UiO : Department of Mathematics
University of Oslo

Monolithic solver for Computational Fluid Structure Interaction

Sebastian Gjertsen

Master's Thesis, Spring 2017

Contents

Chapter 1

Introduction to Fluid-Structure Interaction

The interaction between fluids and structures can be observed all around us in nature. From a flag waving in the wind to a large windmill at sea, when we breath in air and our lungs expand, when our hearts fill up with blood and muscles contract to push blood into our arteries, are all examples of fluids and structures interacting. A flag waving in the wind is mainly air(fluid) exerting force on the flag(structure), making the flag flutter. The structure can also move the fluid. When we breathe air into our lungs, our diaphram contracts and moves downward, increasing the space in our chest cavity making the lungs expand. As the lungs expand air is sucked through the mouth and nose, filling up the lungs. We can see that both fluid and structures can exert force, and consequently interact with eachother.

A bridge is an example of Fluid-Structure Interaction(FSI) where the bridge is a rigid structure, with wind interacting with the structure. A profound example of wind interacting with a bridge, is the collapse of the Tacoma Narrows Bridge in 1940 [1]. The bridge collapsed only two months after being opened. The collapse was due to strong winds(64 km/h) interacting with the bridge, making it flutter and ultimately collapse. No human life was lost in the collapse, but a Cocker Spaniel named Tubby left in a car was not so lucky.

Modeling FSI is used today when constructing for instance windmills. These windmills are usually rigid, hence giving a big difference in density between fluid and structure. The structure will in this case only give rise to small deformations. However modeling FSI in hemodynamics(dynamics of blood

Figure 1.1: Tacoma bridge still standing with large deformations

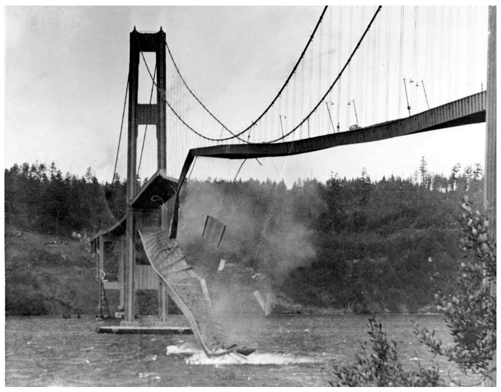


Figure 1.2: Tacoma bridge collapsed

flow) deems more challenging. In the latter cases the densities in fluid and solid are much more similar and blood flow usually induces large deformations and the veins and arteries. Also in both of these cases the fluid flow tends to transition to turbulence. We can see that these challenges gives the need to have rigid stabile FSI solvers, solvers which can handle large deformations and high fluid velocity.

A branch of fluid dynamics is called Computational Fluid Dynamics, where computers and numerical analysis is used to solve fluid problems. I would argue that a similar name is used for FSI, namely *Computational Fluid Structure Interaction* (CFSI). However in the literature FSI is used to describe CFSI and the name FSI will be used in this thesis. Also even though FSI is Fluid *Structure* Interaction, the word solid will also be used to describe a structure.

The main goal of this master thesis is to build a framework to solve the FSI problem, investigating different approaches and schemes. The framework will be validated and verified using the *Method of Manufactured Solutions*, accompanying a classic, well known benchmark, known to test the robustness of a FSI solver.

Chapter 2

Continuum mechanics in different frames of reference

All materials are made up of atoms, with vacuum between atoms. The laws that govern these atoms, are complex and very difficult to model. The length scales at which atoms are defined is in the nanometer ($10^{-9}m$) range. However if we zoom out to the *macroscale*, the scale on which phenomena are large enough to be visible for the human eye without help of magnifying instruments, matter like solids and fluids can be modeled if we assume them to exist as a continuum. The continuum hypothesis states that there exist no space inside the matter and the matter completely fills up the space it occupies. We can use mathematics with basic physical laws to model fluids and solids when they are assumed continuous. These laws are generally expressed in two frames of reference, Lagrangian and Eulerian, depending on the the scientific branch.

To exemplify these frameworks we can imagine a river running down a mountain. In the Eulerian framework we are the observer standing besides the river looking at the flow. We are not interested in each fluid particle but only how the fluid acts as a whole flowing down the river. This approach fits the fluid problem as we can imagine the fluid continuously deforming along the river side.

In the Lagrangian description we have to imagine ourselves on a leaf going down the river with the flow. Looking out as the mountain moves and we stand still compared to the fluid particles. This description fits a solid problem nicely since we are generally interested in where the solid particles are in relation to each other. For instance modeling a deflecting beam attached to a wall, one can imagine the deflection and to model this deflection one needs to where all the particles are compared to each other. The more force that is applied the more the particles move in relation to each other. The Lagrangian description helps us to easier model a problem where need to know the particles spatial-relation to each other

This chapter will be devoted to briefly introduce both the solid and fluid equations and their respective boundary conditions. A detailed description of the Lagrangian framework and the stress and strain relations are covered in the appendix.

2.1 Solid equation

The solid equation describes the motion of a solid. It is derived from the principles of conservation of mass and momentum. The solid equation is stated in the Lagrangian reference system [14], in the solid time domain $\mathcal{S}$ as:

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

written in terms of the deformation d. P is the first Piola-Kirchhoff stress tensor. See Appendix for a detailed description of strain and stress leading to the Piola-Kirchhoff stress tensor. f is a force acting on the solid body.

2.2 Fluid equations

The fluid equation is stated in an Eulerian framework. In the Eulerian framework the domain has fixed points where the fluid passes through. The Navier-Stokes(N-S) equations are, like the solid equation, derived using principles of mass and momentum conservation. N-S describes the velocity and pressure in a given fluid continuum. Written in the fluid time domain $\mathcal{F}$ as an incompressible fluid:

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

$$\nabla \cdot u = 0 \quad in \ \mathcal{F} \tag{2.3}$$

where u is the fluid velocity, p is the fluid pressure, ρ stands for density which, will be kept constant. f is body force and σ_f is the Cauchy stress tensor, $\sigma_f = \mu_f(\nabla u + \nabla u^T) - pI$ denoting a newtonian fluid. I is the Identity matrix.

There does not yet exist an analytical solutions to the N-S equations for every fluid problem. Only simplified fluid problems can be solved [22] analytically using the N-S equations. 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 solutions [4], as apart 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. Non-linearity is most often handled using a lagging velocity function, Newtons method, or Picard iterations. Another difficulty is finding a suitable equation to solve for the pressure field [?]. The difficulty with pressure has been tackled using for instance the incremental pressure correction scheme (IPCS).

2.3 Fluid and Structure Boundary conditions

To complete the fluid and solid equations, boundary conditions need to be imposed. The fluid flows and the solid moves within the boundaries noted as $\partial\mathcal{F}$ and $\partial\mathcal{S}$ respectively.

On the Dirichlet boundary, $\partial\mathcal{F}_D$ and $\partial\mathcal{S}_D$, we impose a given value. Dirichlet conditions can be initial conditions or set values, such as zero on the fluid boundary for a "no slip" condition. The Dirichlet conditions are defined for u and p :

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

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

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

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

The forces on the boundaries equal an eventual external force $\mathbf{f}$. These are

are enforced on the Neumann boundaries $\partial\mathcal{F}_N$ and $\partial\mathcal{S}_N$:

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

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

Chapter 3

Fluid Structure Interaction Problem

This chapter will be devoted to introducing the full FSI problem, with all the equations, conditions, and discretizations needed to to build a Fluid Structure Interaction solver.

When computing FSI problems the computing domain is split into three parts, 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 gives the two main methods for solving FSI problems [8]. 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 [19]. The fully Eulerian description is suited for fluid problems but is problematic for structure problems and certainly FSI where tracking of the interface across the domain is a difficult task.

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 best of both world, in that fluid and solid are described in their natural states. The structure equation will remain as previously stated (2.1), and we will need to change the fluid equations to take into account the moving domain changing the fluid velocity. Dealing with the movement of the domain is done 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 every time. Moving the domain gives advantages as we can explicitly represent the fluid-structure interface, and the equations are stated in a more familiar manner. But problems arise when there are large deformations in the solid giving large deformations into the fluid domain. Moving the mesh with large deformation can be a challenge in that the domain can overlap causing singularities.

In this thesis the ALE approach is used from a reference frame. 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 in the domain that determines the value of the mappings between frames of reference.

From a technical point of view, moving both the mesh and using a reference frame are equivalent [14]. 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.

There are generally two types approaches when discretizing an FSI scheme. The first is the partitioned approach where fluid and structure are solved sequentially. The partitioned approach is appealing in that we have a wealth of knowledge and techniques on how 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 needed in FSI, and the coupling of these conditions is where problems arise. 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 [6], [20]. 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 [5]. Later a scheme was made with an extension coupling with a thick walled structure by Fernandez et.al,2015 [6]. The scheme for coupling thick walled structures is rather complex and uses a number of techniques that are out of the scope of this thesis. (This may be in more detail in a later chapter (discussion and further work.))

In this thesis the other approach is used named monolithic. In the monolithic approach all of the equations are solved at once. This approach has the advantage of offering numerical stability for problems with strong added-mass effects [8], and are fully coupled. The disadvantage over the partitioned approach is that we loose flexibility when solving many equations simultaneously, and the problems can quickly become large and computationally costly. But the overall the ease of implementation and numerical stability makes monolithic the preferred choice.

This chapter will start by introducing the mappings needed to change between current and reference domain. Lastly the equations will be discretized following the notation and ideas from [14]:

3.1 Mapping between different frames of reference

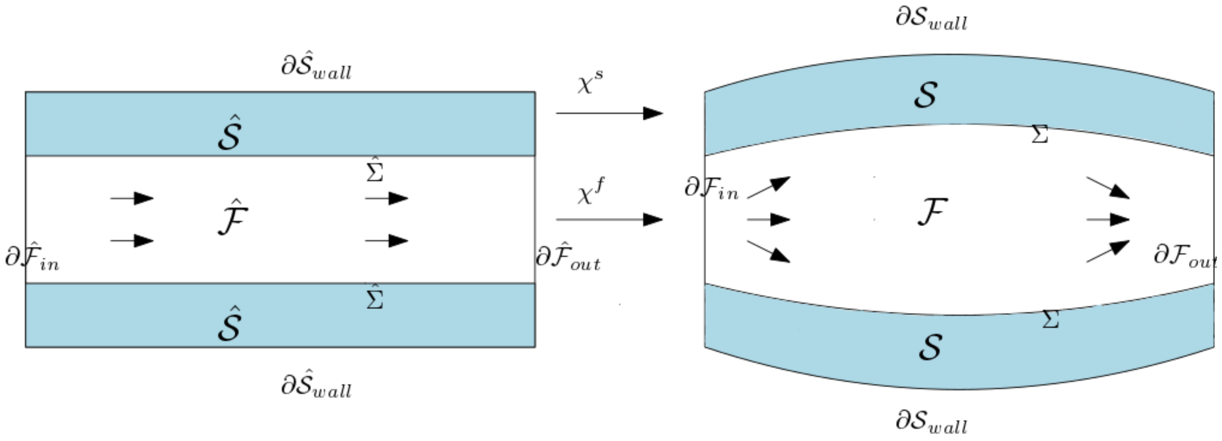

$\partial\hat{\mathcal{S}}_{wall}$
$\hat{\mathcal{S}}$
$\hat{\Sigma}$
$\hat{\mathcal{F}}$
$\partial\hat{\mathcal{F}}_{in}$
$\partial\hat{\mathcal{F}}_{out}$
χ^s
χ^f
$\partial\mathcal{S}_{wall}$
$\mathcal{S}$
Σ
$\mathcal{F}$
$\partial\mathcal{F}_{in}$
$\partial\mathcal{F}_{out}$

Figure 3.1: Mapping of domain from reference to current

Let $\hat{\mathcal{V}}$ be a reference volume and $\mathcal{V}(t)$ be the current time volume. Then using (A.5) and (A.6) we define a mapping between the volumes from the current to reference configurations:

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

The gradients acting on a vector $\mathbf{u}$ will also be mapped between current and reference configurations:

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

Same for the divergence of a vector $\mathbf{u}$:

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

3.2 Governing equations for Fluid Structure Interaction

In this section I formulate the equations in the Eulerian, Lagrangian and the ALE description. I start by Briefly talking about time derivatives in the different configurations, to understand the need to change the fluid equation in the ALE description [24]:

3.2.1 Derivatives in different frameworks

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

in the Eulerian framework we have the following relation between total and partial derivatives:

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

whilst in the ALE framework we extend this concept to take into account the motion of the domain:

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

For the Lagrangian framework $\mathbf{w}$ is zero and for Eulerian $\mathbf{w} = \mathbf{u}$.

3.2.2 Solid equation

Let $\Omega \in \hat{\mathcal{S}} \cup \hat{\mathcal{F}}$ be a global domain that is made up of the fluid, structure, and the interface. The interface is stated as: $\hat{\Sigma} \in \hat{\mathcal{S}} \cap \hat{\mathcal{F}}$. We define a global velocity function $\mathbf{u}$ 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 a global velocity function makes the velocity continuous across the entire domain. .

The solid equation will be written in terms of the solid velocity $\mathbf{u_s}$, in contrast to 2.1 which was written in terms of the deformation. We express the solid balance laws in the Lagrangian formulation from the initial configuration:

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

3.2.3 Fluid equations

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

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

d is the deformation in the fluid domain. We see that the domain velocity $\mathbf{w}$ is written as $\frac{\partial \mathbf{d}}{\partial t}$. The actual fluid convection in the ALE approach will be $(\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}) \cdot \nabla \mathbf{u}$

I now show the change in the fluid equations denoted from the initial configuration using the aforementioned mappings:

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

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

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

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

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

Assembling all these terms together with (2.2) **gives the fluid equations from a reference frame**:

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

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

3.3 Lifting operators for upholding mesh quality

Using the ALE method the solid domain moves into to fluid domain. The deformations from the structure through the interface is extrapolated to the fluid domain using different lifting operators. The lifting operators redistributes the interior node locations to uphold the mesh quality. The choice of lifting operator is important for the overall FSI problem to be calculated [25]. When large deformations occur, we need a good lifting operator to uphold the integrity of the computing domain. A poor choice will make the mesh overlap and singularities may occur. 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 5.2.2 the techniques will be tested and investigated.

3.3.1 Harmonic extension

For small to moderate deformations we can use a harmonic extension. The harmonic extension uses the Laplace equation, transporting the deformations from the solid into the fluid domain. A variable $\alpha_u > 0$ can be multiplied to the Laplace equation, to control the amount of lifting of deformations to the fluid domain.

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

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

$$\mathbf{d}_f = \mathbf{d}_s \quad on \ \hat{\Sigma} \tag{3.18}$$

When using the harmonic lifting operator the variable α_u is very important when calculating moderate deformations. For small deformations a constant can be used for α_u. But for larger deformations we need to be a bit more clever. A good strategy for choosing α_u is proposed by Wick in [25], and further discussed in [18] and [11]. Which is an alpha that gets bigger when closer to the interface.

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

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

3.3.2 Biharmonic extension

The last extension is the biharmonic extension. The biharmonic extension provides more freedom than the harmonic and linear elastic in choosing boundary conditions and choice of parameter $\alpha_u > 0$. This is because the biharmonic extension, extends the deformation in a manner that upholds the integrity of the cells even in large deformations, even with α_u as a constant. In its simplest form it is written as:

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

The biharmonic extension is calculated with a mixed formulation where we introduce a new function ω (not to be confused with the deformation velocity), this function is added to the system so that we solve for 4 functions:

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

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

$$\begin{aligned} \mathbf{d} = \partial_n \mathbf{d} =& 0 \quad on \;\; \partial\hat{\mathcal{F}} \setminus \hat{\Sigma} && (3.22) \\ \mathbf{d}_f =& \mathbf{d}_s \quad on \;\; \hat{\Sigma} && (3.23) \end{aligned}$$

The second imposes conditions on $\mathbf{d}$ and ω, and are written in terms of single component functions $d^{(1)}, d^{(2)}$ and $\omega^{(1)}, \omega^{(2)}$

$$\begin{aligned} \mathbf{d}^{(1)} =& \partial_n d^{(1)} = 0, and \quad \omega^{(1)} = \partial_n \omega^{(1)} = 0 \quad on \;\; \partial\hat{\mathcal{F}}_{in,out} && (3.24) \\ \mathbf{d}^{(2)} =& \partial_n d^{(2)} = 0, and \quad \omega^{(2)} = \partial_n \omega^{(2)} = 0 \quad on \;\; \partial\hat{\mathcal{F}}_{walls} && (3.25) \\ & && (3.26) \end{aligned}$$

Since the biharmonic extension is of fourth order character it will have a higher computational cost [14] than the harmonic or linear elastic.

3.4 Interface conditions

In figure 3.1 we see a typical Fluid Structure Interaction. The fluid is surrounded by elastic walls, like a blood vessel. The inflow at $\partial\hat{\mathcal{F}}_{in}$, a change in pressure, or a motion of the solid determines the flow of the fluid. The fluid‘s stress on the walls causes deformation in the solid domain and vice versa. The interface is where these energies are transferred and we therefore need conditions on the interface.

The three interface conditions comes from simple physical properties and consist of [14]:

- *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 $\mathbf{u}_s = \frac{\partial d}{\partial t}$. Setting $\mathbf{u}_f = \frac{\partial d}{\partial t}$ at the interface.

- *Dynamic condition*: $\sigma_f n_f = \sigma_s n_s \ \ on \ \ \hat{\Sigma}$.
  This relates to Newtons third law of action and reaction. The forces on the interface area, here written as the normal stresses are balanced on the interface. These will be written in a Lagrangian formulation: $J\sigma_f F^{-T} n_f = P n_s \ \ on \ \ \hat{\Sigma}$.
  The dynamic condition is a Neumann condition that belongs to both subproblems.

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

3.5 Discretization of monolithic FSI equations

After introducing all the equations and boundary conditions needed to solve a FSI problem. We are ready to discretize the equations into one monolithic scheme. The equations will be discretized and solved using finite difference and finite element methods. I will introduce a so called θ-scheme which will make it easy to implement different schemes, by choosing a value for θ. I will briefly introduce the spaces needed to discretize, following the ideas and notations of [24]:

Spaces

Let $X \in \mathbb{R}^d, d \in \{1,2\}$ be a time dependent domain we define:

$$\hat{V}_X := H^1(X), \qquad \hat{V}_X^1 := H_0^1(X) \tag{3.27}$$

H^1 indicating a Hilbert space and

$$\hat{L}_X := L^2(X), \qquad \hat{L}_X^0 := L^2(X)/\mathbb{R} \tag{3.28}$$

L^2 indicating a standard Lebesque space.

The trial and test spaces for the velocity variable in the fluid domain,

$$\hat{V}^0_{f,u} := \{\hat{u} \in H^1_0(\mathcal{F}) : \hat{u}_f = \hat{u}_s \ \ on \ \ \hat{\Sigma}\} \tag{3.29}$$

and the same for the artificial displacement in the moving fluid domain:

$$\hat{V}^0_{f,d} := \{\hat{d} \in H^1_0(\mathcal{F}) : \hat{d}_f = \hat{d}_s \ \ on \ \ \hat{\Sigma}\} \tag{3.30}$$

$$\hat{V}^0_{f,d} := \{\hat{d} \in H^1_0(\mathcal{F}) : \hat{\psi}_f = \hat{\psi}_s \ \ on \ \ \hat{\Sigma}\} \tag{3.31}$$

Now that the spaces have been defined we are ready to discretize. The temporal discretization is done using finite difference schemes and the spatial is treated with the finite element method. I will employ a $\theta-$ scheme that will enables us to easily switch between schemes.

Discretization

In the domain Ω and time interval [0,T]:

Find $U = \{\mathbf{u}, d, p\} \in \hat{X}^0_D$ where $\hat{X}^0_D := \{\mathbf{u}^D_f + \hat{V}^0_{f,\mathbf{u}}\} \times \hat{L}_f \times \{d^D_f + \hat{V}^0_{f,\hat{f}}\} \times \{d^D_f + \hat{V}^0_{f,\hat{f}}\} \times \hat{L}^0_f \times \hat{L}^0_s$ such that:

$$\int_0^T A(U)(\Psi)dt = \int_0^T \hat{F}(\Psi)dt \quad \forall \Psi \in \hat{X} \tag{3.32}$$

where $\Psi = \{\phi, \psi, \gamma\}$

$\hat{X} = \hat{V}^0_{f,\mathbf{u}} \times \hat{L}_f \times \hat{V}^0_{f,d,\hat{\Sigma}} \times \hat{V}^0_s \times \hat{L}^0_f \times \hat{L}^0_s$

I first introduce the scheme using, for simplicity, the harmonic mesh motion. Let $\mathbf{u}$ be a global function in the entire domain instead of $\mathbf{u}_f$ in the fluid and $\mathbf{u}_s$ in the solid. Same for the test functions. This is done for ease of reading and also for the ease of implementation later.

The full monolithic FSI variational form reads:

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

The condition (3.37) , is weighted with a δ value. This is a critical detail for the program (detailed later) to run. The only two places where we use the test function ψ is on this condition and the lifting operator. The weighting says in a weak manner that this condition is important for the overall program.

I will here formulate the *One step-θ scheme* from [24]. This θ scheme has the advantage of easily being changed from the backward (implicit), forward(excplicit) or Crank-Nicholson (implicit) scheme. The Crank-Nicholson scheme is of second order, but suffers from instabilities in this monolithic scheme [24]. A remedy for these instabilities is to chose a Crank-Nicholson scheme that is shifted towards the implicit side. How this is done will become evident once the scheme is defined.

We define the variational form by dividing into four categories, this may seem strenuous at first but the need for it will become evident when implementing the θ-scheme. The four divided categories consists of: a time term, implicit, pressure and the rest (stress, convection):

$$
\begin{aligned}
A_T(U) &= (J\rho_f \partial_t \mathbf{u}, \phi) - (J(\nabla u)F^{-1}(\partial_t \mathbf{d}), \phi)_{\hat{\mathcal{F}}} && (3.39)\\
&+ (\rho_s \partial_t \mathbf{u}, \phi)_{\hat{\mathcal{S}}} + (\partial_t \mathbf{d}, \psi)_{\hat{\mathcal{S}}} && (3.40)\\
A_I(U) &= (\alpha_u \nabla \mathbf{u}, \nabla\psi)_{\hat{\mathcal{F}}} + (\nabla\cdot(JF^{-1}\mathbf{u}), \gamma)_{\hat{\mathcal{F}}} && (3.41)\\
A_E(U) &= (J(\nabla u)F^{-1}\mathbf{u}, \phi)_{\hat{\mathcal{F}}} + (J\sigma_{f,u}F^{-T}, \nabla\phi)_{\hat{\mathcal{F}}} && (3.42)\\
&+ \big(FS_s, \nabla\phi\big)_{\hat{\mathcal{S}}} - (\mathbf{u}, \psi)_{\hat{\mathcal{S}}} && (3.43)\\
A_P(U) &= \big(J\sigma_{f,p}F^{-T}, \nabla\phi\big) && (3.44)
\end{aligned}
$$

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

$$\sigma_{f,u} = \mu(\nabla u F^{-1} + F^{-T}\nabla u) \tag{3.45}$$

$$\sigma_{f,p} = -pI \tag{3.46}$$

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

$$A_T(U^{n,k}) \approx \frac{1}{k}\big(\rho_f J^{n,\theta}(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi\big)_{\hat{\mathcal{F}}} - \frac{1}{k}\big(\rho_f(\nabla u)(\mathbf{d}^n - \mathbf{d}^{n-1}), \phi\big)_{\hat{\mathcal{F}}} \tag{3.47}$$

$$+\frac{1}{k}\big(\rho_s(\mathbf{u}^n - \mathbf{u}^{n-1}), \phi\big)_{\hat{\mathcal{S}}} + \frac{1}{k}\big((\mathbf{d}^n - \mathbf{d}^{n-1}), \psi\big)_{\hat{\mathcal{S}}} \tag{3.48}$$

And the Jacobian is written with superscript θ as:

$$J^{n,\theta} = \theta J^n + (1-\theta)J^{n-1} \tag{3.49}$$

We can now introduce the *One step-θ scheme*: Find $U^n = \{\mathbf{u}^n, \mathbf{d}^n, p^n\}$

$$A_T(U^{n,k}) + \theta A_E(U^n) + A_P(U^n) + A_I(U^n) = \tag{3.50}$$

$$-(1-\theta)A_E(U^{n-1}) + \theta\hat{f}^n + (1-\theta)\hat{f}^{n-1} \tag{3.51}$$

We notice that the scheme is selected by the choice of θ. By choosing $\theta = 1$ we get the back 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 which will be ignored in this thesis. The shifting toward the implicit side is important for long term stability for certain time step values [24]. The shifting will be investigated in the next chapter.

Chapter 4

Implementation of Fluid Structure Interaction in FEniCS

This Chapter shows the implementation of the FSI code in FEniCS [9]. FEniCS is platform used to solve partial differential equations. Code is written in python and FEniCS makes it easy to 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.

4.1 Mappings

The deformation gradient and the Jacobian is made using python functions. The deformation gradient python function takes in the deformation and returns the deformation gradient. Identity in FEniCS is a function that makes the identity matrix, with the size determined by a input value. The identity takes the length of the deformation function which for 2D cases is 2. grad is a built in FEniCS function that takes a vectorfunction and take the gradient.

```python
def F_(d): """inputs deformation """
```

```python
        return (Identity(len(d)) + grad(d))  """Returns the deformation
    gradient"""

def J_(d):
        return det(F_(d))
```

4.2 Variational form

The variational form can be written directly into FEniCS. A big advantage in FEniCS is that one can write all the forms and add them together to make the full variational form. 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.

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

theta is the value θ that determines the scheme. k is the timestep Δt F_fluid_linear is the linear part of the variational form, and $F_fluid_nonlinear$ is the nonlinear part. Each part is assembled later when using newtons-method.

Constant is a function in FEniCS that represents a constant value unknown at compile-time. Its value can be changed without the need to re-compilation of C++ code, which FEniCS utilizes. $sigma_f$ and $sigma_f$ is the velocity and pressure parts of the fluid stress tensor. grad and div are built-in functions in FEniCS that computes the gradient and divergence and given vector. dx_f denotes the fluid domain. $inv()$ is a built-in function in FEniCS outputs the inverse of the given matrix. psi, phi and gamma are the test functions.

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

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

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

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

```

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

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

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

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

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

Fsolidlinear is the linear part of the variational form, and Fsolidnonlinear is the nonlinear part. Piola1 is the first Piola-Kirchhoff stress tensor.

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

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

The weighed delta is talked about in section 3.5

4.3 Newtons method implementation for solving Fluid structure interaction

To solve a non-linear FSI problem, the newtons method has been implemented, ideas and code taken from from [23].

The derivative of the nonlinear variational parts is take with respect to dvp, which is the mixed function of displacement, velocity and pressure. -F is assembly of full variational form.

There is also an if test which only assembles the Jacobian the first and tenth time. This reuses the Jacobian to improve runtime, which is discussed in chapter 6. Lastly the the mpi line is when the code is running in parallel.

```python
def newtonsolver(F, J_nonlinear, A_pre, A, b, bcs, \
                dvp_, up_sol, dvp_res, rtol, atol, max_it, T, t, **monolithic):
    Iter      = 0  """ Setting initial values """
    residual   = 1
    rel_res    = residual
    lmbda = 1

    while rel_res > rtol and residual > atol and Iter < max_it:
        if Iter % 10 == 0:
            A = assemble(J_nonlinear, tensor=A, form_compiler_parameters = {"quadrature_degree": 4})
            A.axpy(1.0, A_pre, True)
            A.ident_zeros()

        b = assemble(-F, tensor=b)

        [bc.apply(A, b, dvp_["n"].vector()) for bc in bcs]
        up_sol.solve(A, dvp_res.vector(), b)
        dvp_["n"].vector().axpy(lmbda, dvp_res.vector())
        [bc.apply(dvp_["n"].vector()) for bc in bcs]
        rel_res = norm(dvp_res, 'l2')
        residual = b.norm('l2')
        if isnan(rel_res) or isnan(residual):
            print "type rel_res: ",type(rel_res)
            t = T*T

        if MPI.rank(mpi_comm_world()) == 0:
            print "Newton iteration %d: r (atol) = %.3e (tol = %.3e), r (rel) = %.3e (tol = %.3e) " \
        % (Iter, residual, atol, rel_res, rtol)
        Iter += 1

    return dict(t=t)
```

Chapter 5

Verification and Validation of the Fluid Structure Interaction Implementation.

The general approach when solving a real world problem with numerical computing, starts by defining the mathematics, implementing the equations numerically and solve the equations on a computer.

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 answer another question, are the equations solved correct numerically, if so is the problem defined correct mathematically. Without answering these questions, being confident that your solutions are correct is difficult [17]. 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. Following will be Validation of the code where I implement at a well known benchmark testing the fluid and structure parts individually and as a full FSI problem. After that I investigate the impact of using different time order schemes, and different lifting operator.

5.1 Verification

Verification, in the context of scientific computing, is the process of determining wether or not the implementation of numerical algorithms in computer code, is done correctly. [12]. In verification, we get evidence that the numerical model derived from mathematics is solved correctly by the computer. The strategy is to identify, quantify and reduce errors cause by mapping a mathematical model to a computational model. Verification does not address wether or not the computed solutions are in alignment with physics in the real world. It only tells us that our model is computed correctly or not. To verify that we are computing correctly we can compare our computed solution to an exact solution. But the problem is that there are no known exact solutions to, for instance, the Navier-Stokes equations, other than for very simplified problems.

In Verification there are multiple classes of test that can be performed, and the most rigorous is the *Method of Manufactured Solution*(MMS) [12]. 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 than run the PDE with the source term to get a solution hopefully matching the manufactured one. The manufactured solution does not need to have a physically realistic relation, since the solution deals only with the mathematics. The overall idea is to make solutions in a way that all terms of the given PDE are tested. The procedure of MMS is as follows [12]:

- 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 $\mathbf{u}$, $L(\mathbf{u}) = f$

After the solution has been computed we perform systematic convergence tests [15]. The idea behind order of convergence tests is based on the behavior of the error between the manufactured exact solution and the computed solution. 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:

$$E = ||\mathbf{u} - \hat{\mathbf{u}}|| \tag{5.1}$$

When we decrease the node spacing ($\Delta x, \Delta y$ or Δz) or decrease timestep size(Δt), we expect the solution to convergence towards a given solution and hence the error to get smaller. It is the rate of this error that lets us know wether the solution is converging correctly. If we assume that the number of spatial points are equal in all directions the error is expressed as

$$E = C_1 \Delta x^k + C_2 \Delta t^l \tag{5.2}$$

where $k = m+1$ and m is the polynomial degree of the spatial elements. The error is hence dependent on the number of spatial points and the timestep. If we for instance reduce Δt significantly, Δx will dominate, and Δt will be negligible. If we then look at two error where E_{n+1} has finer mesh than E_n, using (5.2):

$$\frac{E_{n+1}}{E_n} = \left(\frac{\Delta x_{n+1}}{\Delta x_n}\right)^k \tag{5.3}$$

$$k = \frac{log(\frac{E_{n+1}}{E_n})}{log(\frac{\Delta x_{n+1}}{\Delta x_n})} \tag{5.4}$$

We use k to find the observed order of convergence and match with the theoretical order of convergence for each given problem.

The manufactured solutions should be chosen to be non-trivial and analytic [12, 15]. The solutions should be manufactured so that no 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.

Starting with the verification of the solid part of the code with given displacement and velocity. Then verifying the fluid part with a given displacement, also testing the mappings between configurations. To do a full verification of the entire FSI problem, one needs to take into account the condition of continuity of velocity on the interface [3], the stresses need to equal on the interface and the flow needs to be divergence free. Manufacturing such a solution is very difficult [2]. The author has yet to find a paper that manufactures a solution fulfilling all the condition in a rigorous manner. MMS of full FSI is therefore out the scope of this thesis.

5.1.1 Method of Manufactured Solution on the implementation of the Solid equation

The first MMS is with the solid problem alone. Testing the solid equation 3.7 with the condition $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$.

Solutions $\hat{\mathbf{d}}$ and $\hat{\mathbf{u}}$ are manufactured to meet the requirements of MMS such as smoothness and complexity, choosing functions with sine and cosine. The derivatives does not become zero and we have time and space dependencies.

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

The manufactured solutions are used to make a sourceterm f_s:

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

The solid variational formulation is written in weak form with test functions ϕ and ψ as:

$$\big(\rho_s \frac{\partial \mathbf{u}}{\partial t}, \phi\big)_{\hat{\mathcal{S}}} + \big(P, \nabla \phi\big)_{\hat{\mathcal{S}}} = f_s \tag{5.5}$$

$$\big(\mathbf{u} - \frac{\partial \mathbf{d}}{\partial t}, \psi\big)_{\hat{\mathcal{S}}} = 0 \tag{5.6}$$

These equations are solved together and we solve for d and u on a unit square domain. 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. The computations ran with 10 timesteps and the error was calculated for each time step and then the mean of all the errors were used to calculate the convergence rates.

Even though there are two equations, we do not make a source for the second equation, since the solutions are made to uphold the criteria $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$.

In the tables below we investigate convergence in space and time. Starting with checking order of convergence in space, setting m = 1, the expected order of convergence will be 2.

N	Δt	**m**	E_u	$\mathbf{k_u}$	E_d	$\mathbf{k_d}$
4	1×10^{-7}	**1**	0.0068828	**-**	3.7855×10^{-9}	**-**
8	1×10^{-7}	**1**	0.0017204	**2.0002577**	9.4622×10^{-10}	**2.0002577**
16	1×10^{-7}	**1**	0.0004300	**2.0000622**	2.3654×10^{-10}	**2.0000622**
32	1×10^{-7}	**1**	0.0001075	**2.0000154**	5.9136×10^{-11}	**2.0000154**
64	1×10^{-7}	**1**	0.0000268	**2.0000038**	1.4783×10^{-11}	**2.0000038**

Table 5.1: Method of Manufactured Solution on the implementation of the Solid equation in space with m = 1

Lastly I check convergence in time. The scheme tested is of first order, with a value of $\theta = 1$, hence a order of convergence of 1 is expected. Here i set N = 64 and the timestep is halved for each computation.

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

Table 5.2: Method of Manufactured Solution on the implementation of the Solid equation in time

Comments on the Solid MMS

The table 5.1 shows a decrease in error with increased number of spatial points. The order of convergence tends toward the expected value of 2, hence increasing our confidence in the implementation of the solid equation.

Table 5.2 shows a order of convergence tending towards 1, which was expected. With order of convergence tests passed for both time and space for the solid equation and the condition $\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t}$, the implementation of the solid equation is satisfactory.

5.1.2 MMS of Fluid equations with prescribed motion

Starting by prescribing a motion to d and w and manufacturing solutions $\hat{\mathbf{u}}$ and $\hat{p}$. Setting $\hat{\mathbf{u}} = \mathbf{w}$:

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

Solutions are made to uphold the criterias :

$$\nabla \cdot \mathbf{u} = 0, \ \frac{\partial \mathbf{d}}{\partial t} = \mathbf{w}$$

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 hence made without mappings:

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

Then f_f is used and mapped to the reference configuration and computed:

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

The computations are done on a unit square domain and the computations ran with 10 timesteps and the error was calculated for each time step and then the mean of all the errors was taken and used to calculate the convergence rates.

The scheme tested for convergence time is of first order, with a value of $\theta = 1$, hence a order of convergence of 1 is expected. Here i set N = 64 and the timestep is halved for each computation.

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

Table 5.3: MMS ALE FSI u=w

In testing the spatial convergence I set $m = 2$, hence giving an expected order of convergence in the velocity of 3 and pressure of 2.

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

Table 5.4: MMS ALE FSI u=w

Comments on the Solid MMS

In table **??** the order of convergence converges as expected towards 1. In 5.4 the MMS testing convergence in space converges toward 3 for velocity and 2 for pressure as was expected when setting $m = 2$. Both results helps in building confidence that the implementation of the mapped fluid equation is correct.

5.2 Validation

After the code has been verified, we move on to Validation which is the process of determining if a model gives an accurate representation of real world physics within the bounds of the intended use [17]. A model is made for a specific purpose, its only valid with respect to that purpose [10]. If the purpose 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 build more complexity and eventually test the whole model.

Three issues have been identified in this process [17]: 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. Well known benchmarks will be used as reference points and these tests supply us with a problem setup, initial and boundary conditions, and lastly results that we can compare with.

The process of Validation is also, as I have experienced, a way to figure out at what size timestep and number of spatial points the model can handle. As we will see in the chapter all the benchmarks are run with different timesteps and number of cells to see how the model reacts. 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 [17] puts it "Knowing the "correct" answer beforehand is extremely seductive, even to a saint.". Knowing the limitations of our tests will therefore strengthen our confidence in the model. It really can be an endless process of verifying and validating if one does not clearly know the bounds of sufficient accuracy. [17]

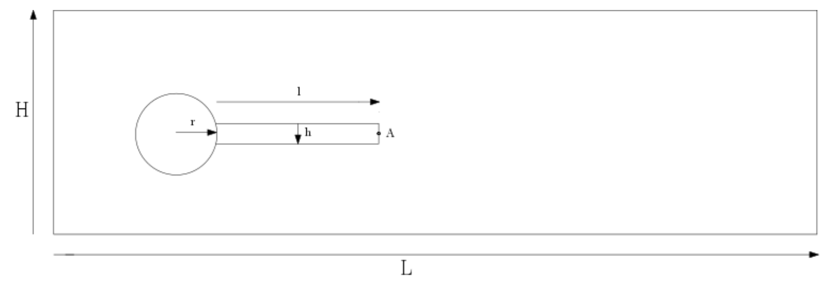
The following tests are done with the solid and fluid alone. Testing both steady and unsteady cases. Lastly the full FSI model is tested with steady and unsteady cases.

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

The goal of this benchmark is to verify the fluid and solid solver first separately and then together as a full FSI problem [7]. This benchmark is based on the older benchmark " flow around cylinder" with fluid considered incompressible and in the laminar regime where the structure deformations are large, in one case the deformations are about 2.5 larger than the height of the elastic object. The problem is setup with the solid submerged in the fluid, so that oscillations in the fluid can deform the structure. Measuring the drag and lift around the circle and bar, and measure structural displacement at a given point. This benchmark will be used to test and verify different numerical methods and code implementations. Testing the robustness and efficiency of the FSI implementation

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 of center from bottom to top, **this shift in the domain is done to induce oscillations to an otherwise steady flow**. The fluid oscillations gives a force to the elastic bar. The parameters of the domain are:
L = 2.5, H = 0.41, l = 0.35, h = 0.02, A = (0.2, 0.6)

Boundary conditions

A parabolic profile has been prescribed to the velocity that increases over time until a time = 2 has been reached.

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

Setting no slip on the "floor" and "ceiling" so to speak.

$$u(x,y,t) = 0 \text{ on } \partial\mathcal{F}_{floorandceiling}$$
$$p(x,y,t) = 0 \text{ on } \partial\mathcal{F}_{outlet}$$

Quantities for comparison

When the fluid moves around the circle and bar it exerts a force. These 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. This point is used to track the deformation.
In the unsteady time dependent problems, the values are represented by mean and amplitude values:

$$mean = \frac{1}{2}(max + min) \tag{5.7}$$
$$amplitude = \frac{1}{2}(max - min) \tag{5.8}$$
$$\tag{5.9}$$

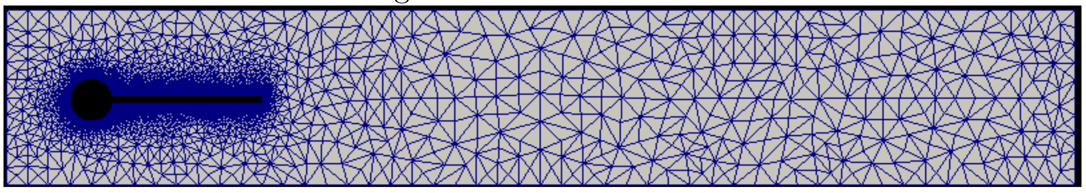
In each test the numbers with ref are the values taken from the original benchmark paper [7]

Results

CFD test

The first two CFD tests are run with Reynolds numbers 20 and 100 converging to steady solutions of drag and lift around the circle. The CFD 3 problem has a Reynolds number 200 which will induce oscillations behind the circle, giving fluctuations in the drag and lift, hence giving unsteady solutions. The CFD tests can be run in two ways. The first way assumes the bar to be rigid object, meaning that the computing domain is just the fluid domain. The other way, which is implemented here, is running the full FSI code and setting $\rho_s = 10^6$ and $\mu_s = 10^{12}$, so that the bar is almost completely rigid, only giving rise to very small deformation (in the 10^{-9} range).

Figure 5.1: Fluid mesh



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

Table 5.5: Summary of all the parameters in CFD tests

elements	**dofs**	**Drag**	**Lift**
6616	32472	14.2439	1.0869
26464	124488	14.2646	1.11085
105856	487152	14.2755	1.11795
ref		**14.29**	**1.119**

Table 5.6: CFD 1

elements	**dofs**	**Drag**	**Lift**
6616	32472	135.465	6.27158
26464	124488	136.566	9.82166
105856	487152	136.573	10.4441
ref		**136.7**	**10.53**

Table 5.7: CFD 2 with

elements	**dofs**	**Drag**	**Lift**
6616	32472	-10.42 ± 446.57	435.13 ± 5.02
26464	124488	-18.42 ± 445.57	440.73 ± 5.22
ref	ref	$\mathbf{-11.893 \pm 437.81}$	$\mathbf{439.45 \pm 5.61}$

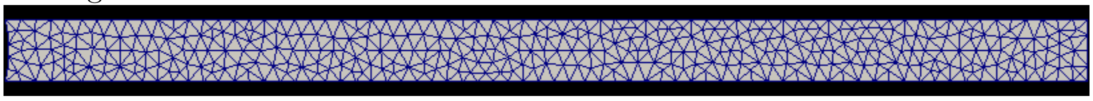
Table 5.8: CFD 3 with $\Delta t = 0.001$

CSM test

The CSM test are calculated using only the bar and adding a gravity term g with the same value but changing the parameters of solid. The tests CSM1 and CSM2 are steady state solutions. The difference is a more slender bar. The CSM 3 test is unsteady and even more slender causing the bar to move up and down. Since there is no resistance from any fluid the unsteady test should if energy is preserved make the bar move up and down infinitely.

Our quantity for comparing there will be the deformation at the point A.

Figure 5.2: 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
ν_s	0.4	0.4	0.4
$\mu_s[10^6\frac{m^2}{s}]$	0.5	2.0	0.5
g	2	2	2

Table 5.9: Summary table of the parameters used in the CSM tests

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

Table 5.10: Results of the steady CSM1 case from coarse to fine mesh.

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

Table 5.11: Results of the steady CSM2 case from coarse to fine mesh.

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

Table 5.12: Results of the unsteady CSM3 case with mesh from coarse to fine.

Figure 5.3: Results for CSM3 showing Displacement of point A

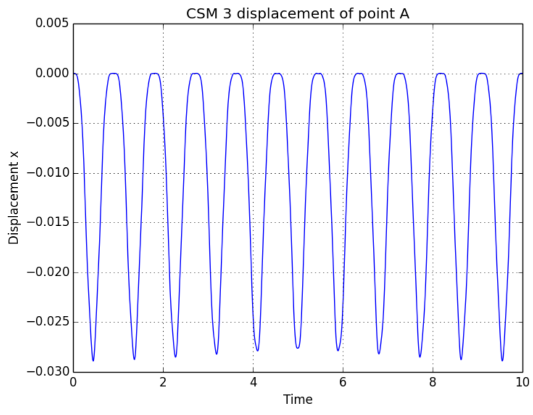


Figure 5.4: Displacement in x direction, timeinterval (0,10)

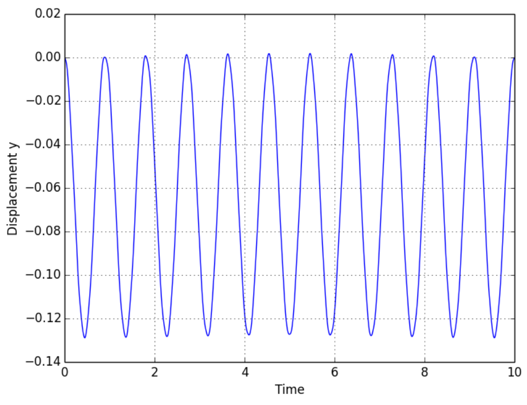


Figure 5.5: Displacement in x direction, timeinterval (0,10)

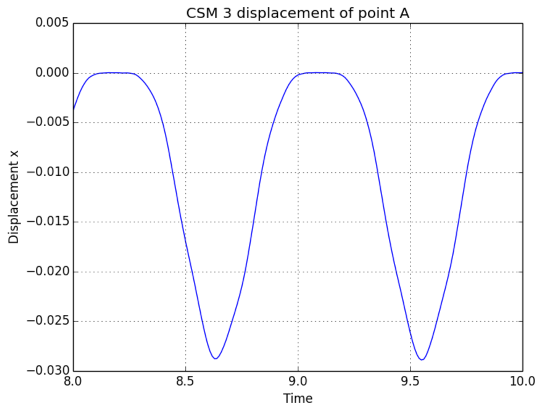


Figure 5.6: Displacement in x direction, timeinterval (8,10)

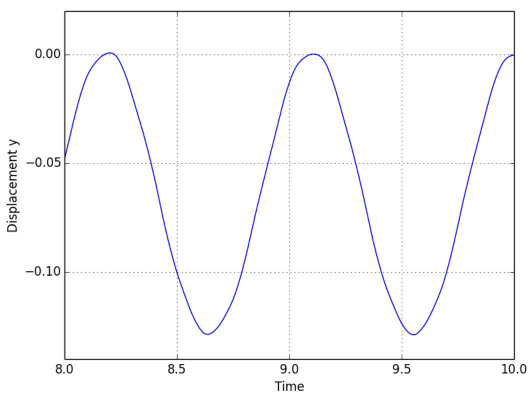


Figure 5.7: Displacement in x direction, timeinterval (8,10)

Discussion of the CSM results

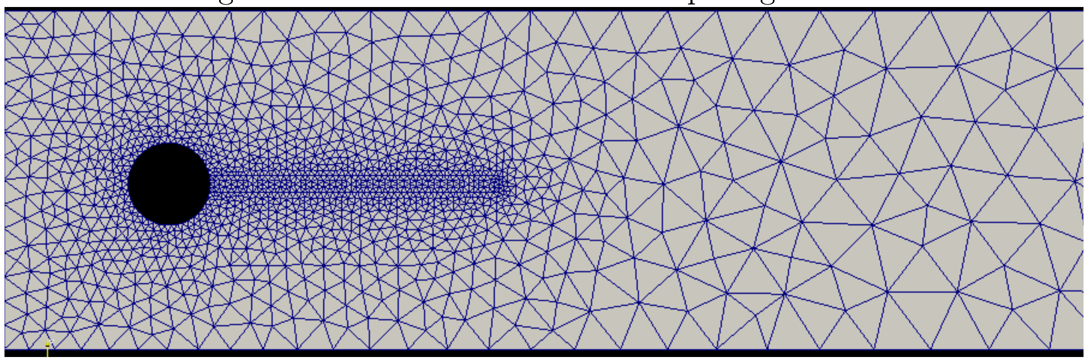
The tables 5.10, 5.11 shows the results of the two steady state cases with different bar-stiffness. As the number of spatial points in the domain increases the solutions converge towards the referential value. The CSM3 case shown in table 5.12 shows the same trend of converging toward referential value when the number of spatial points are increased.

The plot 5.2.1 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.

FSI tests

Lastly the full FSI problem ran. Here we can see in 5.2.2 that now both fluid and structure has a mesh. The 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 [13], giving deformations of 2.5 times greater than the flag height. Vinje 2016 [21] reported not being able to compute FSI-2 model due to shortcomings in the linear elasticity solid model. The FSI-3 test has the highest inflow speed giving medium deformations but more rapid oscillations.

Figure 5.8: Fluid and Structure computing mesh



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

Table 5.13: FSI Parameters

FSI1

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift	Spaces
2698	23563	0.0227418	0.799314	14.1735	0.761849	P2-P2-P1
10792	92992	0.0227592	0.80795	14.1853	0.775063	P2-P2-P1
43168	369448	0.0227566	0.813184	14.2269	0.771071	P2-P2-P1
ref	**ref**	**0.0227**	**0.8209**	**14.295**	**0.7638**	**ref**

Table 5.14: FSI 1

FSI2

Cells	Dofs	$d_x(A)[x10^{-3}]$	$d_y(A)[x10^{-3}]$	Drag	Lift
2698	23563	-15.10 ± 13.32	1.16 ± 82.46	159.53 ± 17.44	0.68 ± 259.10
10792	92992	-14.85 ± 13.14	1.21 ± 81.72	160.72 ± 17.84	0.93 ± 255.14
43168	369448	-14.83 ± 13.11	1.24 ± 81.6	161.50 ± 18.17	0.62 ± 254.40
ref	**ref**	**-14.58 ± 12.44**	**1.23 ± 80.6**	**208.83 ± 73.75**	**0.88 ± 234.2**

Table 5.15: FSI-2 with $\Delta t = 0.01$ harmonic

Cells	Dofs	$d_x(A)[\times 10^{-3}]$	$d_y(A)[\times 10^{-3}]$	Drag	Lift
2698	23563	15.42 ± 13.10	1.14 ± 83.39	157.01 ± 15.69	-0.77 ± 274.36
10792	92992	15.16 ± 12.94	1.20 ± 82.5	158.26 ± 16.03	-0.09 ± 267.81
ref	**ref**	**-14.58 ± 12.44**	**1.23 ± 80.6**	**208.83 ± 73.75**	**0.88 ± 234.2**

Table 5.16: FSI-2 with $\Delta t = 0.001$

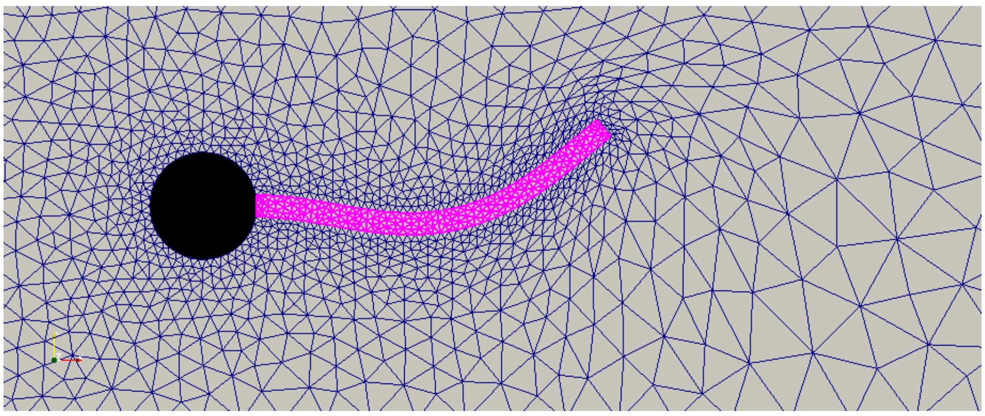


Figure 5.9: Deformation at $t = 9.20$sec. The bar is marked with pink colour and deformed using warp by vector in paraview.

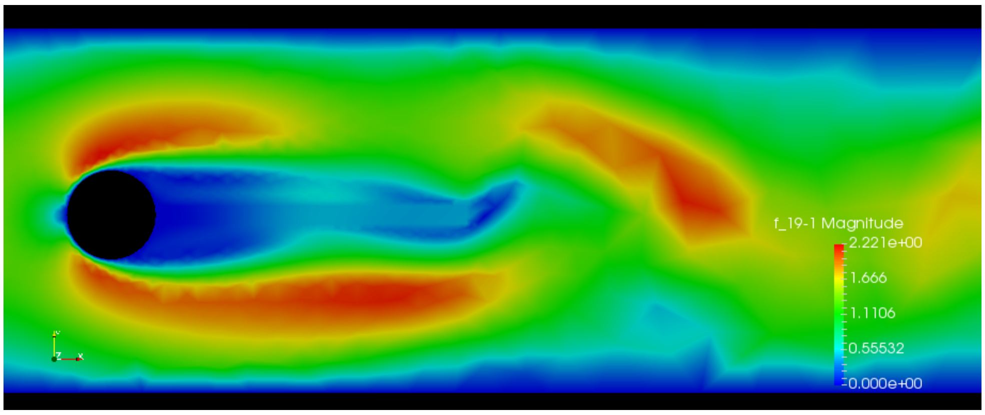


Figure 5.10: Fluid velocity at $t = 9.20$sec on the reference mesh

FSI3

		$d_x(A)[\times 10^{-3}]$	& $d_y(A)[\times 10^{-3}]$	Lift	Drag
0	bi_bc2	-1.74 ± 1.766	$3.565e \pm 26.01$	-1.357 ± 138.74	439.41 ± 12.218
	bi_bc1	-1.77 ± 1.793	3.582 ± 26.21	-1.843 ± 139.29	439.60 ± 12.218
	laplace	-1.791 ± 1.807	3.290 ± 26.48	1.960 ± 142.306	439.35 ± 12.04
1	bi_bc2	-2.397 ± 2.406	1.774 ± 32.282	3.668 ± 150.02	449.71 ± 18.172
	bi_bc1	-2.440 ± 2.446	1.766 ± 32.562	3.929 ± 150.76	449.98 ± 18.030
	laplace	-2.481 ± 2.486	1.655 ± 32.851	3.318 ± 153.58	449.74 ± 18.052
ref.		$?2.69 \pm 2.53$	1.48 ± 34.38	2.22 ± 149.78	457.3 ± 22.66

Table 5.17: FSI3 with $\Delta t = 0.01$

Comment on FSI tests

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 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.

FSI2 has a Reynolds number of 100 with medium fluid speeds. It does however induce great deformations and is therefore a great FSI test. Using a mesh motion technique that upholds the mesh structure is crucial. The results shown are with the Harmonic extension but I got similar results for biharmonic extension. The FSI2 test could be run with a fairly large time $\Delta t = 0.01$, since the fluid velocity is not very high. And it can been seen from the results that choosing a time step $\Delta t = 0.001$ did not yield much better results. The Drag was computed to be about a value of 40 less than reported by Hron and Turek. This i believe is because a combination of the way the mesh was constructed, without a good enough boundary layer, and the mapping to a reference configuration. +++

Lastly the FSI3 test has a Reynolds number of 200 with greater inflow speeds. The run time improvements such as reusing the jacobian did not work with the FSI3 test. The Jacobian needed to be more precise than for FSI2. For this reason a different mesh, with lower number of cells needed to be used in FSI3. +++

5.2.2 Comparing different lifting operators

This section is devoted to comparing different lifting operators from 3.3. The comparisions will be run using a version of the CSM test discussed earlier, with the same computing domain as the Hron Turek benchmark. The tests will compare the different techniques by looking at the how the deformation is lifted into the fluid domain. This is done by investigating a plot of the mesh after deformation to see how much cells distort and where the cells distort. This is visualized using Paraview.
In these test cases we have the fluid initially at rest and with no inflow on the fluid. 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 5.2.1 . The tests are run as time-dependent with a the backward Euler scheme, leading to a steady state solution. In the first test case the parameters from CSM1 are used, and in the CSM4 the gravitational force has just been increased from 2 to 4.

A plot has been added at the end of the FSI2 case run with different techniques to investigate the impact these have on the overall FSI problem.

Boundary conditions

The upper, lower and left boundary is set as no slip, that is no velocity in the fluid. On the left boundary there is a do nothing, and zero pressure.

Quantities for comparison

The different techniques will be plotted with the minimal value of the Jacobian. The Jacobian determinant of the deformation rate, and **if the Jacobian is zero anywhere in the domain it means that the cells overlap and can cause singularity in the matrices during assembly.**
We will also look at the a plot of the deformation in the domain, 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.

CSM1

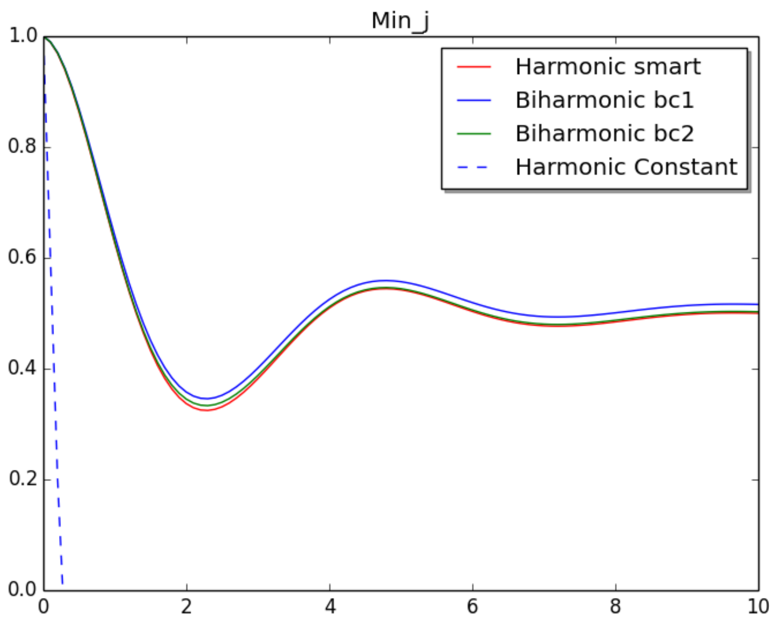


Figure 5.11: min J of the CSM1 test

Figure 5.12: Results of testing different lifting operator using the CSM1 testcase computing full FSI

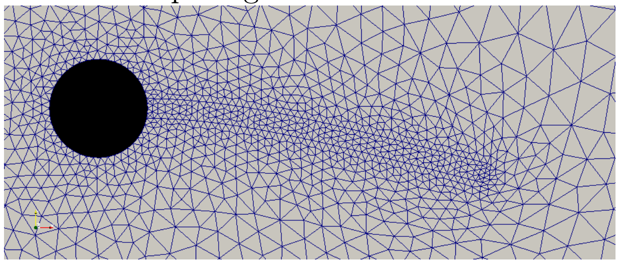


Figure 5.13: Harmonic smart

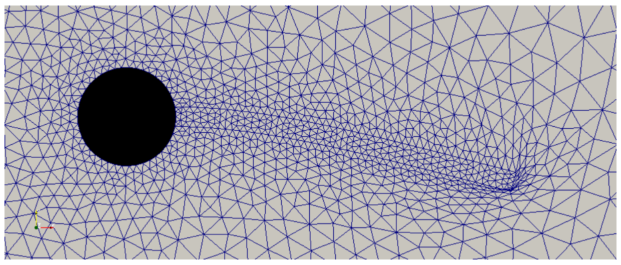


Figure 5.14: Harmonic constant

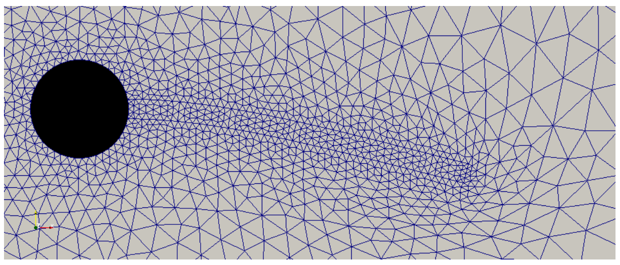


Figure 5.15: Biharmonic bc1

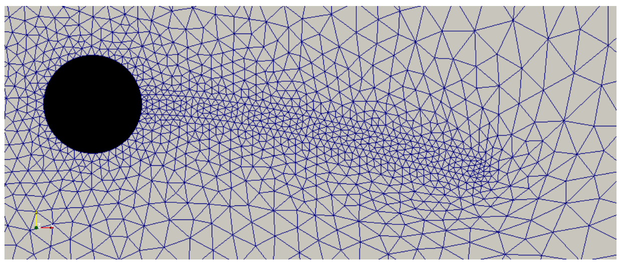


Figure 5.16: Biharmonic bc2

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

Table 5.18: Displacements results of different lifting operators of CSM1 test

Looking at figure 5.2.2 which shows the minimum of the Jacobian of the entire domain. The Harmonic with a constant α_u parameter. Gives overlapping cells quickly and computations stop. While the harmonic named smart meaning an α_u that is greater closer to the interface, and both the biharmonic techniques gives good and similar results. All three uphold, as we can see in 5.2.2, the integrity of the cells.

FSI2 with Lifting operator

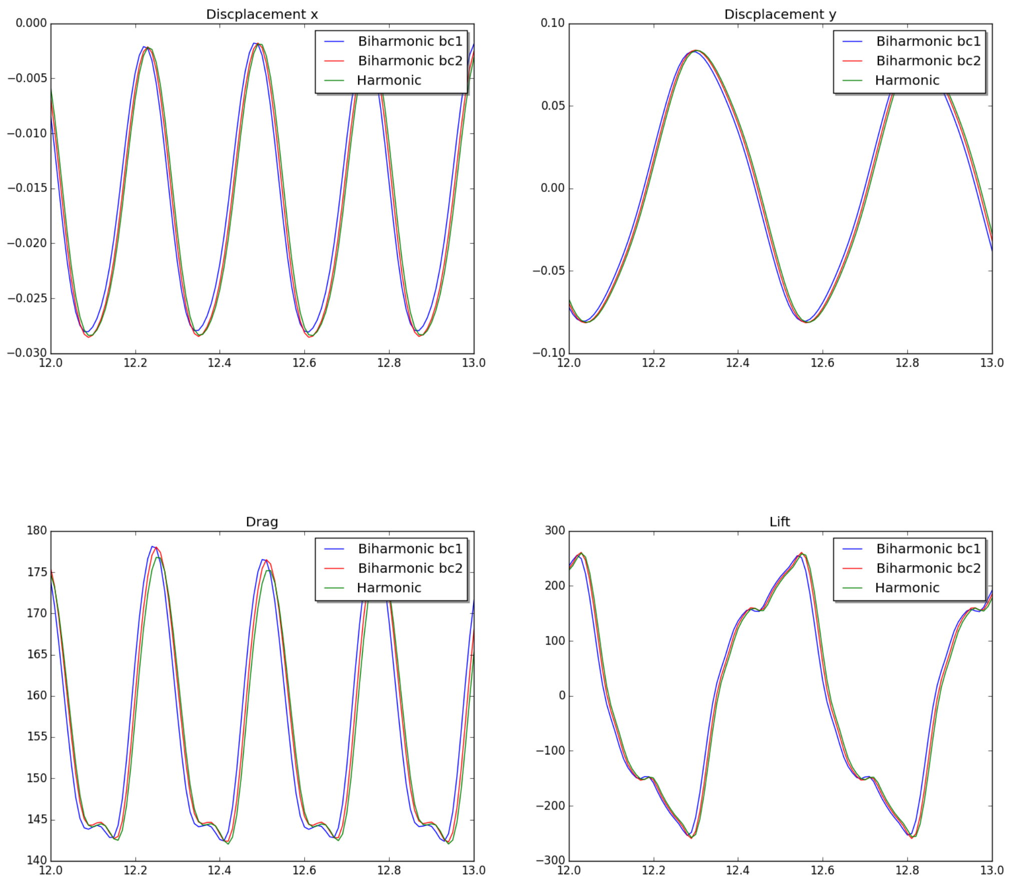


Figure 5.17: FSI2 with different lifting operators: Harmonic, Biharmonic bc1 and bc2. $\Delta t = 0.01$

Figure 5.2.2 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. This indicates that with a clever α_u the harmonic technique can be chosen. This is an advantage since the harmonic techniques is the least computationally costly.

5.2.3 Temporal stability

The following section will be looking at the results from choosing different θ values in our scheme. The benchmark test FSI-2 has been used to since it is known to blow up with certain values of θ and Δt. Only the effects of Drag as been studied as the three other quantities shows the same behavior.

Figure 5.18: Drag for FSI2 with $\Delta t = 0.01$ with different values for θ

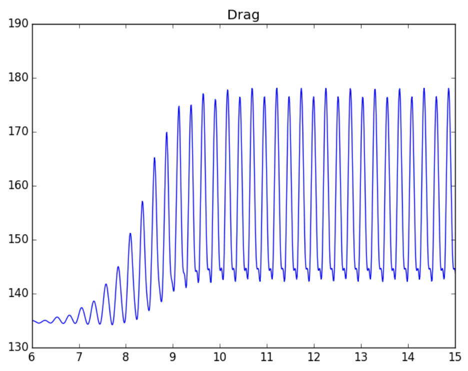


Figure 5.19: $\theta = 0.51$

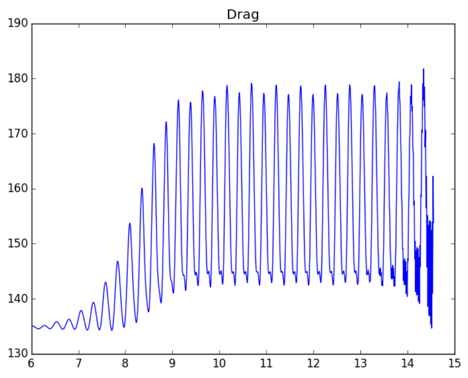


Figure 5.20: $\theta = 0.50$

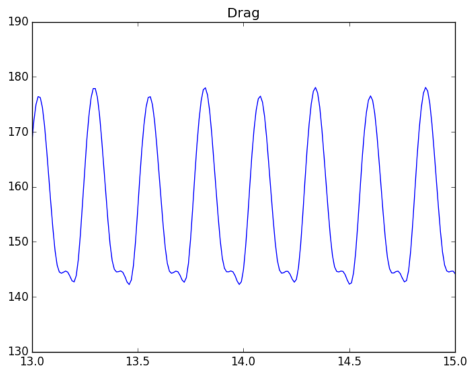


Figure 5.21: $\theta = 0.51$

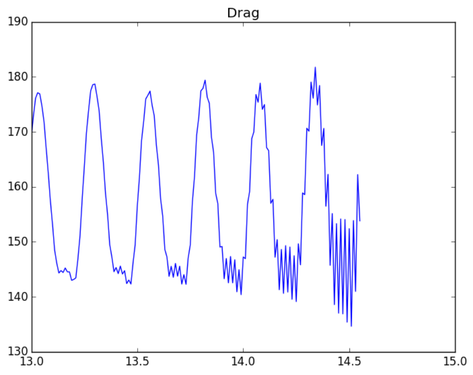


Figure 5.22: $\theta = 0.50$

5.2.3 show the plots of Drag with $\Delta t = 0.01$. It shows the instability when choosing $\theta = 0.5$. The Crank-Nicholson scheme is stable until about 13 seconds where we see that it blows up and the solver diverges. While the shifted

Crank-Nicholson, $\theta = 0.5 + \Delta t$, is stable throughout the computing time.

It has been reported in [24] that the Crank-Nicholson, $\theta = 0.5$, scheme is stable throughout the computing time by setting $\Delta t = 0.001$.

Put in plot of $\theta = 0.5$ and $\Delta t = 0.001$ to see if its stable longer? +++++++

Chapter 6

CPU time improvement techniques

Modeling Fluid Structure Interaction accurately has been difficult not only because of the lack of stable 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 strain on computers increases giving increased runtime.

To solve non-linear FSI problems I use in this these a Newton solver. 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, taking days and weeks for a simulation to finish.

I have in this chapter added the runtime improvements I have made to my newton solver. The point of this chapter is not to rigorously experiment with time improvement techniques. But merely to show what helped me make the simulation time bearable. And hopefully help others when employing a newton solver. The ideas covered are not new and has been implemented in Gabriel Balaban?s master thesis in 2012 [16].

6.1 Newton runtime profile

In table 6.1 I ran the FSI1 problem with no optimizations, checking the amount of time spent on: Jacobian Assembly, residual assembly and time to solve.

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

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

In table 6.1 it shows that the majority of time spent is in the assembly of the Jacobian. Therefore energy will be spent on reducing the time spent on assembling the Jacobian. In the next sections I will introduce 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.

6.2 Reusing the Jacobian

The majority of the time spent on in the newton solver is assembling the Jacobian. In most cases a small $\delta t = 10^-2, 10^-3$ is employed, which in turn means that the Jacobian only differs by a small amount. The trick therefore is to reuse the Jacobian. This is done by telling the solver that we only assemble the Jacobian for a given number of iterations. The rest of the newton solver stays the same and keeps iterating until convergence, but the Jacobian matrix stays the same for a set 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.

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

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

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.

6.3 Quadrature reduce

Assembling of the Jacobian matrix with non-linear functions, requires a high number of quadrature points. 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 blow up of the system in some cases. But is of great help in many other cases.

The FSI1 test is run with $\Delta t = 0.5$ like the section above.

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

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

Reducing the quadrature degree as we can see 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 %.

6.4 Summary of runtime improvement techniques

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

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

Finally I used 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 I used these two techniques as often as i could. Both techniques worked with great success in the FSI2 and FSI2 case.

Chapter 7

Conclusions and further work

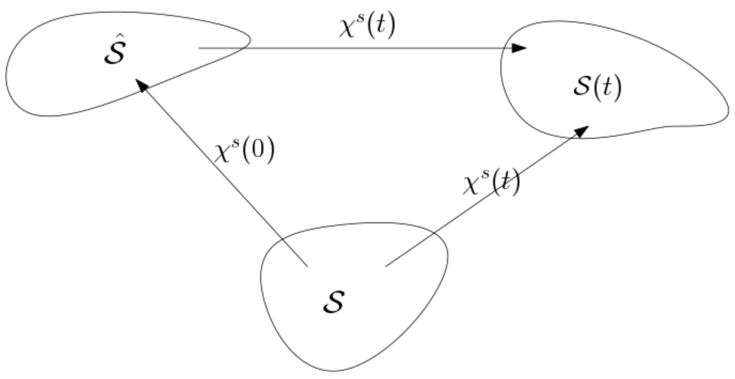
7.1 Partitioned

Appendices

Appendix A

Appendix

A.1 Lagrangian physics



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

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

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

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

which can be written in terms of the displacement field:

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

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

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

A.1.1 Deformation gradient

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

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

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

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

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

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

A.1.2 Strain

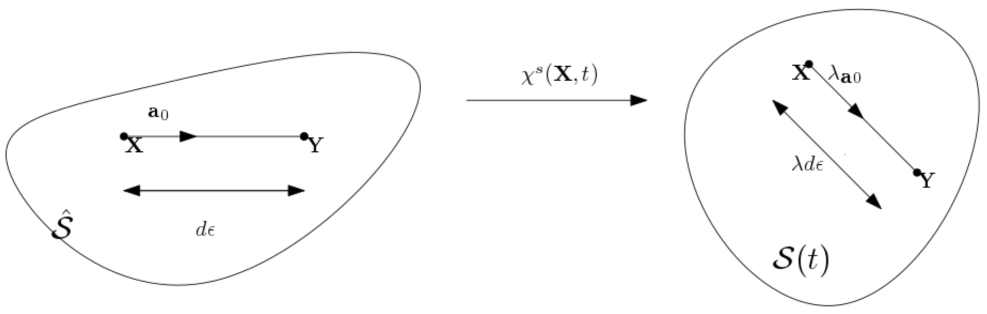


Figure A.1: Deformation of a line element with length $d\epsilon$ into a line element with length $\lambda d\epsilon$

Strain is the relative change of location between two particles. Strain, strain rate and deformation is used to describe the relative motion of particles in a continuum. This is the fundamental quality that causes stress [14].

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

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

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

$$d\mathbf{X} = d\epsilon \mathbf{a}_0 \tag{A.8}$$

$$d\epsilon = |\mathbf{Y} - \mathbf{X}| \tag{A.9}$$

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

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

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

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

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

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

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

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

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

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

Looking at the square of λ:

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

We have not introduced the important right Cauchy-Green tensor:

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

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

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

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

A.1.3 Stress

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

that completely define the state of stress at a point inside a material in the deformed state, placement, or configuration. The tensor relates a unit-length direction vector n to the stress vector T(n) across an imaginary surface perpendicular to n:

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

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

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

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

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

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

Introducing the second Piola-Kirchhoff stress tensor S:

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

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

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

Bibliography

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

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

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

[14] Thomas Richter. Fluid Structure Interactions. 2016.

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

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

[17] Noelle Selin. Verification and Validation. (February), 2014.

[18] 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.

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

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

[21] V Vinje. Simulating Cerebrospinal Fluid Flow and Spinal Cord Movement Associated with Syringomyelia. 2016.

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

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

[24] 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.

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