Thesis Title

Institution Name

Author Name

Day Month Year

Innhold

Kapittel 1

Introduction to FSI

The Fluid-Structure Interaction problem can be observed all around us in nature, from large industrial engineering complexes to the smallest blood vessels in the human body. A large scale example is the collapse of the Tacoma Narrows Bridge that collapsed in 1940 only two months after being opened. The collapse was due to aero-elastic fluttering from strong winds. No human life was lost in the collapse, but a cocker spaniel name Tubby left in a car was not so lucky. The construction of windmills are a second example of the Fluid-Structure Interaction problem. Todays windmills are rigid and hence giving a big difference in density between fluid and structure, $\rho_s >> \rho_f$. The structure will therefore only give rise to small deformations. However applying FSI to hemodynamics(dynamics of blood flow) deems more challenging. One FSI hemodynamic problem are inter-cranial aneurysms, which are balloon shaped geometries often occurring where a blood vessel splits into two parts, due to weak vessel walls. Bursting of one of these aneurysms in the skull can have fatale consequences. With fluid and structure densities more equal than the previous example, the structure has an elastic character giving under the right circumstances large deformations. The blood flow also transitions to turbulent flow. This combination gives the need for a rigid stabile solver. Therefore 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 MMS, companying a wide range of benchmarks.

Kapittel 2

Equations and notation

2.1 Notation

Let $\Omega \in \mathbb{R}^d$ for $d \in \{1, 2\}$, be a bounded domain with boundary $\partial\Omega$. The domain is made up of of two sub domains $\mathcal{F}$ for the fluid domain, and $\mathcal{S}$ for the solid. The interface between the domains are denoted by $\Sigma = \mathcal{F} \cap \mathcal{S}$. The reference or initial is denoted by $\hat{\mathbf{\Sigma}} = \hat{\mathcal{F}} \cap \hat{\mathcal{S}}$ The functions will be denoted as:
u - Velocity in fluid and structure.
d - Displacement in the solid and in the fluid mesh.
p - Pressure in the fluid.

2.1.1 Spaces

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

$$\hat{\mathbf{V}}_X := H^1(X), \qquad \hat{\mathbf{V}}_X^1 := H_0^1(X) \tag{2.1}$$

H^1 indicating a Hilbert space and

$$\hat{\mathbf{L}}_X := L^2(X), \qquad \hat{\mathbf{L}}_X^0 := L^2(X)/\mathbb{R} \tag{2.2}$$

L^2 indicating a standard Lebesque space.
The trial and test spaces for the velocity variable in the fluid domain,

$$\hat{\mathbf{V}}_{f,u}^0 := \{\hat{\mathbf{u}} \in H_0^1(\mathcal{F}) : \hat{\mathbf{u}}_f = \hat{\mathbf{u}}_s on \hat{\mathbf{\Sigma}}\} \tag{2.3}$$

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

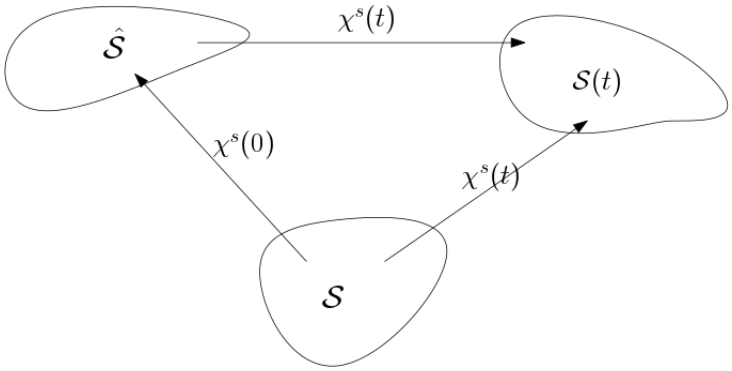
$$\hat{\mathbf{V}}_{f,d}^0 := \{\hat{\mathbf{d}} \in H_0^1(\mathcal{F}) : \hat{\mathbf{d}}_f = \hat{\mathbf{d}}_s on \hat{\mathbf{\Sigma}}\} \tag{2.4}$$

$$\hat{\mathbf{V}}_{f,d}^0 := \{\hat{\mathbf{d}} \in H_0^1(\mathcal{F}) : \hat{\psi_{\mathbf{f}}} = \hat{\psi_{\mathbf{s}}} on \hat{\mathbf{\Sigma}}\} \tag{2.5}$$

In this chapter we will look briefly into Lagrangian and Eulerian physics. The solid equation is most commonly described in a Lagrangian description. In a Lagrangian description the material particles are fixed with the grid points, this is a useful property when tracking the solid domain. The fluid equations are described in an Eulerian framework where the material points are not fixed to the grid points but move through.

2.2 Lagrangian physics

We will start by providing a short introduction to Lagrangian physics for the sake of completeness.



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

$$\chi^s(t) : \hat{\mathcal{S}} \to \mathcal{S}(t) \tag{2.6}$$

Following the notation of [4],where $\mathbf{X}$ denote a material point in the reference domain and χ^s denotes the mapping from the reference configuration. Let $d^s(\mathbf{X}, t)$ denote the displacement field and w($\mathbf{X}$,t) the domain velocity. We then set the mapping

$$\chi^s(\mathbf{X}, t) = \mathbf{X} + d^s(\mathbf{X}, t) \tag{2.7}$$

where $d^s(\mathbf{X}, t)$ represents displacement field

$$d^s(\mathbf{X}, t) = \chi^s(\mathbf{X}, t) - \mathbf{X} \tag{2.8}$$

and the domain velocity is the partial time derivative:

$$w(\mathbf{X}, t) = \frac{\partial \chi^s(\mathbf{X}, t)}{\partial t} \tag{2.9}$$

Next we will need a function that describes the rate of deformation in the solid.

2.2.1 Deformation gradient

When a continuum body undergoes deformation and is moved from the reference configuration to some current configuration we need a deformation gradient that describes the rate of deformation in the body. If $d(\mathbf{X}, t)$ is a differentiable deformation field in a given body. We define the deformation gradient as:

$$F = \frac{\partial \chi}{\partial \mathbf{X}} = I + \nabla d(\mathbf{X}, t) \tag{2.10}$$

which denotes relative change of position under deformation in a Lagrangian frame of reference. A change in volume between reference (dv) and current (dV) configuration is defined as :

$$dv = JdV \tag{2.11}$$

$$J = \det(F) \tag{2.12}$$

Where J is the determinant of the deformation gradient F known as the Jacobian determinant or volume ratio. If there is no motion, that is $F = I$ and $J = 1$, there is no change in volume. But we can also have the constraint $J = 1$ with motion, but preserving the volume. If we assume infinitesimal line and area elements in the current ds, dx and reference dV, dX configurations. The volume elements dv, dV can be expressed by the dot product:

$$dv = ds \cdot dx = JdSdX \tag{2.13}$$

This is used to get the Nansons formula:

$$ds = JF^{-T}dS \tag{2.14}$$

which holds for an arbitrary line element in different configurations.

2.2.2 Strain

In continuum mechanics relative change of location of particles is called strain and this is the fundamental quality that causes stress in a material. [10]. An import strain measure is the right Cauchy-Green tensor

$$C = F^T F$$

which is symmetric and positive definite $C = C^T$. We also introduce the Green-Lagrangian strain tensor E:

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

which is also symmetric since C and I are symmetric. This measures the squared length change under deformation.

2.2.3 Stress

Stress is the internal forces between neighboring particles. We introduce the Cauchy stress tensor:

$$\sigma_s = \frac{1}{J}F(\lambda_s(trE)I + 2\mu_s E)F^T \tag{2.15}$$

Using (2.14) we get the first Piola-Kirchhoff stress tensor P:

$$P = J\sigma F^{-T} \tag{2.16}$$

We also introduce the second Piola-Kirchhoff stress tensor S:

$$S = JF^{-1}\sigma F^{-T} = F^{-1}P = S^T \tag{2.17}$$

from this relation we can write the first Piola-Kirchhoff tensor by the second:

$$P = FS \tag{2.18}$$

2.3 Solid equation

From the principles of conservation of mass and momentum, we get the solid equation stated in the Lagrangian reference system (Following the notation from [10]):

$$\rho_s \frac{\partial d^2}{\partial t^2} = \nabla \cdot (P) + \rho_s f \tag{2.19}$$

where we used the first Piola-Kirchhoff stress tensor.

Locking

The problem og shear locking can happen FEM computations with certain elements. [mek4250 Kent] - Locking occurs if $\lambda >> \nu$ that is, the material is nearly incompressible. The reason is that all the elements discussed in this course are poor at approximating the divergence. Locking refers to the case where the displacement is to small because the divergence term essentially lock the displacement. It is a numerical artifact not a physical feature. [Verbatum]

2.4 Fluid equations

The Navier-Stokes equations are derived using principles of mass and momentum conservation. These equations describes the velocity and pressure in a given fluid continuum. They are here written in the time domain $\mathcal{F}$:

$$\rho\frac{\partial u}{\partial t} + \rho u \cdot \nabla u = \nabla \cdot \sigma_f + f \tag{2.20}$$

$$\nabla \cdot u = 0 \tag{2.21}$$

where u is the fluid velocity, p is the fluid pressure, ρ stands for constant density, f is body force and $\sigma_f = \mu_f(\nabla u + \nabla u^T) - pI$
We will only compute incompressible fluids.
There does not yet exist an analytical solutions to the N-S equations, only simplified problems can be solved [14]. But this does not stop us from discretizing and solving them numerically.
Before these equations can be solved we need to impose boundary conditions.

2.4.1 Boundary conditions

On the Dirichlet boundary $\partial\mathcal{F}_D$ we impose a given value. This can be initial conditions or set to zero as on walls with no slip"condition. These conditions needs to be defined for both u and p

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

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

The forces on the boundaries need to equal an eventual external force $\mathbf{f}$

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

Kapittel 3

Fluid Structure Interaction Problem

In FSI the computing domain is split into three parts. Fluid, structure and interface. The Navier-Stokes equations are solved on the fluid domain and the structure equation on the structure. The interface is where the solid and fluid meet, and FSI is generally solved using two methods. These two main methods differ in their treatment of the interface [6]. The first uses a fixed mesh, known as Eulerian, where the interface is tracked across the mesh. This treatment is useful for a fluid problem since fluids are continuously deformable and tracking each fluid particle is not essential in computing fluid dynamics. For the solid this formulation is not very practical as the distance between particles are used to track the deformation, hence the particles need to be tracked. Tracking where the particles are, and consequentially the interface is, across meshpoints is a difficult task. One would need short spacing between meshpoints to accurately track the interface. In the second method (ALE) the interface moves with the mesh. The ALE formulation stands for Arbitrary Lagrangian Eulerian. This entails formulating the fluid equations in an Eulerian and the solid in a Lagrangian framework. The mesh itself moves with the structure and hence interface displacements, and the fluid moves through these points. In this way we get best of both world so to speak. The structure equation will remain as previously stated (2.19), and we will need to change the fluid equations to take into account the moving mesh changing the fluid velocity. This is done in two ways. One is to move the mesh itself in relation to the structural displacements, and calculate using this new mesh every time. This approach 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 this thesis the ALE approach is used from a reference frame. Meaning we solve the equation on a initial, stress free domain, and use a series of mappings to account for the movements of the domain. The equations are mapped to the current domain, that is where the domain has moved to in the present time. It is the displacement in the solid and the displacement extrapolated from the solid to the fluid domain that determines the mappings. From a technical point of view, both formulations are equivalent.[10] But the ease of computing and time efficiency, in that we do not need an extra function to move the mesh between every timestep, contributes to the choice of using the mapped approached. Lastly in this chapter we look at how to solve the equations. This problem is tackled using either a monolithic or a partitioned approach. We will go into details later, but the overall idea is that a monolithic scheme involves computing all the equations together into one block. The partitioned splits up into parts, that is we solve the fluid problem and structure problem separately. The advantage of this is that we can use existing solvers and techniques for each of the problems, but the difficulty is the treatment of the interface. The monolithic approach however, offers more stability but is more costly as the size of the problem increases[6].

3.1 Mapping

Let $\hat{\mathcal{V}}$ be a reference volume and $\mathcal{V}(t)$ be the current time volume using [**?**]. Then using (2.10) and (2.11) 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 Balance laws

We will formulate the equations in the Eulerian, Lagrangian and the ALE description.

3.2.1 Solid

We express the solid balance laws in the Lagrangian formulation from the initial configuration

$$\rho_s \frac{\partial^2 d}{\partial t^2} = \nabla \cdot (P) \quad in \quad \hat{\mathcal{S}} \tag{3.4}$$

3.2.2 Fluid

The fluid domain is moving, and therefore we need to redefine the velocity in the convective term in (2.20) to account for the moving domain

$$\mathbf{u} \cdot \nabla \mathbf{u} \rightarrow (\mathbf{u} - \frac{\partial d_f}{\partial t}) \cdot \nabla \mathbf{u} \tag{3.5}$$

where d_f is the deformation in the fluid domain. Now the actual fluid velocity will be $u - \frac{\partial d}{\partial t}$
The fluid equations are denoted from the initial configuration using the aforementioned mappings:

$$\int_{\mathcal{V}(t)} \rho_f \frac{\partial \mathbf{u}}{\partial t} dx = \int_{\hat{\mathcal{V}}} \rho_f J \frac{\partial \mathbf{u}}{\partial t} dx \tag{3.6}$$

$$\int_{\mathcal{V}(t)} \nabla \mathbf{u} (\mathbf{u} - \frac{\partial d}{\partial t}) dx = \int_{\hat{\mathcal{V}}} ((\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial d}{\partial t}) dx \tag{3.7}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \mathbf{u} dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (JF^{-1}\mathbf{u}) dx \tag{3.8}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \sigma_f dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (JF^{-1}\hat{\sigma_{\mathbf{f}}}) dx \tag{3.9}$$

$$\hat{\sigma_{\mathbf{f}}} = -pI + \mu(\nabla \mathbf{u} F^{-1} + F^{-T}\mathbf{u}^T) \tag{3.10}$$

Putting this together (2.20) then turns into:

$$\rho_f J\big(\frac{\partial \mathbf{u}}{\partial t} dx + (\nabla \mathbf{u}) F^{-1} (\mathbf{u} - \frac{\partial d}{\partial t})\big) = \nabla \cdot (JF^{-1}\hat{\sigma_{\mathbf{f}}}) + J\rho_f f \tag{3.11}$$

$$\nabla \cdot (JF^{-1}\mathbf{u}) = 0 \tag{3.12}$$

3.3 Mesh motion

In the ALE format the deformation from the solid needs to be lifted into the deformation of the fluid mesh. The choice of mesh motion technique is important for the overall FSI problem to be calculated [16]. For small to moderate deformations we can use a harmonic extension, eg the Laplace equation:

$$-\alpha_u \nabla^2 d = 0 \quad in \;\; \hat{\mathcal{F}} \tag{3.13}$$

$$d = 0 \quad on \;\; \partial\hat{\mathcal{F}}/\hat{\mathbf{\Sigma}} \tag{3.14}$$

This equation is chosen for its good regularity and smoothing properties. A strategy for choosing α_u is proposed by Wick in [15], and further discussed in [12]. Where they set

$$\alpha_u = J^{-1} \tag{3.15}$$

This is a smart choice since J gets smaller closer to the interface, which again makes alpha large, this upholds the cell structure closer to the interface where most of the cell distortion appears. It is also possible to chose an harmonic extension with stiffening, which can give better control of the deformed meshes. This in practice behaves like a transport problem, transporting the deformation into the fluid domain. Another possibility is extension by pseudo-elasticity which defines the extension operator by means of the Navier-Lame equation. And lastly we can chose a biharmonic extension. This provides more freedom in choosing boundary conditions and choice of parameter α_u. This is because the biharmonic extension, extends the deformation in a manner that upholds the integrity of the cells even in large deformations. In its simplest form it is written as:

$$-\alpha_u \nabla^4 d = 0 \quad in \;\; \hat{\mathcal{F}} \tag{3.16}$$

The model used is 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 now 4 functions:

$$\omega = \alpha_u \nabla^2 d \quad and \quad -\alpha_u \nabla^2 \omega = 0 in \;\; \hat{\mathcal{F}} \tag{3.17}$$

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

$$d = \partial_n d = 0 \quad on \;\; \partial\hat{\mathcal{F}}/\hat{\mathbf{\Sigma}} \tag{3.18}$$

The second also imposes conditions on the function ω:

$$d = \partial_n d = 0, and \quad \omega = \partial_n \omega = 0 \quad on \;\; \partial\hat{\mathcal{F}}/\hat{\mathbf{\Sigma}} \tag{3.19}$$

Since the biharmonic extension is of fourth order it will have a higher computational cost [10]. The difference in mesh motion technique choices will be discussed later.

3.4 Coupled Fluid Structure Interface conditions

We introduce a global domain $\Omega \in \mathcal{S} \cup \mathcal{F}$ that is made up of the fluid and the structure and the interface. We define a global velocity function u that is the fluid velocity in the fluid domain and the structure velocity in the structure domain. This can be done due to the interface condition making the velocity field continuous over the entire domain. Then the interface will be $\Gamma \in \mathcal{S} \cap \mathcal{F}$ We need to define conditions that couple that motions of the fluid and structure together. These consist of:

- Kinematic condition: $\mathbf{u}_f = \mathbf{u}_s \quad on\Gamma$. The fluid and structure velocities need to be equal on the boundary.
  We later realize this later by setting $\mathbf{u} - \frac{\partial d}{\partial t} = 0$ on the solid domain to strongly imply that the velocity on the solid is the derivative of the deformation.

- Dynamic condition: $\sigma_f n_f = \sigma_s n_s \quad on \ \Gamma$.
  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:
  $J\sigma_f F^{-T} n_f = F\Sigma n_s \quad on \ \Gamma$.
  This condition is often added to structure problem, since it is most often in FSI problem the fluid exerting force that cause deformation.

- 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 continouos across the entire domain.

3.5 Monolithic FSI Problem

As stated in the introduction there are generally two types of schemes used when simulating FSI. The partitioned approach where fluid and structure are solved sequentially, is appealing in that we have a wealth of knowledge and techniques on how to solve these kinds of problems in an efficient manner. The difficulty however is dealing with the interface. As we know there are kinematic and dynamic conditions needed in FSI, and the coupling of these conditions is where the problems arise. So called explicit coupled schemes are known to be unconditionally unstable for standard Dirichlet-Neumann strategies when there is a large amount of added-mass in the system [3], [13]. There are schemes which offer added-mass free stability with explicit coupling, where the interface is treated through a Robin-Neumann coupling. First for a coupling with a thin walled structure [2] and later an extension to thick wall [3]. These schemes are 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.))
The other is approach is monolithic, where all of the equations are solved at once. This approach has the advantage of offering numerical stability for problems with strong added-mass effects [6], 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 very large and computationally costly.
We start by stating the entire FSI ALE problem in a monolithic framework using the mapped approach:
Find $\mathbf{u} \in \hat{\mathcal{F}}, p \in \hat{\mathcal{F}}$ and $d \in \hat{\mathcal{S}}$ such that :

$$\rho_f J\big(\frac{\partial u}{\partial t} + (\nabla \mathbf{u})F^{-1}(\mathbf{u} - \frac{\partial d}{\partial t})\big) + \nabla \cdot (J\hat{\sigma_\mathbf{f}} F^{-T}) = 0 \qquad \text{on } \hat{\mathcal{F}} \tag{3.20}$$

$$\nabla \cdot (J\mathbf{u}F^{-T})\big) = 0 \qquad \text{on } \hat{\mathcal{F}} \tag{3.21}$$

$$\rho_s \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot FS_s, = 0 \qquad \text{on } \hat{\mathcal{S}} \tag{3.22}$$

$$\nabla^2 d = 0 \qquad \text{on } \hat{\mathcal{F}} \tag{3.23}$$

$$\mathbf{u} - \frac{\partial d}{\partial t} = 0 \qquad \text{on } \hat{\mathcal{S}} \tag{3.24}$$

$$J\sigma_f F^{-T} n_f = \sigma_s n_s \quad \text{on } \Gamma \tag{3.25}$$

3.6 Discretization

Now that we have stated the full FSI monolithic scheme we need to specify the ways in which the scheme is discretized. The temporal discretization is done using finite difference schemes and the spatial is treated with the finite element method. Following the ideas and notations of [15]. In the domain Ω and time interval [0,T].
Find $U = \{\mathbf{u}, d, p\} \in \hat{\mathbf{X}}_D^0$ where $\hat{\mathbf{X}}_D^0 := \{\mathbf{u}_f^D + \hat{\mathbf{V}}_{f,\mathbf{u}}^0\} \times \hat{\mathbf{L}}_f \{d_f^D + \hat{\mathbf{V}}_{f,\hat{\mathbf{f}}}^0\} \times \{d_f^D + \hat{\mathbf{V}}_{f,\hat{\mathbf{f}}}^0\} \times \hat{\mathbf{L}}_f^0 \times \hat{\mathbf{L}}_s^0$ such that:

$$\int_0^T A(U)(\Psi)dt = \int_0^T \hat{\mathbf{F}}(\Psi)dt \quad \forall \Psi \in \hat{\mathbf{X}} \tag{3.26}$$

where $\Psi = \{\hat{\psi}_f^{\mathbf{u}}, \hat{\psi}_s^{\mathbf{u}}, \hat{\psi}_f^d, \hat{\psi}_s^d, \hat{\psi}_f^p, \hat{\psi}_f^p\}$ and
$\hat{\mathbf{X}} = \hat{\mathbf{V}}^0_{f,\mathbf{u}} \times \hat{\mathbf{L}}_f \times \hat{\mathbf{V}}^0_{f,d,\hat{\mathbf{\Sigma}}} \times \hat{\mathbf{V}}^0_s \times \hat{\mathbf{L}}^0_f \times \hat{\mathbf{L}}^0_s$
We use for simplicity the harmonic mesh motion and we use a global function for $\mathbf{u}$ 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:

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

The condition (3.31) , 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. In [**?**] they have ρ_s in front of the condition which has no physical meaning, but will act as a weight. ρ_s is 7800 for steel and 1400 PVC [5] so these could work as weights. While in our program we set it to a large value $\delta = 10^{10}$. Whilst in [16], where the same scheme is used there is no weighting.
I will here formulate the *One step-θ scheme* from [15]. This θ scheme has the advantage of easily being changed from the backward (implicit), forward(excplicit) or Crank-Nicholson (implicit) scheme. The backward Euler scheme is of first order and is implicit in that it is using the newest time step appears on both sides of the equation. The Crank-Nicholson is of second order and both the current and previous time step is used. This scheme suffers from instabilities in its normal context, we will therefore look at a *shifted* Crank-Nicholson scheme.
We define the variational form by dividing into four categories: time term, implicit, pressure and the rest (stress, convection):

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

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

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

We also notice that the we have split up
For the time group we discretize in the following way:

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

And the Jacobian is written with superscript θ as:

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

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

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

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

We see here that 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 shifted Crank-Nicholson we set $\theta = \frac{1}{2} + k$, effectively shifting the scheme towards the implicit side. $\hat{\mathbf{f}}$ is the body forces which will be ignored in this thesis.

3.6.1 Spaces and Elements

The velocity and pressure copling in the fluid domain must satisfy the inf-sup condition. If not stabilization has to added. We here need to define some spaces that will have these desired properties. We denote $u_h \in V_h$ and $d_h \in W_h$, here the finite element pair og pressure and velocity mush satisfy the inf-sup condition given in ALE coordinates:

$$\inf_{p_h \in L_{h,f}} \sup_{v_h \in V_{h,f}} \frac{(p_h, div(J_f F_f^{-1} u_h))_{\mathcal{F}}}{\|\|J^{\frac{1}{2}} p_h\|\|_{\mathcal{F}} \|\|J_f^{\frac{1}{2}} \nabla u_h F_f^{-T}\|\|_{\mathcal{F}}} \geq \gamma$$

A good choice of spaces will be P2-P2-P1 for velocity, displacement and fluid pressure respectively.

Kapittel 4

FSI Implementation in FEniCS

Here we will look at the implementation of the monolithic FSI Code in FENiCS. Not every part of the code will be handled here, but hopefully enough so that someone familiar with FEniCS could in short time implement the scheme.

4.1 Mesh, mappings and stress tensors

The mesh was made with Gmesh, with a clear straight boundary splitting the fluid and solid domains. This is converted to a xml file and loaded into FEniCS.

```
mesh_file = Mesh("Mesh/fluid_new.xml")
```

The boundaries are defined using built in domain functions. We only here show the inlet where we specified where the spatial points are, and give it a mark value to be used when the Dirichlet conditions are set. This is also done using CellFunctions to define the fluid and structure domain.

```
Inlet = AutoSubDomain(lambda x: "on_boundary" and near(x[0],0))
boundaries = FacetFunction("size_t",mesh_file)
Inlet.mark(boundaries, 3)
```

All the mappings are made with python functions, this is done so we can call the mappings later in the variational form.

```
def F_(d):
        return (Identity(len(d)) + grad(d))

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

The stress tensors are also defined as functions to be used in the variational form.

```
def E(d):
        return 0.5*(F_(d).T*F_(d) - Identity(len(d)))

def S(d,lamda_s,mu_s):
    I = Identity(len(d))
    return 2*mu_s*E(d) + lamda_s*tr(E(d))*I

def Piola1(d,lamda_s,mu_s):
        return F_(d)*S(d,lamda_s,mu_s)

def sigma_f_u(u,d,mu_f):
    return  mu_f*(grad(u)*inv(F_(d)) + inv(F_(d)).T*grad(u).T)

def sigma_f_p(p, u):
    return -p*Identity(len(u))
```

4.2 Variational form

The variational form can be written directly into FEniCS. We write all the forms and add them together to make one big form to be calculated in the upcoming timeloop. We start by looking at the fluid variational form

```
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
```

where dx_f is the fluid domain. The theta value choses the scheme we want. inv() gives the inverse of the matrix.
Next is the solid variational form:

```
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 3.6

4.3 NewtonSolver

To solve a non-linear problem we need make a newton solver, taken from [Mikael kompendium]. F is derivated wrt to *dvp* and is assembled to a matrix. -F is assembled as b and we solve until the residual is smaller than a give tolerance. There is also an if test which only assembles the Jacobian the first and tenth time. This reuses the Jacobian to improve speed, as we shall see later. Lastly the the mpi line is when the code is running in parallell that we only print out the values for one of the computational nodes.

```
    Iter      = 0
    residual  = 1
    rel_res   = residual
    chi = TrialFunction(DVP)
    Jac = derivative(F, dvp_["n"], chi)

    while rel_res > rtol and residual > atol and Iter < max_it:
```

```
        if Iter == 0 or Iter == 10:
            A = assemble(Jac, tensor=A)#, keep_diagonal = True)
            A.ident_zeros()

        b = assemble(-F)

        [bc.apply(A, b, dvp_["n"].vector()) for bc in bcs]
        solve(A, dvp_res.vector(), b)


        dvp_["n"].vector()[:] = dvp_["n"].vector()[:] + lmbda*dvp_res.vector()[:]
        [bc.apply(dvp_["n"].vector()) for bc in bcs]

        rel_res = norm(dvp_res, 'l2')
        residual = b.norm('l2')

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

4.4 Timeloop

In the time loop we call on the solver and update the functions v, d, p for each round. The counter value is used when we only want to take out values every certain number of time iterations.

```
while t <= T:
    print "Time t = %.5f" % t
    time_list.append(t)
    if t < 2:
        inlet.t = t;
    if t >= 2:
        inlet.t = 2;

    #Reset counters
    atol = 1e-6;rtol = 1e-6; max_it = 100; lmbda = 1.0;

    dvp = Newton_manual(F, dvp, bcs, atol, rtol, max_it, lmbda,dvp_res,VVQ)


    times = ["n-2", "n-1", "n"]
    for i, t_tmp in enumerate(times[:-1]):
        dvp_[t_tmp].vector().zero()
        dvp_[t_tmp].vector().axpy(1, dvp_[times[i+1]].vector())

    t += dt
    counter +=1
```

Kapittel 5

Verification and validation.

When we set out to solve a real world problem with numerical computing, we start by defining the mathematics, we implement the equations numerically and solve them on a computer. We then use the solutions to extract data that will answer the questions we set out to solve. A problem then immediately arises, is this solution correct? To answer this we need to answer another question, are the equations solved correct numerically, if so is the problem defined correct mathematically in accordance with the governing laws and equations? Without answering these questions, being confident that your solutions are correct is difficult [11]. The goal of this section will hence be to verify and validate the different numerical schemes.
We start with Verification, which is the process of assessing numerical correctness and accuracy of a computed solution. Then comes Validation, which is assessing physical accuracy of the numerical model, a process which is done by comparing numerical simulation with experimental data. In simple terms we check that we are solving the equations right and then that we are solving the right equations. The process of Verification has to always come before Validation. Because there is no need in checking if we are using the right equations if the equations are not solved right.

Verification

In verification we get evidence that the numerical model derived from mathematics is solved correctly by the computer. The strategy will be to identify, quantify and reduce errors cause by mapping a mathematical model to a computational model. This does not address wether or not the mathematical model is in alignment with the real world only that our model is computed correctly. 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 solution to for instance the Navier-Stokes equations, other than for very simplified problems. In tackling these problems there are multiple classes of test that can be performed, and the most rigorous is the *Method of manufactured solution* [9]. 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 procedure is as follows [9]:

- We define a mathematical model on the form $L(u) = 0$ where $L(u)$ 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 u, $L(u) = f$

After the solution has been computed we perform systematic convergence tests [**?**]. The idea of order of convergence test is based on the behavior of the error between the manufactured exact solution and the computed solution. When we increase the number of spatial points ($\Delta x, \Delta y$ or Δz) or decrease timestep(Δt), we expect the error to get smaller. Its the rate of this error that lets us now wether the solution is converging correctly. If we let u be the numerical solution and u_e be the exact solution, $||.||$ be the L^2 norm, we define the error as:

$$E = ||u - u_e|| \tag{5.1}$$

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. If we for instance reduce Δt significantly than Δx will dominate, and Δt is negligible . If we then look at the errors in two timesteps, using (5.2):

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

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

We can use this to find the observed order of convergence and match with the theoretical for given

5.1 Structure MMS

We also want to test the coupled solid solver. We make a sourceterm f_s:

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

Solid variational formulation:

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

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

We should make another sourceterm f for the second equation but the solutions will be made so this line becomes zero. Using the solutions

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

and doing the order of convergence test we get, first in space:
Time:

5.2 MMS on FSI ALE

In this section we use the method of manufactured solutions to verify the FSI ALE monolithic solver. We start by prescribing a motion to d and w and give a solution to u and p. We set $u = w$ to start with:

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

Tabell 5.1: Structure MMS

N	Δt	m degree	E_u	k_u	E_d	k_d
4	$1x10^{-6}$	1	0.00688260782038		3.7854343057e-08	
8	$1x10^{-6}$	1	0.00172039793734	2.00021299904	9.46218870811e-09	2.00021299271
16	$1x10^{-6}$	1	0.000430084236596	2.00005114684	2.36546335807e-09	2.00005112024
32	$1x10^{-6}$	1	0.000107520101545	2.00001284898	5.91360617339e-10	2.00001274007
64	$1x10^{-6}$	1	2.68799509236e-05	2.00000399654	1.47839789951e-10	2.00000355583
4	$1x10^{-6}$	2	6.60233871884e-05	-	3.63128629891e-10	-
8	$1x10^{-6}$	2	8.28397328621e-06	2.9945823485	4.55618532822e-11	2.99458234332
16	$1x10^{-6}$	2	1.03646090432e-06	2.99865720273	5.70053508884e-12	2.99865718019
32	$1x10^{-6}$	2	1.2958771861e-07	2.99966479692	7.12732513531e-13	2.99966470221
64	$1x10^{-6}$	2	1.61994158345e-08	2.99991530218	8.90968200767e-14	2.99991489187

Tabell 5.2: Structure MMS Time

N	Δt	E_u	k_u	E_u	k_d
64	0.0008	2.40113737032e-06	-	1.76531251763e-08	-
64	0.0004	1.20432501777e-06	0.995493150627	8.68459764556e-09	1.02339269394
64	0.0002	5.91307436945e-07	1.02624446535	4.14332593927e-09	1.06766969495
64	0.0001	2.93267031994e-07	1.01169352445	2.02357953875e-09	1.03387975932
64	0.00005	1.46821750169e-07	0.998149192911	1.00209817454e-09	1.01388570182

We make the solutions to uphold the criterias : $\nabla \cdot u = 0$ and $\frac{\partial d}{\partial t} = w$

To test the mapping we make the source term f without mappings:

$$\rho_f \frac{\partial u}{\partial t} + \nabla u(u - \frac{\partial d}{\partial t}) - \nabla \cdot \sigma_f = f$$

Then we use this f and map it to the reference configuration and compute:

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

The computations are done on a unitsquare 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.

Tabell 5.3: MMS ALE FSI u=w

N	Δt	m	E_u	k_u	E_p	k_p
64	0.1	2	0.0140496662424	-	4.78779559903	-
64	0.05	2	0.00697215098985	1.01086014072	2.38002096658	1.00838727906
64	0.025	2	0.00341287458821	1.03061641184	1.18981484439	1.00023719999
64	0.0125	2	0.00164214907307	1.05540230133	0.595733372533	0.99799839775
2	$10x^{-6}$	2	0.000520027806571	-	0.0194221106771	-
4	$10x^{-6}$	2	6.60205272446e-05	2.97760220293	0.00480815191132	2.01414560945
8	$10x^{-6}$	2	8.28184559099e-06	2.99489045	0.00118568799584	2.0197580517
16	$10x^{-6}$	2	1.0417232845e-06	2.99098020306	0.000281586546806	2.0740741124

Validation

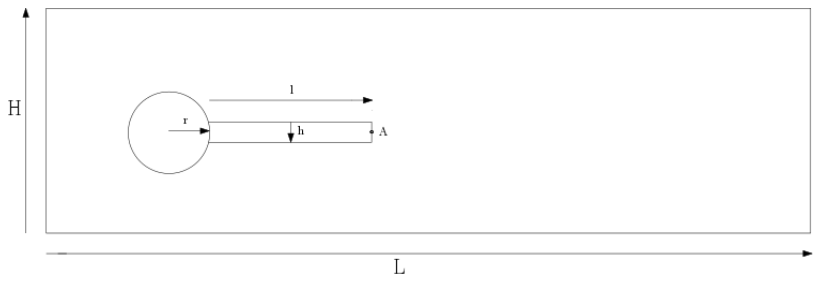
After the code has been verified to see that we are indeed computing in the right fashion. We move on to Validation which is the process of determining if the model gives an accurate representation of the real world within the bounds of the intended use [11]. A model is made for a specific purpose, its only valid in respect to that purpose [7]. If the purpose is complex and trying to answer multiple question then the validity need to be determined to each question. The idea is to validate the solver *brick by brick*. We start with simple testing of each part of the model and build more complexity and eventually testing the whole model.Three issues have been identified in this process [11]: 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. For example if our solver needs to model fluid which is turbulent we have to validate our model to catch these turbulences and as we shall see later the Taylor-Green benchmark is a good test. Well known benchmarks will be used as validation, we will see in this chapter that 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 to run. As we will see in the chapter all the benchmarks are run with different timesteps and number of cells to see how it 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 [11] 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 now the bounds of sufficient accuracy. [11]
In the following we will look at tests for the fluid solvers both alone, testing laminar to turbulent flow, and with solid. We will test the solid solver, and lastly the entire coupled FSI problem.

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

The goal of this benchmark is to test the fluid and solid solver first separately and then together as a full FSI problem [5]. This benchmark is based on the older benchmark flow around cylinderẅith fluid considered incompressible and in the laminar regime, and the structure deformations are significant. The problem is setup with the solid submerged in the fluid, so that oscillations in the fluid deform the structure. We will measure the drag and lift around the circle and bar, and measure structural displacement at a given point.

5.3.1 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 is done to induce oscillations to an otherwise laminar flow. This 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

The fluid velocity has a parabolic profile on the inlet that changes over time:

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

We set no slip on the floor"and "ceilingso to speak.

$$u(x,y,t) = 0 \text{ on}$$

On the fluid solid interface the boundary conditions are set to:

$$\sigma_f n_f = \sigma_s n_s \quad on \ \ \Gamma^0(interface)$$

In our variational form we leave this out and so implying that they are equal.

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 on the right side of the bar. This point is used to track the deformation in CSM and FSI tests.
In each test the numbers with ref are the values taken from the benchmark paper [5] We integrate the mapped fluid stress tensor over the bar and circle and appended to lists:

```
Dr = -assemble((sigma_f_new(v,p,d,mu_f)*n)[0]*ds(6))
Li = -assemble((sigma_f_new(v,p,d,mu_f)*n)[1]*ds(6))
Dr += -assemble((sigma_f_new(v("-"),p("-"),d("-"),mu_f)*n("-"))[0]*dS(5))
Li += -assemble((sigma_f_new(v("-"),p("-"),d("-"),mu_f)*n("-"))[1]*dS(5))
Drag_list.append(Dr)
Lift_list.append(Li)
```

The deformation is calculated on the point A, and also added to lists:

```
dsx = d(coord)[0]
dsy = d(coord)[1]
dis_x.append(dsx)
dis_y.append(dsy)
```

5.3.2 Results

CFD test

The first two CFD tests are run with Reynolds number 20 and 100 giving steady drag and lift around the circle. CFD 3 has a Reynolds number 200 which will induce oscillations behind the circle, giving fluctuations in the drag and lift. The CFD tests were run using the the bar as rigid object, that is the domain calculated is just the fluid domain. It is possible to also calculate with the bar and setting ρ_s and μ_s to a large value.

Tabell 5.4: CFD parameters

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

Tabell 5.5: CFD 1

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**

Tabell 5.6: CFD 2

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**

CSM test

The CSM test are calculated using only the bare and adding a gravity term g with the same value but changing the parameters of solid. As with the CFD test the first to CSM test cause a steady state solution, and CSM 3 is more slender causing the bar to go up and down in time. Our quantity for comparing there will be the deformation of the point A. In CSM 3 the energy is conserved by using a Crank-Nicholson scheme as can be seen in the plots fig7 (hvordan citer man et plot?)

Tabell 5.7: Parameters

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

Tabell 5.8: CSM 1

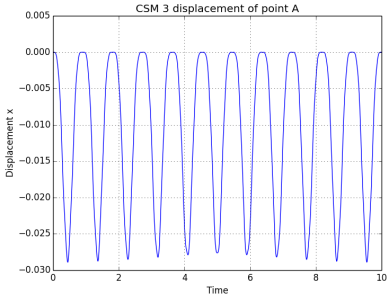
elements	dofs	ux $[10^{?3}]$	uy $[10^{?3}]$
725	1756	-5.80951654915	-59.4781430115
2900	6408	-6.77960453995	-64.2130757639
11600	24412	-7.08597041285	-65.635825349
46400	95220	-7.11626976966	-65.7456687273
ref	ref	-7.187	-66.10

Tabell 5.9: CSM 2

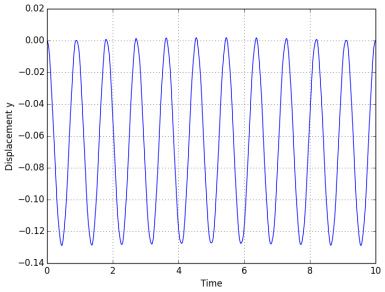
Elements	Dofs	ux $[10^{-3}]$	ux $[10^{-3}]$
725	1756	-0.375962146908	-15.1950342598
2900	6408	-0.441308781709	-16.4643196042
11600	24412	-0.462087305294	-16.8478689583
46400	95220	-0.464128022327	-16.8782135872
ref	ref	-0.4690	-16.97

Tabell 5.10: CSM 3

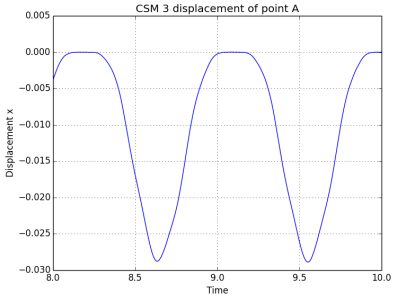
elements	dofs	ux $[10^3]$	uy $[10^3]$
725	1756	-11.743 ± 11.744	-57.952 ± 58.940
2900	6408	-13.558 ± 13.559	-61.968 ± 63.440
11600	24412	-14.128 ± 14.127	-63.216 ± 64.744
46400	95220	-14.182 ± 14.181	-63.305 ± 64.843
ref		-14.305 ± 14.305	-63.607 ± 65.160



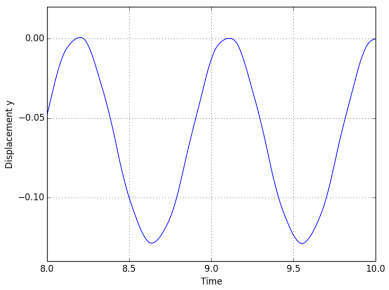
(a) Displacement in x-direction



(b) Displacement in x-direction



(c) Displacement in x-direction



(d) Displacement in x-direction

Figur 5.1: Displacement of point A

FSI test

Results:

5.4 Mesh motion techniques

5.5 Temporal stability

Tabell 5.11: FSI Parameters

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
ν_s	0.4	0.4	0.4
$\mu_s[10^6 \frac{m^2}{s}]$	0.5	0.5	2

Tabell 5.12: FSI 1

Cells	Dofs	ux of A $[x10^{-3}]$	uy of A $[x10^{-3}]$	Drag	Lift	Spaces
2698	7095	0.0234594	0.797218	14.4963	0.915801	P1-P1-P1 stab= 0.01
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	00.227566	0.813184	14.2269	0.771071	P2-P2-P1
ref	**ref**	**0.0227**	**0.8209**	**14.295**	**0.7638**	**ref**

Kapittel 6

Speed improvements

6.1 Jacobian reuse

When solving a monolithic FSI problem, the size of the solution matrix can quickly become quite large. To solve non-linear FSI problems we as discussed before use a Newton solver. The majority of the time spent on in the newton solver is assembling the Jacobian. In most cases we employ a small $\delta t = 10^-2, 10^-3$, 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.
RESULTS: FSI3 does not work with jacobian reuse or quadreduce.
FSI2 works perfectly with reuse and quad, going pretty fast.

Kapittel 7

Conclusions and further work

7.1 Partitioned

Bibliografi

[1] James DeBonis. Solutions of the Taylor-Green Vortex Problem Using High-Resolution Explicit Finite Difference Methods. *51$ˆ{st}$ AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, (February):1–9, 2013.

[2] 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.

[3] 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.

[4] G Holzapfel. Nonlinear solid mechanics: A continuum approach for engineering, 2000.

[5] 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.

[6] 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.

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

[8] Mikael Mortensen and Kristian Valen-Sendstad. Oasis: A high-level/high-performance open source Navier–Stokes solver. *Computer Physics Communications*, 188:177–188, 2015.

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

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

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

[12] 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.

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

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

[15] 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.

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