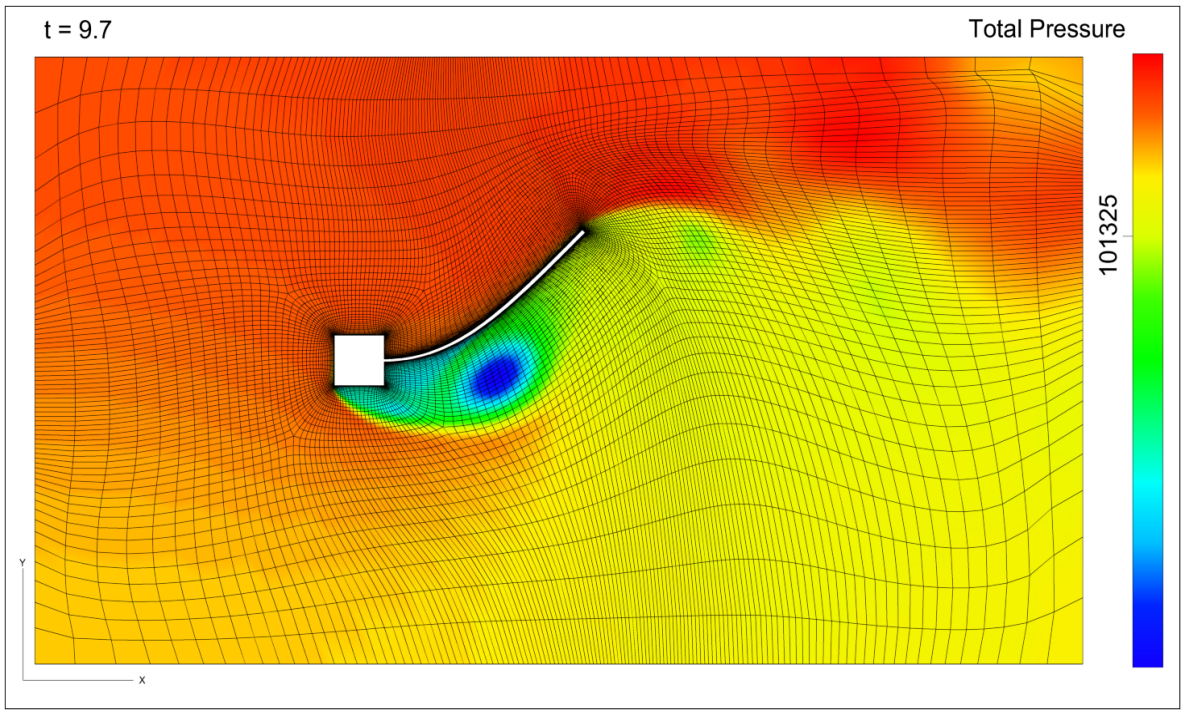
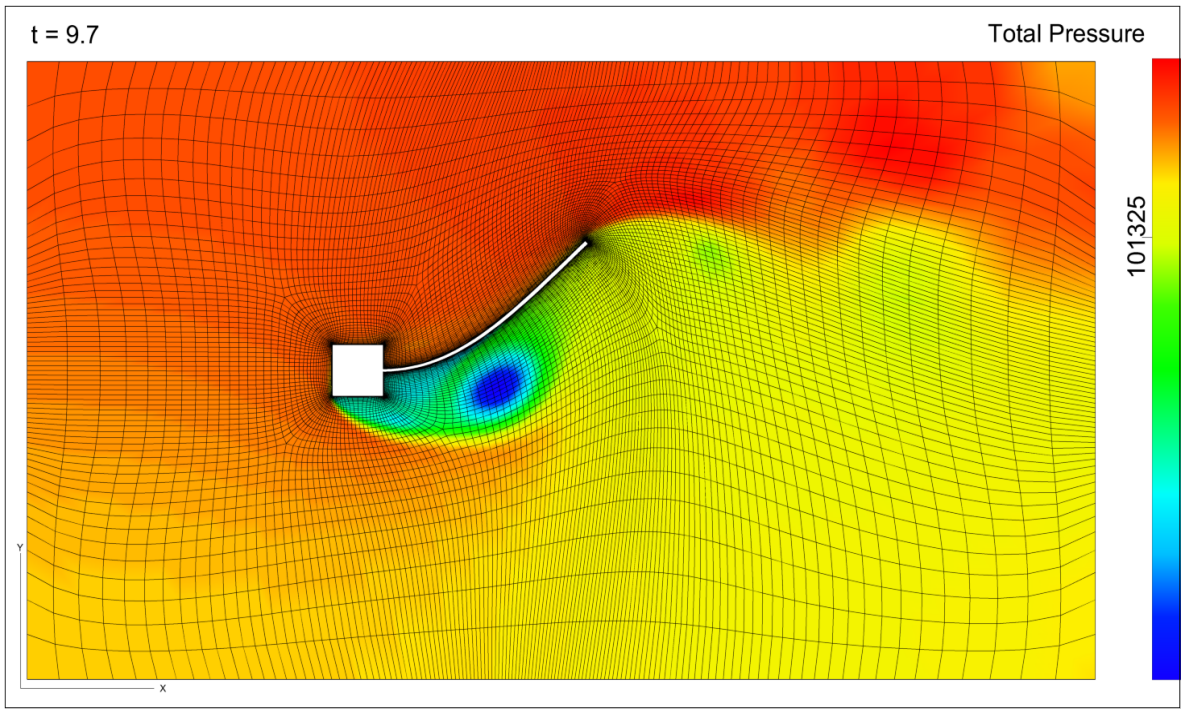


Figure 7.10: Tip displacement comparison for IDW and RBF mesh deformation in the vortex induced beam vibration test case.

.



(a) IDW



(b) RBF

Figure 7.11: Total pressure during the vortex induced beam vibration simulation at time $= 9.7\,\mathrm{s}$.

7.5 Modal CFD Simulation of AGARD 445.6 Wing Deflection

This last test illustrates the efficiency of the IDW method with coarsening in a full CFD simulation. An unsteady simulation of the coarse mesh AGARD 445.6 wing is done, using a modal analysis with the first six mode shapes of the wing. The CFD simulation settings are given in appendix C.4. For the coarsening parameters criterion 4 is chosen with values $k_1 = 1$ and $k_2 = 0.01$. The resulting tip displacement during the simulation is shown in figure 7.12. These results are comparable to the tip displacement as presented in [15].

The average time per IDW deformation step is equal to 29.29 s. Note that the deformation time varies between time steps due to the fact that coarsening is applied. The greedy algorithm will select more nodes for larger displacements, leading to a higher deformation time. The average time per deformation step is only 7% of the time to perform one CFD time step, which is equal to 7m 09.87s. This leads to the conclusion that IDW with coarsening can be applied efficiently during FSI simulations.

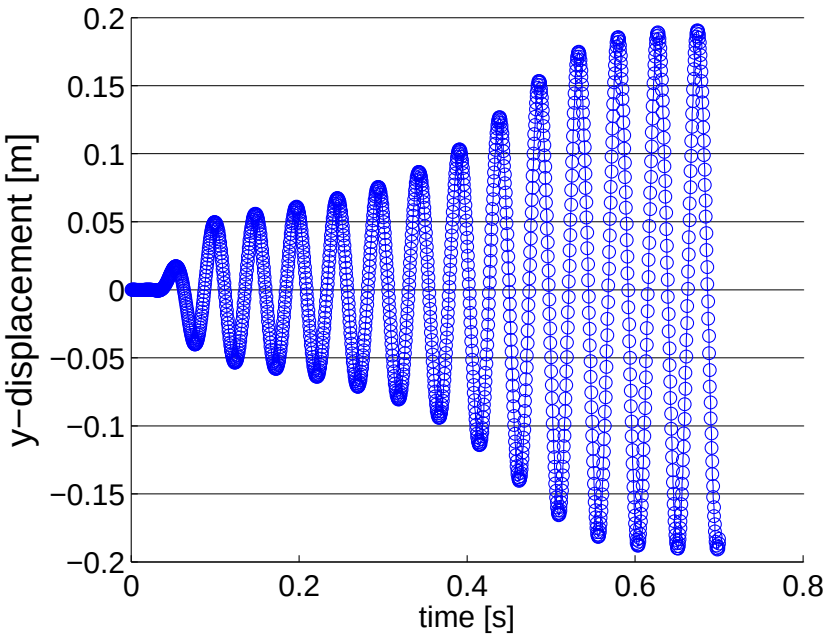


Figure 7.12: Deflection of the wing tip trailing edge during the unsteady, coarse mesh AGARD 445.6 simulation

Chapter 8

Conclusions and Recommendations

In this thesis it has been attempted to find the best possible mesh deformation method for use in industrial fluid-structure interaction problems with any type of unstructured mesh. The goal was to identify, develop and test a method that is both robust, efficient and user-friendly. From this research the following conclusions and recommendations have been drawn.

8.1 Conclusions

A mesh deformation algorithm based on IDW interpolation, incorporating sliding boundary nodes and boundary nodes coarsening, has been developed and tested. The algorithm is robust, efficient and user-friendly and applicable to a large variation of FSI problems of industrial scale.

Firstly, a literature review concerning existing mesh deformation methods highlighted that many mesh deformation algorithms have been designed in the past. These methods can be divided into point-by-point schemes, mesh connectivity schemes and hybrid schemes. However, so far, none of these methods have come forward as the best technique, as they all have their specific drawbacks. One relatively new method did stand out thanks to its high robustness in combination with a low computational cost and complexity. This method is the inverse distance weighting (IDW) interpolation method.

IDW is a point-by-point technique, which can easily be implemented in parallel and applied to any type of mesh. The method interpolates the displacements into the volume mesh by means of an interpolation function that depends on the inverse of the distance between the boundary and the inner nodes. Thanks to the explicit nature of the IDW function, it is not necessary to solve a system, which is the expensive part of the RBF mesh deformation method.

It has been shown that by including boundary node rotations, the deformed mesh quality is increased compared to when the displacements are simply treated as translation. The rotations are represented by quaternions. Several interpolation methods for the quaternions were identified. The method where the displacement due to rotation is interpolated for each volume node, performs the best for general bending and deformation problems. For problems where very large or pure rotations occur, the linear interpolation of the quaternions should be used.

A new strategy to impose a sliding boundary condition has been proven to significantly increase the robustness of the method for deformation problems where there is only a small space between the moving and the outer boundaries. A common example of such a problem is the deformation of blades within turbo-machinery components. For the rotor 67 test case, the mesh quality after deformation of the blade virtually stays the same as the original mesh quality, when the shroud is treated as a sliding boundary. On the other hand, without this sliding condition, negative cells are formed.

An efficiency improvement, which coarsens the boundary mesh by means of a greedy algorithm, has been implemented. Even though the original IDW method is already very efficient, the cost still grows linearly with the amount of boundary nodes. One deformation step of a test problem with approximately 3 million cells of which over 100,000 are boundary nodes, takes about two hours to complete in serial mode. When applying the coarsening strategy to the same test case, the computation time can drop down to six minutes while maintaining the same mesh quality. Alternatively, the computation time can drop even further to less than one minute, by allowing a slightly lower resulting mesh quality. The boundary coarsening method makes the IDW scheme efficient for large-scale industrial problems. Moreover, the deformation time becomes negligible compared to the CFD computation.

Compared to the radial basis function and the elastic analogy method, the IDW method has superior near-boundary mesh quality. This allows to have a high quality deformation of high-Reynolds numbers meshes. Furthermore the overall minimum mesh quality is of a comparable level to the very robust RBF method. In problems that include large boundary rotations or small gaps between the moving and the outer boundary, the IDW method even proves to be superior than RBF and EAMD. Also the IDW computation time is significantly lower than both RBF and EAMD, even without applying boundary mesh coarsening. When only the function evaluation phases of RBF and IDW are compared, IDW is up to three times faster than RBF for 3D problems. Moreover IDW does not require an expensive matrix inversion at the start of the simulation, which means that several hours of computation time and several gigabytes of memory can be saved. Also compared to the EAMD method, IDW is at least 7 times faster for all test cases.

Moreover, IDW has proven to be the only method capable to deform the mesh for all five of the presented test problems. As such it has been proven to be a widely applicable method. Also the application to the different test cases required a low amount of user input. Only the sliding boundaries have to be chosen and sometimes the α-parameters or rotation method can be changed to increase mesh quality. The variation of these parameters is intuitive and straightforward.

8.2 Recommendations

Even though the presented IDW scheme could already be released in a commercial software, there are still several improvements that could be considered in future research

- Currently IDW is only implemented in a serial mode. Its true performance will become apparent when the method is implemented in parallel. The most costly part of the method is the computation of the displacement of the inner mesh nodes, which takes up nearly 100% of the total computation time. Since this step consists of a simple double loop over all inner and active outer boundary nodes, it can easily be coded in parallel by dividing the inner nodes over several processors, while the information about the boundary nodes is sent to each processor.

- Section 5.5 already briefly introduced a comparison between absolute and relative displacements. It could be recommended to investigate an automatic procedure to choose and switch between the two methods. For example, the relative method can be used during a simulation and each time the shape of the object approaches the initial position, the mesh deformation is reset by an absolute displacement step.

- Even though the boundary coarsening method has been proven to provide very good results for IDW, another option could be to use the kd-tree method of Luke et al [38]. At the moment there is no information about which of the two methods would perform the best. However, the advantage of the coarsening method remains that it can be applied to both RBF and IDW.

- Also the option to use a local radius for the IDW interpolation could be investigated further, especially in case one is interested in optimisation problems where the geometry is only deformed locally. For FSI problems this method is less relevant.

- In order to further increase the robustness of IDW, it could be investigated whether different power parameters should be used for rotations and translations as proposed by Witteveen [70].

- When coarsening is applied in an unsteady simulation, the selected boundary nodes at two consecutive time steps will be similar. As such it might not be necessary to perform the full greedy scheme at each iteration. Section 6.4 briefly introduced some possibilities for coarsening across multiple time steps. However so far this method is only useful for increasing deformations and the gain in computation time is small. Further methods could be investigated, such as performing the coarsening every N time steps, and re-using the active boundary in between.

- For modal analysis CFD simulations, the mode shapes could be used to perform efficient mesh deformations. Once the interpolated displacement for each of the mode shapes is known, the values for the generalised displacements can be used to add the different deformed positions together in order to obtain the final deformed mesh at each time step. This would mean that IDW only has to be applied once for each mode shape at the start of the deformation, which would make this method extremely efficient.

- Since the IDW method is more efficient without the use of boundary node rotations it could be useful to first deform the complete mesh, without rotations, and then apply a rotation optimisation step only to the cell layers closest to the boundaries in order to ensure a good mesh quality there. This is based on the IDW mesh optimisation scheme as presented by Witteveen in [70].

Bibliography

[1] C. Allen. Parallel universal approach to mesh motion and application to rotors in forward flight. *International Journal for Numerical Methods in Engineering*, 69(10):2126–2149, 2007.

[2] S. L. Altmann. *Rotations, quaternions, and double groups*. DoverPublications. com, 2005.

[3] P. Bar-Yoseph, S. Mereu, S. Chippada, and V. Kalro. Automatic monitoring of element shape quality in 2-d and 3-d computational mesh dynamics. *Computational Mechanics*, 27(5):378–395, 2001.

[4] J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA paper*, 115:1989, 1989.

[5] Y. Bazilevs, V. Calo, Y. Zhang, and T. Hughes. Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38(4-5):310–322, 2006.

[6] R. Beatson and N. G.N. Fast evaluation of radial basis functions: part i. *Advances in Computational Mathematics*, 11:253–270, 1999.

[7] H. Bijl, A. van Zuijlen, A. de Boer, and D. Rixen. Fluid-structure interaction (wb1417) - an introduction to numerical coupled simulation. Lecture Notes, Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands, May 2008.

[8] F. J. Blom. Considerations on the spring analogy. *International Journal for Numerical Methods in Fluids*, 32(6):647–668, 2000.

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

[10] D. E. Breen, S. Mauch, and R. T. Whitaker. 3d scan conversion of csg models into distance volumes. In *Volume Visualization, 1998. IEEE Symposium on volume visualisation*, pages 7–14, New York, NY, USA, 1998.

[11] G. Chiandussi, G. Bugeda, and E. Oñate. A simple method for automatic update of finite element meshes. *Communications in Numerical Methods in Engineering*, 16(1):1–19, 2000.

[12] E. B. Dam, M. Koch, and M. Lillholm. *Quaternions, interpolation and animation.* Datalogisk Institut, Københavns Universitet, 1998.

[13] A. de Boer. *Computational fluid-structure interaction.* PhD thesis, Technische Universiteit Delft, 2008.

[14] A. De Boer, M. Van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11):784–795, 2007.

[15] F. Debrabandere. *Computational methods for industrial fluid-structure interactions.* PhD thesis, Université de Mons, 2014.

[16] J. Donea, S. Giuliani, and J. Halleux. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1):689–723, 1982.

[17] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering*, 163(1):231–245, 1998.

[18] R. Franke. Scattered data interpolation: tests of some methods. *Mathematics of computation*, 38(157):181–200, 1982.

[19] Fraunhofer Institute for Algorithms and Scientific Computing (SCAI). *MpCCI 4.2.1-3 Documentation*, 2012.

[20] T. Gerhold and J. Neumann. The parallel mesh deformation of the dlr tau-code. In *New Results in Numerical and Experimental Fluid Mechanics VI*, pages 162–169. Springer, 2008.

[21] P. Girodroux-Lavigne, J. Grisval, S. Guillemot, M. Henshaw, A. Karlsson, V. Selmin, J. Smith, E. Teupootahiti, and B. Winzell. Comparison of static and dynamic fluid-structure interaction solutions in the case of a highly flexible modern transport aircraft wing. *Aerospace science and technology*, 7(2):121–133, 2003.

[22] M. Glück, M. Breuer, F. Durst, A. Halfmann, and E. Rank. Computation of fluid-structure interaction on lightweight structures. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(14):1351–1368, 2001.

[23] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[24] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[25] B. T. Helenbrook. Mesh deformation using the biharmonic operator. *International journal for numerical methods in engineering*, 56(7):1007–1021, 2003.

[26] S.-Y. Hsu and C.-L. Chang. Mesh deformation based on fully stressed design: the method and 2-d examples. *International Journal for Numerical Methods in Engineering*, 72(5):606–629, 2007.

[27] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & fluids*, 36(6):1119–1136, 2007.

[28] A. A. Johnson and T. E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer methods in applied mechanics and engineering*, 119(1):73–94, 1994.

[29] D. B. Kholodar, S. A. Morton, and R. M. Cummings. Deformation of unstructured viscous grids. *AIAA paper*, 926:2005, 2005.

[30] K. Kovalev. *Unstructured hexahedral non-conformal mesh generation*. PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium, December 2005.

[31] D. S. C. Kowollik, M. C. Haupt, and P. Horst. Mesh deformation with exact surface reconstruction using a reduced radial basis function approach. In *International Conference on Computational Methods for Coupled Problems in Science and Engineering*. Institute of Aircraft Design and Lightweight Structures (IFL), 2013.

[32] E. Lefrançois. A simple mesh deformation technique for fluid–structure interaction based on a submesh approach. *International Journal for Numerical Methods in Engineering*, 75(9):1085–1101, 2008.

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

[34] Y. Liu, Z. Guo, and J. Liu. Rbfs-msa hybrid method for mesh deformation. *Chinese Journal of Aeronautics*, 25(4):500–507, 2012.

[35] O. Livne and G. Wright. Fast multilevel evaluation of 1d piecewise smooth radial basis function expansions. In *SIAM Conference on Geometric Design and Computing*, 2006.

[36] R. Löhner and C. Yang. Improved ale mesh velocities for moving bodies. *Communications in numerical methods in engineering*, 12(10):599–608, 1996.

[37] G. Y. Lu and D. W. Wong. An adaptive inverse distance weighting spatial interpolation technique. *Computers & Geosciences*, 34(9):1044–1055, September 2008.

[38] E. Luke, E. Collins, and E. Blades. A fast mesh deformation method using explicit interpolation. *Journal of Computational Physics*, 231(2):586–601, 2012.

[39] D. R. Lynch and K. O'Neill. Elastic grid deformation for moving boundary problems in two space dimensions. *Finite elements in water resources*, 2, 1980.

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

[41] D. Martineau and J. Georgala. A mesh movement algorithm for high quality generalised meshes. *AIAA Paper*, 614:2004, 2004.

[42] D. Maruyama, D. Bailly, and G. Carrier. High quality mesh deformation using quaternions for orthogonality preservation. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Nashville, Tenessee, January 2012. ONERA.

[43] D. R. McDaniel and S. A. Morton. Efficient mesh deformation for computational stability and control analyses on unstructured viscous meshes. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pages 2009–1363, 2009.

[44] G. D. Nayer. *Interaction fluide-structure pour les corps élancés*. PhD thesis, Ecole Central de Nantes, 2008.

[45] M. Ninyerola, X. Pons, and J. M. Roure. Monthly precipitation mapping of the iberian peninsula using spatial interpolation tools implemented in a geographic information system. *Theoretical and Applied Climatology*, 89(3-4):195–209, 2007.

[46] NUMECA International. *User Manual FINE^{TM}/Open v2.12 (Including OpenLabs) Flow Integrated Environment*. Chausse de la Hulpe 189, B-1170 Brussels, BELGIUM, December 2012.

[47] Numeca International. Fine^{TM}/open with openlabs, 2013. http://www.numeca.com/en/products/finetmopen-openlabs, accessed 26 September 2013.

[48] Numeca International. Hexpress^{TM}: Unstructured full-hexahedral meshing, 2013. http://www.numeca.com/en/products/automeshtm/hexpresstm, accessed 26 September 2013.

[49] NUMECA International. *User Manual HEXPRESS ^{TM} v2.12 Unstructured Grid Generator*. Chausse de la Hulpe 189, B-1170 Brussels, BELGIUM, March 2013.

[50] NUMECA International. *User Manual HEXPRESS^{TM}/Hybrid*. Chausse de la Hulpe 189, B-1170 Brussels, BELGIUM, 2013.

[51] Open Engineering. Oofelie::multiphysics, 2012. http://www.open-engineering.com/index.php/eng/Products/OOFELIE-Multiphysics2, accessed 14 July 2013.

[52] G. Papari and N. Petkov. Reduced inverse distance weighting interpolation for painterly rendering. In *Computer Analysis of Images and Patterns*, pages 509–516. Springer, 2009.

[53] G. Parwatha. *Development of Elastic Analogy Mesh Deformation (EAMD) in FINE/Open*. Numeca International, 2012.

[54] T. Rendall and C. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228(17):6231–6249, 2009.

[55] T. Rendall and C. Allen. Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors. *International journal for numerical methods in engineering*, 81(1):89–105, 2010.

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

[57] T. Robinson and G. Metternicht. Testing the performance of spatial interpolation techniques for mapping soil properties. *Computers and electronics in agriculture*, 50(2):97–108, 2006.

[58] J. A. Samareh et al. Application of quaternions for mesh deformation. *NASA TM*, 211646:2002, 2002.

[59] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[60] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *ACM National Conference*, pages 517–524, Cambridge, Massachusetts, 1968. Harvard College.

[61] K. Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH computer graphics*, 19(3):245–254, 1985.

[62] D. Stadler, F. Kosel, D. Čelič, and A. Lipej. Mesh deformation based on artificial neural networks. *International Journal of Computational Fluid Dynamics*, 25(8):439–448, 2011.

[63] A. J. Strazisar, J. R. Wood, M. D. Hathaway, and K. L. Suder. Laser anemometer measurements in a transonic axial-flow fan rotor. *NASA STI/Recon Technical Report N*, 90:11245, 1989.

[64] S. Sun and B. Chen. An algebraic deformation approach for moving grid based on barycentric coordinates. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4. IEEE, 2010.

[65] G. Q. Tabios and J. D. Salas. A comparative analysis of techniques for spatial interpolation of precipitation. *JAWRA Journal of the American Water Resources Association*, 21(3):365–380, 1985.

[66] University of Washington Computer Science and Engineering. K-d trees, 2002.

[67] G. Van Kuik and J. Dekker. The flexhat program, technology development and testing of flexible rotor systems with fast passive pitch control. *Journal of Wind Engineering and Industrial Aerodynamics*, 39(1):435–448, 1992.

[68] A. van Zuijlen, A. de Boer, and H. Bijl. Higher-order time integration through smooth mesh deformation for 3d fluid–structure interaction simulations. *Journal of Computational Physics*, 224(1):414–430, 2007.

[69] G. Wang, H. H. Mian, Z.-Y. Ye, and J.-D. Lee. An improved point selection method for hybrid-unstructured mesh deformation using radial basis functions. In *21st AIAA computational Fluid Dynamics Conference*, San Diego, CA, June 2013.

[70] J. A. Witteveen. Explicit and robust inverse distance weighting mesh deformation for cfd. *AIAA Paper*, 165, 2010.

[71] Z. Yang and D. J. Mavriplis. Unstructured dynamic meshes with higher-order time integration schemes for the unsteady navier-stokes equations. *AIAA paper*, 1222(2005):1, 2005.

[72] E. C. Yates Jr. Agard standard aeroelastic configurations for dynamic response i-wing 445.6. Technical report, DTIC Document, 1988.

[73] D. Zeng and C. R. Ethier. A semi-torsional spring analogy model for updating unstructured meshes in 3d moving domains. *Finite Elements in Analysis and Design*, 41(11):1118–1139, 2005.

[74] X. Zhou and S. Li. A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing. *Journal of Computational Physics*, 2012.

Appendix A

Tree Code Optimisation

In order to understand the tree-code optimisation it is first necessary to briefly introduce what kd-trees are. A kd-tree or k-dimensional tree, is a data structure that organises spatial data in order to be able to perform faster processing for specific problems such as a nearest neighbour search or a fast look-up. The kd-tree is organised in such a way that data clusters are found. An example of a kd-tree is shown in figure A.1. Starting from the complete set of data points, the data is split in two sets of equal number of data points along the axis of widest spread. This process is repeated until each set consists of a defined number of nodes. The bottom level partition consists of the leaves of the tree. The partitions at all other levels are called the vertices of the tree.

The kd-tree method requires the approximation of the effect of a group of boundary nodes. Therefore the following sum will have to be approximated

$$f(\boldsymbol{x}) = \sum_{b=1}^{n_b} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_b\|^p}, \tag{A.1}$$

Now it is possible to find a small set of pseudo nodes, that approximates the influence of a group of nodes at a distance. When choosing four pseudo nodes, the sum of equation A.1 can be approximated as follows

$$f(\boldsymbol{x}) \approx \frac{1}{4}\left(\frac{1}{\|\boldsymbol{x} - \boldsymbol{q}_1\|^3} + \frac{1}{\|\boldsymbol{x} - \boldsymbol{q}_2\|^3} + \frac{1}{\|\boldsymbol{x} - \boldsymbol{q}_3\|^3} + \frac{1}{\|\boldsymbol{x} - \boldsymbol{q}_4\|^3}\right), \tag{A.2}$$

where $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$ are the location of the pseudo nodes, which will be called quad points or QPs. One property of the QPs is that they have to have the same center as the collection of boundary nodes that they approximate. Therefore the location of $\boldsymbol{q}_4$ can be computed, once the other three QP locations are known

$$\boldsymbol{q}_4 = 4\left(\sum_{b=1}^{n_b} \boldsymbol{x}_b\right) - (\boldsymbol{q}_1 + \boldsymbol{q}_2 + \boldsymbol{q}_3)\,. \tag{A.3}$$

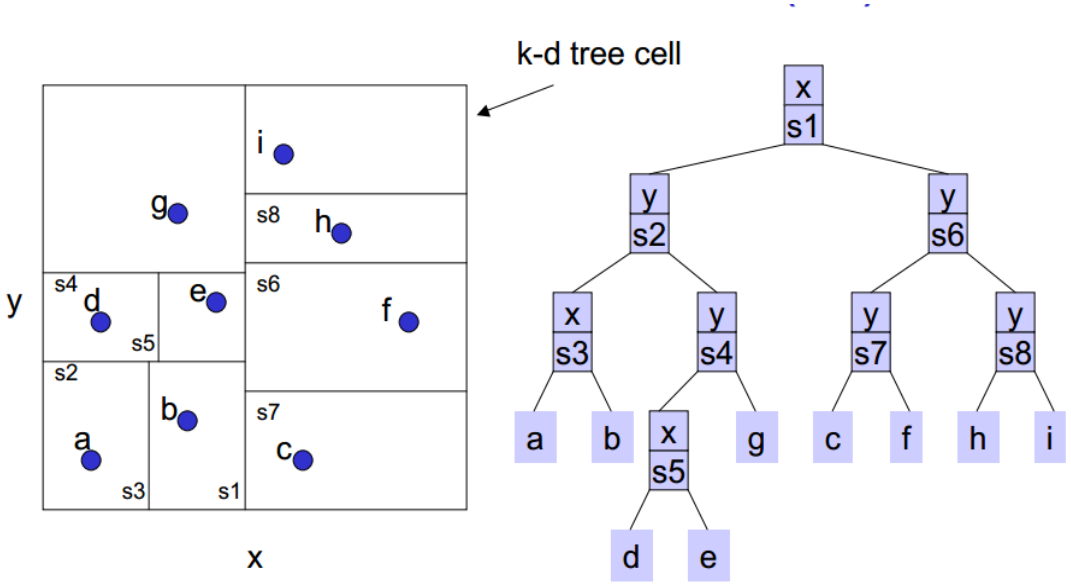


Figure A.1: Example of kd-tree [66]

The other three QP locations will have to be computed by means of a non-linear least-squares method which minimises the approximation error given by

$$err = \left\| f(\boldsymbol{x}) - \frac{1}{4}\left(\frac{1}{\|\boldsymbol{x}-\boldsymbol{q}_1\|^p} + \frac{1}{\|\boldsymbol{x}-\boldsymbol{q}_2\|^p} + \frac{1}{\|\boldsymbol{x}-\boldsymbol{q}_3\|^p} + \frac{1}{\|\boldsymbol{x}-\boldsymbol{q}_4\|^p}\right)\right\|^2 \tag{A.4}$$

In order to be able to minimise the error, a test sphere is created which has a radius $r = 3R_e$, where R_e is the radius of the smallest enclosing sphere that contains all the nodes $\boldsymbol{x}_b$ with the center located at the center of the collection of nodes. On this test sphere 20 nodes are selected, being the corners of a dodecahedron that is circumscribed by the test sphere. In figure A the procedure of finding the QPs is illustrated. The larger blue dots represent the computed QPs, for the collection of nodes illustrated by the smaller white dots. Once the location of the QPs are known, the errors of the approximation are sampled at different locations on different test sphere radii. This provides a model for the approximation errors.

Next to finding the location of the QPs it is also important to find the value of the rotation and translation component at the QPs. In order to assign a rotation and translation component to the four QPs, a term-by-term least squares fit is done, which again preserves the centroidal value.

From the presented approximations, it is possible to build a fast kd-tree evaluator. The architecture of the optimised IDW mesh deformation method is illustrated in figure A. The blocks with a double frame are only executed once, at the start of the computation. The other blocks are computed at each deformation step. Firstly, a kd-tree of all the boundary nodes is constructed. In order to reduce the depth of the tree, each leaf is a bin of 32 boundary nodes. Then for each tree vertex the four QP locations are computed. Next the error estimates of

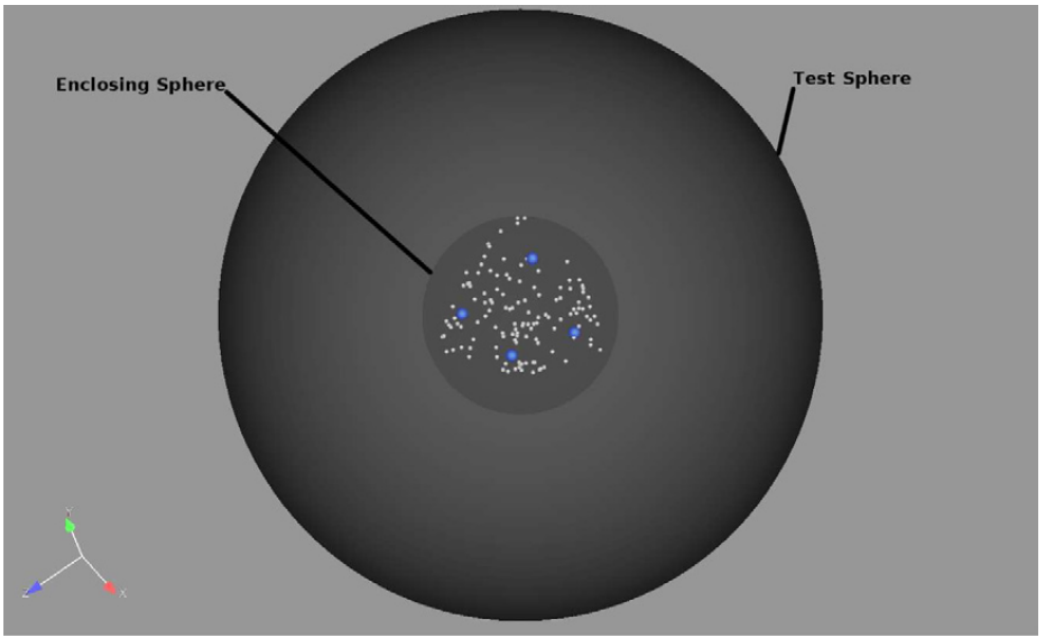


Figure A.2: Illustration of QP computation [38]

the QP approximations have to be computed. Hence for each kd-tree vertex the QPs, the sphere radius and the measured errors are stored.

At each simulation step the rotation (M) and translation (b) component have to be computed for each QP, together with the corresponding error estimate. Once all components of the approximation are known, the fast evaluation per volume node can start. For each volume node the kd-tree is descended recursively. At a vertex of the tree it is checked whether the error of the approximation meets a certain tolerance. If this is the case the QPs are used, otherwise the tree has to be descended further. Note that when descending the tree, the closest children are visited first, because they will have the largest impact on the error.

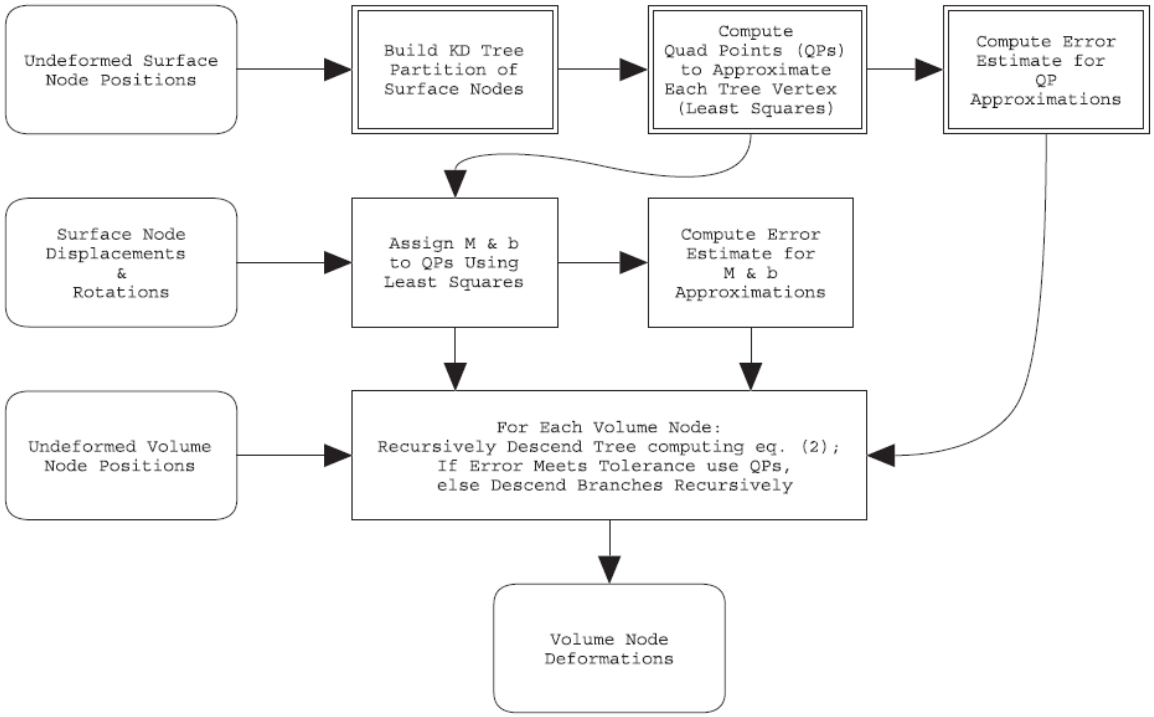


Figure A.3: KD-tree optimisation software architecture [38]

Appendix B

Coarsening Results

B.1 Elastic Flap

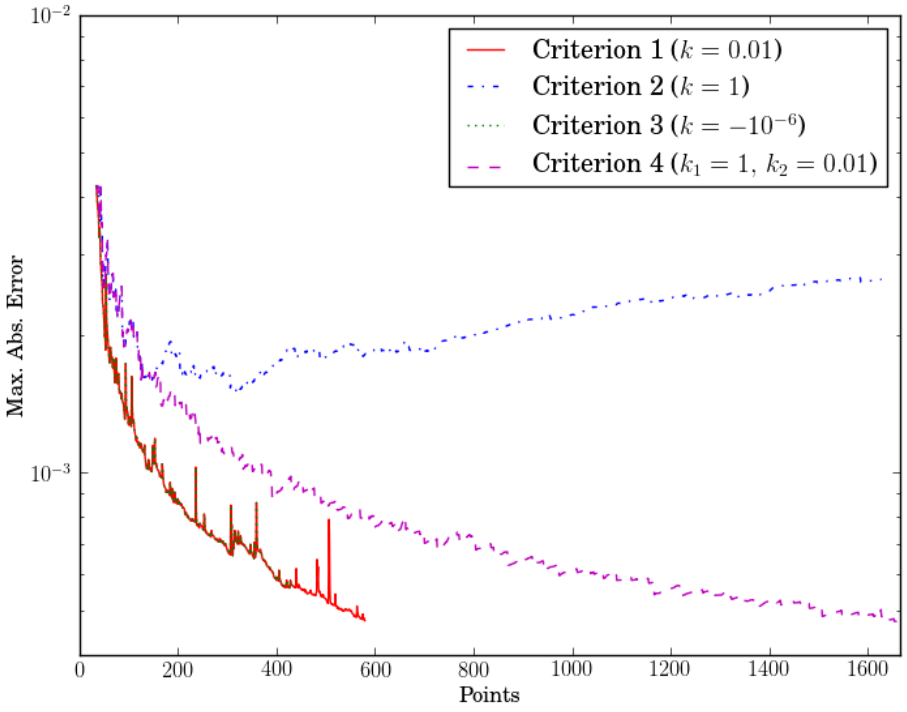
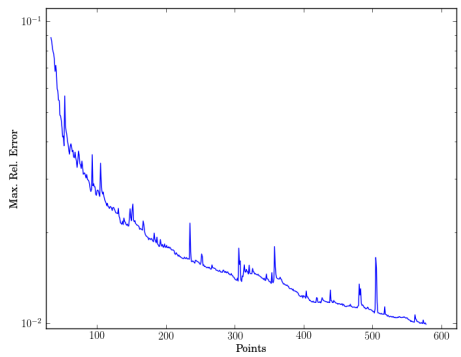
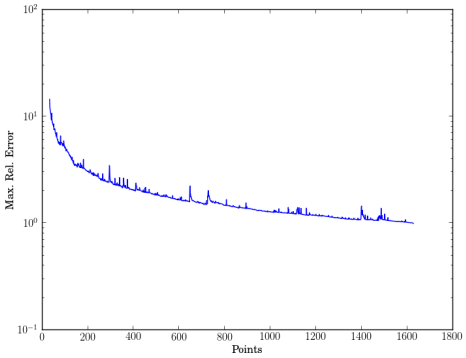


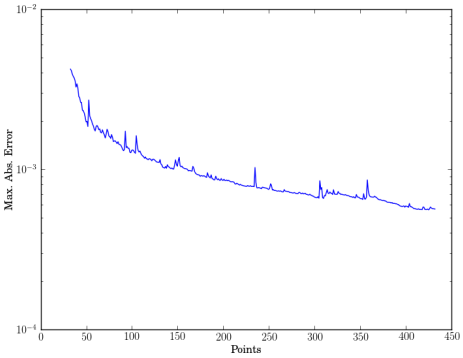
Figure B.1: Comparison of the absolute errors of the different greedy criteria for the elastic flap test case.



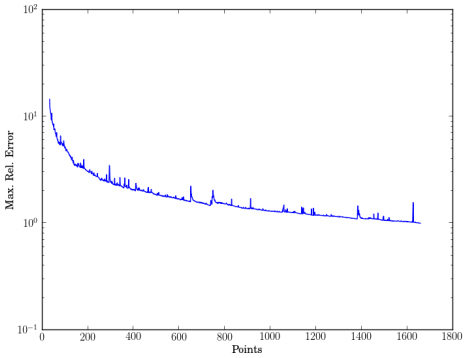
(a) Criterion 1 ($k = 0.01$)



(b) Criterion 2 ($k = 1$)

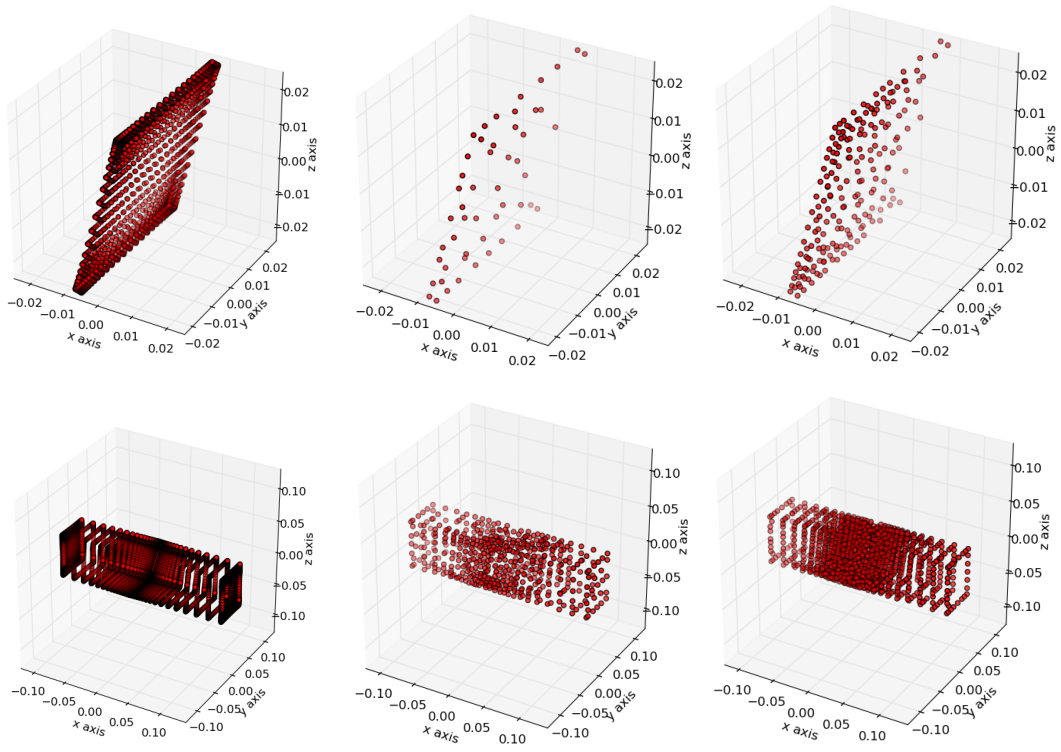


(c) Criterion 3 ($k = -10^{-6}$)

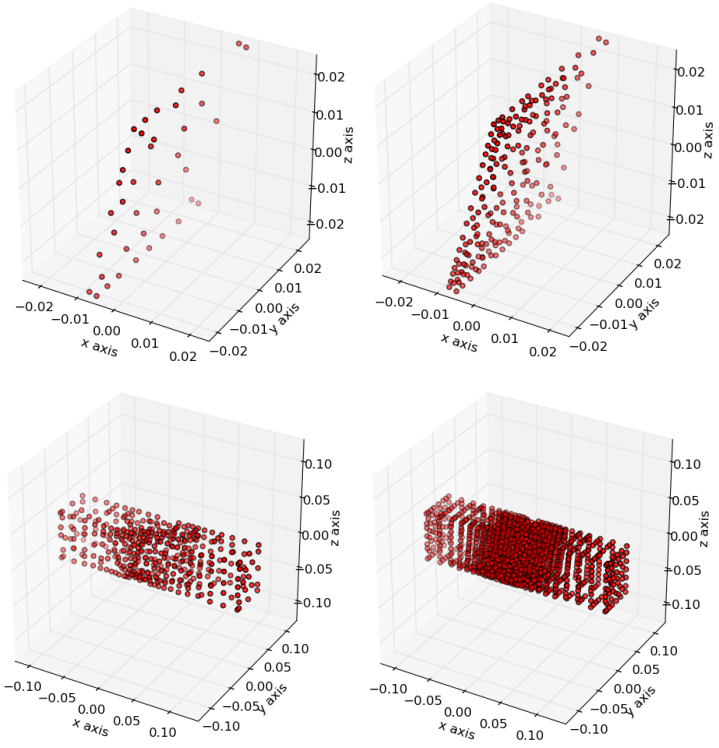


(d) Criterion 4 ($k_1 = 1$, $k_2 = 0.01$)

Figure B.2: Maximum relative error with respect to the number of active boundary nodes for the different greedy criteria for the elastic flap test case.



(a) Without coarsening (b) Criterion 1 ($k = 0.01$) (c) Criterion 2 ($k = 1$)



(d) Criterion 3 ($k = -10^{-6}$) (e) Criterion 4 ($k_1 = 1$, $k_2 = 0.01$)

Figure B.3: Comparison of the selected nodes for the elastic flap test case between the different greedy criteria (top = flap, bottom = channel).

B.2 Rotor 67 Coarse Mesh

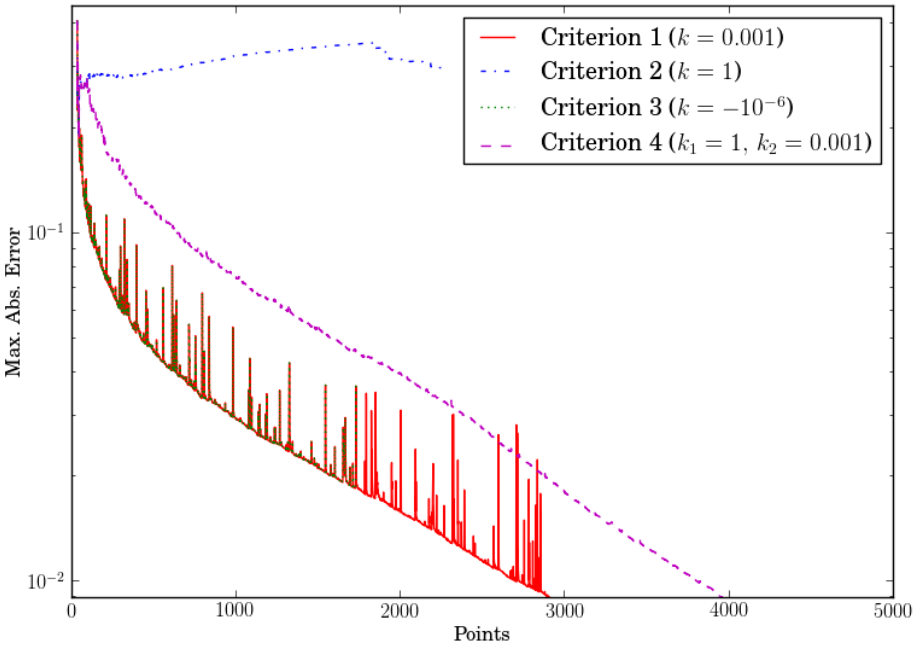
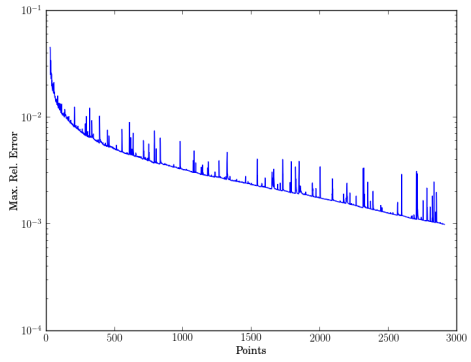
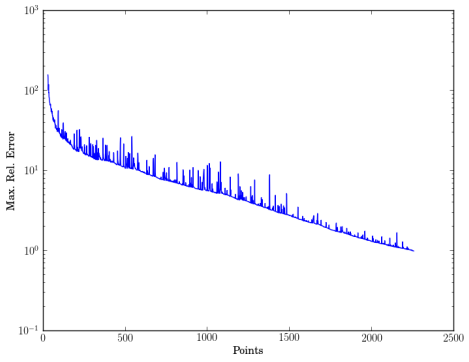


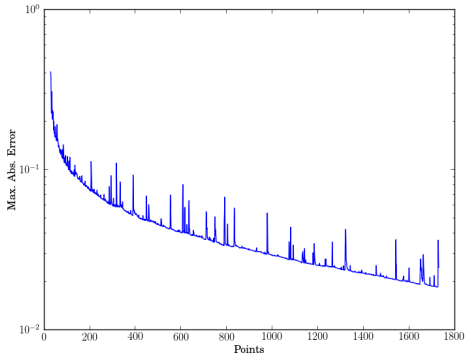
Figure B.4: Comparison of the absolute errors of the different greedy criteria for the coarse mesh rotor 67 test case.



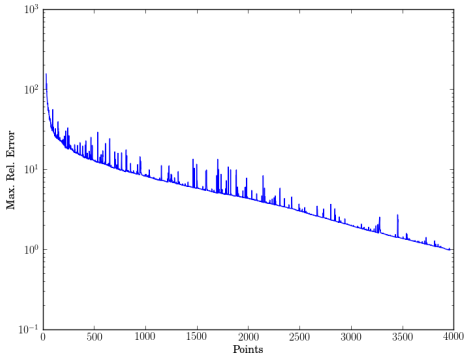
(a) Criterion 1 ($k = 0.001$)



(b) Criterion 2 ($k = 1$)

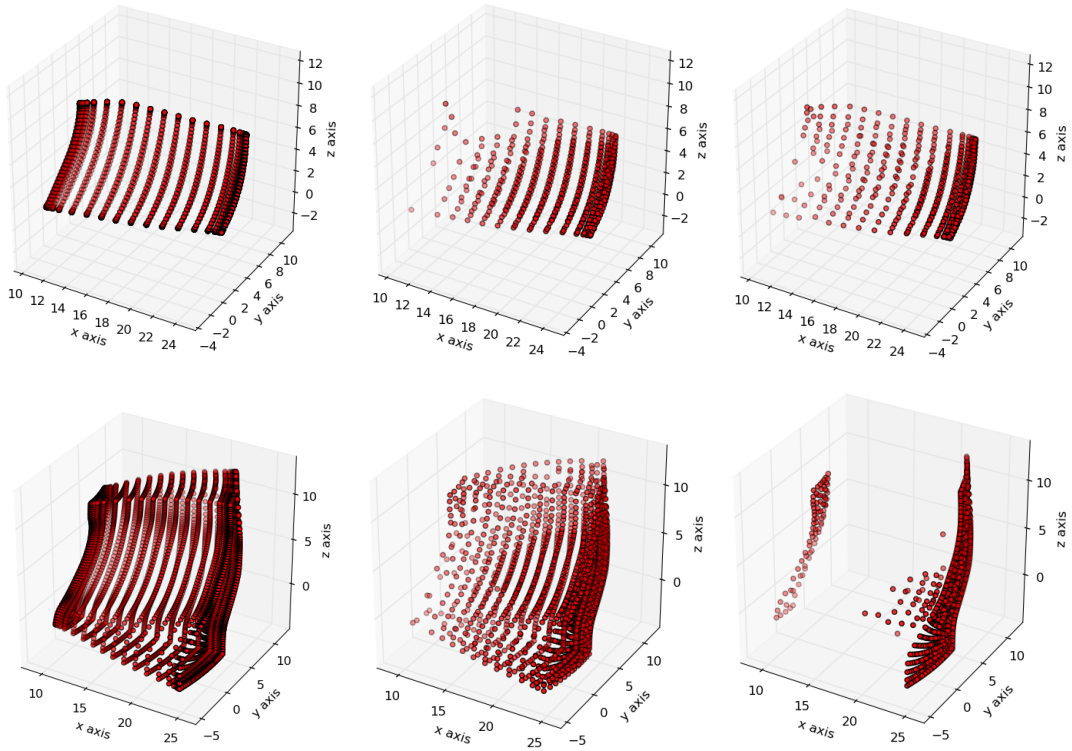


(c) Criterion 3 ($k = -10^{-6}$)

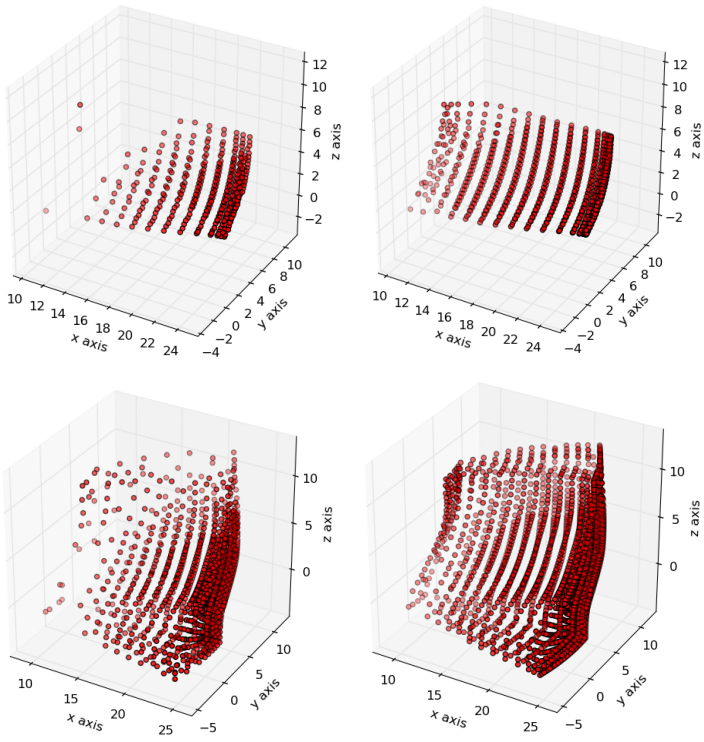


(d) Criterion 4 ($k_1 = 1$, $k_2 = 0.001$)

Figure B.5: Maximum relative error with respect to the number of active boundary nodes for the different greedy criteria for the coarse mesh rotor 67 test case.



(a) Without coarsening (b) Criterion 1 ($k = 0.001$) (c) Criterion 2 ($k = 1$)



(d) Criterion 3 ($k = -10^{-6}$) (e) Criterion 4 ($k_1 = 1$, $k_2 = 0.001$)

Figure B.6: Comparison of the selected nodes for the coarse mesh rotor 67 test case between the different greedy criteria (top = blade, bottom = exterior boundary).

B.3 Rotor 67 Fine Mesh

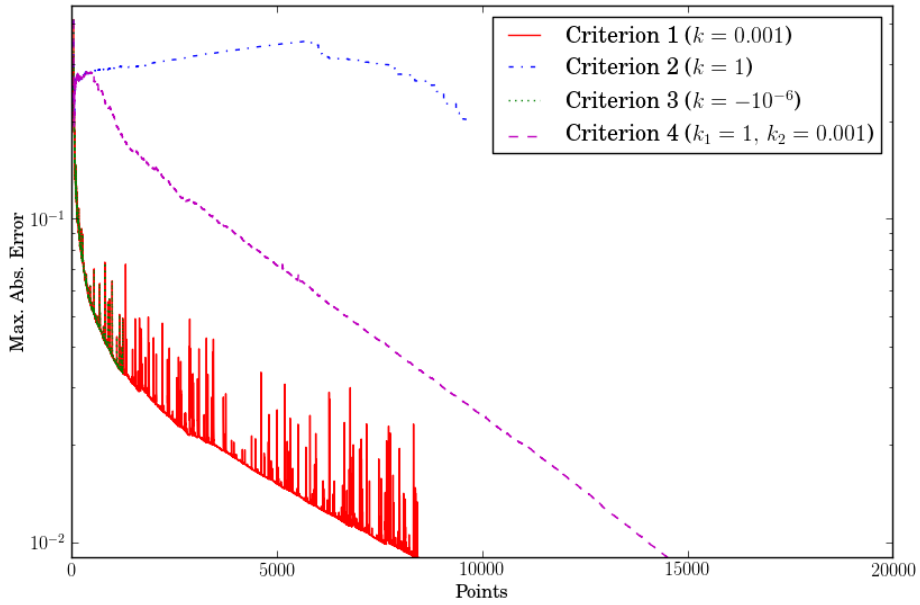
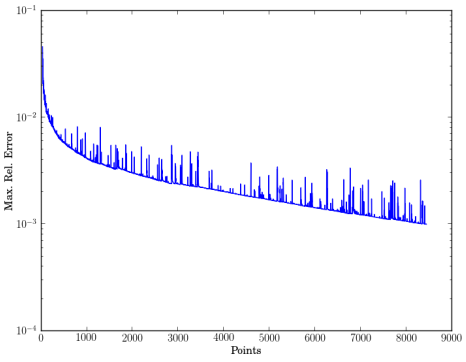
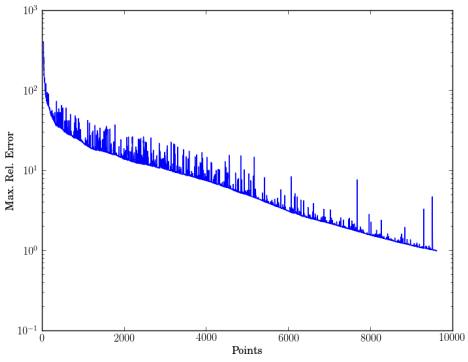


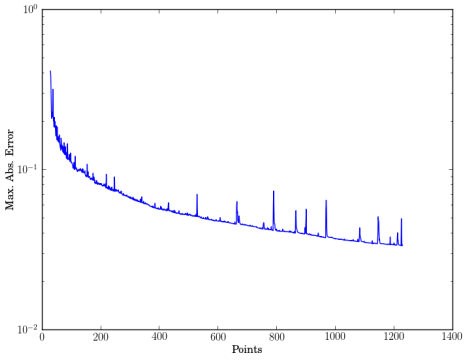
Figure B.7: Comparison of the absolute errors of the different greedy criteria for the fine mesh rotor 67 test case.



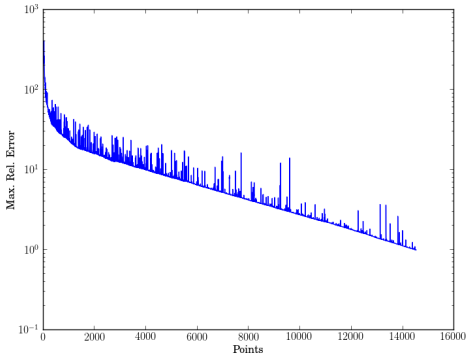
(a) Criterion 1 ($k = 0.001$)



(b) Criterion 2 ($k = 1$)

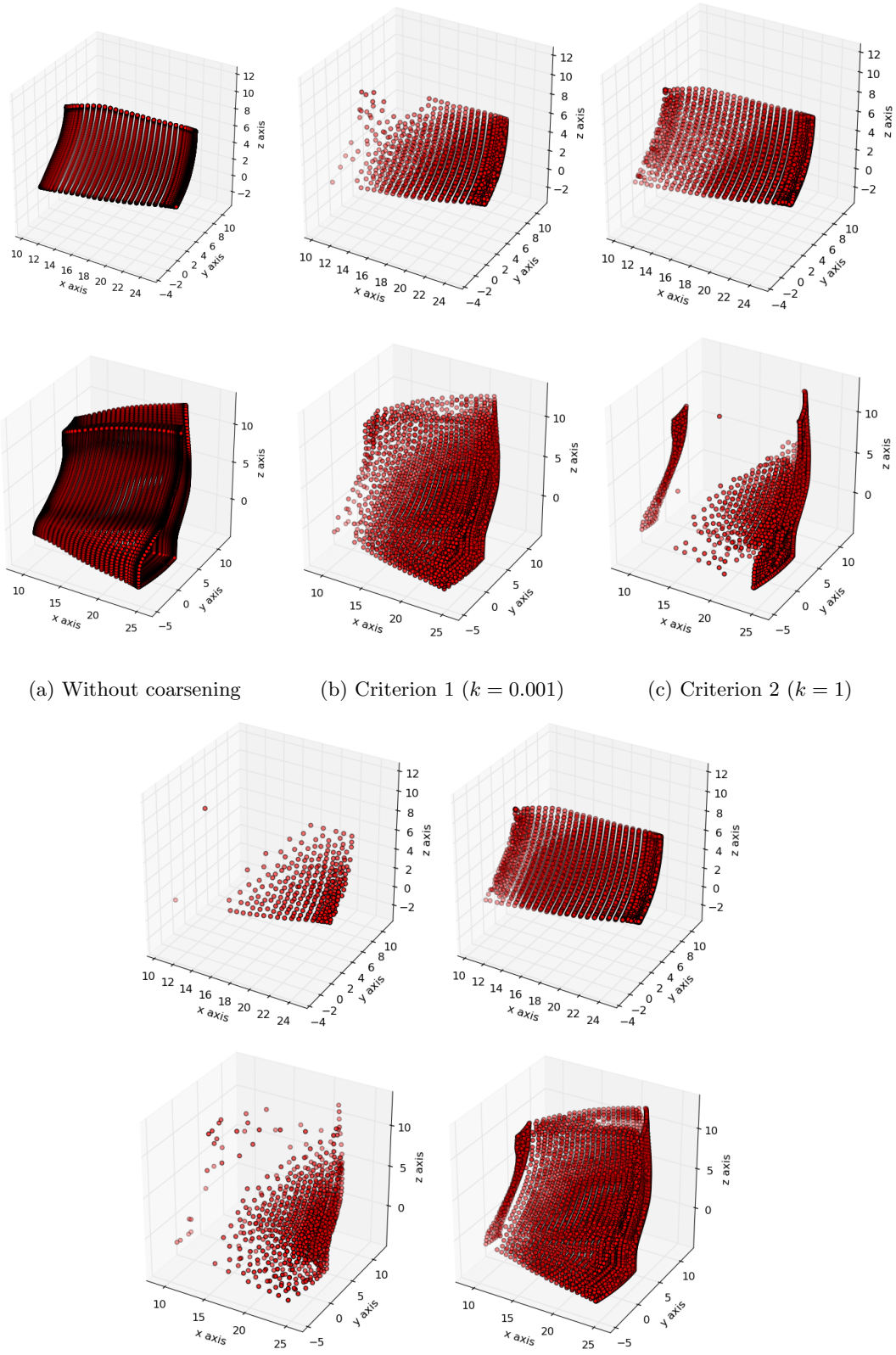


(c) Criterion 3 ($k = -10^{-6}$)



(d) Criterion 4 ($k_1 = 1$, $k_2 = 0.001$)

Figure B.8: Maximum relative error with respect to the number of active boundary nodes for the different greedy criteria for the fine mesh rotor 67 test case.

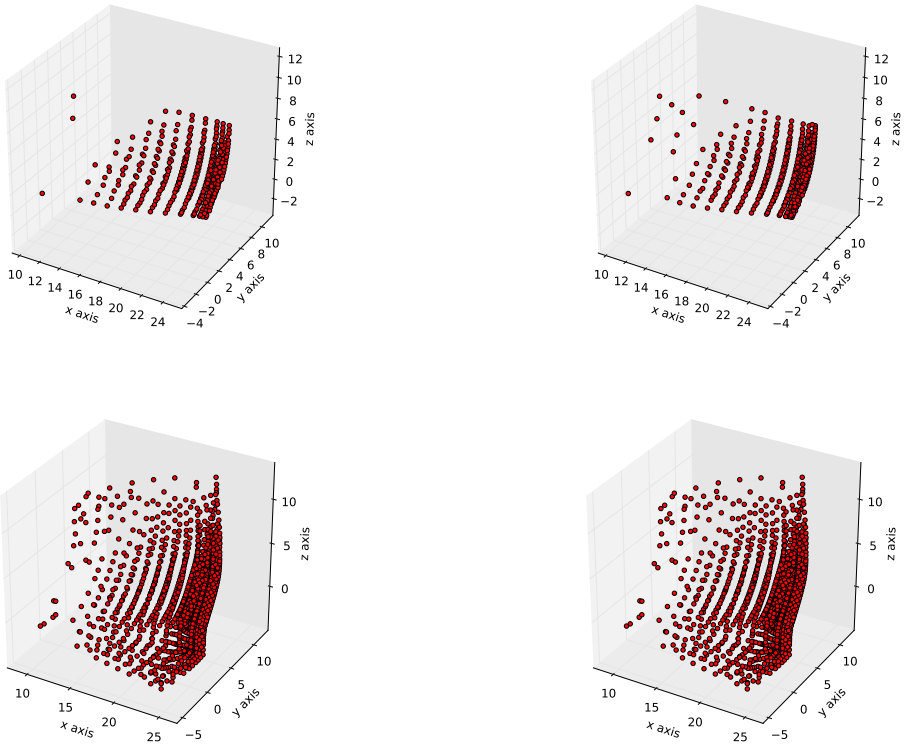


(a) Without coarsening (b) Criterion 1 ($k = 0.001$) (c) Criterion 2 ($k = 1$)

(d) Criterion 3 ($k = -10^{-6}$) (e) Criterion 4 ($k_1 = 1$, $k_2 = 0.001$)

Figure B.9: Comparison of the selected nodes for the fine mesh rotor 67 test case between the different greedy criteria (top = blade, bottom = exterior boundary).

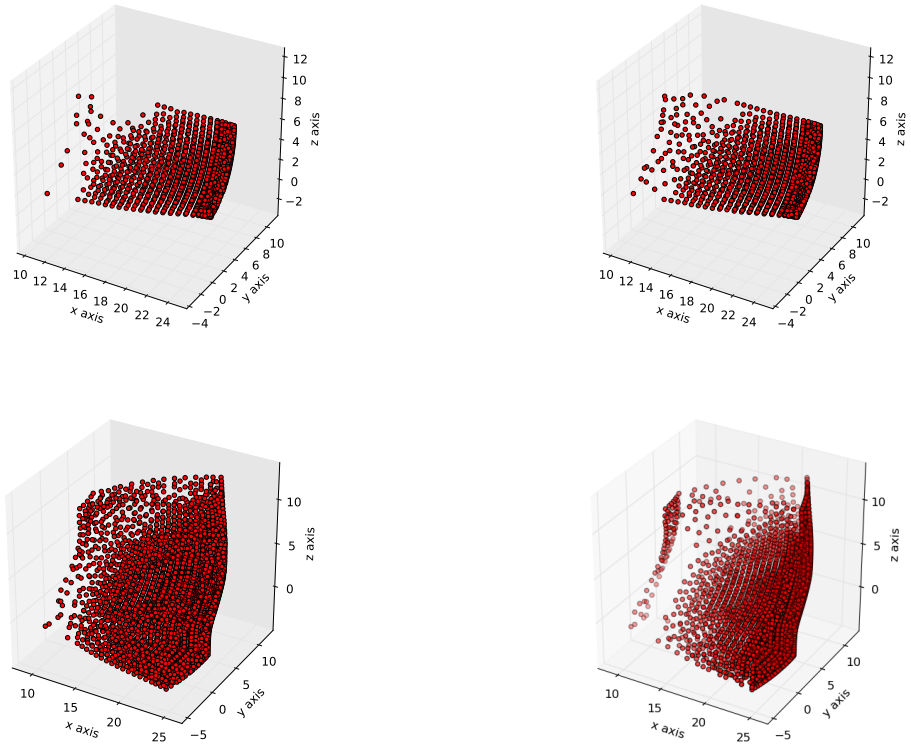
B.4 Comparison of Selection Methods for Rotor 67



(a) Selection method 1 ($n_b = 349$ moving + 514 sliding + 910 fixed = 1773)

(b) Selection method 4 ($n_b = 418$ moving + 718 sliding + 637 fixed = 1773)

Figure B.10: Comparison of the selected nodes for the coarse rotor 67 test case for selection method 1 and 4, when the greedy iteration is stopped as soon as 10% of the boundary nodes are selected (top = blade, bottom = exterior boundary).



(a) Selection method 1 (n_b = 1588 moving + 2066 sliding + 3437 fixed = 7091).

(b) Selection method 4 (n_b = 1708 moving + 3027 sliding + 2284 fixed = 7091).

Figure B.11: Comparison of the selected nodes for the fine rotor 67 test case for selection method 1 and 4, when the greedy iteration is stopped as soon as 10% of the boundary nodes are selected (top = blade, bottom = exterior boundary).

Appendix C

Simulation Settings

C.1 Vortex Induced Beam Vibration

Table C.1: CFD simulation settings of the vortex induced beam vibration test case

Fluid model	Incompressible air
Flow model	Laminar Navier-Stokes
Inlet velocity	0.315 $m.s^{-1}$
Outlet pressure	101325 Pa
Solid boundary	Adiabatic
Time step size	0.01 s
Number of time steps	1000
Number of iterations per time step	100
Number of modes	4
Damping coefficient	0 for all modes

Table C.2: CSD simulation settings of the vortex induced beam vibration test case

Number of elements	768
Element type	CPE4
Young's modulus	200 kPa
Poisson's ratio	0.35
Density	2 $kg.m^3$

C.2 Elastic flap in a duct

Table C.3: CFD simulation settings of the elastic flap test case

Fluid model	Air as perfect gas
Flow model	Turbulent Navier-Stokes
Turbulence model	Spalart-Allmaras
Inlet velocity	8 $m.s^{-1}$
Outlet pressure	101325 Pa
Solid boundary	Adiabatic
Time step size	0.00015 s
Number of time steps	1000
Number of iterations per time step	100
Number of modes	14
Damping coefficient	0 for all modes

Table C.4: CSM simulation settings of the elastic flap test case

Number of elements	800
Element type	C3D20R
Young's modulus	100 MPa
Poisson's ratio	0.49
Density	1000 $kg.m^3$

C.3 Rotor 67 Blade Deflection

Table C.5: CFD simulation settings of the rotor 67 test case

Fluid model	Air as perfect gas
Flow model	Turbulent Navier-Stokes
Turbulence model	Spalart-Allmaras
Rotation speed	16,043 RPM
Inlet tip relative Mach number	1.35
Inlet tip relative velocity	454 $m.s^{-1}$
Inlet total pressure	Exp. profile [63]
Inlet total temperature	Exp. profile [63]
Inlet turbulent viscosity	$1 \times 10^{-5} m^2 \cdot s^{-1}$
Outlet pressure at $r = 0.1961m$	121,833.2 Pa
Solid boundary	Adiabatic, no slip
Number of iterations	1000
Number of modes	30
Damping coefficient	0 for all modes

Table C.6: CSM simulation settings of the elastic flap test case

Number of elements	1,702
Element type	C3D15
Young's modulus	142.2 GPa
Poisson's ratio	0.3
Density	4,539.5 $kg.m^3$
Rotation speed	16,043

C.4 AGARD 445.6 Wing Deflection

Table C.7: CFD simulation settings of the AGARD 445.6 test case

Fluid model	Air as perfect gas
Flow model	Turbulent Navier-Stokes
Turbulence model	Spalart-Allmaras with extended wall function
Mach number	0.5
Freestream velocity	172.4555 $m.s^{-1}$
Freestream temperature	297.27
Freestream pressure	36494.82 Pa
Freestream turbulent viscosity	0.0001 $m^2.s^{-1}$
Solid boundary	Adiabatic, no slip
Time step size	0.0005 s
Number of time steps	1400
Number of iterations per time step	25 (with CPU booster)
Number of modes	6
Damping coefficient	0 for all modes

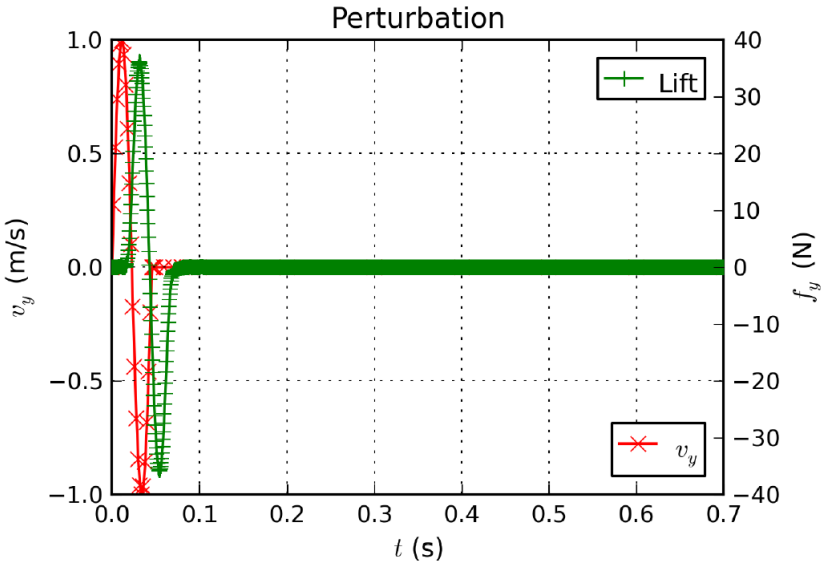
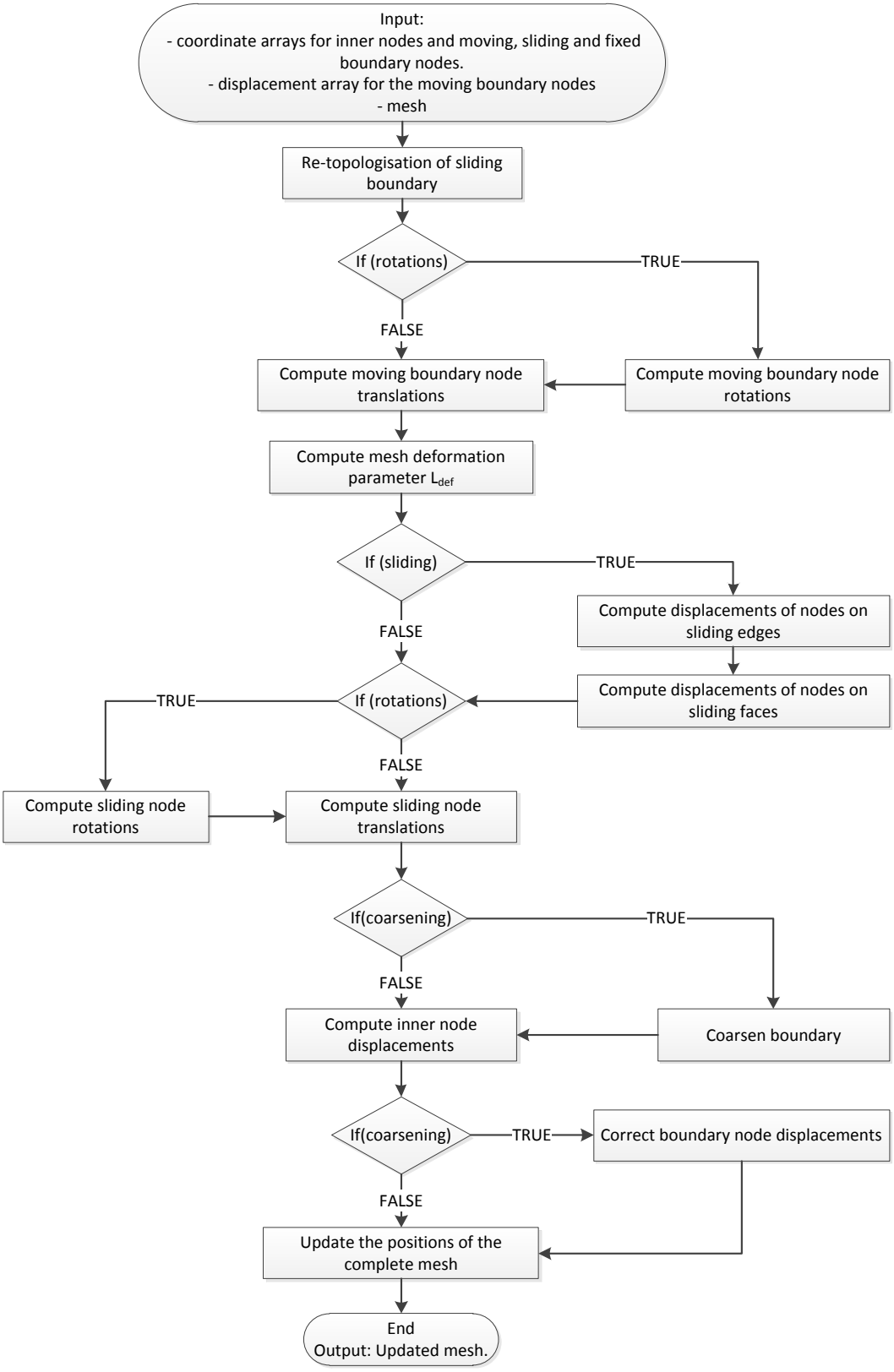


Figure C.1: Imposed perturbation [15].

Appendix D

IDW Mesh Deformation Flow Chart


Input:
- coordinate arrays for inner nodes and moving, sliding and fixed boundary nodes.
- displacement array for the moving boundary nodes
- mesh
Re-topologisation of sliding boundary
If (rotations)
TRUE
FALSE
Compute moving boundary node translations
Compute moving boundary node rotations
Compute mesh deformation parameter L_{def}
If (sliding)
TRUE
FALSE
Compute displacements of nodes on sliding edges
Compute displacements of nodes on sliding faces
If (rotations)
TRUE
FALSE
Compute sliding node rotations
Compute sliding node translations
If(coarsening)
TRUE
FALSE
Compute inner node displacements
Coarsen boundary
If(coarsening)
TRUE
Correct boundary node displacements
FALSE
Update the positions of the complete mesh
End
Output: Updated mesh.

1