Computational fluid-structure interaction

Spatial coupling, coupling shell and mesh deformation

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 22 december 2008 om 15:00 uur

door

Augustina DE BOER

ingenieur toegepaste wiskunde
geboren te Allingawier

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. ir. drs. H. Bijl
Prof. dr. D. J. Rixen

Samenstelling promotiecommissie:

Rector Magnificus,	Voorzitter
Prof. dr. ir. drs. H. Bijl,	Technische Universiteit Delft, promotor
Prof. dr. D. J. Rixen,	Technische Universiteit Delft, promotor
Prof. dr. C. Allen,	University of Bristol
Prof. dr. ir. M. J. L. van Tooren,	Technische Universiteit Delft
Prof. dr. J. Vierendeels,	Universiteit Gent
Prof. dr. ir. C. Vuik,	Technische Universiteit Delft
Prof. dr. H. Wendland,	University of Sussex

Summary

Computers and numerical algorithms have significantly advanced the last decade, such that the simulation of problems involving more than one discipline has become feasible. Future computational research is therefore expected to increasingly be multidisciplinary. In this thesis we mainly consider the multidisciplinary field known as fluid-structure interaction. This field concerns the interaction between a flow and a deformable structure and combines the two monodisciplinary fields Computational Fluid Dynamics and Computational Structure Dynamics. The focus will be on transfer algorithms for the spatial coupling, a separate coupling shell that handles the data transfer between two solvers and a new mesh deformation scheme based on radial basis function interpolation.

For the simulation of multidisciplinary problems it is possible to reuse proven monodisciplinary solvers, that have been developed and tuned for tens of years. In this case each physical system is solved individually and interaction effects are treated as external conditions. To be able to transfer information between two solvers, generally a transfer algorithm is needed to deal with the incompatible meshes at the interface. In fluid-structure interaction computations this transfer algorithm can either use a conservative or consistent approach. When the transfer method is based on a weighted residual formulation of the coupling conditions, the highest accuracy and efficiency are obtained with the conservative approach. For other transfer methods the conservative approach results in large unphysical oscillations in the pressure received by the structure. When the structure is flexible enough these oscillations can result in deviations in the displacement of the flow interface. For these methods the consistent approach provides the best accuracy and efficiency.

During a multidisciplinary computation, two or more monodisciplinary solver programs have to share information about data at their common interface and the execution of the solvers has to be synchronized. The coordination of the different solvers is commonly handled by a coupling shell, where the majority of coupling shells used today are embedded subprograms that have been developed for coupling two specific solvers. This makes it hard, if not impossible, to replace one solver by another. Also, numerical techniques developed to accurately and efficiently handle the information transfer between the solvers cannot be easily reused. Therefore, Flecs, a generic, flexible coupling shell designed for implementing and applying an interface for multidisciplinary simulations is developed. The aim of Flecs is to provide a flexible platform for developing new data transfer algorithms and coupling schemes to be

able to perform large multidisciplinary computations. The design of FLECS is based on a client-server model in which two solvers communicate with a separate program called the coupling server that is responsible for transferring data from one physical domain to another. This design is minimally intrusive in the sense that there is no need to change the structure of the solver programs, only some subroutine calls have to be made. In order to make FLECS suitable for large applications, it supports solvers that run on parallel computers. In particular, FLECS can deal with data sets that have been distributed over multiple parallel processes and supports the implementation of parallel data transfer algorithms. Numerical acceleration techniques can be implemented as subroutines within the coupling server without having to change the separate solvers involved. This means that these acceleration techniques can be reused when one or more different solvers are coupled.

In fluid-structure interaction computations the computational flow mesh has to follow the movement of the deforming domain. For the interpolation of the displacement of the boundary nodes of the mesh to the inner domain radial basis functions can be used. This results in a new point-by-point mesh deformation scheme. The method can handle large deformations caused by translation, rotation and deformation of a structure for both 2D and 3D meshes. In a first comparison the new method produces meshes of higher quality than the popular semi-torsional spring analogy for very large deformations. The new method also preserves the performance of higher order time integration methods, due to its smooth deformation in time. However, further investigation is needed to improve the efficiency of solving the ill-conditioned full matrix system and to speed up the many evaluations of the interpolation function.

An FSI computation of flow around a cylinder with deformable flap is performed to investigate the performance of four transfer algorithms in a 2D setting and the use of FLECS and the new mesh deformation method in a real application. It is shown that when FLECS is used, only a few subroutine calls have to be added to the flow and structure solver. With the new deformation method the mesh quality remains high throughout the computation. Even as the mesh deformation is performed by solving the system and evaluating the interpolation with direct methods, the time needed for the mesh deformation is less than 10% of the time needed for the flow solve. The results for the non-matching case are compared with a reference solution obtained with matching meshes. Overall we conclude that transferring pressure with simple nearest neighbour interpolation and displacement with consistent radial basis function interpolation gives the best results for this test case.

Samenvatting

Computers en numerieke algoritmen zijn significant verbeterd in het laatste decennium, zodanig dat het mogelijk is geworden om problemen te simuleren waarbij meer dan één discipline betrokken is. Het wordt dan ook verwacht dat toekomstig onderzoek steeds vaker multidisciplinair zal zijn. In dit proefschrift beschouwen we vooral het multidisciplinaire onderzoeksgebied dat bekend staat als vloeistof-vaste stof interactie. Dit betreft de interactie tussen een stroming en een vervormbare structuur en combineert de twee monodisciplinaire onderzoeksgebieden Numerieke stromingsleer en Numerieke structuurdynamica. De aandacht richt zich vooral op overdrachtsalgoritmen voor de ruimtelijke koppeling, een aparte koppelingsinterface die de overdracht van data tussen twee rekenprogramma's afhandelt en een nieuw roosterdeformatie algoritme gebaseerd op interpolatie door middel van radiale basis functies. Om multidisciplinaire problemen te simuleren is het mogelijk om bestaande monodisciplinaire rekenprogramma's her te gebruiken, die in de afgelopen tientallen jaren zijn ontwikkeld en afgeregeld. In dit geval wordt elk fysisch systeem individueel opgelost en worden interactie effecten als externe condities behandeld. Om informatie tussen twee rekenprogramma's te kunnen overbrengen is er in het algemeen een overdrachtsalgoritme nodig om met niet-aansluitende roosters om te kunnen gaan. In vloeistof-vaste stof interactie berekeningen kan dit overdrachtsalgoritme of een conservatieve of een consistente benaderingswijze gebruiken. Wanneer het overdrachtsalgoritme is gebaseerd op een gewogen residu formulering van de koppelingscondities wordt de hoogste nauwkeurigheid en efficiëntie behaald met de conservatieve benaderingswijze. Voor andere overdrachtsalgoritmen resulteert de conservatieve benaderingswijze in grote niet-fysische oscillaties in de druk die ontvangen wordt door de structuur. Wanneer de structuur flexibel genoeg is kunnen deze oscillaties afwijkingen in de verplaatsing van de interface van de stroming veroorzaken. Voor deze methoden geeft de consistente benaderingswijze de beste nauwkeurigheid en efficiëntie.

Gedurende een multidisciplinaire berekening moeten twee of meer monodisciplinaire rekenprogramma's informatie delen over data op hun gezamenlijke interface en de uitvoering van de twee rekenprogramma's moet worden gesynchroniseerd. De cordinatie van de verschillende rekenprogramma's wordt meestal afgehandeld door een koppelingsinterface, waarbij de meerderheid van de koppelingsinterfaces die vandaag de dag worden gebruikt ingebouwde subprogramma's zijn die ontwikkeld zijn voor de koppeling van twee specifieke rekenprogramma's. Dit maakt het moeilijk, zo niet onmogelijk, om het ene rekenprogramma te vervangen door een andere. Ook

kunnen numerieke technieken die zijn ontwikkeld om nauwkeurig en efficiënt de informatieoverdracht tussen de rekenprogramma's af te handelen niet makkelijk worden hergebruikt. Daarom is FLECS ontwikkeld, een generieke, flexibele koppelingsinterface, ontworpen om een interface voor multidisciplinaire simulaties te kunnen implementeren en toepassen. Het doel van FLECS is om een flexibel platform te leveren voor het ontwikkelen van nieuwe algoritmen voor dataoverdracht en koppelingsschema's, om grote multidisciplinaire berekeningen uit te kunnen voeren. Het ontwerp van FLECS is gebaseerd op een client-server model waarin twee rekenprogramma's met een apart programma, de koppelingsserver, communiceren. Deze koppelingsserver is verantwoordelijk voor het overbrengen van data van het ene fysische domein naar het andere. Dit ontwerp is minimaal ingrijpend wat betekent dat de structuur van de rekenprogramma's niet hoeft worden aangepast, alleen een paar subroutines moeten worden aangeroepen. Om FLECS geschikt te maken voor grote toepassingen, ondersteunt het rekenprogramma's die op parallelle computers rekenen. In het bijzonder kan FLECS omgaan met data die gedistribueerd is over meerdere parallelle processen en ondersteunt het de toepassing van parallelle dataoverdracht algoritmen. Numerieke versnellingstechnieken kunnen worden ingebouwd als subroutines binnen de koppelingsserver, zonder de betrokken rekenprogramma's aan te hoeven passen. Dit betekent dat deze versnellingstechnieken kunnen worden hergebruikt wanneer één of meer andere rekenprogramma's worden gekoppeld.

In vloeistof-vaste stof interactie berekeningen moet het rekenrooster van de vloeistof de beweging volgen van het vervormende domein. Om de verplaatsingen van de randpunten van het rooster naar het binnendomein van het rooster te interpoleren kunnen radiale basis functies worden gebruikt. Dit resulteert in een nieuw punt-bij-punt roosterdeformatie schema. Deze methode kan grote deformaties aan veroorzaakt door translatie, rotatie en vervorming van een structuur voor zowel 2D als 3D roosters. In een eerste vergelijking genereert de nieuwe methode, voor hele grote vervormingen, roosters van een hogere kwaliteit dan de populaire semi-torsional spring analogie. De nieuwe methode behoudt ook de orde van hogere orde tijdsintegratiemethoden doordat de vervorming glad in de tijd verloopt. Er is echter verder onderzoek nodig om de efficiëntie te verbeteren van het oplossen van het slecht geconditioneerde matrix systeem en om de vele evaluaties van de interpolatie functie te versnellen.

Een vloeistof-vaste stof berekening van een stroming rond een cilinder met vervormbare flap is uitgevoerd om de vier overdrachtsalgoritmen en het gebruik van FLECS en de nieuwe roosterdeformatie methode te onderzoeken in een echte 2D toepassing. Door het gebruik van FLECS hoeven er maar een paar subroutine aanroepen aan de rekenprogramma's te worden toegevoegd. Met de nieuwe roosterdeformatie methode blijft de kwaliteit van het rooster hoog gedurende de berekening. De tijd die nodig is voor de roosterdeformatie is minder dan 10% van de tijd nodig voor de stroming, zelfs al wordt de roosterdeformatie uitgevoerd door het systeem op te lossen en te evalueren met directe methoden. De resultaten voor niet-aansluitende roosters worden vergeleken met een referentieoplossing verkregen met aansluitende roosters. In het algemeen kunnen we concluderen dat het overdragen van de druk met simpele nearest-neighbour interpolatie en de verplaatsing met consistente radiale basis functie interpolatie de beste resultaten geeft voor dit probleem.

Contents

Chapter 1

Introduction

1.1 Motivation

A current trend in innovative engineering applications is that structures become lighter, more flexible and interact with multiple physical fields. Examples can be found in wind turbines, micro-systems, satellites, aircraft and cars. As a consequence they are more prone to potentially dangerous oscillations resulting from different kinds of excitations. In addition, new technologies emerge where complex interactions between different components are at the core of their functionality. An example is smart, flexible, materials which in the future might be applied for load reduction on wind turbine blades, or noise reduction in cars [53, 77, 106]. Another important field is applications in bio-engineering [5, 15, 54, 117, 112]. For these new designs simulation is an important tool to explore the interaction phenomena. As a result a growing number of breakthroughs of the near future require interdisciplinary simulation tools that can deal with complex multiple physical models and with very different scales.

Computers and numerical algorithms have significantly advanced the last decade, such that the simulation of problems involving more than one discipline has become feasible. Future computational research is therefore expected to increasingly be multidisciplinary, combining existing insights from different areas. However, as simulating each of the problems separate usually is already demanding (tens of millions of unknowns for computations of flow around aircraft is common practice), solution of the dynamic interaction will require enormous computational times. Computational efficiency is therefore of the uttermost importance.

For the simulation of multidisciplinary problems it is possible to reuse proven monodisciplinary solvers, that have been developed and tuned for tens of years. In this case each physical system is solved individually and interaction effects are treated as external conditions. This means that the monodisciplinary solver programs have to share information about data at their common interface and the execution of the solvers has to be synchronized. The coordination of the different solvers is commonly handled by a coupling shell, where the majority of coupling shells used today are embedded subprograms that have been developed for coupling two specific solvers

[38, 51, 102, 104, 112]. This makes it hard, if not impossible, to replace one solver by another. Also, numerical techniques developed to accurately and efficiently handle the information transfer between the solvers cannot easily be reused. Therefore, one of the aims of this thesis is to develop a generic, flexible coupling shell suitable to couple different parallel solvers.

One of the oldest and most popular multidisciplinary fields investigated is fluid-structure interaction (FSI). This field concerns the interaction between a flow and a deformable structure and combines the two monodisciplinary fields Computational Fluid Dynamics (CFD) and Computational Structure Dynamics (CSD) and will be the main research field considered in this thesis. Two of the main interest areas in fluid-structure interaction research are aeronautical [46, 47, 38, 55, 93] and biomechanical applications [5, 15, 54, 117, 112], but there are also other examples like parachute dynamics [99, 104], airbag deployment [86] and industrial applications [2, 36, 90, 107]. In FSI computations not only the two fields have to be coupled, but the flow solver also has to adapt its mesh to the deforming domain caused by the deformation of the structure. When the structure is flexible enough, these deformations can become very large. This provides the motivation for another branch of research in this thesis, to develop a new automatic mesh-movement technique that can deal with very large deformations.

1.2 Background

In this section we address several of the main research topics for multidisciplinary computations, and fluid-structure interaction in particular. They concern partitioned coupling techniques, data transfer between non-matching meshes, the use of a generic coupling shell and mesh deformation techniques which are needed when an Arbitrary-Lagrangian formulation is used for the flow discretization.

Partitioned coupling

There are two ways to numerically solve interdisciplinary problems. The first is to develop a new dedicated solver that solves the whole system at once, the so-called monolithic approach. Major advantage is that the solver can be optimized for the specific problem. However, for each different problem an entirely new solver has to be developed and therefore only a few papers have appeared on monolithic solution techniques [21, 62, 88].

The other possibility is to reuse proven monodisciplinary solvers, that have been developed and tuned for tens of years. In this case each physical system is solved individually and interaction effects are treated as external conditions. This is called the partitioned approach. Major disadvantage of this approach is that the coupling of two different solvers is not straightforward. Without much care the accuracy of the coupled problem easily reduces to first-order in time [60], irrespective of the order of the separate solvers. In addition, the partitioning process can lead to instabilities, such that time step restrictions come from stability, rather than accuracy requirements.

Basically two approaches can be found for the temporal coupling in partitioned schemes: loosely coupled and strongly coupled algorithms. In loosely coupled schemes the flow and structure are solved only once for each time step. This introduces a partitioning error caused by the time lag between the two systems. This partitioning error can lead to numerical instabilities, which can only be prevented by reducing the time step. In literature many examples of efficient, loosely coupled algorithms can be found, which are at most second order accurate in time [45, 47, 75, 93]. An exception is the higher order IMEX scheme [121, 123], which can be designed for any order of accuracy and is proven to work for up to fifth order of accuracy.

To reduce the partitioning error sub-iteration methods can be used. The flow and structure are solved multiple times within one time step, leading to a strongly coupled scheme. Strongly coupled schemes are generally more robust than loosely coupled schemes, especially in problems where the structure is very flexible and/or light compared to the surrounding flow. Therefore research is currently aimed at limiting the number of sub-iterations using numerical acceleration techniques, such as Aitken sub-iterations [67, 89], Newton-Krylov methods [87] and reduced order methods [112]. The computational costs of a simple sub-iteration can be reduced by multilevel techniques [124].

Non-matching meshes

In fluid structure interaction computations pressure loads have to be transmitted from the fluid side of the fluid-structure interface to the structural nodes on that interface. Also the fluid mesh points on the interface have to follow the deforming structure. In fluid-structure interaction simulations the meshes at the fluid-structure interface usually do not match due to the different mesh requirements for the flow and structure. This means that there can be gaps and/or overlaps between the meshes. The exchange of data over the discrete interface becomes then far from trivial and an interpolation/projection step has to be carried out to enable transfer of information between the two domains. In literature different methods can be found to transfer data between non-matching meshes, such as nearest neighbour interpolation [108], projection methods [37, 83, 85] and methods based on interpolation by splines [13, 100, 101].

The general opinion is that energy should be conserved over the interface. In [46] a conservative transfer approach in space is introduced. This approach is based on the global conservation of virtual work over the interface, where a transformation matrix performs the transfer of displacements and the transposed of this matrix the transfer of pressure loads between the two discrete interfaces. However, for a general transfer method this can lead to unphysical oscillations in the pressure forces received by the structure as is briefly mentioned by Ahrem et al [3].

Instead of using the same transformation matrix for both transferring the displacement and pressure loads over the interface, two different transformation matrices can be defined, which we will define as the consistent approach. This leads to a transfer approach without unphysical oscillations in the pressure forces. However, conservation of energy over the interface is not guaranteed, which is generally already not the

case when a partitioned coupling strategy is used. In this thesis we try to answer the question wether conservation of energy over the interface or a consistent interpolation is preferred.

Coupling shell

The reuse of existing solvers is one of the main benefits of the partitioned coupling approach. Coordination of the different solvers is commonly handled by a coupling shell. This can be a separate program, or a sub-program (a set of subroutines) embedded in one of the solver programs, or a part of one large program that contains the individual solvers as sub-programs. The coupling shell synchronizes the execution of the solvers and handles the transfer of data from one physical domain to another.

The majority of coupling shells are embedded subprograms that have been developed for coupling two specific solvers [38, 51, 102, 104, 112], which makes it hard, if not impossible, to replace one solver by another. One exception is the coupling library MpCCI (Mesh based Parallel Code Coupling) [52, 56], a generic coupling shell that can be used both as an embedded sub-program and as a separate program. MpCCI is relatively easy to use and provides many advanced features. However, MpCCI only contains a few simple transfer algorithms and no numerical acceleration techniques, and since MpCCI is a commercial product, the source code is not accessible. This means that MpCCI is not suitable for research on developing new data transfer algorithms or acceleration techniques and a more generic coupling shell is needed.

A generic coupling shell should meet the following conditions:

- The software must be open source, such that everyone can make modifications and extensions to the code.
- The coupling shell has to be minimally intrusive in the solvers. The main structure of the solvers must remain the same and only a few subroutine calls have to be added to the code on a high level.
- The coupling shell must be interoperable with different programming languages, such as `Fortran 90`, `C` and `C++`.
- The information transfer has to be handled within a separate server program.
- The transfer algorithm must be based on a plug-in architecture, such that a new transfer algorithm can be implemented without having to deal with the general communication between the two solvers.
- The coupling shell must be able to deal with parallel solvers and parallel transfer algorithms in order to perform large simulations.
- It must be possible to couple one solver to more than one other solver.

One advantage of a generic coupling shell is that numerical algorithms which are implemented to efficiently couple two solvers, can be reused when using one or more different solvers. The development of such a coupling shell would contribute to the research possibilities on efficient coupling techniques. In Munich they already started with the development of the open-source coupling shell FSIce [91], but there the focus

lies on fast search algorithms for the coupling between Cartesian meshes. In this thesis we introduce a flexible coupling shell which provides a platform for developing new data transfer and coupling algorithms to be able to perform large multidisciplinary computations and which satisfies the conditions given above.

Arbitrary Lagrangian-Eulerian formulation

The flow equations are generally discretized using the Eulerian description. In the Eulerian description the meshes remain fixed in time and material is allowed to flow through the mesh. This is in contrast to the Lagrangian description, where the mesh is fixed to the material of interest and as the material deforms, the mesh deforms with it. This description is generally used to discretize the structure equations. However, when the flow domain moves or deforms in time due to a moving boundary caused by the deformation of the structure, a fixed mesh becomes inconvenient, because it requires the explicit tracking of the domain boundary. Therefore the Arbitrary Lagrangian-Eulerian (ALE) formulation [41] is often used to discretize the flow equations on moving meshes. In the ALE formulation the discretization is both Lagrangian and Eulerian. Lagrangian, because it allows for a flexible mesh that follows the moving boundaries, and Eulerian, since the flow quantities still flow through the mesh. The ALE formulation has become standard for problems with a high Reynolds number on moving domains in which the domain undergoes large deformations and distortions.

Mesh deformation

When an ALE formulation is used for the flow, the computational flow mesh has to follow the movement of the deforming domain in time. Regenerating a mesh each time step is a time-consuming and nontrivial task. Therefore, several algorithms have been developed to update the mesh automatically. From the viewpoint of computational efficiency, the mesh moving scheme should yield a good fluid mesh with the least amount of computational expenses.

For structured meshes Transfinite Interpolation [103, 109, 113] is generally used to update the mesh. The displacements of points at the boundaries of the mesh are interpolated along grid lines to points in the interior of the mesh. However, this technique is unsuitable for unstructured grids.

For unstructured grids two different mesh movement strategies are known: grid-connectivity and point-by-point schemes. The first exploits the connectivity of the internal grid points. The connection between the grid points is represented for example by springs [6, 22, 39, 43, 119] or as solid body elasticity [65, 84, 105]. Special instances of this continuous approach include moving grids based on Laplacian and Biharmonic operators [64, 70]. In all the methods based on grid connectivity a (sparse) system of equations has to be solved, involving all the flow points and can therefore be quite expensive. Hanging nodes, encountered in unstructured meshes when only one of the adjacent elements at an edge is subdevided, require special treatment. The grid-connectivity methods perform very well for small deformations, but generally have difficulties to maintain a high mesh quality when large deformations are involved.

In point-by-point schemes each grid point is moved individually based on its position in space and no grid-connectivity information is needed. This can be a simple master-slave coupling [61] or based on a volume spline interpolation [103]. However, until now point-by-point schemes are only applied to the boundary nodes of multi-grid blocks, the interior mesh of the blocks is adapted with fast techniques available for structured grids.

Radial basis functions (RBFs) have become a well-established tool to interpolate scattered data, because of their excellent approximation properties [33]. They have been successfully applied to areas as diverse as computer graphics [34], geophysics [19, 18], error estimation [73] and the numerical solution of partial differential equations [71, 72]. They can also be used in fluid-structure interaction computations to transfer information over the discrete non-matching fluid-structure interface [13, 100, 101]. A global interpolation function is used to interpolate the displacements known at the boundary of the structural mesh to the boundary of the flow mesh. But why not interpolate the displacement to all the nodes of the flow mesh, instead of only to the boundary? This idea has already been applied to the block boundaries in multi-block grids [94, 103]. There it was mentioned that applying it to the whole internal grid would be computationally very expensive. This is because for the structured part of multi-block meshes very efficient techniques are known and using RBF interpolation requires to solve a dense system involving all the nodes on the boundary of the domain. We therefore want to investigate in this thesis if it is worthwhile to use a point-by-point scheme based on interpolating the displacement with radial basis functions to deform totally unstructured meshes. This point-by-point mesh movement scheme has to effectively be able to deal with large deformations.

1.3 Objectives and outline

The main objectives of this thesis are:

- To investigate the difference in accuracy and efficiency between conservative and consistent approaches for the transfer of data between non-matching meshes.
- To design a flexible coupling shell for implementing and applying an interface for multidisciplinary simulations, which satisfies the conditions given in Section 1.2.
- To develop a new point-by-point mesh deformation algorithm based on interpolating the displacements of the moving boundary to the whole mesh using radial basis functions.
- To perform a real fluid-structure interaction computation in which the three previous points are combined.

In order to meet these objectives the contents of this thesis are organized as follows.

In Chapter 2 we investigate the difference in accuracy and efficiency between conservative and consistent approaches for the transfer of data between non-matching meshes. This is done for an analytical test problem as well as a steady quasi-1D FSI problem, for different transfer algorithms found in literature.

FLECS, a flexible coupling shell, designed for implementing and applying an interface for multidisciplinary simulations is introduced in Chapter 3. The aim of FLECS is to provide a flexible platform for developing new data transfer algorithms and coupling schemes to be able to perform large multidisciplinary computations. The design of FLECS is based on a client-server model in which two solvers communicate with a separate program called the coupling server that is responsible for transferring data from one physical domain to another. This design is minimally intrusive in the sense that there is no need to change the structure of the solver programs, only some subroutine calls have to be made. In order to make FLECS suitable for large applications, it supports solvers that run on parallel computers.

In Chapter 4 a new mesh movement algorithm for unstructured grids is developed which is based on interpolating displacements of the boundary nodes to the whole mesh with radial basis functions (RBFs). The method is tested for large mesh deformations caused by translations, rotations and deformations, for various RBFs found in literature. These tests are preformed both on 2D and 3D meshes. Also the effect of mesh deformation on the performance of higher order time integration methods and different techniques to improve the overall efficiency of the computation are investigated.

The findings of Chapters 2 till 4 are incorporated to perform a real 2D FSI computation in Chapter 5. The flow and structure solver, which are separate solver programs, are coupled with FLECS, and the flow mesh is repeatedly adapted to the deforming flow domain with the new mesh movement method based on radial basis function interpolation. For the FSI problem we use the numerical benchmark problem $FSI3$ of 2D flow around a cylinder with deformable flap described in [110]. In the numerical experiments we investigate four transfer algorithms that can deal with data transfer over non-matching interfaces. These four transfer algorithms are implemented within a separate FLECS server program that handles the coordination of the two solvers. The results for the non-matching case are compared with a reference solution obtained with matching meshes.

Chapters 6 and 7 contain concluding remarks and suggestions for future research, respectively. Parts of Chapters 2 to 4 have been published before. Their content is based on the following journal publications [27, 29, 32, 123] and conference proceedings [23, 24, 25, 26, 28, 30, 31, 92].

Chapter 2

Conservative and consistent approaches for the coupling of non-matching meshes

In FSI simulations it is usually not desirable to generate matching meshes at the fluid-structure interface, because different solvers may take care of the different physical domains. In addition, also the flow generally requires a much finer mesh than the structure. This means that the discrete interface between the domains may not only be non-conforming, but there can also be gaps and/or overlaps between the meshes. The exchange of data over the discrete interface becomes then far from trivial. In Figure 2.1 a 2D example of a non-matching discrete interface between a flow and structure domain is shown. When the meshes are non-matching, an interpolation/projection step has to be carried out to enable transfer of information between the two domains. In literature different methods can be found to transfer data between non-matching meshes, such as nearest neighbour interpolation [108], projection methods [37, 83, 85] and methods based on interpolation by splines [13, 100, 101].

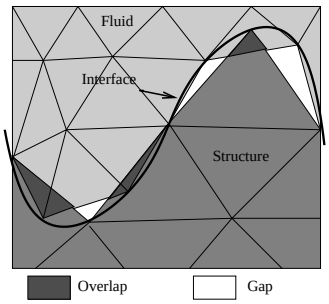


Fig. 2.1: Non-matching meshes in 2D.

The general opinion is that energy should be conserved over the interface. The overall conservation properties depend both on the time and the spatial coupling used, which cannot be investigated separately if the system is solved in a partitioned way. In this chapter we focus only on the spatial coupling. In [46] a conservative transfer approach in space is introduced. This approach is based on the global conservation of virtual work over the interface, where a transformation matrix performs the transfer of displacements and the transposed of this matrix the transfer of pressure loads between the two discrete interfaces. However, for a general transfer method this can

lead to unphysical oscillations in the pressure forces received by the structure as is briefly mentioned by *Ahrem et al* [3]. Especially for flexible structures this can have a large negative influence on the accuracy of the solution.

Instead of using the same transformation matrix for both transferring the displacement and pressure loads over the interface, two different transformation matrices can be defined, which we will define as the consistent approach. This leads to a transfer approach without unphysical oscillations in the pressure forces. However, conservation of energy over the interface is not guaranteed. When a partitioned coupling technique is used to advance in time this does not have to be a problem. In unsteady partitioned computations energy is generally already not conserved due to errors caused by the time lag between flow and structure. When the coupling error introduced by the information transfer is smaller than the spatial and temporal discretization error, this coupling error does not have to affect the stability and accuracy of the computation, especially when the spatial and time discretization themselves are very dissipative. However, when stability must be ensured, the variation of energy caused by the non-matching interface should be negative.

In this chapter we investigate the difference in accuracy and efficiency between the conservative and consistent approach for the coupling methods described in [29]. First the two approaches are presented followed by a short description of the different coupling methods. The difference in the interpolation properties between the two approaches is investigated using two analytical test problems. Finally, a simple steady quasi-1D FSI problem is used to investigate the performance of the methods in FSI computations with multiple transfers between flow and structure side. The work of this chapter has also been accepted for publication in a journal paper [32].

2.1 Conservative and consistent coupling approach

In this section the conservative and consistent coupling approach are presented. With the conservative approach the total energy is conserved when transferring displacement and pressure forces over the interface. The consistent approach ensures that a constant displacement and a constant pressure are exactly interpolated over the interface. The starting point of both approaches are the kinematic and dynamic boundary conditions at the interface, which are commonly used to couple the fluid and structure equation, and are given by

$$\mathbf{u}_f = \mathbf{u}_s \quad \text{on} \quad \Gamma, \tag{2.1a}$$

$$p_s \mathbf{n}_s = p_f \mathbf{n}_f \quad \text{on} \quad \Gamma, \tag{2.1b}$$

with $\mathbf{u}_{f,s}$ the displacement field, $p_{f,s}$ the pressure field or stress tensor and $\mathbf{n}_{f,s}$ the outward normal of the flow and structure interface, respectively. The continuous interface between the flow and structure is represented by Γ. The first of these two boundary equations, (2.1a), expresses the compatibility between the displacement fields of the structure and the fluid at the fluid-structure interface. The second equation, (2.1b), states that the tractions of the wet surface of the structure are in equilibrium with those on the fluid side. In the continuous formulation and for steady

state problems either displacement, velocity or acceleration can be used for the kinematic boundary condition (2.1a). However, in the discretized form this changes the stability properties for dynamic problems when a time integration scheme is involved [57].

Whichever coupling method is chosen to define the discrete form of these conditions, its outcome can be formulated as

$$\mathbf{U}_f = H_{fs}\mathbf{U}_s \tag{2.2a}$$

$$\mathbf{P}_s = H_{sf}\mathbf{P}_f, \tag{2.2b}$$

with the $(n_f^u \times n_s^u)$ matrix H_{fs} and $(n_s^p \times n_f^p)$ matrix H_{sf} transformation matrices between the flow and structure interface, where $n^{u,p}$ is the number of unknowns on the interface for the displacement and pressure, respectively and the subscript f or s denotes whether this is on the flow or the structure side. The subscripts fs and sf for the transformation matrices denote that information is transferred from structure to flow or flow to structure, respectively. The discrete values in the interface points of the displacement and pressure are contained in $\mathbf{U}$ and $\mathbf{P}$, respectively. They are defined by the approximations

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n^u} N^i(\mathbf{x})\mathbf{U}_i, \qquad p(\mathbf{x})\mathbf{n}(\mathbf{x}) = \sum_{j=1}^{n^p} D^j(\mathbf{x})\mathbf{P}_j, \tag{2.3}$$

where $N(\mathbf{x})$ is a function that depends on the spatial discretization method used for the displacement (for example, a step function in the finite volume formulation or the shape function in the finite element formulation) and $D(\mathbf{x})$ is a function that depends on the discretization method used for the pressure. When the row-sum of H is equal to one, constant values are interpolated exactly.

Conservative approach

The general opinion is that energy should be conserved over the interface leading to a conservative coupling approach [46]. In this thesis we focus only on the spatial coupling and therefore we do not consider the energy conservation properties of the temporal discretization. This is valid when a monolithic solution procedure is used, with the same time integration method applied for both the flow and the structure, or in the case of steady state solutions. In this case energy is globally conserved over the interface when

$$\int_{\Gamma_f} \mathbf{u}_f \cdot p_f \mathbf{n}_f \, ds = \int_{\Gamma_s} \mathbf{u}_s \cdot p_s \mathbf{n}_s \, ds, \tag{2.4}$$

with $\mathbf{u}$ the displacement of the interface.

Writing out the left hand side of (2.4) using (2.3) gives

$$\int_{\Gamma_f} \mathbf{u}_f \cdot p_f \mathbf{n}_f \, ds = \sum_{j=1}^{n_f^p} \left[\sum_{i=1}^{n_f^u} \mathbf{U}_{f_i}^T \left(\int_{\Gamma_f} N_f^i D_f^j \, ds \right) \right] \mathbf{P}_{f_j} = \mathbf{U}_f^T M_{ff} \mathbf{P}_f. \tag{2.5}$$

In a similar way we find for the right hand side of (2.4)

$$\int_{\Gamma_s} \mathbf{u}_s \cdot p_s \mathbf{n}_s \, ds = \mathbf{U}_s^T M_{ss} \mathbf{P}_s, \tag{2.6}$$

where matrices M_{ff} $(n_f^u \times n_f^p)$ and M_{ss} $(n_s^u \times n_s^p)$ are defined as follows

$$M_{ff}^{ij} = \int_{\Gamma_f} N_f^i D_f^j \, ds, \qquad M_{ss}^{ij} = \int_{\Gamma_s} N_s^i D_s^j \, ds. \tag{2.7}$$

Substituting (2.5) and (2.6) into (2.4) shows that energy is globally conserved when

$$\begin{aligned} \mathbf{U}_f^T M_{ff} \mathbf{P}_f = \mathbf{U}_s^T M_{ss} \mathbf{P}_s \quad \Rightarrow \quad \mathbf{U}_s^T H_{fs}^T M_{ff} \mathbf{P}_f = \mathbf{U}_s^T M_{ss} \mathbf{P}_s \quad \forall \mathbf{U}_s \\ \Rightarrow M_{ss} \mathbf{P}_s = H_{fs}^T M_{ff} \mathbf{P}_f. \end{aligned} \tag{2.8}$$

So choosing

$$H_{sf} = M_{ss}^{-1} H_{fs}^T M_{ff} \tag{2.9}$$

in (2.2b) for the transformation of pressure over the interface results in global conservation of energy over the interface.

Note that this is, not surprisingly, the same result as obtained in [46] for the forces. When we define the forces $\mathbf{F}_s = M_{ss} \mathbf{P}_s$ and $\mathbf{F}_f = M_{ff} \mathbf{P}_f$ we can rewrite (2.2.3) as $\mathbf{F}_s = H_{fs}^T \mathbf{F}_f$ and the transposed transformation matrix has to be used to exchange forces to obtain global conservation of energy over the interface.

Consistent approach

In order to obtain a consistent interpolation, a constant displacement and constant pressure should be exactly interpolated over the interface (similar to the patch test criterion in finite element and domain decomposition methods [63]). This means that in the conservative approach both the row-sum of H_{fs} and $H_{sf} = M_{ss}^{-1} H_{fs}^T M_{ff}$ should be equal to one. For a general transformation matrix H_{fs} this is not the case as we will see in the following section.

To ensure a consistent interpolation of the pressure the matrix H_{sf} can be directly obtained by using one of the transfer methods, as is done to obtain matrix H_{fs}. In this way H_{sf} is independent of H_{fs} and both can be created with a row-sum equal to one. However in this way global conservation of energy over the interface is not guaranteed and in the remainder of the thesis we will address this as the consistent approach. The main question is whether global conservation of energy or a consistent interpolation is preferred in fluid-structure interaction computations.

2.2 Transfer methods

This section concerns three different transfer techniques which are commonly found in literature to transfer information between non-matching meshes in FSI computations: nearest neighbour interpolation, the weighted residual method and radial basis function interpolation. All methods result in a transformation matrix H_{BA} to transfer known values at the interface of mesh A to the interface of mesh B.

2.2.1 Nearest neighbour interpolation

Nearest neighbour interpolation (NN) is a very simple method to transfer data from mesh A to mesh B [108]. A search algorithm determines the point x_A in mesh A that is closest to a given point x_B in mesh B. The variable in x_B is then assigned the same value as in x_A. In this way the transformation matrix H_{BA} becomes a Boolean matrix, with a single one in each row which implies that the transformation is indeed consistent when the consistent approach is used.

In order to interpolate constant pressure values exactly when the conservative approach is used, we need according to (2.2.3):

$$M_{ss}\boldsymbol{\beta}_s = M_{ff}H_{fs}^T\boldsymbol{\beta}_f, \tag{2.10}$$

with $\boldsymbol{\beta}_s$ and $\boldsymbol{\beta}_f$ vectors of length n_s^p and n_f^p respectively, with all constant components equal to a constant value β. Substituting (2.7) this becomes

$$\sum_{j=1}^{n_s^p}\left[\int_{\Gamma_s} N_s^k D_s^j\,dx\right]\beta = \sum_{i=1}^{n_f^u}(H_{fs}^T)^{ki}\sum_{j=1}^{n_f^p}\left[\int_{\Gamma_f} N_f^i D_f^j\,dx\right]\beta \quad \text{for } k=1,...,n_s^u. \tag{2.11}$$

When we use the fact that $\sum_{j=1}^{n_\alpha^p} D_\alpha^j = 1$ and define $\int_{\Gamma_\alpha} N_\alpha^k\,dx = \Delta x_\alpha^k$ (where we assume constant or linear basis functions), we can derive for the left hand side

$$\sum_{j=1}^{n_s^p}\left[\int_{\Gamma_s} N_s^k D_s^j\,dx\right]\beta = \beta\int_{\Gamma_s} N_s^k\left[\sum_{j=1}^{n_s^p} D_s^j\right]dx = \beta\Delta x_s^k \quad \text{for } k=1,...,n_s^u, \tag{2.12}$$

and for the right hand side

$$\begin{aligned}\sum_{i=1}^{n_f^u}(H_{fs}^T)^{ki}\sum_{j=1}^{n_f^p}\left[\int_{\Gamma_f} N_f^i D_f^j\,dx\right]\beta &= \beta\sum_{i=1}^{n_f^u} H_{fs}^{ik}\int_{\Gamma_f} N_f^i\left[\sum_{j=1}^{n_f^p} D_f^j\right]dx \\ &= \beta\sum_{i=1}^{n_f^u} H_{fs}^{ik}\Delta x_f^i \quad \text{for } k=1,...,n_s^u.\end{aligned} \tag{2.13}$$

The condition for exact interpolation of constant pressure values becomes

$$\sum_{i=1}^{n_f^u} H_{fs}^{ik}\frac{\Delta x_f^i}{\Delta x_s^k} = 1 \quad \text{for } k=1,...,n_s^u. \tag{2.14}$$

For a simple equidistant grid this is equal to

$$\sum_{i=1}^{n_f^u} H_{fs}^{ik} = \frac{\Delta x_s}{\Delta x_f} \quad \text{for } k = 1, ..., n_s^u. \tag{2.15}$$

In other words, the column-sum of H_{fs} should be equal to $\Delta x_s/\Delta x_f$. As H_{fs} is a Boolean matrix this condition is generally not met. The difference between the left and right side of equation (2.15) does not decrease by refining both the flow and structure grid simultaneously, where the ratio between flow and structure cells (and therefore $\Delta x_s/\Delta x_f$) remains the same. This is caused by the fact that the general structure of H_{fs} does not change when both the flow and structure grid are refined simultaneously and therefore the column-sum also remains the same. Only when the grids converge to a matching and conforming mesh the difference will go to zero.

2.2.2 Weighted residual method

The weighted residual (WR) method or Lagrange multiplier method described in this section is based on the weak formulation of the conservation of loads or displacements over the interface [37, 83]. Starting point is the kinematic (2.1a) or dynamic boundary condition (2.1b) at the fluid-structure interface in the weak continuous form

$$\int_\Gamma (\mathbf{w}_B(x) - \mathbf{w}_A(x))\lambda(x)\, dx = 0, \qquad \mathbf{w} = \{\mathbf{u}, p\mathbf{n}\}, \tag{2.16}$$

where $\lambda(x)$ is the so called Lagrange multiplier. Because a Neumann-to-Dirichlet condition has to be satisfied at the interface a separate condition for the pressure and the displacement has to be found. This is different from the 'normal' Lagrange multiplier method, where the equilibrium of forces is automatically satisfied when $\int_\Gamma (\mathbf{u}_B(x) - \mathbf{u}_A(x))\lambda(x)\, dx = 0$ is imposed.

The following discretization for the quantities is used

$$\mathbf{w}_B(x) = \sum_{i=1}^{n_B} \Psi_B^i(x)\mathbf{W}_{B_i}, \quad \mathbf{w}_A(x) = \sum_{j=1}^{n_A} \Psi_A^j(x)\mathbf{W}_{A_j}, \tag{2.17}$$

with $\mathbf{W}_{A,B}$ containing the values of $\mathbf{w}_{A,B}$ in the points on the interface of mesh A and B, respectively, $\Psi_{A,B}$ the basis function of mesh A or B and $n_{A,B}$ the number of unknowns at the interface of mesh A or B. Substituting this into (2.16) gives

$$\int_\Gamma \lambda(x) \sum_{i=1}^{n_B} \Psi_B^i(x)\mathbf{W}_{B_i}\, dx = \int_\Gamma \lambda(x) \sum_{j=1}^{n_A} \Psi_A^j(x)\mathbf{W}_{A_j}\, dx. \tag{2.18}$$

When the mortar approach [14, 42] is used, the multiplier λ is chosen to be piecewise polynomial and to have the same interpolation order as side A or B, so $\lambda(x) = \sum_{k=1}^{n_\alpha} \Psi_\alpha^k$ with $\alpha \in \{A, B\}$. In this way the approximation can be established to be

optimal for non-conforming (but matching) meshes. Substituting this choice for λ into (2.18) gives

$$\sum_{i=1}^{n_B} \underbrace{\left[\int_\Gamma \Psi_\alpha^k \Psi_B^i \, dx\right]}_{C_{\alpha B}^{ki}} \mathbf{W}_{B_i} = \sum_{j=1}^{n_A} \underbrace{\left[\int_\Gamma \Psi_\alpha^k \Psi_A^j \, dx\right]}_{C_{\alpha A}^{kj}} \mathbf{W}_{A_j} \quad \text{for } k = 1, ..., n_\alpha. \tag{2.19}$$

This can be written in matrix form as

$$C_{\alpha B} \mathbf{W}_B = C_{\alpha A} \mathbf{W}_A, \tag{2.20}$$

with $C_{\alpha B}$ an $n_\alpha \times n_B$ matrix and $C_{\alpha A}$ an $n_\alpha \times n_A$ matrix.

Since we transfer data from mesh A to mesh B we need to solve for the side of mesh B, because the value of $\mathbf{w}$ on mesh A is assumed to be known. This means that we have to choose $\alpha = B$ to be able to solve system (2.20). As a result we obtain

$$\mathbf{W}_B = C_{BB}^{-1} C_{BA} \mathbf{W}_A. \tag{2.21}$$

So in short the transformation matrix becomes $H_{BA} = C_{BB}^{-1} C_{BA}$.

Consistent approach

In order to interpolate constant values exactly as required for a consistent interpolation we need

$$C_{BB} \boldsymbol{\beta}_B = C_{BA} \boldsymbol{\beta}_A, \tag{2.22}$$

with $\boldsymbol{\beta}_{A,B}$ a vector of length n_A or n_B respectively, with all constant components equal to a constant value β. Using (2.19) this becomes

$$\sum_{i=1}^{n_B} \left[\int_\Gamma \Psi_B^k \Psi_B^i \, dx\right] \beta = \sum_{j=1}^{n_A} \left[\int_\Gamma \Psi_B^k \Psi_A^j \, dx\right] \beta \quad \text{for } k = 1, ..., n_B. \tag{2.23}$$

Using the fact that $\sum_{k=1}^{n_\alpha} \Psi_\alpha^k = 1$ we can derive for the left hand side

$$\sum_{i=1}^{n_B} \left[\int_\Gamma \Psi_B^k \Psi_B^i \, dx\right] \beta = \beta \int_\Gamma \Psi_B^k \left[\sum_{i=1}^{n_B} \Psi_B^i\right] dx = \beta \int_\Gamma \Psi_B^k \, dx, \tag{2.24}$$

and for the right hand side

$$\sum_{j=1}^{n_A} \left[\int_\Gamma \Psi_B^k \Psi_A^j \, dx\right] \beta = \beta \int_\Gamma \Psi_B^k \left[\sum_{j=1}^{n_A} \Psi_A^j\right] dx = \beta \int_\Gamma \Psi_B^k \, dx. \tag{2.25}$$

All that remains is the selection of the discrete interface over which the integrals in (2.19) are taken, because generally $\Gamma_A \neq \Gamma_B \neq \Gamma$. For the matrix C_{BB} it is most accurate to integrate over Γ_B because both the values of Ψ_B^k and Ψ_B^i are known at that discretized interface and then no projection is needed. To obtain a consistent interpolation the integrals in matrix C_{BA} then also have to be integrated over Γ_B, otherwise (2.25) is unequal to (2.24).

Conservative approach

We now investigate the consistency of the pressure when the conservative transfer approach is used. We start again with the discretized kinematic boundary condition (2.2a), where the weighted residual method gives us

$$H_{fs} = C_{ff}^{-1} C_{fs}. \tag{2.26}$$

Substituting this into (2.2.3) gives

$$M_{ss}\mathbf{P}_s = C_{fs}^T C_{ff}^{-1} M_{ff}\mathbf{P}_f. \tag{2.27}$$

When in the derivation of the matrices C_{ff} and C_{fs} in equation (2.18) the Lagrange multiplier is chosen to be $\lambda(x) = \sum_{k=1}^{n_f^p} D_f^k$ instead of $\lambda(x) = \sum_{k=1}^{n_f} N_f^k$, then $C_{ff} = M_{ff}$ and (2.27) becomes

$$M_{ss}\mathbf{P}_s = C_{fs}^T \mathbf{P}_f. \tag{2.28}$$

In order to interpolate a constant pressure exactly we need

$$M_{ss}\boldsymbol{\beta}_s = C_{fs}^T \boldsymbol{\beta}_f, \tag{2.29}$$

with $\boldsymbol{\beta}_{s,f}$ a vector of length n_s or n_f respectively, with all constant components equal to a constant value β. We can derive in a similar way as in (2.24) and (2.25) that (2.29) is equal to

$$\int_{\Gamma_s} N_s^k \, dx = \int_{\Gamma_f} N_s^k \, dx \quad \text{for } k = 1, ..., n_s^u. \tag{2.30}$$

So only when the meshes are matching, $\Gamma_f = \Gamma_s$, equation (2.30) is always satisfied and the method is both conservative and consistent. This means that the patch-test is not satisfied for non-matching meshes with gaps and/or overlaps as was already shown in [78] for the mortar element method.

For non-matching meshes, refining both the flow and structure grid simultaneously causes the difference between Γ_f and Γ_s to become smaller. In that case the left side of (2.30) converges to the right side and the error that is made decreases. In contrast to NN the meshes only have to converge to a matching mesh and not to a conforming mesh in order to obtain convergence in the error.

Gauss integration

To be able to compute the transformation matrix H_{BA} the integrals appearing in matrix C_{BA}, which are defined as

$$C_{BA}^{kj} = \int_{\Gamma_B} \Psi_B^k(x) \Psi_A^j(x) \, dx, \tag{2.31}$$

have to be evaluated. The integrals can be computed using Gauss integration [37, 85] where an overlay mesh has to be created to ensure an exact evaluation of the integral.

The overlay mesh is actually a kind of intersection of the two meshes. Within each element of this overlay mesh the basis function on both sides of the interface (Ψ_B and Ψ_A) are continuous, which is needed for an accurate Gauss integration. This results in the following evaluation

$$C_{BA}^{kj} \approx \sum_{i=1}^{n_{\text{over}}} \sum_{g=1}^{n_{gp,i}} w_g \Psi_B^k(x_{g,i}) \Psi_A^j(\Pi_A(x_{g,i})), \tag{2.32}$$

where n_{over} is the number of overlay cells, $n_{gp,i}$ is the number of Gauss quadrature points x_g in overlay cell i; w_g the weight of the g^{th} quadrature point and $\Pi_A(x_{g,i})$ the orthogonal projection of $x_{g,i}$ from mesh B on mesh A. The projection between the two meshes is needed, because Ψ_A is only defined on Γ_A and the integral is taken over Γ_B. The procedure to calculate (2.32) consists of the following five steps (see also Figure 2.2):

1. Reconstruct the interfaces of mesh A and B by using their underlying basis functions.
2. Orthogonally project the nodes of mesh A onto the reconstructed interface of mesh B.
3. Define the overlay mesh as the nodes of mesh B together with the projected nodes of mesh A.
4. Define Gauss quadrature points within each cell of the overlay mesh.
5. Project the quadrature points back to the reconstructed interface of mesh A and evaluate the basis function in that point.

The orthogonal projection has to accurately compute the normal with respect to the reconstructed interface, otherwise the order of the total interpolation decreases. For linear basis functions the normal with respect to the reconstructed interface is equal to the normal with respect to the line connecting two mesh points, but for higher order basis functions this is not the case as is depicted in Figure 2.2 for quadratic basis functions. This means that for higher order discretizations the projection is more elaborate. The number of Gauss points to be used should be chosen equal to the underlying order of the discretization.

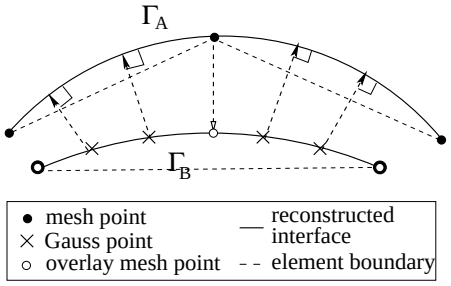


Fig. 2.2: Orthogonal projection and overlay mesh.

Nearest Neighbour with projection (NN proj)

Instead of using a weighted residual method also a collocated Lagrange multiplier method can used. In this case the lagrange multiplier is chosen to be $\lambda(x) = \sum_{k=1}^{n_\alpha} \delta_k$, with δ_k the Dirac-delta function. The collocated Lagrange multiplier method is similar to Nearest Neighbour interpolation, but instead of taking the value from the closest node x_B in the other mesh, the node x_A is orthogonally projected on that mesh and the reconstructed value of the quantity of interest in that point is taken. This is actually the same approach as used in [46]. The transformation matrix H is no longer a Boolean matrix, but still the error for the pressure only converges when the conservative approach is used, when the grids converge to a matching and conforming mesh.

2.2.3 Radial basis function interpolation (RBFI)

Interpolation with radial basis functions (RBF's) has become a very powerful tool in multivariate approximation theory through scattered data, because of its excellent approximation properties [33]. They have been successfully applied to areas as diverse as computer graphics [34], geophysics [19, 18], mesh deformation [27, 95, 96, 97], error estimation [73] and the numerical solution of partial differential equations [71, 72]. They can also be used to interpolate between non-matching meshes in FSI computations [13, 100, 101]. The quantity to be transferred from mesh A to mesh B is approximated by a global interpolation function which is a sum of basis functions

$$w_i(\mathbf{x}) = \sum_{j=1}^{n_A} \gamma_j \phi(||\mathbf{x} - \mathbf{x}_{A_j}||) + q(\mathbf{x}), \qquad w_i = \{\mathbf{u}_i, p\mathbf{n}_i\}, \quad i = \{1, .., d\}, \tag{2.33}$$

where $\mathbf{x}_{A_j}$ are the centres in which the values are known, in this case the nodes at the interface of mesh A, q a polynomial, ϕ a given radial basis function with respect to the Euclidean distance $||\mathbf{x}||$ and d the dimension of the problem ($d = 2$ in 2D, and $d = 3$ in 3D problems). The coefficients γ_j and the polynomial q are determined by the interpolation conditions

$$w_i(\mathbf{x}_{A_j}) = \mathbf{W}^i_{A_j}, \quad i = \{1, .., d\} \tag{2.34}$$

with $\mathbf{W}^i_A$ containing the discrete values of $\mathbf{w}_i$ at the interface of mesh A, and the additional orthogonality requirements for the polynomial

$$\sum_{j=1}^{n_A} \gamma_j s(\mathbf{x}_{A_j}) = 0, \tag{2.35}$$

for all polynomials s with a degree less than or equal to that of polynomial q. The minimal degree of polynomial q depends on the choice of the basis function ϕ. A unique interpolant is given when the basis function is a conditionally positive definite function (Definition 3.1 of [13]). If the basis functions are conditionally positive definite of order $m \leq 2$, as is the case for the functions used in this thesis, a linear polynomial

can be used [13]. A consequence of using linear polynomials is that constant values are exactly interpolated leading to a consistent interpolation. The interpolation function (2.33) is defined in the whole domain in contrast to for example [44], where spline-like polynomials are used for the Lagrange multiplier to glue together nonconforming meshes. The spline-function is then only defined on the non-conforming interface.

The interpolation conditions (2.34) and (2.35) can be written in matrix form as follows

$$\begin{bmatrix} \mathbf{W}_A^i \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_{AA} & Q_A \\ Q_A^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{2.36}$$

with $\boldsymbol{\gamma}$ containing the coefficients γ_j, $\boldsymbol{\beta}$ the coefficients of the linear polynomial q, Φ_{AA} an $n_A \times n_A$ matrix containing the evaluation of the basis function $\phi_{A_i A_j} = \phi(||\mathbf{x}_{A_i} - \mathbf{x}_{A_j}||)$. The matrix Q_A is an $n_A \times (d+1)$ matrix with row j given by $\begin{bmatrix} 1 & \mathbf{x}_{A_j}^1 & \mathbf{x}_{A_j}^2 & \cdots & \mathbf{x}_{A_j}^d \end{bmatrix}$.

In order to obtain the values for the unknown quantity at the interface of mesh B we have to evaluate (2.33) in the nodes on the interface of mesh B which can be written in matrix form as

$$\mathbf{W}_B^i = \begin{bmatrix} \Phi_{BA} & Q_B \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{bmatrix}. \tag{2.37}$$

Combining (2.36) and (2.37) gives the relation

$$\mathbf{W}_B^i = \underbrace{\begin{bmatrix} \Phi_{AB} & Q_B \end{bmatrix} \begin{bmatrix} \Phi_{AA} & Q_A \\ Q_A^T & 0 \end{bmatrix}^{-1}}_{\widetilde{H}} \begin{bmatrix} \mathbf{W}_A^i \\ 0 \end{bmatrix}. \tag{2.38}$$

We now can define the transformation matrix H_{BA} as the first n_B rows and n_A columns of matrix $\widetilde{H}$ to obtain $\mathbf{W}_B^i = H_{BA}\mathbf{W}_A^i$. Since matrix H_{AB} only contains spatial information it is equal for each coordinate direction and we can therefore also compute directly $\mathbf{W}_B = H_{BA}\mathbf{W}_A$. Contrary to the weighted residual method and nearest neighbour method, no orthogonal projection and search algorithm is needed to obtain H_{BA}. This is because the radial basis functions are defined in all space and not only on the interface. The computation of H_{BA} only involves the inversion of a relatively small matrix. The number of rows and columns of this matrix is equal to the number of flow or structure points on the fluid-structure interface, which is usually very small compared to the total number of structure and flow points. However, this is a full matrix when the radial basis function does not have compact support. In practice matrix H_{BA} is not computed explicitly, because we are only interested in the value of $\mathbf{W}_B$ which can be obtained by solving system (2.36) and then evaluating the matrix vector product (2.37).

Radial basis functions

RBF's can be divided into two groups, functions with compact support and functions with global support. Beckert and Wendland [13] use for their FSI computations compact supported radial basis functions based on polynomials to interpolate between non-matching meshes. Their C^2 radial basis function gives the best results in their computations [13]. This function is defined as

$$\phi(||\bar{\mathbf{x}}||) = (1 - ||\bar{\mathbf{x}}||/r)_+^4 \, (4||\bar{\mathbf{x}}||/r + 1) \, . \tag{2.39}$$

The subscript + means that only positive values are taken into account (this function is in the remainder of the thesis abbreviated by CP, an abbreviation for Compact Polynomial). The distance between two points is normalized with the largest distance between two points, so $||\bar{\mathbf{x}}|| = ||\mathbf{x}||/||\mathbf{x}||_{\max}$. The radius r defines the support of the radial basis function. A large support radius yields a good approximation order, but then a full matrix system has to be solved. What is more, too large radii lead to singular matrices, because then all the entries of Φ_{AA} are approximately equal to one. A small support radius leads to a well conditioned system with a band matrix that can be easily solved, but the interpolation is less accurate than with a large support radius. For an accurate computation the support radius for a fluid-structure interaction problem should be chosen at least as large as the normalized distance between the centre which is farthest from its neighbours and its nearest neighbour. This nearest neighbour can be in either of the two meshes.

Several global radial basis functions have been tested and evaluated for analytical interpolation tests as well as real fluid-structure interaction computations by Smith et al [100, 101]. From this work the following two functions are shown to be the most robust, cost effective and accurate of the methods tested:

- Multi-quadric Biharmonic spline (MQ)

$$\phi(||\bar{\mathbf{x}}||) = \sqrt{\|\bar{\mathbf{x}}\|^2 + a^2}. \tag{2.40}$$

- Thin-plate spline (TPS)

$$\phi(||\bar{\mathbf{x}}||) = ||\bar{\mathbf{x}}||^2 \, \ln ||\bar{\mathbf{x}}||. \tag{2.41}$$

Both functions do not vanish when $||\bar{\mathbf{x}}||$ goes to infinity as is the case for the compact supported basis function. The MQ-method uses a parameter a that controls the shape of the basis functions. A large value of a gives a flat sheetlike function, while a small value of a gives a narrow cone-like function. In literature it is still an open question how to find the optimal value of a. Smith et all choose a typically in the range $10^{-5} - 10^{-3}$ when a domain of size 1 is used [100, 101]. In this thesis we use the value $a = 10^{-3}$. In contrast with the radial basis functions used by Beckert and Wendland, these two functions are defined on the entire domain. As a result, always a full matrix system has to be solved.

Conservative approach

Due to the addition of the linear polynomial constant values are exactly recovered, and therefore the interpolation is consistent. However, when the conservative transfer approach is used, assuming that RBF interpolation is used for the displacement, the interpolation is not consistent for the transformation of pressure values. The reason for this is shown below for positive definite functions (as for example the MQ). For positive definite functions the addition of the linear polynomial is not necessary to make the system uniquely solvable. In this case we can write, according to equation (2.38), for the transformation of displacements

$$\mathbf{U}_f = \Phi_{fs}\Phi_{ss}^{-1}\mathbf{U}_s. \tag{2.42}$$

According to the following requirement for the pressure must hold with the conservative approach

$$M_{ss}\mathbf{P}_s = \Phi_{ss}^{-1}\Phi_{fs}^{T}M_{ff}\mathbf{P}_f. \tag{2.43}$$

We replace the pressure vectors $\mathbf{P}_s$ and $\mathbf{P}_f$ with the vectors with all constant components $\boldsymbol{\beta}_s$ and $\boldsymbol{\beta}_f$, respectively, and obtain

$$M_{ss}\boldsymbol{\beta}_s = \Phi_{ss}^{-1}\Phi_{sf}M_{ff}\boldsymbol{\beta}_f. \tag{2.44}$$

If non-equidistant grids or higher order basis function are involved in the flow domain, the pressure force on the flow side, $M_{ff}\boldsymbol{\beta}_f$, can be highly oscillatory. The multiplication with $\Phi_{ss}^{-1}\Phi_{sf}$ is a smoothing operation, leading to an overall smooth right hand side. However, the pressure force on the structure side, $M_{ss}\boldsymbol{\beta}_s$, can also be highly oscillatory if non-equidistant grids or higher order basis functions are used in the structure domain. In this case equation (2.44) does not hold.

If for example third order basis functions are used on an 1D equidistant grid, as is the case for the test cases in this thesis, the pressure force on the flow side $M_{ff}\boldsymbol{\beta}_f$ alternates with the values $\frac{2}{3}\beta\Delta x_f$ and $\frac{4}{3}\beta\Delta x_f$, where Δx_f is the grid size on the flow side. The multiplication with $\Phi_{ss}^{-1}\Phi_{sf}$ results in a vector with all constant values $\beta\Delta x_s$, with Δx_s the grid size on the structure side. However, for a constant pressure the pressure force on the structure side, $M_{ss}\boldsymbol{\beta}_s$, should be alternating with the values $\frac{2}{3}\beta\Delta x_s$ and $\frac{4}{3}\beta\Delta x_s$. This means that the error in pressure forces $\boldsymbol{\epsilon}_F = M_{ss}\boldsymbol{\beta}_s - \Phi_{ss}^{-1}\Phi_{sf}M_{ff}\boldsymbol{\beta}_f$ is alternating with values $\pm\frac{1}{3}\beta\Delta x_s$. The amplitude of this error does decrease when the structure grid is refined, but the error in pressure itself $\boldsymbol{\epsilon}_p = \boldsymbol{\beta}_s - M_{ss}^{-1}\Phi_{ss}^{-1}\Phi_{sf}M_{ff}\boldsymbol{\beta}_f$ is alternating with values $\pm\frac{1}{3}\beta$ and its amplitude does not decrease by refining the flow and structure grid simultaneously, only the frequency increases. The error $\boldsymbol{\epsilon}_p$ only converges if the meshes converge to a mesh that is both matching and conforming at the interface, whereby we mean with conforming that also the underlying discretization of the flow and structure mesh must be the same.

The addition of the linear polynomial for conditionally positive definite radial basis functions does not solve this problem as is shown in sections 2.3 and 2.4. The conclusion on the convergence of $\boldsymbol{\epsilon}_p$ actually holds for any transfer algorithm that uses an interpolation scheme and does not incorporate information of the underlying basis functions.

2.2.4 Summary

In this section three different transfer techniques are considered which are commonly found in literature to transfer information between non-matching meshes in FSI computations. For all three methods we derived a conservative and consistent approach. When the consistent approach is used, all three methods interpolate constant values exactly. However, it was found that when the Weighted residual method is used with the conservative approach, a constant pressure is only exactly interpolated when the meshes are conforming. For Nearest Neighbour interpolation or Radial basis function interpolation the meshes do not only have to be conforming, but also matching to ensure that a constant pressure is exactly interpolated with the conservative approach.

2.3 Analytical test problems

In this section the different transfer methods are compared for a smooth and non-smooth analytical problem, to be able to investigate their general interpolation properties. For all the methods both the conservative and consistent approach are employed.

The tests consists of a single transfer of a displacement and pressure field over an interface. This interface has the form of a sine, $q_e = 0.2\sin(2\pi x)$, with $x \in [-0.5, 0.5]$. The profile of the pressure and displacement fields is either smooth or non-smooth. The procedure of the tests is as follows:

1. Start with the continuous form of the displacement and pressure field at the flow or structure side of the interface, respectively.
2. Discretize the continuous fields using a third order order finite element method. Because the number of cells that are used to discretize the interface differ between the flow and structure side, the interface becomes non-matching.
3. Transfer the discretized displacement field from the discrete structure side of the interface to the discrete flow side using one of the transfer methods.
4. Compare the obtained results at the discrete flow interface to the exact values of the displacement field at this interface by looking at the relative transfer error.
5. Transfer the discretized pressure field from the discrete flow side of the interface to the discrete structure side using the conservative or consistent approach.
6. Compare the obtained results at the discrete structure interface to the exact values of the pressure field at this interface by looking at the relative transfer error.

The relative L_2 transfer error is defined as

$$\epsilon = \sqrt{\frac{\sum_{i=1}^{n_\alpha} ||\mathbf{w}_{ex}^i - \mathbf{w}_\alpha^i||^2}{\sum_{i=1}^{n_\alpha} ||\mathbf{w}_{ex}^i||^2}}, \tag{2.45}$$

where $\mathbf{w}_\alpha^i$ is the vector containing the values received at the flow ($\alpha = f$) or structure ($\alpha = s$) side using one of the transfer methods and $\mathbf{w}_{ex}^i$ the vector with the exact values on that side of the interface.

By discretizing the displacement and pressure fields, already an error is made with respect to the continuous field. The relative L_2 discretization error of a continuous function $w_{ex}(x)$ on the flow ($\alpha = f$) or the structure ($\alpha = s$) interface is defined as

$$\epsilon_{disc} = \sqrt{\frac{\int_{\Gamma_\alpha} ||w_{ex}(x) - \sum_{i=1}^{n_\alpha} N_\alpha^i(x)\mathbf{w}_{ex}^i||^2 \, dx}{\int_{\Gamma_\alpha} ||w_{ex}(x)||^2 \, dx}}, \tag{2.46}$$

where $\sum_{i=1}^{n_\alpha} N_\alpha^i(x)\mathbf{w}_{ex}^i$ is the discretized form of $w_{ex}(x)$ using the basis functions of the flow, N_f, or structure, N_s. The integrals are computed with Gauss integration. As long as the transfer error is smaller than the spatial discretization error, the transfer error does not effect the spatial discretization order of the total system.

The applications we are interested in (wing flutter, deforming wind turbine blades) typically have 5 till 10 times more cells on the flow interface than on the structure interface. Therefore we use $n_f = 26 \cdot 2^k$ flow cells and $n_s = 5 \cdot 2^k$ structure cells, with $k \in \{0, 1, 2, 3, 4, 5\}$, leading to a ratio of about 20%.

The transfer error in the displacement field received by the discrete flow side and in the pressure field received by the discrete structure side for both the conservative and consistent approach is investigated for the projection of a smooth and a non-smooth field in the following two sections.

2.3.1 Transferring a smooth field

In this section a smooth discretized displacement field is transferred from the discrete structure side to the discrete flow side of the interface and a smooth discretized pressure field from flow to structure using the different transfer algorithms. The values of the transferred fields are compared to the exact values of the continuous field at the discrete locations. Both fields have the form $w(x) = 0.01\cos(2\pi x)$.

Displacement of the flow boundary

The L_2 transfer error of the displacement (2.45) in the flow points versus the number of structure points after one interpolation step is depicted in Figure 2.3 for NN (with and without projection), CP with $r = 2$ and WR and in Figure 2.4 for the RBFI methods (CP ($r = 0.25$ and $r = 2$), TPS and MQ). Note that the CP method is actually no longer compactly supported with $r = 2$, because then all basis functions cover the whole domain which has length one.

The number next to a line represents the order of the method represented by the line. The gray lines (disc) represent the discretization error (2.46); the solid gray line is the discretization error on the flow interface and the dashed line on the structure interface. Above these lines the transfer error of a method is higher than the discretization error. The interpolation of the displacement is the same for both the conservative and consistent approach.

It can be seen that simple NN is only first order accurate. The error of the WR method and NN with projection are smaller than the discretization error of the structure and are almost on top of each other. For a discretization order higher than two

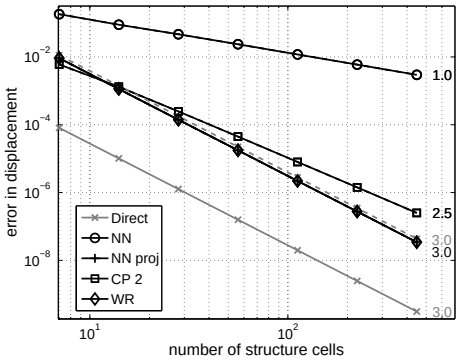


Fig. 2.3: Error in displacement.

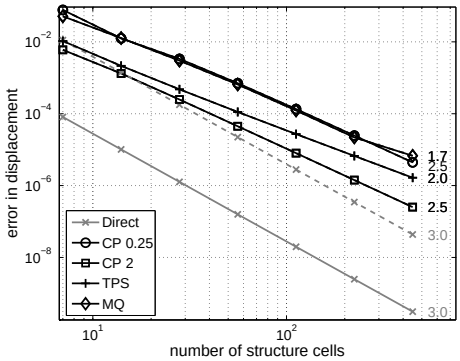


Fig. 2.4: Error in displacement - RBFI methods.

they are the most accurate methods. The CP method has an order of about 2.5, but the accuracy depends on the value of the radius: the larger the radius r, the more accurate the method. The MQ and TPS method are approximately second order accurate where the accuracy of the TPS method is higher. With $r = 2$, CP is more accurate than TPS and with $r = 0.25$ it is comparable to MQ.

Pressure received by the structure

The L_2 transfer error of the pressure versus the number of points on the structure interface after one interpolation step is depicted in Figures 2.5 and 2.6. The solid

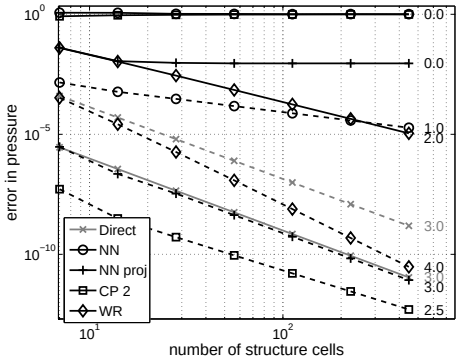


Fig. 2.5: Error in pressure (−: conservative −−: consistent).

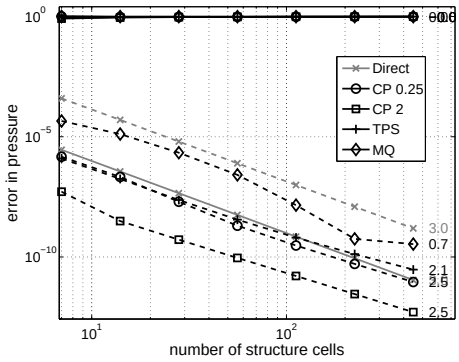


Fig. 2.6: Error in pressure - RBFI methods (−: conservative −−: consistent).

lines are obtained with the conservative and the dashed lines with the consistent approach. In the conservative approach we use the transposed of the matrix used in the previous section to transfer the displacement, whereas for the consistent approach

a new transfer matrix has to be build. It can be seen that, as we expected, only the WR method converges when the conservative approach is used. The order of the conservative WR method is one order lower than the discretization order. When the consistent approach is used, simple NN is again only first order accurate. For all other methods the interpolation error is smaller than the discretization error of the structure. This is due to the fact that the pressure is transferred from the finer flow grid to the coarser structure grid.

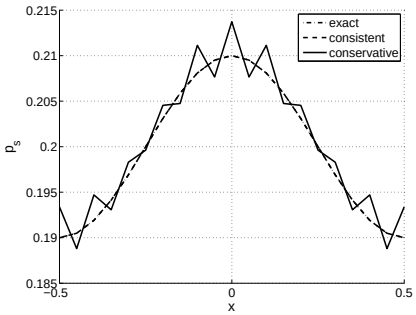


Fig. 2.7: Pressure received by the structure obtained with the WR method for $n_f = 52$ and $n_s = 10$.

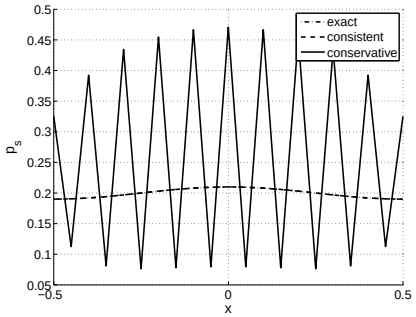


Fig. 2.8: Pressure received by the structure obtained with the CP method with $r = 2$ for $n_f = 52$ and $n_s = 10$.

The reason for the lower order of convergence for the conservative approach can be seen in Figures 2.7 and 2.8, where the exact pressure together with the pressures obtained with the conservative and consistent approach are shown for WR and CP with $r = 2$, respectively. The difference between the exact solution and the one obtained with the consistent approach is barely visible, but large oscillations are visible in the solution obtained with the conservative approach. Except for the WR method, the amplitude of these wiggles does not decrease when the meshes are refined simultaneously (see section 2.2.1 and 2.2.3), leading to the zeroth order convergence. The reason for convergence of the WR is the weak imposition of the interface conditions. The transfer error for WR is caused by the fact that the meshes are non-matching (2.30), and by refining both the flow and structure mesh this error decreases. For the other methods the transfer error only decreases when the meshes become conforming, which is not the case when both meshes are refined simultaneously.

In the the original paper on the mortar element method [14] it is recommended to use a lower order discretization of the Lagrange multiplier for the WR method in the boundary elements, to avoid over-constrained gluing. Therefore we performed the computation also with a second order discretization of the Lagrange multiplier in the boundary elements. The results are almost identical to the ones obtained with the third order discretization in the boundary elements. Therefore we can conclude that the wiggles are not caused by over-constrained gluing, but are due to the non-matching meshes at the interface.

Energy conservation

When energy is globally conserved over the interface, the work exerted on the flow side of the interface, $W_f = \mathbf{U}_f^T \mathbf{F}_f = \mathbf{U}_f^T M_{ff}^T \mathbf{P}_f$, should be equal to the work exerted on the structure side $W_s = \mathbf{U}_s^T \mathbf{F}_s = \mathbf{U}_s^T M_{ss}^T \mathbf{P}_s$ as was shown in (2.2.3). In Figures 2.9 and 2.10 the difference in work exerted on the interface between the flow and structure side, $dW = |W_f - Ws|$, is depicted against the number of structure cells. By construction this difference is zero for the conservative approach (and therefore not shown in the graphs). The difference converges only with order one for the simple NN

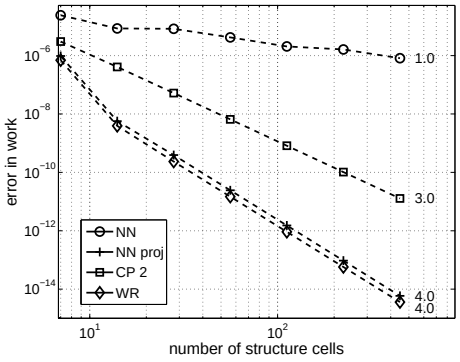


Fig. 2.9: Difference in work for the consistent approach.

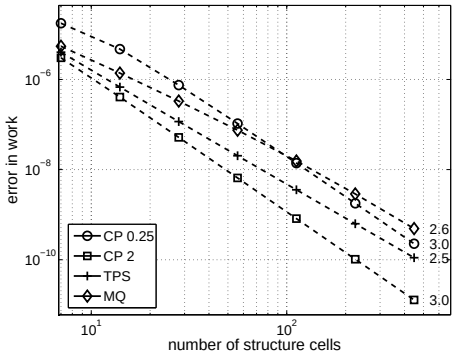


Fig. 2.10: Difference in work for the consistent approach - RBFI methods.

method. For the other methods the difference decreases with approximately one order higher than that of the error in displacement and pressure. So for these methods, even as the consistent approach is not strictly globally conservative for the energy transfer over the interface, the error in the energy transfer decreases consistently with the discretization error.

Efficiency

The efficiency of the transfer method is not the most important issue, because the computation time needed for the transfer is usually much smaller than, for example, the time needed for the flow solve. However, we want to investigate if the difference in efficiency between the methods is considerable. To obtain a global estimation of the efficiency of the methods, the computation time needed to obtain a certain accuracy using Matlab version 7.0.1 on a 3 GHz computer is shown for the displacement and pressure in Figures 2.11 and 2.12, respectively. The closer the line is to the lower left corner, the more efficient the method. For the displacement the conservative approach is most efficient for all methods, because only one transformation matrix has to be calculated. However, when larger problems are considered the storage of the transformation matrix might not be feasible, such that two interpolation problems have to be solved, just as for the consistent approach. For the pressure the highest

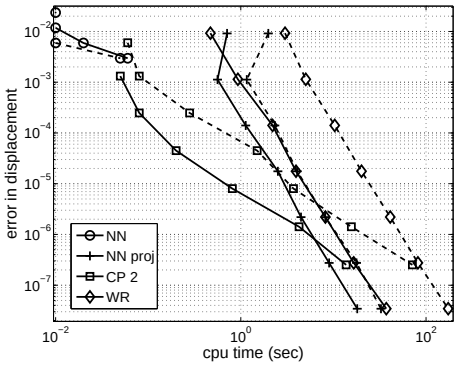


Fig. 2.11: Efficiency for the displacement (−: conservative −−: consistent).

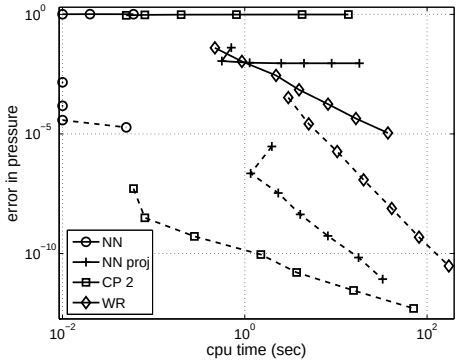


Fig. 2.12: Efficiency for the pressure (−: conservative −−: consistent).

efficiency is obtained using the CP method with $r = 2$. The other RBFI methods show similar results with slightly larger computation times and are not shown in the figures. The WR and NN method with projection are the least efficient because they need a projection algorithm.

For this test case a fast and simple search algorithm is used for the WR and NN methods which performs only a loop over the two closest elements in both directions, therefore the computation time needed by simple NN is very low and does only slightly increase with the number of elements. For the pressure the consistent approach is most efficient, because the conservative approach converges with a lower order (if it converges at all). The main conclusion for this simple problem is that the WR method, although it is more accurate for higher order discretizations, does need more computation time than the RBFI methods. The computational costs of the WR and NN method with projection also increase when a higher order discretization method is used. For higher order discretizations the projection algorithm becomes more complicated, because more Gauss points are needed and the normal has to be calculated with respect to a higher order polynomial.

2.3.2 Transferring a non-smooth field

In this section the same calculations are performed as in the previous section, but this time a hat-shape function is transferred instead of a cosine, so

$$w(x) = \begin{cases} 0.01\,(2 - |x|/a) & |x| < a, \\ 0.01 & |x| \geq a. \end{cases} \tag{2.47}$$

The value of a is chosen in such a way that the two outer discontinuities are located exactly at a grid point. For the displacement which is defined on the structure interface we use $a = 0.5 - 2/5$ and for the pressure which is located on the flow interface $a = 0.5 - 9/26$. The discontinuity in the centre is always located exactly at a grid point.

Displacement of the flow boundary

The L_2-error of the displacement (2.45) in the flow points versus the number of structure points after one interpolation step is depicted in Figure 2.13. This time the results for the different RBFI methods are not shown separately, because they were all very similar to the ones obtained with CP with $r = 2$. The discretization error

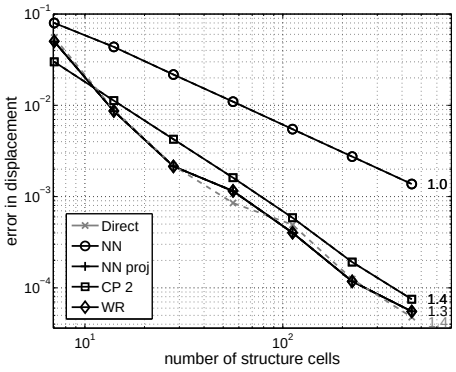


Fig. 2.13: Error in displacement.

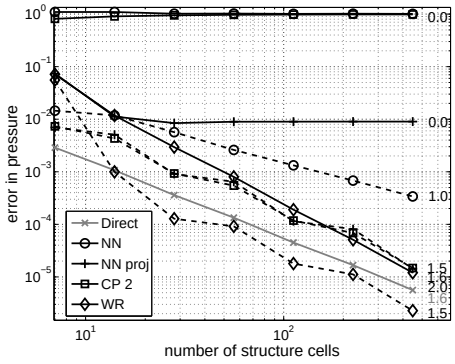


Fig. 2.14: Error in pressure (−: conservative −−: consistent).

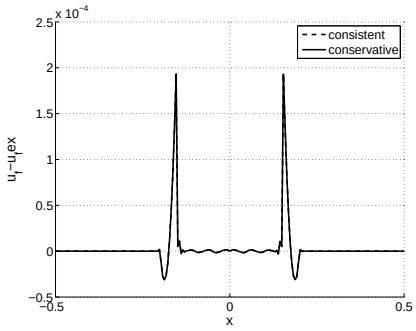


Fig. 2.15: Error in displacement received by the flow obtained with the WR method for $n_f = 104$ and $n_s = 20$.

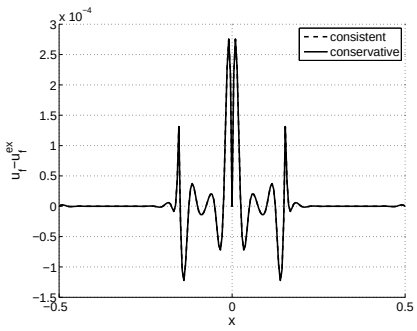


Fig. 2.16: Error in displacement received by the flow obtained with the CP method with $r = 2$ for $n_f = 104$ and $n_s = 20$.

on the structure side is by construction equal to zero and is therefore not depicted in the figure. For the discretization error on the flow side the third order convergence is not obtained, because only the discontinuity in the centre coincides with a grid point. The resulting order of convergence for the discretization error is 1.4.

It can be seen that simple NN is again only first order accurate. For the other methods the interpolation error is almost equal to the discretization error on the flow side. For the WR method and CP method with $r = 2$ the error between the

displacement received by the flow and the exact displacement field, $\mathbf{u}_f - \mathbf{u}_f^{ex}$, is plotted in Figures 2.15 and 2.16, respectively. For the WR method this error has a peak at the outer discontinuities which are not located at a grid point, but at the discontinuity in the centre the error is almost zero. Larger oscillations are visible with the CP method with $r = 2$. These oscillations are caused by the fact that the RBFI method tries to generate a global smooth function through all grid points and therefore the highest peak in the error is visible close to the largest discontinuity which is located in the centre.

Pressure received by the structure

The L_2-error of the pressure versus the number of points on the structure interface after one interpolation step is depicted in Figure 2.14. The solid lines are obtained with the conservative and the dashed lines with the consistent approach. Again only the WR method converges when the conservative approach is used and the order is the same as for the smooth pressure field. When the consistent approach is used, simple NN is again only first order accurate. For all other methods the order of the interpolation error is almost equal to the discretization error, leading to convergence order of 1.5. This time only for the WR method the interpolation error is smaller than the discretization error. The small wiggles in the error convergence of the consistent WR, CP and NN with projection are caused by the fact that the discontinuity is not always located in the same position within a structure element when the grids are refined.

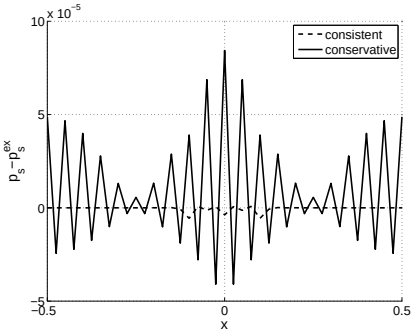


Fig. 2.17: Error in pressure received by the structure obtained with the WR method for $n_f = 104$ and $n_s = 20$.

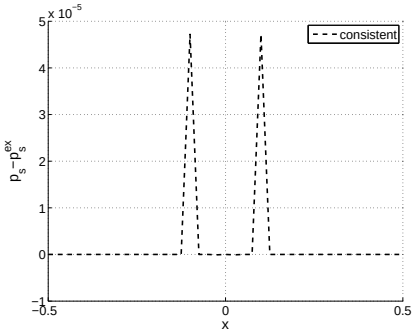


Fig. 2.18: Error in pressure received by the structure obtained with the CP method with $r = 2$ for $n_f = 104$ and $n_s = 20$.

The error between the exact pressure field and the one obtained by the structure, $\mathbf{p}_s - \mathbf{p}_s^{ex}$, is depicted in Figures 2.17 and 2.18 for the WR method and CP method with $r = 2$, respectively. For the CP method only the results for the consistent approach are shown, because with the conservative approach the same large oscillations are obtained as with the transfer of the smooth pressure field and then the error for the consistent approach would not be visible anymore. The error for the conservative WR

method is almost the same as for the smooth case. This is caused by the fact that the transformation matrix is build on the overlay mesh which is based on the discrete flow interface, on which the pressure can be exactly represented. This also explains the second order convergence. For the consistent WR method the transformation matrix is build on the overlay mesh on the discrete structure interface, which can not exactly represent the pressure at the discontinuities and we can see that the error is the largest at these locations. However, this error is much smaller than with the conservative approach. For the consistent CP method the error is also largest at these locations, but we see less oscillations than for the displacement. This is due to the fact that the structure mesh is much coarser than the flow mesh.

Energy conservation

In Figure 2.19 the difference in work exerted on the interface between the flow and structure side is depicted for the consistent approach. The convergence rate for the WR method and the simple NN method are comparable to the smooth field. For the NN method with projection and CP with $r = 2$ the error converges only with order two, but still the error decreases consistently with the discretization error.

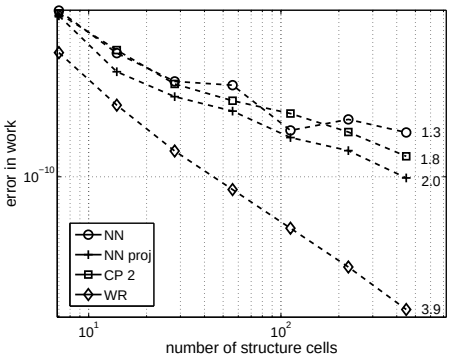


Fig. 2.19: Difference in work for the consistent approach.

2.3.3 Conclusions

The conclusions for the above described test cases also hold for other parameter settings. Overall it can be concluded that for these simple analytical problems the consistent approach is preferred over the conservative approach. Simple NN is not a good choice, because it is only first order accurate. When the discretization order of the total system is higher than two, the WR and NN method with projection are the best choice. However, their implementation is more elaborate due to the projection and search algorithm and the computation time is higher than for the RBFI methods. Therefore, when the discretization of the total system is of order two or lower, the RBFI methods are preferred where CP with $r = 2$ is the best choice.

2.4 Quasi-1D FSI problem

For the investigation of the behaviour of the methods in FSI simulations, where multiple transfers of pressure and displacement are performed, a quasi-1D problem is used. It is chosen such that it allows for the investigation of the problems arising with non-matching meshes. We consider a quasi-1D channel with a flexible wall. The main velocity of the compressible flow is in the x-direction and the structure is modeled as a membrane. The diameter of the channel may vary due to a pressure difference between the pressure in the flow and the pressure behind the wall. Considering only the steady state allows us to analyze the coupling in space separately, excluding errors based on coupling in time. In order to obtain the steady state solution an iterative approach is used. The existence of a numerical steady state solution is determined by computation of a numerically 'exact' solution on a very fine mesh by directly solving the steady state problem on matching meshes.

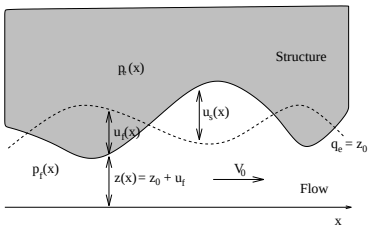


Fig. 2.20: Configuration of the quasi-1D FSI problem.

2.4.1 Flow equations

A simple flow model is used which is valid for supersonic flow over a panel:

$$p_f = -\rho_0 c_0 V_0 \partial_x z, \tag{2.48}$$

with ρ_0, c_0 and V_0 the density, speed of sound and velocity, respectively, assumed to be constant, p_f the pressure and $z = z_0 + u_f$ the location of the panel which is equal to the initial location of the panel, z_0, plus the displacement from this initial position, u_f.

For convenience the variables are scaled as follows

$$\bar{x} = \frac{x}{L}, \quad \bar{V}_0 = \frac{V_0}{c_0}, \quad \bar{p_f} = \frac{p_f}{\rho_0 c_0^2}, \quad \bar{z} = \frac{z}{L}, \tag{2.49}$$

with L the length of the channel. This results in the following non-dimensional equation:

$$\bar{p}_f = -\bar{V}_0 \partial_x \bar{z}. \tag{2.50}$$

For ease of notation the bars are dropped in the remainder of the thesis. To discretize the equations, a third order finite element discretization is used.

2.4.2 Structure equations

The equation that describes the behaviour of the flexible wall is given by

$$\kappa u_s - T \partial_{xx} u_s = p_s - p_e, \tag{2.51}$$

where u_s is the displacement from the 'dry' equilibrium position, $q_e(x)$ (when $p_s = p_e$); p_s is the pressure acting on the wall, p_e is the pressure behind the wall, assumed to be constant, κ the elasticity per unit length and T the longitudinal tension per unit length. In this test case the 'dry' equilibrium position is equal to the initial location of the panel, so $q_e = z_0$. Again the variables are scaled using the non-dimensional variables of (2.49) and the additional variables

$$\bar{u_s} = \frac{u_s}{L}, \quad \bar{p_s} = \frac{p_s}{\rho_0 c_0^2}, \quad \bar{p_e} = \frac{p_e}{\rho_0 c_0^2}, \quad \bar{\kappa} = \frac{\kappa L}{\rho_0 c_0^2}, \quad \bar{T} = \frac{T}{L\rho_0 c_0^2}. \tag{2.52}$$

This results in an equation which has two non-dimensional physical parameters $\bar{\kappa}$ and $\bar{T}$ and has the same form as (2.51). In the remainder of the thesis the bars are dropped. Again a third order finite element discretization is used to discretize the equations.

2.4.3 Coupling procedure

Coupling between the fluid and the structure equations is obtained through the kinematic (2.1a) and dynamic (2.1b) boundary conditions at the fluid-structure interface. A simple iterative coupling procedure is implemented to obtain the steady state solution. This iterative approach proceeds as follows

1. Calculate $\mathbf{p}_s = H_{sf}\mathbf{p}_f$.
2. Calculate the new displacement of the structure, $\mathbf{u}_s$ from (2.51).
3. Obtain $\mathbf{u}_f = H_{fs}\mathbf{u}_s$ and update the location of the wall $\mathbf{z} = \mathbf{z}_0 + \mathbf{u}_f$.
4. Calculate the new pressure in the flow $\mathbf{p}_f$ from (2.50).

These four steps are repeated until the change in $\mathbf{u}_s$ is smaller than a certain threshold.

To obtain a numerically 'exact' solution the steady state problem is solved monolitically on very fine matching meshes ($n_f = n_s = 2000$). When the meshes are matching we have $p = p_f = p_s$ and $u = u_f = u_s$ and therefore we can solve the following equation for the displacement

$$\kappa u + V_0 \partial_x u - T\partial_{xx} u = -V_0 \partial_x q_e - p_e, \tag{2.53}$$

after which the pressure can be evaluated as

$$p = -V_0 \partial_x u. \tag{2.54}$$

In order to obtain the 'exact' solution a fourth order finite element discretization is used to discretize the equations.

2.4.4 Results

For the test cases the following configuration is used. The boundaries of the domain are $x_{\min} = -0.5$ and $x_{\max} = 0.5$ and the initial shape of the tube wall is given by

$$z_0(x) = a_0 - a_1 e^{-a_2 x^2}, \tag{2.55}$$

where the parameters have the values $a_0 = 0.5$, $a_1 = 0.25$ and $a_2 = 80$. This means that we have a smooth converging/diverging channel as shown in Figure 2.21. The 'dry' equilibrium position of the membrane, q_e, is equal to this initial shape.

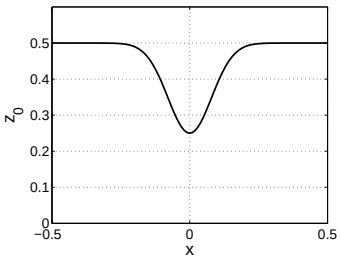


Fig. 2.21: Shape of the quasi-1D channel.

The values used for the non-dimensional structure parameters are: $\kappa = 50$ and $T = 0.04$, which results in a rather flexible membrane. The flow velocity is equal to $V_0 = 3$, corresponding to a supersonic flow of Mach 3. Initially the pressure in the flow, $\mathbf{p}_f$, the pressure behind the wall, $\mathbf{p}_e$, and the displacement $\mathbf{u}_s$ are all equal to zero. We use again $n_f = 26 \cdot 2^k$ flow cells and $n_s = 5 \cdot 2^k$ structure cells, with $k \in \{0, 1, 2, 3, 4, 5\}$, leading to a ratio of approximately 20%.

Instead of using the same method to interpolate displacements and pressures, also different methods can be used as is done in [123]. We therefore investigated also the performance where an accurate method (CP with $r = 2$) is used to transfer from coarse to fine mesh (displacement) and a simpler and faster method (simple NN) from fine to coarse mesh (pressure). The results are shown with the abbreviation CP-NN.

Location of the flow interface

The L_2-error (2.45) of the position of the nodes on the flow interface, which is a measure for the error made in the transfer of the displacement field, versus the number of structure points is depicted in Figures 2.22 and 2.23. The solid line is obtained with the conservative and the dashed line with the consistent approach. The gray lines represent the discretization error (2.46) of the location: the solid gray line is the discretization error on the flow interface and the dashed line on the structure interface.

Simple NN is the least accurate method and the consistent approach for this method does not converge at all. This can be explained by the fact that the error in the energy transfer only converges with order one as was already shown for the analytical test case. When projection is used NN is second order accurate and the conservative approach performs slightly better than the consistent approach. The performance of CP-NN is comparable to NN with projection. The transfer error for the WR method is always smaller than the discretization error for both approaches. The RBFI methods are second order accurate with the consistent approach giving the most accurate results. Only for higher values of n_s the transfer error is larger than the discretization error, when CP or TPS is used. When the discretization of the total system is second order or lower, instead of the third order discretization that is used in the examples in this thesis, the transfer error is smaller than the discretization error for all RBFI methods.

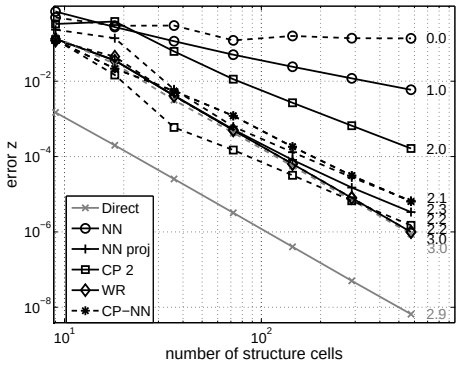


Fig. 2.22: Error in location flow interface (−: conservative −−: consistent).

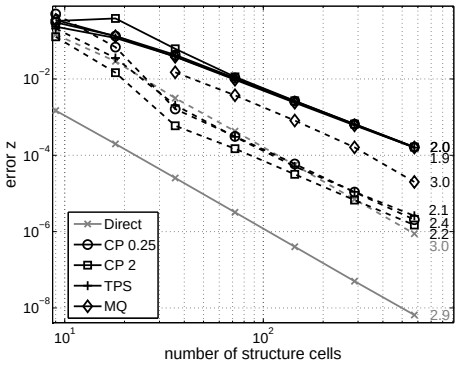


Fig. 2.23: Error in location flow interface - RBFI methods (−: conservative −−: consistent).

Pressure received by the structure

The relative L_2-error of the pressure versus the number of points on the structure interface is depicted in Figures 2.24 and 2.25. Because the value for the pressure is

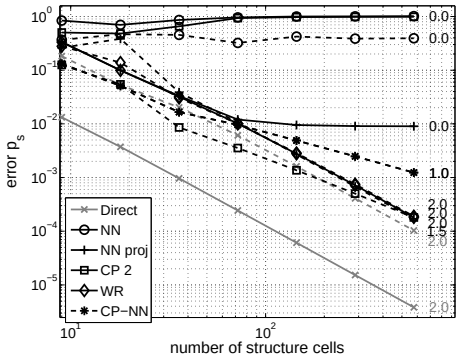


Fig. 2.24: Error in pressure (−: conservative −−: consistent).

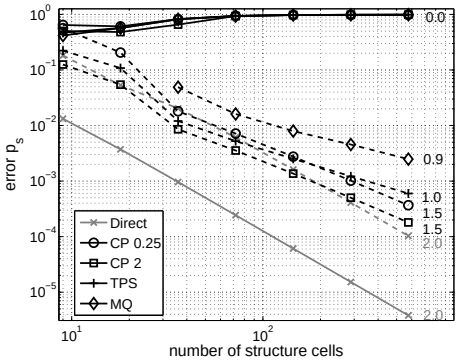


Fig. 2.25: Error in pressure - RBFI methods (−: conservative −−: consistent).

obtained from the spatial derivative of z, the convergence is one order lower than for the displacement. It can be seen that the conservative approach leads again to a zeroth order error for all methods, except for WR. The transfer error for the WR method for both approaches and consistent NN with projection is only a little higher than the discretization error. The reduction in order for the conservative WR as can be seen in the analytical test case is not visible in Figure 2.24, because of the order reduction already caused by the spatial derivative. The consistent RBFI methods are first till 1.5 order accurate where CP with $r = 2$ gives the most accurate results. Only

for larger values of n_s the transfer error is higher than the discretization error, when CP or TPS is used. The CP-NN method is first order accurate which can be explained by the fact that simple NN is used for the pressure transfer.

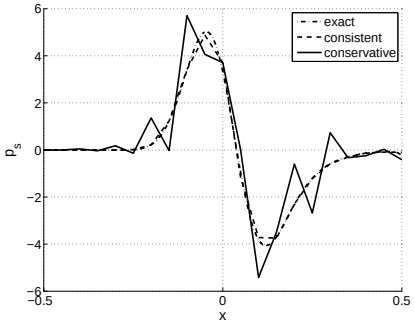


Fig. 2.26: Pressure obtained with CP ($r = 2$) for $n_f = 52$ and $n_s = 10$.

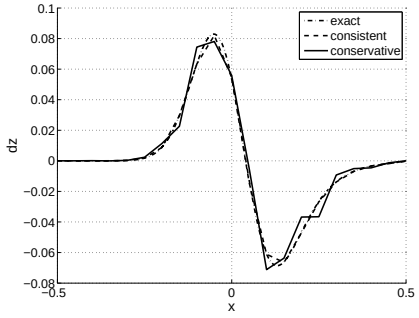


Fig. 2.27: Location of the flow interface obtained with CP ($r = 2$) for $n_f = 52$ and $n_s = 10$.

In Figures 2.26 and 2.27, the exact solution and the ones obtained with the conservative and consistent approach using the CP method with $r = 2$ are shown for the pressure received by the structure, p_s, and the displacement of the flow interface u_f, respectively. It can be seen that the large oscillations in the pressure felt by the structure result in small deviations in the displacement of the flow interface. The more flexible the structure, the larger these deviations become. The results for the difference in work are similar to the ones obtained by the analytical test case.

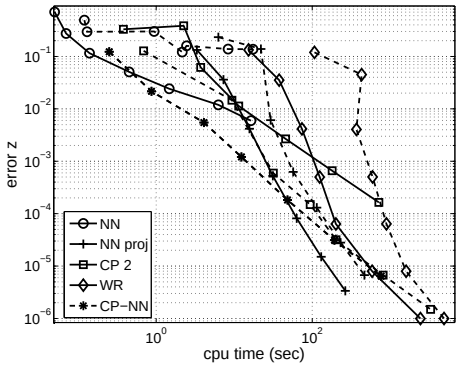


Fig. 2.28: Efficiency for the location of the flow interface (−: conservative −−: consistent).

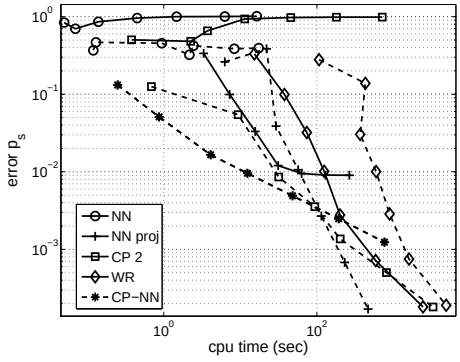


Fig. 2.29: Efficiency for the pressure (−: conservative −−: consistent).

Efficiency

For this simple quasi-1D test problem the computation time needed for the transfer algorithm is still considerable compared to the overall computation time. In Figures 2.28 and 2.29 the total cpu-time is plotted against the error in the position of the flow interface and the pressure on the structure interface, respectively. It can be seen that for an error of 1% in both the location of the flow interface and the pressure received by the structure the CP-NN method is the most efficient. When going to higher dimensions the search and projection algorithms needed by the WR method and NN method with projection will become more elaborate. For the RBFI methods the system to be solved is only dependent on the number of points on the interface (and independent of the dimension of the problem). However, this is generally a system with a large fill-in and therefore the computation time will grow fast with an increasing number of points on the interface if no fast solution techniques [11, 82] are used.

2.4.5 Conclusions

The conclusions drawn for the analytical test case also hold for this steady quasi-1D FSI test case. Furthermore, the NN method is not suitable for the coupling of non-matching meshes in FSI-computations, because the error in energy transfer over the interface does not converge. The conservative approach should be used with the WR method, because it gives the highest accuracy and efficiency. The other methods show large oscillations in the pressure obtained by the structure when the conservative approach is used. For these transfer methods the consistent approach provides the best accuracy and efficiency.

2.5 Conclusions

In this chapter the difference in accuracy and efficiency between a conservative and a consistent approach for different transfer methods is investigated. The performance is investigated for two analytical test problems as well as a simple quasi-1D FSI problem. When the transfer method is based on a weighted residual formulation of the coupling conditions, the highest accuracy and efficiency are obtained with the conservative approach. For other transfer methods the conservative approach results in large unphysical oscillations in the pressure received by the structure. When the structure is flexible enough these oscillations can result in deviations in the displacement of the flow interface. For these methods the consistent approach provides the best accuracy and efficiency. Simple Nearest Neighbour interpolation should not be used, because the error in both displacment and pressure does not converge when the consistent approach is used in the quasi-1D testcase.

Overall, when the discretization order of the total system is higher than two, the WR method is the best choice. However, its implementation is more elaborate and the computation time is higher than for the methods based on radial basis function interpolation. This is because the higher order of the WR method is only obtained

when the projection step is accurately performed. Therefore, when the discretization of the total system is of order two or lower, the methods based on radial basis function interpolation are preferred where the compact RBF with a large support radius is the best choice. Using two different methods to interpolate displacements and pressures proves also to be very efficient. An accurate method (RBF with high support radius) is used to interpolate from coarse to fine and a less accurate, but much faster method (simple Nearest Neighbour) is used to interpolate from fine to coarse.

Up to now only a simple steady quasi-1D fluid-structure interaction problem has been considered. The conclusions drawn on the accuracy of using a conservative or consistent approach for the different methods also hold for 2D and 3D. In these higher dimensions the computation time needed by the transfer algorithm will be small compared to the overall computation time. This is caused by the fact that the fluid-structure interface is one dimension lower than the total domain. Therefore the efficiency of a transfer method is not the most important issue.

In chapter 5 four of the transfer algorithms are applied to a real FSI computation of 2D flow around a cylinder with deformable flap, to investigate their performance for 2D applications and time-dependent problems.

Chapter 3

FLECS: a flexible coupling shell

Computers and numerical algorithms have advanced significantly over the last decade such that the simulation of complex problems involving more than one discipline has become possible. This is good news, as today's research more and more demands dynamic simulation of a complex combination of systems involving multiple physical models with very different length and time scales.

Numerical simulations involving multiple, physically different domains can be solved effectively by coupling multiple simulation programs, or *solvers*. Each solver deals with one particular physical domain, applying the numerical algorithms that are most efficient for that domain. The solvers regularly exchange data to take into account the effects of the other domains.

The coordination of the different solvers is commonly handled by a *coupling shell*. This can be a separate program, or a sub-program (a set of subroutines) embedded in one of the solver programs, or a part of one large program that contains the individual solvers as sub-programs. The coupling shell synchronizes the execution of the solvers and handles the transfer of data from one physical domain to another. In a coupled fluid-structure simulation, for instance, through the coupling shell pressures are transferred from the fluid to the structure interface, and displacements from the structure to the fluid interface. The transfer operation can be a straightforward copy operation in the case that the two solvers use compatible meshes and discretizations. In general, however, the transfer operation involves the execution of a non-trivial interpolation or projection algorithm when non-matching meshes are involved, see chapter 2.

The majority of coupling shells are embedded subprograms that have been developed for coupling two specific solvers, using one or a limited set of transfer algorithms. This makes it hard, if not impossible, to replace one solver by another, or to experiment with new transfer algorithms. One exception is the coupling library MpCCI (Mesh based Parallel Code Coupling) [52], a generic coupling shell that can be used

both as an embedded sub-program and as a separate program. From the users point of view, MpCCI is a collection of subroutines and data types that can be called from a solver program to transfer data between solvers. Although MpCCI is relatively easy to use and provides many advanced features, it is less suitable for a scientific research community. MpCCI only contains a few simple transfer algorithms and no numerical acceleration techniques. These acceleration techniques, such as Aitken sub-iterations [67, 89], Newton-Krylov methods [87], reduced order methods [112] or multi-level techniques [124], are needed to increase the efficiency of partitioned multi-disciplinary computations. However, since MpCCI is a commercial product, the source code is not accessible and therefore the user can not modify the implementation schedule of MpCCI. This means that MpCCI is not suitable for research on developing new data transfer algorithms or acceleration techniques.

The flexible coupling shell Flecs is a generic, open-source coupling shell that tries to overcome the disadvantages of MpCCI. We developed Flecs jointly with the Dutch Centrum Wiskunde & Informatica (CWI), and Habanera, a small company specialized in numerical software development. The contribution of the author of this thesis to this development consists of thinking along about the functionality of Flecs from the user's point of view, the implementation of four transfer algorithms and the application of the coupling shell to perform a real fluid-structure interaction simulation with a separate flow and structure solver. The aim of Flecs is not to achieve the best possible efficiency or to support a large feature set, but to provide a flexible platform for developing new data transfer algorithms and coupling schemes. Another advantage of Flecs is that numerical algorithms implemented within a Flecs coupling server can be reused when using different solvers. A similar setup has been started in Munich by the development of FSIce [91], but there the focus lies on fast search algorithms for the coupling between Cartesian meshes.

Since Flecs has to deal with separate solver processes, that have been started independently, it is not possible to use MPI-1 for exchanging data between those processes. For that reason, Flecs will be based on MPI-2 [59]. Also the language interoperability, that enables mixed-language programming, e.g., `C++` and `Fortran 90`, is an interesting feature of MPI-2 for Flecs, enlarging the set of usable solvers.

The remainder of this chapter describes the design of Flecs and explains how Flecs can be used to couple two or more solvers. Section 3.1 starts with a high-level description of Flecs. In particular, it explains how Flecs is composed of a client-side library that is to be linked to the solver programs, and a server-side library that handles the data transfer between the solvers. In section 3.2 a description is given of the client library. The server library, together with the server program and transfer algorithm, is discussed in section 3.3. Finally, section 3.2.1 contains an example of the use of Flecs to couple a flow and structure solver to perform fluid-structure interaction computations.

3.1 Design Overview

The design of FLECS is based on a client-server model in which two solvers, also called clients, communicate with a separate program called the coupling server that is responsible for transferring data from one physical domain to another. This design is minimally intrusive in the sense that each solver program can be started independently, using its standard start-up procedure; there is no need to change the structure of the solver programs. This design also makes it possible to run the solvers on two different computers that are located at different research institutes.

FLECS consists of a client library that is to be called from a solver program, and a server library that can be used to implement a coupling server. The client library provides subroutines for establishing a connection with the coupling server; for describing the setup of the items that have to be coupled; for describing the data that are to be transferred to and from the server; for sending data to the server; and for receiving data back from the server. The server library provides subroutines for establishing a connection with the solver programs; for listening for requests from those solvers; and for handling the communication with the solvers. It does not, however, provide subroutines for transferring data from one domain to another; these must be implemented by the user.

Both the client library and the server library have been implemented in `C`. However, they have been specifically designed in such a way that one can easily implement bindings for other programming languages such as `Fortran 90` or `C++`. This means that one can use the FLECS libraries in programs that have been implemented in other programming languages than `C`. In fact, the solvers and the coupling server can all be implemented in different programming languages.

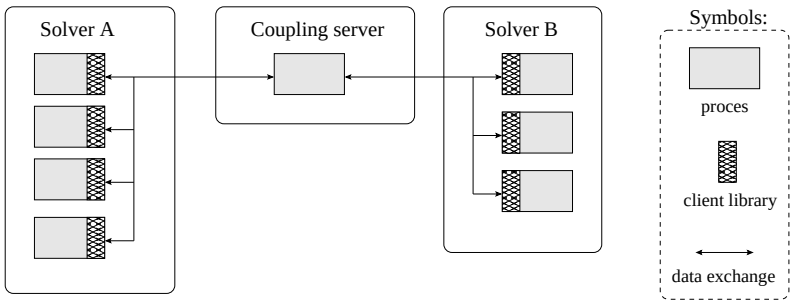


Fig. 3.1: Schematic representation of FLECS.

When a solver program consists of multiple parallel processes, then each process will be linked to its own copy of the client library, and will communicate with the coupling server through one of the solver processes. This is illustrated schematically in Figure 3.1. Obviously also data is exchanged between the processes within one solver, which is not indicated in the figure. Because the data transfer is handled in a separate program – the coupling server – it is possible to adopt different parallelization strategies in the solvers and the transfer algorithm. In the simplest case the coupling

server consists of a single process (as in Figure 3.1) that executes the transfer algorithm sequentially. In a more advanced setup, the transfer algorithm is executed in parallel by multiple server processes, as illustrated in Figure 3.2. However, the data exchange with the solvers is handled by a single server process that distributes the data over the other server processes. This is already possible with the current version of FLECS. The ultimate parallel strategy of the solvers and the transfer algorithm

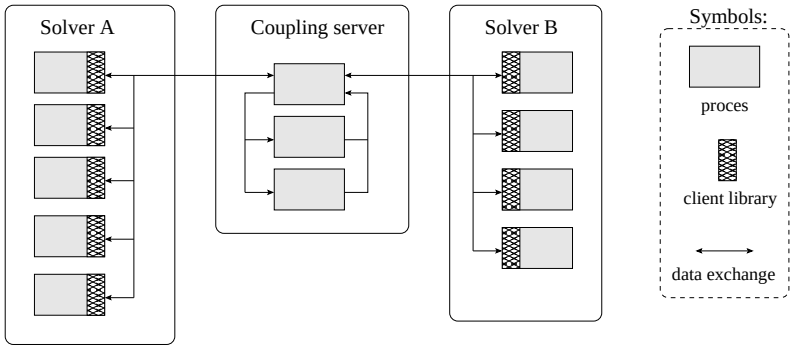


Fig. 3.2: Parallel execution of the coupling server.

is shown in Figure 3.3, where the different processes of the solvers directly exchange data with the (multiple) processes of the coupling server. This strategy is currently

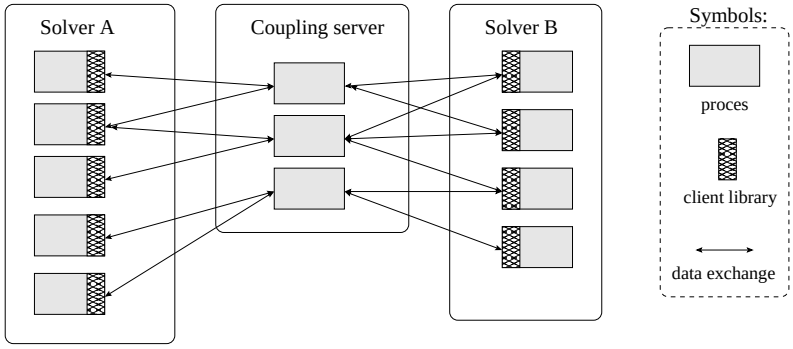


Fig. 3.3: Total parallel execution.

under development.

In order to limit the complexity of the coupling server, it can only couple two solver programs at a time. However, one can couple a solver program to two other solver programs by starting a second coupling server, as illustrated in Figure 3.4. To enable such a setup the client library keeps track of multiple sessions with multiple coupling servers. Whenever a solver process calls one of the subroutines in the client library it has to pass an additional argument that specifies the coupling server with which it wants to exchange data.

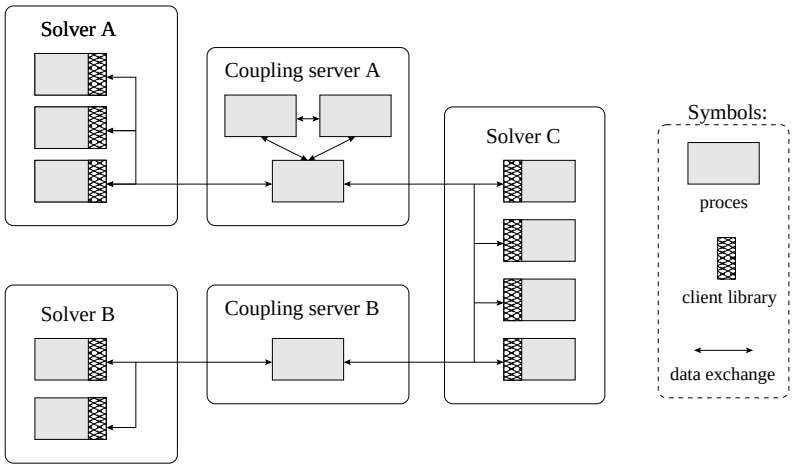


Fig. 3.4: Configuration in which one solver is coupled to two other solvers through two coupling servers.

FLECS uses MPI-2 to transfer data between the solver processes and the coupling server. This has two advantages. First, MPI-2 hides many low-level, system-dependent communication details so that FLECS can easily be ported to different systems. Secondly, MPI-2 implementations typically provide fast communication subroutines that help to minimize the communication overhead. A disadvantage is that the MPI-2 standard is not yet as widely implemented as the MPI-1 standard. Note that the solver program does not have to use MPI-2 itself to exchange data between its own processes, it is only needed in the FLECS functions added to the program.

3.2 The Client Library and its Usage

The client library provides subroutines for establishing a connection with the coupling server; for describing the items that define the coupling; for describing the data that are to be transferred to and from the server; for sending data to the server; and for receiving data back from the server. When a solver program consists of multiple parallel processes, then each process will be linked to its own copy of the client library, and will establish a separate connection with the coupling server.

The FLECS client library exports a number of constants (Table 3.1) and functions (Table 3.2) with which a solver program can setup a connection to a coupling server; define a number of item sets on the interface between two physical domains; define a number of data sets associated with the item sets; send a data set to another solver; receive data from the other solver; and perform a number of miscellaneous tasks, including error handling. Most functions return an integer value that equals zero if no error occurred, and non-zero otherwise. A more detailed description of the the functions and their arguments is given in [80]. The constant `FLECS_SUCCESS` – which

is defined as zero – can be used to check for errors. The function `FLECS_ErrorString` can be used to convert an error code to a human-readable string. By default FLECS terminates a program when an error occurs. This behaviour can be changed by calling the function `FLECS_SetErrorMode`.

Table 3.1: An overview of the constants exported by the FLECS client library.

`FLECS_VERSION`	Expands to the version number of the client library
`FLECS_SUCCESS`	Indicates that a function call completed without errors
`FLECS_NULL_HANDLE`	Represents a null-handle
`FLECS_ERRORS_ABORT`	Indicates that a program should be aborted if an error occurs
`FLECS_ERRORS_RETURN`	Indicates that an error code should be returned if an error occurs
`FLECS_CHAR`	Indicates that an array contains data of type `char`
`FLECS_INT`	Indicates that an array contains data of type `int`
`FLECS_DOUBLE`	Indicates that an array contains data of type `double`

Initialization

To start a coupled simulation, two solvers first have to initialize the client library by calling the function `FLECS_Init`. This function also initializes the MPI library by calling `MPI_Init` if the solver has not already done that, and must be called by all solver processes. No other client function may be called before `FLECS_Init`; the only exception is `FLECS_SetErrorMode`. After the initialization, the solvers must set up a connection to a common coupling server by calling the client function `FLECS_Connect`.

To be able to create a coupling between two solvers, the solvers first have to inform the coupling server about the structure of the coupling. This is done by creating one or more item sets. An item set is for example a description of the grid points that lie on one side of the interface between two physical domains. It is created by calling the function `FLECS_NewItemSet`. With the function `FLECS_UpdateItems` (a part of) a specific item set can be updated or initialized. When the items are relocated over the different processes in a parallel computation, for example after grid adaptation, the function `FLECS_RemapItems` has to be called to pass this information on to the server.

After one or more item sets have been defined on each solver, a coupling between two solvers can be created. A coupling establishes a relation between two item sets such that data can be transferred between them. It can be viewed as a pair of item sets on different sides of an interface, and is created by calling the function `FLECS_NewCoupling`. More than one coupling can be created between two solvers, for example when data has to be exchanged at several grid levels in a multi-level approach. A single item set can also take part in more than one coupling.

The solvers also have to inform the coupling server about the structure of the data that has to be exchanged between them. This is done by creating one or more

Table 3.2: An overview of the functions exported by the FLECS client library

Initialization functions	
FLECS_Init	Initializes MPI, parses the program names, opens the MPI port, publish the name of the server, save the server address
FLECS_Connect	Connects solver and coupling server
FLECS_NewItemSet	Registers set of items on the coupling interface with the coupling server
FLECS_NewCoupling	Defines coupling between an item set on this solver to an item set on another solver
FLECS_NewDataSet	Defines a data set associated with a point set
Finalization functions	
FLECS_Disconnect	Disconnects a solver from a coupling server
FLECS_Shutdown	Calls MPI_finalize and cleans up the allocated memory
FLECS_DelItemSet	Destroys a previously registered item set
FLECS_DelCoupling	Destroys a previously created coupling
FLECS_DelDataSet	Destroys a previously defined data set
Data exchange functions	
FLECS_SendDataSet	Sends adata set to another solver via a coupling server
FLECS_RecvDataSet	Receives a *transformed* data set from another solver via a coupling server
FLECS_Send	Sends an *arbitrary* data array to another solver
FLECS_Recv	Receives an *arbitrary* data array from another solver
Miscellaneous functions	
FLECS_UpdateItems	Updates (a part of) all the items of a registered item set
FLECS_RemapItems	Remaps the items "owned" by the current client process in parallel computations
FLECS_TransferHint	Sends a "hint" (a string) to the transfer algorithm on the server
FLECS_ErrorString	Converts an error code to a human-readable error message
FLECS_SetErrorMode	Specifies how errors are to be handled

data sets. A data set is essentially a two-dimensional floating point array of which the columns are associated with the items in an item set. A data set often represents a physical quantity, like velocity or pressure, but it can also represent data that does not have a direct physical meaning. Data sets are created by calling the function FLECS_NewDataSet. Obviously, more than one data set can be associated with an item set.

Exchanging data

Once the solvers have set up a connection to the coupling server and have defined one or more item sets and data sets, they can exchange data by calling the functions `FLECS_SendDataSet`, `FLECS_RecvDataSet`, `FLECS_Send` and `FLECS_Recv`. The first pair transfers a data set from one solver to the other, and typically invokes a transformation algorithm on the coupling server. The second pair transfers an arbitrary data array between the solvers and does not involve any transformation algorithm. This pair of functions is typically used to communicate convergence and time stepping information between two solvers. Both pairs of functions require that a send operation executed by one solver is matched by a corresponding receive operation executed by the other solver. To be precise, when one solver calls `FLECS_SendDataSet`, it will wait until the other solver has called `FLECS_RecvDataSet`, and the other way around. The same is true for the other pair of functions. An example of the use of the functions of the client library in a coupled fluid-structure interaction computation is given in section 3.2.1.

To improve performance, a solver can tell the coupling server to start a certain computation, or to postpone a computation by calling the function `FLECS_TransferHint`. This is particularly useful when the coupling scheme makes it possible to run certain computations in parallel.

A solver can be coupled to more than one other solver by setting up connections to different coupling servers and by defining multiple item sets and data sets. For instance, to create the configuration in Figure 3.4, solver C makes two calls to the function `FLECS_Connect`: one to set up a connection to Coupling server A, and one to set up a connection to Coupling server B. After that it creates at least two item sets and two data sets, one for each coupling server.

The FLECS client library and coupling server manage a number of internal data structures that are related to client-server connections, item sets and data sets. Each data structure is identified by a unique integer number called a handle. Such a handle is returned to a solver whenever it calls a client function that creates a new internal data structure, and remains valid until the solver calls a function that destroys the internal data structure.

Use with parallel solvers

The client library supports parallel solvers provided that they use the same implementation of the MPI-2 standard as the one used by FLECS. Except when indicated otherwise, the FLECS client functions have collective calling semantics. This means that they must be called by all solver processes at the same point in the solver program. Formulated differently, a collective function blocks the calling process until all other processes have called the same function. To be precise, not all solver processes have to call the client functions, but only those that have called the function `FLECS_Connect`. For example, when in a fluid-structure computation a process of the flow solver does not contain any points on the fluid-structure interaction interface, this process does not have to exchange data with the structure solver and therefore only has to call the function `FLEC_Init`, but no other FLECS functions. In this way it

is possible to limit the data exchanges between two solver programs to those solver processes that manage one or more items on the coupling interface.

3.2.1 Implementation example

In Chapter 5 Flecs is used to couple a flow and a structure solver to perform a fluid-structure interaction simulation. In Table 3.3 the setup of the flow and structure solver are shown after they are adapted to work with Flecs. To be able to perform the coupled computation only function calls from the Flecs client library have to be added, whereas the main structure of the solvers remains the same.

Both solvers start with their own standard initialization followed by the initialization of Flecs. In this example the flow solver creates two different item sets, since the displacement is defined in the vertices of the cells, while the pressure is definined in the cell centres. The structure solver uses only one item set, as both displacement and pressure force are defined in the same location. The item set of the structure is coupled to both item sets of the flow by creating two couplings, one for transferring the displacement (with coupling handle `coupU_id`) and one for transferring the pressure (with coupling handle `coupP_id`). The two data sets (pressure and displacement) are in the structure both associated with the same item set, while in the flow solver they are each associated to a different item set, one located in the cell vertices and the other in the cell centres. During the initialization part the solvers also exchange information about the size and the number of the time steps, convergence criteria, setup of the time integration scheme and the nature of the data to be transferred.

During the time-loop (and sub iterations within a time step) the flow and structure solver repeatedly exchange their data sets and update the item sets. Also convergence information is exchanged to synchronize the two solvers. When both solvers are finished they call the Flecs finalization functions before they invoke their own standard finalization procedure.

For the server we use the minimalistic sequential program shown in section 3.3.1. The functions that make up the data transfer algorithms are implemented separately and the different transfer algorithms mentioned at the end of section 3.3.2 can be used. Both solvers and the transfer algorithm are running sequential in this example, but they can each be located on a different computer.

Table 3.3: Example of the use of FLECS in a Fluid-Structure interaction computation

FLOW SOLVER	STRUCTURE SOLVER
Initialization	
*// * initialize flow solver * //* my_name = "flow"; server_name = "flecs"; FLECS_SetErrorMode(FLECS_ERRORS_ABORT); FLECS_Init(&argc,&argv); FLECS_Connect(server_name,my_name,&conn_id); FLECS_Send(conn_id,&timeStep,1,FLECS_DOUBLE); FLECS_Send(conn_id,&time,1,FLECS_DOUBLE); FLECS_Send(conn_id,&timeIter,1,FLECS_INT); FLECS_Send(conn_id,&nbTimeIter,1,FLECS_INT); FLECS_Send(conn_id,&esd_order,1,FLECS_INT); FLECS_Send(conn_id,&subitNbMax,1,FLECS_INT); FLECS_Send(conn_id,&subitCrit,1,FLECS_DOUBLE); FLECS_Send(conn_id,&press_out,1,FLECS_INT); FLECS_NewItemSet(conn_id,"coords_disp",nbDisp, pointU_ids,&isetU_id); FLECS_NewItemSet(conn_id,"coords_press",nbPress, pointP_ids,&isetP_id); FLECS_NewDataSet(isetP_id,"pressure",1,&dsetP_id); FLECS_NewDataSet(isetU_id,"displacement",dim, &dsetU_id); FLECS_UpdateItems(isetU_id,coords_disp); FLECS_UpdateItems(isetP_id,coords_pres); FLECS_NewCoupling(isetU_id,&coupU_id); FLECS_NewCoupling(isetP_id,&coupP_id);	*// * initialize structure solver * //* my_name = "structure"; server_name = "flecs"; FLECS_SetErrorMode(FLECS_ERRORS_ABORT); FLECS_Init(&argc,&argv); FLECS_Connect(server_name,my_name,&conn_id); FLECS_Recv(conn_id,&timeStep,1,FLECS_DOUBLE); FLECS_Recv(conn_id,&time,1,FLECS_DOUBLE); FLECS_Recv(conn_id,&timIter,1,FLECS_INT); FLECS_Recv(conn_id,&nbTimeIter,1,FLECS_INT); FLECS_Recv(conn_id,&esd_order,1,FLECS_INT); FLECS_Recv(conn_id,&subitNbMax,1,FLECS_INT); FLECS_Recv(conn_id,&subitConv,1,FLECS_DOUBLE); FLECS_Recv(conn_id,&press_in,1,FLECS_INT); FLECS_NewItemSet(conn_id,"coords_struc",nbStruc, point_ids,&iset_id); FLECS_NewDataSet(iset_id,"pressure",1,&dsetP_id); FLECS_NewDataSet(iset_id,"displacement",dim, &dsetU_id); FLECS_UpdateItems(iset_id,coords_struc); FLECS_NewCoupling(iset_id,&coupU_id); FLECS_NewCoupling(iset_id,&coupP_id);
During timestepping and subiterations	
*// * solve flow equations * //* FLECS_SendDataSet(coupP_id,dsetP_id,press); FLECS_Recv(conn_id,&subitFinished,1,FLECS_INT); FLECS_RecvDataSet(coupU_id,dsetU_id,disp); *// * update flow mesh * //* FLECS_UpdateItems(isetU_id,coords_disp); FLECS_UpdateItems(isetP_id,coords_pres);	 FLECS_RecvDataSet(coupP_id,dsetP_id,press); *// * solve structure equations * //* FLECS_Send(conn_id,&subitFinished,1,FLECS_INT); FLECS_SendDataSet(coupU_id,dsetU_id,disp); FLECS_UpdateItems(iset_id,coords_struc);
Finalization	
FLECS_DelCoupling(coupU_id); FLECS_DelCoupling(coupF_id); FLECS_DelItemSet(psetF_id); FLECS_DelItemSet(psetU_id); FLECS_Disconnect(conn_id); FLECS_Shutdown(); *// * finalize flow solver * //*	FLECS_DelCoupling(coupU_id); FLECS_DelCoupling(coupF_id); FLECS_DelItemSet(pset_id); FLECS_Disconnect(conn_id); FLECS_Shutdown(); *// * finalize structure solver * //*

3.3 The Coupling Server

The coupling server consists of two parts, as shown in Fig. 3.5: a *communication and coordination layer*, and a *transfer algorithm*.

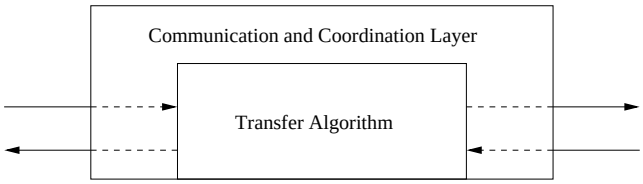


Fig. 3.5: The coupling server, consisting of a communication and coordination layer and a transfer algorithm. The arrows indicate the flow of data between the coupling server and two solver programs.

The communication and coordination layer handles the initialization and finalization of the server; exchanges data between the server and the two solver programs; manages the data structures – including item sets, couplings, and data sets – that have been created by the client library on behalf of the solver programs; and manages the coupling-specific data structures that have been created by a transfer algorithm.

A number of constants (Table 3.4) and functions (Table 3.5) are exported by the server library that can be used to implement a coupling server. A more detailed description of the the functions and their arguments is given in [80].

Table 3.4: An overview of the constants exported by the FLECS server library.

`FLECS_VERSION`	Expands to the version number of the client library
`FLECS_SUCCESS`	Indicates that a function call completed without errors
`FLECS_NULL_HANDLE`	Represents a null-handle
`FLECS_ERRORS_ABORT`	Indicates that a program should be aborted if an error occurs
`FLECS_ERRORS_RETURN`	Indicates that an error code should be returned if an error occurs
`FLECS_LEFT_SIDE`	Identifies the “left” side of a coupling
`FLECS_RIGHT_SIDE`	Identifies the “right” side of a coupling
`FLECS_LEFT_TO_RIGHT`	Indicates that data is to be transferred from the left to the right side of a coupling
`FLECS_RIGHT_TO_LEFT`	Indicates that data is to be transferred from the right to the left side of a coupling

A server program must first initialize the server library by calling the function `FLECS_Init`. After that, it must call the function `FLECS_SetTransfer` to install a data transfer algorithm. Such an algorithm is represented by a struct of type `flecs_transfer_t` containing a number of function pointers, see also section 3.3.2. These functions will be called by the server library on behalf of a solver. Next, the server must set up

Table 3.5: An overview of the functions exported by the FLECS server library

Main functions	
FLECS_Init	Initializes the server library
FLECS_Shutdown	Closes the server library
FLECS_Connect	Sets up a connection between the coupling server and two solvers
FLECS_Disconnect	Ends a connection with the two solver programs
FLECS_MainLoop	Processes one transaction with a solver
FLECS_SetTransfer	Installs a transfer algorithm
FLECS_InitTransfer	Initializes a transfer algorithm
FLECS_NewDataSet	Defines a data set associated with a point set
Miscellaneous functions	
FLECS_SetLogLevel	Controls the amount of informational messages
FLECS_SetServerName	Sets the name of the coupling server
FLECS_SaveServerAddr	Saves the communication address of the server to a file
FLECS_ErrorString	Converts an error code to a human-readable error message
FLECS_SetErrorMode	Specifies how errors are to be handled

a connection with two solver programs by calling the function FLECS_Connect. This function blocks until both solvers have contacted the server. After that, the server must repeatedly call the function FLECS_MainLoop that processes one solver request at a time. If necessary, it will call one of the data transfer functions installed by FLECS_SetTransfer to handle the request. The function FLECS_MainLoop returns a special error code to indicate that both solvers have finished and want to close the connection with the server. The server should then call the function FLECS_Disconnect to close the connection with the solvers. Finally, the server should call FLECS_Shutdown to close the server library and free all resources that have been allocated by the library. After calling this function, no other server functions can be called. More details on the setup of the server program are given in section 3.3.1.

The second part, the transfer algorithm, handles the conversion of a data set from one point set to another point set. It is represented by a small number of functions that are called by the communication and coordination layer to initialize the algorithm; to initialize the data associated with a coupling between two point sets; and to transfer a data set between two point sets. This part of the server is based on a plug-in architecture, that makes it easy to implement new transfer algorithms, see also Sect. 3.3.2. Although these functions have been written in the programming language C, the transfer algorithm itself can be implemented in any programming language.

By splitting the coupling server into two parts, one can experiment with different types of transfer algorithms without having to worry about non-essential details such as communication between the server and the solver programs.

3.3.1 The Server Program

The server program itself is in principle not given, but must be programmed by the user using the functions of the server library. In this way the coupling server can be easily adapted to specific needs. An example of a minimalistic server program is shown below in Program 1. To keep the code clear, the actual data transfer algorithm is not included. That is, the functions making up a data transfer algorithm (NewCoupling, DelCoupling, etc.) are declared in a separate header file `transfer.h` and are implemented elsewhere. More details on the transfer algorithm are given in section 3.3.2.

```
#include <flecs_server.h>
#include "transfer.h"

int main ( int argc, char** argv )
{ flecs_transfer_t  transf;
  int               result;

  FLECS_InitTransfer ( &transf );
  transf.NewCoupling  = NewCoupling;
  transf.DelCoupling  = DelCoupling;
  transf.InitItems    = InitItems;
  transf.UpdateItems  = UpdateItems;
  transf.TransferData = TransferData;

  FLECS_SetTransfer  ( &transf );
  FLECS_SetErrorMode ( FLECS_ERRORS_ABORT );
  FLECS_Init         ( &argc, &argv );
  FLECS_Connect      ( );

  do
  { result = FLECS_MainLoop ( ); }
  while ( ! result );

  FLECS_Shutdown ();
  return FLECS_SUCCESS;
}
```

Program 1: Simpel server program

The `do while` loop is executed repeatedly until `result` becomes `FALSE`. The subroutine `FLECS_MainLoop` '*listens*' if there is a message to be received, the server copies the data out of the send buffer and will act accordingly. Assume that `FLECS_MainLoop` receives a message that a new coupling (e.g., the application demands a multigrid approach) has to be made, because solver A has called `Flecs_NewCoupling`. Then the coupling server expects a similar request by solver B, followed by new item sets and associated data sets for both solvers. This means that the user has to take care that the calls to FLECS routines, included in the programs of the solvers, match, otherwise he/she will receive a (human-readable) error.

A coupling server may be started with multiple parallel processes so that one can implement a parallel data transfer algorithm. However, in the current version only one

server process (the ROOT) will communicate with the solver programs and will call the transfer algorithm functions. These latter functions are responsible for sending data to and collecting data from the other server processes. Some server functions, like `FLECS_Connect`, must be called by all server processes. These functions synchronize the error status so that the same error code is returned on each process. The example program in Program 2 shows how a parallel coupling server could be implemented. All processes call the functions `FLECS_Init` and `FLECS_Connect` to initialize the server library and to set up a connection with the solver programs. After that, only the ROOT calls `FLECS_MainLoop` to process solver requests. All the other processes wait for data to be processed from the ROOT. Once the ROOT has notified the workers that they should stop, all processes close the connection with the solvers and close the server library.

```
int main ( int argc, char** argv )
{
  int my_rank;
  int ierror;

  FLECS_Init    ( &argc, &argv );
  MPI_Comm_rank ( MPI_COMM_WORLD, &my_rank );

  /* Initialize the transfer algorithm ... */

  FLECS_Connect ();

  if ( root )
  {
    do
    {
       ierror = FLECS_MainLoop ();
    }
    while ( ierror == 0 );

    stop_workers ();
  }
  else
  {
    worker_task ();
  }

  FLECS_Disconnect ();
  FLECS_Shutdown ();

  return FLECS_SUCCESS;
}
```

Program 2: Parallel server program

The communication between the ROOT and the other processes is not shown here. This takes place in the data transfer algorithm functions, and the function

worker_task and stop_worker. The latter are not part of the server library and are assumed to be implemented elsewhere. The initialization of the transfer algorithm has been omitted for brevity; see the previous example program.

3.3.2 The transfer algorithm

The actual data transfer algorithm must be implemented by the user. This can be done by specifying the functions (see Table 3.6) that are called from the server function FLECS_MainLoop. The FLECS server library uses a struct of type flecs_transfer_t which contains pointers to these functions.

Table 3.6: The function pointers in a struct flecs_transfer_t that make up a transfer algorithm

Essential transfer functions	
NewCoupling	Creates a new coupling between two point sets
DelCoupling	Deletes a coupling
InitItems	Initializes the items on one side of a coupling
UpdateItems	Updates (a part of) all the items on one side of a coupling
TransferData	Transfers data from one side of a coupling to the other side
Optional transfer functions	
UseHint	Handles a "hint" sent by one of the solvers

The first five functions are obligatory. The function NewCoupling should allocate the data structures that are necessary to transfer a data set from one side of a physical domain to the other side. It will be called when two solvers create a new coupling between two point sets. For initializing the data structure of the items on one side of a coupling the function InitItems is used. This function is called twice after the transfer function NewCoupling has been called; once for the first, or 'left' side of the coupling, and once for the second, or 'right' side of the coupling. During the computation the items of one side of a coupling have to be updated by the function UpdateItems. It will be called at least twice after the transfer function InitItems has been called to initialize the items; once for the 'left' side of the coupling and once for the 'right' side of the coupling. It may be called more than once if one of the solvers updates its items. The actual data transfer is handled by the function TransferData. It transfers a data set from one side of a coupling to the other side. This function will be called when the two solvers have called the client function FLECS_SendDataSet and FLECS_RecvDataSet.

The function UseHint is optional; if not specified, FLECS will use a dummy function that does nothing. It can be used to improve performance, a solver can tell the coupling server to start a certain computation, or to postpone a computation. This is particularly useful when the coupling scheme makes it possible to run certain computations in parallel.

Currently, the functions are implemented for several transfer algorithms given in chapter 2. Consistent simple nearest neighbour interpolation (section 2.2.1), consistent

and conservative radial basis function interpolation (section 2.2.3) and a mix of nearest neighbour interpolation and consistent radial basis function interpolation (see also section 2.4.4) can be used for the transfer of information.

3.3.3 Acceleration techniques

For strongly coupled multi-disciplinary problems, sub-iterations are needed within one time step to stabilize the computation. Since performing one iteration in these coupled problems is very computationally demanding the use of acceleration techniques is needed to limit the number of sub-iterations in order to increase the efficiency of the computations. When higher order time integration methods are used, which are proven to be very efficient for FSI simulations [123], the larger time step allowed by these methods gives the need to use even more sub-iterations within one time step and reducing their number is crucial for further efficiency improvements.

To reduce the number of sub-iterations Aitken acceleration [67, 89], Newton-Krylov methods [87] or reduced order models [112] can be used. All three methods can be implemented as subroutines within the coupling server without having to change the separate solvers involved. The server program only has to store data from previous iterations.

The computational costs of a simple sub-iteration can be reduced by multilevel acceleration techniques [124]. Coupling the two solvers not only on the fine meshes, but also on coarser meshes leads to a significant efficiency improvement, since computations on a coarse mesh are less computatinally demanding. The use of these techniques is simplified by FLECS, as FLECS supports the use of multiple grids on a single interface. In this way the information transfer over the interface at both coarse and fine grid levels can be handled.

Chapter 4

Mesh movement based on radial basis function interpolation

To be able to perform the unsteady flow computations accurately and efficiently, a fast and reliable method is needed to adapt the computational grid to the new domain. Regenerating a grid each time step in an unsteady computation is a natural choice. However, the generation of a complex grid is a time-consuming and nontrivial task. Also remapping the flow state on a new grid involves a loss of accuracy. Therefore, a fast and accurate algorithm is needed to update the grid automatically.

For structured meshes there are efficient techniques available to deform the mesh, such as Transfinite Interpolation [113]. The displacements of points at the boundaries of the mesh are interpolated along grid lines to points in the interior of the mesh. However, these techniques are unsuitable for unstructured grids. The greater flexibility of unstructured grids is required for the meshing of complex domains and grid adaptation. Therefore, we are in this chapter interested in efficient mesh movement techniques for unstructured grids.

Two different mesh movement strategies are known for unstructured grids: grid-connectivity and point-to-point schemes. The first exploits the connectivity of the internal grid points. The connection between the grid points is represented for example by springs [6, 43, 39] or as solid body elasticity [84]. Special instances of this continuous approach include moving grids based on Laplacian and Biharmonic operators [64]. All the methods based on grid connectivity involve solving a system of equations involving all the flow points and can therefore be quite expensive. Hanging nodes, encountered in unstructured meshes when only one of the adjacent elements at an edge is subdivided, require special treatment.

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

is straightforward. This might be especially useful when Finite Volume flow solvers are adapted to deal with moving meshes, because they generally do not already incorporate efficient algorithms to deform the mesh with a pseudo-structural approach. However, until now point-by-point schemes are only applied to the boundary nodes of multi-grid blocks [94]. The interior mesh of the blocks is adapted with fast techniques available for structured grids.

Radial basis functions (RBF's) have become a well-established tool to interpolate scattered data, because of their excellent approximation properties [33]. They have been successfully applied to areas as diverse as computer graphics [34], geophysics [19, 18], error estimation [73] and the numerical solution of partial differential equations [71, 72]. They can also be used in fluid-structure interaction computations to transfer information over the discrete fluid-structure interface, which is often non-matching [13, 100, 101] as is shown in chapter 2. An interpolation function is used to transfer the displacements known at the boundary of the structural mesh to the boundary of the aerodynamic mesh. But why not interpolate the displacement to all the nodes of the flow mesh, instead of only to the boundary? This idea has already been applied to the block boundaries in multi-block grids [94, 103]. There it was mentioned that applying it to the whole internal grid would be computationally very expensive. This is because for the structured part of multi-block meshes much more efficient techniques are known. We want to investigate if interpolation of the displacement with radial basis functions does result in an efficient point-by-point mesh movement scheme for completely unstructured grids. Only recently similar investigations have started at Bristol University with promising results [95, 96, 97].

The objective of this chapter is to develop a new mesh movement scheme for unstructured meshes based on interpolation with radial basis functions. The principle of interpolation with RBF's applied to mesh movement is introduced in section 4.1. There are various RBF's available in literature that can be used for the new method and we want to determine which one generates the best meshes and which one is the most efficient. To be able to compare the deformed meshes generated with the different RBF's, a mesh quality metric is introduced in section 4.2. This metric is used in section 4.3 to determine the best RBF's for our mesh movement scheme, by applying the method to several severe test cases. For one of the test cases the results are also compared with mesh deformation using semi-torsional springs. Furthermore, realistic results on a distorted 2D mesh are presented, where flow computations are performed around a NACA-0012 airfoil. In section 4.4 it is investigated whether incorporating rotational information is beneficial for the new mesh movement strategy. The importance of smooth mesh deformation for higher order time-integration schemes is shown in section 4.5. To show the capability of the new method for 3D applications two test cases are considered in section 4.6. Finally, in section 4.7, different techniques are investigated to improve the overall efficiency of the computation.

4.1 Radial basis function interpolation

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

$$s(\mathbf{x}) = \sum_{j=1}^{n_\mathrm{b}} \gamma_j \phi(||\mathbf{x} - \mathbf{x}_{\mathrm{b}_j}||) + q(\mathbf{x}), \tag{4.1}$$

where $\mathbf{x}_{\mathrm{b}_j} = [x^1_{\mathrm{b}_j}, x^2_{\mathrm{b}_j}, x^d_{\mathrm{b}_j}]$ are the centres in which the values are known, in this case the boundary nodes, with d the dimension, q a polynomial, n_b the number of boundary nodes and ϕ a given basis function with respect to the Euclidean distance $||\mathbf{x}||$. The coefficients γ_j and the polynomial q are determined by the interpolation conditions

$$s(\mathbf{x}_{\mathrm{b}_j}) = \mathbf{d}_{\mathrm{b}_j}, \tag{4.2}$$

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

$$\sum_{j=1}^{n_\mathrm{b}} \gamma_j p(\mathbf{x}_{\mathrm{b}_j}) = 0, \tag{4.3}$$

for all polynomials p with a degree less or equal than that of polynomial q. The minimal degree of polynomial q depends on the choice of the basis function ϕ. A unique interpolant is given if the basis function is a conditionally positive definite function. If the basis functions are conditionally positive definite of order $m \leq 2$, a linear polynomial can be used [13]. In this thesis we only apply basis functions that satisfy this criterion. A consequence of using a linear polynomial is that rigid body translations are exactly recovered.

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

$$\begin{bmatrix} \mathbf{d}_\mathrm{b} \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_{\mathrm{bb}} & Q_\mathrm{b} \\ Q_\mathrm{b}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{4.4}$$

with $\boldsymbol{\gamma}$ containing the coefficients γ_j, $\boldsymbol{\beta}$ the coefficients of the linear polynomial q, Φ_{bb} an $n_\mathrm{b} \times n_\mathrm{b}$ matrix containing the evaluation of the basis function $\phi_{\mathrm{b}_i\mathrm{b}_j} = \phi(||\mathbf{x}_{\mathrm{b}_i} - \mathbf{x}_{\mathrm{b}_j}||)$ and Q_b an $n_\mathrm{b} \times (d+1)$ matrix with row j given by $[\ 1 \quad \mathbf{x}_{\mathrm{b}_j}\ ]$. The possibilities of solving the system in a more efficient way than with a direct solve are discussed in section 4.7.

The values for the displacement in the interior of the flow mesh $\mathbf{d}_\mathrm{in}$, can then be derived by evaluating the interpolation function (4.1) in the internal grid points:

$$\mathbf{d}_{\mathrm{in}_j} = s(\mathbf{x}_{\mathrm{in}_j}). \tag{4.5}$$

The displacement can be interpolated separately for each spatial direction. The process of mesh deformation with RBF's for one direction is visualized in Figures 4.1 and 4.2. The block in the middle is moved to the right and the resulting interpolation function is shown in Figure 4.2. This interpolation function is equal to zero at the

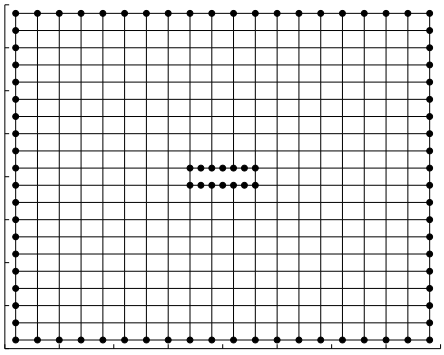


Fig. 4.1: Initial mesh.

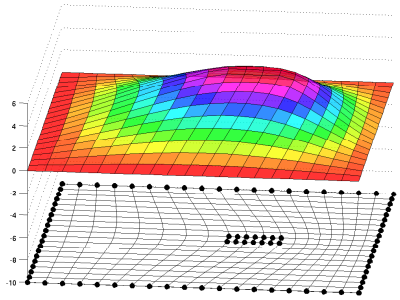


Fig. 4.2: Interpolation function and resulting deformed mesh.

outer boundaries and equal to the displacement of the block at the location of the block boundaries. The displacement of the nodes in the interior of the mesh can then be derived from this interpolation function and the resulting deformed mesh is shown in Figure 4.2. Each individual grid point is moved individually based on its position in space according to the interpolation function, which means that no mesh-connectivity information is needed at all.

The size of the system that has to be solved in (4.4) is equal to $(n_\mathrm{b}+4)\times(n_\mathrm{b}+4)$ which is usually very small compared to the systems that have to be solved in mesh-connectivity schemes. The systems encountered there are approximately as large as $n_\mathrm{in}\times n_\mathrm{in}$, with n_in the total number of mesh points. The total number of mesh points is a dimension higher than the number of points on the boundary of the mesh. However, the systems encountered in the mesh connectivity schemes are very sparse, whereas for the radial basis function interpolation generally a dense system has to be solved. The new moving mesh technique is very easy to implement, even for 3D applications, because no mesh-connectivity information is needed. Also the implementation for partitioned meshes, occurring in parallel flow computations, is straightforward.

Radial basis functions with compact support

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

$$\phi(x) = \begin{cases} f(x) & 0 \le x \le 1, \\ 0 & x > 1, \end{cases} \tag{4.6}$$

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

nr.	**name**	$f(\xi)$
1	CP C^0	$(1-\xi)^2$
2	CP C^2	$(1-\xi)^4(4\xi+1)$
3	CP C^4	$(1-\xi)^6(\frac{35}{3}\xi^2+6\xi+1)$
4	CP C^6	$(1-\xi)^8(32\xi^3+25\xi^2+8\xi+1)$
5	CTPS C^0	$(1-\xi)^5$
6	CTPS C^1	$1+\frac{80}{3}\xi^2-40\xi^3+15\xi^4-\frac{8}{3}\xi^5+20\xi^2\log(\xi)$
7	CTPS C^2_a	$1-30\xi^2-10\xi^3+45\xi^4-6\xi^5-60\xi^3\log(\xi)$
8	CTPS C^2_b	$1-20\xi^2+80\xi^3-45\xi^4-16\xi^5+60\xi^4\log(\xi)$

Table 4.1: Radial basis functions with compact support, from [114].

In Table 4.1 various radial basis functions with compact support are given. In this thesis all compact RBF's are scaled with r, so we use $\xi = x/r$. The first four are based on polynomials [114]. These polynomials are chosen in such a way that they have the lowest degree of all polynomials that create a C^n continuous basis function with $n \in \{0, 2, 4, 6\}$. The last four are a series of functions based on the thin plate spline which create C^n continuous basis functions with $n \in \{0, 1, 2\}$ [114]. There are two possible CTPS C^2 continuous functions which are distinguished by subscript a and b.

Radial basis functions with global support

Functions with global support are not equal to zero outside a certain radius, but cover the whole interpolation space, which leads to dense matrix systems. In Table 4.2 six radial basis functions are given which are frequently used, for example in neural networks, the computer graphics community [34] and for data transfer in fluid-structure interaction computations [100]. The MQB and IMQB methods use a parameter a, that controls the shape of the basis functions. A large value of a gives a flat sheetlike function, whereas a small value of a gives a narrow cone-like function. The value of a is typically chosen in the range $10^{-5} - 10^{-3}$ and in this thesis we use the value $a = 10^{-3}$. More information about RBF's and their error and convergence properties can be found in [33, 116, 115].

nr.	**name**	**abbrev.**	$f(x)$
9	Thin plate spline	TPS	$x^2 \log(x)$
10	Multiquadric Biharmonics	MQB	$\sqrt{a^2 + x^2}$
11	Inverse Multiquadric Biharmonics	IMQB	$\sqrt{\frac{1}{a^2+x^2}}$
12	Quadric Biharmonics	QB	$1 + x^2$
13	Inverse Quadric Biharmonics	IQB	$\frac{1}{1+x^2}$
14	Gaussian	Gauss	e^{-x^2}

Table 4.2: Radial basis functions with global support.

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

4.2 Mesh quality metrics

To be able to compare the quality of different meshes after mesh movement we introduce mesh quality metrics [74]. The mesh quality metrics are based on a set of Jacobian matrices which contain information on basic element qualities such as size, orientation, shape and skewness.

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

The relative size metric measures the change in element size. Let τ be the ratio between the current and initial element volume. The *relative size metric* [74] is then given by $f_{size} = \min(\tau, 1/\tau)$. Essential properties of the relative size metric are: $f_{size} = 1$ if and only if the element has the same total area as the initial element and $f_{size} = 0$ if and only if the element has a total area of zero or infinity. The relative size metric can detect elements with a negative total area (degenerate) and elements which change in size due to the mesh deformation. Actually, the mesh quality does not decrease when all elements change in size equally. Therefore it would be better to measure the change in size relatively to the adjacent elements. However, in the cases used in this thesis the total computational domain is fixed and a decrease in element size in one part of the domain automatically means that the size of at least one other element in the domain is increased.

The skew metric measures the skewness and therefore the distortion of an element. When a node of an element possesses a local negative area, this metric value is set to zero. The expressions for the *skew metric* for triangular, quadrilateral, tetrahedral and quadrilateral elements can be found in [74]. Essential properties of the skew metric

are: $f_{skew} = 1$ if and only if the element has equal angles and $f_{skew} = 0$ if and only if the element is degenerate.

In finite element methods the aspect ratio is also very important for the mesh quality. A high aspect ratio leads to a decrease of the numerical accuracy. This can be taken into account by using the *shape metric* instead of the skew metric [74]. The shape metric is equal to one if and only if the element has equal sides and equal angles and zero if the element is degenerate. However, the main consideration of this thesis is on finite volume methods and therefore the skew metric is used.

To measure both the change in element size and the distortion of an element, the *size-skew metric* [74] is introduced which is defined as the weighted product of the relative size and skew metrics: $f_{ss} = \sqrt{f_{size}} f_{skew}$, since changes in element volume have a smaller influence on the mesh quality than element distortion. Essential properties of the quadrilateral size-skew metric are:

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

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

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

4.3 2D mesh movement

The new mesh movement strategy is tested with the 14 radial basis functions introduced in section 4.1. First, three simple 2D test problems are performed to investigate the difference in quality of the mesh obtained with the RBF's after movement of the boundary. The tests include mesh movement due to rigid body rotation and translation of a rectangle block and deformation of an airfoil-flap configuration. After that the efficiency of the most promising RBF's is investigated. Furthermore realistic results on a distorted mesh are presented, where flow computations are performed around a NACA-0012 airfoil. An incremental approach is used to perform the mesh deformation, where several intermediate steps are performed to arrive at the final mesh. The displacement is always taken with respect to the mesh at the previous step and not to the initial mesh. In this way large deformations are subdivided in a number of smaller deformations.

The quality and robustness of the new method depend on the value of the support radius when a radial basis function with compact support is used. When the support

radius is chosen large enough, the quality and robustness converge to an optimum. Therefore, a relatively high value, r is 2.5 times the characteristic length of the computational domain, is used in the first three test cases, where we only investigate the accuracy of the different RBF's. This means that the support of the radial basis functions cover the whole domain and are not really compact anymore. The effect of varying the compact support radius r on the computation time is investigated for the most promising RBF's in section 4.3.4.

4.3.1 Test case 1: Rotation and translation

The first test case consists of mesh movement due to severe rotation and translation of a block in a small domain. The mesh nodes on the block follow its movement, while the nodes on the outer boundary are fixed. The block has dimension $5D \times 1D$, with D the thickness of the block, and is initially located in the center of a domain which has dimension $25D \times 25D$. The initial mesh is triangular and given in Figure 4.5. The block is translated $10D$ down and to the left and is rotated 60 degrees around the center of the block. The mesh deformation is performed in a variable number of steps between the initial and final location, with a minimum of 1 step and a maximum of 15 steps. The less intermediate steps are taken, the larger the deformation between two steps.

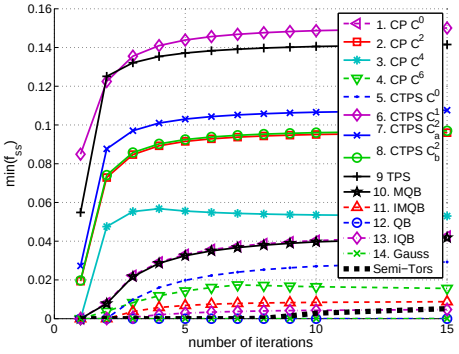


Fig. 4.3: Quality of the worst cell of the mesh for the different RBF's (test case 1).

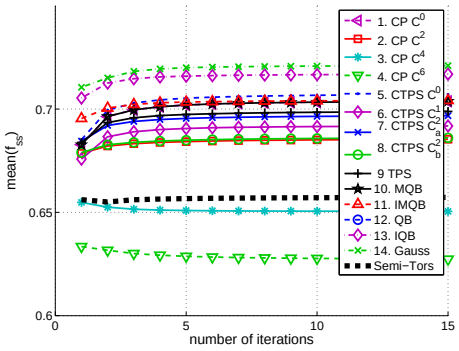


Fig. 4.4: Average quality of the mesh for the different RBFs (test case 1).

The minimum value of f_{ss} after mesh movement with the different RBF's is shown in Figure 4.3 for an increasing number of intermediate steps. It can be seen that for all RBF's the minimum value of f_{ss} indeed increases when more intermediate steps are taken. In Table 4.3 the ranking of the RBFs is given for both the minimum value (min) and the average value (mean) of f_{ss} . The numbers in the table correspond to the numbers of the RBFs which are given in Tables 4.2 and 4.1. Only the five best performing RBFs (nr. 2, 6, 7, 8 and 9) are included. The first row of the table contains the number of the RBF that performs best, the second row the number of the RBF that performs second best, etc. When more RBFs perform equally well, they are listed

on the same line. Figure 4.4 shows the average value of the mesh quality metric. The

Table 4.3: Ranking of the five best RBFs for the three test cases for both the minimum value (min) and the average value (mean) of f_{ss}.

	test-case 1		test case 2		test case 3	
rank	min	mean	min	mean	min	mean
first	6	9	2, 8	2, 8	2, 7, 8	6, 9
second	9	7	7	7, 9	6, 9	7
third	7	6	9	6		2, 8
fourth	2, 8	2, 8	6			

Gaussian basis function (nr. 14), has the best average quality, however, the minimum of f_{ss} for this function is equal to zero. This results in highly distorted meshes as can be seen in Figure 4.6 where the final mesh generated with the Gaussian basis function with 15 intermediate steps is shown. Only cells at a certain distance from the block

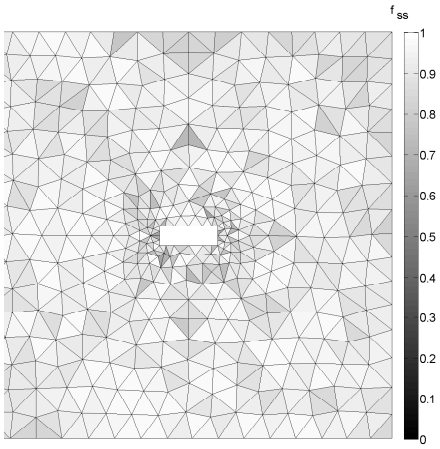


Fig. 4.5: Initial mesh.

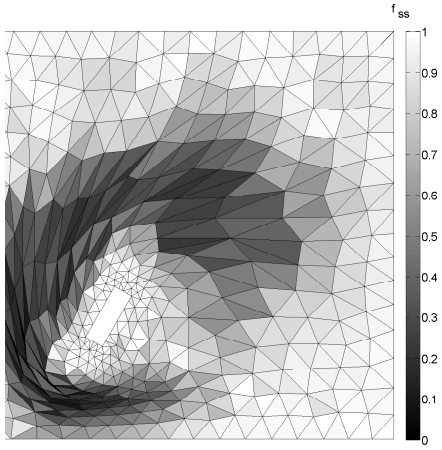


Fig. 4.6: Final mesh using Gaussian basis function with 15 intermediate steps.

are heavily deformed, resulting in a high average mesh quality, but the flow solver will probably crash due to the degenerate cells. The mesh with the highest minimum value for the mesh quality is generated with CTPS C^1 (nr. 6) and is shown in Figure 4.7. The average value of the mesh quality metric is lower than for the Gaussian function, because all cells are deformed, but this results in a much smoother mesh. This will have a positive effect on the accuracy, stability and efficiency of an unsteady flow computation. In the next two test cases we will only consider the RBF's 6, 9, 7, 2 and 8, because they are the most promising.

To show the performance of the new method in comparison with existing mesh moving methods, the mesh deformation is also performed with a method based on

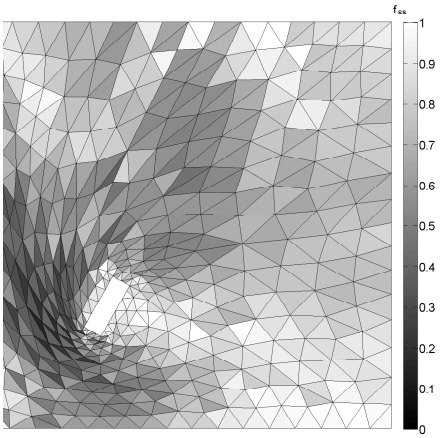


Fig. 4.7: Final mesh using CTPS C^1 with 15 intermediate steps.

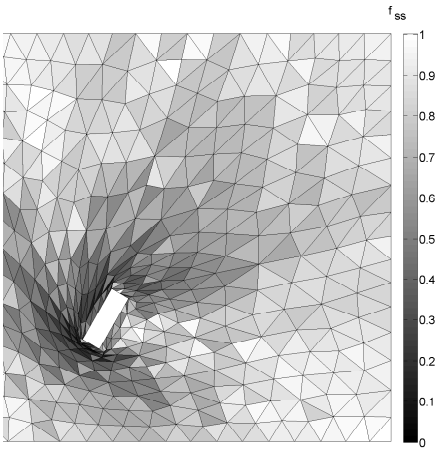


Fig. 4.8: Final mesh using semi-torsional springs with 15 intermediate steps.

semi-torsional springs [119], which is an improvement of the popular spring analogy [6]. This method is based on elastic deformation of element edges where element edges are modeled as springs producing forces to propagate boundary displacements into the interior of the mesh. The formation of mis-shaped elements is penalized by incorporating angle information into the spring stiffness. A boundary improvement technique [22] is implemented which increases the stiffness of springs close to the moving boundary so that surface displacement spreads further into the mesh. In accordance with [119] we imposed one layer of boundary stiffness by increasing the spring stiffness with a factor 3.5. The difference between the semi-torsional spring method and the torsional spring method [39, 43], lies in the fact that the latter models real torsional springs at the element vertices, whereas the former incorporates the angle information in the spring stiffness of the element edges.

The results for the minimal and average value for the mesh quality metric are added to Figures 4.3 and 4.4, respectively (bold dashed line). More than 8 intermediate steps are needed to avoid degenerate cells, where each step means that the total system has to be solved, and the minimum value of the mesh quality metric is very low. The final mesh using 15 intermediate steps is shown in Figure 4.8. It can be seen that the displacement does not spread very far into the domain and the cells close to the moving block are heavily deformed. The mesh quality obtained with the 5 best RBF's is much higher, because the deformation is more evenly spread through the domain.

4.3.2 Test case 2: Rigid body Rotation

For rigid body translations the interpolation is exact and therefore the initial mesh is recovered when the domain returns to its initial form. However, it is not guaranteed that this is the case when rotations are present when the meshes are deformed with

an incremental approach. Therefore we investigate the mesh quality when the block is severely rotated and brought back to its initial position. The nodes on the outer boundary can freely move along this boundary. The initial mesh is the same as for test case 1 (Figure 4.5). First the block is rotated 180° counterclockwise, then 360° clockwise and back to the starting position by rotating it again 180° counterclockwise. The rotation is performed with a variable number of intermediate steps. In Figure 4.9

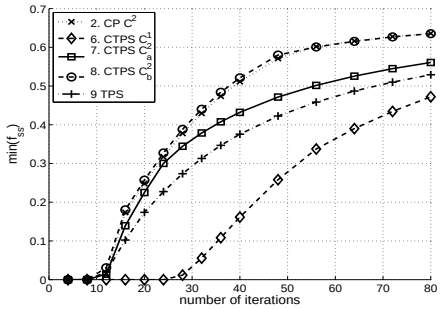


Fig. 4.9: Quality of the worst cell of the mesh for the different RBF's.

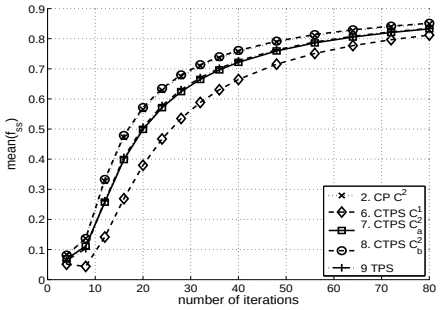


Fig. 4.10: Average quality of the mesh for the different RBF's.

again the minimal value and in Figure 4.10 the average value of the mesh quality metric is shown against the number of intermediate steps. The resulting ranking for the RBF's is shown in Table 4.3.

As can be seen from Figure 4.9, more than 10 intermediate steps are needed with all functions to obtain a positive value of f_{ss} for all cells. Figures 4.11 and 4.12 show

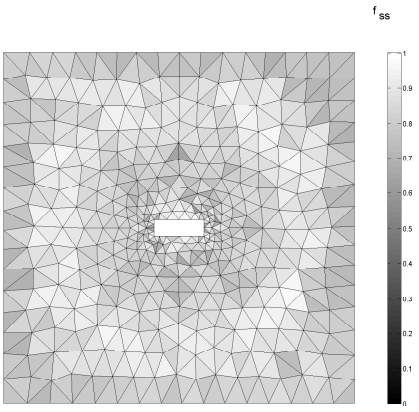


Fig. 4.11: Final mesh using CTPS C_b^2 after 40 intermediate steps.

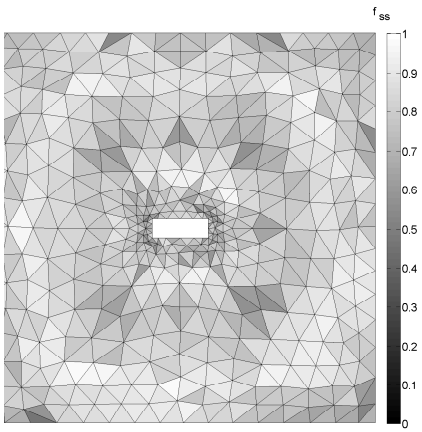


Fig. 4.12: Final mesh using CTPS C^1 after 40 intermediate steps.

the final meshes with 40 intermediate steps using the best RBF, CTPS C_b^2 (nr. 8) and the worst RBF, CTPS C^1 (nr. 6), respectively. Figure 4.12 shows that especially the

cells close to the block and boundary are deformed. With the CTPS C_b^2 function the mesh is also distorted compared to the initial mesh. However, this distortion is rather small considering the very large rotation of the block. The final meshes obtained with TPS, CP C^2 and CTPS C_a^2 are very similar to that of CTPS C_b^2.

4.3.3 Test case 3: Airfoil flap

Until now we only studied rigid body rotation and translation of a block. In the third test case we investigate a more realistic situation of an airfoil-flap configuration. The test case starts with the initial mesh given in Figure 4.13. A close up of the mesh around the gap between the airfoil and flap is shown in Figure 4.14. The flap is rotated 30 degrees in clockwise direction around an axis a tenth of a cord below the trailing edge of the airfoil. In Figure 4.15 the minimal value and in Figure 4.16

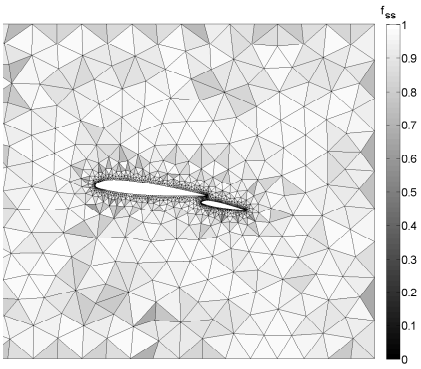


Fig. 4.13: Initial mesh.

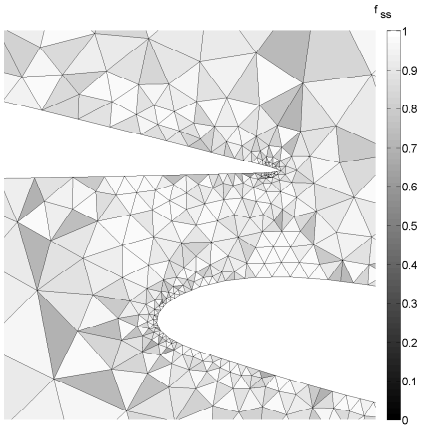


Fig. 4.14: Close up of initial mesh.

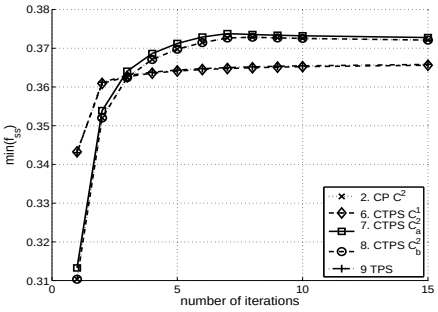


Fig. 4.15: Quality of the worst cell of the mesh for the different RBF's.

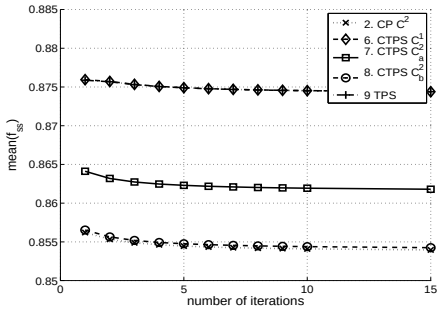


Fig. 4.16: Average quality of the mesh for the different RBF's.

the average value of f_{ss} is shown for the remaining RBF's. It can be seen that all

the RBF's are able to deform the mesh well, and all functions give meshes of similar quality. The ranking is given in Table 4.3.

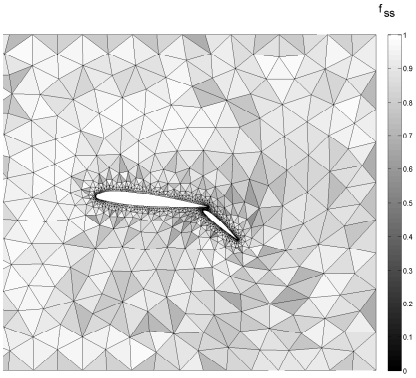


Fig. 4.17: Final mesh using CTPS C_a^2 with 10 intermediate steps.

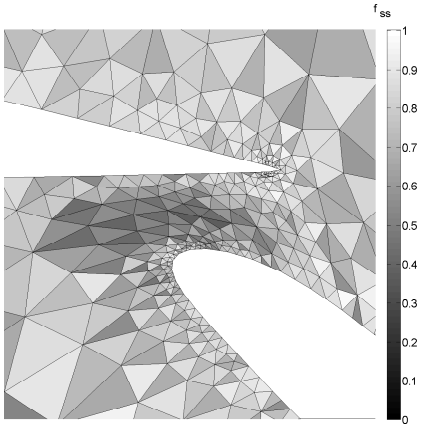


Fig. 4.18: Zoom of final mesh using CTPS C_a^2 with 10 intermediate steps.

The final configuration of the airfoil-flap and the resulting mesh is shown in Figure 4.17 using 10 intermediate steps with the CTPS C_a^2 function. Figure 4.18 displays a close up of this mesh and shows that the mesh quality is still very good in the region where the largest deformation takes place and suitable for flow computations. The final meshes obtained with the other functions look very similar.

4.3.4 Efficiency

The first test case with a rotating and translating block is also used to investigate the efficiency of the five remaining RBF's. The number of boundary nodes, the total number of nodes and the support radius is varied to investigate their effect on the computation time needed for the mesh movement with the different RBF's. At this time we are not interested in the most efficient way to implement the new method, but only in the effect of the different RBF's on the computation time. For the comparison the same implementation of the new method is used, only the RBF is changed. The system is solved directly by a decomposition algorithm. The possibilities for improving the efficiency of the computation are discussed in section 4.7.

In Figure 4.19 the computation time needed for the mesh movement with an increasing number of structure nodes is shown. The number of nodes in the inner domain of the mesh is kept at a constant value of 100. It can be seen that CP C^2 requires the least computation time, followed by TPS. The functions CTPS C^1, CTPS C_a^2 and CTPS C_b^2 require exactly the same computation time, but more than the other two functions. The difference in computation time can be explained by the difficulty of the evaluation of the RBF. CP C^2 only involves the evaluation of a fifth order polynomial, whereas in the other functions an evaluation of a log-function has to be

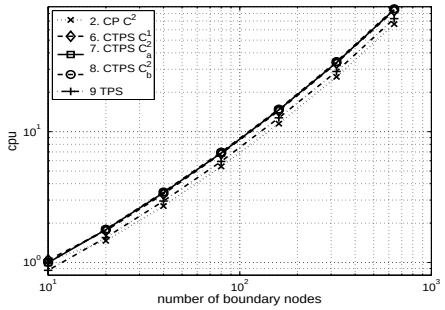


Fig. 4.19: Influence of the number of boundary nodes on CPU time.

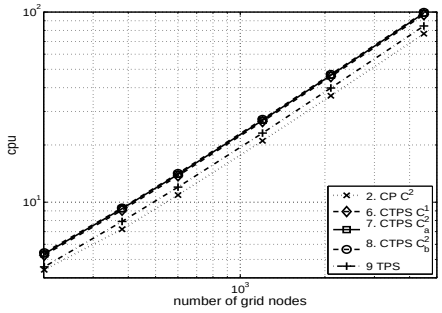


Fig. 4.20: Influence of the total number of nodes on CPU time.

carried out. On top of that, CTPS C^1, CTPS C^2_a and CTPS C^2_b, require the evaluation of a higher order polynomial and are therefore more computational expensive than TPS. The difference in computation time between CP C^2 and CTPS C^2_b is about 20 percent.

Figure 4.20 shows again the computation time needed for the mesh movement but this time the number of internal nodes is varied. The relative results between the RBF's are the same as when only the number of boundary nodes is varied. The difference between the two figures is caused by the fact that the number of boundary nodes determines the size of the system to be solved, whereas increasing the number of internal nodes only results in more function evaluations. Therefore the computation time increases much faster with an increasing number of boundary nodes than with an increasing number of internal nodes. Note that scaling of the y-axis is different for Figures 4.19 and 4.20.

Finally the effect of the support radius r on the computation time is investigated. For radial basis functions with compact support the matrix system to be solved becomes less dense, when the support radius is decreased. This means that less function evaluations are needed and the system can be solved more efficiently. In Figure 4.21 the computation time is plotted against the support radius. Since TPS is a global radial basis function the CPU-time does not change with r. The CPU time needed by CP C^2 only mildly increases with r, whereas the computation time needed for mesh movement with the remaining three functions CTPS C^1, CTPS C^2_a and CTPS C^2_b, is very sensitive to the support radius. Only for very small values of the support radius they are approximately as fast as CP C^2. However, for these small values of r the quality of the resulting meshes is very poor as is shown in Figure 4.22 where the minimum value of the mesh quality is plotted against the support radius. Especially for the CTPS C^1 function a very high support radius is needed to avoid degenerate cells. Overall it can be concluded that CP C^2 requires the least computation time, followed by TPS.

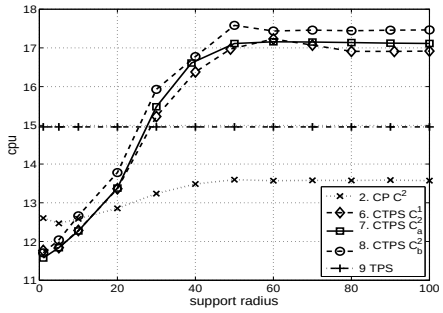


Fig. 4.21: Influence of support radius on CPU time.

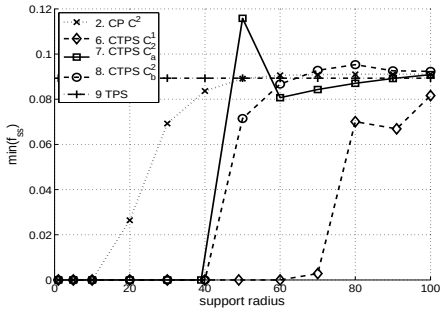


Fig. 4.22: Influence of support radius on mesh quality.

4.3.5 Flow around airfoil

Until now only the mesh quality of the deformed meshes has been investigated without solving a single physical problem. In this section more realistic flow results on a distorted mesh are presented. We perform two calculations of viscid flow around a NACA-0012 airfoil. The airfoil is rotated 8° and moved 5 cords downstream and 2 cords upwards. A steady state solution of Mach 0.3 with Re = 1000 is computed around the airfoil. In the first computation a new unstructured hexahedral mesh is generated around the moved airfoil. A close-up of the mesh together with the pressure field is shown in Figure 4.23. In the second computation the mesh is deformed in one

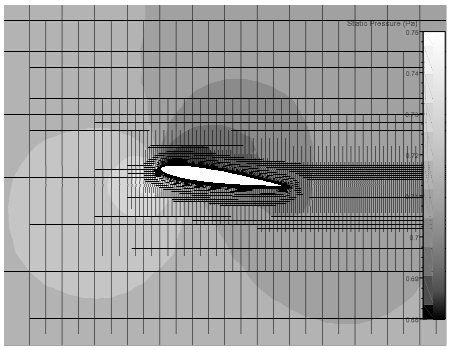


Fig. 4.23: Pressure field around wing on a new generated mesh.

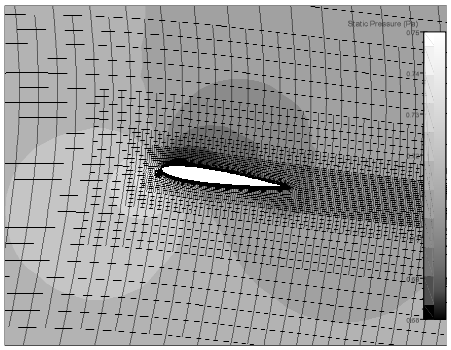


Fig. 4.24: Pressure field around wing on a deformed mesh.

step with the thin plate spline from its original position. The result is shown in Figure 4.24. The resulting pressure distributions over the wing are identical as can be seen in Figure 4.25. The difference in lift computed on the two different meshes is only 0.8%.

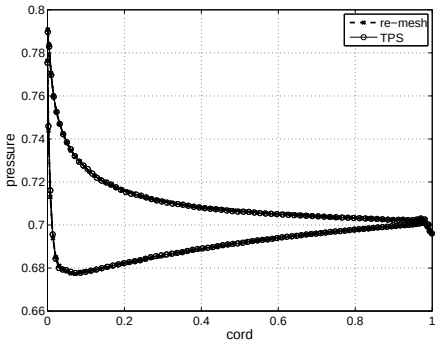


Fig. 4.25: Pressure distribution over the wing.

4.4 Including rotations

Sometimes rotational information is also known in the boundary points. In [4] a method is presented to take this rotational information into account for the data transfer with radial basis functions between non-matching flow and structure meshes. In this section we investigate if incorporating rotational information is also beneficial for our new mesh movement strategy.

For small angles, rotations can be recovered from derivatives of translations through

$$\theta = \tfrac{1}{2}\left(\partial_x \mathbf{d}_{\mathrm{b}_y} - \partial_y \mathbf{d}_{\mathrm{b}_x}\right) = \tfrac{1}{2}\nabla \times \mathbf{d}_{\mathrm{b}}, \tag{4.7}$$

for 2D mesh movement or

$$\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -\partial_z & \partial_y \\ \partial_z & 0 & -\partial_x \\ -\partial_y & \partial_x & 0 \end{bmatrix}\begin{bmatrix} \mathbf{d}_{\mathrm{b}_x} \\ \mathbf{d}_{\mathrm{b}_y} \\ \mathbf{d}_{\mathrm{b}_z} \end{bmatrix} = \tfrac{1}{2}\nabla \times \mathbf{d}_{\mathrm{b}} \tag{4.8}$$

in three dimensions.

When only translations are taken into account, the displacement can be interpolated separately for each spatial direction, as shown in the previous sections. With the inclusion of rotational information the displacements for the spatial directions become coupled. According to [4], the interpolation functions for the spatial directions in 2D then become:

$$s_1(\mathbf{x}) = \sum_{j=1}^{n_{\mathrm{b}}} \left[\alpha_{2j-1}\phi_{*j} + \alpha_{2(N+j)}\tfrac{\partial_2}{\partial y}\phi_{*j}\right] + p_1(\mathbf{x}) \tag{4.9a}$$

$$s_2(\mathbf{x}) = \sum_{j=1}^{n_{\mathrm{b}}} \left[\alpha_{2j}\phi_{*j} + \alpha_{2(N+j)}\tfrac{\partial_2}{\partial x}\phi_{*j}\right] + p_2(\mathbf{x}), \tag{4.9b}$$

where $\phi_{*j} = \phi(||\mathbf{x} - \mathbf{x}_j||)$, and in 3D:

$$s_1(\mathbf{x}) = \sum_{j=1}^{n_\mathrm{b}} \left[\alpha_{3j-2}\phi_{*j} + \alpha_{3(N+j)-1}\tfrac{\partial_2}{\partial z}\phi_{*j} - \alpha_{3(N+j)}\tfrac{\partial_2}{\partial y}\phi_{*j}\right] + p_1(\mathbf{x}) \tag{4.10a}$$

$$s_2(\mathbf{x}) = \sum_{j=1}^{n_\mathrm{b}} \left[\alpha_{3j-1}\phi_{*j} - \alpha_{3(N+j)-2}\tfrac{\partial_2}{\partial z}\phi_{*j} + \alpha_{3(N+j)}\tfrac{\partial_2}{\partial x}\phi_{*j}\right] + p_2(\mathbf{x}) \tag{4.10b}$$

$$s_3(\mathbf{x}) = \sum_{j=1}^{n_\mathrm{b}} \left[\alpha_{3j}\phi_{*j} + \alpha_{3(N+j)-2}\tfrac{\partial_2}{\partial y}\phi_{*j} - \alpha_{3(N+j)-1}\tfrac{\partial_2}{\partial x}\phi_{*j}\right] + p_3(\mathbf{x}). \tag{4.10c}$$

The linear polynomials are defined as $p_i(\mathbf{x}) = \beta_i + \beta_{i+2}x + \beta_{i+4}y$ in 2D and $p_i(\mathbf{x}) = \beta_i + \beta_{i+3}x + \beta_{i+6}y + \beta_{i+9}z$ in 3D.

When both translation and rotation are taken into account the interpolation condition (4.2) is extended to

$$\mathbf{s}(\mathbf{x}_{\mathrm{b}_i}) = \mathbf{d}_{\mathrm{b}_i}, \tag{4.11a}$$

$$(\nabla \times \mathbf{s})(\mathbf{x}_{\mathrm{b}_i}) = 2\theta(\mathbf{x}_{\mathrm{b}_i}), \tag{4.11b}$$

for $i = 1 \ldots n_b$, where $\mathbf{s}(\mathbf{x}) = (s_1(\mathbf{x}), ..., s_d(\mathbf{x}))^T$. The first condition ensures that the translational information is exactly recovered in the centres and the second condition ensures that the interpolation function incorporates the rotational information. Together with the additional orthogonality requirements for the polynomial:

$$\sum_{j=1}^{3n_\mathrm{b}} \alpha_j p_i(\mathbf{x}_{\mathrm{b}_j}) = 0, \qquad \sum_{j=1}^{3n_\mathrm{b}} \alpha_j \beta_{i+2} = 0, \qquad \sum_{j=1}^{3n_\mathrm{b}} \alpha_j \beta_{i+4} = 0, \qquad i = \{1, 2\}, \tag{4.12}$$

in 2D, or

$$\sum_{j=1}^{6n_\mathrm{b}} \alpha_j p_i(\mathbf{x}_{\mathrm{b}_j}) = 0, \qquad \sum_{j=1}^{6n_\mathrm{b}} \alpha_j (\beta_{i+9} - \beta_{i+6}) = 0, \qquad \sum_{j=1}^{6n_\mathrm{b}} \alpha_j (\beta_{i+3} - \beta_{i+9}) = 0,$$
$$\sum_{j=1}^{6n_\mathrm{b}} \alpha_j (\beta_{i+6} - \beta_{i+3}) = 0, \qquad i = \{1, 2, 3\}, \tag{4.13}$$

in 3D, this results in a coupled symmetric system of linear equations of the form

$$\begin{bmatrix} A_{\phi,\mathrm{b}} & P_\mathrm{b} \\ P_\mathrm{b}^T & \varnothing \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ 0 \end{bmatrix}. \tag{4.14}$$

The entries of the $(3(d+1)n_\mathrm{b} \times 6(d+1)$ matrix P_b and the $3(d-1)n_\mathrm{b}$ vector $\mathbf{r}$ are given in more detail in Appendix A, where d is the dimension of the problem and n_b the number of points on the boundary. The $(3(d-1)n_\mathrm{b} \times 3(d-1)n_\mathrm{b})$ symmetrical block matrix $A_{\phi,\mathrm{b}}$ is given by

$$A_{\phi,\mathrm{b}} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{4.15}$$

The exact setup of the $(dn_\mathrm{b} \times dn_\mathrm{b})$ symmetrical matrix A_{11}, the $(2d-3)n_\mathrm{b} \times (2d-3)n_\mathrm{b}$ symmetrical matrix A_{22} and the $(dn_\mathrm{b} \times (2d-3)n_\mathrm{b}$ matrix $A_{12} = A_{21}^T$ is also given in Appendix A.

The dimension of the problem to be solved now grows with $3n_\mathrm{b}$ for 2D and $6n_\mathrm{b}$ for 3D rather than n_b, which was the case for the problem without rotations. Therefore the recovery of rotations is expensive and should only be used with small models. However, especially for small models (for example reduced order models) it is crucial to incorporate as much information as possible, which means that the additional recovery of rotations may yield a better solution.

4.4.1 Derivatives of RBFs

Taking into account rotations is only possible with C^2 continuous RBFs because the second derivative of the basis function is needed in matrix A_{22}. This means that we can not use TPS, for example, because it has a discontinuity in the second derivative around zero.

The general form of an RBF is $\phi(a(\mathbf{x}))$ with $a(\mathbf{x}) = ||\mathbf{x}||/r$, and r a given constant. The first and second derivative with respect to $\xi \in \{\mathbf{x}^1, \ldots, \mathbf{x}^d\}$, are then given by

$$\frac{\partial \phi}{\partial \xi} = \frac{\partial \phi}{\partial a} \cdot \frac{\partial a}{\partial \xi}, \tag{4.16}$$

$$\frac{\partial^2 \phi}{\partial \xi^2} = \frac{\partial}{\partial \xi}\left(\frac{\partial \phi}{\partial a} \cdot \frac{\partial a}{\partial \xi}\right) = \frac{\partial^2 \phi}{\partial a^2} \cdot \left(\frac{\partial a}{\partial \xi}\right)^2 + \frac{\partial \phi}{\partial a} \cdot \frac{\partial^2 a}{\partial \xi^2}, \tag{4.17}$$

where

$$\frac{\partial a}{\partial \xi} = \frac{1}{r} \cdot \frac{\xi}{||\mathbf{x}||} = \frac{\xi}{a r^2}, \tag{4.18}$$

$$\frac{\partial^2 a}{\partial \xi^2} = \frac{1}{r} \cdot \frac{||\mathbf{x}||^2 - \xi^2}{||\mathbf{x}||^3} = \frac{a^2 r^2 - \xi^2}{a^3 r^4}. \tag{4.19}$$

With this information available rotations can be included.

4.4.2 Results

We only tested incorporation of rotations with the CP C^2 function, because it performed best in earlier test cases. The TPS, which also gave very good results, is not considered, because it is only C^1 continuous and we need an evaluation of a second derivative. We start with a square domain meshed with all equal triangles. The point in the middle is rotated 60° in clockwise direction. This rotation is performed in 1 and 10 intermediate steps. The results are shown in Figures 4.26 and 4.27 respectively. It can be seen that the rotation is interpolated through the whole domain and the smaller the rotation per step (with 10 intermediate steps), the better the interpolation, especially close to the rotated point. This can be explained by the fact that a small angle approximation is used in (4.7) to incorporate the rotational information.

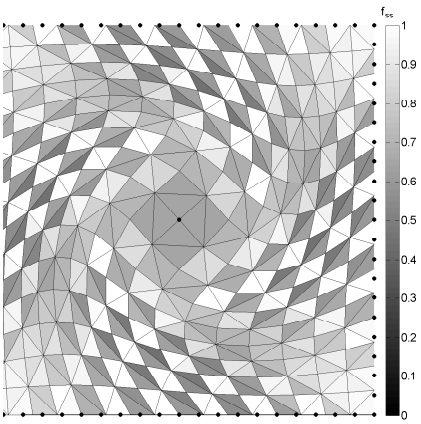


Fig. 4.26: Rotation of the centre point in 1 step.

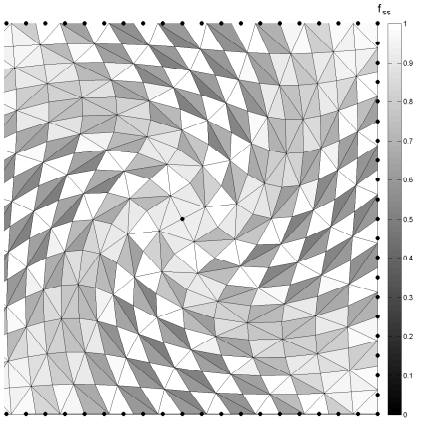


Fig. 4.27: Rotation of the centre point in 10 steps.

When only displacements are taken into account the internal grid points do not move at all.

When there is translation information available, such as in the case when a block is rotated, the advantage of including rotations is not very clear as is shown in the following test case. We start with the same initial mesh as for test case 1 (Figure 4.5). Now the block in the middle is rotated 60° in clockwise direction in one step. The results without rotational information included are shown in Figure 4.28 and with rotational information in Figure 4.29. There is hardly any difference visible between the two figures and also the difference in the mesh quality metrics is negligible. When

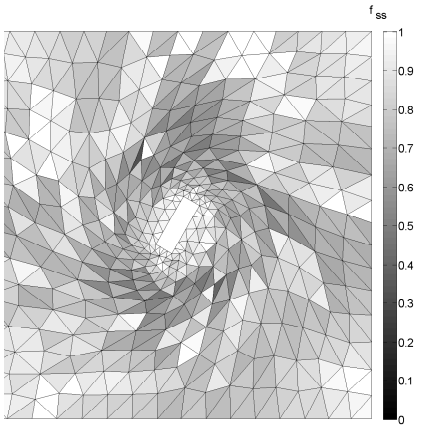


Fig. 4.28: Rotation of the block in 1 step without including rotations.

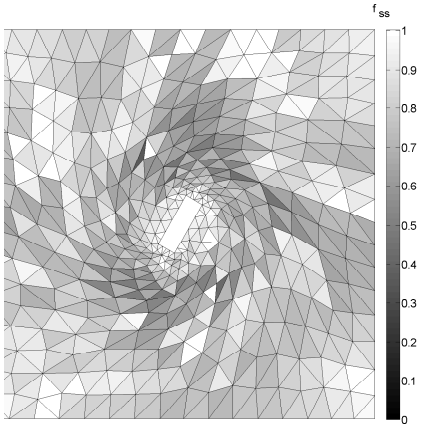


Fig. 4.29: Rotation of the block in 1 step with rotations included.

the number of intermediate steps is increased to 10, as is shown in Figures 4.30 without and in Figure 4.31 with rotations, the difference is still very small. And again

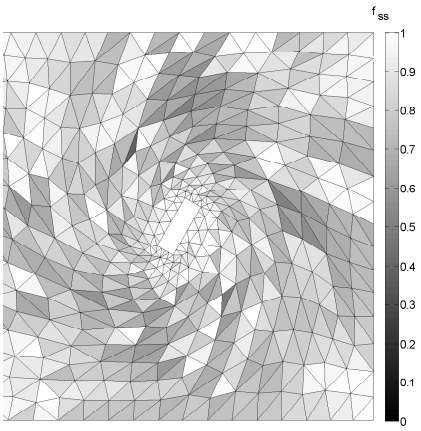


Fig. 4.30: Rotation of the block in 10 steps without including rotations.

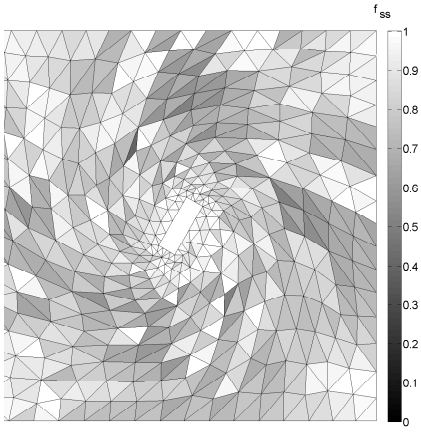


Fig. 4.31: Rotation of the block in 10 steps with rotations included.

the difference in the mesh quality metrics is negligible.

The small difference can be explained by the fact that when the rotation becomes smaller, the small-angle approximation (4.7) is more accurate, but also only taking into account displacements gives then better results (see also section 4.3). Therefore we can conclude that when there is displacement information available, the additional recovery of rotations does not improve the solution. Especially when we take into account the much higher computational costs. Even for this simple 2D problem, the computational costs are approximately 3 times higher when rotations are included. This means that in practical applications only the translation has to be taken into account.

4.5 Importance of smooth mesh deformation for higher order time-integration

In order to investigate the effect of the mesh deformation algorithm on the temporal accuracy of higher order time integration schemes we consider the one-dimensional piston problem [120]. The flow equations are solved in a two-dimensional domain on an "imperfect" mesh which has cells that are not perfectly orthogonal, see Figure 4.32. The nodes on the upper and lower boundary of the piston can freely move along this boundary. We use the fourth order IMEX scheme [120] for the partitioned time integration.

The mesh deformation technique based on RBF interpolation is compared to a technique which solves the Laplace equation to create a displacement field [76]. After displacing the flow vertices with this second method, the mesh is optimized with a

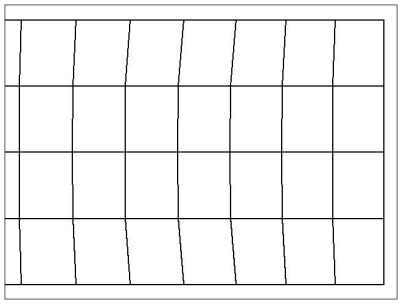


Fig. 4.32: Part of the "imperfect" piston mesh.

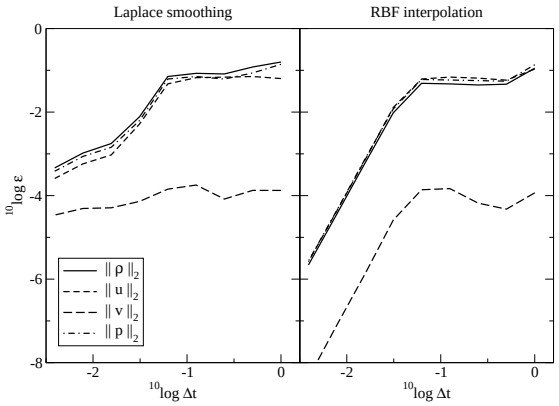
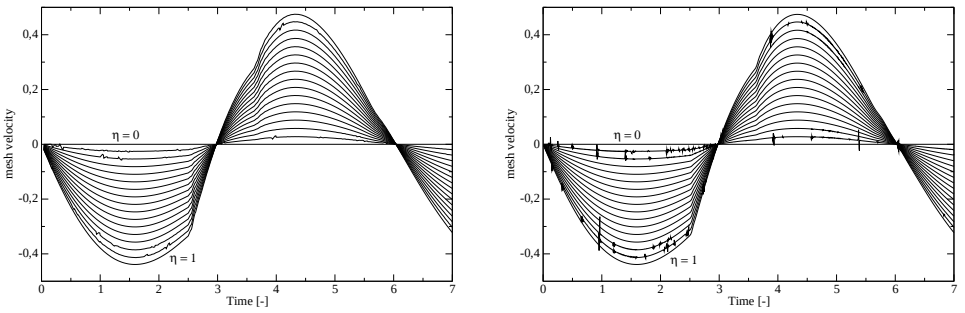


Fig. 4.33: Convergence for the L_2-norm of the density, pressure and u- and v-velocity components for the piston problem with Laplace smoothing and radial basis function interpolation.

smoothing procedure [76] which is necessary to avoid degenerate cells. In Fig. 4.33 the L_2-norms of the fluid density ρ, pressure p, velocity in x-direction u and y-direction v for the different meshes and mesh deformation schemes are shown.

The results obtained with the Laplace smoothing are not satisfactory. The fourth order of the scheme is not observed and although the test problem is essentially one-dimensional, the v-velocity is not zero due to the imperfect mesh. For the large time steps this error in the v-velocity is only small compared to the other errors. However, since the convergence for the v-velocity is clearly not fourth order, its influence becomes more apparent at smaller time steps. RBF interpolation for the imperfect mesh has the same nonzero v-velocity for the large time steps. This time, however, the perturbation does converge with fourth order accuracy and therefore its influence on the solution remains negligible. Therefore we can conclude that the RBF interpolation does not aggravate imperfections in the flow mesh sot that design order of the IMEX scheme is recovered.

In order to explain the bad convergence with the Laplace smoothing we study the mesh face velocities for the cell faces which are displayed in Fig. 4.34. These mesh face velocities are computed according to the Discrete Geometric Conservation Law (DGCL) as derived in [123]. The line with $\eta = 0$ corresponds to a stationary cell face and the line with $\eta = 1$ to a cell face on the piston. It shows that the Laplace smoothing introduces irregularities (wiggles) in the mesh face velocities which are worse for small Δt. It is not clear what causes these irregularities, but in our opinion they are generated during the smoothing process. The mesh face velocity does not converge to a consistent solution so the design order of the IMEX scheme can not be expected. Due to the high accuracy and regularity of the displacement field obtained with RBF interpolation the RBF mesh deformation algorithm does not exhibit these convergence problems.



(a) $\Delta t = 1/16$.

(b) $\Delta t = 1/64$.

Fig. 4.34: Mesh face velocities with Laplace smoothing. The line with $\eta = 0$ corresponds to a stationary cell face and the line with $\eta = 1$ to a cell face on the piston.

When higher order time integration methods are used it is possible to take larger time steps. This means that the mesh deformation within a time step becomes larger. It is therefore needed to use a mesh deformation method that is capable of dealing with large deformations, otherwise the time step is limited by the mesh deformation method which destroys the efficiency gain obtained with the higher order time integration method. As the RBF mesh deformation algorithm can deal with large rotations and translation it very well suited for the use with higher order time integration methods.

4.6 3D mesh deformation

To show the capability of the new method for 3D applications we consider in this section a test case where a block is rotated and translated in a 3D domain and a real FSI computation considering the AGARD 445.6 test case.

4.6.1 Rotation and translation of 3D block

We consider a test case in a 3D domain. A cross-section showing the initial mesh and location of the block is given in Figure 4.35. The block is translated 2.5 times the thickness of the block and rotated 15 degrees in all three directions. A cross-section of the final mesh and location of the block is shown in Figure 4.36. Visually it is quite hard to judge the quality of the mesh and therefore the values of the mesh quality metric in the cross-sections is displayed in Figures 4.37 and 4.38 for the CP C^2 and TPS function, respectively. It can be seen that the mesh quality is everywhere larger than 0.5 and therefore the meshes are suitable for computational analysis. The main difference between the two figures is that with CP C^2 the mesh quality close to the moving structure remains a little higher than with TPS.

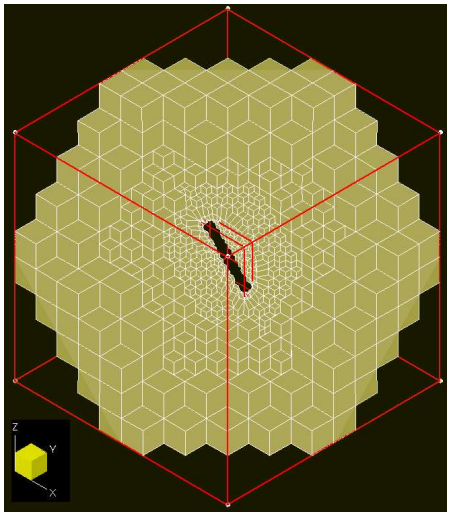


Fig. 4.35: Cross-section of initial mesh.

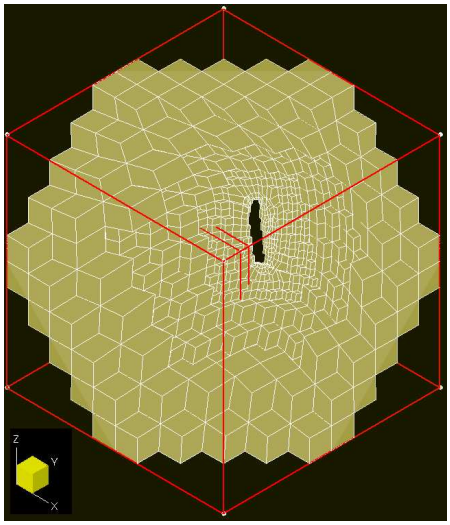


Fig. 4.36: Cross-section of final mesh.

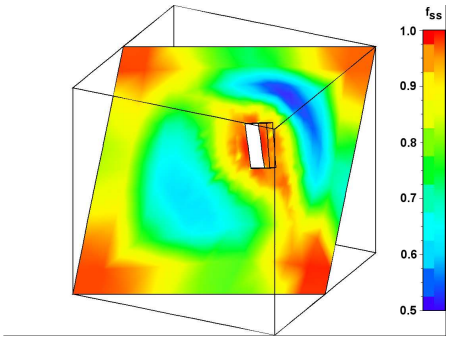


Fig. 4.37: Mesh quality in cross-section using CP C^2.

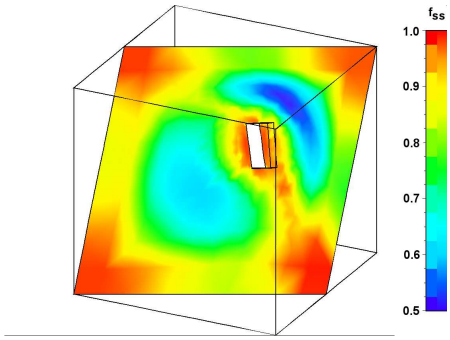
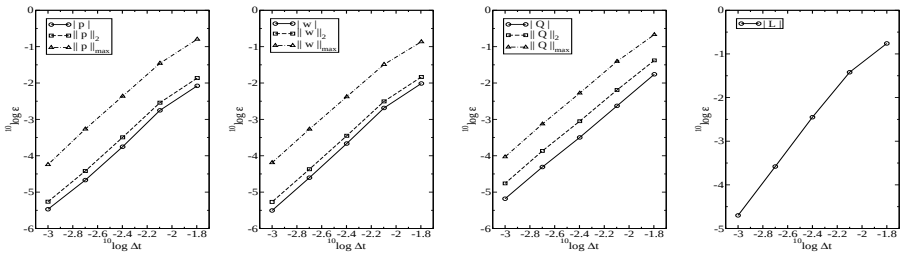


Fig. 4.38: Mesh quality in cross-section using TPS.

4.6.2 Flutter of the AGARD 445.6 wing

To demonstrate the practical applicability for real-world three-dimensional fluid-structure interaction the AGARD 445.6 test case [118, 123] is used. For the investigation into the performance of the third order IMEX scheme [120] with the RBF mesh deformation a time step convergence study is performed. The solution obtained with $\Delta t = 0.001$ is taken as the temporally exact solution. In Fig. 4.39 the results for the pressure field, velocity in vertical direction w, structural displacement and lift at the end of the simulation are shown. The figures show a clear third order convergence for all the properties in all the norms, which shows that the combination of the third order partitioned IMEX scheme with RBF mesh deformation retains the order of the



(a) Pressure. (b) w-velocity. (c) Displacement. (d) Lift.

Fig. 4.39: Time step convergence for the third order IMEX scheme.

time integration scheme without the necessity to sub-iterate.

The displacements generated in the AGARD 445.6 test case are rather small and do not place a large demand on the mesh deformation method. To investigate the performance of the mesh deformation method for large displacements we also enforced a large deformation on a 3D beam. The resulting mesh on the beam and the mirror plane is shown in Figure 4.40.

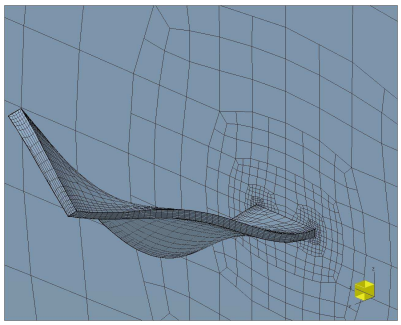


Fig. 4.40: Mesh obtained with RBF interpolation around a heavily deformed 3D beam.

It can be seen that the mesh is still regular and also the mesh quality metrics indicate a high mesh quality, such that accurate flow calculations are possible.

4.7 Improving efficiency

The computational costs of the new mesh deformation method increase very fast with an increasing number of boundary and internal points when direct methods are used. In this section we investigate different techniques to improve the overall efficiency of the computation.

The two main computational challenges of interpolation with radial basis function interpolation are:

(A) Fitting: given the points $\mathbf{x}_{\mathrm{b}_j}$ and values $\mathbf{d}_{\mathrm{b}_j}$ determine the coefficients γ_j and $\boldsymbol{\beta}$ (solve system (4.4)). Because Φ_{bb} is generally a dense $n_{\mathrm{b}} \times n_{\mathrm{b}}$ symmetric matrix, standard direct solvers require $\mathcal{O}(n_{\mathrm{b}}^3)$ operations.

(B) Evaluation: given the points $\mathbf{x}_{\mathrm{b}_j}$ and coefficients γ_j and $\boldsymbol{\beta}$, evaluate (4.1) at all n_{in} internal grid points. The cost of direct summation is $\mathcal{O}(n_{\mathrm{b}} n_{\mathrm{in}})$.

For large 2D and 3D meshes both (A) and (B) can become rather computationally expensive when direct methods are used. To indicate whether the computational costs of (A) or (B) are the bottleneck in the mesh deformation we consider a square domain in 2D and 3D with an equidistant Cartesian grid, with N nodes on each edge of the square. The number of internal and boundary nodes for this configuration are given in Table 4.4 for both the 2D and 3D case. The order of magnitude of the computational costs for fitting and evaluation with a direct approach are given in the last two columns of this table. It can be seen that for the 2D mesh the computational costs for fitting

Table 4.4: Number of internal and boundary nodes and the computational costs for fitting and evaluation for a 2D and 3D square domain with equidistant Cartesian grid.

	# internal nodes	# boundary nodes	Fitting (A)	Evaluation (B)
2D	N^2	$4N$	$64N^3$	$4N^3$
3D	N^3	$6N^2$	$216N^6$	$6N^5$

and evaluation both scale with N^3. This means that (A) and (B) are equally expensive. For the 3D mesh the computational costs for fitting are a factor N higher than for the evaluation and therefore (A) is the bottleneck in the computation.

Fitting

It is not necessary to solve (A) and (B) exactly, because the movement of the internal grid points can be arbitrary, as long as it results in a good quality mesh. This means that an iterative method with a weak convergence criterion can be used for (A). A few Krylov-subspace iterative methods have been developed especially for the fitting of radial basis functions. These methods are based on preconditioning the matrices with approximate cardinal functions [7] or using a Lagrange form of the interpolation function for the preconditioning [48, 49, 50]. The iterative algorithms require the ability to efficiently multiply Φ_{bb} with a vector. Thus, improving the efficiency of (B) is also important to improve the efficiency of (A). For our mesh deformation application the interpolation problem is most of the times very ill conditioned. For example, when the Thin Plate Spline is used the condition number for the simple test problems in sections 4.3.1 and 4.3.2 is $\mathcal{O}(10^{10})$. This is due to the fact that the centers are clustered at the boundaries of the flow domain and therefore there is a

large difference in distances between points. For these ill conditioned problems the iterative methods are still quite expensive. In this case a direct solver working in parallel might be a better choice, for example SuperLU [40, 79] (mainly developed for sparse systems) or ScaLAPACK [20] (a parallel version of the linear algebra package LAPACK).

Another way to improve the efficiency of the computation is to reduce the number of centres involved in computing and evaluating the interpolation function. This can be obtained by removing the centres on the fixed outer boundary and multiplying the interpolation function with a cut-off function instead [68]. This cut-off function is equal to one close to the moving boundary and decays to zero at the outer boundaries. Furthermore, when there is a clustering of centres at certain locations it is possible to only take into account a selection of these centres that are positioned at a minimal distance d_{ref} from each other. This both reduces the number of centres and improves the condition number of the problem [68]. This approach is further improved in [95, 96, 97] by choosing the involved centres in an optimal way such that geometrical accuracy is preserved.

Evaluation

For large meshes the number of internal grid points is very high, especially in 3D. In these cases the evaluation of the interpolation function (B) is very time consuming. The evaluation part (B) can easily be performed in parallel as the displacement in each point can be computed independently of all other points once the location of the centers and the coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ computed in (A) is known. In literature also various fast evaluation algorithms can be found and three of them are listed below:

- **Fast Multipole Method** (FMM) [8, 9, 10, 12]: The evaluation complexity is $\mathcal{O}((n_{\text{b}} + n_{\text{in}})(\ln(n_{\text{b}}))(\ln(1/\delta))^{d+1})$, where δ is the desired accuracy and d the dimension of the problem. This method is inspired by fast methods to perform many-body computations. The total domain is subsequently subdivided into child panels, such that a tree data structure is formed. The algorithm uses the data structure to separate contributions of the interpolation function into 'near field' and 'far field' components. For all the fine level panels a Laurent series expansion is formed and translated to all higher (less refined levels). This Laurent series is then used to form local Taylor series expansions of the 'far field' for each panel at the fine level. To evaluate the interpolation function in a point, the fine level panel containing this point has to be located. The evaluation is then performed by direct calculation of the 'near field' and using the Taylor series expansion to approximate the 'far field'.

  While this method is successfully applied in applications [18, 19, 35], it is rather complicated to program, especially in higher dimensions, because of the complex hierarchical structure and tree codes required to decompose ϕ into 'near field' and 'far field' components. Therefore this method is not further investigated for the new mesh movement strategy in the first instance.

- **Fast Gauss Transform** (FGT) [98]: The algorithm's cost is $\mathcal{O}(m+n)$, where the constant increases with the desired accuracy and dimension. The method exploits

the conditionally positive definite properties of certain radial basis functions to replace (4.1) with the Gaussian basis function using the Fast Gauss Transform [58], followed by an appropriate Gaussian quadrature rule to translate the Gaussian result back to the original basis function. The FGT algorithm can not be used for the compactly supported Wendland functions, because no Gaussian basis functions are defined for them (yet).

For the TPS a complicated quadrature rule is needed, which turns out to reduce the efficiency of the method considerably. Also the computational costs increase when the sum of the absolute values of the coefficients, $\sum_j |\gamma_j|$, increases. Unfortunately this sum is very large in our mesh deformation problems. In a number of 2D tests in Matlab and C++ the use of the FGT algorithm did not decrease the computational costs for the mesh deformation method. The conclusion is therefore that FGT is not suited to improve the efficiency of the new mesh movement strategy.

- **Fast Multilevel Evaluation** (FMLE) [81, 82]: The computational complexity is $\mathcal{O}((n_\mathrm{b} + n_\mathrm{in}) \ln(1/\delta)^d)$. The method interpolates the coefficients $\boldsymbol{\alpha}$ to a coarser Cartesian mesh using a pth order Lagrange interpolation. A coarse level summation of (4.1) is performed to obtain an interpolation function on the coarse mesh. This coarse interpolation function is then interpolated back to the points in which the information is needed. The order p of the Lagrange interpolation and the grid size H of the coarse mesh depend on the desired accuracy δ, the dimension of the problem d and the 'shape parameter' a and are different for various radial basis function. The method can be applied to smooth radial basis functions and indeed gives an efficiency increase in these cases. For the evaluation with the multiquadric biharmonic (function 10 in Table 4.2, with $a = 1.8$) of 100.000 points uniformly distributed in a square 3D domain the fast multilevel evaluation is 35 times faster for an accuracy of 10^{-13} and 2000 times faster for an accuracy of 10^{-3}, compared to direct evaluation.

  For the TPS a hierarchy of coarse grids is needed because the function has a singularity around $r = 0$ and is therefore not smooth enough. The TPS is decomposed into a smooth polynomial softened part and a local part. The smooth part can be evaluated with the fast multilevel evaluation technique described above, and the local part is evaluated directly, since its is compactly supported. Results for the 1D case are published in [81] and show promising results. We extended the method for TPS to higher dimensions, using the coefficients of the 1D case, but encountered that working with a hierarchy of coarse grid levels in higher dimensions decreases the computational efficiency to a large extent, especially because the density of the original centres (the boundary points) is highly non-uniform in our mesh-moving problems. An Adaptive Mesh Refinement (AMR) strategy might overcome this problem, as suggested in [81]. The FMLE algorithm is not suitable (yet) for the compactly supported Wendland functions. The Wendland functions have a singularity around $r = 1$ instead of $r = 0$. This means that the decomposition in a smooth and local part, in the exact same way as for the TPS, is not possible.

Overall it can be concluded that improving the efficiency of the mesh moving based on radial basis function is not straightforward and needs further investigation. First a fast evaluation technique has to be found, because fast evaluation is also needed in iterative solution methods. It is worthwhile to extend the FMLE method with an Adaptive Mesh Refinement strategy to increase the efficiency for the highly non-uniform located centres encountered in our mesh deformation problems. If this gives no satisfactory results, using the FMM method is the second option. This method, although rather complicated to program, shows promising result, but until now only results with uniformly distributed centres are published. It remains therefore to be seen if these efficiency gains are also obtained in our mesh deformation problems.

Once a suitable fast evaluation technique has been found, the next step is to further investigate the use of an iterative solution technique for (A). For the preconditioning with approximate cardinal functions only results with uniformly distributed centres are published [7]. When a Lagrange form of the interpolation function is used for the preconditioning, an efficiency increase is obtained for various positions of the centres [49]. However, when the TPS is used and the distance between the centres is locally very small, the convergence rate in the iterative procedure becomes very slow. These locally small distances between centres are typically the case in our mesh deformation problems and therefore it is suggested to first investigate the iterative technique based on approximate cardinal functions.

4.8 Conclusions

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

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

In a first comparison the RBF-method produces meshes of higher quality than the popular semi-torsional spring analogy for very large deformations. The quality of the meshes after deformation is high enough to perform accurate flow calculations. If there is translation information available, the additional recovery of rotations does not improve the solution, but it increases the computation time dramatically.

It is shown that working with higher quality meshes can increase the efficiency of the computation. This is due to the fact that the new method preserves the design

order of higher order time integration methods, due to its smooth deformation in time. Also the larger deformations due to a larger time step, which are possible when using higher order time integration methods, can be handled with the RBF mesh deformation algorithm. The capability of the method to deform 3D meshes is shown by rotating and translating a 3D block and performing a case study on flutter of the AGARD 445.6 wing.

The two main computational challenges of mesh moving with radial basis function interpolation are: (A) solving a dense, ill-conditioned matrix system, and (B) evaluating the interpolation function in all internal grid points. The computational costs of both (A) and (B) increase fast with an increasing number of boundary and internal points when direct methods are used. For 2D problems the computational costs of both (A) and (B) are of the same order of magnitude, while for 3D problems the costs for (A) are approximately a factor $\sqrt{n_\mathrm{b}}$ higher than for (B), with n_b the number of boundary points. It is not necessary to solve (A) and (B) exactly, because the movement of the internal grid points can be arbitrary, as long as it results in a good quality mesh. This means that an iterative method with a weak convergence criterion can be used for (A). However, iterative techniques require also the evaluation of the interpolation function in the centres during each iteration. Thus, improving the efficiency of (B) is also important to improve the efficiency of (A). When iterative methods are still too expensive for the very ill conditioned problems encountered in our mesh deformation problems, a direct solver working in parallel might be a better choice

The evaluation of the interpolation function in the internal grid points (B) can easily be parallelized. In literature also various fast evaluation algorithms can be found. The use of the Fast Gauss Transform did not decrease the computational costs for the mesh deformation method in a number of 2D tests. Therefore we extended the Fast Multilevel Evaluation (FMLE) method for TPS to higher dimensions, using the coefficients of the 1D case. Here we encountered that working with a hierarchy of coarse grid levels in higher dimensions decreases the computational efficiency to a large extent, especially because the density of the original centres (the boundary points) is highly non-uniform in our mesh-moving problems. It would be worthwhile to extend the FMLE method with an Adaptive Mesh Refinement strategy to try to increase the efficiency. If this gives no satisfactory results, using the Fast Multipole Method method is the next option. This method, although rather complicated to program, shows promising result, but until now only results with uniformly distributed centres are published. It remains therefore to be seen if these efficiency gains are also obtained in our mesh deformation problems.

Chapter 5

Fluid-Structure interaction between laminar flow and a deformable flap

In this chapter a real two-dimensional FSI computation is performed in which the findings of the three previous chapters are combined. To couple the flow and structure solver, which are separate solver programs, the flexible coupling shell Flecs, introduced in Chapter 3, is used. The transfer of data over the non-matching interface is performed with four of the transfer algorithms which were investigated in Chapter 2. These four transfer algorithms are implemented within the separate Flecs server program that handles the data transfer between the two solvers. During the unsteady computation, the flow mesh is repeatedly adapted to the deforming flow domain with the new mesh movement method based on radial basis function interpolation proposed in Chapter 4. For the FSI problem we use the numerical benchmark problem of 2D flow around a cylinder with deformable flap described in [110]. In contrast to the quasi-1D FSI problem in section 2.4 both the flow and structure are two dimensional and the solution is time-dependent.

In our numerical experiments on the 2D FSI problem, we investigate the difference in the results obtained with the four different transfer algorithms for non-matching meshes. These results are compared with the results obtained with matching meshes. We also study the performance of the new mesh deformation method based on radial basis function interpolation.

The contents of this chapter are organized as follows: Section 5.1 presents the benchmark problem of 2D flow around a cylinder with a deformable flap [110]. The numerical experiments and results with both matching and non-matching meshes are presented in Section 5.2 and Section 5.3 contains concluding remarks.

5.1 Problem statement

In this section we present the benchmark problem of 2D flow around a cylinder with a deformable flap. First the flow and structure model are given in Sections 5.1.1 and 5.1.2, respectively. The time integration and partitioning scheme are outlined in Section 5.1.3. Section 5.1.4 contains a short description of the coupling of the flow and structure solver with FLECS and the used transfer algorithms. The definition of the computational domain and the meshes used for the flow and structural part are presented in 5.1.5. Finally, in Section 5.1.6 a description of the initial and boundary conditions is given.

5.1.1 Flow model

The flow used in this chapter is laminar and compressible, where the fluid is an ideal gas. The governing equations for the flow are the Navier-Stokes equations and the ideal gas law: $p = \rho_f R_g T$, with p the pressure, ρ_f the density, T the temperature and R_g the gas constant. Since the fluid domain is deforming, the Navier-Stokes equations are written in the Arbitrary Lagrangian-Eulerian (ALE) formulation [41]. The general purpose, compressible Finite Volume flow solver Hexstream, developed by NUMECA Int. is used to solve the equations on an unstructured, hexahedral mesh [17].

The semi-discrete system for a cell l can be written as

$$\frac{\mathrm{d}(\Omega \mathbf{U})_l}{\mathrm{d}t} + \sum_{i=1}^{N_{l,face}} (\mathbf{\Phi}_i - \mathbf{U}_i \boldsymbol{\kappa}_{l,i}) \cdot \mathbf{S}_{l,i} = 0, \tag{5.1}$$

where $\mathbf{U}_l$ is the fluid state in conservative variables in the cell centre, Ω is the cell volume, $N_{l,face}$ denotes the number of faces that define cell l, $\mathbf{\Phi}_i$ is the numerical flux for the fluid dynamics equations as computed on static meshes for face i, $\mathbf{U}_i$ the fluid state at face i, $\mathbf{S}_{l,i}$ is the face surface times its normal and $\boldsymbol{\kappa}_{l,i}$ the velocity of the face surface in the moving mesh. In our case the numerical flux $\mathbf{\Phi}$ is computed using the standard second order central scheme with Jameson type artificial dissipation [69] for the inviscid part of the flux. The semi-discrete system can be written more compactly as

$$\frac{\mathrm{d}(\Omega \mathbf{U})}{\mathrm{d}t} + \mathbf{R}_\mathrm{f}(\mathbf{U}, \boldsymbol{\kappa}) = 0, \tag{5.2}$$

wherein $\mathbf{R}_\mathrm{f}$ the semi-discrete fluid dynamics model. The exact values of the system parameters are given in Section 5.2.

The deformation of the flow mesh is performed with the method based on radial basis function interpolation introduced in Chapter 4. For the radial basis function we use the Thin Plate Spline, which was shown to give good quality meshes and does not depend on a user-defined parameter. The deformation of the mesh is each step defined with respect to the initial configuration to ensure that the initial mesh is obtained when the computational domain is back in its initial position.

5.1.2 Structure model

The structure model is a linear structure dynamics model without damping:

$$M\frac{\mathrm{d}^2\mathbf{Q}}{\mathrm{d}t^2} + K\mathbf{Q} = \mathbf{F}_{\mathrm{sf}}, \tag{5.3}$$

with M the mass matrix, K the stiffness matrix, $\mathbf{F}_{\mathrm{sf}}$ the pressure load from the flow that acts on the structure and $\mathbf{Q}$ the structural displacement vector. A separate structure solver is used to simulate the structural dynamics, where we use *OpenFEM* [1], a Finite Element toolbox for *Matlab*, to construct the matrices M and K according to the structure density ρ_s, Poisson ratio ν_s and Young modulus E_s. The values of these parameters are given in Section 5.2. The structure state vector $\mathbf{W} = \left(\begin{smallmatrix} M\dot{\mathbf{Q}} \\ \mathbf{Q} \end{smallmatrix}\right)$, which contains the structural momentum vector $M\dot{\mathbf{Q}}$ and the structural displacement vector $\mathbf{Q}$, is used to write the structure dynamics as a system of ordinary differential equations

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} + A_{\mathrm{s}}\mathbf{W} = \begin{pmatrix} \mathbf{F}_{\mathrm{sf}} \\ \mathbf{0} \end{pmatrix}, \tag{5.4}$$

with

$$A_{\mathrm{s}} = \begin{bmatrix} 0 & K \\ -M^{-1} & 0 \end{bmatrix}, \tag{5.5}$$

which is the semi-discrete formulation of the structure dynamics. A more general formulation of (5.4) is

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} + \mathbf{R}_{\mathrm{s}}(\mathbf{W}, \mathbf{F}_{\mathrm{sf}}) = 0, \tag{5.6}$$

with $\mathbf{R}_{\mathrm{s}}$ the semi-discrete structure dynamics model.

5.1.3 Time integration and partitioning scheme

A mixed implicit/explicit (IMEX) higher order Runge-Kutta (RK) scheme [122] is used for the coupled time integration of the flow and the structure. This scheme uses an implicit ESDIRK scheme [16] for the integration of the fluid and structure dynamics and an explicit RK scheme to integrate the coupling term of the flow to the structure. In this chapter we use the third order version of the IMEX scheme.

Since the integration of the coupling term is explicit, sub-iterations are needed to obtain a stable simulation in strongly coupled systems. The sub-iteration process takes place within each stage of the IMEX scheme. In order to reduce the number of sub-iterations within a stage, Aitken acceleration [67, 89] is used on the explicit part: the coupling term $\mathbf{F}_{\mathrm{sf}}$.

5.1.4 Coupling shell and transfer algorithms

The flow and structure solver are separate programs. In order to perform a coupled fluid-structure interaction simulation they are coupled by using the flexible coupling shell FLECS, introduced in Chapter 3. In Table 3.3 of Chapter 3 the setup of the flow

and structure solver are shown after they are adapted to work with FLECS. To be able to perform the coupled computation, only function calls from the FLECS client library have to be added, whereas the main structure of the solvers remains the same.

To transfer information between non-matching meshes, different transfer algorithms can be used with either a conservative or a consistent approach as discussed in Chapter 2. In the current chapter we consider four of the transfer algorithms introduced in Chapter 2: consistent simple nearest neighbour (NN) interpolation (section 2.2.1), consistent and conservative radial basis function (RBF) interpolation (section 2.2.3) and a method that uses consistent radial basis function interpolation to transfer displacement and simple nearest neighbour interpolation for the pressure function (RBF-NN, see section 2.4.4). As we are using a Finite Volume flow solver, we do not investigate the weighted residual method (section 2.2.2), because this method explicitly uses the shape functions at the interface, which are in this case constant for the flow. The four transfer algorithms are implemented within the separate server program that is created to couple the two solvers using FLECS.

5.1.5 Domain and mesh definition

The domain is the same as in the well-known CFD benchmark of 2D flow around a cylinder [111], only this time a deformable flap is attached to the cylinder. The exact configuration is shown in Figure 5.1. The domain dimensions are given by the length

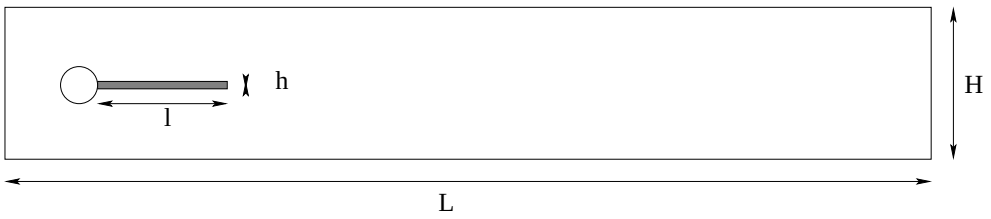


Fig. 5.1: Computational domain.

$L = 2.5$ and height $H = 0.41$. The centre of the cylinder is positioned at $C = (0.2, 0.2)$ from the left bottom corner of the channel and its radius is $r = 0.05$. The elastic flap has length $l = 0.35$ and height $h = 0.02$ and the right bottom corner is positioned at (0.6,0.19), where the left end is fully attached to the fixed cylinder. The setting is intentionally non-symmetric to prevent the dependence of the onset of any possible oscillation on the precision of the computation. A close-up of the structure part is given in Figure 5.2.

For the flow mesh we use 20.737 hexahedral elements and the initial mesh is shown in Figure 5.3. A close-up of the mesh on the cylinder with flap is given in Figure 5.4 and 244 vertices of the flow mesh lie on the moving interface with the flap.

For modeling the two-dimensional solid for the case with non-matching meshes, we use the standard 8 node, 16 degrees-of-freedom (DOF), quadrilateral-element [66] which uses quadratic shape functions. For the case with matching meshes the standard 4 node, 8 DOF quadrilateral-element is used, which uses linear shape functions. To

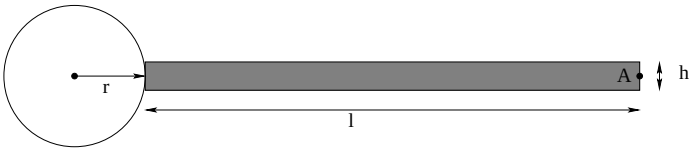


Fig. 5.2: Detail of the structure part.

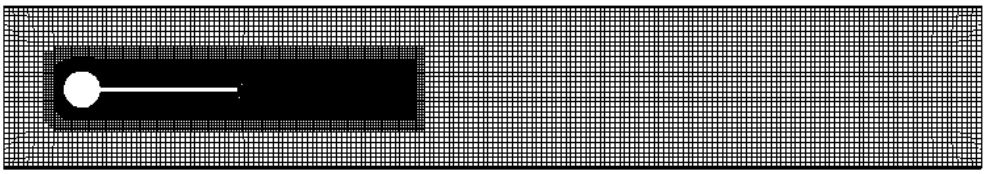


Fig. 5.3: Initial mesh.

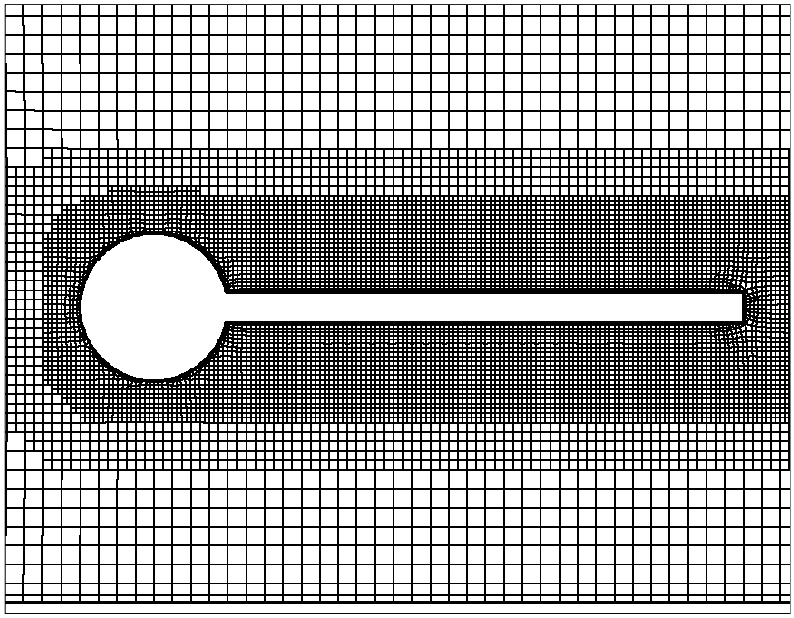


Fig. 5.4: Close-up of initial mesh

obtain matching meshes at the fluid-structure interface 1287 (117×11) elements are needed. In Figure 5.5 the error in the y-displacement of the free moving end of the flap versus $1/N$ is plotted for the linear and quadratic basis functions, where N is the number of elements. The solutions are obtained by performing the $CSM2$ test case, given in [110], which consists of placing the flap in a gravitational field with $g = 2m/s^2$ and looking at the steady state solution. The error is computed with

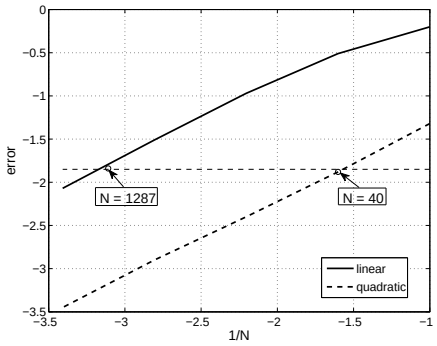


Fig. 5.5: A loglog plot of the error in the y-displacement of the free moving end of the flap versus $1/N$, with N the number of elements, for linear and quadratic elements.

respect to the solution obtained with a very fine mesh. It can be seen that using 1287 linear elements gives approximately the same error as using 40 quadratic elements. Therefore we use 40 (20×2) quadratic elements in the non-matching case, to obtain an FSI solver which is comparable to the one with matching meshes.

5.1.6 Boundary and initial conditions

For the flow a parabolic velocity profile is prescribed at the left channel inflow such that the mean inflow velocity is $\bar{v}$ and the maximum of the inflow velocity profile is $1.5\bar{v}$. At the outflow the static pressure is prescribed. On the solid boundary parts of the flow (top and bottom wall, cylinder and flap) a no-slip condition is prescribed for the fluid. For the structure the left end of the flap is fully attached to the fixed cylinder. The right end can move freely according to the structure dynamics.

As initial condition we use fully developed flow with vortex-shedding obtained when performing the flow computations with a rigid flap, where the flap is in the position as shown in Figure 5.2.

5.2 Numerical results

In [110] three different benchmarks for the fluid, structure and the fluid-structure interaction solver are presented. We consider here only the *FSI3* test case for the fluid-structure interaction, which results in a periodic motion of the flap. The parameter settings for this test case are: $\rho_f = \rho_s = 10^3 kg/m^3$, $\bar{v} = 2m/s$, $\nu_f = 10^{-3}m^2/s$, $\nu_s = 0.4$ and $E_s = 5.6 \cdot 10^6 kg/ms^2$. This corresponds to a flow with a Reynolds number of $Re = \frac{\bar{v}d}{\nu_f} = 200$, with $d = 2r$ the diameter of the cylinder, a density ratio between structure and flow of $\beta = \rho_s/\rho_f = 1$ and a ratio between the stiffness of the structure and aerodynamic forces of the flow of $Ae = \frac{E_s}{\rho_f \bar{v}^2} = 1.4 \cdot 10^3$. The values of R_g and T in the compressible flow solver are chosen in such a way that the Mach number

$Ma = \bar{v}/\sqrt{\gamma R_g T}$ is equal to 0.1, such that incompressible flow can be assumed. The displacement is each time step computed with regard to the initial configuration.

To compare the results of the different computations the following quantities of comparison are defined: the lift force on the total body (cylinder and flap together) and the displacement in y-direction of point A (see Figure 5.2). The lift force is calculated by integrating the pressure and viscous forces over the surfaces of the cylinder and the flap. The lift force is then the force in vertical direction.

As the motion of the flap is periodic, so are the quantities of comparison. Therefore these quantities are represented by the mean value, amplitude, standard deviation of the amplitude (stdv) and frequency of the last two full periods:

$$\text{mean} = \tfrac{1}{2}(\max + \min), \tag{5.7}$$

$$\text{amplitude} = \max - \text{mean}, \tag{5.8}$$

$$\text{stdv} = \tfrac{1}{2}\left(\text{stdv}(\max) + \text{stdv}(\min)\right), \tag{5.9}$$

$$\text{frequency} = \frac{1}{\text{period}}, \tag{5.10}$$

where 'max' is the average of the maximum values, 'min' the average of the minimum values, 'stdv(max)' the standard deviation of the maximum values, 'stdv(min)' the standard deviation of the minimum values and 'period' the average time of the last two periods.

We do not expect to obtain the exact same results as in [110] as a linear structure and a compressible flow solver are used instead of a non-linear structure and an incompressible flow solver. However the order of magnitude of the quantities of comparison should be the same. First, the results for matching meshes are presented in section 5.2.1. Here we investigate the difference between transferring pressure or the pressure force over the interface for two different time steps. After that the results obtained with different transfer algorithms for non-matching meshes are given in section 5.2.2.

5.2.1 Matching meshes

When the meshes are matching, either the pressure force or the pressure itself can be transferred from the flow interface to the structure interface. In the first case the pressure force in a flow vertex is obtained by first computing the pressure force on a cell face by multiplying the pressure in a cell face with the area of this face. The pressure force on a cell face is then equally distributed over the adjacent vertices. This corresponds to using constant basis functions for the pressure forces. In the second case the pressure values in the cell faces are linearly interpolated to the vertices. To obtain the pressure force on the structure these pressure values are multiplied with a 'mass matrix' of the structure interface, which corresponds to using linear basis functions for the pressure forces.

As starting condition for the comparison we use a fully developed solution, where the flap exhibits a periodic motion due to the vortex shedding from the cylinder. This solution is obtained with a time step of $\Delta t = 0.01\,s$ where the pressure force

is transferred over the interface. From here we advance one second (approximately 5 periods) with four different settings: transferring forces or pressure with the same time step ($\Delta t = 0.01\,s$) and transferring forces or pressure with a 10 times smaller time step ($\Delta t = 0.001\,s$). For the time step of $\Delta t = 0.01\,s$ on average ten sub-iterations were performed within one stage of the IMEX scheme, while for the time step of $\Delta t = 0.001\,s$ three sub-iterations per stage were performed on average.

Lift

The results for the lift force are plotted in Figure 5.6 and in Table 5.1 the mean, amplitude, standard deviation and frequency are given for the four different settings. These quantities are computed using the last two full periods. It can be seen that

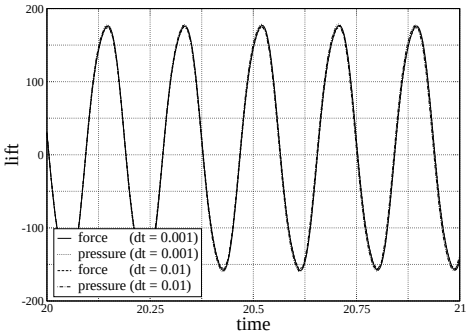


Fig. 5.6: Lift force on the cylinder+flap for matching meshes.

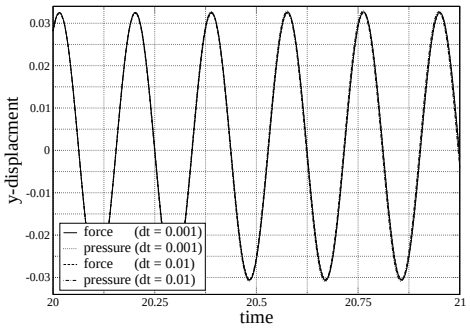


Fig. 5.7: Displacement in y-direction of point A for matching meshes.

Table 5.1: Lift results for matching meshes.

		mean (N)	amplitude (N)	stdv (N)	frequency (Hz)
$\Delta t = 0.01\,s$	forces	9.3	166.5	0.34	5.34
	pressure	**9.5**	**168.0**	**0.66**	**5.36**
$\Delta t = 0.001\,s$	forces	9.3	167.2	0.59	5.34
	pressure	9.6	168.7	0.97	5.35

the results are very similar for all four settings and the difference between the lines in Figure 5.6 is barely visible. Using a smaller time step results both for transferring forces and pressures in a slightly larger amplitude, which can be explained by the fact that the numerical damping decreases with decreasing time steps. However, the difference is only 0.4% and falls within the error range given by the standard deviation. When pressure is transferred instead of force, the mean and amplitude are a little higher (3% and 0.9%, respectively). The difference can be explained by the fact that

with transferring pressures a linear approximation of the pressure forces is used and with transferring forces a constant approximation.

Displacement

The results for the displacement in y-direction of point A are shown in Figure 5.7 and Table 5.2. The difference between the results of the four settings is very small

Table 5.2: Displacement in y-direction of point A for matching meshes.

		mean (10^{-3} m)	amplitude (10^{-3} m)	stdv (10^{-3} m)	frequency (Hz)
$\Delta t = 0.01\,s$	forces	1.00	31.4	0.037	5.35
	pressure	**1.01**	**31.7**	**0.099**	**5.36**
$\Delta t = 0.001\,s$	forces	0.99	31.6	0.073	5.34
	pressure	1.01	31.8	0.14	5.36

and the lines in Figure 5.7 lie almost on top of eachother. Again the amplitude is slightly higher (0.5%) for the smaller time step, but falls within the error range of the standard deviation. We therefore can conclude that a time step of $\Delta t = 0.01\,s$ is small enough for an accurate solution. Also the difference between transferring pressure and force is only 1.9% for the mean and 0.7% for the amplitude.

Reference solution

As a reference solution for the case with non-matching meshes we will use the results obtained with a time step of $\Delta t = 0.01\,s$ and where the pressure is transferred over the interface, because it was shown that this time step is small enough for an accurate solution and with the transfer of pressures a linear approximation of the forces is obtained.

5.2.2 Non-matching meshes

In this section we compare the results obtained when different transfer methods are used to transfer the pressure (or pressure force) and displacement between the non-matching flow and structure mesh. For the structure we use the mesh with 40 quadratic elements and for the flow the mesh as is shown in Figure 5.3.

As starting condition for the comparison we use a fully developed solution, where the flap exhibits a periodic motion due to the vortex shedding from the cylinder. This solution is obtained by starting from the initial solution given in section 5.1.6: fully developed flow with vortex-shedding obtained when performing the flow computations with a rigid flap. To obtain the starting condition for the comparison we advance from the initial condition with a time step of $\Delta t = 0.01\,s$ where the pressure and displacement are transferred over the interface using the consistent radial basis function interpolation method until periodic motion is obtained. In Section 5.2.1 it

was shown that a time step of $\Delta t = 0.01\,s$ is small enough to obtain a time accurate solution.

From here we compute one second (approximately 5 periods) with the same time step using one of the four transfer methods: transferring pressures and displacements with nearest neighbour interpolation (NN), transferring pressures and displacements with consistent radial basis function interpolation (RBF consis), transferring forces and displacements with conservative radial basis function interpolation (RBF conserv) or transferring displacements with consistent radial basis function interpolation, but pressures with nearest neighbour interpolation (RBF-NN). For the radial basis function the thin plate spline is used, which was shown to give good results in Chapter 2 and does not depend on a user-defined parameter. For all transfer methods on average 7 sub-iterations were performed within one stage of the IMEX scheme.

Lift

The results for the lift force are plotted in Figure 5.8 and in Table 5.3 the mean, amplitude, standard deviation and frequency are given for the four different transfer algorithms. These quantities are computed using the last two full periods. The difference with the reference solution with matching meshes is given between brackets.

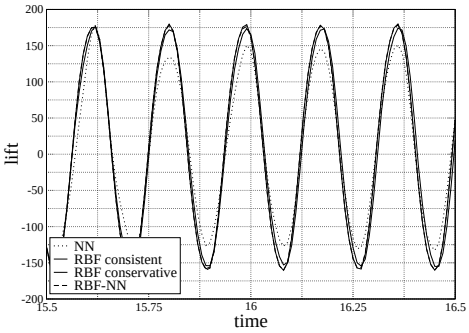


Fig. 5.8: Lift force on the cylinder+flap for non-matching meshes.

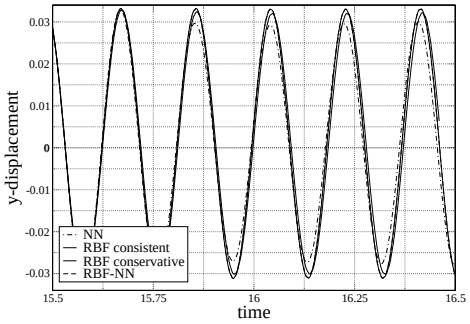


Fig. 5.9: Displacement in y-direction of point A for non-matching meshes.

Table 5.3: Lift results for non-matching meshes. Between brackets the difference with the reference solution is given.

	mean (N)		amplitude (N)		stdv (N)	frequency (Hz)	
NN	9.50	(0.12%)	139.0	(17.28%)	3.28	5.46	(1.95%)
RBF consis	9.48	(0.29%)	169.8	(1.08%)	0.75	5.38	(0.35%)
RBF conserv	10.23	(7.63%)	164.5	(2.09%)	0.72	5.37	(0.18%)
RBF-NN	9.47	(0.39%)	168.9	(0.56%)	0.82	5.38	(0.51%)

It can be seen that the solution obtained with NN deviates the most from the reference solution. This can be explained by the fact that transferring the displacement from the coarser structure mesh to the fine flow mesh with nearest neighbour interpolation results in a 'staircase' shape of the flap as shown in Figure 5.10. This

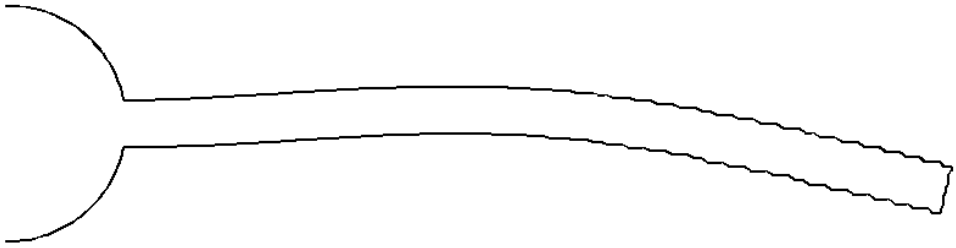


Fig. 5.10: Deformed flap with 'staircase' shape obtained when transferring displacements with nearest neighbour interpolation.

gives an incorrect representation of the flap, resulting in a smaller lift force.

The results obtained by transferring both pressure and displacement with consistent RBF (RBF consis) or transferring pressure with NN and displacement with consistent RBF (RBF-NN) are closest to the reference solution of the four methods investigated, and the difference between them is negligible. However, the computational costs for nearest neighbour interpolation are much smaller than for consistent radial basis function interpolation, as is shown in Chapter 2. Therefore, RBF-NN is preferred over RBF consis.

The results obtained with the conservative approach (RBF conserv) deviate more from the reference solution than the results obtained with the consistent approach (RBF consis), especially for the mean value. The larger deviation can be explained by the fact that quadratic shape functions are used for the structure. In Figure 5.11 the difference between the pressure and the reference pressure is plotted versus the length of the flap. The results are shown for the pressure calculatated by the flow and the pressure received by the structure. The pressure received by the structure is computed both with RBF consis and RBF conserv. It can be seen that pressure on the structure obtained with RBF consis is almost equal to the pressure calculated by the flow. In the pressure obtained with RBF conserv oscillations are visible. This can be explained by the fact that using the conservative approach with radial basis function interpolation results in a highly oscillatory pressure force on the structure when higher order shape functions are used, as was shown in Section 2.2.3.

Displacement

The results for the displacement in y-direction of point A are shown in Figure 5.9 and Table 5.4. Again NN is the least accurate and the reason is the same as for the lift force. The results obtained by the other three transfer methods are all within 2%

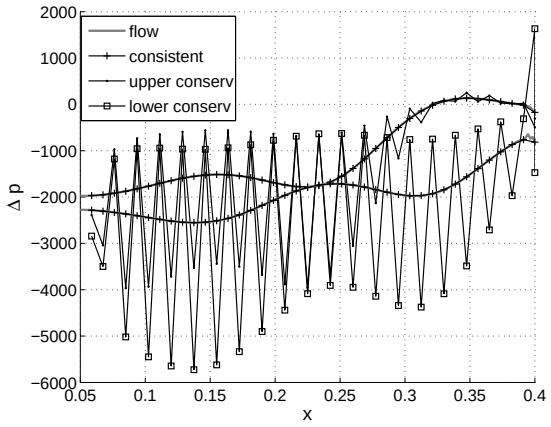


Fig. 5.11: Pressure difference over the flap.

Table 5.4: Displacement in y-direction of point A for non-matching meshes.

	mean (10^{-3} m)		amplitude (10^{-3} m)		stdv (10^{-3} m)	frequency (Hz)	
NN	1.17	(15.13%)	28.6	(9.50%)	0.250	5.44	(1.49%)
RBF consis	1.00	(1.35%)	32.1	(1.45%)	0.032	5.38	(0.40%)
RBF conserv	1.02	(0.27%)	31.1	(1.72%)	0.079	5.36	(0.05%)
RBF-NN	1.00	(1.35%)	32.1	(1.33%)	0.038	5.38	(0.42%)

from the reference solution, and there is no method that performs significantly better than the others. RBF conserv is closer to the reference solution for the mean value and the frequency, while RBF consis and RBF-NN perform better for the amplitude. The difference between RBF-NN and RBF consis is again negligible.

With RBF conserv we did not observe oscillations in the shape of the structure due to the large oscillations in the pressure force as was the case in the quasi-1D channel with a flexible curved wall of Section 2.4. This can be explained by the fact that the flap is stiffer than the flexible wall in the quasi-1D case. When the structure is rather stiff, oscillations in the pressure force are not 'felt' by the structure, because the structure only responds to the global force.

Conclusions

Nearest neighbour interpolation should not be used to transfer displacements from a coarse to a fine mesh, because it results in a 'staircase' shape of the flap. The difference in the results obtained with transferring both pressure and displacement with consistent RBF (RBF consis) or transferring pressure with NN and displacement with consistent RBF (RBF-NN) is negligible, but the computational costs of RBF-

NN are smaller than those of RBF consis. The results obtained with RBF conserv are comparable to the ones obtained with RBF consis and RBF-NN for the displacement in y-direction of point A, located at the free moving end of the flap. For the lift the results obtained with RBF consis and RBF-NN are closer to the reference solution than the results obtained with RBF conserv. This is due to the fact that quadratic shape functions are used for the structure and therefore a highly oscillatory pressure force is obtained. Overall we conclude that RBF-NN is preferred to transfer pressure and displacement between non-matching meshes for this $FSI3$ test case.

5.2.3 Mesh deformation

The deformation of the mesh each step is defined with respect to the initial configuration. This means that the deformations can become quite large within one step, but it ensures that the mesh is equal to the initial mesh when the domain returns to its initial form. A close-up of the mesh when the flap is at its maximum deflection is shown in Figure 5.12. The mesh movement is smoothly spread through the domain

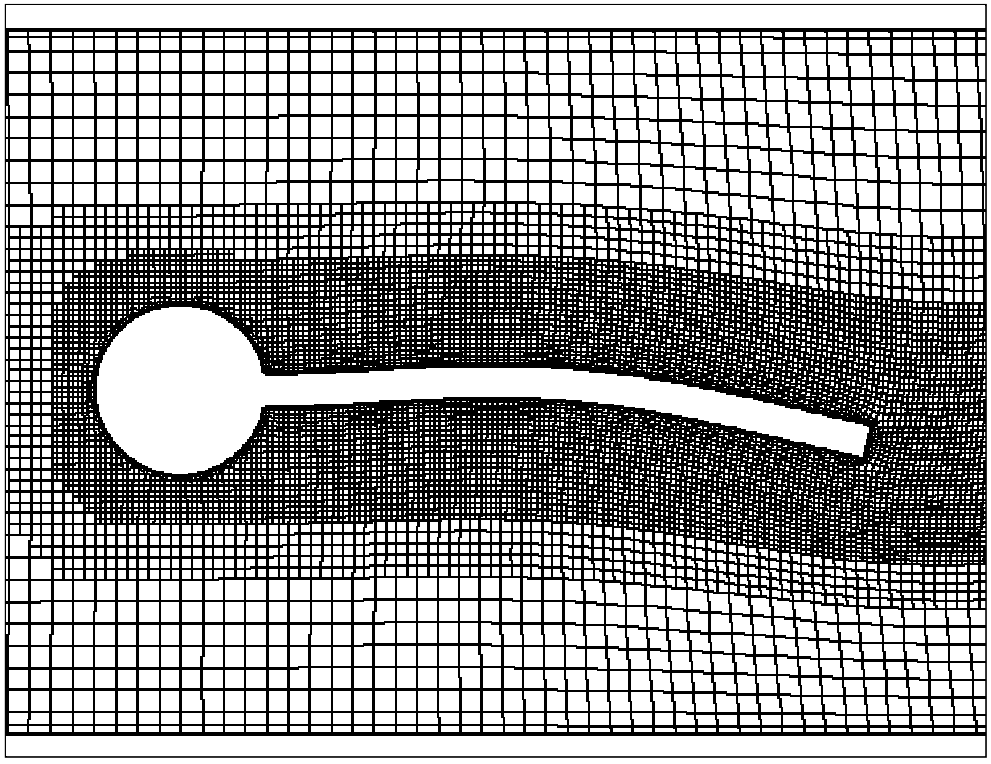


Fig. 5.12: Deformed mesh at maximum deflection of the flap.

and close to the flap the cells follow almost exactly its movement, therefore the mesh quality remains high throughout the computation.

The mesh deformation is performed with a direct solve of the matrix system and direct evaluation in the internal grid points. For this small 2D test case the time needed for the mesh deformation is less than 10% of the time needed for the flow solve. With the original mesh deformation method, based on solving the Laplace equation to create a displacement field [76], the time needed for the deformation is about 10% more than the time needed for the flow solve. This means that the new mesh deformation method is more than a factor 10 faster for this particular problem.

For large 2D and 3D meshes it is not possible anymore to solve the matrix system directly and the direct evaluation in the internal grid points becomes very time consuming. For these meshes it is needed to improve the efficiency of the new mesh deformation method by using a combination of a reduction of the number of centres, parallel computation strategies, iterative techniques and fast evaluation algorithms as discussed in Section 4.7.

5.3 Conclusions

To investigate the performance of four transfer algorithms introduced in Chapter 2 for a 2D test case, we performed an FSI computation of 2D flow around a cylinder with deformable flap. The flow and structure solver are coupled by using the flexible coupling shell FLECS, introduced in Chapter 3, and only function calls from the FLECS client library have to be added to the solvers, whereas their main structure remains the same.

To obtain a reference solution we first performed computations with matching meshes. A time step of $\Delta t = 0.01\ s$ was shown to be small enough for an time-accurate solution. Furthermore, when pressures are transferred over the interface an approximation with linear functions is obtained for the forces is contrast to the approximation with constant functions obtained by directly transferring pressure forces.

For the non-matching case we investigated four of the transfer algorithms introduced in Chapter 2: consistent simple nearest neighbour (NN) interpolation (section 2.2.1), consistent and conservative radial basis function (RBF) interpolation (section 2.2.3) and a mix of consistent radial basis and nearest neighbour interpolation function (RBF-NN) interpolation (see section 2.4.4). The four transfer algorithms are implemented within the separate FLECS server program that handles the data transfer between the two solvers.

The solution obtained with NN deviates the most from the reference solution, because transferring the displacement from the coarser structure mesh to the fine flow mesh with nearest neighbour interpolation results in a 'staircase' shape of the flap. This gives an incorrect representation of the flap, resulting in a smaller lift force and smaller amplitude of the displacement of the flap.

The difference between transferring both pressure and displacement with consistent RBF (RBF consis) or transferring pressure with NN and displacement with consistent RBF (RBF-NN) is negligible. However, the computational costs for nearest neighbour interpolation are much smaller than for consistent radial basis function interpolation, as is shown in Chapter 2. Therefore, RBF-NN is preferred over RBF

consis.

The results obtained with conservative RBF are comparable to the ones obtained with RBF consis and RBF-NN for the displacement in y-direction of point A. However, the results for the lift deviate further from the reference solution compared to RBF consis and RBF-NN. This is due to the fact that quadratic shape functions are used for the structure and therefore a highly oscillatory pressure force is obtained. Overall we conclude that RBF-NN is preferred to transfer pressure and displacement between non-matching for this $FSI3$ test case.

The deformation of the flow mesh is performed with the method based on radial basis function interpolation introduced in Chapter 4, where the Thin Plate Spline is used for the radial basis function. The deformation of the mesh is each step defined with respect to the initial configuration. This means that the deformations can become quite large within one step, but it ensures that the mesh is equal to the initial mesh when the domain returns to its initial form. During the computation the mesh movement is smoothly spread through the domain and close to the flap the cells follow almost exactly its movement, therefore the mesh quality remains high throughout the computation. The mesh deformation is performed by using direct methods for solving the matrix system and the evaluation of the interpolation function. For this 2D test case the time needed for the mesh deformation is less than 10% of the time needed for the flow solve and the method is more than a factor 10 faster then the original mesh deformation method, based on solving the Laplace equation to create a displacement field.

Chapter 6

Conclusions

In this thesis we investigated several aspects of fluid-structure interaction computations. To be able to transfer information between the flow and the structure solver, a transfer algorithm is needed to deal with incompatible meshes at the interface. We showed that generally a consistent approach should be used to avoid unphysical oscillations and the best choice for a particular transfer method depends on the spatial discretization order of the total system. To facilitate the computation of multidisciplinary problems and fluid-structure interaction simulations in particular, a flexible coupling shell for the coupling of different solvers is developed. Also the flow domain, and the mesh defined on it, has to adapt itself to the deforming structure. We developed a new point-by-point mesh deformation technique that can handle large deformations. Finally, a real 2D fluid-structure interaction computation is performed in which the three previous findings are incorporated. We summarize here the main conclusions with respect to these developments.

Conservative and consistent transfer algorithms

The difference in accuracy and efficiency between conservative and consistent approaches for different transfer methods between non-matching meshes is investigated. When the transfer method is based on a weighted residual formulation of the coupling conditions, the highest accuracy and efficiency are obtained with the conservative approach. For other transfer methods the conservative approach results in large unphysical oscillations in the pressure received by the structure leading to a zeroth order convergence for the pressure. When the structure is flexible enough these oscillations can result in deviations in the displacement of the flow interface. For these methods the consistent approach provides the best accuracy and efficiency. Simple Nearest Neighbour interpolation should not be used, because the error in both displacement and pressure does not converge when the consistent approach is used in the quasi-1D test case.

Overall, when the spatial discretization order of the total system is higher than two, the weighted residual method is the best choice. However, its implementation is

more elaborate and the computation time is higher than for the methods based on radial basis function interpolation. This is because the higher order of the weighted residual method is only obtained when the projection step is accurately performed. Therefore, when the discretization of the total system is of order two or lower, the methods based on radial basis function (RBF) interpolation are preferred where the compact RBF with a large support radius is the best choice.

Using two different methods to interpolate displacements and pressures proves also to be very efficient: an accurate method (RBF with high support radius) is used to interpolate from coarse to fine and a less accurate, but much faster method (simple Nearest Neighbour) is used to interpolate from fine to coarse.

Flecs: a flexible coupling shell

A generic, flexible coupling shell, FLECS, designed for implementing and applying an interface for multidisciplinary simulations, is developed. The aim of FLECS is to provide a flexible platform for developing new data transfer algorithms and coupling schemes to be able to perform large multidisciplinary computations. The design of FLECS is based on a client-server model in which two solvers communicate with a separate program called the coupling server that is responsible for transferring data from one physical domain to another. This design makes it possible to run the solvers on two different computers that are located at different research institutes. FLECS satisfies the following specifications:

- The software is open source, such that everyone can make modifications and extensions to the code.
- The coupling shell is minimally intrusive in the solvers. The main structure of the solvers remains the same, only a few subroutine calls have to be added in a high level of the code.
- The coupling shell is interoperable with different programming languages, such as `Fortran 90`, `C` and `C++`.
- The information transfer is handled within a separate server program.
- The transfer algorithm is based on a plug-in architecture, such that a new transfer algorithm can be implemented without having to deal with the general communication between the two solvers.
- The coupling shell can deal with parallel solvers and parallel transfer algorithms, which means that large simulations can be performed.
- It is possible to couple one solver to more than one other solver by starting up a separate coupling shell for each combination of solvers.

Numerical acceleration techniques can be implemented as subroutines within the coupling server without having to change the separate solvers involved. This means that these acceleration techniques can be reused when one or more different solvers are coupled. FLECS supports the use of multiple grids on a single interface and therefore simplifies the use of multilevel acceleration techniques.

Mesh deformation based on radial basis function interpolation

Radial basis functions (RBFs) can be used to interpolate the displacements of the boundary nodes of the mesh to the inner domain, to obtain a point-by-point mesh deformation scheme. The method requires solving a small system of equations, only involving the nodes on the boundary of the flow domain. The implementation of the method is relatively simple, even for 3D applications, because no grid-connectivity information is needed. The method can handle large deformations of a mesh caused by translation, rotation and deformation of a structure both on 2D and 3D meshes. When efficiency is important, the CP C^2 RBF with compact support is the best choice, closely followed by the thin plate spline.

In a comparison the RBF-method produces meshes of higher quality than the popular semi-torsional spring analogy for very large deformations. The quality of the meshes after deformation is high enough to perform accurate flow calculations. If there is translation information available, the additional recovery of rotations does not improve the solution, but it increases the computation time dramatically.

It is shown that working with higher quality meshes obtained by the RBF-method can increase the efficiency of the computation. This is explained by the fact that the new method preserves the performance of higher order time integration methods, due to its smooth deformation in time. Also the larger deformations due to a larger time step, which are possible when using higher order time integration methods, can be handled with the RBF mesh deformation algorithm.

Two dimensional application

An FSI computation of flow around a cylinder with deformable flap is performed to investigate the performance of four transfer algorithms in a 2D setting and the use of Flecs and the new mesh deformation method in a real application. It was shown that when Flecs is used only a few subroutine calls had to be added to the flow and structure solver. With the new deformation method the mesh quality remains high throughout the computation. Even as the mesh deformation is performed by solving the system and evaluating the interpolation with direct methods, for this testcase the new method is more than a factor 10 faster than a method based on solving the Laplace equation to create a displacement field.

The solution obtained with the transfer algorithm based on only nearest neighbour interpolation deviates the most from the reference solution, because transferring the displacement from the coarser structure mesh to the fine flow mesh with nearest neighbour interpolation results in a 'staircase' shape of the flap. The difference between transferring both pressure and displacement with consistent RBF or transferring pressure with nearest neighbour and displacement with radial basis function interpolation is negligible. However, the computational costs for nearest neighbour interpolation are much smaller than for consistent radial basis function interpolation. Therefore, the former is preferred over the latter.

The use of quadratic shape functions for the structure, results in a highly oscillatory pressure force when radial basis function interpolation with a conservative approach is used. However, this has only an effect on the results for the lift and not

on the displacement of the structure. Overall we conclude that transferring pressure with simple nearest neighbour interpolation and displacement with consistent radial basis function interpolation gives the best results for this FSI test case.

Chapter 7

Recommendations

Based on the outcome of the investigations presented in this thesis, some recommendations can be given for further investigations.

In Chapter 2 it was shown that a transfer algorithm using a consistent approach does not conserve total energy over the interface. However, in unsteady partitioned computations energy is generally already not conserved due to errors caused by the time lag between flow and structure. When the coupling error introduced by the information transfer is smaller than the spatial and temporal discretization error, this coupling error does not have to affect the stability and accuracy of the computation, especially when the spatial and time discretization themselves are very dissipative. However, the conservation properties of consistent transfer algorithms in combination with a partitioned method in unsteady computations needs further investigation. Also the effect of spatial transfer algortithms on the accuracy of the global time integration scheme has not been investigated yet.

In Chapter 3 a generic, flexible coupling shell, Flecs is developed. So far, Flecs has only been applied to sequential solvers and sequential data transfer algorithms. As Flecs fascilitates the coupling of parallel solvers and the use of parallel data transfer algorithms it is recommended that Flecs is tested for these configurations. In the current setup for parallel computations, the data exchange with the solvers is handled by a single server process that distributes the data over the other server processes as shown in Figure 3.2. A more advanced setup is shown in Figure 3.3, where the different processes of the solvers directly exchange data with the (multiple) processes of the coupling server. This setup is currently under development.

The Flecs server library provides subroutines for establishing a connection with the solver programs; for listening for requests from those solvers; and for handling the communication with the solvers. It does not, however, provide subroutines for transferring data from one domain to another; these must be implemented by the user. However, it would be usefull to provide some standard transfer algorithms. In Chapter 5 nearest neighbour and radial basis function interpolation algorithms are implemented in a separate server program. These algorithms could be added to the server library, such that they can be applied by other users.

For strongly coupled multi-disciplinary problems, sub-iterations are needed within one time step to stabilize the computation. Since performing one iteration in these coupled problems is very computationally demanding the use of numerical acceleration techniques is needed to limit the number of sub-iterations in order to increase the efficiency of the computations. These acceleration methods could be implemented as subroutines within the FLECS server library, because they only need information located on the interface. The server program only has to store data from previous iterations. In this way numerical algorithms which are implemented to efficiently couple two solvers, can be reused when using one or more different solvers.

The computational costs of a simple sub-iteration can be reduced by multilevel acceleration techniques. Coupling the two solvers not only on the fine meshes, but also on coarser meshes leads to a significant efficiency improvement, since computations on a coarse mesh are less computatinally demanding. Untill now, these multi-level techniques are only tested for (quasi) 1D problems and need further investigation for mulitdimensional problems. The application of these multilevel techniques with existing solvers is simplified by FLECS, as FLECS supports the use of multiple grids on a single interface.

In Chapter 4 a new mesh movement algorithm for unstructured grids is developed which is based on interpolating displacements of the boundary nodes to the whole mesh with radial basis functions. The method can handle large deformations with good quality meshes, but further investigation is needed to increase the efficiency of the method. This can be obtained by reducing the number of centres involved and to improve the efficiency of the computation itself. The two main computational challenges of mesh moving with radial basis function interpolation are: (A) solving a dense, ill-conditioned matrix system, and (B) evaluating the interpolation function in all internal grid points. The computational costs of both (A) and (B) increase fast with an increasing number of boundary and internal points when direct methods are used. It is not necessary to solve (A) and (B) exactly, because the movement of the internal grid points can be arbitrary, as long as it results in a good quality mesh. This means that an iterative method with a weak convergence criterion can be used for (A). However, first a fast evaluation technique has to be found for (B), because fast evaluation is also needed in iterative solution methods. It is worthwhile to extend the Fast Multilevel Evaluation method with an Adaptive Mesh Refinement strategy to increase the efficiency for the highly non-uniform located centres encountered in our mesh deformation problems. If this gives no satisfactory results, using the Fast Multipole method is the second option. This method, although rather complicated to program, shows promising result, but until now only results with uniformly distributed centres are published. It remains therefore to be seen if these efficiency gains are also obtained in our mesh deformation problems.

Once a suitable fast evaluation technique has been found, the next step is to further investigate the use of an iterative solution technique for (A). For the preconditioning with approximate cardinal functions only results with uniformly distributed centres are published. When a Lagrange form of the interpolation function is used for the preconditioning, an efficiency increase is obtained for various positions of the centres. However, when the Thin plate spline is used and the distance between the centres

is locally very small, the convergence rate in the iterative procedure becomes very slow. These locally small distances between centres are typically the case in our mesh deformation problems and therefore it is suggested to first investigate the iterative technique based on approximate cardinal functions.

Another way to improve the efficiency of the computation is to reduce the number of centres involved in computing and evaluating the interpolation function. This can be obtained by removing the centres on the fixed outer boundary and multiplying the interpolation function with a cut-off function instead. This cut-off function is equal to one close to the moving boundary and decays to zero at the outer boundaries. Furthermore, when there is a clustering of centres at certain locations it is possible to only take into account a selection of these centres that are positioned at a minimal distance d_{ref} from each other. This both reduces the number of centres and improves the condition number of the problem. However, the exact shape of the cut-off function and the value for the minimal distance are problem dependent and need further investigation.

In Chapter 5 the numerical benchmark problem $FSI3$ of 2D flow around a cylinder with deformable flap is performed. However, we used a linear structure and a compressible flow solver, instead of a non-linear structure and an incompressible flow solver as given in the benchmark problem. Therefore we did not expect to obtain the same results. In order to compare the results with the benchmark problem, the computations should be repeated with the appropriate solvers. Also more extensive convergence studies should be performed both for the time step and the grid size, in order to draw stronger conlusions on the accuracy of the data transfer algorithms. Furthermore, further investigation is needed on the efficiency of the transfer algorithms for this two-dimensional problem and their computation time relative to the overall computation time.

Bibliography

[1] *OpenFEM – A Finite Element Toolbox for Matlab and Scilab*, 2006.

[2] D. Abouri, A. Parry, A. Hamdouni & E. Longatte, A stable fluid-structure-interaction algorithm: Application to industrial problems, *Journal of Pressure Vessel Technology-Transactions of the Asme*, vol. 128, no. 4, pp. 516–524, 2006.

[3] R. Ahrem, A. Beckert & H. Wendland, A new multivariate interpolation method for large-scale coupling problems in aeroelasticity, Conference proceedings IFADS, Munich, 2005.

[4] R. Ahrem, A. Beckert & H. Wendland, Recovering rotations in aeroelasticity, *Journal of Fluids and Structures*, vol. 23, no. 6, pp. 874–884, 2007.

[5] F. P. T. Baaijens, A fictitious domain/mortar element method for fluid-structure interaction, *International Journal for Numerical Methods in Fluids*, vol. 35, pp. 743–761, 2001.

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

[7] R. K. Beatson, J. B. Cherrie & C. T. Mouat, Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration, *SIAM Journal fo Scientific Computing*, vol. 22, no. 5, pp. 1717–1740, 2000.

[8] R. K. Beatson, J. B. Cherrie & D. L. Ragozin, Fast evaluation of radial basis functions: Methods of four-dimensional polyharmonic splines, *SIAM Journal on Mathematical Analysis*, vol. 32, pp. 1272–1310, 2001.

[9] R. K. Beatson & L. Greengard, A short course on fast multipole methods, in *Wavelets, Multilevel Methods, and Elliptic PDEs* (edited by J. Levesley, W. Light & M. Marletta), pp. 1–37, Oxford University Press, Oxford, 1997.

[10] R. K. Beatson & W. A. Light, Fast evaluation of radial basis functions: Methods for 2-dimensional polyharmonic splines, *IMA Journal of Numerical Analysis*, vol. 17, pp. 343–372, 1997.

[11] R. K. Beatson, W. A. Light & S. Billings, Fast solution of the radial basis function interpolation equations: Domain decomposition methods, *Advances in Computational Mathematics*, vol. 11, pp. 253–270, 1999.

[12] R. K. Beatson & G. N. Newsam, Fast evaluation of radial basis functions: part I, *Computers and Mathematics with Applications*, vol. 24, pp. 7–19, 1992.

[13] A. Beckert & H. Wendland, Multivariate interpolation for fluid-structure-interaction problems using radial basis functions, *Aerospace Science and Technology*, vol. 5, no. 2, pp. 125–134, 2001.

[14] C. Bernardi, Y. Maday & A. T. Patera, A new nonconforming approach to domain decomposition: The mortar element method, Nonlinear Partial Differential Equations and their Applications, College de France Seminar, 1994.

[15] C. Bertram & D. P. G. DP, Biofluid mechanics of the pulmonary system, *Annals Of Biomedical Engineering*, vol. 33, no. 12, pp. 1681–1688, 2005.

[16] H. Bijl, M. H. Carpenter, V. N. Vatsa & C. A. Kennedy, Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: Laminar flow, *Journal of Computational Physics*, vol. 179, pp. 313–329, 2002.

[17] H. Bijl, A. H. van Zuijlen & A. van Mameren, Validation of adaptive unstructured hexahedral mesh computations of flow around a wind turbine airfoil, *International Journal for Numerical Methods in Fluids*, vol. 48, pp. 929–945, 2005.

[18] S. D. Billings, R. K. Beatson & G. N. Newsam, Interpolation of geophysical data with continuous global surfaces, *Geophysics*, vol. 67, pp. 1810–1822, 2002.

[19] S. D. Billings, G. N. Newsam & R. K. Beatson, Smooth fitting of geophysical data with continuous global surfaces, *Geophysics*, vol. 67, pp. 1823–1834, 2002.

[20] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker & R. C. Whaley, *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[21] F. J. Blom, A monolithical fluid-structure interaction algorithm applied to the piston problem, *Computer Methods in Applied Mechanics and Engineering*, vol. 167, no. 3–4, pp. 369–391, 1998.

[22] F. J. Blom, Considerations on the spring analogy, *International Journal for Numerical Methods in Fluids*, vol. 32, pp. 647–668, 2000.

[23] A. de Boer, H. Bijl & A. van Zuijlen, Comparing different methods for the coupling of non-matching meshes in fluid-structure interaction computations, 17th AIAA Computational Fluid Dynamics Conference, 2005.

[24] A. de Boer, H. Bijl & A. van Zuijlen, Coupling of non-matching meshes in fluid-structure interaction computations, ECCOMAS Thematic Conference: Coupled Problems 2005, 2005.

[25] A. de Boer, M. S. van der Schoot & H. Bijl, Moving mesh algorithm for unstructured grids based on interpolation with radial basis functions, 3th European Conference on Computational Mechanics, ECCM 2006, 2006.

[26] A. de Boer, M. S. van der Schoot & H. Bijl, New method for mesh moving based on radial basis function interpolation, European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006, 2006.

[27] A. de Boer, M. S. van der Schoot & H. Bijl, Mesh deformation based on radial basis function interpolation, *Computers and Structures*, vol. 85, no. 11—14, pp. 784–795, 2007.

[28] A. de Boer, A. H. van Zuijlen & H. Bijl, Comparison of a conservative and a consistent approach for the coupling of non-matching meshes, European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006, 2006.

[29] A. de Boer, A. H. van Zuijlen & H. Bijl, Review of coupling methods for non-matching meshes, *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 8, pp. 1515–1525, 2006.

[30] A. de Boer, A. H. van Zuijlen & H. Bijl, Comparison of conservative and non-conservative coupling approaches for non-matching meshes, in *2nd GACM Colloquium on Computational Mechanics*, German Association for Computational Mechanics, 2007.

[31] A. de Boer, A. H. van Zuijlen & H. Bijl, Mesh movement based on radial basis function interpolation, in *Computational Methods for Coupled Problems in Science and Engineering II*, ECCOMAS / International Center for Numerical Methods in Engineering, 2007.

[32] A. de Boer, A. H. van Zuijlen & H. Bijl, Comparison of conservative and consistent approaches for the coupling of non-matching meshes, *Computer Methods in Applied Mechanics and Engineering*, vol. 0, pp. 1–16, 2008.

[33] M. D. Buhmann, Radial basis functions, *Acta Numerica*, vol. 9, pp. 1–38, 2000.

[34] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan & T. J. Mitchell, Smooth surface reconstruction from noisy range data, First International Conference on Computer Graphics and Interactive Techniques, 2003.

[35] J. C. Carr, W. R. Fright & R. K. Beatson, Surface interpolation with radial basis functions for medical imaging, *IEEE Transactions on Medical Imaging*, vol. 96–107, 1997.

[36] V. Carstens, R. Kemme & S. Schmitt, Coupled simulation of flow-structure interaction in turbomachinery, *Aerospace Science and Technology*, vol. 298–306, no. 4, 2003.

[37] J. R. Cebral & R. Löhner, Conservative load projection and tracking for fluid-structure problems, *AIAA Journal*, vol. 35, no. 4, pp. 687–692, 1997.

[38] C. H. Charbel Farhat & D. J. Rixen, Expanding a flutter envelope using accelerated flight data: Application to an F16 fighter configuration, AIAA 2000-1702, 41th AIAA/ASME/ ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2000.

[39] C. Degand & C. Farhat, A three-dimensional torsional spring analogy method for unstructured dynamic meshes, *Computers and Structures*, vol. 80, pp. 305–316, 2002.

[40] J. W. Demmel, J. R. Gilbert & X. S. Li, An asynchronous parallel supernodal algorithm for sparse gaussian elimination, *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 4, pp. 915–952, 1999.

[41] J. Donea, An arbitrary lagrangian-eulerian finite element method for transient fluid-structure interactions, *Computer Methods in Applied Mechanics and Engineering*, pp. 689–723, 1982.

[42] D. Dureisseix & H. Bavestrello, Information transfer between incompatible finite element meshes: Application to coupled thermo-viscoelasticity, *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44–47, pp. 6523–6541, 2006.

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

[44] C. Farhat & M. Géradin, On a component mode synthesis method and its application to incompatible substructures, *Computers and Structures*, vol. 51, pp. 459–473, 1994.

[45] C. Farhat & M. Lesoinne, Improved staggered algorithms for the serial and parallel solution of three dimensional nonlinear transient aeroelastic problems, Tech. Rep. CU-CAS-97-11, 1997.

[46] C. Farhat, M. Lesoinne & P. Tallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity, *Computer Methods in Applied Mechanics and Engineering*, vol. 157, pp. 95–114, 1998.

[47] C. Farhat, K. van der Zee & P. Geuzaine, Provably second-order time-accurate loosely coupled solution algorithms for transient nonlinear computational aeroelasticity, *Computer Methods in Applied Mechanics and Engineering*, vol. (in press), 2004.

[48] A. C. Faul, G. Goodsell & M. J. D. Powell, A Krylov subspace algorithm for multiquadric interpolation in many dimensions, *IMA Journal of Numerical Analysis*, vol. 25, pp. 1–24, 2005.

[49] A. C. Faul & M. J. D. Powell, Krylov subspace methods for radial basis function interpolation, in *Numerical Analysis 1999* (edited by D. F. Griffiths), pp. 115–141, Chapman and Hall, London, 1999.

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

[51] C. A. Felippa, K. C. Park & C. Farhat, Partitioned analysis of coupled mechanical systems, *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 3247–3270, 2001.

[52] Fraunhofer Institut for Algorithms and Scientific Computing SCAI, *MpCCI, Mesh-based parallel Code Coupling Interface - Specification of MpCCI Version 2.0*, 2003.

[53] P. P. Friedmann, Vibration reduction in rotorcraft using active control: A comparison of various approaches, *Journal of Guidance, Control and Dynamics*, vol. 18, pp. 664–673, 1995.

[54] J.-F. Gerbeau & M. Vidrascu, A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows, *Mathematical Modelling and Numerical Analysis*, vol. 37, no. 4, pp. 631–647, 2003.

[55] P. Geuzaine, G. Brown, C. Harris & C. Farhat, Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions, *AIAA Journal*, vol. 41, no. 3, pp. 363–371, 2003.

[56] M. Gluck, M. Breuer, F. Durst, A. Halfmann & E. Rank, Computation of wind-induced vibrations of flexible shells and membranous structures, *Journal of Fluids and Structures*, vol. 17, no. 5, pp. 739–765, 2003.

[57] A. Gravouil & A. Combescure, Multi-time-step explicit-implicit method for non-linear structural dynamics, *International Journal for Numerical Methods in Engineering*, vol. 50, no. 1, pp. 199–225, 2001.

[58] L. Greengard & J. Strain, The fast gauss transform, *SIAM Journal of Scientific Computing*, vol. 1, pp. 79–94, 1991.

[59] W. Gropp, E. Lusk & R. Thakur, *Using MPI-2: Advanced Features of the Message Passing Interface*, The MIT Press, 1999.

[60] H. Guillard & C. Farhat, On the significance of the geometric conservation law for flow computations on moving meshes, *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 1467–1482, 2000.

[61] P. M. Hartwich & S. Agrawal, Method for perturbing multiblock patched grids in aeroelastic and design optimization applications, AIAA-97-2038, 1997.

[62] M. Heil, An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems, *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 1–2, pp. 1–23, 2004.

[63] M. W. Heinstein & T. A. Laursen, A three dimensional surface-to-surface projection algorithm for non-coincident domains, *Communications in Numerical Methods in Engineering*, vol. 19, pp. 421–432, 2003.

[64] B. T. Helenbrook, Mesh deformation using the biharmonic operator, *International Journal for Numerical Methods in Engineering*, vol. 56, pp. 1007–1021, 2003.

[65] S.-Y. Hsu & C.-L. Chang, Mesh deformation based on fully stressed design: The method and 2d examples, *International Journal of Numerical Methods in Engineering*, vol. 72, no. 5, pp. 606–629, 2007.

[66] T. Hughes, *The Finite Element Method, Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall Int., 1987.

[67] B. Irons & R. Tuck, A version of the Aitken accelerator for computer iteration, *International Journal of Numerical Methods in Engineering*, vol. 1, no. 3, pp. 275–277, 1969.

[68] S. Jakobsson & O. Amoignon, Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization, *Computers & Fluids*, vol. 36, no. 6, pp. 1119–1136, 2007.

[69] A. Jameson, W. Schmidt & E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes, Tech. Rep. AIAA-81-1259, 1981.

[70] H. Kanchi & A. Masud, A 3D adaptive mesh moving scheme, *International Journal for Numerical Methods in Fluids*, vol. 54, no. 6–8, pp. 923–944, 2007.

[71] E. J. Kansa, Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – I: Surface approximations and partial derivative estimates, *Computers & Mathematics with Applications*, vol. 19, pp. 127–145, 1990.

[72] E. J. Kansa, Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – II: Solutions to parabolic, hyperbolic and elliptic partial differential equations, *Computers & Mathematics with Applications*, vol. 19, pp. 147–161, 1990.

[73] B. B. T. Kee, G. R. Liua, G. Y. Zhanga & C. Luc, A residual based error estimator using radial basis functions, *Finite Elements in Analysis and Design*, vol. 44, no. 9–10, pp. 139–181, 2008.

[74] P. M. Knupp, Algebraic mesh quality metrics for unstructured initial meshes, *Finite Elements in Analysis and Design*, vol. 39, pp. 217–241, 2003.

[75] B. Koobus & C. Farhat, Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes, *Computer Methods in Applied Mechanics and Engineering*, vol. 170, pp. 103–129, 1999.

[76] K. Kovalev, *Unstructured Hexahedral Non-conformal Mesh Generation*, Ph.D. thesis, Vrije Universiteit Brussel, 2005.

[77] G. A. M. van Kuik & J. W. M. Dekker, The flexhat program, technology development and testing of flexible rotor systems with fast passive pitch control, *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 39, pp. 435–448, 1992.

[78] T. A. Laursen & M. W. Heinstein, Consistent mesh tying methods for topologically distinct discretized surfaces in non-linear solid mechanics, *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 1197–1242, 2003.

[79] X. S. Li & J. W. Demmel, SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems, *ACM Transactions on Mathematical Software*, vol. 29, no. 2, pp. 110–140, 2003.

[80] E. J. Lingen, *FLECS, a Flexible Coupling Shell*, 2007.

[81] O. E. Livne & G. B. Wright, Fast multilevel evaluation of 1D piecewise smooth radial basis function expansions, SIAM Conference on Geometric Design and Computing, 2005.

[82] O. E. Livne & G. B. Wright, Fast multilevel evaluation of smooth radial basis function expansions, *Electronic Transactions on Numerical Analysis*, vol. 23, pp. 263–287, 2006.

[83] R. Löhner, C. Yang, J. Cebral, J. D. Baum, H. Luo, D. Pelessone & C. Charman, Fluid-structure interaction using a loose coupling algorithm and adaptive unstructured grids, in *Computational Fluid Dynamics Review* (edited by M. Hafez & K. Oshima), John Wiley, 1995.

[84] D. Lynch & K. ONeill, Elastic grid deformation for moving boundary problems in two space dimensions, in *Finite Elements in Water Resources* (edited by S. Wang), 1980.

[85] N. Maman & C. Farhat, Matching fluid and structure meshes for aeroelastic computations: A parallel approach, *Computers and Structures*, vol. 54, no. 4, pp. 779–785, 1995.

[86] P. O. Marklund & L. Nilsson, Simulation of airbag inflation processes using a coupled fluid structure approach, *Computational Mechanics*, vol. 29, no. 4–5, pp. 289–297, 2002.

[87] C. Michler, E. H. van Brummelen & R. de Borst, An interface Newton-Krylov solver for fluid-structure interaction, *International Journal for Numerical Methods in Fluids*, vol. 47, no. 10–11, pp. 1189–1195, 2005.

[88] C. Michler, S. J. Hulshoff, E. H. van Brummelen, S. J. Hulshoff & R. de Borst, A monolithic approach to fluid-structure interaction, *Computers & Fluids*, vol. 33, no. 5–6, pp. 839–848, 2004.

[89] D. P. Mok, W. A. Wall & E. Ramm, Accelerated iterative substructure schemes for instationary fluid-structure interaction, in *First MIT Conference on Computational Fluid and Solid Mechanics*, pp. 1325–1328, 2001.

[90] G. Morgenthal & A. McRobie, A comparative study of numerical methods for fluid structure interaction analysis in long-span bridge design, *Wind and Structures*, vol. 5, no. 2–4, pp. 101–114, 2002.

[91] T. Neckel, FSIce for fluid-structure interactions on cartesian grids, in *2nd GACM Colloquium on Computational Mechanics*, Fakultät für Informatik, Technische Universität München, München, Germany, 2007.

[92] M. Nool, E. J. Lingen, A. de Boer & H. Bijl, Flecs, a flexible coupling shell application to fluid-structure interaction, in *PARA'06: State-of-the-Art in Scientific and Parallel Computing*, Umeøa, Sweden, 2006.

[93] S. Piperno, , C. Farhat & B. Larrouturou, Partitioned procedures for the transient solution of coupled aroelastic problems part I: Model problem, theory and two-dimensional application, vol. 124, pp. 79–112, 1995.

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

[95] T. Rendall & C. Allen, Unified cfd-csd interpolation and mesh motion using radial basis functions, *International Journal for Numerical Methods in Engineering*, vol. 74, no. 10, pp. 1519–1559, 2008.

[96] T. C. S. Rendall & C. B. Allen, Efficient mesh motion using radial basis functions with data reduction algorithms, AIAA 2008-305, 46th AIAA Aerospace Sciences Meeting, Jan. 2008.

[97] T. C. S. Rendall & C. B. Allen, Improved radial basis function fluid-structure coupling via efficient localised implementation, AIAA 2008-4058, 38th AIAA Fluid Dynamics Conference, June 2008.

[98] G. Roussos & B. J. C. Baxter, Rapid evaluation of radial basis functions, *Journal of Computational and Applied Mathematics*, vol. 180, pp. 51–70, 2005.

[99] S. Sathe, R. Benney, R. Charles, E. Doucette, J. Miletti, M. Senga, K. Stein & T. E. Tezduyar, Fluid-structure interaction modeling of complex parachute designs with the space-time finite element techniques, *Computers & Fluids*, vol. 36, no. 1, pp. 127–135, 2007.

[100] M. J. Smith, C. E. S. Cesnik & D. H. Hodges, Evaluation of some data transfer algorithms for noncontiguous meshes, *Journal of Aerospace Engineering*, vol. 13, no. 2, pp. 52–58, 2000.

[101] M. J. Smith, D. H. Hodges & C. E. S. Cesnik, Evaluation of computational algorithms suitable for fluid-structure interactions, *Journal of Aircraft*, vol. 37, no. 2, pp. 282–294, 2000.

[102] A. Soria & F. Casadei, Arbitrary lagrangian-eulerian multicomponent compressible flow with fluid-structure interaction, *International Journal for Numerical Methods in Fluids*, vol. 25, pp. 1263–1284, 1997.

[103] S. Spekreijse, B. Prananta & J. Kok, A simple, robust and fast algorithm to compute deformations of multi-block structured grids, Tech. rep., 2002.

[104] K. Stein, R. Benney, V. Kalro, T. E. Tezduyar, J. Leonard & M. Accorsi, Parachute fluid-structure interactions: 3-d computation, *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 373–386, 2000.

[105] K. Stein, R. Benney & T. E. Tezduyar, Mesh moving techniques for fluid-structure interactions with large displacements, *Journal of Applied Mechanics-Transactions of the ASME*, vol. 70, no. 1, pp. 58–63, 2003.

[106] F. K. Straub, A feasibility study of using smart materials for rotor control, *Smart Material Structures*, vol. 5, pp. 1–10, 1996.

[107] P. le Tallec & J. Mouro, Fluid structure interaction with large structural displacements, *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 3039–3067, 2001.

[108] P. Thévenza, T. Blu & M. Unser, Interpolation revisited, *IEEE Transactions on Medical Imaging*, vol. 19, no. 7, pp. 739–758, 2000.

[109] H. M. Tsai & A. S. F. Wong, Unsteady flow calculations with a parallel multiblock moving mesh algorithm, *AIAA Journal*, vol. 39, no. 6, pp. 1021–1029, 2001.

[110] S. Turek & J. Hron, Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow, in *Fluid-Structure Interaction: Modelling, Simulation, Optimisation* (edited by H.-J. Bungartz & M. Schäfer), Springer, 2006.

[111] S. Turek & M. Schäfer, Benchmark computations of laminar flow around cylinder, in *Flow Simulation with High-Performance Computers II* (edited by E. H. Hirschel), no. 52 in Notes on Numerical Fluid Mechanics, Vieweg, 1996.

[112] J. Vierendeels, lieve Lanoye, J. Degroote & P. Verdonck, Implicit coupling of partitioned fluid-structure interaction problems with reduced order models, *Computers and Structures*, vol. 85, no. 11–14, pp. 970–976, 2007.

[113] Z. J. Wang & A. J. Przekwas, Unsteady flow computation using moving grid with mesh enrichment, Tech. Rep. AIAA-94-0285, 1994.

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

[115] H. Wendland, Error estimates for interpolation by compactly supported radial basis functions of minimal degree, *Journal of Approximation Theory*, vol. 93, pp. 258–272, 1998.

[116] H. Wendland, On the smoothness of positive definite and radial functions, *Journal of Computational and Applied Mathematics*, vol. 101, pp. 177–188, 1999.

[117] J. White & M. Heil, Three-dimensional instabilities of liquid-lined elastic tubes: A thin-film fluid-structure interaction model, *Physics of Fluids*, vol. 17, no. 3, p. 031506, 2005.

[118] E. C. Yates jr., AGARD standard aeroelastic configurations for dynamic response. candidate configuration I.-Wing 445.6, Tech. Rep. Technical Memorandum 100492, 1987.

[119] D. Zeng & C. R. Ethier, A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains, *Finite Elements in Analysis and Design*, vol. 41, pp. 1118–11139, 2005.

[120] A. H. van Zuijlen, *Fluid-structure interaction simulations - Efficient higher order time integration of partitioned systems*, Ph.D. thesis, Delft University of Technology, 2006.

[121] A. H. van Zuijlen & H. Bijl, A higher-order time integration algorithm for the simulation of non-linear fluid-structure interaction on moving meshes, *Nonlinear Analysis-Theory Methods & Applications*, vol. 63, pp. 1597–1605, 2005.

[122] A. H. van Zuijlen & H. Bijl, Implicit and explicit higher-order time integration schemes for fluid-structure interaction computations, *International Journal of Multiscale Computational Engineering*, vol. 4, no. 2, pp. 255–263, 2006.

[123] A. H. van Zuijlen, A. de Boer & H. Bijl, Higher-order time integration through smooth mesh deformation for 3D fluid-structure interaction simulations, *Journal of Computational Physics*, vol. 224, pp. 414–430, 2007.

[124] A. H. van Zuijlen, S. Bosscher & H. Bijl, Two level algorithms for partitioned fluid-structure interaction computations, *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 8, pp. 1458–1470, 2006.

Appendix A

This appendix contains the entries of the matrices and vectors given in (4.14) in more detail, both for the inclusion of rotations in 2D and 3D, where $\phi_{ij} = \phi(||\mathbf{x}_{\mathrm{b}_i} - \mathbf{x}_{\mathrm{b}_j}||)$ and we use $N = n_{\mathrm{b}}$ for lay-out purposes.

Including rotational information in 2D

In 2D $A_{\phi,\mathrm{b}}$ is a symmetric $(3N \times 3N)$ matrix with

$$A_{11} = \left[\begin{array}{cc|cc|c|cc} \phi_{11} & 0 & \phi_{12} & 0 & \dots & \phi_{1N} & 0 \\ 0 & \phi_{11} & 0 & \phi_{12} & \dots & 0 & \phi_{1N} \\ \hline \phi_{21} & 0 & \phi_{22} & 0 & \dots & \phi_{2N} & 0 \\ 0 & \phi_{21} & 0 & \phi_{22} & \dots & 0 & \phi_{2N} \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \phi_{N1} & 0 & \phi_{N2} & 0 & \dots & \phi_{NN} & 0 \\ 0 & \phi_{N1} & 0 & \phi_{N2} & \dots & 0 & \phi_{NN} \end{array}\right],$$

$$A_{12} = \left[\begin{array}{c|c|c|c} -\partial_y\phi_{11} & -\partial_y\phi_{12} & \dots & -\partial_y\phi_{1N} \\ \partial_x\phi_{11} & \partial_x\phi_{12} & \dots & \partial_x\phi_{1N} \\ \hline -\partial_y\phi_{21} & -\partial_y\phi_{22} & \dots & -\partial_y\phi_{2N} \\ \partial_x\phi_{21} & \partial_x\phi_{22} & \dots & \partial_x\phi_{2N} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline -\partial_y\phi_{N1} & -\partial_y\phi_{N2} & \dots & -\partial_y\phi_{NN} \\ \partial_x\phi_{N1} & \partial_x\phi_{N2} & \dots & \partial_x\phi_{NN} \end{array}\right],$$

and

$$A_{22} = \left[\begin{array}{c|c|c|c} \nabla^2\phi_{11} & \nabla^2\phi_{12} & \dots & \nabla^2\phi_{1N} \\ \hline \nabla^2\phi_{21} & \nabla^2\phi_{22} & \dots & \nabla^2\phi_{2N} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \nabla^2\phi_{N1} & \nabla^2\phi_{N2} & \dots & \nabla^2\phi_{NN} \end{array}\right],$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

P_{b} is an $(3N \times 6)$ matrix and the vector $\mathbf{r}$ contains the displacements and rotations in the

following way:

$$P_{\mathrm{b}} = \left[\begin{array}{cc|cc|cc} 1 & 0 & x_{\mathrm{b}_1} & 0 & y_{\mathrm{b}_1} & 0 \\ 0 & 1 & 0 & x_{\mathrm{b}_1} & 0 & y_{\mathrm{b}_1} \\ \hline 1 & 0 & x_{\mathrm{b}_2} & 0 & y_{\mathrm{b}_2} & 0 \\ 0 & 1 & 0 & x_{\mathrm{b}_2} & 0 & y_{\mathrm{b}_2} \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 1 & 0 & x_{\mathrm{b}_N} & 0 & y_{\mathrm{b}_N} & 0 \\ 0 & 1 & 0 & x_{\mathrm{b}_N} & 0 & y_{\mathrm{b}_N} \\ \hline 0 & 0 & 0 & 1 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 & -1 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 1 & -1 & 0 \end{array}\right], \qquad \mathbf{r} = \left[\begin{array}{c} d_x^1 \\ d_y^1 \\ \hline d_x^2 \\ d_y^2 \\ \hline \vdots \\ \hline d_x^N \\ d_y^N \\ \hline 2\theta^1 \\ \hline 2\theta^2 \\ \hline \vdots \\ \hline 2\theta^N \end{array}\right]$$

Including rotational information in 3D

$A_{\phi,\mathrm{b}}$ is now a $(6N \times 6N)$ matrix with

$$A_{11} = \left[\begin{array}{ccc|ccc|c|ccc} \phi_{11} & 0 & 0 & \phi_{12} & 0 & 0 & \dots & \phi_{1N} & 0 & 0 \\ 0 & \phi_{11} & 0 & 0 & \phi_{12} & 0 & \dots & 0 & \phi_{1N} & 0 \\ 0 & 0 & \phi_{11} & 0 & 0 & \phi_{12} & \dots & 0 & 0 & \phi_{1N} \\ \hline \phi_{21} & 0 & 0 & \phi_{22} & 0 & 0 & \dots & \phi_{2N} & 0 & 0 \\ 0 & \phi_{21} & 0 & 0 & \phi_{22} & 0 & \dots & 0 & \phi_{2N} & 0 \\ 0 & 0 & \phi_{21} & 0 & 0 & \phi_{22} & \dots & 0 & 0 & \phi_{2N} \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline \phi_{N1} & 0 & 0 & \phi_{N2} & 0 & 0 & \dots & \phi_{NN} & 0 & 0 \\ 0 & \phi_{N1} & 0 & 0 & \phi_{N2} & 0 & \dots & 0 & \phi_{NN} & 0 \\ 0 & 0 & \phi_{N1} & 0 & 0 & \phi_{N2} & \dots & 0 & 0 & \phi_{NN} \end{array}\right],$$

$$A_{22} = \left[\begin{array}{c|c|c|c} \partial\Phi_{11} & \partial\Phi_{12} & \dots & \partial\Phi_{1N} \\ \hline \partial\Phi_{21} & \partial\Phi_{22} & \dots & \partial\Phi_{2N} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \partial\Phi_{N1} & \partial\Phi_{N2} & \dots & \partial\Phi_{NN} \end{array}\right]$$

and

$$A_{22} = \left[\begin{array}{c|c|c|c} \partial^2\Phi_{11} & \partial^2\Phi_{12} & \dots & \partial^2\Phi_{1N} \\ \hline \partial^2\Phi_{21} & \partial^2\Phi_{22} & \dots & \partial^2\Phi_{2N} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \partial^2\Phi_{N1} & \partial^2\Phi_{N2} & \dots & \partial^2\Phi_{NN} \end{array}\right],$$

with

$$\partial\Phi_{ij} = \begin{bmatrix} 0 & \partial_z\phi_{ij} & -\partial_y\phi_{ij} \\ -\partial_z\phi_{ij} & 0 & \partial_x\phi_{ij} \\ \partial_y\phi_{ij} & -\partial_x\phi_{ij} & 0 \end{bmatrix},$$

and

$$\partial^2\Phi_{ij} = \begin{bmatrix} \partial_y^2\phi_{ij} + \partial_z^2\phi_{ij} & -\partial_{xy}\phi_{ij} & -\partial_{xz}\phi_{ij} \\ -\partial_{xy}\phi_{ij} & \partial_x^2\phi_{ij} + \partial_z^2\phi_{ij} & -\partial_{yz}\phi_{ij} \\ -\partial_{xz}\phi_{ij} & -\partial_{yz}\phi_{ij} & \partial_x^2\phi_{ij} + \partial_y^2\phi_{ij} \end{bmatrix}.$$

P_b is an $(6N \times 12)$ matrix and the vector $\mathbf{r}$ contains the displacements and rotations in the following way:

$$P_\mathrm{b} = \left[\begin{array}{ccc|ccc|ccc|ccc}
1 & 0 & 0 & x_{\mathrm{b}_1} & 0 & 0 & y_{\mathrm{b}_1} & 0 & 0 & z_{\mathrm{b}_1} & 0 & 0 \\
0 & 1 & 0 & 0 & x_{\mathrm{b}_1} & 0 & 0 & y_{\mathrm{b}_1} & 0 & 0 & z_{\mathrm{b}_1} & 0 \\
0 & 0 & 1 & 0 & 0 & x_{\mathrm{b}_1} & 0 & 0 & y_{\mathrm{b}_1} & 0 & 0 & z_{\mathrm{b}_1} \\
\hline
1 & 0 & 0 & x_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} & 0 & 0 \\
0 & 1 & 0 & 0 & x_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} & 0 \\
0 & 0 & 1 & 0 & 0 & x_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} & 0 & 0 & y_{\mathrm{b}_2} \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
1 & 0 & 0 & x_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} & 0 & 0 \\
0 & 1 & 0 & 0 & x_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} & 0 \\
0 & 0 & 1 & 0 & 0 & x_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} & 0 & 0 & y_{\mathrm{b}_N} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0
\end{array}\right], \quad \mathbf{r} = \left[\begin{array}{c}
d_x^1 \\ d_y^1 \\ d_z^1 \\ \hline
d_x^2 \\ d_y^2 \\ d_z^1 \\ \hline
\vdots \\ \hline
d_x^N \\ d_y^N \\ d_z^N \\ \hline
2\theta_x^1 \\ 2\theta_y^1 \\ 2\theta_z^1 \\ \hline
2\theta_x^2 \\ 2\theta_y^2 \\ 2\theta_z^2 \\ \hline
\vdots \\ \hline
2\theta_x^N \\ 2\theta_y^N \\ 2\theta_z^N
\end{array}\right]$$

Dankwoord

Veel mensen hebben bijgedragen aan het tot stand komen van dit proefschrift. Ten eerste wil ik mijn promotor Hester Bijl bedanken voor haar onafgebroken steun en niet-aflatend enthousiasme gedurende de laatste vier jaar en mijn collega en kamergenoot Sander van Zuijlen voor de vele discussies en ondersteuning. Ik wil ook mijn waardering uitspreken voor het werk van Martijn van der Schoot, Erik Jan Lingen en Margreet Nool, want zij zijn hiermee verantwoordelijk voor een deel van wat in dit proefschrift beschreven staat.

Verder zou ik dit proefschrift niet hebben kunnen voltooien zonder de continue steun van mijn familie en Martijn in het bijzonder. Als laatste wil ik Elske bedanken voor het verrijken van ons leven gedurende het laatste jaar van mijn onderzoek en hopelijk blijft ze dat nog vele jaren volhouden.

Aukje Spruyt-de Boer
Zoetermeer, November 2008

Curriculum Vitae

28 August 1980	Born in Allingawier
1992 – 1998	Marne College, Bolsward Graduated in seven core and one basic subject
Sept. 1998 – Aug. 2003	Twente University, Enschede Masters degree in Applied Mathematics Chair: Numerical Analysis and Computational Mechanics Graduated cum laude
Feb. 2004 – June 2008	Delft University of Technology, Delft Ph.D.-student at the Aerodynamics group of the Faculty of Aerospace Engineering