

**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.

## Acknowledgments

First of I want to thank by brilliant supervisors Kristian Valen-Sendstad 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, bar none.

## Abstract

This thesis has developed a framework for solving Fluid-Structure Interaction (FSI) problem using a monolithic scheme. Introducing the mathematics and physics that govern fluid and structures, and the conditions needed to model FSI problems. I have discussed the challenges one meets when trying to model and compute FSI problems. I have introduced the  $\theta$ -scheme meant to uphold stability for long time FSI problems. The code has been verified in parts using MMS.

Using the proposal benchmark from Hron and Turek 2010, I have validated the FSI solver first in fluid and structure separately and finally for a full FSI problem. I have compared data to contributions made by others imposing similar schemes. I have shown how crucial the choice of lifting operator is for computing FSI problems with moderate to large deformations, and the need for a stable scheme for long run times.

Investigations have been made into the effects of different lifting operators and different ordered numerical schemes.



# Contents

<b>1</b>	<b>Introduction to Fluid-Structure Interaction</b>	<b>1</b>
<b>2</b>	<b>Continuum mechanics in different frames of reference</b>	<b>3</b>
2.1	Conservation of mass and momentum for solid matter . . . . .	4
2.2	Conservation of mass and momentum for fluid matter . . . . .	5
2.3	Fluid and Structure Boundary conditions . . . . .	6
<b>3</b>	<b>Fluid-Structure Interaction Problem Formulation</b>	<b>7</b>
3.1	Mapping between different frames of reference . . . . .	9
3.2	Governing equations for Fluid-Structure Interaction . . . . .	10
3.2.1	Derivatives in different frameworks . . . . .	10
3.2.2	Solid equation . . . . .	11
3.2.3	Fluid equations . . . . .	11
3.3	Interface conditions at the boundary . . . . .	12
3.4	Methods for domain representation and mesh quality preservation . . . . .	13
3.4.1	Harmonic lifting operator . . . . .	14
3.4.2	Biharmonic lifting operator . . . . .	14
3.5	Discretization of monolithic Fluid-Structure Interaction equations . . . . .	15
<b>4</b>	<b>Verification and Validation of the Fluid Structure Interaction Implementation.</b>	<b>19</b>
4.1	Verification . . . . .	20
4.1.1	Method of Manufactured Solution on the implementation of the Solid equation . . . . .	21
4.1.2	MMS of Fluid equations with prescribed motion . . . . .	23
4.2	Validation . . . . .	25
4.2.1	Fluid-Structure Interaction between an elastic object and laminar incompressible flow . . . . .	26

<b>5</b>	<b>Comparing the effects of different lifting operators and Investigating Numerical Stability in Fluid-Structure Interaction problems</b>	<b>41</b>
5.1	Methods for comparing lifting operators . . . . .	42
5.2	Investigating Numerical stability for Fluid-Structure Interaction Problems . . . . .	49
<b>6</b>	<b>CPU time reduction techniques</b>	<b>55</b>
6.1	Newton runtime profile . . . . .	55
6.2	Reusing the Jacobian . . . . .	56
6.3	Quadrature reduce . . . . .	57
6.4	Summary of runtime improvement techniques . . . . .	57
<b>7</b>	<b>Conclusions</b>	<b>59</b>
	<b>Appendices</b>	<b>61</b>
<b>A</b>	<b>Appendix</b>	<b>63</b>
A.1	Lagrangian description of solid mechanics . . . . .	63
A.1.1	Deformation gradient . . . . .	64
A.1.2	Strain . . . . .	65
A.1.3	Stress . . . . .	67
<b>B</b>	<b>Results</b>	<b>69</b>
<b>C</b>	<b>Implementation of Fluid Structure Interaction in FEniCS</b>	<b>71</b>
C.1	Variational form . . . . .	71
C.2	Newtons method implementation for solving Fluid structure interaction in FEniCS . . . . .	74

# Chapter 1

## Introduction to Fluid-Structure Interaction

The interaction between fluids and structures can be observed all around us in nature. Flags waving in the wind, windmills, and inhalation of air into the lungs. It is rather intuitive that the fluids and structures must exert mutual 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. 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  $\frac{\rho_f}{\rho_s} \ll 1$ , the structural deformations will be small and the FSI problem “easy”. However, modeling arterial FSI deems more challenging as the density of the fluid (blood) and the structure (artery) is similar, resulting in large deformations of the arteries. Large deformations are challenging to model and require hyperelastic structure models, and energy stable numeri-



Figure 1.1: Tacoma Narrows bridge still standing with large deformations

Figure 1.2: Tacoma Narrows bridge collapsed

cal 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 biomechanics, namely large deformations. The effects of different numerical schemes and approaches will be investigated to maintain acceptable accuracy. The framework will be verified using the *Method of Manufactured Solutions*, and validated using a classic, well-known benchmark, known to be especially challenging.



# 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. The associated characteristic length scale far smaller than most things of interest 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.

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.

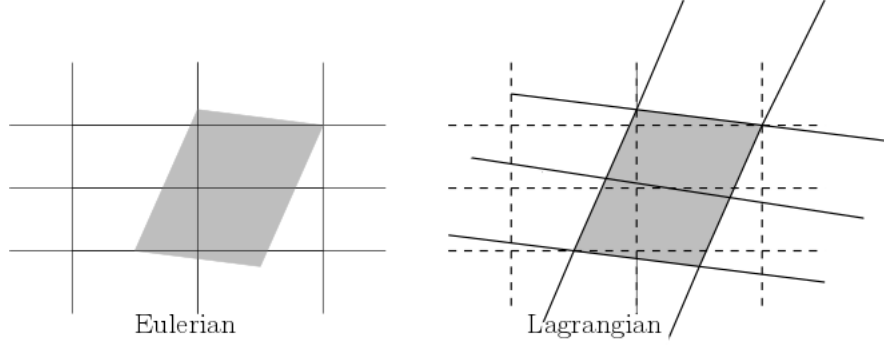


Figure 2.1: Comparison between the Eulerian and the Lagrangian description, the lines represent the grid and the gray represents the matter.

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 matter

With matter assumed continuous, 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 added through constitutive relations. The differential solid equation will be stated in the Lagrangian reference system [8], in the solid domain  $\mathcal{S}$  as:

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

written in terms of the deformation  $\mathbf{d}$ , of the solid.  $\mathbf{P}$  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.  $\rho_s$  is the solid density, and  $\frac{\partial}{\partial t^2}$  is the second time derivative.

## 2.2 Conservation of mass and momentum for fluid matter

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. In the Eulerian framework the domain has fixed spatial points where the fluid passes through. The N-S equations are written in the fluid time domain  $\mathcal{F}(t)$  as an incompressible fluid:

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

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

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

There does not yet exist an analytical solutions to the N-S 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 [25]. Actually there is a prize set out by the Clay Mathematics Institute of 1 million dollars to whomever can show the existence and smoothness of Navier-Stokes [5], as a part of their millennium problems. Nonetheless this does not stop us from discretizing and solving N-S numerically. One difficulty in the N-S 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 at the inlet of a pipe. For a problem to be well posed we need to prescribe initial conditions. The Dirichlet conditions are defined for  $\mathbf{u}$  and  $p$  as :

$$\mathbf{u} = u_0 \text{ on } \partial\mathcal{F}_D \quad (2.4)$$

$$p = p_0 \text{ on } \partial\mathcal{F}_D \quad (2.5)$$

$$\mathbf{d} = d_0 \text{ on } \partial\mathcal{S}_D \quad (2.6)$$

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

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  $\mathbf{f}$ :

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

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

For the sake of completeness, it should be noticed that there exists the Roben boundary conditions, which is 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.

In describing the FSI problems the domain is split into three: fluid, structure, and interface. 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 [10]. 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 [23], 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 frame 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 nu-

merical stability for problems with strong added-mass effects [10], and are fully coupled. The disadvantage over the partitioned approach is that we lose flexibility when solving many equations simultaneously, and the computing matrices can quickly become large, hence computationally costly and needing large memory. 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 [?].

### 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  $\chi^s$  and  $\chi^f$  to the time domain denoted as  $\mathcal{S}$  and  $\mathcal{F}$ . While the interface in the time domain is denoted as  $\Sigma$

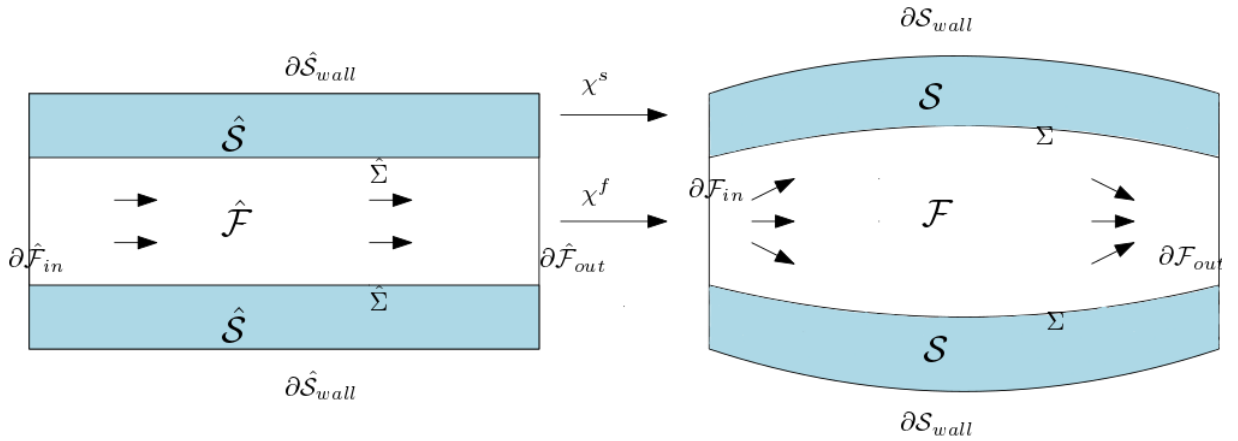


Figure 3.1: Illustration of domain mapping

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  $\mathbf{u}$  will be mapped with the deformation gradient, shown in equation 3.2. Lastly the divergence of a vector  $\mathbf{u}$  will be mapped in a slightly different manner, shown in equation 3.3.

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

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

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

## 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 3.4-3.6, to understand the need to change the the convective term in the fluid equation when stated in an ALE description [27]:

### 3.2.1 Derivatives in different frameworks

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

$$D_t f(x, t) = \partial_t f(x, t) \quad (3.4)$$

The total and partial derivatives have the following relation in the Eulerian framework, where  $\mathbf{u}$  is the convecting velocity:



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

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

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

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 2.1 which was defined w.r.t the deformation  $\mathbf{d}$ . Defining the solid equation w.r.t to the deformation  $\mathbf{d}$  can be 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:

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

### 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:

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

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:

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

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

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

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

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

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

$$\rho_f J \left( \frac{\partial \mathbf{u}}{\partial t} dx + (\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) \right) = \nabla \cdot (J F^{-1} \hat{\sigma}_f) + J \rho_f f \quad (3.14)$$

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

### 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 [16]:

- *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:

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

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 continuous 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 [28]. 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 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.

$$-\alpha_u \nabla^2 \mathbf{d} = 0 \quad \text{in } \hat{\mathcal{F}} \quad (3.16)$$

$$\mathbf{d} = 0 \quad \text{on } \partial \hat{\mathcal{F}} / \hat{\Sigma} \quad (3.17)$$

$$\mathbf{d}_f = \mathbf{d}_s \quad \text{on } \hat{\Sigma} \quad (3.18)$$

When using the harmonic lifting operator the variable  $\alpha_u$  is very important when calculating moderate deformations. For small deformations a constant can be used for  $\alpha_u$ . But for larger deformations we need to be a bit more clever. A good strategy for choosing  $\alpha_u$  was proposed by Wick in [28], and further discussed in [21] and [13]. This alpha gets bigger when closer to the interface:

$$\alpha_u = \frac{1}{x^q} \quad (3.19)$$

where  $x^q$  is the distance from the interface. If  $q = 0$  the laplacian is recovered. When the distance becomes larger,  $\alpha_u$  gets smaller, and vice versa. Defining  $\alpha_u$  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  $\alpha_u > 0$  [13, 28]. 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:

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

The biharmonic extension is calculated using a mixed formulation where we introduce a new function  $\omega$  (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  $\omega$ ):

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

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

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

$$\mathbf{d}_f = \mathbf{d}_s \quad \text{on } \hat{\Sigma} \quad (3.23)$$

The second type of boundary condition imposes conditions on  $\mathbf{d}$  and  $\omega$ , 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.

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

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

$$(3.26)$$

Since the biharmonic extension is a fourth order PDE, it will have a higher computational cost [16] than the harmonic or linear elastic.

### 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 [27].

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  $\mathbf{u} \in \Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}^D$ ,  $p \in \Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}$  and  $\mathbf{d} \in \Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}^D$ . Let  $\phi, \psi$  and  $\gamma$  be the test functions used in the weak formulation, which are continuous across the entire domain.

$$(J\rho_f\partial_t\mathbf{u}, \phi) + (J(\nabla\mathbf{u})F^{-1}(\mathbf{u} - \partial_t\mathbf{d}), \phi)_{\hat{\mathcal{F}}} = 0 \quad (3.27)$$

$$(J\sigma_f F^{-T}, \nabla\phi)_{\hat{\mathcal{F}}} = 0 \quad (3.28)$$

$$(\rho_s\partial_t\mathbf{u}, \phi)_{\hat{\mathcal{S}}} + (P, \nabla\phi)_{\hat{\mathcal{S}}} = 0 \quad (3.29)$$

$$(\alpha_u\nabla\mathbf{u}, \nabla\psi)_{\hat{\mathcal{F}}} + (\nabla \cdot (JF^{-1}\mathbf{u}), \gamma)_{\hat{\mathcal{F}}} = 0 \quad (3.30)$$

$$\delta((\partial_t\mathbf{d}, \psi)_{\hat{\mathcal{S}}} - (\mathbf{u}, \psi)_{\hat{\mathcal{S}}}) = 0 \quad (3.31)$$

$$(J\sigma_{f,p} F^{-T}, \nabla\phi) = 0 \quad (3.32)$$

Introducing the  $\theta$ -scheme from [27], 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  $\theta$ . The Crank-Nicholson scheme is of second order, but suffers from instabilities in this monolithic scheme for certain time step values [27]. A remedy for these instabilities is to choose 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):

$$A_T(U) = (J\rho_f\partial_t\mathbf{u}, \phi) - (J(\nabla\mathbf{u})F^{-1}(\partial_t\mathbf{d}), \phi)_{\hat{\mathcal{F}}} \quad (3.33)$$

$$+ (\rho_s\partial_t\mathbf{u}, \phi)_{\hat{\mathcal{S}}} + (\partial_t\mathbf{d}, \psi)_{\hat{\mathcal{S}}} \quad (3.34)$$

$$A_I(U) = (\alpha_u\nabla\mathbf{u}, \nabla\psi)_{\hat{\mathcal{F}}} + (\nabla \cdot (JF^{-1}\mathbf{u}), \gamma)_{\hat{\mathcal{F}}} \quad (3.35)$$

$$A_E(U) = (J(\nabla\mathbf{u})F^{-1}\mathbf{u}, \phi)_{\hat{\mathcal{F}}} + (J\sigma_{f,u} F^{-T}, \nabla\phi)_{\hat{\mathcal{F}}} \quad (3.36)$$

$$+ (P, \nabla\phi)_{\hat{\mathcal{S}}} - (\mathbf{u}, \psi)_{\hat{\mathcal{S}}} \quad (3.37)$$

$$A_P(U) = (J\sigma_{f,p} F^{-T}, \nabla\phi) \quad (3.38)$$

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

part.

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

$$\sigma_{f,p} = -pI \quad (3.40)$$

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

$$A_T(U^{n,k}) \approx \frac{1}{k} (\rho_f J^{n,\theta}(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi)_{\hat{\mathcal{F}}} - \frac{1}{k} (\rho_f(\nabla u)(\mathbf{d}^n - \mathbf{d}^{n-1}), \phi)_{\hat{\mathcal{F}}} \quad (3.41)$$

$$+ \frac{1}{k} (\rho_s(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi)_{\hat{\mathcal{S}}} + \frac{1}{k} ((\mathbf{d}^n - \mathbf{d}^{n-1}), \psi)_{\hat{\mathcal{S}}} \quad (3.42)$$

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

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

Let the  $\theta$  scheme be defined as:

$$A_T(U^{n,k}) + \theta A_E(U^n) + A_P(U^n) + A_I(U^n) = \quad (3.44)$$

$$- (1 - \theta) A_E(U^{n-1}) + \theta \hat{f}^n + (1 - \theta) \hat{f}^{n-1} \quad (3.45)$$

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  $\theta$  will be investigated in chapter 4.



# Chapter 4

## Verification and Validation of the Fluid Structure Interaction Implementation.

The general approach when solving a real world problem with scientific computing, starts by defining the mathematics, implementing the equations numerically, and solve the equations on a computer. To gain insight into a real world problem, the produced solutions are used to extract data of interest.

A question then immediately arises, is the solution and the extracted data trustworthy? To answer this question we need to adress another question, are the equations implemented correct? if so, is the problem defined correct mathematically? Without answering these questions, being confident that your solutions are correct is difficult [15]. This 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 parts of the code will be verified. This will be followed Validation of the code where I implement a well known benchmark testing the fluid and structure parts individually and as a full FSI problem.

## 4.1 Verification

Verification, in the context of scientific computing, is the process of determining whether or not the implementation of numerical algorithms in computer code, is correct [14]. When mapping a mathematical model to a computational model there is always a risk of introducing errors to the computational model. Verification helps us identify, quantify, and reduce the errors. Verification does not address whether 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 [18]. The most rigorous of the order of convergence test is the *Method of Manufactured Solution* (MMS) [14]. When performing MMS tests, rather than looking for an exact solution, we manufacture one. The idea is to make 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 met for a solution to be sufficient. The manufactured solutions should be chosen to be non-trivial and analytic [14, 18]. The solutions should be manufactured so that no eventual derivatives vanish. 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 [14]:

- 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  $\hat{\mathbf{u}}$
- Then use this source term to calculate the solution  $\mathbf{u}$ ,  $L(\mathbf{u}) = f$

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

$$E = ||\mathbf{u} - \hat{\mathbf{u}}|| \quad (4.1)$$

When we decrease the node spacing ( $\Delta x$ ) or decrease time step size ( $\Delta t$ ), we expect the solution to convergence towards a given solution and hence the error to get smaller. If we assume uniform node spacing in all spatial directions:

$$E = C_1 \Delta x^k + C_2 \Delta t^l \quad (4.2)$$

where  $k = m + 1$  and  $m$  is the polynomial degree of the spatial elements. The error is hence dependent on the node spacing and the time step. If for instance  $\Delta t$  is small enough,  $\Delta x$  will dominate, and  $\Delta t$  will be negligible. Let  $E_{n+1}$  and  $E_n$  be the computed errors of a solution with fine and coarse node spacing respectively. If we divide the errors using (4.2):

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

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

After refining the mesh while keeping  $\Delta t$  fixed and sufficiently small,  $k$  can be found and 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, we can be satisfied that the implementation of the code is correct.

#### 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  $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$ .

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  $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$ .

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

The manufactured solutions are used to produce a source term  $f_s$  :

$$\rho_s \frac{\partial \hat{\mathbf{u}}_{\mathbf{e}}}{\partial t} - \nabla \cdot (P(\hat{\mathbf{d}}_{\mathbf{e}})) = f_s \quad (4.5)$$

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 functions  $u$  and  $d$  will be computed to match the source term  $f_s$ . 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.

$N$	$\Delta t$	$m$	$E_u$	$k_u$	$E_d$	$k_d$
4	$1 \times 10^{-7}$	1	0.0068828	-	$3.7855 \times 10^{-9}$	-
8	$1 \times 10^{-7}$	1	0.0017204	<b>2.0002577</b>	$9.4622 \times 10^{-10}$	<b>2.0002577</b>
16	$1 \times 10^{-7}$	1	0.0004300	<b>2.0000622</b>	$2.3654 \times 10^{-10}$	<b>2.0000622</b>
32	$1 \times 10^{-7}$	1	0.0001075	<b>2.0000154</b>	$5.9136 \times 10^{-11}$	<b>2.0000154</b>
64	$1 \times 10^{-7}$	1	0.0000268	<b>2.0000038</b>	$1.4783 \times 10^{-11}$	<b>2.0000038</b>

Table 4.1: Method of Manufactured Solution on the implementation of the Solid equation in space with  $m = 1$

In table 4.2 we check the temporal convergence. The number of spatial points has been fixed to 64, with varying  $\Delta t$  from 0.1 halving each step to 0.0065. The error  $E_u$  and  $E_d$  gets smaller for decreasing values of  $\Delta t$ . The scheme tested is temporal first order accurate, by setting a value  $\theta = 1$  expecting a temporal order of convergence of 1. In table 4.2 convergence  $k_u$  and  $k_d$  tends towards 1.

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

Table 4.2: Method of Manufactured Solution on the implementation of the Solid equation in time

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 could be because of the number of spatial points  $N = 64$  is not high enough. 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 MMS of Fluid equations with prescribed motion

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

The functions  $\hat{\mathbf{u}}$ ,  $\hat{\mathbf{d}}$  and  $\hat{p}$  are manufactured to uphold the restriction (??) and incompressible fluid (??), and are made with sine and cosine function to uphold the criteria of MMS. The fluid and domain velocity are set equal:  $\hat{\mathbf{u}} = \mathbf{w}$ .

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

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:

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

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

$$\rho_f J \frac{\partial \mathbf{u}}{\partial t} + (\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) + \nabla \cdot (J \hat{\sigma}_f F^{-T}) = J f_f$$

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  $\Delta t$  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	$\Delta t$	$E_u$	$k_u$	$E_p$	$k_p$
<b>64</b>	0.1	$5.1548 \times 10^{-5}$	-	0.008724	-
<b>64</b>	0.05	$2.5369 \times 10^{-5}$	<b>1.0228</b>	0.004290	<b>1.0240</b>
<b>64</b>	0.025	$1.2200 \times 10^{-5}$	<b>1.0561</b>	0.002058	<b>1.0596</b>
<b>64</b>	0.0125	$0.56344 \times 10^{-5}$	<b>1.1145</b>	0.0009556	<b>1.1068</b>

Table 4.3: Results for Method of Manufacture Solutions test for fluid equations

In table 4.4 we check the spatial convergence keeping the time step constant as  $\Delta t = 10^{-6}$ . 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.

N	$\Delta t$	m	$E_u$	$k_u$	$E_p$	$k_p$
<b>2</b>	$1 \times 10^{-6}$	<b>2</b>	$8.6955 \times 10^{-4}$	-	0.01943	-
<b>4</b>	$1 \times 10^{-6}$	<b>2</b>	$1.0844 \times 10^{-4}$	<b>3.0032</b>	0.00481	<b>2.0140</b>
<b>8</b>	$1 \times 10^{-6}$	<b>2</b>	$0.1354 \times 10^{-4}$	<b>3.0007</b>	0.00119	<b>2.0120</b>
<b>16</b>	$1 \times 10^{-6}$	<b>2</b>	$0.0169 \times 10^{-4}$	<b>3.0001</b>	0.00029	<b>2.0074</b>

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 MMS 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 [4], the stresses need to equal on the interface and the flow needs to be divergence free. Manufacturing such a solution is very difficult [3]. 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.

## 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 [15]. A computational model is made for a specific purpose, its only valid with respect to that purpose [12]. If the purpose of the computational model is complex and trying to answer multiple questions, then the validity needs to be determined for each question. The idea is to 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.

Three aspects have been identified in the process of validating a computational model [15]. These are: quantifying the accuracy of the model by comparing responses with experimental responses, interpolation of the model to conditions corresponding to the intended use and determining the accuracy of the model for the conditions under which its meant to be used.

The process of Validation is also, as personally experienced, a way to figure

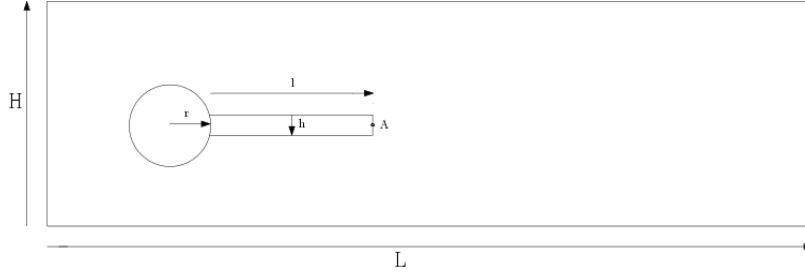
out at what size timestep and number of spatial points the model can handle. As we will see in this chapter all the benchmarks are run with different timesteps and number of cells to see how the model responds. The problem with using benchmarks with known data for comparison is that we do not test the model blindly. It is easier to mold the model to the data we already have. As Oberkampf and Trucano in [15] 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 [15]. This chapter comprises of the implementation of a well known benchmark test case, used to validate the computational FSI model.

#### **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” [9], based on an older well known CFD benchmark[19]. 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 circle with an elastic bar behind the circle. The circle is positioned at  $(0.2, 0.2)$  making it 0.05 off center from bottom to top. This shift in the domain is done to induce oscillations 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 made using Gmsh, with 11556 Cells, which is the mesh with the smallest node spacing. Table 4.2.1 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

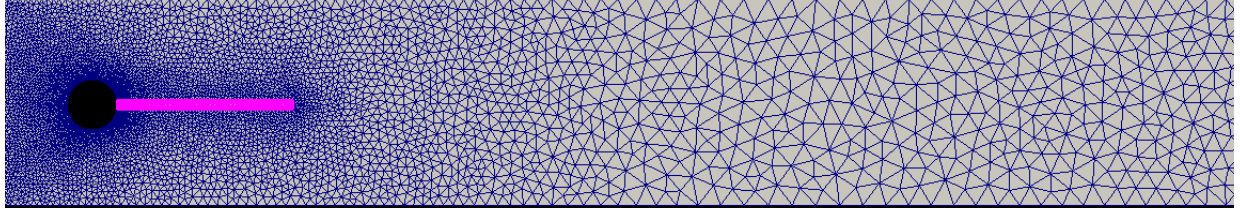


Figure 4.1: Picture of entire FSI computing domain with 11556 cells

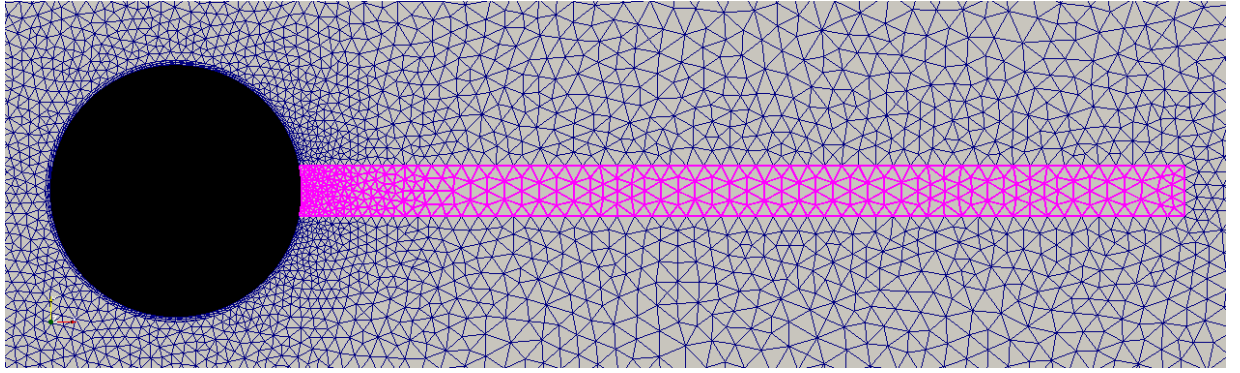


Figure 4.2: Picture of FSI computing domain with 11556 cells, zoomed in with the solid domain marked in pink. Around the circle we can see a small boundary layer with the width of 2 cells

## 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.

$$u(0, y) = 1.5u_0 \frac{y(H-y)}{(\frac{H}{2})^2}$$

$$u(0, y, t) = u(0, y) \frac{1 - \cos(\frac{\pi}{2}t)}{2} \text{ for } t < 2.0$$

$$u(0, y, t) = u(0, y) \text{ for } t \leq 2.0$$



## 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 set a point  $A = (0.2, 0.6)$  on the right side of the bar. Where this point is in the spatial direction gives a value 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:

$$mean = \frac{1}{2}(max + min) \quad (4.6)$$

$$amplitude = \frac{1}{2}(max - min) \quad (4.7)$$

$$(4.8)$$

In each test the values denoted as **ref** are the values taken from the original benchmark proposal paper [9].

## CFD test cases

The CFD tests can be simulated in two ways. 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 as a full FSI problem 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 test case has a Reynolds number 200 which will induce oscillations behind the circle and bar, giving fluctuations in the fluid velocity, hence giving unsteady solutions.

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
$Re = \frac{Ud}{\nu_f}$	20	100	200

Table 4.7: Summary of all the parameters in CFD tests

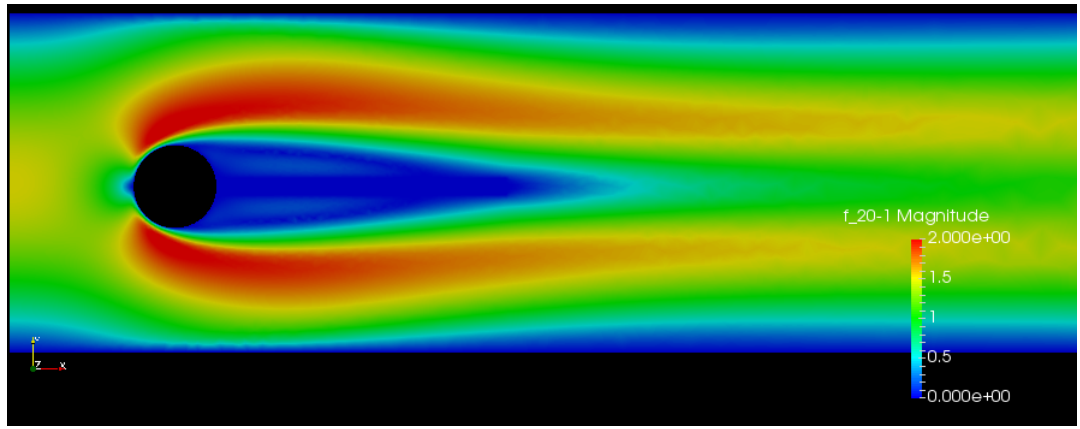


Figure 4.3: CFD2 steady state case fluid flow with 11556 cells

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. This non symmetry is what induces oscillations when the fluid inlet velocity is higher. 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
<b>ref</b>		<b>14.29</b>	<b>1.119</b>

Table 4.8: Results of CFD1 case run 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
<b>ref</b>		<b>136.7</b>	<b>10.53</b>

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 testcase with  $\Delta t = 0.01$ , the results show clear convergence toward the **ref** value for increased number of cells and dofs.

elements	dofs	Drag	Lift
2474	21749	$434.42 \pm 4.28$	$-15.63 \pm 407.59$
7307	63365	$435.54 \pm 5.06$	$-11.77 \pm 425.73$
11556	99810	$438.13 \pm 5.42$	$-10.01 \pm 435.40$
<b>ref</b>	<b>ref</b>	<b><math>439.45 \pm 5.61</math></b>	<b><math>-11.893 \pm 437.81</math></b>

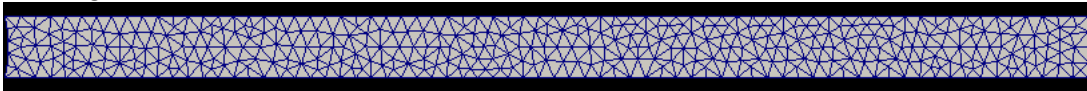
Table 4.10: Results of unsteady state case CFD 3 with  $\Delta t = 0.01$ , where the **ref** was computed with  $\Delta t = 0.005$

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 computing 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, bounce down and back up infinitely.

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



Parameters	CSM1	CSM2	CSM3
$\rho_f[10^3 \frac{kg}{m^3}]$	1	1	1
$\nu_f[10^{-3} \frac{m^2}{s}]$	1	1	1
$u_0$	0	0	0
$\rho_s[10^3 \frac{kg}{m^3}]$	1	1	1
$\nu_s$	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 4.13 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, and gives satisfactory results.

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
<b>ref</b>	<b>ref</b>	<b>-7.187</b>	<b>-66.10</b>

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
<b>ref</b>	<b>ref</b>	<b>-0.469</b>	<b>-16.97</b>

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	$-11.743 \pm 11.744$	$-57.952 \pm 58.940$
2900	6408	$-13.558 \pm 13.559$	$-61.968 \pm 63.440$
11600	24412	$-14.128 \pm 14.127$	$-63.216 \pm 64.744$
46400	95220	$-14.182 \pm 14.181$	$-63.305 \pm 64.843$
<b>ref</b>		<b><math>-14.305 \pm 14.305</math></b>	<b><math>-63.607 \pm 65.160</math></b>

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

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.

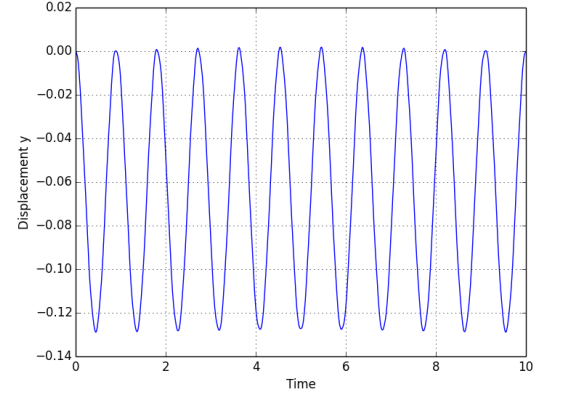
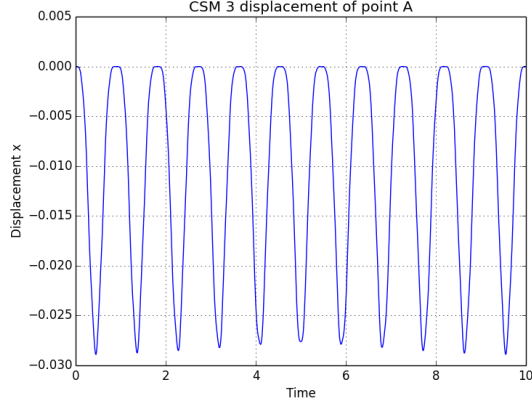


Figure 4.5: Displacement in x direction, Figure 4.6: Displacement in y direction, timeinterval (0,10)

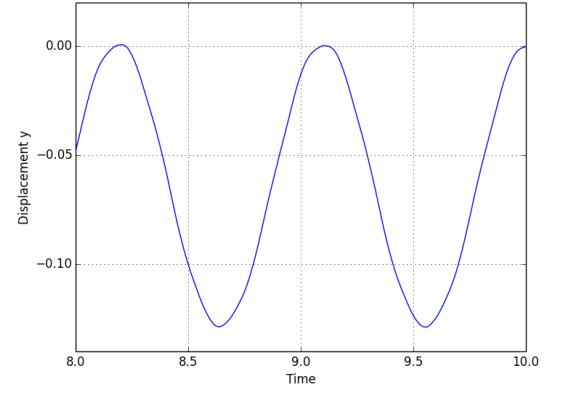
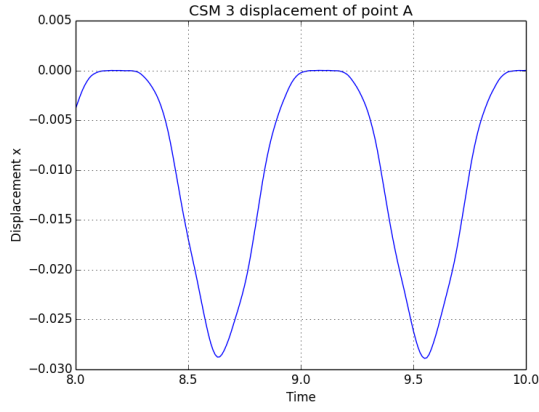


Figure 4.7: Displacement in x direction, Figure 4.8: Displacement in y direction, timeinterval (8,10)

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. FSI-2 gives the largest deformation is therefore considered the most difficult of the three [17], giving deformations of 2.5 times greater than the flag height. The FSI-3 test has the highest inflow speed giving medium deformations but more rapid oscillations.

The parameters for FSI1-3 are shown in table

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
$Re = \frac{Ud}{\nu_f}$	20	100	200
$\rho_s [10^3 \frac{kg}{m^3}]$	1	10	1
$\nu_s$	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
<b>ref</b>	<b>ref</b>	<b>0.0227</b>	<b>0.8209</b>	<b>14.295</b>	<b>0.7638</b>

Table 4.16: Results of FSI 1 test case

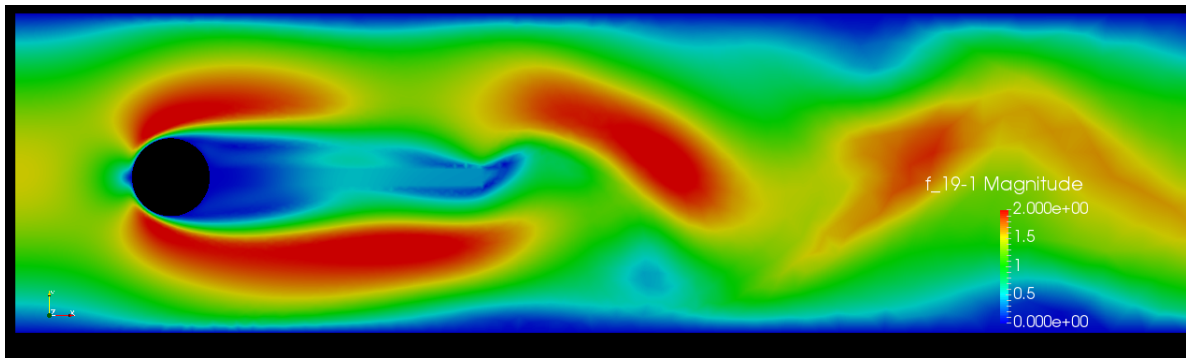


Figure 4.10: FSI2 test case Fluid velocity at  $t = 9.70\text{sec}$  on the reference mesh

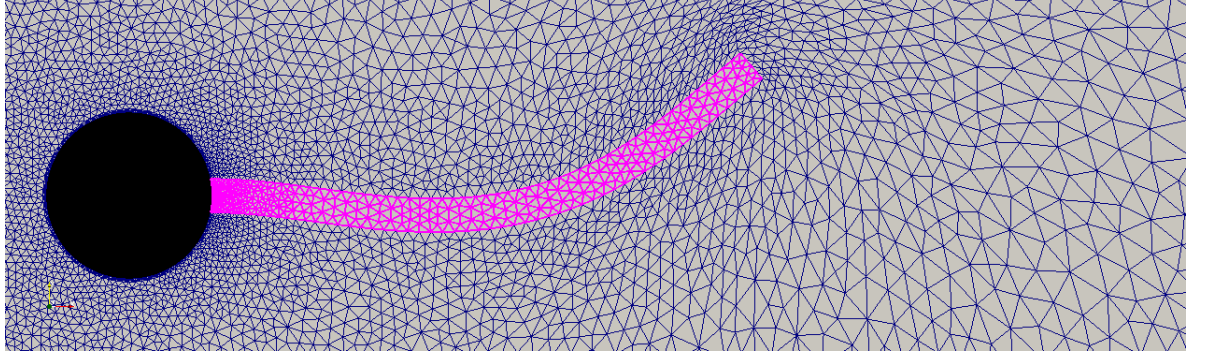


Figure 4.11: FSI2 test case deformation at  $t = 9.70\text{sec}$ . The bar is marked with pink colour and deformed using warp by vector in paraview.

The tables 4.17 and 4.18 shows results for the FSI2 test case. The displacements in both results for both of the  $\Delta t = 0.01$  and  $\Delta t = 0.001$  show convergence towards the **ref** values. The lift converges more slowly towards the ref value, while the drag values are off by about a value of 45 in both cases.

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

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	$-15.10 \pm 13.32$	$1.16 \pm 82.46$	$159.53 \pm 17.44$	$0.68 \pm 259.10$
7307	63365	$-14.85 \pm 13.14$	$1.21 \pm 81.72$	$160.72 \pm 17.84$	$0.93 \pm 255.14$
11556	99810	$-14.83 \pm 13.11$	$1.24 \pm 81.6$	$161.50 \pm 18.17$	$0.62 \pm 254.40$
<b>ref</b>	<b>ref</b>	<b><math>-14.58 \pm 12.44</math></b>	<b><math>1.23 \pm 80.6</math></b>	<b><math>208.83 \pm 73.75</math></b>	<b><math>0.88 \pm 234.2</math></b>

Table 4.18: FSI-2 with  $\Delta t = 0.001$ , using harmonic lifting operator



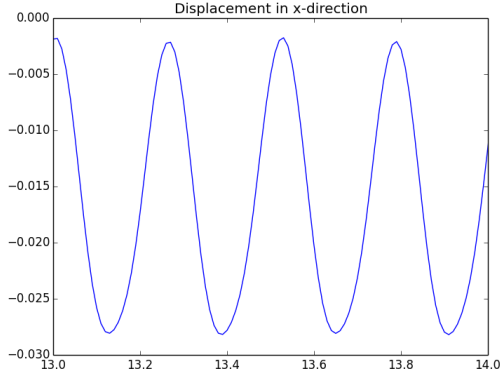


Figure 4.12: Displacement x

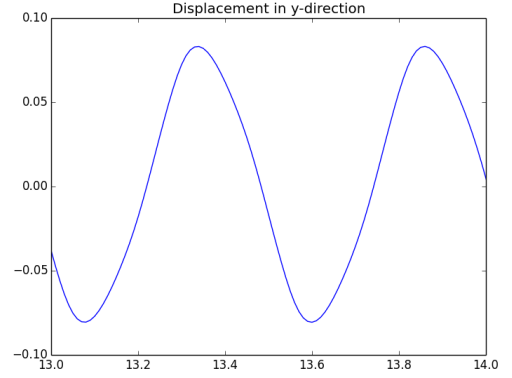


Figure 4.13: Displacement y

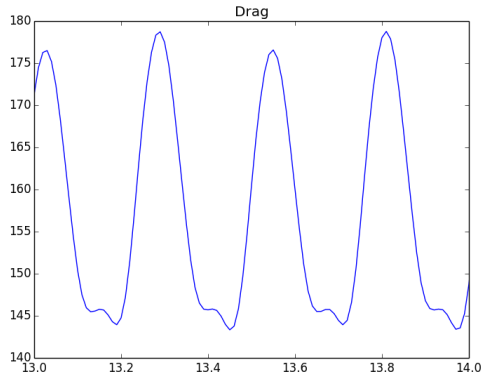


Figure 4.14: Drag

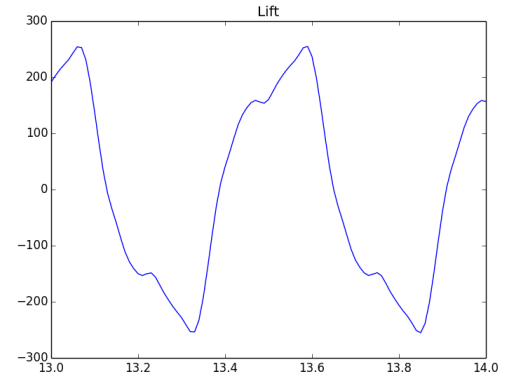


Figure 4.15: Lift

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 but still within 50 % of the referential values.

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2474	21249	$-1.79 \pm 1.80$	$3.29 \pm 2.64$	$439.36 \pm 12.04$	$1.96 \pm 142.31$
7307	63365	$-2.48 \pm 2.48$	$1.64 \pm 3.28$	$449.77 \pm 18.02$	$3.41 \pm 153.47$
11556	99810	$-2.47 \pm 2.45$	$1.27 \pm 3.28$	$456.60 \pm 18.73$	$1.55 \pm 153.46$
ref	ref	<b><math>-2.69 \pm 2.56</math></b>	<b><math>1.48 \pm 34.38</math></b>	<b><math>457.3 \pm 22.66</math></b>	<b><math>2.22 \pm 149.78</math></b>

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	$-2.188 \pm 2.11$	$3.56 \pm 2.90$	$435.19 \pm 9.77$	$-1.57 \pm 151.43$
7307	63365	$-1.42 \pm 4.70$	$0.77 \pm 28.50$	$454.38 \pm 19.75$	$1.79 \pm 155.08$
11556	99810	$-2.23 \pm 6.164$	$1.72 \pm 4.48$	$459.12 \pm 22.97$	$3.12 \pm 171.22$
ref	ref	<b><math>-2.69 \pm 2.56</math></b>	<b><math>1.48 \pm 34.38</math></b>	<b><math>457.3 \pm 22.66</math></b>	<b><math>2.22 \pm 149.78</math></b>

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

## Comment on FSI tests

A very important thing to notice about this benchmark [9] 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, [22] that compared results of different discretizations and solution approaches. This paper [22] 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 values which 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 thesis. 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 with a wrong implementation I got good FSI1 results. However it is a good test for early checks, because if FSI1 is wrong the rest will definitely not work.

For the FSI2 test case we only have results from the initial Hron and Turek paper [9]. 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 initial 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 is upheld. There is also smaller node spacing around the circle and around the bar, leading to larger node spacing as we move down stream. In the unsteady CFD and FSI cases there is vortex shedding happening downstream. With large node spacing in this area the vortices 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  $\Delta t$  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 when in FSI problems [21]. 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 computational cost varies the different lifting operators, and it is therefore crucial to chose the right lifting operator for a specific problem.

The second section investigates briefly the impact of choosing different value for  $\theta$  in the  $\theta$ -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 computing domain and parameters. 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 much like the previous CSM test. The only difference is that we now use the full domain from the 4.2.1. The test is run 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. Which 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 visualize how the different mesh motion techniques work. 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 looked at how different lifting operators react differently in the FSI2 and FSI3 test cases from the previous section. Here I investigated plots of the lift, drag and displacements too see how the different lifting operators respond to different inflow velocities and solid parameters. Bc1 and bc2 denotes the boundary conditions 1 and 2 used when employing the biharmonic lifting operator.

## Different lifting operators with testcase CSM1

Figure 5.1 which shows the minimum of the Jacobian of the entire domain. The harmonic operator with a constant  $\alpha_u$  parameter, gives overlapping cells. While the harmonic lifting operator with variable  $\alpha_u$  and both the biharmonic techniques, uphold the cells quality.

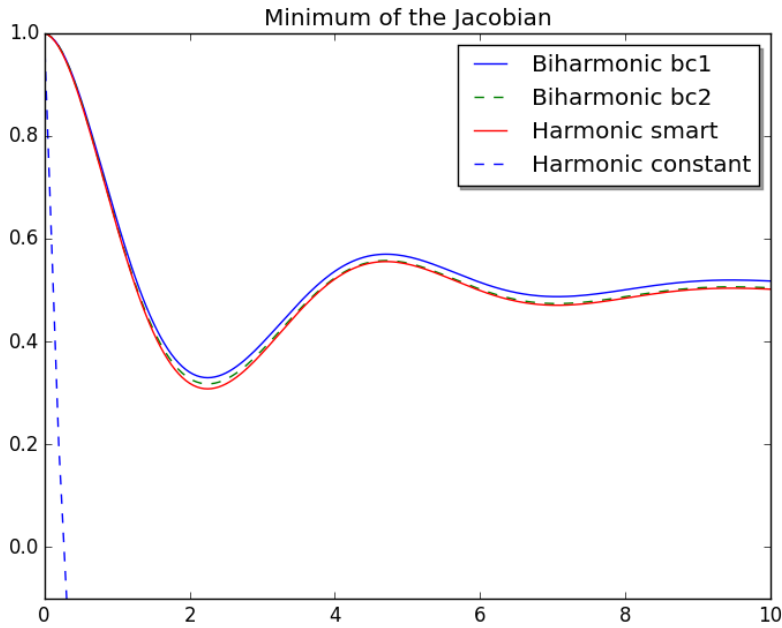


Figure 5.1: plot of the minimum of  $J$  in entire domain, using CSM1 test.  $\Delta t = 0.05$

Figure 5.6 shows the meshes of the different lifting operators, for the steady state solutions from the CSM1 test case. The harmonic with a spatial dependent variable  $\alpha_u$  and the two biharmonic lifting operators uphold good integrity of the cells. While the harmonic lifting operator with a constant  $\alpha_u$  shows degeneration of cells at the tip of the bar. However, the computed numerical code was able to compute with the harmonic constant  $\alpha_u$  operator, but as we can see in table 5.1 the displacement values are incorrect.

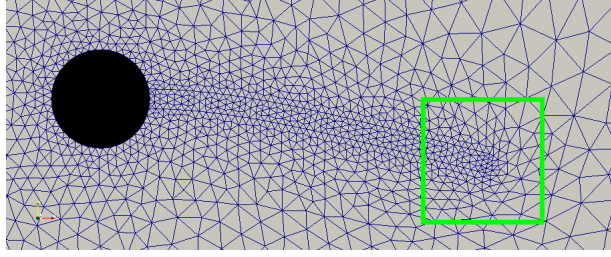


Figure 5.2: Harmonic lifting operator with spatial dependent  $\alpha_u$

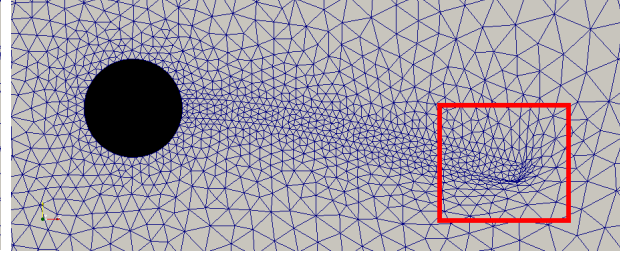


Figure 5.3: Harmonic lifting operator with constant  $\alpha_u$

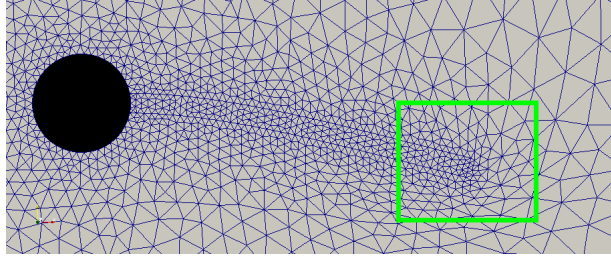


Figure 5.4: Biharmonic lifting operator with boundary condition 1

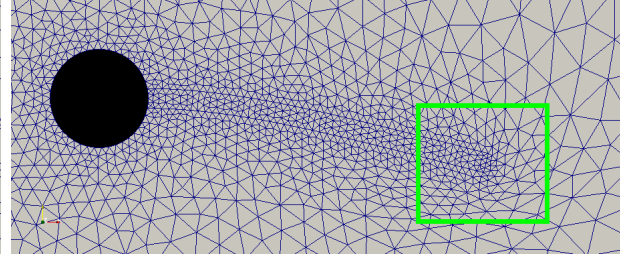


Figure 5.5: Biharmonic lifting operator with boundary conditions 2

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	<b>-66.10</b>	<b>-7.187</b>

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.



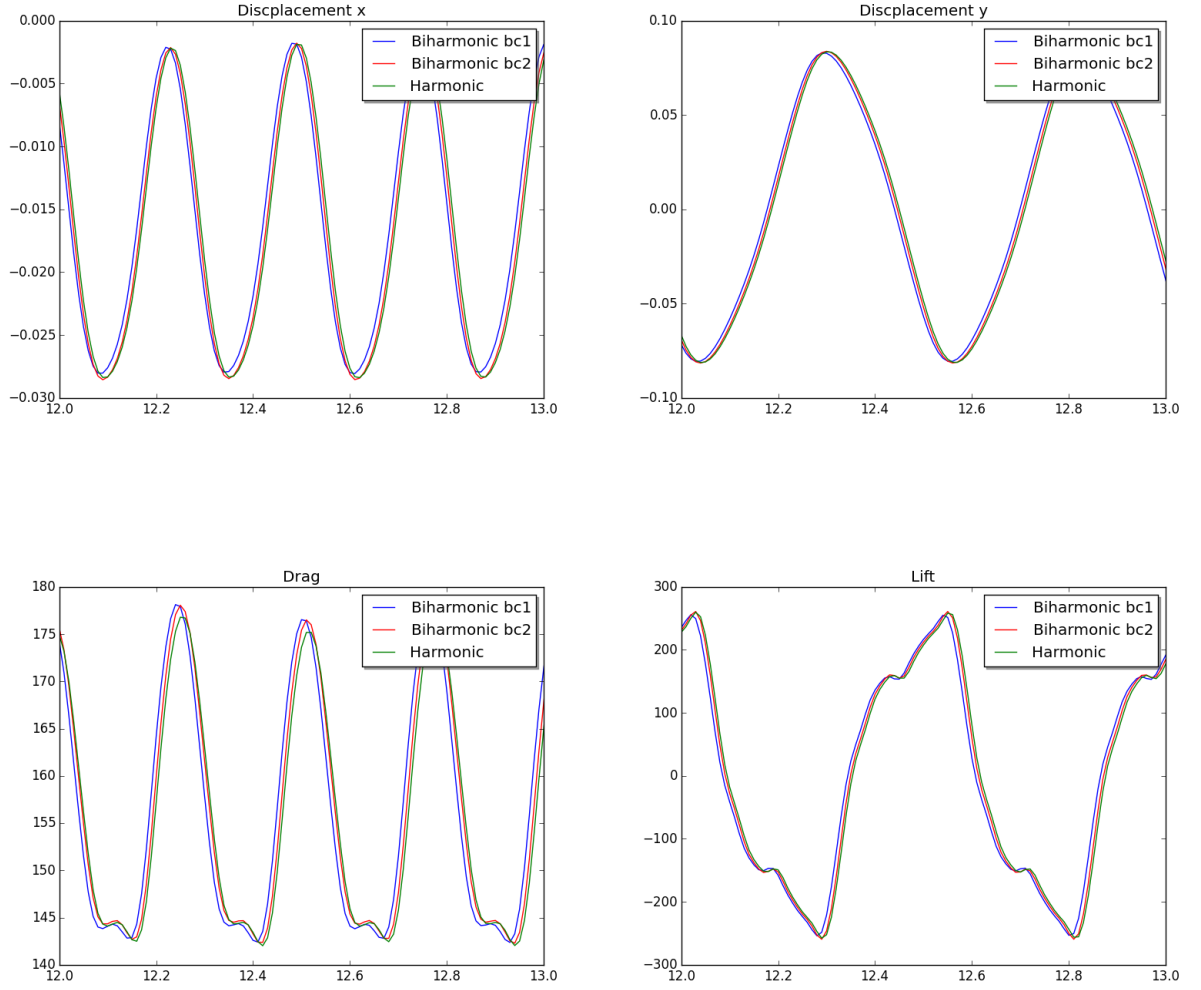


Figure 5.7: FSI2 with different lifting operators: harmonic with varying  $\alpha_u$ , biharmonic bc1 and bc2.  $\Delta t = 0.01$

Figure 5.1, 5.1, 5.1 and 5.1 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  $\alpha_u$  shows an increasing in the Drag value of about 10.

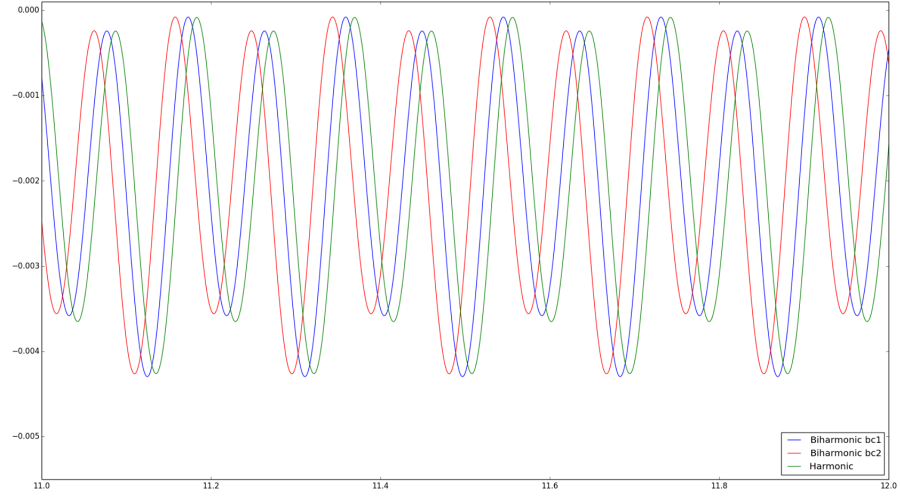


Figure 5.8: Displacement in x direction for FSI3 with different lifting operators: Harmonic varying  $\alpha_u$ , Biharmonic bc1 and bc2.  $\Delta t = 0.01$

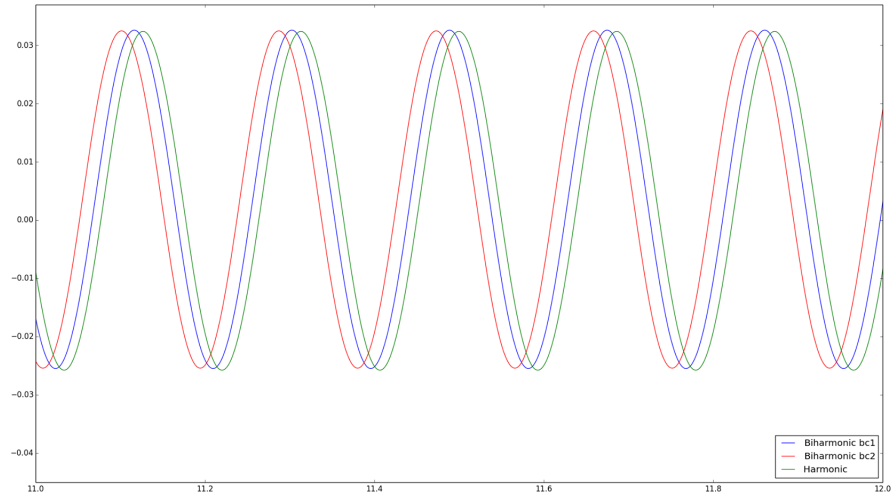


Figure 5.9: Displacement in x direction for FSI3 with different lifting operators: Harmonic varying  $\alpha_u$ , Biharmonic bc1 and bc2.  $\Delta t = 0.01$

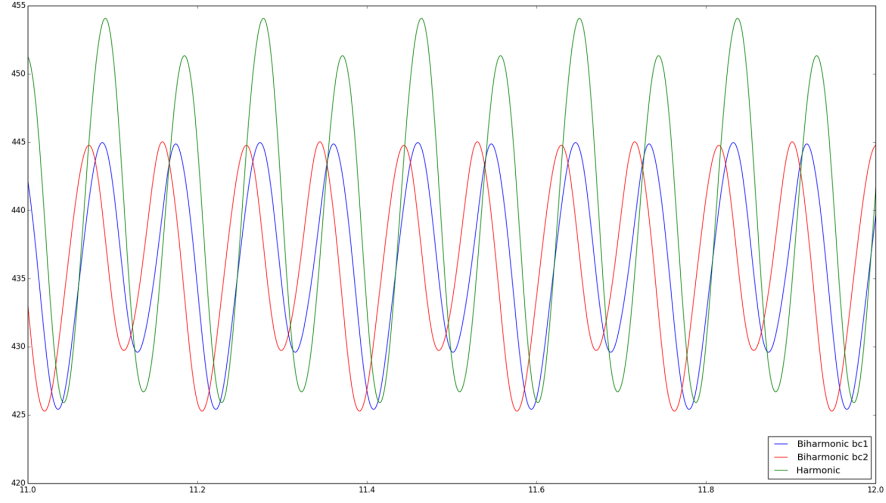


Figure 5.10: Drag results for FSI3 with different lifting operators: Harmonic varying  $\alpha_u$ , Biharmonic bc1 and bc2.  $\Delta t = 0.01$

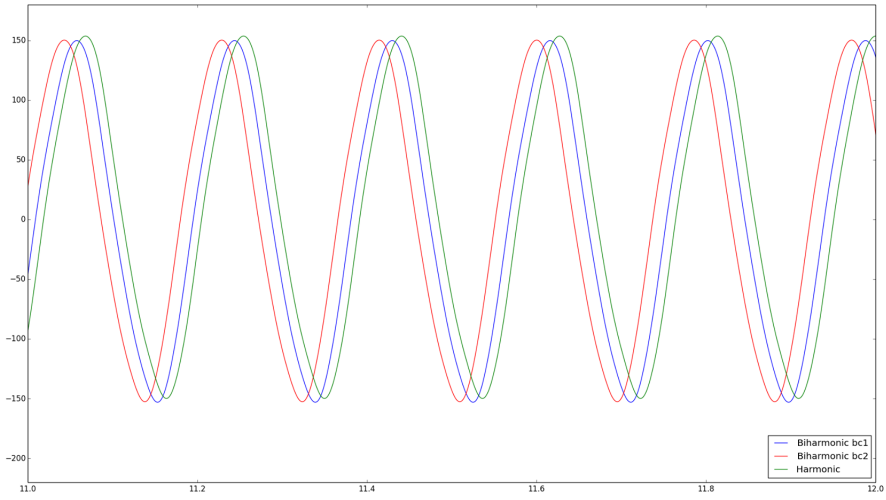


Figure 5.11: Lift results for FSI3 with different lifting operators: Harmonic varying  $\alpha_u$ , Biharmonic bc1 and bc2.  $\Delta t = 0.01$

## 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  $\alpha_u$ , 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 speed 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.

It should also be noted that the biharmonic lifting operators are able to compute in parallel with multiple computer cores, while the harmonic with variable  $\alpha_u$  is not able to run in parallel. This concludes that even though the harmonic lifting operator 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  $\theta$  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  $\theta$  and  $\Delta t$ . Only the effects of Drag as been studied as the three other quantities shows similar behavior. The impact of different  $\theta$  values on energy stability in the solid mechanical benchmark CSM3 is also investigated.

5.16 show the plots of Drag with  $\Delta t = 0.01$ , showing the instability when choosing  $\theta = 0.5$ . The Crank-Nicholson scheme is stable until about 13 seconds where we see that it is numerically unstable and the solver diverges. While the shifted Crank-Nicholson,  $\theta = 0.5 + \Delta t$ , is stable throughout the computing time.

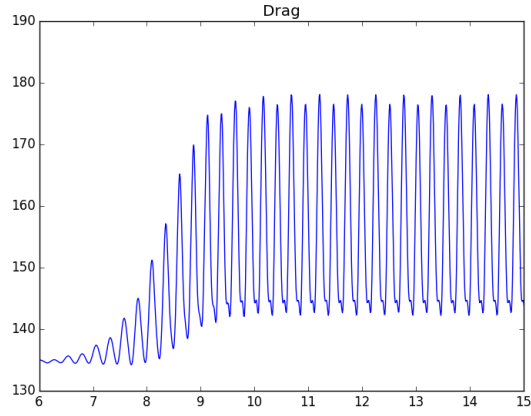


Figure 5.12:  $\theta = 0.50 + \Delta t$

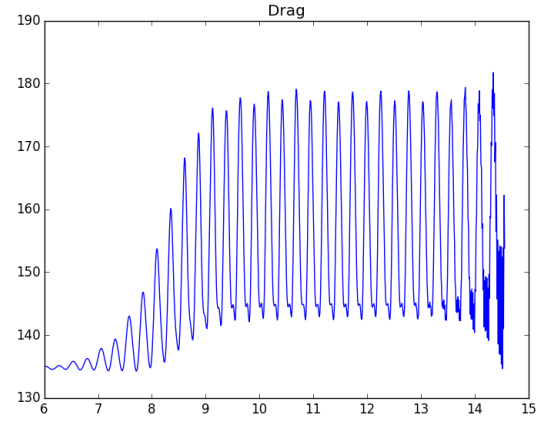


Figure 5.13:  $\theta = 0.50$

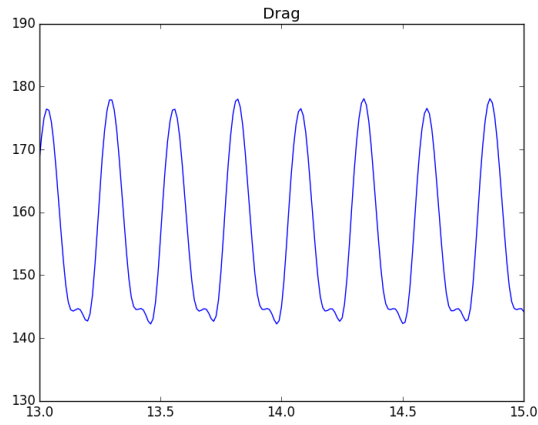


Figure 5.14:  $\theta = 0.50 + \Delta t$

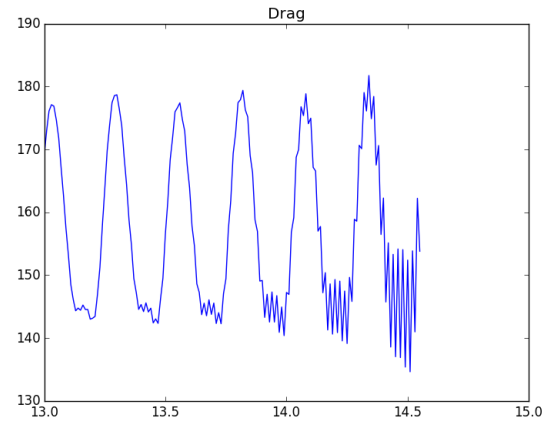


Figure 5.15:  $\theta = 0.50$

Figure 5.16: Drag for FSI2 with  $\Delta t = 0.01$  with different values for  $\theta$

Figures 5.17 show drag for FSI3 simulation with  $\Delta t = 0.001$  and  $\theta = 0.5$ , showing long term stability for the normal Crank-Nicholson scheme.

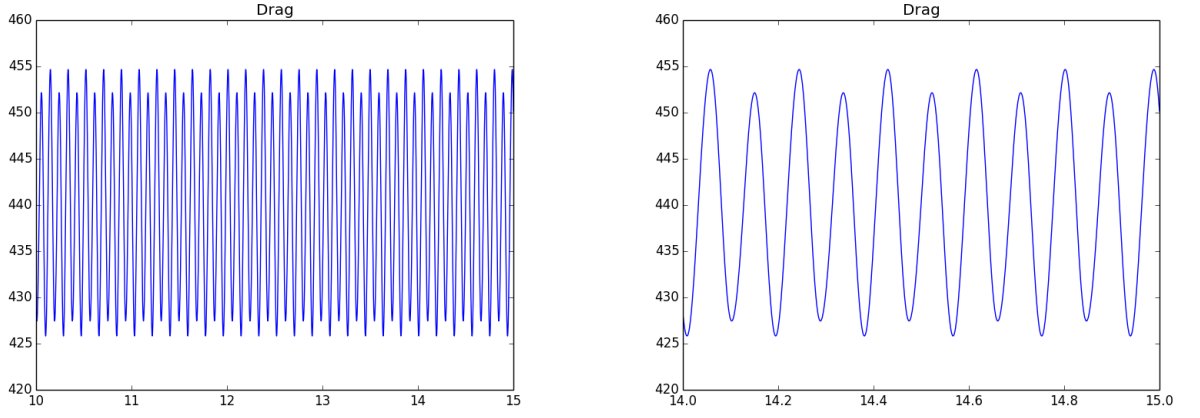


Figure 5.17: FSI3 drag plot, with  $\Delta t = 0.001$  and  $\theta = 0.5$  showing long term numerical stability

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 bounce 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.

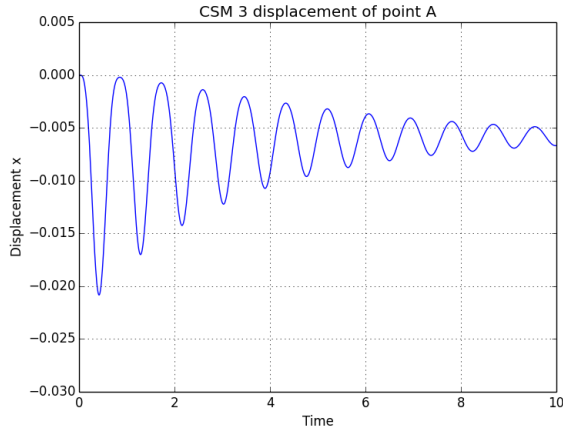


Figure 5.18:  $\theta = 1$

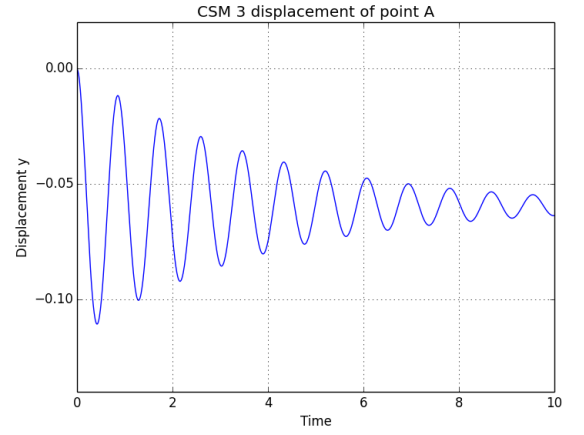


Figure 5.19:  $\theta = 1$

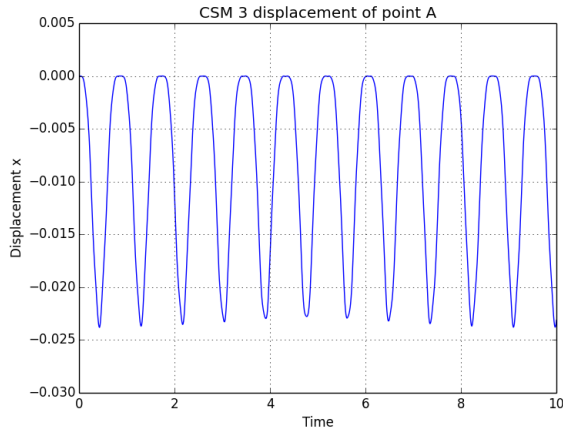


Figure 5.20:  $\theta = 0.5$

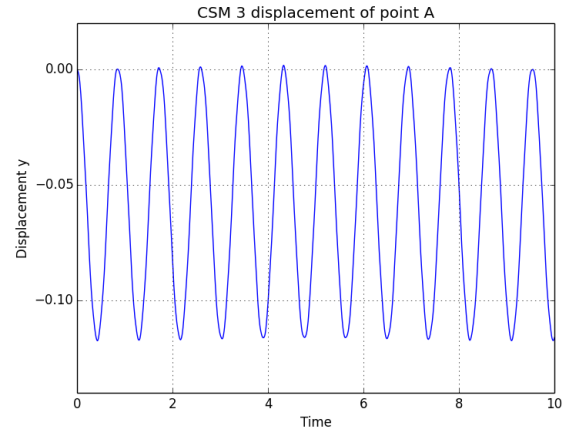


Figure 5.21:  $\theta = 0.5$

Figure 5.22: CSM3 displacements with  $\Delta t = 0.01$  with different values for  $\theta$

### 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. It



has also been reported by Wick 2011 [27] 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.



# Chapter 6

## CPU time reduction techniques

Computing FSI accurately is difficult not only because of the lack of stable partitioned schemes. It is also difficult because of the increase in runtime when modeling both fluid and structure, especially with monolithic schemes. When accuracy is needed in CFSI the number of cells can quickly become large. The matrices can become so large that the amount of computing time is not feasible.

In the Newton solver we have to assemble the Jacobian of the matrix, assemble the residual and solve for each iteration, until convergence is met. Working on this thesis the issue of runtime was certainly a problem, the wall clock time being days and weeks for a simulation to finish.

This chapter describes the runtime reduction techniques made to the Newton solver. The point of this chapter is not to rigorously experiment with time improvement techniques, but merely to show what helped make simulation times bearable. Hopefully, helping others when employing a Newton solver. In fact, the ideas covered here are not new and has been implemented in Gabriel Balaban's master thesis in 2012 [20].

### 6.1 Newton runtime profile

In table 6.1 shows the runtime of the FSI1 problem with no optimization. Listing the amount of time spent on Jacobian Assembly, residual assembly, and solve call. In table 6.1 it shows that the majority of time spent is in

the assembly of the Jacobian. Therefore the effort will be spent on reducing the time spent on assembling the Jacobian. The next sections introduces two ways of improving time spent on assembling greatly. The first is reusing of the Jacobian and the second is employing a function in FENiCS called quadrature reduce. I will compare the timesaving techniques to table 6.1.

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

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

## 6.2 Reusing the Jacobian

As the time step is low in all the benchmarks, the effects of assembling the Jacobian for everytime step is assumed low. The trick therefore is to reuse the Jacobian. This is done by specifying reassembly for a certain number of iterations. When employed it was noticed that we needed a larger number of iterations to reach convergence, but the overall time of the Newton solver was much faster. In table 6.2 the same FSI1 test is done with  $\Delta t = 0.5$ . Even with a timestep that is fairly large, we get great improvements in runtime (-74%). The Jacobian assembles once meaning that at this timestep the Jacobian was calculated once and used again 9 times. What happens when we reuse the Jacobian, we have to iterate more times (9 times in this case) and as we can see the runtime in assembling the residual has increased by 50 %. This is a much less costly process, so even when iterating more times we get a reduce in runtime.

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

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

## 6.3 Quadrature reduce

Assembling of the Jacobian matrix with non-linear terms, induces a high number of quadrature points, leading to a dense matrix. When the accuracy of the Jacobian can be reduced, we can reduce the number of quadrature degree. This improves the runtime but reduces the accuracy leading to more iterations per time step. Reducing the quadrature degree can lead to numerical divergence of the system in some cases, but is of great help in many other cases.

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 %.

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

Table 6.3: Parts of the Newtonsolver timed with quadrature reduce run with FSI1 , $\Delta t = 0.5$

## 6.4 Summary of runtime improvement techniques

Finally using both the reuse of Jacobian and the reduction of the quadrature. As we can see in table 6.4 the total runtime went down 89%. In my work on this thesis these two techniques were used as often as possible. Both techniques worked with great success in the FSI2 and FSI3 case.

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

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

# Chapter 7

## Conclusions

The work on this thesis started with a plan to build a FSI solver meant to be applied to a hemodynamical problem. The plan was to build a monolithic solver to obtain reference data, and a partitioned scheme, specifically the scheme introduced by Fernandez 2013 [6]. It was discovered how difficult it is to build an accurate monolithic FSI solver that can handle large deformations and high fluid velocity. This complexity can be seen in the variability of the results provided in the appendix B, from highly renowned scientists. An added difficulty is that the field of FSI is still very young, meaning good information is not yet accessible. Work on the partitioned scheme was put aside and full focus was set on building, verifying, and validating a monolithic solver in a rigorous manner. Lessons were learned about the importance of lifting operators, energy stable numerical schemes and the need to compute on reference domains.

For the FSI fields future the main problem of long run time should be tackled by looking at partitioned schemes. When a FSI scheme is partitioned we can use legacy code for the structure and fluid parts, increasing run times significantly. The difficulty is still to transfer the energies 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 [7, 24]. There are however, schemes which offer added-mass free stability with explicit coupling, where the interface is treated through a Robin-Neumann coupling. The first scheme was made for coupling of a thin walled structure by Fernandez et al., 2013 [6]. Later a scheme was made with an extension coupling with a thick walled structure by Fernandez et.al,2015 [7]. The scheme for coupling thick walled structures is rather complex. Hopefully my work on a

building a monolithic solver can help others advance the field of FSI.



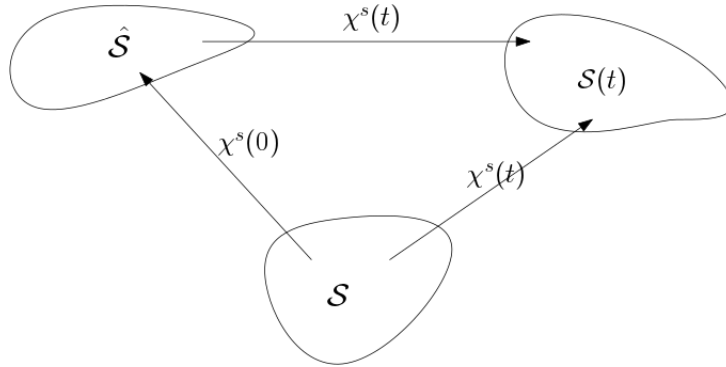
# Appendices



# 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:

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

Where  $\mathbf{X}$  denotes a material point in the reference domain and  $\chi^s$  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  $\chi^s$  can then be specified from a current position plus the displacement from that position:

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

which can be written in terms of the displacement field:

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

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

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

### 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:

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

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:

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

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

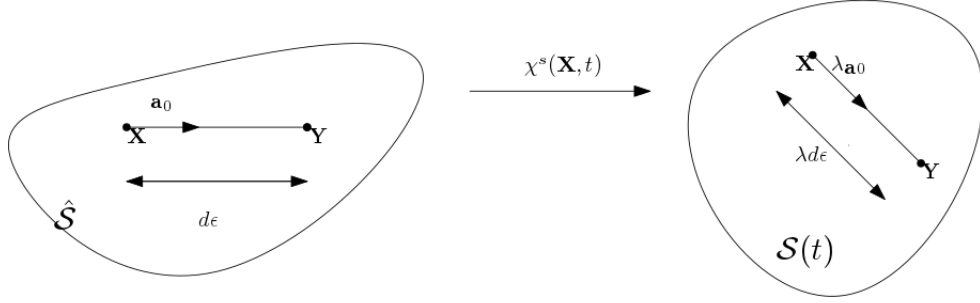


Figure A.1: Deformation of a line element with length  $d\epsilon$  into a line element with length  $\lambda d\epsilon$

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 [16].

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  $d\mathbf{X}$  :

$$\mathbf{Y} = \mathbf{Y} + \mathbf{X} - \mathbf{X} = \mathbf{X} + |\mathbf{Y} - \mathbf{X}| \frac{\mathbf{Y} - \mathbf{X}}{|\mathbf{Y} - \mathbf{X}|} = \mathbf{X} + d\mathbf{X} \quad (\text{A.7})$$

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

$$d\mathbf{X} = d\epsilon \mathbf{a}_0 \quad (\text{A.8})$$

$$d\epsilon = |\mathbf{Y} - \mathbf{X}| \quad (\text{A.9})$$

$$\mathbf{a}_0 = \frac{\mathbf{Y} - \mathbf{X}}{|\mathbf{Y} - \mathbf{X}|} \quad (\text{A.10})$$

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:

$$\mathbf{y} = \chi^s(\mathbf{Y}, t) = \chi^s(\mathbf{X} + d\epsilon \mathbf{a}_0, t) \quad (\text{A.11})$$

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

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

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

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

$$= F(\mathbf{Y} - \mathbf{X}) + \mathcal{O}(\mathbf{Y} - \mathbf{X}) \quad (\text{A.14})$$

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

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

Looking at the square of  $\lambda$ :

$$\lambda^2 = \lambda_{\mathbf{a}_0} \lambda_{\mathbf{a}_0} = F(\mathbf{X}, t) \mathbf{a}_0 F(\mathbf{X}, t) \mathbf{a}_0 \quad (\text{A.16})$$

$$= \mathbf{a}_0 F^T F \mathbf{a}_0 = \mathbf{a}_0 C \mathbf{a}_0 \quad (\text{A.17})$$

Introducing the right Cauchy-Green tensor:

$$C = F^T F \quad (\text{A.18})$$

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:

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

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  $\sigma_s$ , 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:

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

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

$$P = J \sigma F^{-T} \quad (\text{A.21})$$

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:

$$S = J F^{-1} \sigma F^{-T} = F^{-1} P = S^T \quad (\text{A.22})$$

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

$$P = F S \quad (\text{A.23})$$







# Appendix B

## Results

**Table 3** Results for unsteady benchmark FSI3.

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

Figure B.1: Results from different contributions in from the paper Turek et.al 2010 [22]

# Appendix C

## Implementation of Fluid Structure Interaction in FEniCS

The discretization described in chapter 3.5, is implemented using FEniCS [11]. 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 form

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 and 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  $\theta$  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.

```

1 d_["n"]  """deformation in the current timestep """
2 u_["n-1"] """velocity in the last timestep """
3 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:  $\psi$ ,  $\phi$  and  $\gamma$ .

```

1 J_theta = theta*J_(d_["n"]) + (1 - theta)*J_(d_["n-1"])
2
3 F_fluid_linear = rho_f/k*inner(J_theta*(v_["n"] - v_["n-1"]), psi)*dx_f
4
5 F_fluid_nonlinear = Constant(theta)*rho_f*\
6 inner(J_(d_["n"])*grad(v_["n"])*inv(F_(d_["n"]))*v_["n"], psi)*dx_f
7
8 F_fluid_nonlinear += inner(J_(d_["n"])*sigma_f_p(p_["n"], d_["n"])*\
9 inv(F_(d_["n"])).T, grad(psi))*dx_f
10
11 F_fluid_nonlinear += Constant(theta)*inner(J_(d_["n"])\
12 *sigma_f_u(v_["n"], d_["n"], mu_f)*inv(F_(d_["n"])).T, grad(psi))*dx_f
13
14 F_fluid_nonlinear += Constant(1 - theta)*inner(J_(d_["n-1"])*\
15 sigma_f_u(v_["n-1"], d_["n-1"], mu_f)*inv(F_(d_["n-1"])).T, grad(psi))*
    dx_f
16
17 F_fluid_nonlinear += \
18 inner(div(J_(d_["n"])*inv(F_(d_["n"]))*v_["n"]), gamma)*dx_f
19
20 F_fluid_nonlinear += Constant(1 - theta)*rho_f*\
21 inner(J_(d_["n-1"])*grad(v_["n-1"])*inv(F_(d_["n-1"]))*v_["n-1"], psi)*
    dx_f
22
23 F_fluid_nonlinear -= rho_f*inner(J_(d_["n"])*\
24 grad(v_["n"])*inv(F_(d_["n"]))*((d_["n"]-d_["n-1"])/k), psi)*dx_f
25
26 delta = 1E10
27 F_solid_linear = rho_s/k*inner(v_["n"] - v_["n-1"], psi)*dx_s +\
28 delta*(1/k)*inner(d_["n"] - d_["n-1"], phi)*dx_s -\
29 delta*inner(Constant(theta)*v_["n"] + Constant(1-theta)*v_["n-1"], phi)*
    dx_s
30
31 F_solid_nonlinear = inner(Piola1(Constant(theta)*d_["n"] +\
32 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 [26]. Following is a print out of the full Newtonsolver, that is called for each time iteration.

```

1 def newtonsolver(F, J_nonlinear, A_pre, A, b, bcs, \
2                 dvp_, up_sol, dvp_res, rtol, atol, max_it, T, t, **
3                 monolithic):
4     Iter      = 0  """ Setting initial values """
5     residual   = 1
6     rel_res    = residual
7     lambda = 1  """ """
8     while rel_res > rtol and residual > atol and Iter < max_it:
9         if Iter % 10 == 0: """Assembles the Jacobian for each tenth
10         round, in this instance. """
11             A = assemble(J_nonlinear, tensor=A, form_compiler_parameters
12             = {"quadrature_degree": 4}) """ Assembles the Jacobian with
13             reduction of the quadrature """
14             A.axpy(1.0, A_pre, True)
15             A.ident_zeros() """ Sets values of zero to 1 to ensure zero
16             pressure in the solid domain """
17
18             b = assemble(-F, tensor=b) """ assembling the residual """
19
20             [bc.apply(A, b, dvp_["n"].vector()) for bc in bcs] """ Applies
21             boundary conditions to the mixed function dvp """
22             up_sol.solve(A, dvp_res.vector(), b) """ Solves the matrix
23             equation A * dvp = b """
24             dvp_["n"].vector().axpy(lambda, dvp_res.vector()) """ A fast
25             """
26             [bc.apply(dvp_["n"].vector()) for bc in bcs]
27             rel_res = norm(dvp_res, 'l2')
28             residual = b.norm('l2')
29             if isnan(rel_res) or isnan(residual):
30                 print "type rel_res: ",type(rel_res)
31                 t = T*T
32
33             if MPI.rank(mpi_comm_world()) == 0: """ Prints only out the
34             numeber 0 process when running code in paralell """
35                 print "Newton iteration %d: r (atol) = %.3e (tol = %.3e), r
36                 (rel) = %.3e (tol = %.3e) " \
37                 % (Iter, residual, atol, rel_res, rtol)
38                 Iter += 1
39
40     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] S. Étienne, A. Garon, and D. Pelletier. Some manufactured solutions for verification of fluid-structure interaction codes. *Computers and Structures*, 106-107:56–67, 2012.
- [4] 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.
- [5] Charles L. Fefferman. Existence and smoothness of the Navier-Stokes equation. *The millennium prize problems*, (1):1–5, 2000.
- [6] 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.
- [7] 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.
- [8] G Holzapfel. Nonlinear solid mechanics: A continuum approach for engineering, 2000.

- [9] 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.
- [10] 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.
- [11] Anders Logg, Harish Narayanan, Marie Rognes, Johannes Ring, Kristian B. Ølgaard, and Garth N. Wells. FEniCS Project, 2011.
- [12] Cm Macal. Proceedings of the 2005 Winter Simulation Conference ME Kuhl, NM Steiger, FB Armstrong, and JA Joines, eds. *Simulation*, pages 1643–1649, 2005.
- [13] Selim MM and Koomullil RP. Mesh Deformation Approaches – A Survey. *Journal of Physical Mathematics*, 7(2), 2016.
- [14] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge, 2010.
- [15] William L Oberkampf and Timothy G Trucano. Verification and validation benchmarks. *Nuclear engineering and Design*, 238(3):716–743, 2008.
- [16] T Richter. Numerical Methods for Fluid-Structure Interaction Problems. *Course of lectures*, 2010(August 2010), 2010.
- [17] T Richter and T Wick. On time discretizations of Fluid-structure interactions. *Multiple Shooting and Time Domain Decomposition MEthods*, pages 377–400, 2013.
- [18] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, 124(1):4, 2002.
- [19] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark Computations of Laminar Flow Around a Cylinder. pages 547–566, 1996.
- [20] Natural Sciences. A Newton ’ s Method Finite Element Algorithm for Fluid-Structure Interaction. (October), 2012.
- [21] K. Stein, T. Tezduyar, and R. Benney. Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements. *Journal of Applied Mechanics*, 70(1):58, 2003.

- [22] S Turek, J Hron, M Razzaq, H Wobker, and M Sch. Fluid Structure Interaction II. 73, 2010.
- [23] Boris Valkov, Chris H Rycroft, and Ken Kamrin. Eulerian method for fluid – structure interaction and submerged solid – solid contact problems.
- [24] E. H. van Brummelen. Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction. *Journal of Applied Mechanics*, 76(2):021206, 2009.
- [25] Frank M White. Viscous Fluid Flow Viscous. *New York*, Second:413, 2000.
- [26] Frank M. White. Chapter 3 - Solutions of the Newtonian viscous-flow equations. *Viscous Fluid Flow*, (5), 2006.
- [27] 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.
- [28] Thomas Wick. Fluid-structure interactions using different mesh motion techniques. *Computers and Structures*, 89(13-14):1456–1467, 2011.