UiO : Department of Mathematics
University of Oslo

Development of a Verified and Validated Computational Framework for Fluid-Structure Interaction: Investigating Lifting Operators and Numerical Stability

Sebastian Gjertsen

Master's Thesis, Spring 2017

This master's thesis is submitted under the master's programme *Computational Science and Engineering*, with programme option *Mechanics*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Acknowlegments

First of I want to thank by brilliant supervisors Dr.Kristian Valen-Sendstad, Dr.Mikael Mortensen and Aslak Bergersen. Kristian your guidance and humor as made my time working on my thesis very pleasurable. Aslak, your tough love has kept me on my tippy toes at all times, and your problem solving abilities are of the absolute highest caliber. And to Mikael who has always kept his door open and answered my at times not so intelligent questions.

I also want to thank my good friend and study buddy Andreas Slyngstad. I have much appreciated the many fun times and discussions I have had with Andy during the 5 years of study and in the last year at Simula working on a related problem, I have gained much insight from him.

I want to thank my family for letting me become whatever I want, as long as it is an engineer. And also my wonderful girlfriend Marlene, for her uncanny love and support.

Lastly I want to thank my dog Sara, for always waging her tail, making me happy.

Abstract

Computational models of fluids, structures, and the interaction between them shows good promise in science and engineering, with nearly infinitely many applications. However, fluid-structure interaction (FSI) is poorly understood from a mathematical and computational stand point. The goal of this thesis was to develop a virtual framework for computational FSI problems using a monolithic scheme. The mathematics and physics that govern fluid and structures was introduces, and the necessary conditions to model FSI problems. The θ-scheme was implemented in FEniCS because of its ability to uphold stability for long time FSI simulations. The code has been verified in parts using MMS.

The computational FSI solver was validated against the benchmark proposed by Hron and Turek 2010 [11]. Both the fluid and structure were validated separately, before adressing the FSI problem. Data was compared with contributions made by leading scientists in the field, and shown good agreement.

In the fifth chapter of this thesis, the crucial choice of lifting is discussed for FSI problems with moderate to large deformations, and the need for long-term numerical stability schemes.

Contents

Chapter 1

Introduction to Fluid-Structure Interaction

The interaction between fluids and structures can be observed all around us in nature. Examples of fluid-structure interaction include flags waving in the wind, windmills, and inhalation of air into the lungs. It is rather intuitive that fluids and structures must exert mutual force on each other and that the fluid and structure both can have dominant and passive properties. In the example of a flag waving in the wind, it is the forces from the flowing air that are dominant, whereas in the example of inhalation, the structure (diaphragm) is dominating.

Understanding and modeling fluid-structure interaction (FSI) can greatly assist in design of structures such as windmill and aircraft wings. A famous example of design flaw is the collapse of the Tacoma Narrows Bridge in 1940 [1], only two months after being opened, see Figure 1.1. The bridge was literally shaken apart due to strong winds (64 km/h) interacting with the structure, making it resonate. No human lives were lost in the collapse, but a Cocker Spaniel named Tubby left behind in a car was not that lucky and lost its life in the bridge collapse.

FSI has matured and is now routinely used to model and design the motion and wakes of windmills. Since there is a big difference in density between fluid and structure $\frac{\rho_f}{\rho_s} << 1$, the structural deformations are small and the interaction is “easy” to model. However, modeling arterial FSI deems more challenging as the density of the fluid (blood) and the structure (artery) are similar, resulting in large deformations of the arteries. Large deformations

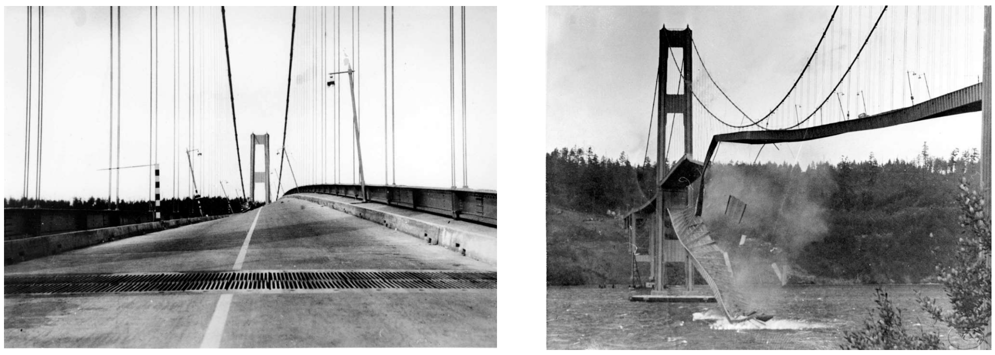


Figure 1.1: Left: Tacoma Narrows bridge still standing with large deformations, Right: Tacoma Narrows bridge collapsed

are challenging to model and require hyperelastic constitutive laws, and energy stable numerical schemes.

A scientific branch of fluid mechanics is Computational Fluid Dynamics, where computers and numerical algorithms are used to solve fluid problems. Similarly, a name used for solving fluid-structure interaction problems is simply "FSI". However, it should be emphasized that it is actually *Computational Fluid Structure Interaction* (CFSI) that will be addressed in this thesis. The same name applies to the word "Structure" where the actual problem to be solved belongs to the scientific branch of solid mechanics.

The goal of this master thesis is to develop a computational framework to solve FSI problems arising in biomechincs, namely large deformations. The effects of different numerical schemes and approaches will be investigated to maintain acceptable accuracy.

Chapter 2

Continuum Mechanics in Different Frames of Reference

Matter is made up of small building blocks called atoms, that are separated by vacuum, and the fundamental laws at this small scale are described using quantum mechanics [3]. The associated characteristic length scale far smaller than most things of interest to describe and model at a macroscopic level. Since it has been observed that the characteristic behavior is statistically uniform, it is reasonable to assume the matter can be mathematically described, and modeled, as a continuum. This allows for a mathematical description at the macroscopic level using Newtonian physical laws to model fluids and solids. The Newtonian laws are generally expressed in one out of two frames of reference, Lagrangian or Eulerian, depending on the physical problem, as illustrated in Figure 2

To exemplify the differences between these frameworks, one can imagine a river running down a mountain. In the Eulerian framework we observe an object following the flow, standing besides the river. We are not necessarily interested in each fluid particle or the history of it, but only how the fluid acts as a whole flowing down the river. The Eulerian framework is especially advantageous to describe fluid mechanics.

A Lagrangian description of solid mechanics is particularly beneficial, as one is generally interested in where the solid particles are in relation to each other and the particles initial spatial state. This is best exemplified with a deflecting beam attached to a wall. The more force applied to the beam the more it will deflect, relative to its historical stress free configuration. As described later this frame of reference is generally beneficial to describe stress and strain for solid mechanical problems.

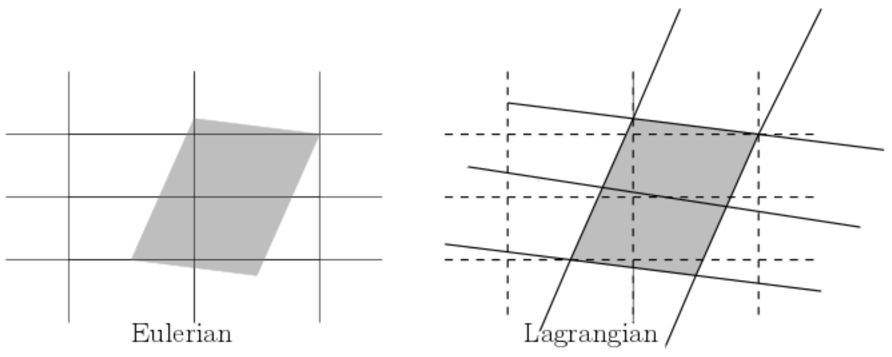


Figure 2.1: Comparison between the Eulerian and the Lagrangian description, the lines represent the grid, the dashed line represents the initial grid and the gray represents the matter.

The goal of this chapter is to briefly introduce conservation of mass and momentum for a fluid and a solid, respectively together with the respective boundary conditions. A derivation and a more detailed description of the Lagrangian framework and the stress and strain relations are covered in the appendix A1.

2.1 Conservation of Mass and Momentum for Solid

Assuming matter to behave like a continuum, fundamental physical laws like conservation of mass and conservation of momentum can be applied to derive a differential equation describing the motions of a solid. Information about the particular material of a solid is described through constitutive relations. The differential solid equation will be stated in the Lagrangian reference system [10], in the solid domain $\mathcal{S}$ as:

$$\rho_s \frac{\partial \mathbf{d}^2}{\partial t^2} = \nabla \cdot (P) + \rho_s f \quad in \ \mathcal{S} \tag{2.1}$$

written in terms of the deformation $\mathbf{d}$, of the solid. P is the first Piola-Kirchhoff stress tensor, its derivation is included in the appendix A1 for the sake of completeness. Body forces are denoted as f, and are forces that originate outside the body and act on the mass of the body e.g. gravitational force. ρ_s is the solid density, and $\frac{\partial}{\partial t^2}$ is the second time derivative.

2.2 Conservation of Mass and Momentum for Fluid

The differential equations describing the velocity and pressure in a fluid are called the Navier-Stokes (N-S) equations. The N-S equations are derived, like the solid equation, from principles of mass and momentum conservation, assuming fluid to act as a continuum. The fluid equations are stated in an Eulerian framework. The N-S equations are written in the fluid time domain $\mathcal{F}(t)$ as an incompressible fluid:

$$\rho_f (\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = \nabla \cdot \sigma_f + \rho_f f \quad in \ \mathcal{F}(t) \tag{2.2}$$

$$\nabla \cdot \mathbf{u} = 0 \quad in \ \mathcal{F}(t) \tag{2.3}$$

where $\mathbf{u}$ is the fluid velocity, p is the fluid pressure, ρ_f stands for density. f is body force and σ_f is the Cauchy stress tensor, $\sigma_f = \mu_f(\nabla \mathbf{u} + \nabla \mathbf{u}^T) - pI$, setting μ_f to be constant hence denoting a Newtonian fluid. I denotes the identity matrix.

There does not yet exist an analytical solutions to the Navier-Stokes equations for every fluid problem. Analytical solutions can only be found for fluid problems with certain boundary conditions and geometries using the N-S equations [29]. Actually there is a prize set out by the Clay Mathematics Institute of one million dollars to whomever can show the existence and smoothness of Navier-Stokes equations [6], as a part of their millennium problems. Nonetheless this does not stop us from discretizing and solving N-S numerically. One difficulty in the Navier-Stokes equations is the nonlinearity appearing in the convection term on the left hand side. This non-linearity can be handled using Newtons method, or Picard iterations. Another difficulty is finding a suitable equation to solve for the pressure field [2]. As there is no natural pressure update.

2.3 Fluid and Structure Boundary conditions

In order to obtain a unique solution to the fluid and solid equations, we need to specify a computational domain and impose boundary conditions. The fluid flow and the solid moves within the boundaries noted as $\partial\mathcal{F}$ and $\partial\mathcal{S}$, respectively.

Dirichlet boundary conditions, often referred to as essential ones, are defined on the boundaries $\partial\mathcal{F}_D$ and $\partial\mathcal{S}_D$. Dirichlet boundary conditions can be fixed or time varying values, such as zero at the fluid boundary for a "no slip" condition, or a Womersley profile [9] at the inlet of a pipe. For a problem to be well posed we need also to prescribe initial conditions. The Dirichlet boundary conditions are defined for $\mathbf{u}$ and p as :

$$\mathbf{u} = u_0 \text{ on } \partial\mathcal{F}_D \tag{2.4}$$

$$p = p_0 \text{ on } \partial\mathcal{F}_D \tag{2.5}$$

$$\mathbf{d} = d_0 \text{ on } \partial\mathcal{S}_D \tag{2.6}$$

$$\mathbf{w}(\mathbf{X},t)_0 = \frac{\partial\mathbf{d}(t=0)}{\partial t} \text{ on } \partial\mathcal{S}_D \tag{2.7}$$

In addition, there are Neumann boundary conditions, often referred to as natural, which states a specific value of the derivative of a solution at the boundary. More specifically, $\partial\mathcal{F}_N$ and $\partial\mathcal{S}_N$. One can also control eventual forces on the Neumann boundary, to possibly equal an external force $\mathbf{f}$:

$$\sigma \cdot \mathbf{n} = f \text{ on } \partial\mathcal{F}_N \tag{2.8}$$

$$P \cdot \mathbf{n} = f \text{ on } \partial\mathcal{S}_N \tag{2.9}$$

For the sake of completeness, it should be noticed that there exists other boundary conditions as well, for instance the Robin boundary conditions, which are the Dirichlet and Neumann conditions combined.

Chapter 3

Fluid-Structure Interaction Problem Formulation

The following chapter will be devoted to introducing the full FSI problem mathematically. The equations and interface conditions will be introduced in a strong and weak form. From the weak form both equations will be discretized into a scheme which will be used in the FSI solver.

When describing the FSI problems the domain is split into three: fluid, structure, and interface. The fluid and structure domains are separated, and different constitutive equations are solved in each domain. The spatial points in which fluid and structure join is called the interface. The treatment of the interface separates the two most commonly used methods for solving FSI problems [12]. The first method is called fully Eulerian. In a fully Eulerian framework, both the fluid and structure equations are defined and solved in a purely Eulerian description. The interface in a fully Eulerian framework is tracked across a fixed domain [27], which is a difficult task. The fully Eulerian description is suited for fluid problems but is problematic for structure problems because of the interface tracking.

The second approach is the *Arbitrary Lagrangian Eulerian* (ALE). The ALE method entails formulating the fluid equations in a type of Eulerian framework and the solid in a Lagrangian framework. The entire domain itself moves with the structural displacements and the fluid moves through these points. In the ALE framework we get the best of both worlds, in that fluid and solid are described in their most commonly stated mathematical forms.

The structure equation will remain as previously stated (2.1), and in the fluid equation we need to take into account the change in convection arising from the mesh deformation.

Dealing with the movement of the domain is performed in two ways. One way is to move the domain itself in relation to the structural displacements, and use this new domain to calculate the equations for every iteration. This requires a specific function to move the mesh between each timestep. This process can be time consuming as problems get large. The structural deformation history can also give rise to problems as the points on the domain have changed location.

The second approach to ALE, which will be used in this thesis, is to calculate from reference domain. When solving equations from a reference domain we solve the equations on an initial, stress free domain, and use a series of mappings to account for the movements of the current time domain. It is the displacements in the domain that determines the value of the mappings between frames of reference. The solid equation is already stated in a Lagrangian formulation and does not need any mappings. It is the fluid velocities and fluid pressure that needs to be mapped from the reference domain into the time domain in which they are stated. Since the reference frame method does not need a function to move the mesh between each time iteration, it can be less time consuming. The interface is also located in the same position, making the interface easy to track. With the domain always remaining the same variational coupling of forces is easier when computing from a reference domain.

There are generally two different approaches when discretizing an FSI scheme. The first is a partitioned approach where fluid and structure are solved sequentially. The partitioned approach is appealing in that we have a wealth of knowledge and legacy code that can be used to solve each of these kinds of problems in an efficient manner. The difficulty however is dealing with the interface. There are kinematic and dynamic conditions that has to be fulfilled in FSI, and the coupling of these conditions is where problems arise. Artificial added mass can appear when a partitioned approach is used, arising from poorly coupling between fluid and solid velocities (this is discussed further in discussion chapter). However, in this thesis the monolithic approach is used. In the monolithic approach all of the equations are solved simultaneously. The monolithic approach has the advantage of offering numerical stability for problems with strong added-mass effects [12], and are fully coupled. The disadvantage over the partitioned approach is that we loose flexibility when solving many equations simultaneously, and the computing matrices can quickly become large, hence computationally costly and more memory demanding. The overall ease of implementation and numerical stability makes monolithic the preferred choice in this thesis.

The following chapter starts by introducing the mappings needed to change between current and reference domain. Lastly, the equations will be discretized following the notation and ideas from [18].

3.1 Mapping Between Different Frames of Reference

Figure 3.1 depicts a simple Fluid-Structure Interaction domain. The fluid is surrounded by elastic walls, like for instance a blood vessel. $\hat{\mathcal{S}}$ and $\hat{\mathcal{F}}$ denotes the solid and fluid reference domain respectively. $\hat{\Sigma}$ denotes the interface in the reference domain. $\partial\hat{\mathcal{F}}_{in}$ and $\partial\hat{\mathcal{F}}_{out}$ denotes the fluid in and outflow. $\partial\hat{\mathcal{S}}$ is the outer solid wall. The reference domain is mapped using χ^s and χ^f to the time domain denoted as $\mathcal{S}$ and $\mathcal{F}$. While the interface in the time domain is denoted as Σ

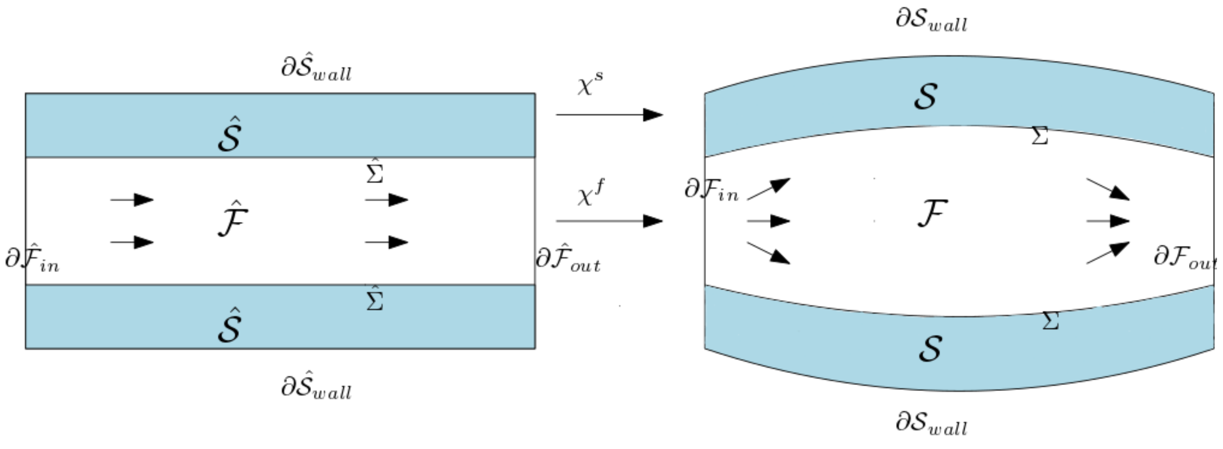

$\partial\hat{\mathcal{S}}_{wall}$
$\hat{\mathcal{S}}$
$\hat{\Sigma}$
$\hat{\mathcal{F}}$
$\partial\hat{\mathcal{F}}_{in}$
$\partial\hat{\mathcal{F}}_{out}$
$\hat{\Sigma}$
$\hat{\mathcal{S}}$
$\partial\hat{\mathcal{S}}_{wall}$
χ^s
χ^f
$\partial\mathcal{S}_{wall}$
$\mathcal{S}$
Σ
$\partial\mathcal{F}_{in}$
$\mathcal{F}$
$\partial\mathcal{F}_{out}$
Σ
$\mathcal{S}$
$\partial\mathcal{S}_{wall}$

Figure 3.1: Illustration of domain mapping

Let $\hat{\mathcal{V}}$ be a reference domain and $\mathcal{V}(t)$ be the current time domain. Then using the deformation gradient (A.5) and the determinant of the deformation gradient named the Jacobian (A.6), to define a mapping between the volumes from the current to the reference configurations.

The Jacobian is used to map the domain from the current to the reference domain shown in equation 3.1, and the gradients acting on a vector **u** will be mapped with the deformation gradient, shown in equation 3.2. Lastly the divergence of a vector **u** will be mapped in a slightly different manner, shown in equation 3.3.

$$\int_{\mathcal{V}(t)} 1 dx = \int_{\hat{\mathcal{V}}} J dx \tag{3.1}$$

$$\int_{\mathcal{V}(t)} \nabla \mathbf{u} dx = \int_{\hat{\mathcal{V}}} J \nabla \mathbf{u} F^{-1} dx \tag{3.2}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \mathbf{u} dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (J F^{-1} \mathbf{u}) dx \tag{3.3}$$

3.2 Governing Equations for Fluid-Structure Interaction

The following section formulates the fluid and solid equations in the ALE description. The solid equation will remain in its Lagrangian description but the fluid equations will be changed in terms of the convective term and mapping between configurations. The derivatives in the different configurations are shown in equations 3.4-3.6, to understand the need to change the the convective term in the fluid equation when stated in an ALE description [31].

3.2.1 Derivatives in Different Frameworks

In the Lagrangian framework the total and partial derivatives have the following relation:

$$D_t f(x,t) = \partial_t f(x,t) \tag{3.4}$$

The total and partial derivatives have the following relation in the Eulerian framework, where **u** is the convecting velocity:

$$D_t f(x,t) = \mathbf{u} \cdot \nabla f + \partial_t f(x,t) \tag{3.5}$$

whilst in the ALE framework this concept is extended to take into account the motion of the domain:

$$D_t f(x,t) = \mathbf{w} \cdot \nabla f + \partial_t f(x,t) \tag{3.6}$$

Equation 3.6 shows that in the Lagrangian framework $\mathbf{w}$ is zero and for the Eulerian framework $\mathbf{w} = \mathbf{u}$.

3.2.2 Solid Equation

Let $\Omega \in \hat{\mathcal{S}} \cup \hat{\mathcal{F}}$ be a global domain that consists of the fluid, structure, and the interface. The interface is defined as: $\hat{\Sigma} \in \hat{\mathcal{S}} \cap \hat{\mathcal{F}}$. Let $\mathbf{u}$ be a global velocity function that describes the fluid velocity in the fluid domain and the structure velocity in the structure domain, $\mathbf{u} = \mathbf{u_s} \cup \mathbf{u_f} \in \Omega$. The deformation function is also defined as a global function $\mathbf{d} = \mathbf{d_s} \cup \mathbf{d_f} \in \Omega$. Using global velocity and displacement functions adds the feature of the velocity and displacement being continuous across the entire domain. This is an important feature which will be apparent once the interface conditions are stated.

The solid equation will be written in terms of the solid velocity $\mathbf{u_s}$, in contrast to equation (2.1) which was defined with respect to the deformation $\mathbf{d}$. Defining the solid equation w.r.t to the deformation $\mathbf{d}$ can de done since the velocity is defined as the partial derivative of the deformation: $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$ The solid equation takes the following form in the Lagrangian formulation:

$$\rho_s \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot (P) + \rho_s f \quad in \quad \hat{\mathcal{S}} \tag{3.7}$$

3.2.3 Fluid Equations

In the ALE description the fluid domain is moving, giving the need to redefine the velocity in the convective term in (2.2) to account for the moving domain:

$$\mathbf{u} \cdot \nabla \mathbf{u} \rightarrow (\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) \cdot \nabla \mathbf{u} \tag{3.8}$$

Where $\mathbf{d}$ is the deformation in the fluid domain. The domain velocity $\mathbf{w} = \frac{\partial \mathbf{d}}{\partial t}$ is defined w.r.t deformation as the partial time derivative.

Applying the mappings from (3.1)-(3.3) and including the mesh velocity in the convecting velocity as shown in (3.6), we end up with the fluid equations mapped from the time domain to the reference domain. These are shown in 3.9-3.13, split up into a transient part, convection part, incompressible part, and the stress part:

$$\int_{\mathcal{V}(t)} \rho_f \frac{\partial \mathbf{u}}{\partial t} = \int_{\hat{\mathcal{V}}} \rho_f J \frac{\partial \mathbf{u}}{\partial t} dx \tag{3.9}$$

$$\int_{\mathcal{V}(t)} \nabla \mathbf{u}(\mathbf{u} - \frac{\partial d}{\partial t}) dx = \int_{J\hat{\mathcal{V}}} (\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial d}{\partial t}) dx \tag{3.10}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \mathbf{u} dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (J F^{-1} \mathbf{u}) dx \tag{3.11}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \sigma_f dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (J F^{-1} \hat{\sigma_f}) dx \tag{3.12}$$

$$\hat{\sigma_f} = -pI + \mu(\nabla \mathbf{u} F^{-1} + F^{-T} \mathbf{u}^T) \tag{3.13}$$

Assembling all these terms together gives strong form the fluid equations from a reference frame:

$$\rho_f J \big(\frac{\partial \mathbf{u}}{\partial t} dx + (\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) \big) = \nabla \cdot (J F^{-1} \hat{\sigma_f}) + J \rho_f f \tag{3.14}$$

$$\nabla \cdot (J F^{-1} \mathbf{u}) = 0 \tag{3.15}$$

3.3 Interface Conditions at the Boundary

The fluid‘s forces on the walls causes deformation in the solid domain and vice versa. The interface is where the energies between solid and fluid are transferred and we therefore need conditions on the interface.

The three interface conditions comes from simple physical properties and consist of [18]:

- *Kinematic condition*: $\mathbf{u}_f = \mathbf{u}_s$ *on* $\hat{\Sigma}$. The fluid and structure velocities are equal on the interface, meaning the fluid moves with the interface at all times. Since we use a global function for $\mathbf{u}$ in both fluid and structure domains, this condition is upheld. The fluid and solid velocities are usually in different coordinate systems, the solid velocity is then not available in Eulerian coordinates. We instead link fluid velocity at the interface by using the fact that $\mathbf{u}_s = \frac{\partial \mathbf{d}}{\partial t}$. Setting $\mathbf{u}_f = \frac{\partial \mathbf{d}}{\partial t}$ at the interface.

- *Dynamic condition*: $\sigma_f n_f = \sigma_s n_s$ *on* $\hat{\Sigma}$. The dynamic interface condition relates to Newtons third law of action and reaction. The forces on the interface area, here written as the normal forces are balanced on the interface. The forces from the fluid are defined in the time domain and is therefore written in terms of the reference domain:

  $$J\sigma_f F^{-T} n_f = P n_s \ on \ \hat{\Sigma}$$

  The dynamic condition is a Neumann condition that belongs to both subproblems.

- *Geometrical condition*: The geometric condition implies that the fluid and structure domains should not overlap, but rather that elements connect so the functions needing to transfer force are continuos across the entire domain.

3.4 Methods for Domain Representation and Mesh Quality Preservation

The kinematic interface conditions states that the fluid moves with the solid, and therefore the fluid domain needs to move in accordance with the solid deformations. The deformations from the structure are extrapolated through the interface into the fluid domain using, what is known in the literature as lifting operators. The lifting operators redistributes the interior node locations to uphold the mesh quality in the fluid domain. The choice of lifting operator is important for the overall FSI problem to be calculated [32]. When large deformations occur, we need a good lifting operator to uphold the integrity of the computing domain. A poor choice may cause the cells to overlap and singularities may occur. In the best case the numerical solution diverges, in the worst case the numerical solution will be wrong. When extrapolating deformation from the solid to the fluid domain, the fluid domain itself acts as a structure, deforming according to the deformations from the structure domain.

I will in this section present different lifting operators, that act differently on the computational domain. In chapter 5 the techniques will be tested and investigated.

3.4.1 Harmonic Lifting Operator

The harmonic lifting operator can be used for small to moderate deformations. The harmonic lifting operator is the Laplace equation, transporting the deformations from the solid into the fluid domain. A variable $\alpha_u > 0$,which can be constant or varied spatially, can be multiplied to the Laplace equation, to control the amount of lifting of deformations to the fluid domain.

$$\begin{aligned} -\alpha_u \nabla^2 \mathbf{d} &= 0 \quad in \ \hat{\mathcal{F}} && (3.16) \\ \mathbf{d} &= 0 \quad on \ \partial\hat{\mathcal{F}} \backslash \hat{\Sigma} && (3.17) \\ \mathbf{d}_f &= \mathbf{d}_s \quad on \ \hat{\Sigma} && (3.18) \end{aligned}$$

When using the harmonic lifting operator the variable α_u is very important when calculating moderate deformations. For small deformations a constant can be used for α_u. But for larger deformations we need to be a bit more clever. A good strategy for choosing α_u was proposed by Wick in [32], and further discussed in [23] and [15]. This alpha gets bigger when closer to the interface:

$$\alpha_u = \frac{1}{x^q} \tag{3.19}$$

where x is the distance from the interface. If $q = 0$ the laplacian i recovered. When the distance becomes larger, α_u gets smaller, and vice versa. Defining α_u in this manner is a smart choice since it upholds the cell structure closer to the interface where most of the cell distortion appears.

3.4.2 Biharmonic Lifting Operator

The biharmonic lifting operator provides more freedom than the harmonic in choosing boundary conditions and choice of parameter $\alpha_u > 0$ [15, 32]. This is because the biharmonic extension, extends the deformation such that it upholds the integrity of the cells even in large deformations. In its simplest form it is written as:

$$-\alpha_u \nabla^4 \mathbf{d} = 0 \quad in\ \hat{\mathcal{F}} \tag{3.20}$$

The biharmonic extension is calculated using a mixed formulation where we introduce a new function ω (not to be confused with the domain velocity), the function is added to the system so that we solve for 4 functions ($\mathbf{u}, \mathbf{d}, p,$ and ω):

$$\omega = \alpha_u \nabla^2 \mathbf{d} \quad and \quad -\alpha_u \nabla^2 \omega = 0\ \ in\ \hat{\mathcal{F}} \tag{3.21}$$

with the two types of boundary conditions. The first boundary conditions being:

$$\mathbf{d} = \partial_n \mathbf{d} = 0 \quad on\ \partial\hat{\mathcal{F}} \setminus \hat{\Sigma} \tag{3.22}$$

$$\mathbf{d}_f = \mathbf{d}_s \quad on\ \hat{\Sigma} \tag{3.23}$$

The second type of boundary condition imposes conditions on $\mathbf{d}$ and ω, and are written in terms of single component functions $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}$ and $\omega^{(1)}, \omega^{(2)}$, in the x and y directions.

$$\mathbf{d}^{(1)} = \partial_n \mathbf{d}^{(1)} = 0, and \quad \omega^{(1)} = \partial_n \omega^{(1)} = 0 \quad on\ \partial\hat{\mathcal{F}}_{in,out} \tag{3.24}$$

$$\mathbf{d}^{(2)} = \partial_n \mathbf{d}^{(2)} = 0, and \quad \omega^{(2)} = \partial_n \omega^{(2)} = 0 \quad on\ \partial\hat{\mathcal{F}}_{walls} \tag{3.25}$$

$$\tag{3.26}$$

Since the biharmonic extension is a fourth order PDE, it will have a higher computational cost [18] than the harmonic.

3.5 Discretization of Monolithic Fluid-Structure Interaction Equations

The temporal discretization is performed using a finite difference scheme and the spatial discretization is treated with the finite element method, following the ideas and notations of [31].

The scheme is first introduced in a weak form and using, for simplicity, the harmonic lifting operator.

The full monolithic FSI weak formulations reads:
In the domain $\Omega \in \mathbb{R}^D(D = 1, 2, 3)$ and time interval [0,T], find $\mathbf{u} \in \Omega\times\mathbb{R}^+ \to \mathbb{R}^D$, $p \in \Omega \times \mathbb{R}^+ \to \mathbb{R}$ and $\mathbf{d} \in \Omega \times \mathbb{R}^+ \to \mathbb{R}^D$. Let ϕ, ψ and γ be the test functions used in the weak formulation, which are continuous across the entire domain.

$$\begin{aligned}
(J\rho_f\partial_t\mathbf{u}, \phi) + (J(\nabla\mathbf{u})F^{-1}(\mathbf{u} - \partial_t d), \phi)_{\hat{\mathcal{F}}} &=0 && (3.27)\\
(J\sigma_f F^{-T}, \nabla\phi)_{\hat{\mathcal{F}}} &=0 && (3.28)\\
(\rho_s\partial_t\mathbf{u}, \phi)_{\hat{\mathcal{S}}} + \big(P, \nabla\phi\big)_{\hat{\mathcal{S}}} &=0 && (3.29)\\
(\alpha_u\nabla\mathbf{u}, \nabla\psi)_{\hat{\mathcal{F}}} + (\nabla\cdot(JF^{-1}\mathbf{u}), \gamma)_{\hat{\mathcal{F}}} &=0 && (3.30)\\
\delta\big((\partial_t\mathbf{d}, \psi)_{\hat{\mathcal{S}}} - (\mathbf{u}, \psi)_{\hat{\mathcal{S}}}\big) &=0 && (3.31)\\
\big(J\sigma_{f,p}F^{-T}, \nabla\phi\big) &=0 && (3.32)
\end{aligned}$$

Introducing the *θ-scheme* from [31], which has the advantage of easily being changed from a backward (implicit), forward(explicit), or a Crank-Nicholson (implicit) scheme, by changing the value of θ. The Crank-Nicholson scheme is of second order, but suffers from instabilities in this monolithic scheme for certain time step values [31]. A remedy for these instabilities is to chose a Crank-Nicholson scheme that is shifted towards the implicit side. How this is performed will become evident once the scheme is defined.

The variational form is defined by dividing the equations (3.27) - (3.32) into four categories. The four divided categories consists of: a time group A_T (with time derivatives), implicit A_I (terms always kept implicit), pressure A_P and the rest A_E (stress terms and convection):

$$\begin{aligned}
A_T(U) &= (J\rho_f\partial_t\mathbf{u}, \phi) - (J(\nabla\mathbf{u})F^{-1}(\partial_t\mathbf{d}), \phi)_{\hat{\mathcal{F}}} && (3.33)\\
&\quad + (\rho_s\partial_t\mathbf{u}, \phi)_{\hat{\mathcal{S}}} + (\partial_t\mathbf{d}, \psi)_{\hat{\mathcal{S}}} && (3.34)\\
A_I(U) &= (\alpha_u\nabla\mathbf{u}, \nabla\psi)_{\hat{\mathcal{F}}} + (\nabla\cdot(JF^{-1}\mathbf{u}), \gamma)_{\hat{\mathcal{F}}} && (3.35)\\
A_E(U) &= (J(\nabla u)F^{-1}\mathbf{u}, \phi)_{\hat{\mathcal{F}}} + (J\sigma_{f,u}F^{-T}, \nabla\phi)_{\hat{\mathcal{F}}} && (3.36)\\
&\quad + \big(P, \nabla\phi\big)_{\hat{\mathcal{S}}} - (\mathbf{u}, \psi)_{\hat{\mathcal{S}}} && (3.37)\\
A_P(U) &= \big(J\sigma_{f,p}F^{-T}, \nabla\phi\big) && (3.38)
\end{aligned}$$

Notice that the fluid stress tensors have been split into a velocity and pressure part.

$$\sigma_{f,u} = \mu(\nabla u F^{-1} + F^{-T}\nabla u) \tag{3.39}$$
$$\sigma_{f,p} = -pI \tag{3.40}$$

For the time group, discretization is done in the following way:

$$A_T(U^{n,k}) \approx \frac{1}{k}\big(\rho_f J^{n,\theta}(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi\big)_{\hat{\mathcal{F}}} - \frac{1}{k}\big(\rho_f(\nabla u)(\mathbf{d}^n - \mathbf{d}^{n-1}), \phi\big)_{\hat{\mathcal{F}}} \tag{3.41}$$
$$+\frac{1}{k}\big(\rho_s(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi\big)_{\hat{\mathcal{S}}} + \frac{1}{k}\big((\mathbf{d}^n - \mathbf{d}^{n-1}), \psi\big)_{\hat{\mathcal{S}}} \tag{3.42}$$

The Jacobian $J^{n,\theta}$ is expressed as:

$$J^{n,\theta} = \theta J^n + (1-\theta)J^{n-1} \tag{3.43}$$

Let the θ *scheme* be defined as:

$$A_T(U^{n,k}) + \theta A_E(U^n) + A_P(U^n) + A_I(U^n) = \tag{3.44}$$
$$-(1-\theta)A_E(U^{n-1}) + \theta \hat{f}^n + (1-\theta)\hat{f}^{n-1} \tag{3.45}$$

By choosing a value of $\theta = 1$ we obtain the backward Euler scheme, for $\theta = \frac{1}{2}$ we get the Crank-Nicholson scheme and for the shifted Crank-Nicholson we set $\theta = \frac{1}{2} + k$, effectively shifting the scheme towards the implicit side. $\hat{f}$ is the body forces. The impact of choosing values for θ will be investigated in chapter 4.

Chapter 4

Verification and Validation of the Fluid-Structure Interaction Implementation

When investigating a real world problem with a numerical model, the general approach is to: describe the problem with a mathematical model, discretize and implement the model on a computer, and finally simulate the implemented model to gain insight into the real world problem.

A question then immediately arises, computer models have been known to be incorrect in the past [17], how can we trust the insight gained from numerical simulation? To answer this question we need to adress another question, are the equations implemented correct? If so, is the mathematical description of the problem adequately defined? Without answering these questions, being confident that your solutions are correct is difficult [17]. The process of generating evidence that computed solutions meets certain requirements to fulfill an intended purpose, in the context of scientific computing, is known as Verification and Validation. The goal of this section will hence be to verify and validate the different numerical schemes outlined in the two previous chapters.

The chapter starts with the process of Verification, where the fluid and structure numerical schemes will be verified separately. Then use a well known benchmark to validate the fluid, structure, and FSI models, separately.

4.1 Verification

Verification, in the context of scientific computing, is the process of determining wether or not the implementation of numerical algorithms in computer code, is correct [16]. Mapping a mathematical model to a computational model there will always be introduced an error, often referred to as truncation error. Verification helps us identify, quantify, and reduce the errors, assuring that there are no coding mistakes which effects the truncation error. Verification does not address wether or not the computed solutions are in alignment with physics in the real world. It only tells us that our model is computed correctly or not.

In Verification there are multiple classes of test that can be performed, one of which is *order of convergence tests*. Order of convergence are based on the behavior of the error between a known exact solution and a computed solution [20]. The most rigorous of the order of convergence test is the *Method of Manufactured Solution* (MMS) [16]. When performing a MMS test, rather than looking for an exact solution, we manufacture one. The idea is to create a solution *a priori*, and use this solution to generate an analytical source term for the governing PDEs and then compute the PDE with the source term to produce a solution. The manufactured solution does not need to have a physically realistic relation, since the solution is only testing the mathematics.

When manufacturing a solution in MMS tests there are a number of criteria that needs to be met for a solution to be sufficient. The manufactured solutions should be chosen to be non-trivial and analytic [16, 20]. The solutions should be manufactured such that all terms of the equation are of the same order of magnitude. For this reason trigonometric and exponential functions can be a smart choice, since they are smooth and infinitely differentiable. In short, a good manufactured solution is one that is complex enough so that it rigorously tests each part of the equations.

The procedure of MMS is as follows [16]:

- We define a mathematical model on the form $L(\mathbf{u}) = 0$, where $L(\mathbf{u})$ is a differential operator and u is a dependent variable.
- Define the analytical form of the manufactured solution $\hat{\mathbf{u}}$

- Use the model $L(u)$ with $\hat{\mathbf{u}}$ inserted to obtain an analytical source term $f = L(\hat{\mathbf{u}})$
- Initial and boundary conditions are enforced from $\hat{\mathbf{u}}$
- Find the numerical solution of the problem with the given source term, $L(\mathbf{u}) = f$

If we let $\mathbf{u}$ be the numerical solution and $\hat{\mathbf{u}}$ be the exact solution, $||.||$ be the L^2 norm, the error can be computed as:

$$E = ||\mathbf{u} - \hat{\mathbf{u}}|| \tag{4.1}$$

When we decrease the node spacing (Δx) or decrease time step size (Δt), we expect the solution to convergence towards a given solution and hence the error becomes smaller. If we assume uniform node spacing in all spatial directions:

$$E = C_1 \Delta x^k + C_2 \Delta t^l \tag{4.2}$$

where C_1 and C_2 are constants, $k = m + 1$ and m is the polynomial degree of the spatial elements. The error is hence dependent on the node spacing and the time step. In order to compute the convergence k based on the error, we first have to let the term with $C_2 \Delta t^l$ be negligible compared to $C_1 \Delta x^2$. Let E_{n+1} and E_n be the computed errors of a solution with fine and coarse node spacing respectively. Using equation (4.2) we can find k by:

$$\frac{E_{n+1}}{E_n} = \left(\frac{\Delta x_{n+1}}{\Delta x_n}\right)^k \tag{4.3}$$

$$k = \frac{log(\frac{E_{n+1}}{E_n})}{log(\frac{\Delta x_{n+1}}{\Delta x_n})} \tag{4.4}$$

After refining the mesh while keeping Δt fixed and sufficiently small, k can be compared to the theoretical order of convergence for each given problem. If the k that we have found matches the theoretical order of convergence, with small margin of error, there are no coding mistakes present which effects the order of convergence, and thus the accuracy of the numerical scheme.

4.1.1 Method of Manufactured Solution on the Implementation of the Solid Equation

The MMS test is constructed to verify the implementation of the solid equation (3.7), with the restriction:

$$\mathbf{u} = \frac{\partial d}{\partial t} \tag{4.5}$$

Solutions $\hat{\mathbf{d}}$ and $\hat{\mathbf{u}}$ are manufactured with sine and cosine such that the derivatives are guaranteed and we have temporal and spatial dependencies. The solutions are also manufactured to uphold the restriction $\mathbf{u} = \frac{\partial d}{\partial t}$.

$$\begin{aligned}\hat{\mathbf{d}}_\mathbf{e} =& (cos(y)sin(t), cos(x)sin(t))\\ \hat{\mathbf{u}}_\mathbf{e} =& (cos(y)cos(t), cos(x)cos(t))\end{aligned}$$

The manufactured solutions are used to produce a sourceterm f_s :

$$\rho_s \frac{\partial \hat{\mathbf{u}}_\mathbf{e}}{\partial t} - \nabla \cdot (P(\hat{\mathbf{d}}_\mathbf{e})) = f_s \tag{4.6}$$

The equations are solved for $\mathbf{d}$ and $\mathbf{u}$ on a unit square domain, and the number N denotes the number of spatial points in x and y direction. The computations were simulated for 10 time steps and the error was calculated for each time step and then the mean of all the errors were used as a measure of the error.

In Table 4.1 we set m = 1, and vary the number of spatial points from 4 to 64 keeping $\Delta t = 10^{-7}$. The error E_u and E_d decreases for increasing values of N.The order of convergence k_u and k_d converges toward the expected value of 2.

N	Δt	**m**	E_u	$\mathbf{k_u}$	E_d	$\mathbf{k_d}$
4	1×10^{-7}	**1**	0.0068828	**-**	3.7855×10^{-9}	**-**
8	1×10^{-7}	**1**	0.0017204	**2.0002**	9.4622×10^{-10}	**2.0002**
16	1×10^{-7}	**1**	0.0004300	**2.0000**	2.3654×10^{-10}	**2.0000**
32	1×10^{-7}	**1**	0.0001075	**2.0000**	5.9136×10^{-11}	**2.0000**
64	1×10^{-7}	**1**	0.0000268	**2.0000**	1.4783×10^{-11}	**2.0000**

Table 4.1: Order of convergence test for the spatial resolution using the method of manufactured solution on the implementation of the Solid equation with m = 1

In table 4.1.1 we check the temporal convergence, setting the number of spatial points has been fixed to 64, with varying Δt from 0.1 halving each step to 0.0065. The error E_u and E_d decreases for decreasing values of Δt. The scheme tested is theoretically temporal first order accurate, by setting a value $\theta = 1$ expecting a temporal order of convergence of 1. In table 4.1.1 we can observe that the convergence of k_u and k_d tends towards 1.

N	$\mathbf{\Delta t}$	$E_u[\times 10^{-6}]$	$\mathbf{k_u}$	$E_u[\times 10^{-8}]$	$\mathbf{k_d}$
64	**0.1**	0.027663		0.0034221	
64	**0.05**	0.013390	**1.0467**	0.0018093	**0.9194**
64	**0.025**	0.007016	**0.9324**	0.0009246	**0.9685**
64	**0.0125**	0.003645	**0.9444**	0.0004688	**0.9798**
64	**0.00625**	0.001828	**0.9957**	0.0002414	**0.9571**

Table 4.2: Order of convergence test for the temporal resolution using the method of manufactured solution on the implementation of the Solid equation

The MMS test of the solid equation has a clear trend toward 2 in spatial direction, and 1 in temporal direction. The temporal convergence rate k is not exactly 1, and is likely caused by the number of spatial points $N = 64$ is not high enough, such that the spatial error is negligible. With this in mind the trends shows convergence towards the theoretical convergence, which concludes that the solid equation has been implemented correctly.

4.1.2 Method of Manufactured Solutions on Fluid Equations with Prescribed Motion

In this section we verify the fluid equations (3.14) in the ALE framework computed on a reference domain, with a prescribed motion.

The functions $\hat{\mathbf{u}}$, $\hat{\mathbf{d}}$ and $\hat{p}$ are manufactured to uphold the restrictions $\mathbf{u} = \frac{\partial d}{\partial t}$ and incompressible fluid, and are made with sine and cosine function to uphold the criteria of MMS.

$$\begin{aligned}
\hat{\mathbf{d}} =& (cos(y)sin(t), cos(x)sin(t)) \\
\hat{\mathbf{u}} = \hat{\mathbf{w}} =& (cos(y)cos(t), cos(x)cos(t)) \\
\hat{p} =& cos(x)cos(t)
\end{aligned}$$

Whilst testing the implementations of the fluid equations, the opportunity arises to also test the mappings between current and reference configurations. The source term f_f is produced without mappings:

$$\rho_f \frac{\partial \hat{\mathbf{u}}}{\partial t} + \nabla \hat{\mathbf{u}}(\hat{\mathbf{u}} - \frac{\partial \hat{\mathbf{d}}}{\partial t}) - \nabla \cdot \sigma(\hat{\mathbf{u}}, \hat{p})_f = f_f \text{ in } \mathcal{F}$$

To be specific, we use f_f from the current configuration and map it to the reference:

$$\rho_f J \frac{\partial \mathbf{u}}{\partial t} + (\nabla \mathbf{u}) F^{-1}(\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) + \nabla \cdot (J \sigma_f(\hat{\mathbf{u}}, p) F^{-T}) = J f_f \text{ in } \hat{\mathcal{F}}$$

The computations are performed on a unit square domain and the computations were simulated with 10 timesteps and the error was calculated for each time step and then the mean of all the errors was taken and used as a measure of the error.

In table 4.3 we check the temporal convergence keeping the spatial points constant with $N = 64$, by varying Δt by half, from 0.1 to 0.0125. The errors for fluid velocity and pressure E_u and E_p decrease with decreasing time steps. The compute convergence k_u and k_p tends toward a value of 1.

N	Δt	E_u	$\mathbf{k_u}$	E_p	$\mathbf{k_p}$
64	0.1	5.1548×10^{-5}	**-**	0.008724	**-**
64	0.05	2.5369×10^{-5}	**1.0228**	0.004290	**1.0240**
64	0.025	1.2200×10^{-5}	**1.0561**	0.002058	**1.0596**
64	0.0125	0.56344×10^{-5}	**1.1145**	0.0009556	**1.1068**

Table 4.3: Order of convergence test for the temporal resolution using the method of manufactured solution on the implementation of the fluid equations

In table 4.4 we check the spatial convergence keeping the time step fixed $\Delta t = 10^{-6}$. Increasing spatial points N from 2 to 16. The error E_u and E_p decreases with increasing spatial points. Computed convergence k_u tends toward 3 and k_p tends towards 2, which is expected when computing with P2-P2-P1 elements, for velocity, deformations and pressure, respectively.

N	Δt	m	E_u	k_u	E_p	k_p
2	1×10^{-6}	**2**	8.6955×10^{-4}	**-**	0.01943	**-**
4	1×10^{-6}	**2**	1.0844×10^{-4}	**3.0032**	0.00481	**2.0140**
8	1×10^{-6}	**2**	0.1354×10^{-4}	**3.0007**	0.00119	**2.0120**
16	1×10^{-6}	**2**	0.0169×10^{-4}	**3.0001**	0.00029	**2.0074**

Table 4.4: Results of MMS ALE FSI u=w

The MMS test of the fluid equations computed from the reference domain shows trends in the spatial convergence toward 3 in the fluid velocity and 2 in pressure, which is expected. For the temporal convergence of the fluid equation the trend is towards 1 but is not exactly 1. The reason could be that the number of spatial points are not high enough, and also that the fluid equations have been computed on a reference domain. Nonetheless the convergence rates are sufficient and giving the conclusion that the fluid equations are implemented correctly.

Discussion of the Method of Manufactured Solutions Tests

It should be noted that a more rigorous MMS test of the FSI problem would be to test the entire FSI problem, and not splitting the test into parts. To do a full MMS of the entire FSI problem, one needs to take into account the condition of continuity of velocity on the interface [5], the stresses need to equal on the interface and the flow needs to be divergence free. Manufacturing such a solution is very difficult [4]. The author has yet to find a paper that manufactures a solution fulfilling all the condition in a rigorous manner. For this reason the MMS was split into parts, and for the intended use the author finds the results from MMS tests sufficient to proceed with validation.

4.2 Validation

After the code has been verified, we move on to Validation which is the process of determining if a model gives an accurate representation of real world physics within the bounds of the intended use [17]. Depending on the quantities of interest and the problem at hand the computational model has to be validated. However, when solving a multiphysics problem, good benchmarks and trustworthy experimental data might be difficult to produce [14]. Therefore we will here validate the solver, *brick by brick*, starting with simple testing of each part of the model and building more complexity and eventually testing the entire model.

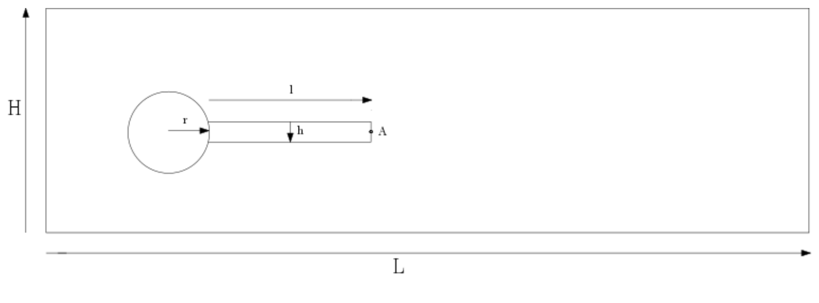
In the benchmarks used for validation, there are 9 different tests. For each test a refinement with respect to temporal an spatial resolution is performed. However, one major draw back is that the results of the benchmark were known a priori. It is easier to mold the model to the data we already have, and as Oberkampf and Trucano in [17] puts it “Knowing the ‘correct’ answer beforehand is extremely seductive, even to a saint”. Knowing the limitations of our tests will therefore strengthen our confidence in the model. The process of verifying and validating, if one does not clearly know the bounds of sufficient accuracy, is an endless process [17].

4.2.1 Fluid-Structure Interaction between an elastic object and laminar incompressible flow

The benchmark used in this chapter is called “Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow” [11], based on an older well known CFD benchmark[21]. The authors provides a computational domain, and boundary conditions, splitting up into a: computational solid mechanical (CSM) part called CSM1-3, a CFD part called CFD1-3 and a full FSI part named FSI1-3. Providing in total 9 subproblems with 3 problems in each part. The chapter starts by defining the computing domain, the boundary conditions and quantities for comparison. For the sake of completeness we split up into the three parts, CSM, CFD and FSI, listing how the subtests are computed and providing results.

Problem Defintion

Domain



The computational domain consists of a rigid circle with an elastic bar attached behind the circle. The circle is positioned at (0.2, 0.2) making it 0.05 off center vertically. This shift in the domain is done to induce oscillations, for some flow conditions, to an otherwise symmetric flow.

L	H	l	h	A
2.5	0.41	0.35	0.02	(0.2, 0.6)

Table 4.5: Domain parameters

The mesh shown in figures 4.1 and 4.2 were created using Gmsh, with 11556 Cells, which is the mesh with the smallest node spacing. Table 4.6 shows the number of cells and dofs used in each mesh. Elements are the computational cells in the mesh and dofs are the *degrees of freedom*, often called the number of unknowns.

Mesh level	Elements	Dofs
1	2474	21749
2	7307	63365
3	11556	99810

Table 4.6: Mesh levels showing number og cells and dofs in each mesh

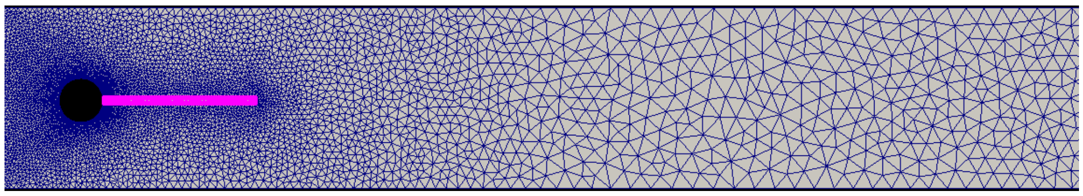


Figure 4.1: Picture of entire FSI computational domain with 11556 cells

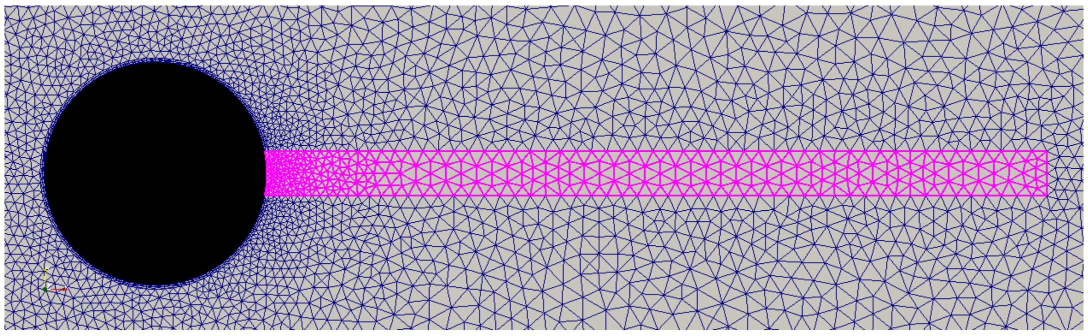


Figure 4.2: Picture of FSI computational domain with 11556 cells, zoomed in with the solid domain marked in pink. Around the circle we can see a small boundary layer with the width of 2 cells

Boundary conditions

A parabolic profile has been prescribed to the inlet velocity that increases from $t = [0, 2]$ and is kept constant after $t = 2$. The fluid velocity on upper and lower walls are set to zero, normally called a "no slip" condition. The pressure is set to zero on the outlet.

$$
\begin{aligned}
u(0, y) &= 1.5u_0 \frac{y(H - y)}{(\frac{H}{2})^2} \\
u(0, y, t) &= u(0, y) \frac{1 - cos(\frac{\pi}{2}t)}{2} \text{ for } t < 2.0 \\
u(0, y, t) &= u(0, y) \text{ for } t \geq 2.0
\end{aligned}
$$

Quantities for comparison

When the fluid moves around the circle and bar it exerts a frictional force. These forces are split into drag and lift and calculated as follows:

$$(F_d, F_L) = \int_S \sigma_f n dS$$

where S is the part of the circle and bar in contact with the fluid.
We define a point $A = (0.2, 0.6)$ on the right side of the bar. Where this point is in the spatial direction gives a measure for how much the bar has deflected.
For some given inflow conditions, unsteady solutions appear. For the unsteady solutions the values, meaning drag and lift, and displacement in x and y directions, are represented by the mean and amplitude values:

$$mean = \frac{1}{2}(max + min) \tag{4.7}$$

$$amplitude = \frac{1}{2}(max - min) \tag{4.8}$$

$$\tag{4.9}$$

In each test the values denoted as **ref** are the values taken from the original benchmark proposal paper [11].

CFD test cases

The CFD tests can be simulated with two approaches. The first way assumes the bar to be rigid object, meaning that the computational domain consists of the fluid domain only, and a no slip condition has been set on the interface. The other way, which is implemented in this thesis, is by computing the problem with the entire FSI scheme, but setting by setting $\rho_s = 10^6$ and $\mu_s = 10^{12}$, such that the bar is almost completely rigid, only giving rise to very small deformation (in the $10^{-9} - 10^{-10}$ range).

The CFD tests cases, CFD1 and CFD2, are simulated with Reynolds numbers 20 and 100 converging to steady solutions, 4.7. The CFD3 benchmark has a Reynolds number 200 at which point the asymmetry in the geometry will cause vortex shedding behind the circle and bar, and thus create an unsteady solution.

Parameters	CFD1	CFD2	CFD3
$\rho_f[10^3 \frac{kg}{m^3}]$	1	1	1
$\nu_f[10^{-3} \frac{m^2}{s}]$	1	1	1
$U[\frac{m}{s}]$	0.2	1	2
$\text{Re} = \frac{Ud}{\nu_f}$	20	100	200

Table 4.7: Summary of all the parameters in CFD tests

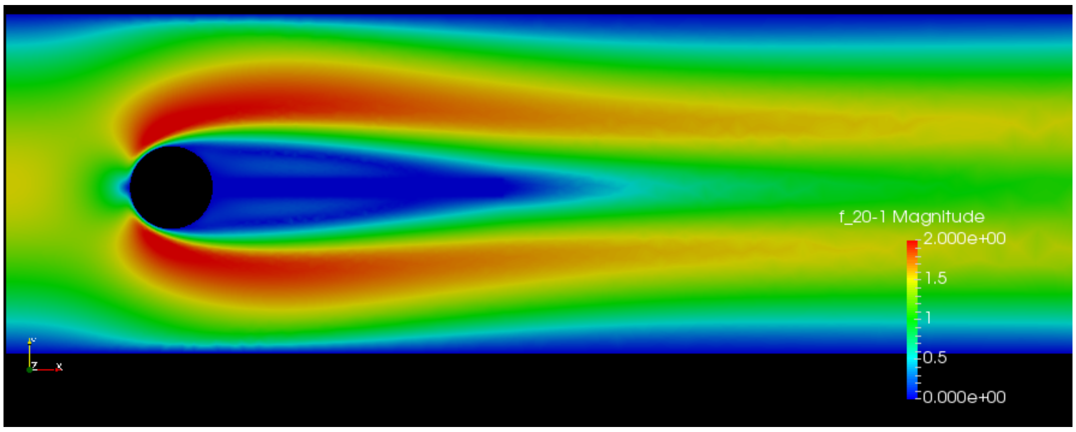


Figure 4.3: CFD2 steady state case fluid flow with 11556 cells

Notice in figure 4.2.1, which shows the fluid flow of the CFD2 case, that since the circle and bar are positioned non symmetric in the y direction there is more fluid flowing closer to the upper boundary. The CFD2 case has an inlet velocity just below the point of inducing oscillations.

Table 4.8 shows the results for the CFD1 testcase, showing convergence towards the referential values in Drag and Lift for increasing elements and dofs.

elements	**dofs**	**Drag**	**Lift**
2474	21749	14.059	1.100
7307	63365	14.110	1.080
11556	99810	14.200	1.1093
ref		**14.29**	**1.119**

Table 4.8: Results of CFD1 case simulated as full FSI, with almost rigid bar

Table 4.9 shows the results for CFD2 tending towards the referential values in Drag and Lift for increasing elements and dofs.

elements	**dofs**	**Drag**	**Lift**
2474	21749	134.9	10.38
7307	63365	135.4	10.0
11556	99810	136.1	10.41
ref		**136.7**	**10.53**

Table 4.9: Results of CFD2 case run as full FSI with almost rigid bar

Table 4.2.1 shows the results for the CFD3 benchmark, the results show clear convergence toward the **ref** value for increased number of cells and dofs.

elements	**dofs**	**Drag**	**Lift**
2474	21749	434.42 ± 4.28	-15.63 ± 407.59
7307	63365	435.54 ± 5.06	-11.77 ± 425.73
11556	99810	438.13 ± 5.42	-10.01 ± 435.40
ref	**ref**	$\mathbf{439.45 \pm 5.61}$	$\mathbf{-11.893 \pm 437.81}$

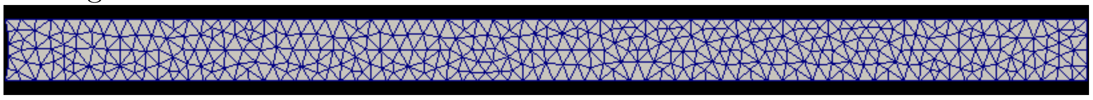
Table 4.10: Results of unsteady state case CFD 3 with $\Delta t = 0.01$, where the **ref** was computed with $\Delta t = 0.005$

The solutions to the CFD1-3 test cases gives satisfactory results compared to the referential values given in the benchmark paper.

Computational Structural Mechanical test cases

The CSM tests are calculated using only the bar as computational domain. A body force f_s is set a gravitational force g, which has been kept fixed throughout the CSM tests, changing only the material parameters of the solid. The tests CSM1 and CSM2 gives rise to steady state solutions. The difference between them is a more slender bar. The CSM 3 gives unsteady solutions, and since there is no friction the bar should, if energy is preserved hence a correct solution, move down and back up infinitely.

Figure 4.4: Picture of the coarsest solid mesh used in the MMS test



Parameters	CSM1	CSM2	CSM3
$\rho_s[10^3 \frac{kg}{m^3}]$	1	1	1
ν_s	0.4	0.4	0.4
$\mu_s[10^6 \frac{m^2}{s}]$	0.5	2.0	0.5
g	2	2	2

Table 4.11: Summary table of the parameters used in the CSM tests

The Tables 4.12 ,4.13 and 4.14 shows the results of the CSM1, CSM2 and CSM3 cases respectively. All three show a clear tendency towards the referential values when increasing the number of elements.

The Figure 4.9 is of displacement at the point A in x and y direction of the CSM3 test. The CSM3 test was run with Crank-Nicholson, $\theta = 0.5$, and it can be seen that in the y displacement the bar returns to it initial state, that is with zero displacement. This results indicates that the energy in the system has been preserved.

The results are within 1% error margin, and we therefore consider the results as satisfactory.

elements	dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$
725	1756	-5.809	-59.47
2900	6408	-6.779	-64.21
11600	24412	-7.085	-65.63
46400	95220	-7.116	-65.74
ref	**ref**	**-7.187**	**-66.10**

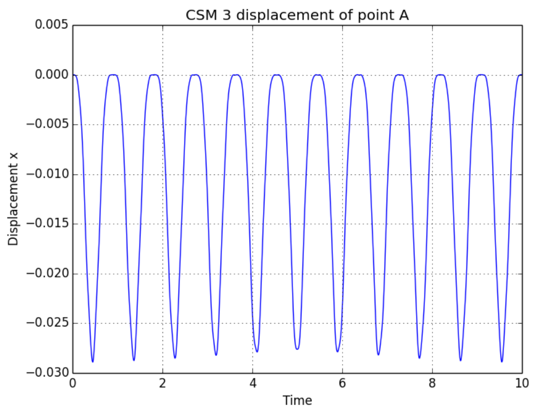
Table 4.12: Results of the steady CSM1 case from coarse to fine mesh.

Elements	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$
725	1756	-0.375	-15.19
2900	6408	-0.441	-16.46
11600	24412	-0.462	-16.84
46400	95220	-0.464	-16.87
ref	**ref**	**-0.469**	**-16.97**

Table 4.13: Results of the steady CSM2 case from coarse to fine mesh.

elements	dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$
725	1756	-11.743 ± 11.744	-57.952 ± 58.940
2900	6408	-13.558 ± 13.559	-61.968 ± 63.440
11600	24412	-14.128 ± 14.127	-63.216 ± 64.744
46400	95220	-14.182 ± 14.181	-63.305 ± 64.843
ref		$\mathbf{-14.305 \pm 14.305}$	$\mathbf{-63.607 \pm 65.160}$

Table 4.14: Results of the unsteady CSM3 case with mesh from coarse to fine.



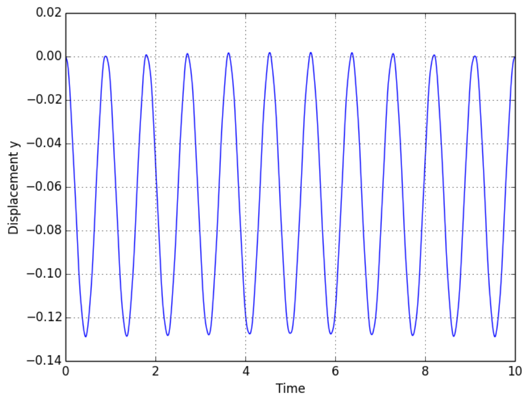
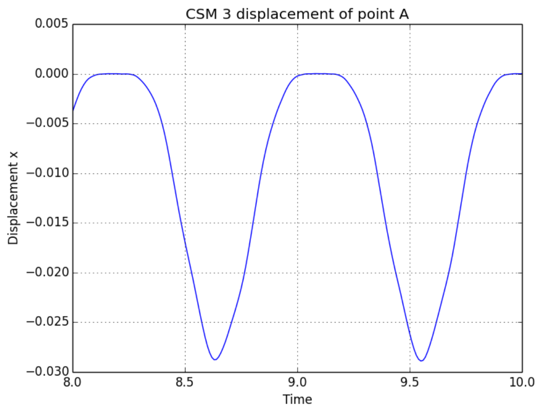


Figure 4.5: Displacement in x direction, timeinterval (0,10)

Figure 4.6: Displacement in y direction, timeinterval (0,10)



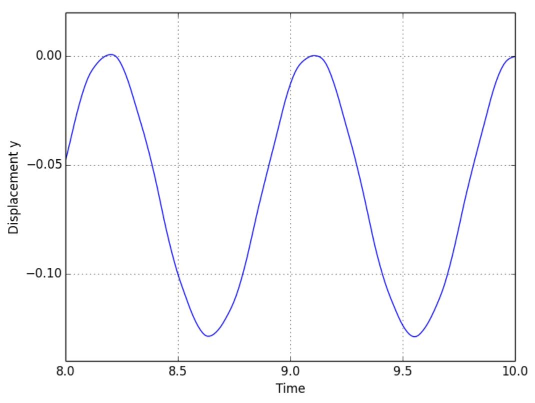


Figure 4.7: Displacement in x direction, timeinterval (8,10)

Figure 4.8: Displacement in y direction, timeinterval (8,10)

Figure 4.9: Plots of the results for CSM3 showing Displacement of point A

FSI tests

The FSI tests are run with 2 different inflows conditions. FSI1 gives a steady state solution while the others are unsteady. FSI2 gives the largest deformation is therefore considered the most difficult of the three [19], giving deformations of 2.5 times greater than the flag height. FSI2 was computed using the harmonic lifting operator with variable α_u for $\Delta t = 0.01, \Delta t = 0.001$.

The FSI3 test has the highest inflow speed giving medium deformations but more rapid oscillations. The parameters for FSI3 are shown in table 4.15. FSI3 was computed using the biharmonic lifting operator with a constant α_u for $\Delta t = 0.01, \Delta t = 0.001$.

Parameters	FSI1	FSI2	FSI3
$\rho_f[10^3 \frac{kg}{m^3}]$	1	1	1
$\nu_f[10^{-3} \frac{m^2}{s}]$	1	1	1
u_0	0.2	1	2
$\text{Re} = \frac{Ud}{\nu_f}$	20	100	200
$\rho_s[10^3 \frac{kg}{m^3}]$	1	10	1
ν_s	0.4	0.4	0.4
$\mu_s[10^6 \frac{m^2}{s}]$	0.5	0.5	2

Table 4.15: FSI Parameters

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2474	21749	0.0229	0.8265	14.0581	0.7546
7307	63365	0.02309	0.7797	14.1077	0.7518
11556	99810	0.02295	0.8249	14.2046	0.7613
ref	**ref**	**0.0227**	**0.8209**	**14.295**	**0.7638**

Table 4.16: Results of FSI 1 test case

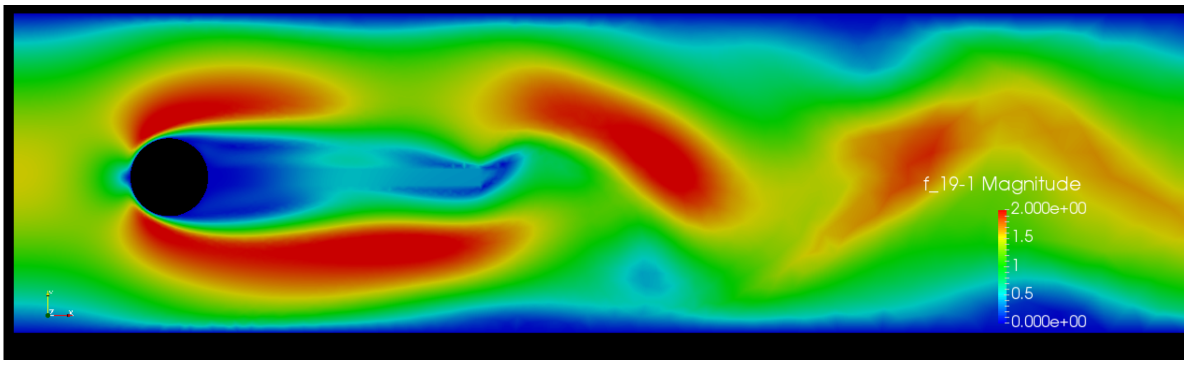


Figure 4.10: FSI2 test case Fluid velocity at $t = 9.70$ s shown on the reference mesh, the bar has in reality moved but is not shown here.

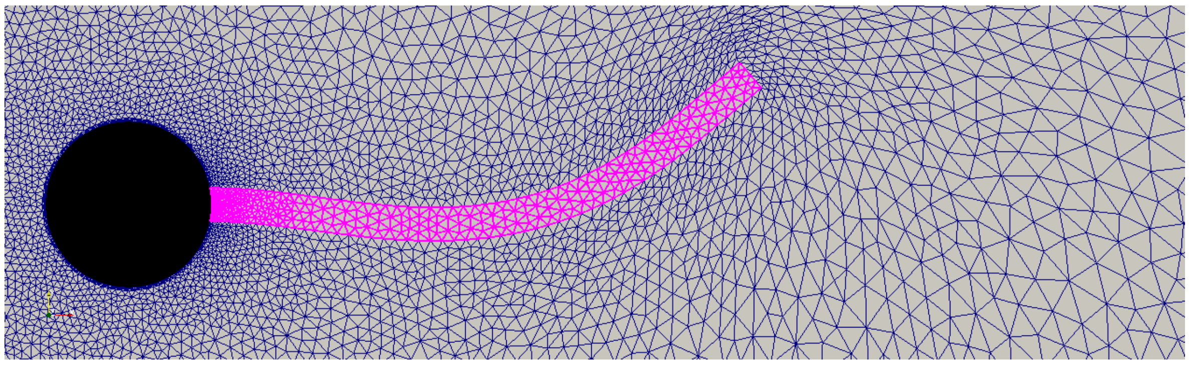


Figure 4.11: FSI2 test case deformation at $t = 9.70\,\mathrm{s}$. The bar is marked with pink colour and deformed using *warp by vector* in Paraview.

The tables 4.17 and 4.18 shows results for the FSI2 test case. First observing the third and fourth column, we can observe that the displacement of the bar appear to converge towards the reference values. However, for the for drag and lift neither is converging for $\Delta t = 0.01$. For a finer time step, the lift is greatly improved, see Table 4.18 , however the drag is off be $\approx 30\%$.

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2474	21749	-15.26 ± 13.44	1.34 ± 82.38	157.02 ± 14.79	-1.426 ± 258.4
7307	63365	-14.96 ± 13.24	1.01 ± 81.67	159.01 ± 16.33	1.88 ± 254.2
11556	99810	-14.96 ± 13.23	1.29 ± 81.9	161.09 ± 17.66	0.06 ± 255.78
ref	**ref**	**-14.58 ± 12.44**	**1.23 ± 80.6**	**208.83 ± 73.75**	**0.88 ± 234.2**

Table 4.17: FSI2 test case results, $\Delta t = 0.01$, using the harmonic lifting operator.

Cells	Dofs	$d_x(A)[x10^{-3}]$	$d_y(A)[x10^{-3}]$	Drag	Lift
2474	21749	-15.10 ± 13.32	1.16 ± 82.46	159.53 ± 17.44	0.68 ± 259.10
7307	63365	-14.85 ± 13.14	1.21 ± 81.72	160.72 ± 17.84	0.93 ± 255.14
11556	99810	-14.83 ± 13.11	1.24 ± 81.6	161.50 ± 18.17	0.62 ± 254.40
ref	**ref**	**-14.58 ± 12.44**	**1.23 ± 80.6**	**208.83 ± 73.75**	**0.88 ± 234.2**

Table 4.18: FSI2 with $\Delta t = 0.001$, using harmonic lifting operator.

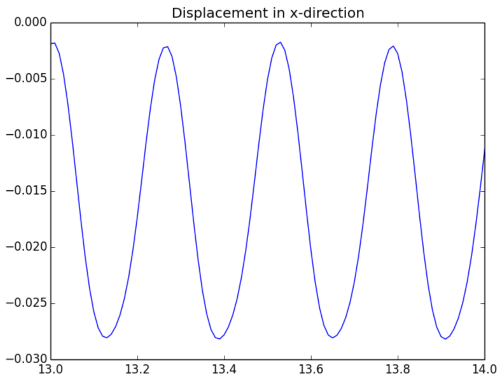


Figure 4.12: Displacement x vs time

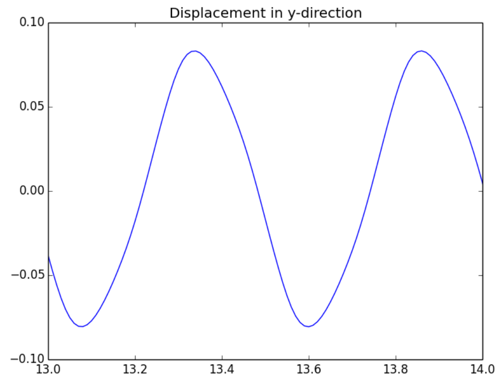


Figure 4.13: Displacement y vs time

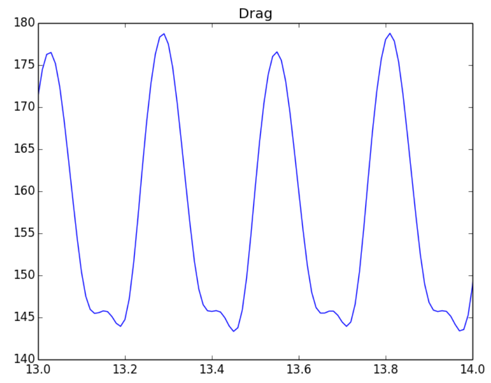


Figure 4.14: Drag vs time

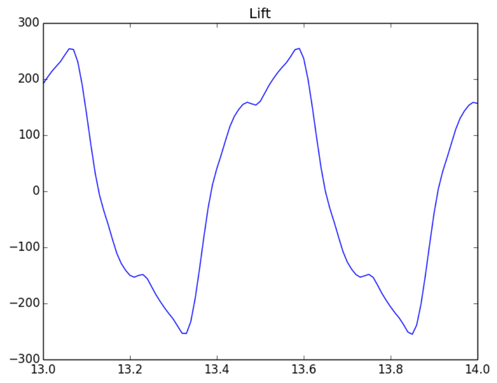


Figure 4.15: Lift vs time

Figure 4.16: Plots of FSI2 result values for, $\Delta t = 0.001$, with 11556 elements

In tables 4.19 and 4.20 shows that results for FSI3 with $\Delta t = 0.01$ and $\Delta t = 0.001$, respectively. Both tables show convergence for displacements in both directions and for drag. In the results for lift the values are more scattered and not showing a clear trend and is about 50 % off the referential values.

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2474	21249	-1.79 ± 1.80	3.29 ± 26.41	439.36 ± 12.04	1.96 ± 142.31
7307	63365	-2.48 ± 2.48	1.64 ± 32.82	449.77 ± 18.02	3.41 ± 153.47
11556	99810	-2.47 ± 2.45	1.27 ± 32.81	456.60 ± 18.73	1.55 ± 153.46
ref	ref	$\mathbf{-2.69 \pm 2.56}$	$\mathbf{1.48 \pm 34.38}$	$\mathbf{457.3 \pm 22.66}$	$\mathbf{2.22 \pm 149.78}$

Table 4.19: FSI3 unsteady test case results with $\Delta t = 0.01$, with biharmonic bc1 lifting operator

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2474	21249	-2.188 ± 2.11	3.56 ± 2.90	435.19 ± 9.77	-1.57 ± 151.43
7307	63365	-1.42 ± 4.70	0.77 ± 28.50	454.38 ± 19.75	1.79 ± 155.08
11556	99810	-2.23 ± 6.164	1.72 ± 44.48	459.12 ± 22.97	3.12 ± 171.22
ref	**ref**	$\mathbf{-2.69 \pm 2.56}$	$\mathbf{1.48 \pm 34.38}$	$\mathbf{457.3 \pm 22.66}$	$\mathbf{2.22 \pm 149.78}$

Table 4.20: FSI3 unsteady test case results with $\Delta t = 0.001$, with biharmonic bc1 lifting operator

Discussion on FSI tests

A very important thing to notice about this benchmark [11] is that it is a proposal for a benchmark, as it is called "Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow". So the point of the paper is give a specific problem setup which others can contribute result-data. A paper was published in 2010 by J. Hron, Turek, et al, [24] that compared results of different discretizations and solution approaches. The study [24] gives 7 different methods and results for two of the FSI test cases. They state in the numerical results that "However, also clear differences between the different approaches with regard to accuracy are visible. Particularly for the drag and lift values, which lead to differences of up to order 50%, and also for the displacement valueswhich are in the range of 10% errors.". With this in mind it is important to know that the referential values used are only those reported from the original paper, which only looked at one implementation. While the paper which compares results, only 2 of the 7 contributions were schemes of monolithic nature, which are the closest one should refer to in this thessis. In the Appendix B is a copy of the results from the paper comparing schemes, showing different results for different discretization, with different time steps and unknowns.

The FSI1 test gives a low fluid velocity and gives very low displacements.

FSI1 is therefore not a rigid test for FSI. In fact I personally experienced in the beginning of making the FSI solver, that even that even with an erronous implementation I obtained adequate results. However it is a good test for early checks, because if FSI1 is wrong more complex cases will definitely not provide good results.

For the FSI2 test case we only have results from the initial Hron and Turek paper [11]. As previously stated the results for the FSI3 case differ by in some cases 50% for Drag and Lift. With this in mind, in the FSI2 case I am off by less then 10% for displacement in x and y direction and for Lift. While Drag is off by about 50%. It is reasonable to assume that since there were such differences in the results for different implementations for the FSI3 results, we would expect similar behavior in the FSI2 results.

If we compare the results reported in the FSI3 case in the inital paper by Hron and Turek 2006, to their reported results in 2010 B implementation 3, we can see that they do not report the same results same scheme, leading one believe that that they have changed their implementation a bit.

A note should be added about the construction of the computational meshes. If we look at figures 4.1 and 4.2. The node spacing on the inlet is small to ensure that the parabolic inlet profile was adequately represented. There is also smaller node spacing around the circle and around the bar, leading to larger node spacing as we move downstream. In the unsteady CFD and FSI cases there is vortex shedding happening downstream. With large node spacing in this area the vortexes may not be produced to its full extent, hence introducing errors in the unsteady results.

In hindsight, larger gaps in the number of elements between each mesh should have been larger. The three meshes that are mainly used go from 2474 cells to 7307 cells to 11556 cells. When calculating in 2D to see converging effects in the results of smaller node spacing, one should make meshes with 4 times the number of cells for each new mesh. This might have helped in converging to the referential values. The reason for not being able to run lower values for Δt or increased number of cells in the meshes was a lack of computational resources.

Chapter 5

Comparing the Effects of Different Lifting Operators and Investigating Numerical Stability in Fluid-Structure Interaction Problems

The first section is devoted to comparing different lifting operators defined in section 3.4. The choice of lifting operators is crucial in FSI problems [23]. When handling large structural deformations one has to be very cautious about the choice of lifting operators. A good lifting operator upholds the integrity of the computational domain, allowing large deformations in the solid, moving into the fluid domain. The most robust lifting operators are also the most computational expensive. There is therefore a trade-off between computational cost and robustness. In an problem where the deformations are small the simplest lifting operator will suffice, however if the deformations are large one of the more computational costly lifting operators have to be used.

The second section briefly investigates the impact of choosing different value for θ in the θ-scheme. The effects of choosing a Crank-Nicholson or a backward Euler scheme is known to have effects on the energy preservation in a computational system. Also the effects of shifting the Crank-Nicholson scheme is investigated using the FSI2 and FSI3 benchmarks from the previous chapter.

5.1 Methods for Comparing Lifting Operators

The comparisons will be performed using a version of the CSM1 test as defined the previous chapter, with the same computational domain as the full FSI benchmark and parameters of the CSM1. The version of CSM1 test case is now computed as a full FSI problem with the fluid initially at rest. A gravitational force is applied to the structure like the previous CSM test. The only difference is that we now use the full domain from the 4.2.1. The test is simulated as time-dependent with a the backward Euler scheme, leading to a steady state solution.

The tests will compare the different operators by investigating how the deformation from the solid domain is lifted into the fluid domain, investigating the mesh after deformation to see how much cells distort and where the cells distort. This is visualized using Paraview with its built in function *warp by vector*, which redistributes nodes based on the displacement values in each nodal point. The computing domain is the same as used in the Hron Turek benchmark, from the previous section. The Dirichlet boundaries are set to the "no slip" condition. The left Neumann boundary is set to "do nothing", and zero pressure.

The different operators will be measured with the minimal value of the Jacobian. The Jacobian is also known as the volume ratio, and if the Jacobian is zero anywhere in the domain it means that the volume is negative, and cells overlap. If cells overlap it can cause singularity in the matrices during assembly. When cells overlap it can in the best case cause the computed numerical code to diverge, and in the worst case just give very wrong results.

A plot of the deformation in the domain has been added, to to visually inspect the result of each lifting operator. It is possible to see that if get thin triangles in the computational domain then the lifting operator is not good enough, and we might get singularities in the computing matrix. I also investigated how different lifting operators react differently in the FSI2 and FSI3 test cases from the previous section. To that end, quantitative results of the drag an lift are compared with plotting drag and lift versus time. Bc1 and bc2 denotes the boundary conditions 1 and 2 used when employing the biharmonic lifting operator.

Results of Investigating Different Lifting Operators

Figure 5.1 shows the minimum of the Jacobian of the entire domain. The harmonic operator with a constant α_u parameter, results in overlapping cells, which can be seen from the the plot since it is below zero. In contrast, the harmonic lifting operator with variable α_u, and both biharmonic lifting operators preserves the cell quality.

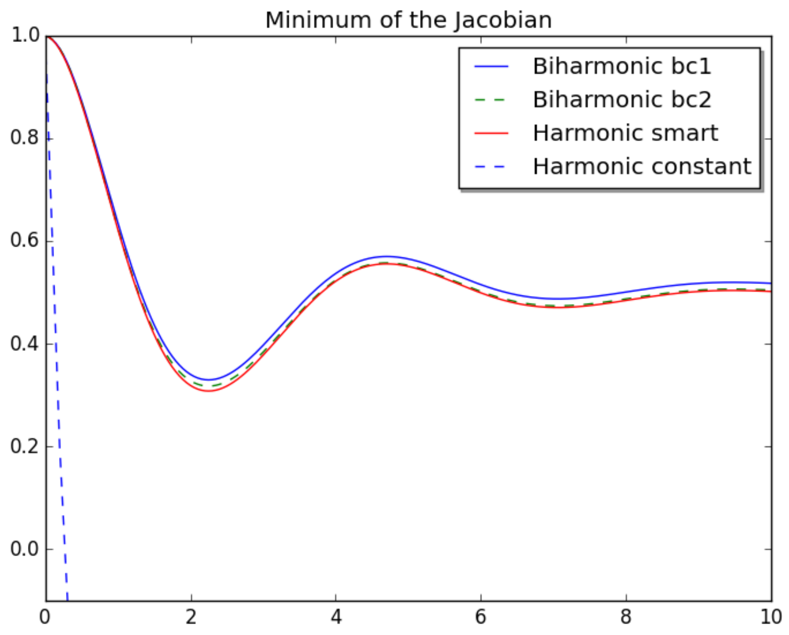


Figure 5.1: plot of the minimum of J vs time in entire domain, using the CSM1 test. $\Delta t = 0.05$

Figure 5.6 displays how the mesh nodes are distrubuted at the steady state solution in the CSM1 test case. The same as we observed in Figure 5.1, is here displayed visually, only the harmonic lifting operator with constant α_u is not able to avoid inverted cells. However, the computed numerical code was able to compute with the harmonic constant α_u operator, but as we can see in table 5.1 the displacement values are incorrect.

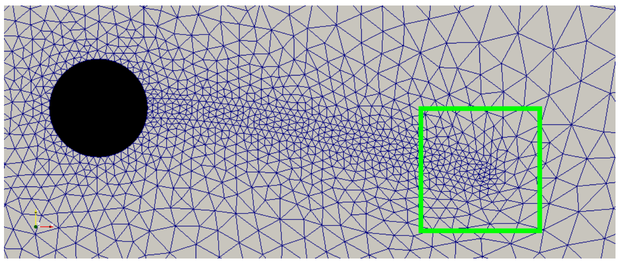


Figure 5.2: Harmonic lifting operator with spatial dependent α_u

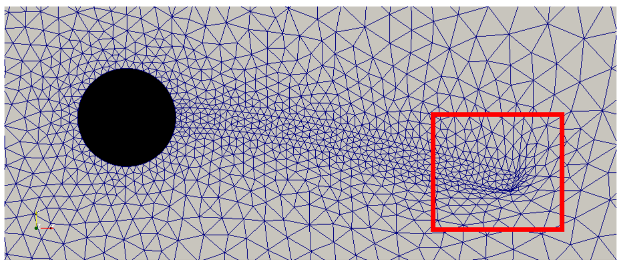


Figure 5.3: Harmonic lifting operator with constant α_u

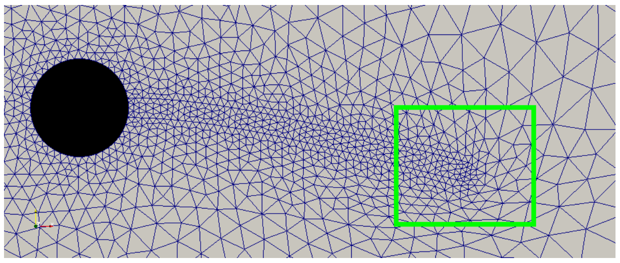


Figure 5.4: Biharmonic lifting operator with boundary condition 1

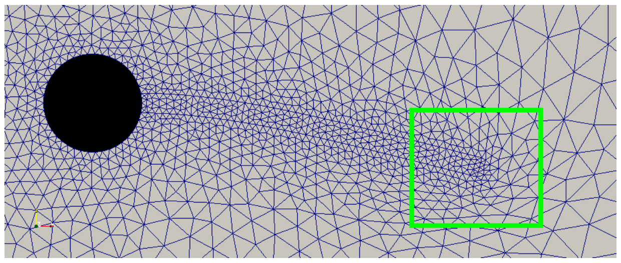


Figure 5.5: Biharmonic lifting operator with boundary conditions 2

Figure 5.6: Results of testing different lifting operator using the CSM1 test-case computing full FSI. Green square denoting good cell integrity.

Technique	$d_y(A)[\times 10^{-3}]$	$d_x(A)[\times 10^{-3}]$
Harmonic	-65.406	-7.036
Constant	-43.033	-2.999
Bibc1	-65.404	-7.036
Bibc2	-65.405	-7.036
Hron & Turek	$\mathbf{-66.10}$	$\mathbf{-7.187}$

Table 5.1: Displacements results of different lifting operators of CSM1 test

FSI2 with Different Lifting Operator

Figure 5.7 shows the harmonic and the two biharmonic mesh motion techniques for the FSI2 case. All three are similar and only a slight change in the period can be noticed.

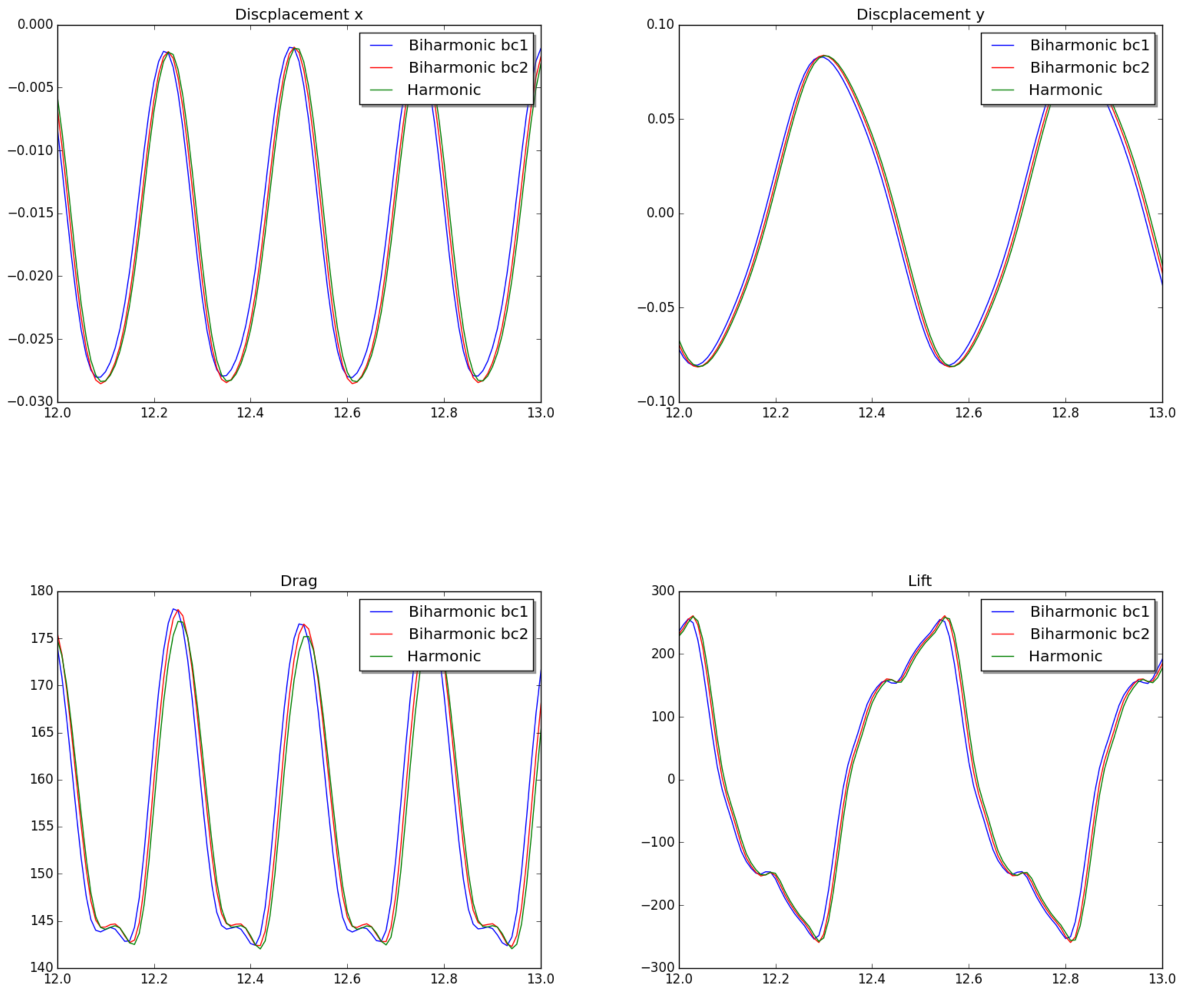


Figure 5.7: FSI2 displacements, drag, and lift, vs time, with different lifting operators: harmonic with varying α_u, biharmonic bc1 and bc2. $\Delta t = 0.01$

Figure 5.8, 5.9, 5.10 and 5.11 shows the displacement in x and y direction, and drag and lift plots respectively. The displacements and Lift plots show only a slight change in the period. While the Drag for the harmonic lifting operator with mesh dependent α_u shows an increasing in the Drag value of about 10.

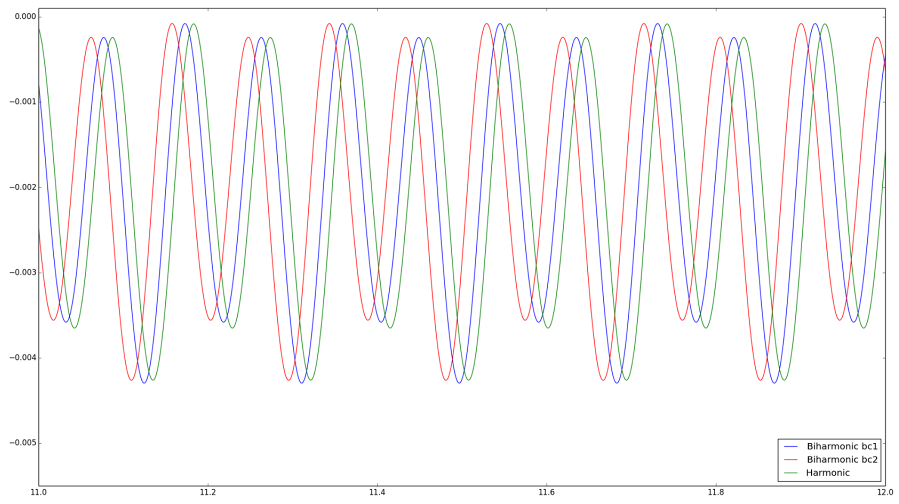


Figure 5.8: Displacement in x direction vs time for FSI3 with different lifting operators: Harmonic varying α_u, Biharmonic bc1 and bc2. $\Delta t = 0.001$

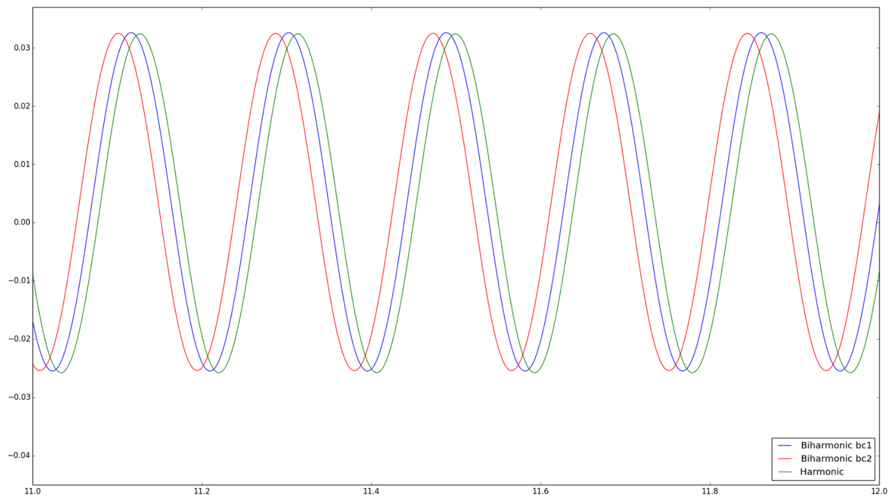


Figure 5.9: Displacement in y direction vs time for FSI3 with different lifting operators: Harmonic varying α_u, Biharmonic bc1 and bc2. $\Delta t = 0.001$

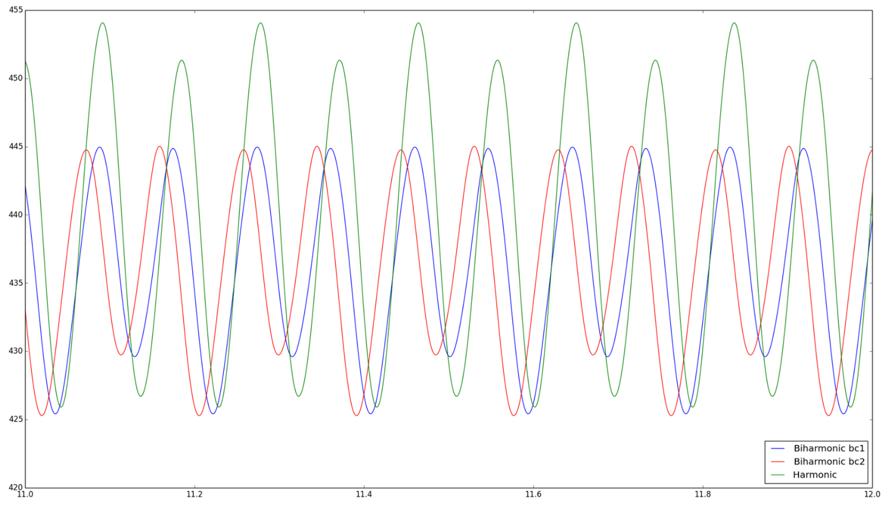


Figure 5.10: Drag vs time results for FSI3 with different lifting operators: Harmonic varying α_u, Biharmonic bc1 and bc2. $\Delta t = 0.001$

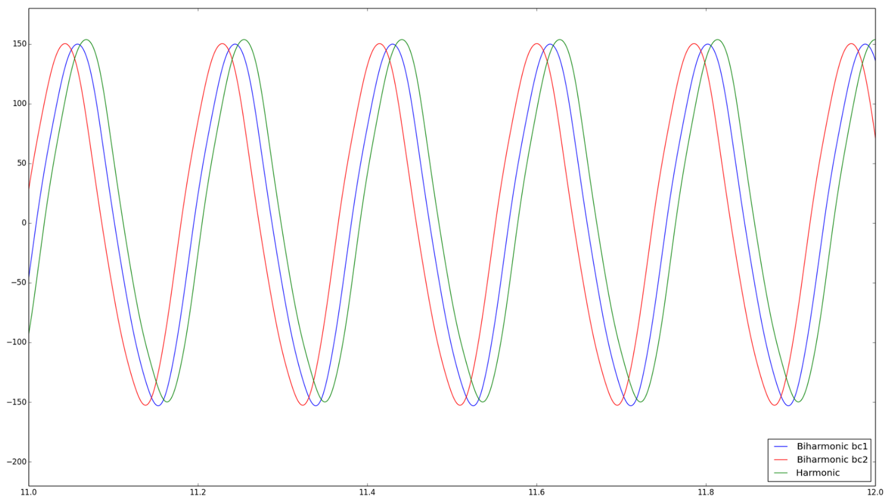


Figure 5.11: Lift vs time results for FSI3 with different lifting operators: Harmonic varying α_u, Biharmonic bc1 and bc2. $\Delta t = 0.001$

Discussion of Comparing Different Lifting Operators

In the FSI2 case all the different lifting operators show similar trend and it is seemingly not important for the results which lifting operator we use. This indicates that with a clever α_u, the harmonic technique can be chosen. This is an advantage since the harmonic techniques is the least computationally costly. Whilst when we increase the inflow velocity as in the FSI3 case there is a change in the period of the unsteady solution and the drag actually gives higher values, indicating that the biharmonic lifting operator may be a wiser choice.

In the FSI3 case there is an observed change in the drag values and the reason could be that the cells integrity are upheld in a different manner for different lifting operators. For the harmonic lifting operator it is reasonable to assume that for larger deformations the cell height on the interface will be smaller than for the biharmonic, hence giving a different value to the integrals when calculating drag. The lift and displacement differences for different lifting operators are similar. It is reasonable to assume that this is because of the normal force applied to the upper and lower sides of the bar, which is originally an effect of asymmetry in the y-direction of the domain, is what induces motion. The displacements are a secondary effect of the instability of the fluid and hence the effects we see in the values of lift are also seen in the displacements.

In short the lifting of the deformations into the fluid domain gives different cell structures which in turn effects the integral of the stress tensors on the interface. This in turns produces different results for problems with larger mesh deformations combined with high fluid velocities. This gives the conclusion that lifting operators are highly problem specific and for cases with large deformations lifting operators should be chosen with care.

A side note is on the computational cost, is that in the current implementation the harmonic lifting operator with a variable α_u can at this point not run in parallel, which all the other implemented lifting operators can. This concludes that even though the harmonic lifting operator is is less computationally costly, computing on multiple cores is faster with the biharmonic lifting operator.

5.2 Investigating Numerical Stability for Fluid-Structure Interaction Problems

The following section will give a brief insight in to the effects of choosing different θ values in the $\theta-$scheme for different time steps. The benchmark tests FSI2 and FSI3, as discussed in the previous section, has been investigated since they are known to be numerical unstable for certain values of θ

and Δt. Only the effects of Drag are reported as the three other quantities shows similar behavior. The impact of different θ values on energy stability in the solid mechanical benchmark CSM3 is also investigated.

Figure 5.12 shows the temporal evolution of drag in a simulation with $\Delta t = 0.01$. In the left panel the results are from simulations with $\theta = 0.5 + \Delta t$, and in the right panel $\theta = 0.5$. Based on the results we can observe that a standard Crank-Nicholson scheme becomes unstable when reaching $\sim$13 s. While the shifted Crank-Nicholson, $\theta = 0.5 + \Delta t$, is stable throughout the simulation.

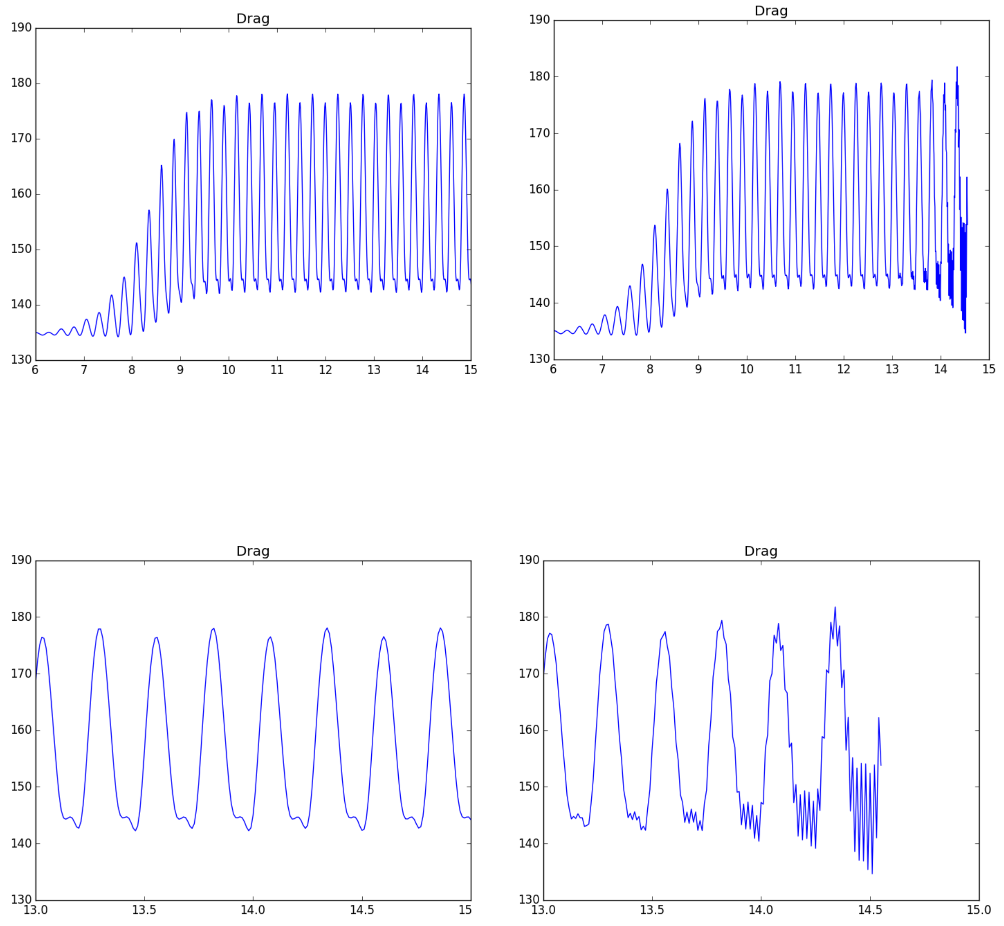


Figure 5.12: Drag vs time for FSI2 with $\Delta t = 0.01$ with different values, left panel with $\theta = 0.5 + \Delta t$ and right panel with $\theta = 0.5$

Figures 5.13 show drag for FSI3 simulation with $\Delta t = 0.001$ and $\theta = 0.5$, showing long term stability for the normal Crank-Nicholson scheme.

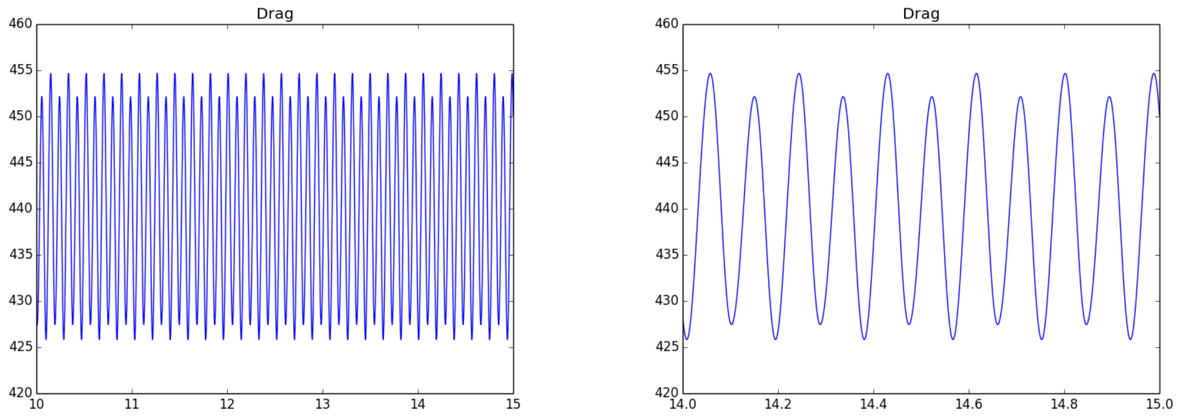


Figure 5.13: FSI3 drag vs time plot for 10-15 seconds and 14-15 seconds, with $\Delta t = 0.001$ and $\theta = 0.5$ showing long term numerical stability

For the CSM3 case only the solid bar is computed, and with an applied force g and no friction, the bar should move down and back up infinitely, for a correct solution.

Figure 5.2 shows plots of the displacements in x and y directions for $\theta = 0.5$ and 1. With the implicit scheme ($\theta = 1$) the bar moves to a steady state solution. This means energy has not been preserved and the energy dissipates. While in the Crank-Nicholson scheme ($\theta = 0.5$), the bar moves down and back up. This indicates that the Crank-Nicholson scheme is energy preserving.

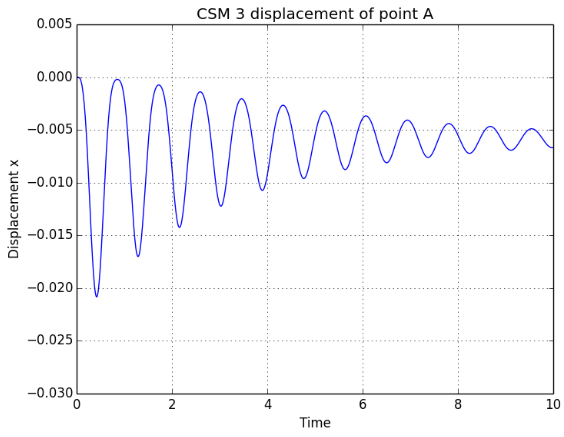


Figure 5.14: $\theta = 1$

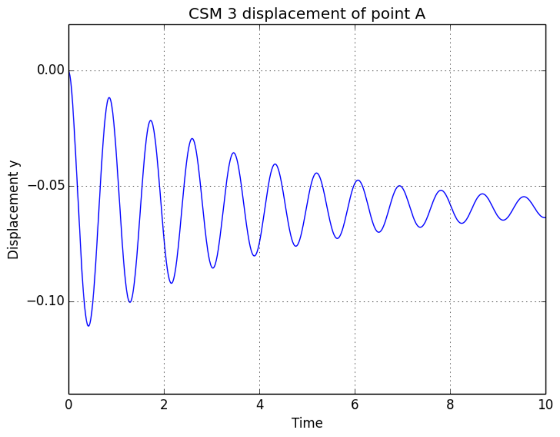


Figure 5.15: $\theta = 1$

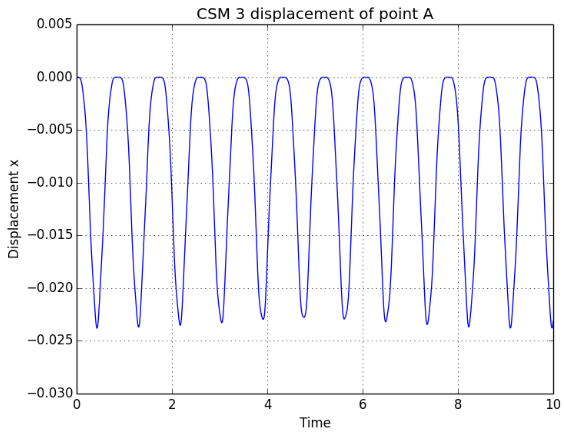


Figure 5.16: $\theta = 0.5$

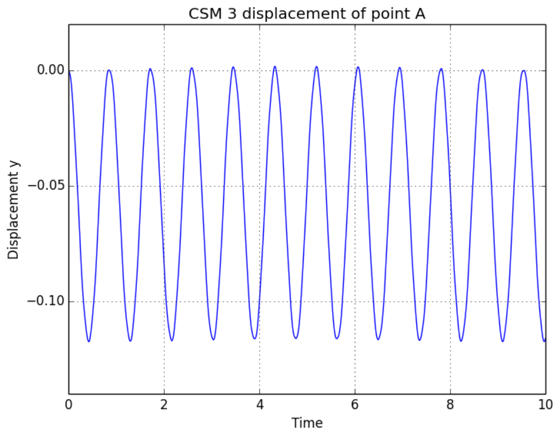


Figure 5.17: $\theta = 0.5$

Figure 5.18: CSM3 displacements with $\Delta t = 0.01$ with different values for θ

Discussion on numerical stability

The shifted version of the Crank-Nicholson scheme is stable when computing for time step values as low as $\Delta t = 0.01$. However with $\Delta t = 0.001$ the normal Crank-Nicholson scheme ($\theta = 0.5$) can be used and is long term stable in the period investigated here. It might be that after 100 s, that the flow would become unstable. However, a rigorous investigation of long-term stability is out of the scope of this thesis. It has also been reported by Wick 2011 [31] that the Crank-Nicholson, $\theta = 0.5$, scheme is stable throughout the computing time by setting $\Delta t = 0.001$.

In the FSI2 case the results for the finest mesh showed, in previous chapter, similar results for $\Delta t = 0.01$ and $\Delta t = 0.001$, meaning that the shifted version of the Crank-Nicholson scheme can be applied, in certain cases, with $\Delta t = 0.01$ greatly reducing computational runtime.

The CSM3 test shows that choosing $\theta = 0.5$ is crucial for preserving energy when computing solid problems. The same property will also be present in a FSI simulation, therefore it is crucial that an energy preserving scheme is applied, otherwise one does not have any control over the artificial numerical dissipation.

Chapter 6

Compute time reduction techniques

Computational fluid and solid mechanics are both well know to be time consuming or computationally intensive individually, and the iterative monolithic FSI problem is naturally an order of magnitude more expensive. As demonstrated in chapter 4, both accuracy and high resolutions are necessary to obtain adequate results. However, it was quickly observed that simulations took days to complete, even on meshes that were coarser than reported in this thesis. It was evident that simulations with adequate mesh and time step size would result in infeasible compute times.

The initial implementation was therefore profiled, and the Newton solver of the monolithic FSI problem was identified to spend the most compute time. The point of this chapter is not to rigorously experiment with multiple speed-up techniques, but merely to describe implementations of the compute time reduction techniques that made simulation times feasible. Retrospectively it was discovered that the same techniques were applied previously[22], but there only briefly addressed.

6.1 Newton runtime profile

Table 6.1 shows compute times for the FSI1 problem for the initial FEniCS implementation. The table contains the amount of time spent on assembly of the Jacobian, assembly of the residual, and a call to solve the linear system of equations. It can be observed that almost 95% of the resources are spent of assembly of the Jacobian, and efforts to reduce this bottleneck was taken. The next sections introduce two ways of speeding up assembly; the first is reuse of the Jacobian and the second is reduction of quadrature degree. The computational speedup will be compared to Table 6.1.

Method	**Runtime [s]**	Runtime [%]	**Calls**
Assembly of Jacobian	**60.7**	94.4	**5**
Assembly or residual	**0.6**	0.9	**5**
Solve	**2.8**	4.4	**5**
Fulltime	**64.3**	100%	-

Table 6.1: Newton solver timed with no optimizations, $\mathbf{\Delta t = 0.5}$

6.2 Reusing the Jacobian

As the time step size is relatively low in all the benchmark experiments, the effects of reassembling the Jacobian at every iteration was hypothesized to have a negligible impact on convergence. It was therefore tested whether a reuse of the Jacobian could speed up computations, by assembling only a few specified times. Table 6.2 shows the same simulation as presented in table 6.1, but now using $\Delta t = 0.5$. Even with a time step size that is fairly large, we obtain an acceptable decrease in compute time by -74%. The effect was an increase in the number of iterations, but and overall decreased compute time since the Jacobian was assembled only once.

Method	**Runtime [s]**	Runtime [%]	**Calls**
Assembly of Jacobian	**11.5 (-80%)**	68.7	**1 (-20%)**
Assembly or residual	**0.9 (+50%)**	5.8	**9 (+46%)**
Solve	**4.2 (+50%)**	25.0	**9 (+46%)**
Fulltime	**16.8 (-74 %)**	-	-

Table 6.2: Parts of the Newtonsolver timed with reuse of the jacobian run with $\mathbf{\Delta t = 0.5}$

6.3 Quadrature reduction

Assembly of the Jacobian matrix with non-linear terms induces a high number of quadrature points, dense matrix, and lower convergence rates. However, faster convergence rates can be obtained if the accuracy of the Jacobian is reduced by specifying the of quadrature degree. Table 6.3 shows that reducing the quadrature degree gives a 92 % decrease in time spent assembling the Jacobian even with the same number of calls. The full time spent went down by 87 %. The effect is therefore more iterations per time step, but an overall decrease in compute time as both convergence of the linear solvers and assembly of the residual, is much faster, respectively. However, it should be noted that reduction of the quadrature degree can lead to numerical divergence of the system for some problems.

Method	**Runtime [s]**	Runtime [%]	**Calls**
Assembly of Jacobian	**4.9 (-92%)**	60.3	**5 (0%)**
Assembly or residual	**0.5 (-17%)**	6.9	**5 (0%)**
Solve	**2.6 (-7%)**	31.9	**5 (0%)**
Fulltime	**8.2 (-87%)**	-	-

Table 6.3: Parts of the Newtonsolver timed with quadrature reduxe run with FSI1 ,$\mathbf{\Delta t = 0.5}$

6.4 Summary of runtime improvement techniques

Finally, a combination of Jacobian reuse and reduction of the quadrature is presented in Table 6.4, where the total compute time decreased by 89%. Both techniques were applied to the FSI2 and FSI3 problems as well.

Method	**Runtime [s]**	Runtime [%]	**Calls**
Assembly of Jacobian	**1.2 (-98%)**	18.1	**1 (-20%)**
Assembly or residual	**0.9 (+50%)**	14.7	**9 (+46%)**
Solve	**4.4 (+57%)**	66.2	**9 (+46%)**
Fulltime	**6.6 (-89%)**	-	-

Table 6.4: Newton solver timed with jacobian reuse and quadrature reduce run with $\mathbf{\Delta t = 0.5}$

Chapter 7

Conclusions

The intended work in this thesis was admittedly to develop a fast partitioned computational FSI framework to investigate possible high-frequent arterial wall vibrations in arteries in the vicinity of the brain, stemming from flow instabilities [25, 26]. The initial plan was to implement and use a monolithic FSI solver, but solely to obtain reference data. The idea was to implement partitioned scheme, specifically the scheme introduced by Fernandez 2013 [7]. However, it was soon discovered that there are many difficulties in implementing an accurate monolithic FSI solver that can handle large deformations and high fluid velocity. This complexity is reflected by the variability of the results provided in the appendix B, from highly renowned scientists. In addition, the scientific field of computational FSI is still in its infancy, and appropriate reference data is inaccessible. The work started out broadly, addressing monolithic and partitioned schemes simultaneously. However, work on the partitioned scheme was ultimately put aside and the focus during the last six months was on rigorous implementation, verification, and validation of a monolithic solver. Lessons were learned about the importance of lifting operators, energy stable numerical schemes and the need to compute on reference domains.

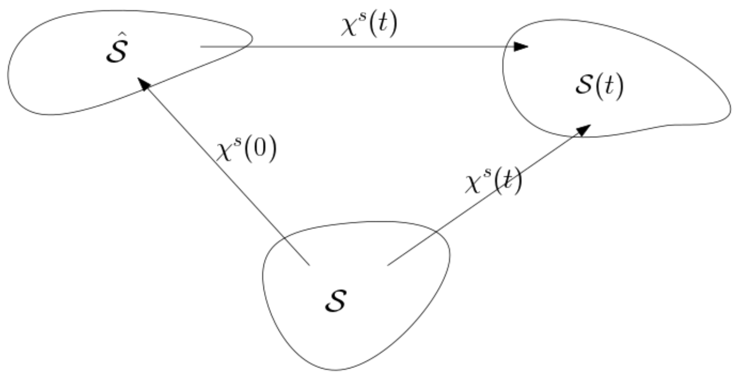
For the scientific field of computational FSI, it seems the most critical challenge to overcome is stability of partitioned schemes. The partitioned approach allows for used of legacy code for the structure and fluid parts, decreasing computing times significantly. The difficulty is still to transfer the stresses between the fluid and structure when each problem is solved sequentially. Explicit coupling schemes are known to be unconditionally unstable for standard Dirichlet-Neumann strategies when there is a large amount of added-mass in the system [8, 28]. There are however, schemes that offer added-mass free stability with explicit coupling, where the interface is treated through a Robin-Neumann coupling. The first scheme was developed for coupling of a thin walled structure[7]. Later a scheme was developed with an extension coupling with a thick walled structure, by the same main author [8]. The partitioned scheme for coupling thick walled structures is rather complex, but with the use of the associated code attached to this thesis, a monolithic solver can hopefully help others advance the field of FSI.

Appendices

Appendix A

Appendix

A.1 Lagrangian Description of Solid Mechanics



Let $\hat{\mathcal{S}}$, $\mathcal{S}$, $\mathcal{S}(t)$ be the initial stress free configuration of a given body, the reference and the current configuration respectively. I define a smooth mapping from the reference configuration to the current configuration:

$$\chi^s(\mathbf{X}, t) : \hat{\mathcal{S}} \to \mathcal{S}(t) \tag{A.1}$$

Where $\mathbf{X}$ denotes a material point in the reference domain and χ^s denotes the mapping from the reference configuration to the current configuration. Let $d^s(\mathbf{X}, t)$ denote the displacement field which describes deformation on a body. The mapping χ^s can then be specified from a current position plus the displacement from that position:

$$\chi^s(\mathbf{X}, t) = \mathbf{X} + d^s(\mathbf{X}, t) \tag{A.2}$$

which can be written in terms of the displacement field:

$$d^s(\mathbf{X},t) = \chi^s(\mathbf{X},t) - \mathbf{X} \tag{A.3}$$

Let w($\mathbf{X}$,t) be the domain velocity which is the partial time derivative of the displacement:

$$w(\mathbf{X},t) = \frac{\partial \chi^s(\mathbf{X},t)}{\partial t} \tag{A.4}$$

A.1.1 Deformation Gradient

The deformation gradient describes the rate at which a body undergoes deformation. Let $d(\mathbf{X},t)$ be a differentiable deformation field in a given body, the deformation gradient is then:

$$F = \frac{\partial \chi^s(\mathbf{X},t)}{\partial \mathbf{X}} = \frac{\partial \mathbf{X} + d^s(\mathbf{X},t)}{\partial \mathbf{X}} = I + \nabla d(\mathbf{X},t) \tag{A.5}$$

which denotes relative change of position under deformation in a Lagrangian frame of reference. We can observe that when there is no deformation. The deformation gradient F is simply the identity matrix.

Let the Jacobian determinant, which is the determinant of the of the deformation gradient F, be defined as:

$$J = \det(F) \tag{A.6}$$

The Jacobian determinant is used to change between volumes, assuming infinitesimal line and area elements in the current ds, dx and reference dV, dX configurations. The Jacobian determinant is therefore known as a volume ratio.

A.1.2 Strain

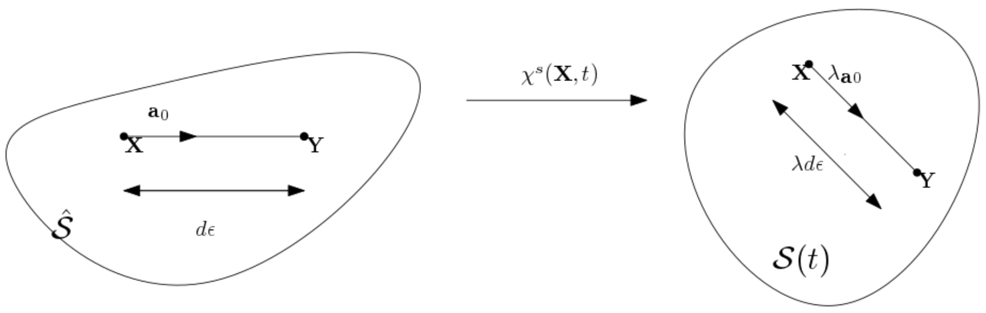


Figure A.1: Deformation of a line element with length $d\epsilon$ into a line element with length $\lambda d\epsilon$

Strain is the relative change of location between two particles. Strain, strain rate and deformation is used to describe the relative motion of particles in a continuum. This is the fundamental quality that causes stress [18].

Observing two neighboring points $\mathbf{X}$ and $\mathbf{Y}$. Let $\mathbf{Y}$ be described by adding and subtracting the point $\mathbf{X}$ and rewriting $\mathbf{Y}$ from the point $\mathbf{X}$ plus a distance $d\mathbf{X}$:

$$\mathbf{Y} = \mathbf{Y} + \mathbf{X} - \mathbf{X} = \mathbf{X} + |\mathbf{Y} - \mathbf{X}|\frac{\mathbf{Y} - \mathbf{X}}{|\mathbf{Y} - \mathbf{X}|} = \mathbf{X} + d\mathbf{X} \tag{A.7}$$

Let $d\mathbf{X}$ be denoted by:

$$d\mathbf{X} = d\epsilon\mathbf{a}_0 \tag{A.8}$$
$$d\epsilon = |\mathbf{Y} - \mathbf{X}| \tag{A.9}$$
$$\mathbf{a}_0 = \frac{\mathbf{Y} - \mathbf{X}}{|\mathbf{Y} - \mathbf{X}|} \tag{A.10}$$

where $d\epsilon$ is the distance between the two points and $\mathbf{a}_0$ is a unit vector

We see now that $d\mathbf{X}$ is the distance between the two points times the unit vector.

A certain motion transform the points $\mathbf{Y}$ and $\mathbf{X}$ into the displaced positions $\mathbf{x} = \chi^s(\mathbf{X}, t)$ and $\mathbf{y} = \chi^s(\mathbf{Y}, t)$. Using Taylor‘s expansion $\mathbf{y}$ can be expressed in terms of the deformation gradient:

$$\begin{aligned}\mathbf{y} =&\chi^s(\mathbf{Y}, t) = \chi^s(\mathbf{X} + d\epsilon\mathbf{a}_0, t) && \text{(A.11)}\\ =&\chi^s(\mathbf{X}, t) + d\epsilon F\mathbf{a}_0 + \mathcal{O}(\mathbf{Y} - \mathbf{X}) && \text{(A.12)}\end{aligned}$$

where $\mathcal{O}(\mathbf{Y} - \mathbf{X})$ refers to the small error that tends to zero faster than $(\mathbf{X} - \mathbf{Y}) \to \mathcal{O}$.

Setting $\mathbf{x} = \chi^s(\mathbf{X}, t)$ It follows that:

$$\begin{aligned}\mathbf{y} - \mathbf{x} =&d\epsilon F\mathbf{a}_0 + \mathcal{O}(\mathbf{Y} - \mathbf{X}) && \text{(A.13)}\\ =&F(\mathbf{Y} - \mathbf{X}) + \mathcal{O}(\mathbf{Y} - \mathbf{X}) && \text{(A.14)}\end{aligned}$$

Let the **stretch vector** be $\lambda_{\mathbf{a}0}$, which goes in the direction of $\mathbf{a}_0$:

$$\lambda_{\mathbf{a}0}(\mathbf{X}, t) = F(\mathbf{X}, t)\mathbf{a}_0 \tag{A.15}$$

Looking at the square of λ:

$$\begin{aligned}\lambda^2 &= \lambda_{\mathbf{a}0}\lambda_{\mathbf{a}0} = F(\mathbf{X}, t)\mathbf{a}_0 F(\mathbf{X}, t)\mathbf{a}_0 && \text{(A.16)}\\ &= \mathbf{a}_0 F^T F\mathbf{a}_0 = \mathbf{a}_0 C\mathbf{a}_0 && \text{(A.17)}\end{aligned}$$

Introducing the right Cauchy-Green tensor:

$$C = F^T F \tag{A.18}$$

Since $\mathbf{a}_0$ is just a unit vector, we see that C measures the squared length of change under deformation. We see that in order to determine the stretch one needs only the direction of $\mathbf{a}_0$ and the tensor C. C is also symmetric and positive definite $C = C^T$. I also introduce the Green-Lagrangian strain tensor E:

$$E = \frac{1}{2}(C - I) \tag{A.19}$$

which is also symmetric since C and I are symmetric. The Green-Lagrangian strain tensor E has the advantage of having no contributions when there is no deformations. Where the Cauchy-Green tensor gives the identity matrix for zero deformation.

A.1.3 Stress

While strain, deformation and strain rate only describe the relative motion of particles in a given volume, stress give us the internal forces between neighboring particles. Stress is responsible for deformation and is therefore crucial in continuum mechanics. The unit of stress is force per area.

Introducing the Cauchy stress tensor σ_s, which define the state of stress inside a material. The version of Cauchy stress tensor is defined by the material model used. If we use this tensor on an area, taking the stress tensor times the normal vector $\sigma_s \mathbf{n}$ we get the forces acting on that area.

Stress tensor defined from the Cauchy by the constitutive law of St. Venant-Kirchhoff hyperelastic material model:

$$\sigma_s = \frac{1}{J} F(\lambda_s (trE)I + 2\mu_s E)F^T \tag{A.20}$$

Using the deformation gradient and the Jacobian determinant, we get the first Piola-Kirchhoff stress tensor P:

$$P = J\sigma F^{-T} \tag{A.21}$$

This is known as the *Piola Transformation* and maps the tensor into a Lagrangian formulation which will be used when stating the solid equation.

Introducing the second Piola-Kirchhoff stress tensor S:

$$S = JF^{-1}\sigma F^{-T} = F^{-1}P = S^T \tag{A.22}$$

from this relation the first Piola-Kirchhoff tensor can be expressed by the second:

$$P = FS \tag{A.23}$$

Appendix B

Results from Renowned Scientists

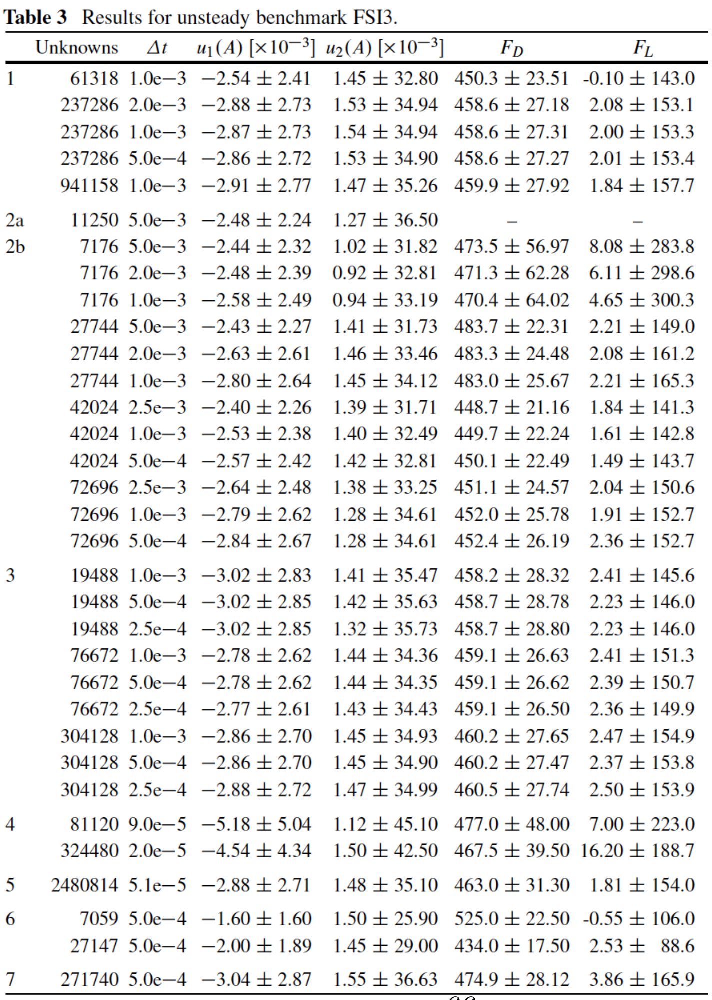


Table 3 Results for unsteady benchmark FSI3.

	Unknowns	Δt	$u_1(A)$ [$\times 10^{-3}$]	$u_2(A)$ [$\times 10^{-3}$]	F_D	F_L
1	61318	1.0e−3	-2.54 ± 2.41	1.45 ± 32.80	450.3 ± 23.51	-0.10 ± 143.0
	237286	2.0e−3	-2.88 ± 2.73	1.53 ± 34.94	458.6 ± 27.18	2.08 ± 153.1
	237286	1.0e−3	-2.87 ± 2.73	1.54 ± 34.94	458.6 ± 27.31	2.00 ± 153.3
	237286	5.0e−4	-2.86 ± 2.72	1.53 ± 34.90	458.6 ± 27.27	2.01 ± 153.4
	941158	1.0e−3	-2.91 ± 2.77	1.47 ± 35.26	459.9 ± 27.92	1.84 ± 157.7
2a	11250	5.0e−3	-2.48 ± 2.24	1.27 ± 36.50	–	–
2b	7176	5.0e−3	-2.44 ± 2.32	1.02 ± 31.82	473.5 ± 56.97	8.08 ± 283.8
	7176	2.0e−3	-2.48 ± 2.39	0.92 ± 32.81	471.3 ± 62.28	6.11 ± 298.6
	7176	1.0e−3	-2.58 ± 2.49	0.94 ± 33.19	470.4 ± 64.02	4.65 ± 300.3
	27744	5.0e−3	-2.43 ± 2.27	1.41 ± 31.73	483.7 ± 22.31	2.21 ± 149.0
	27744	2.0e−3	-2.63 ± 2.61	1.46 ± 33.46	483.3 ± 24.48	2.08 ± 161.2
	27744	1.0e−3	-2.80 ± 2.64	1.45 ± 34.12	483.0 ± 25.67	2.21 ± 165.3
	42024	2.5e−3	-2.40 ± 2.26	1.39 ± 31.71	448.7 ± 21.16	1.84 ± 141.3
	42024	1.0e−3	-2.53 ± 2.38	1.40 ± 32.49	449.7 ± 22.24	1.61 ± 142.8
	42024	5.0e−4	-2.57 ± 2.42	1.42 ± 32.81	450.1 ± 22.49	1.49 ± 143.7
	72696	2.5e−3	-2.64 ± 2.48	1.38 ± 33.25	451.1 ± 24.57	2.04 ± 150.6
	72696	1.0e−3	-2.79 ± 2.62	1.28 ± 34.61	452.0 ± 25.78	1.91 ± 152.7
	72696	5.0e−4	-2.84 ± 2.67	1.28 ± 34.61	452.4 ± 26.19	2.36 ± 152.7
3	19488	1.0e−3	-3.02 ± 2.83	1.41 ± 35.47	458.2 ± 28.32	2.41 ± 145.6
	19488	5.0e−4	-3.02 ± 2.85	1.42 ± 35.63	458.7 ± 28.78	2.23 ± 146.0
	19488	2.5e−4	-3.02 ± 2.85	1.32 ± 35.73	458.7 ± 28.80	2.23 ± 146.0
	76672	1.0e−3	-2.78 ± 2.62	1.44 ± 34.36	459.1 ± 26.63	2.41 ± 151.3
	76672	5.0e−4	-2.78 ± 2.62	1.44 ± 34.35	459.1 ± 26.62	2.39 ± 150.7
	76672	2.5e−4	-2.77 ± 2.61	1.43 ± 34.43	459.1 ± 26.50	2.36 ± 149.9
	304128	1.0e−3	-2.86 ± 2.70	1.45 ± 34.93	460.2 ± 27.65	2.47 ± 154.9
	304128	5.0e−4	-2.86 ± 2.70	1.45 ± 34.90	460.2 ± 27.47	2.37 ± 153.8
	304128	2.5e−4	-2.88 ± 2.72	1.47 ± 34.99	460.5 ± 27.74	2.50 ± 153.9
4	81120	9.0e−5	-5.18 ± 5.04	1.12 ± 45.10	477.0 ± 48.00	7.00 ± 223.0
	324480	2.0e−5	-4.54 ± 4.34	1.50 ± 42.50	467.5 ± 39.50	16.20 ± 188.7
5	2480814	5.1e−5	-2.88 ± 2.71	1.48 ± 35.10	463.0 ± 31.30	1.81 ± 154.0
6	7059	5.0e−4	-1.60 ± 1.60	1.50 ± 25.90	525.0 ± 22.50	-0.55 ± 106.0
	27147	5.0e−4	-2.00 ± 1.89	1.45 ± 29.00	434.0 ± 17.50	2.53 ± 88.6
7	271740	5.0e−4	-3.04 ± 2.87	1.55 ± 36.63	474.9 ± 28.12	3.86 ± 165.9

Figure B.1: Results from different contributions in from the paper Turek et.al 2010 [24]

Appendix C

Implementation of Fluid-Structure Interaction in FEniCS

The discretization described in chapter 3.5, is implemented using FEniCS [13]. FEniCS is a platform used to solve partial differential equations. Code is written in python, using FEniCS to easily run efficient finite element code.

The complete code consist of many hundreds of lines of code, and therefore only the most essential parts are covered. The code that has been added is added so that a reader with a minimal skill set in scientific computing and basic knowledge of the finite element method could implement a version of the code.

C.1 Variational Formulation

The variational form can be written directly into FEniCS. A big advantage in FEniCS is that there is a small "gap" between mathematical notation and FEniCS syntax. For instance the gradient and divergence, is in FEniCS as written as grad and div respectively. For this reason the basic parts of the code should be self evident.

The code structure has a main script named monolithic.py which from command line takes in arguments specifying the problem to be solved, the version of the variational form and the Newtonsolver. The main script gather the called parts from specific folders. The problem folder contains the specific problem, which contains the necessary boundary conditions, mesh, parameters for fluid an structure, and saves the solutions and other data in a post processing function. The variational form folder contains the fluid and solid variational form, which given in the command line takes a value for θ and the chosen lifting operator. The Newtonsolver folder contains different versions of the Newtonsolver, which determines which one of the speedups, outlined in chapter 6, to be implemented.

The main script monolithic.py creates the functions, functionspaces and vectorfunction spaces needed. The main script contains the time loop which iterates in time calling the variational form, the newton solver and updating the functions in time.

The vector functions and functions such as the displacement, velocity, and pressure, are written with a script “n” meaning at which time the function is valued.

```
d_["n"]  """deformation in the current timestep """
u_["n-1"] """velocity in the last timestep """
p_["n-2"] """pressure in the second last timestep """
```

We define the linear and nonlinear parts of the fluid and solid variational. All the linear and nonlinear parts are added together to create the full variational form.

psi, phi and gamma are the test functions: ψ, ϕ and γ.

```
J_theta = theta*J_(d_["n"]) + (1 - theta)*J_(d_["n-1"])

F_fluid_linear = rho_f/k*inner(J_theta*(v_["n"] - v_["n-1"]), psi)*dx_f

F_fluid_nonlinear =  Constant(theta)*rho_f*\
inner(J_(d_["n"])*grad(v_["n"])*inv(F_(d_["n"]))*v_["n"], psi)*dx_f

F_fluid_nonlinear += inner(J_(d_["n"])*sigma_f_p(p_["n"], d_["n"])*\
inv(F_(d_["n"])).T, grad(psi))*dx_f

F_fluid_nonlinear += Constant(theta)*inner(J_(d_["n"])\
*sigma_f_u(v_["n"], d_["n"], mu_f)*inv(F_(d_["n"])).T, grad(psi))*dx_f

F_fluid_nonlinear += Constant(1 - theta)*inner(J_(d_["n-1"])*\
sigma_f_u(v_["n-1"], d_["n-1"], mu_f)*inv(F_(d_["n-1"])).T, grad(psi))*
    dx_f

F_fluid_nonlinear += \
inner(div(J_(d_["n"])*inv(F_(d_["n"]))*v_["n"]), gamma)*dx_f

F_fluid_nonlinear += Constant(1 - theta)*rho_f*\
inner(J_(d_["n-1"])*grad(v_["n-1"])*inv(F_(d_["n-1"]))*v_["n-1"], psi)*
    dx_f

F_fluid_nonlinear -= rho_f*inner(J_(d_["n"])*\
grad(v_["n"])*inv(F_(d_["n"]))*((d_["n"]-d_["n-1"])/k), psi)*dx_f

delta = 1E10
F_solid_linear = rho_s/k*inner(v_["n"] - v_["n-1"], psi)*dx_s +\
delta*(1/k)*inner(d_["n"] - d_["n-1"], phi)*dx_s -\
delta*inner(Constant(theta)*v_["n"] + Constant(1-theta)*v_["n-1"], phi)*
    dx_s

F_solid_nonlinear = inner(Piola1(Constant(theta)*d_["n"] +\
Constant(1 - theta)*d_["n-1"], lamda_s, mu_s), grad(psi))*dx_s
```

C.2 Newtons Method Implementation for Solving Fluid-Structure Interaction in FEniCS

To handle the non-linearities in the scheme we use a Newton solver. FEniCS already has a built-in Newtonsolver, however this solver was not able to compute the monolithic FSI problem because of the pressure function, with our choice of solid stress tensor, only being defined in the fluid domain. We had to manipulate the matrix to ensure that the pressure was zero in the solid domain. This gave the need to implement our own Newtonsolver taking ideas from Mikael Mortensens course in CFD named MEK4300 at the University of Oslo [30]. Following is a print out of the full Newtonsolver, that is called for each time iteration.

```
def newtonsolver(F, J_nonlinear, A_pre, A, b, bcs, \
                dvp_, up_sol, dvp_res, rtol, atol, max_it, T, t, **
    monolithic):
    Iter      = 0  """ Setting initial values """
    residual   = 1
    rel_res    = residual
    lmbda = 1 """  """
    while rel_res > rtol and residual > atol and Iter < max_it:
        if Iter % 10 == 0: """Assembles the Jacobian for each tenth
    round, in this instance. """
            A = assemble(J_nonlinear, tensor=A, form_compiler_parameters
     = {"quadrature_degree": 4}) """ Assembles the Jacobian with
    reduction of the quadrature """
            A.axpy(1.0, A_pre, True)
            A.ident_zeros()  """ Sets values of zero to 1 to ensure zero
     pressure in the solid domain """

        b = assemble(-F, tensor=b) """ assembling the residual  """

        [bc.apply(A, b, dvp_["n"].vector()) for bc in bcs] """ Applies
    boundary conditions to the mixed function dvp """
        up_sol.solve(A, dvp_res.vector(), b) """ Solves the matrix
    equation A * dvp = b """
        dvp_["n"].vector().axpy(lmbda, dvp_res.vector())   """ A fast
    """
        [bc.apply(dvp_["n"].vector()) for bc in bcs]
        rel_res = norm(dvp_res, 'l2')
        residual = b.norm('l2')
        if isnan(rel_res) or isnan(residual):
            print "type rel_res: ",type(rel_res)
            t = T*T

        if MPI.rank(mpi_comm_world()) == 0:   """ Prints only out the
    numeber 0 process when running code in paralell  """
            print "Newton iteration %d: r (atol) = %.3e (tol = %.3e), r
    (rel) = %.3e (tol = %.3e) " \
        % (Iter, residual, atol, rel_res, rtol)
        Iter += 1

    return dict(t=t)
```

Bibliography

[1] K. Yusuf Billah. Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks. *American Journal of Physics*, 59(2):118, 1991.

[2] David J Charlesworth. Solution of the Incompressible Navier- Stokes Equations on Unstructured Meshes by. (August), 2003.

[3] Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

[4] Stephane. Étienne, Andre Garon, and Dominique. Pelletier. Some manufactured solutions for verification of fluid-structure interaction codes. *Computers and Structures*, 106-107:56–67, 2012.

[5] Stéphane Étienne, D Tremblay, and Dominique Pelletier. Code Verification and the Method of Manufactured Solutions for Fluid-Structure Interaction Problems. *36th AIAA Fluid Dynamics Conference and Exhibit*, (June):1–11, 2006.

[6] Charles L. Fefferman. Existence and smoothness of the Navier-Stokes equation. *The millennium prize problems*, (1):1–5, 2000.

[7] Miguel A. Fernández, Jimmy Mullaert, and Marina Vidrascu. Explicit robin-neumann schemes for the coupling of incompressible fluids with thin-walled structures. *Computer Methods in Applied Mechanics and Engineering*, 267:566–593, 2013.

[8] Miguel A. Fernández, Jimmy Mullaert, and Marina Vidrascu. Generalized Robin-Neumann explicit coupling schemes for incompressible fluid-structure interaction: Stability analysis and numerics. *International Journal for Numerical Methods in Engineering*, 101(3):199–229, 2015.

[9] Xiaoyi He and Li-Shi Luo. Lattice boltzmann model for the incompressible navier–stokes equation. *Journal of Statistical Physics*, 88(3):927–944, 1997.

[10] Gerhard Holzapfel. Nonlinear solid mechanics: A continuum approach for engineering, 2000.

[11] Jaroslav Hron and Stefan Turek. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. *Fluid-Structure Interaction*, 53:371–385, 2006.

[12] Jie Liu, Rajeev K. Jaiman, and Pardha S. Gurugubelli. A stable second-order scheme for fluid-structure interaction with strong added-mass effects. *Journal of Computational Physics*, 270:687–710, 2014.

[13] Anders Logg, Harish Narayanan, Marie Rognes, Johannes Ring, Kristian B. Ølgaard, and Garth N. Wells. FEniCS Project, 2011.

[14] Cm Macal. Proceedings of the 2005 Winter Simulation Conference ME Kuhl, NM Steiger, FB Armstrong, and JA Joines, eds. *Simulation*, pages 1643–1649, 2005.

[15] Selim MM and Koomullil RP. Mesh Deformation Approaches – A Survey. *Journal of Physical Mathematics*, 7(2), 2016.

[16] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge, 2010.

[17] William L Oberkampf and Timothy G Trucano. Verification and validation benchmarks. *Nuclear engineering and Design*, 238(3):716–743, 2008.

[18] Thomas Richter. Numerical Methods for Fluid-Structure Interaction Problems. *Course of lectures*, 2010(August 2010), 2010.

[19] Thomas Richter and Thomas Wick. On time discretizations of Fluid-structure interactions. *Multiple Shooting and Time Domain Decomposition MEthods*, pages 377–400, 2013.

[20] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, 124(1):4, 2002.

[21] Michael. Schäfer, Stefan. Turek, Franz. Durst, E. Krause, and Rolf. Rannacher. Benchmark Computations of Laminar Flow Around a Cylinder. pages 547–566, 1996.

[22] Natural Sciences. A Newton ' s Method Finite Element Algorithm for Fluid-Structure Interaction. (October), 2012.

[23] K. Stein, Tayfun. Tezduyar, and R. Benney. Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements. *Journal of Applied Mechanics*, 70(1):58, 2003.

[24] Stefan Turek, Jaroslav Hron, M Razzaq, H Wobker, and M Sch. Fluid Structure Interaction II. 73, 2010.

[25] Kristian Valen-Sendstad, Kent-André Mardal, Mikael Mortensen, Bjørn Anders Pettersson Reif, and Hans Petter Langtangen. Direct numerical simulation of transitional flow in a patient-specific intracranial aneurysm. *Journal of biomechanics*, 44(16):2826–2832, 2011.

[26] Kristian Valen-Sendstad and DA Steinman. Mind the gap: impact of computational fluid dynamics solution strategy on prediction of intracranial aneurysm hemodynamics and rupture status indicators. *American Journal of Neuroradiology*, 35(3):536–543, 2014.

[27] Boris Valkov, Chris H Rycroft, and Ken Kamrin. Eulerian method for fluid – structure interaction and submerged solid – solid contact problems.

[28] Harald. van Brummelen. Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction. *Journal of Applied Mechanics*, 76(2):021206, 2009.

[29] Frank M White. Viscous Fluid Flow Viscous. *New York*, Second:413, 2000.

[30] Frank M. White. Chapter 3 - Solutions of the Newtonian viscous-flow equa- tions. *Viscous Fluid Flow*, (5), 2006.

[31] Thomas Wick. Adaptive Finite Element Simulation of Fluid-Structure Interaction with Application to Heart-Valve Dynamics. *Institute of Applied Mathematics, University of Heidelber*, page 157, 2011.

[32] Thomas Wick. Fluid-structure interactions using different mesh motion techniques. *Computers and Structures*, 89(13-14):1456–1467, 2011.