Thesis Title

Institution Name

Author Name

Day Month Year

Innhold

Kapittel 1

Continuum Mechanics

When studying the dynamics of a mediums with fluid or structure properties under the influence of forces, we need in some sense a good description of how these forces act and alter the system itself.
Any medium on a microscopic scale is built up of a structure of atoms, meaning we can observe empty spaces between each atom or discontinuities in the medium. Discribing any phsycial phenomen on larger scales in such a way are tedious and most often out of bounds due to the high number of particles. Instead we consider the medium to be continously distributed throughout the entire reagion it occupies. Hence we want to study some phsyical properties of the complete volume and not down on atomic scale.
We consider the medium with continuum properties. By a continuum we mean a volume $V(t) \subset \mathbb{R}^3$ consiting of particles, which we observe for some properties. One property of interest could be the veloctity $\mathbf{v}(x,t)$ for some point $x \in V(t)$ in time $t \in (0,T]$, which would mean the average velocity of the particles occupying this point x at time t The intension of this chapter is not to give a thorough introduction of continuum mechanics, but rather present key concepts needed for the evaluation of fluid-strucutre interaction.

1.1 Coordinate system

We assume that our medium is continiously distributed throughout its own volume, and we start our observation of this medium at som time t_0. As this choice is arbitary, we often choose to observe a medium in a stress free initial state. We call this state $V(t_0)$ of the medium as the *reference configuration*. We let $V(t)$ for $t \geq t_0$ denote the *current configuration*.

Central for the coordinate systems introduced in this chapter is the concept of *material* and *spatial* points. *Material* points are simply the points defining the material, moving with it as it undergoes movement. *Spatial* points on the other hand is the relative measure of movement of the *material* points. (Godt nok ??). This concept will be further explained throughout the chapter.

1.1.1 Lagrangian

As some medium is act upon by forces, one of the main properties of interest is the deformation of the medium. Hence we want to know the relative position of some particle from its initial configuration.
Let $\hat{\mathrm{x}}$ be a particle in the reference $\hat{\mathrm{x}} \in \hat{\mathrm{V}}$. Further let $\mathrm{x}(\hat{\mathrm{x}}, \mathrm{t})$ be the new location of a particle $\hat{\mathrm{x}}$ for some time t such that $x \in V(t)$. We assume that no two particles $\hat{\mathrm{x}}_a, \hat{\mathrm{x}}_b \in \hat{\mathrm{V}}$ occupy the same location for some time $V(t)$. Then the transformation $\hat{\mathrm{T}}(\hat{\mathrm{x}}, t) = x(\hat{\mathrm{x}}, t)$ maps a particle $\hat{\mathrm{x}}$ from the *reference configuration* $\hat{\mathrm{V}}$ to the *current configuration* $V(t)$ Assuming that the path for some $\hat{\mathrm{x}}$ is continuous in time, we can define the inverse mapping $\hat{\mathrm{T}}^{-1}(x,t) = \hat{\mathrm{x}}(x,t)$, which maps $x(\hat{\mathrm{x}}, t)$ back to its initial location at time $t = t_0$.
These mappings lets us track each particle from some *reference configuration* to some deformed

state at time t. Such a description of tracking each particle $\hat{\mathrm{x}} \in \hat{\mathrm{V}}$ is often denoted the *Lagrangian Framework* and is a natural choice of describing structure mechanics.
We define the *deformation*

$$\hat{\mathrm{T}}(\hat{\mathrm{x}}, t) = \hat{\mathbf{u}}(\hat{\mathrm{x}}, t) = x(\hat{\mathrm{x}}, t) - \hat{\mathrm{x}} \tag{1.1}$$

and the *deformation velocity*

$$\frac{\partial \hat{\mathrm{T}}(\hat{\mathrm{x}}, t)}{\partial t} = \hat{\mathbf{v}}(\hat{\mathrm{x}}, t) = d_t x(\hat{\mathrm{x}}, t) = d_t \hat{\mathbf{u}}(\hat{\mathrm{x}}, t) \tag{1.2}$$

When tracking each particle as it moves, the *material* and *spatial* points coincide

1.1.2 Eulerian

Considering a flow of fluid particles in a river, a *Lagrangian* description of the particles would be tidious as the number of particles entring and leaving the domain quickly rise to a immense number. Instead consider defining a view-point V fixed in time, and monitor every fluid particle passing the coordinate $x \in V(t)$ as time elapses. Such a description is defined as the *Eulerian framework*. Therefore the Eulerian formulation is natural for describing fluid dynamics.
We can describe the particles occupying the *current configuration* $V(t)$ for some time $t \geq t_0$

$$x = \hat{\mathrm{x}} + \hat{\mathbf{u}}(\hat{\mathrm{x}}, t)$$

Since our domain is fixed we can define the deformation for a particle occupying position $x = x(\hat{\mathrm{x}}, t)$ as

$$\mathbf{u}(x, t) = \hat{\mathbf{u}}(\hat{\mathrm{x}}, t) = x - \hat{\mathrm{x}}$$

and its velocity

$$\mathbf{v}(x, t) = \partial_t u(x, t) = \partial_t \hat{\mathbf{u}}(\hat{\mathrm{x}}, t) = \hat{\mathbf{v}}(\hat{\mathrm{x}}, t)$$

It is important to mention that the we are not interested in which particle is occupying a certain point in our domain, but only its properties. As such the *material* and *spatial* points doesn't coincide in the *Eulerian formulation*

1.2 Deformation gradients

When studying continuum mechanics we observe continious mediums as they are deformed over time. These deformations results in relative changes of positions due to external and internal forces acting.. These relative changes of postition is called *strain*, and is the primary property that causes *stress* within a medium of interest [12]. We define stress as the internal forces that particles within a continuous material exert on each other.

The equations of mechanics can be derived with respect to either a deformed or undeformend configuration of our medium of interest. The choice of refering our equations to the current or reference configuration is indifferent from a theoretical point of view. In practice however this choice can have a severe impact on our strategy of solution methods and physical of modelling. [**?**]. We will therefore define the strain measures for both configurations of our medium.

Definition 1.1. Deformation gradient.

$$\hat{\mathbf{F}} = I + \hat{\nabla}\hat{\mathbf{u}} \tag{1.3}$$

Mind that deformation gradient of $\hat{\mathbf{u}}$ is which respect to the reference configuration. From the assumption that no two particles $\hat{\mathrm{x}}_a, \hat{\mathrm{x}}_b \in \hat{\mathrm{V}}$ occupy the same location for some time $V(t)$, the presented transformation must be linear. As a consequence from the invertible matrix theorem found in linear algebra, the linear operator $\mathbf{F}$ cannot be a singuar. We define the *determinant of the deformation gradient* as J, which denotes the local change of volume of our domain.

Definition 1.2. Determinant of the deformation gradient

$$J = det(\hat{\mathbf{F}}) = det(I + \hat{\nabla}\hat{\mathbf{u}}) \neq 0 \tag{1.4}$$

By the assumption that the medium can't be selfpenetrated, we must limit J to be greater than 0 [**?**]

1.3 Measures of Strain and Stress

The equations describing forces on our domain can be derived in accordinance with the current or reference configuration. With this in mind, different measures of strain can be derived with respect to which configuration we are interested in. We will here by [12] show the most common measures of strain. We will first introduce the right *Cauchy-Green* tensor **C**, which is one of the most used strain measures [**?**].
Uttrykk 1.3 fra Godboka, LAG TEGNING

Let $\hat{x}, \hat{y} \in \hat{V}$ be two points in our referemce configuration and let $\hat{a} = \hat{y} - \hat{x}$ denote the length of the line bewtween these two points. As our domain undergoes deformation let $x = \hat{x} + \hat{\mathbf{u}}(\hat{x})$ and $x = \hat{y} + \hat{\mathbf{u}}(\hat{y})$ be the position of our points in the current configuration, and let $a = y - x$ be our new line segment. By [12] we have by first order Taylor expansion

$$\begin{aligned} y - x = \hat{y} + \hat{\mathbf{u}}(\hat{y}) - \hat{x} - \hat{\mathbf{u}}(\hat{x}) = \ \hat{y} - \hat{x} + \hat{\nabla}\hat{u}(\hat{x})(\hat{y} - \hat{x}) + \mathcal{O}(|\hat{y} - \hat{x}|^2) \\ \frac{y - x}{|\hat{y} - \hat{x}|} = [I + \hat{\nabla}\hat{\mathbf{u}}(\hat{x}]\frac{\hat{y} - \hat{x}}{|\hat{y} - \hat{x}|} + \mathcal{O}(|\hat{y} - \hat{x}|) \end{aligned}$$

This detour from [12] we have that

$$\begin{aligned} a = y - x = \hat{\mathbf{F}}(\hat{x})\hat{a} + \mathcal{O}(|\hat{a}|^2) \\ |a| = \sqrt{(\hat{\mathbf{F}}\hat{a}, \hat{\mathbf{F}}\hat{a}) + \mathcal{O}(|\hat{a}^3|)} = \sqrt{(\hat{a}^T, \hat{\mathbf{F}}^T\hat{\mathbf{F}}\hat{a})} + \mathcal{O}(|\hat{a}^2|) \end{aligned}$$

We let $\hat{C} = \hat{F}^T\hat{F}$ denote the right *Cauchy-Green tensor*. By observation the Cauchy-Green tensor is not zero at the reference configuration

$$\hat{C} = \hat{F}^T\hat{F} = (I + \hat{\nabla}\hat{u})^T(I + \hat{\nabla}\hat{u}) = 1$$

Hence it is convenient to introduce a tensor which is zero at the reference configuration. We define the *Grenn-Lagrange strain tensor*, which arises from the squard rate of change of the linesegment $\hat{a}$ and a. By using the definition of the Cauchy-Green tensor we have the relation

$$\begin{aligned} \frac{1}{2}(|a|^2 + |\hat{a}|^2) = \frac{1}{2}(\hat{a}^T\hat{\mathbf{C}}\hat{a} - \hat{a}^T\hat{a}) + \mathcal{O}(|\hat{a}^3| = \hat{a}^T\big(\frac{1}{2}(\hat{\mathbf{F}}^T\hat{\mathbf{F}} - I)\big)\hat{a} + \mathcal{O}(\hat{a}^3) \\ \hat{\mathbf{E}} = \frac{1}{2}(\hat{\mathbf{C}} - I) \end{aligned}$$

Both the *right Cauchy-Green tensor* $\hat{\mathbf{C}}$ and the *Green-Lagrange* $\hat{\mathbf{E}}$ are refered to the Lagrangian coordinate system, hence the *reference configuration*.
Using similar arguments (see [12], compsda) Eulerian counterparts of the Lagrangian stress tensors can be derived.
The *left Cauchy-Green* strain tensor

$$\mathbf{b} = \hat{F}\hat{F}^T =$$

and the *Euler-Almansi* strain tensor

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

It is important to note that strain itself is nothing else than the measurement of line segments under deformation. Therefore strain alone is purely an observation, and it is not dependent on the material of interest. However one expects that a material undergoing strain, will give forces within the material due to neighboring material interacting with one another. Therefore one derive materialspecific models to describe how a certain material will react to a certain amount of strain. These strain measures are used to define models for *stress*, which is responsible for the deformation in materials (cite holzapfel). The dimention of stress is force per unit area.

1.4 Governing Equations

The fully Fluid-structure interaction problem is based on equations of balance laws, with auxiliary kinematic, dynamic and material relations. In this section, assumptions regarding these relations will be described briefly. A deeper review of the full FSI problem will be considered in the next chapter.

1.4.1 Fluid

We will throughout this thesis consider in-compressible fluids described by Navier-Stokes equations. We define the fluid density as ρ_f and fluid viscosity ν_f to be constant in time. Our phsyical unknowns fluid velocity v_f and pressure p_f both live in the time-dependent fluid domain $\hat{\mathbf{\Omega}}_f(t)$, with an eulerian configuration. Together with the equations of momentum and continuum, the Navier-Stokes equation is defined as,

Equation 1.4.1. *Navier-Stokes equation*

$$\rho\frac{\partial \mathbf{v}_f}{\partial t} + \rho\mathbf{v}_f \cdot \nabla\mathbf{v}_f = \nabla\cdot\sigma + \rho\mathbf{f}_f \quad \text{in } \Omega_f \tag{1.5}$$

$$\nabla\cdot\mathbf{v}_f = 0 \quad \text{in } \Omega_f \tag{1.6}$$

where $\mathbf{f}_s$ is some body force. Assuming a newtonian fluid the *Cauchy stress sensor* σ takes the form
$\sigma = -p_f I + \mu_f(\nabla\mathbf{v}_f + (\nabla\mathbf{v}_f)^T$.
Additional appropriate boundary conditions are supplemented to the equation for a given problem. The first type of of boundary conditions are Dirichlet boundary conditions,

$$\mathbf{v}_f = \mathbf{v}_f^D \quad \text{on } \Gamma_f^D \subset \partial\Omega_f \tag{1.7}$$

The second type of boundary condition are Neumann boundary conditions

$$\sigma_f\cdot\mathbf{n} = \mathbf{g} \quad \text{on } \Gamma_f^N \subset \partial\Omega_f \tag{1.8}$$

1.4.2 The solid

For the structure we use the Vernant-Kirchhoff(STVK) model of deformation of solids, defined in a Lagrangian coordinate system. The solid is often characterized by the Possion ratio and Young modulus. Lamè coefficients λ_s and μ_s are then given by the relation.

$$E_y = \frac{\mu_s(\lambda_s+2\mu_s)}{(\lambda_s+\mu_s)} \qquad \nu_s = \frac{\lambda_s}{2(\lambda_s+\mu_s)}$$

$$\lambda_s = \frac{\nu E_y}{(1+\nu_s)(1-2\nu_s)} \qquad \mu_s = \frac{E_y}{2(1+\nu_s)}$$

Our phsyical unknowns solid velocity v_s and deformation u_s is our physical unknowns defined in a time-dependent solid domain $\hat{\mathbf{\Omega}}_s(t)$. The balance of solid momentum is given by

Equation 1.4.2. *Solid momentum*

$$\rho_s \frac{\partial \mathbf{v}_s}{\partial t} = \nabla \cdot \mathbf{T} + \rho_s \mathbf{f}_s \quad \text{in } \Omega_s \tag{1.9}$$

where $\mathbf{f}_s$ is some body force, and $\mathbf{T}$ is the first *Piola-Kirchhoff* stress tensor. The relation

$$\mathbf{T} = \mathbf{FS} \tag{1.10}$$

connects the first *Piola-Kirchhoff* to the second *Piola-Kirchhoff* tensor. The material is chosen to be isotropic, hence $\mathbf{S}$ is on the form

$$\mathbf{S} = \lambda_s tr(\mathbf{E}) + 2\mu_s \mathbf{E}$$
$$E = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I})$$

Since the solid deformation is a quantity of interest a kinematic condition must be defined for the system of the form

$$\frac{\partial \mathbf{v}_s}{\partial t} = \mathbf{u_s} \quad \text{in } \Omega_s \tag{1.11}$$

One might ask the motivation of such an approach as the Lagrangian system could let us define the problem

$$\rho_s \frac{\partial^2 \mathbf{u}_s}{\partial^2 t^2} = \nabla \cdot \mathbf{T} + \rho_s \mathbf{f}_s \quad \text{in } \Omega_s \tag{1.12}$$

directly solving for the main quantity of interest namely deformation. However solving for $\mathbf{v}_s$ is more convenient, as it lets us handle constraints for the fluid-structure interaction problem easier. As for the fluid problem we define Dirichlet and Neumann boundary conditions on the form

$$\mathbf{v}_s = \mathbf{v}_s^D \quad \text{on } \Gamma_s^D \subset \partial\Omega_s$$
$$\sigma_s \cdot \mathbf{n} = \mathbf{g} \quad \text{on } \Gamma_s^N \subset \partial\Omega_s$$

Kapittel 2

Fluid Structure Interaction

The concepts of Fluid-structure interaction are often introduced in several engineering fields, for example bio mechanics and hydrodynamics. As we will see throughout this chapter, one of the main challenges of this field that our governing equation describing fluid and solids are defined on different coordinate systems.
We define Ω in the *reference configuration* be partitioned in a fluid domain $\hat{\mathbf{\Omega}}_\mathbf{f}$ and a structure domain $\hat{\mathbf{\Omega}}_\mathbf{s}$ such that $\Omega = \hat{\mathbf{\Omega}}_\mathbf{f} \cup \hat{\mathbf{\Omega}}_\mathbf{s}$. Further we define the interface $\hat{\mathbf{\Gamma}}$ as the intersection between these domains such that $\Gamma_i = \partial\hat{\mathbf{\Omega}}_\mathbf{f} \cap \partial\hat{\mathbf{\Omega}}_\mathbf{s}$. The fluid-structure interaction problem is then defined by the fluid and solid equations, and the transmission of the *kinematic* and *dynamic* conditions on the interface $\hat{\mathbf{\Gamma}}$.

$$\mathbf{v}_f = \mathbf{v}_s \tag{2.1}$$

$$\sigma_f \cdot \mathbf{n} = \sigma_s \cdot \mathbf{n} \tag{2.2}$$

A natural dilemma arises as the domain Ω undergoes deformation over time. Recall from chapter ?, that the solid equations are often described in the *Lagrangian coordinate system*, while the fluid equations are on the contrary described in the *Eularian coordinate system*. If the natural coordinate system are used for $\hat{\mathbf{\Omega}}_\mathbf{f}$ and $\hat{\mathbf{\Omega}}_\mathbf{s}$, the domains doesn't match and the interface $\hat{\mathbf{\Gamma}}$ doesn't have a general description for both domains. As such only one of the domains can be described in its natural coordinate system, while the other domain needs to be defined in some transformed coordinate system.

Fluid-structure interaction problems are formally divided into the *monolithic* and *partitioned* frameworks. In the monolithic framework all equations and interfaceconditions are solved simultaneously. If the *kinematic* (1.1) and *dynamic*(1.2) conditions are satisfied exactly at each timestep, the method denoted as a *strongly coupled* which is typically for a monolithic approach. However this strong coupling yields contributes to a stronger nonlinear behaviour of the whole system [17]. There are several nonlinear solution strategies for the monolithic approach.. Further the monolithic framework provides less modular software since the implementation often *ad hoc* for a certain monolithic approach.
In the *partitioned* framework one solves the equations of fluid and structure subsequently. The strength of such an approach is the wast range of optimized solvers developed for both the fluid and solid. However one major problem is that the interface conditions are not naturally met, and as such certain sub-iterations is required to achieve these.
Independent of framework, one of the major aspects of Fluid-structure interaction is the coupling of the fluid and solid equations through (1.1) and (1.2) . As the total system is exerted by external forces, the interface $\hat{\mathbf{\Gamma}}$ must fulfill the physical equilibrium of forces given by the two domains. Therefore, it is critical that the transmission of forces from the two domains are fulfilled in a consistent way. However as the interface conditions are not naturally met such as in a *partitioned* approach, we one must distinguish between *strongly* and *weakly* coupled schemes. *Partitioned*

solvers often enforce a weakly coupled schemes, meaning (1.1) and (1.2) are not strictly enforced in the calculation. Such an approach is often sufficient for some fields such as computations within aeroelasticity [6]. However by sub-iterations at each time step, these conditions can be enforced with high accuracy. When these conditions are met exactly, we define the scheme as *strongly* coupled.
The scope of FSI methods can formally be divided into *interface-tracking* and *interface-capturing* methods.[**?**]. In the *Interface-tracking* method, the mesh moves to accommodate for the movement of the structure as it deformes the spatial domain occupied by the fluid. As such the mesh itself tracksthe fluid-structure interface as the domain undergoes deformation. Such an approach is feasable as it allows for better control of mesh resolution near the interface, which in turn yeilds better control of this critical area. Among the *Interface-tracking* methods, the *arbitary Lagrangian-Eulerian* formulation is the most well-known approach [**?**], [**?**]. In this approach the structure is given in its natural *Lagrangian coordinate system*, while the fluid is transformed into an artificial *Lagrangian* coordinate system. From this approach tracking of the interface $\hat{\mathbf{\Gamma}}$ is more trivial, as it is fixed on a *reference system* and can be tracked by mappings defined in Chapter 2.

In *interface-capturing* methods one distinguish the fluid and solid domains by some phase variable over a fixed mesh. As such one captures the interface $\hat{\mathbf{\Gamma}}$ as it´s moving over the mesh element. This method is often emplyed in simulations of multiphase-flow. This ideas was extended in [4] were the authors proposed to transform the Lagrangian formulated structure equations in an Eulerian formulation, solving the system of equations in a fully Eulerian formulation. This approach was considered in this thesis, but implementation of tracking the interface in time was proven unsuccessful. Therefore ALE approach was finally chosen for this thesis. As such both the *fully Eulerian* and *ALE* concepts, strengths and weaknesses will be introduced following sub-chapters.

2.0.1 Fully Eulerian

This method keeps the fluid in its *Eulerian coordinates*, and such can be seen as the natural counterpart of the ALE method [18]. First proposed by , [3]. One motivation of such and approach is the handling of large-deformation, as the transformation to eulerian coordinates are purely natural.

2.0.2 Arbitary Lagrangian Eulerian formulation

MEANTION ALE CAN BEHAVE EITHER EULERIAN AND LAGRANGIAN The ALE method was initially developed to combine the strengths of the *Lagranngian* and *Eulerian* coordinate systems. As pointed out in chapter 2, the *Lagrangian* description is useful for tracking particles as they are act upon by forces. Hence its main contribution is the ability to track interfaces and materials with history dependent properties. In the ALE method one choose to keep the structure in its *Lagrangian coordinate system*, while transforming the fluid domain into an artificial coordinate system similar to the *Lagrangian coordinate system*. It is however important to note that there is no natural displacement in the fluid domain, hence this domain has no directly physical meaning [**?**], [2].
With this in mind, we will derive these transformations with the help of a new arbitary fixed reference system $\hat{\mathrm{W}}$, following the ideas and approaches found in [12]. Further we denote its deformation gradient as $\hat{\mathbf{F}}_w$ and its determinant $\hat{\mathrm{J}}_w$. Following the ideas from chapter 2, we introduce the invertibale mapping $\hat{\mathrm{T}}_w : \hat{\mathrm{W}} \to V(t)$, with the scalar $\hat{\mathrm{f}}(\hat{\mathrm{x}}_W, t) = f(x, t)$ and vector $\hat{\mathbf{w}}(\hat{\mathrm{x}}_W, t) = \mathbf{w}(x, t)$ counterparts.
For $\hat{\mathrm{V}} = \hat{\mathrm{W}}$, $\hat{\mathrm{W}}$ simply denotes the familiar Lagrangian description. In the case $\hat{\mathrm{V}} \neq \hat{\mathrm{W}}$, $\hat{\mathrm{W}}$ as pointed out earlier have no direct physical meaning. Hence it is important to notice that the physical velocity $\hat{\mathbf{v}}$ and the velocity of arbitrary domain $\frac{\partial \hat{\mathrm{W}}_w}{\partial t}$ doesn't necessary coincide. This observation is essential, as we will soon see.

We will first define the transformation of spatial and temporal derivatives from $V(t)$ to $\hat{\mathrm{W}}$ found in [12]

Lemma 2.1. Transformation of scalar spatial derivatives
Let f be a scalar function such that $f : V(t) \to \mathbb{R}$, then

$$\nabla f = \hat{\mathbf{F}}_W^{-T} \hat{\nabla} \hat{\mathrm{f}} \tag{2.3}$$

Lemma 2.2. Transformation of vector spatial derivatives
Let $\boldsymbol{w}$ be a vector field such that $\mathbf{w} : V(t) \to \mathbb{R}^d$, then

$$\nabla \mathbf{w} = \hat{\nabla} \hat{\mathbf{w}} \hat{\mathbf{F}}_W^{-1} \tag{2.4}$$

Lemma 2.3. Transformation of scalar temporal derivatives
Let f be a scalar function such that $f : V(t) \to \mathbb{R}$, then

$$\frac{\partial f}{\partial t} = \frac{\partial \hat{\mathrm{f}}}{\partial t} - (\hat{\mathbf{F}}_W^{-1} \frac{\partial \hat{\mathrm{T}}_W}{\partial t} \cdot \hat{\nabla}) \hat{\mathrm{f}} \tag{2.5}$$

In addition we need a consistent way to transform the induced stresses in the *Eulerian* coordinate system to $\hat{\mathrm{W}}$. Hence we introduce the *Piloa transformation*, found in most introduction courses in structure mechanics (ORANGE BOOK).

Lemma 2.4. T
Let $\boldsymbol{w}$ be a vector field such that $\mathbf{w} : V(t) \to \mathbb{R}^d$, then the Piola transformation of w is defined by

$$\mathbf{w} = \hat{\mathrm{J}}_W \hat{\mathbf{F}}_W^{-1} \hat{\mathbf{w}} \tag{2.6}$$

The Piola transformation can be further extended to transform tensors, see [12], Orange book. This results is essential as it allows us to transform surface forces induced by the *Cauchy stress tensor* on our arbitrary coordinate system $\hat{\mathrm{W}}$. Lemma 1.4 brings us to *the first Piola Kirchhoff stress tensor* $\hat{\mathbf{P}} = \hat{\mathrm{J}}_W \hat{\sigma} \hat{\mathbf{F}}_W^{-T}$, mentioned in chapter 2.
We now have the necessary tools to transform the conservation principles introduced in the fluid problem in chapter 2. Recall the Navier-Stokes equation defined in the *Eulerian coordinate system* V(t).

$$\begin{aligned} \rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} &= \nabla \cdot \sigma + \rho \mathbf{f} \\ \nabla \cdot \mathbf{v} &= 0 \end{aligned}$$

Using our newly introduced transformations of derivatives we map the equation to the arbitrary reference system $\hat{\mathrm{W}}$. We will first consider the transformation of the*material derivative*. By

$$\begin{aligned} \frac{d\mathbf{v}}{dt}(x,t) &= \frac{\partial \mathbf{v}}{\partial t}(x,t) + \nabla \mathbf{v}(x,t) \cdot \frac{\partial x}{\partial t} \\ \frac{d\mathbf{v}}{dt}(x,t) &= \frac{\partial \mathbf{v}}{\partial t}(x,t) + \nabla \mathbf{v}(x,t) \cdot \mathbf{v} \end{aligned}$$

Note $\frac{\partial x}{\partial t}$ is the velocity of particles and not the transformation velocity $\frac{\partial \hat{\mathrm{T}}_W}{\partial t}$. By lemma(1.1, 1.2, 1.3) we have

$$\begin{aligned} \frac{d\mathbf{v}}{dt}(x,t) &= \frac{\partial \hat{\mathbf{v}}}{\partial t}(x,t) - (\hat{\mathbf{F}}_W^{-1} \frac{\partial \hat{\mathrm{T}}_W}{\partial t} \cdot \hat{\nabla}) \hat{\mathbf{v}} + \hat{\mathbf{F}}_W^{-T} \hat{\nabla} \hat{\mathbf{v}} \cdot \hat{\mathbf{v}} \\ \mathbf{v} \cdot \nabla \mathbf{v} = \nabla \mathbf{v} \mathbf{v} &= \hat{\nabla} \hat{\mathbf{v}} \hat{\mathbf{F}}_W^{-1} \hat{\mathbf{v}} = (\hat{\mathbf{F}}_W^{-1} \hat{\mathbf{v}} \cdot \hat{\nabla}) \hat{\mathbf{v}} \quad \textit{FINN KILDE} \end{aligned}$$

These results can be used to show that

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \frac{\partial \hat{\mathbf{v}}}{\partial t} + (\hat{\mathbf{F}}_W^{-1}(\hat{\mathbf{v}} - \frac{\partial \hat{\mathrm{T}}_W}{\partial t}) \cdot \hat{\nabla})\hat{\mathbf{v}}$$

By applying *the first Piola Kirchhoff stress tensor* directly we transform the surface stress by

$$\nabla \cdot \sigma = \nabla \cdot (\hat{\mathrm{J}}_W \hat{\sigma} \hat{\mathbf{F}}_W^{-T})$$

In general, σ is presumed on the form of a Newtonian fluid. However special care must be taken, as $\sigma \neq \hat{\sigma}$ due to spatial derivatives within the tensor. Hence

$$\sigma = -pI + \mu_f(\nabla \mathbf{v} + (\nabla \mathbf{v})^T$$
$$\hat{\sigma} = -\hat{\mathrm{p}}I + \mu_f(\hat{\nabla}\hat{\mathbf{v}}\hat{\mathbf{F}}_W^{-1} + \hat{\mathbf{F}}_W^{-T}\hat{\nabla}\hat{\mathbf{v}}^T)$$

For the conservation of continuum we apply the *Piola Transformation* such that

$$\nabla \cdot \mathbf{v} = \nabla \cdot (\hat{\mathrm{J}}\hat{\mathbf{F}}_W^{-1}\hat{\mathbf{v}})$$

With the introduced mapping identities we have the necessary tools to derive a full fluid-structure interaction problem defined of a fixed domain. Since the structure already is defined in its natural Lagrangian coordinate system, no further derivations are needed for defining the total problem.

Equation 2.0.1. *ALE problem on a fixed domain*

$$\hat{\mathrm{J}}\frac{\partial \hat{\mathbf{v}}}{\partial t} + \hat{\mathrm{J}}(\hat{\mathbf{F}}_W^{-1}(\hat{\mathbf{v}} - \frac{\partial \hat{\mathrm{T}}_W}{\partial t}) \cdot \hat{\nabla})\hat{\mathbf{v}} = \nabla \cdot (\hat{\mathrm{J}}_W \hat{\sigma} \hat{\mathbf{F}}_W^{-T}) + \rho_f \hat{\mathrm{J}}\mathbf{f}_f \quad \text{in } \Omega_f \tag{2.7}$$

$$\nabla \cdot (\hat{\mathrm{J}}\hat{\mathbf{F}}_W^{-1}\hat{\mathbf{v}}) \quad \text{in } \Omega_f \tag{2.8}$$

$$\rho_s \frac{\partial \hat{\mathbf{v}}_s}{\partial t} = \nabla \cdot \mathbf{FS} + \rho_s \mathbf{f}_s \quad \text{in } \Omega_s \tag{2.9}$$

$$\frac{\partial \hat{\mathbf{v}}_s}{\partial t} = \hat{\mathbf{u}}_s \quad \text{in } \Omega_s \tag{2.10}$$

$$\hat{\mathbf{v}}_s = \hat{\mathbf{v}}_f \quad \text{on } \Gamma_i \tag{2.11}$$

$$\hat{\mathrm{J}}_W \hat{\sigma} \hat{\mathbf{F}}_W^{-T} \cdot \mathbf{n} = \mathbf{FS} \cdot \mathbf{n} \quad \text{on } \Gamma_i \tag{2.12}$$

Fluid mesh movement

In the ALE framwork one of the most limiting factors is the degeneration of the mesh due to large deformations. Even the most advanced ALE formulated schemes reaches a limit when only re-meshing is nesecarry [15]. Consequently the choice of an appropriate mesh moving technique is essential to preserve a feasible mesh quality for the simulation of fluid flow. Let the total domain deformation $\hat{\mathrm{T}}(\hat{\mathrm{x}}, t)$ be divided into the solid and fluid deformation T_s, T_f, were the fluid deformation is mapped to the arbitrary fixed reference system $\hat{\mathrm{W}}$ presented in the last subsection. Then the ALE map T_f on the form

$$\hat{\mathrm{T}}_f(\hat{\mathrm{x}}, t) = \hat{\mathbf{x}} + \hat{\mathbf{u}}_f(\hat{\mathbf{x}}, t)$$

is constructed such that $\hat{\mathbf{u}}_f$ is an extension of the solid deformation $\hat{\mathbf{u}}_s$ from the interface to the fluid domain. Several extentions have been proposed throuhout the litteratur, and for an overview the reader is refered to [11], and the reference therein. The construction of such extensions often involves solving some auxiliary problem on a partial differential equation(PDE) form, mainly of second-order. The *Laplacian* and *pseudo-elasticity* extentions are examples, which will be considered in this thesis. These extensions are beneficial in terms of simplicity and computational efficiency, but comes with a cost of user mesh customization. One often want to ensure a desired mesh position and some regularity of mesh spacing on the boundary, but it is impossible for second order extensions to specify both [7]. Therefore the author of [7], proposes a fourth-order PDE, the *biharmonic* extensions, to improve the regularity of the mesh deformation.

Laplacian model

The main motivation for a *Laplacian* smoothing is due to its simplicity and due to its property of bounding the interior displacements to the boundary values.

$$
\begin{aligned}
&-\hat{\nabla}\cdot(\alpha^q\hat{\nabla}\hat{\mathbf{u}}) = 0\\
&\hat{\mathbf{u}}_f = \hat{\mathbf{u}}_s \ \text{ on } \ \Gamma\\
&\hat{\mathbf{u}}_f = 0 \ \text{ on } \ \partial\hat{\mathbf{\Omega}}_f/\Gamma
\end{aligned}
$$

Most favourable, the largest mesh deformation occuring should be confined to the interal part of the mesh as it causes the least distortion [9]. Therefore the introduced diffusion parameter α, often raised to some power q, is introduced to manipulate this behaviour. The form of this paramter is often problem specific, as selective treatment of the elements may vary from different mesh deformation problems. A jacobian based method was introduced in [13]. In [9], the authors reviewed several distance based options, where α was some function of the distance to the closest moving boundary. This method was adopted in this thesis on the form

$$\alpha(x) = \frac{1}{x^q} \quad q = -1$$

However as pointed out by [8], one of the main disadvantages of using the linear Laplace equation is that the equation solves the mesh deformation components independently of one another. Say one have deformation only in the x-coordinate direction, the interior mesh points will only be moved along this deformation. Such a behavior restricts the use to the Laplace equation of mesh extrapolation purposes.

Linear elastic model

Considering a linear elastic model for mesh moving was first introduced in [14]. Both [5]

$$
\begin{aligned}
&\nabla\cdot\sigma = 0\\
&\sigma = \lambda Tr(\epsilon)I + 2\mu\epsilon\\
&\epsilon = \frac{1}{2}(\nabla u + \nabla u^T
\end{aligned}
$$

Where Lamé constants λ and ν are given as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \ \ \mu = \frac{E}{2(1+\nu)}$$

One of the main motivations for introducing such a model is the manipulation of Young's modulus E, and the poisson´s ration ν. Recall that Young's modulus is the measurement of the a materials stiffness, while the poission's ratio describe the materials stretching in the transverse direction under extension in the axial direction. Manipulating these parameters one can influence the mesh deformation, however the choice of these parameters have proven not to be consistent, and to be dependent of the given problem.

In [16] the author proposed a negative possion ratio, which makes the model mimic an auxetic material. Such materials becomes thinner in the perpendicular direction when they are submitted to compression, and this property is feasible for mesh under deformation.
One of the most common approach is to set ν as a constant in the range $\nu \in [0, 0.5)$ and let E be the inverse of the distance of an interior node to the nearest boundary surface [11]. The authors of [1] used this property and also argued that the Young's modulus also could be chosen as the inversely proportional to the cell volume. They also pointed out that both approaches would give the desired result that the small cells around the solid surface would modeled rigid, moving with the surface of the solid as it undergoes deformation. On the other hand cells further away will deform to counter the effects close to the solid surface.

Biharmonic model

Using a biharmonic mesh deformation model provides further freedom in terms of boundary conditions, and the reader is encoured to consult [7] for a deeper review. We will in combination with [16] present two main approaches the biharmonic model is defined as

$$\hat{\nabla}^2 \hat{\mathrm{u}} = 0 \quad \text{on} \ \ \hat{\mathbf{\Omega}}_f$$

By introducing a second variable on the form $\hat{\mathrm{w}} = -\hat{\nabla}\hat{\mathrm{u}}$, we get the following system defined by

$$\begin{aligned} \hat{\mathbf{w}} &= -\hat{\nabla}^2 \hat{\mathbf{u}} \\ -\hat{\nabla}\hat{\mathbf{w}} &= 0 \end{aligned}$$

This model is defined in a mixed formulation, and as such the prize for quality and control of mesh deformation comes with the cost of more computational demanding problem.
For the boundary conditions two types has been proposed in [16]. Let $\hat{\mathbf{u}}_f$ be decomposed by the components $\hat{\mathbf{u}}_f = (\hat{\mathrm{u}}_f^{(1)}.\hat{\mathrm{u}}_f^{(2)})$. Then we have

$$\begin{aligned} &\textbf{Type 1} \quad \hat{\mathrm{u}}_f^{(k)} = \frac{\partial \hat{\mathrm{u}}_f^{(k)}}{\partial n} = 0 \quad \partial\hat{\mathbf{\Omega}}_f/\Gamma \ \text{ for } k = 1, 2 \\ &\textbf{Type 2} \quad \hat{\mathrm{u}}_f^{(1)} = \frac{\partial \hat{\mathrm{u}}_f^{(1)}}{\partial n} = 0 \ \text{ and } \ \hat{\mathrm{w}}_f^{(1)} = \frac{\partial \hat{\mathrm{w}}_f^{(1)}}{\partial n} = 0 \quad \text{on } \hat{\mathbf{\Omega}}_f^{in} \cup \hat{\mathbf{\Omega}}_f^{out} \\ &\qquad\qquad \hat{\mathrm{u}}_f^{(2)} = \frac{\partial \hat{\mathrm{u}}_f^{(2)}}{\partial n} = 0 \ \text{ and } \ \hat{\mathrm{w}}_f^{(2)} = \frac{\partial \hat{\mathrm{w}}_f^{(2)}}{\partial n} = 0 \quad \text{on } \hat{\mathbf{\Omega}}_f^{wall} \end{aligned}$$

With the first type of boundary condition the model can interpreted as the bending of a thin plate, clamped along its boundaries. The form of this problem has been known since 1811, and its derivation has been connected with names like French scientists Lagrange, Sophie Germain, Navier and Poisson [10].
The main motivation for second type of boundary condition is for a rectangular domain where the coordinate axes match the Cartesian coordinate system [16]. In such a configuration, the mesh movement is only constrained in the perpendicular direction of the fluid boundary, leading to mesh movement in the tangential direction. This special case reduces the effect of distortion of the cells.

2.1 Discretization of the FSI problem

Kapittel 3

Verification and Validation

During the last decade, the amount of reaserch regarding simulations of physical problems has grown vast. Even though computers have changed our ways of solving real world problems, thrusting blindly numbers generated from a computer code has proven to be naive. It doesn't take a lot of coding experience before one realizes the many things that can brake down and produce unwanted and even suprisingly unexpected results. With this in mind, computer scientists and engineers need some common ground to check if a computer code works as expected. And it is here the framework of verification and validation plays and important role.

An elegant and simple definition found throughout the litterature of verification and validation framwork, used by Roache [**?**], states *verification* as "solving the equations right", and *validiation* as "solving the right equations". "Solving the right equations"is rather vaguely, a measurement is needed. We will in this thesis use the more detailed description found in [**?**].

> The code author defines precisely what continuum partial differential equations and continuum boundary conditions are being solved, and convincingly demonstrates that they are solved correctly, i.e., usually with some order of accuracy, and always consistently, so that as some measure of discretization (e.g. the mesh increments) $\nabla \rightarrow 0$, the code produces a solution to the continuum equations; this is Verification.
>
> — *Roache, P.J.*

Roach [**?**], further distinguish between the verification of *code* and *calculation*. Verification of code is seen as achieving the expected order of accuracy of the implementation, while verification of calculation is the measure of error against a known solution. Of these .. has proven to
The goal of this chapter is to verify our implementations using the method of manufactured solution (MMS).

3.1 Verification of Code

For scientists exploring physical phenomena,systems of partial differential equations (PDE´s) are often encountered. For their application it is important that these equations are implemented and solved numerically the right way. Therefore insurence of right implemention is crucial.

Let a partial differential equation of interest be on the form

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

Here **L** is a differential operator, **u** is variable the of interest, and **f** is some sourceterm.
In the method of manufactured solution, one first manufactures a **u**, which is differentiated with **L** which yields a sourceterm **f**. The sourceterm **f** with respect to the selected solution **u** is then used as input in the implementation, yielding a numerical solution. Verification of code and calculation is then performed on the numerical solution against the manifactured solution **u**.

The beauty of such an approach as mentioned by Roache [**?**], is that our exact solution can be constructed without any physical reasoning. As such, code verificaion is purly a mathematical exercise were we are only interested if we are solving our equation right. These sentral ideas have existed for some time, but the concept of combining manufactured exact solution in in partnership with refinement studies of of computational mesh has been absent. One of the earliest was Steinberg and Roache [**?**] using these principles deliberately for *verification of code* (estimate order of convergence)

To deeply verify the robustness of the method of manufactured solution, a report regarding code verification using this approach was published by Salari and Knupp [**?**]. This thorough work applied the method for both compressible and incompressible time-dependent Navier-Stokes equation. To prove its robustness the authors delibritary implemented code errors in a verified Navier-Stokes solver by MMS presented in the report. In total 21 blind testcases where implemented, where different approaches of verifcation frameworks were tested. Of these 10 coding mistakes that reduces the observed order-of-accuracy was implemented. Here the method of manufactured solution captured all of them.

For the purpose of verification of calculation we need to calculate the error of our numerical simulation. Let $\mathbf{u}_h$ denote our numerical solution and $\mathbf{u}$ be our exact solution. By letting $||\cdot||$ be the L^2 norm, we define the error as

$$E = ||\mathbf{u} - \mathbf{u}_h||$$

Assuming our computational mesh is constructed by equilateral triangles, and that our simulations are solved with a constant timestep, the total error contribution from the temporal and spatial discretized PDE can be written as

$$E = A\delta x^l + B\delta t^k$$

Where A and B are constants, and l and k denote the expected convergencerate... FYLL INN REF FRA ANNET KAP OM EXPECTED CONVERGENCERATE
. In order to evalute properties of either the spatial or temporal discretization, we must reduce the numerical error contribution of the discretization not of interest. Say we would like to evaluate the convergencerate of the spatial discretization, then the temporal error must be reduced in order to not poute..
Even though the method of MMS a certain freedom in the construction of a manufactured solution, certain guidelines have been proposed ([**?**], [**?**], [**?**]).

- To ensure theoretical order-of-accuracy, the manufactured solution should be constructed of polynomials, exponential or trigonometric functions to construct smooth solutions.
- The solution should be utilized by every term in the PDE of interest, such that no term yields zero. (få frem at en løsning må velges slik at ingen differentials blir 0
- Certain degree to be able to calculate expected order of convergence (Få frem at må ha grad noktil å kunne regne convergencerate)

Fluid structure interaction consists of several buildingblocks of fluid and structure equations describing forces exerted from one another. With this in mind a verification of the full FSI code can be tedious as implementation errors yielding non-desired results can be hard to find. We will

therefore provide verification of each buildingblock until we reach the total system of equations.

In the following sections we will overlook the implemented solvers. Unlesss specified, all simulations are implemented on an unit square. Simulation parameters will be reported,
For construction of the sourceterm **f** the Unified Form Language (UFL) [**?**] provided in FEniCS Project will be used. UFL provides a simple yet powerfull method of declaration for finite element forms. An example will be provided in the Fluid Problem section.

3.1.1 Fluid Problem

One question which arises during the construction of the manufactured solution is, which formulation of the Navier-Stokes equation do we want to calculate the sourceterm. From a numerical point of view constructing the sourceterm from the Eulerian formulation and then map the equation would be feasable. Such an apporach limits the evaluation of computational demanding routines such as the generation of the deformation gradient $\hat{\mathbf{F}}$ and its Jacobian $\hat{\mathrm{J}}$. Even though refinement studies of spatial and temporal discretizations are often computed on small problems, such speedups are important when running larger simulations. Recall from Chapter ??? the ALE formulation of the Navier Stokes equation.

$$\rho_f \hat{\mathrm{J}} \frac{\partial \hat{\mathbf{u}}}{\partial t} + \hat{\mathrm{J}}\hat{\mathbf{F}}^{-1}(\hat{\mathbf{u}} - \hat{\mathbf{w}}) \cdot \nabla \hat{\mathbf{u}} - \nabla \cdot \hat{\mathrm{J}}\sigma\hat{\mathbf{F}}^{-T} = f$$
$$\hat{\mathrm{d}}\mathrm{iv}(\hat{\mathrm{J}}\hat{\mathbf{F}}^{-1}\hat{\mathrm{u}}) = 0$$

Algorithm 3.1: Descriptive Caption Text

```
u_x = "cos(x[0])*sin(x[1])*cos(t_)"
u_y = "-sin(x[0])*cos(x[1])*cos(t_)"
p_c = "sin(x[0])*cos(x[1])*cos(t_)"

f = rho*diff(u_vec, t_) + rho*dot(grad(u_vec), (u_vec - w_vec)) -
div(sigma_f(p_c, u_vec, mu))
```

We will on the basis of the presented guidelines define the manufactured solution.

$$u = sin(x + y + t)^2$$
$$v = cos(x + y + t)^2$$
$$p = cos(x + y + t)$$

3.2 Validation of Code

From a thorough process of verifying our code, we can pursue validation activities on the assumption that our computational model compute accurate solutions. As we have experienced verification of code can be a tedious task, but its complexity is reduced to issues of mathematical and numerical nature. When it comes to validation on the other hand, numerous potential problems must be assessed. Does the mathematical model describe the the physical process of interest? What about the influence of of experimental measurement methods and their uncertainty ? But a as pointed out by /oberkampf. nevn paper of bok , a successful validation rise thrust in our mathematical computation... FIX
In the literature several different and often conflicting definitions of validation have been proposed /Rykiel . In his thorough work Rykiel also points out a main concern over that in all the confusion, there has never been more arising demands of validating models describing the real world.

/Refsgaard and Henriksen (2004) has propsed the following definition which we will use

Definition 3.1. Model Validation:
Substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model

This means as /Rykiel earlier proposed, the method of validation is not some method "for certifying the truth of current scientific understanding ... Validation means a model is acceptable for its intended use it meets specific performance requirements". SKRIV NOE OM TUREK HVA HVA DET EXPERIMENTET TESTER OSV

Kapittel 4

Implementation of Fluid Structure Interaction

We will in this section give an overview of the total Fluid-Structure interaction implementation introduced in chapter 2. A brief description will be given for the most central components and technologies used for this thesis.

4.1 FEniCS

The main component of this thesis is the FEniCS project, an open-source finite element environment for solving partial differential equations (https://fenicsproject.org/). Using a combination of high-level Python and C++ interfaces, mathematical models can be implemented compactly and efficiently. FEniCS consists of several sub-modules and we will give a brief overview of the most central components used during implementation and computation.

4.1.1 DOLFIN

DOLFIN is the computational C++ backend of the FEniCS project, and the main user interface. It unifies several FEniCs components for implementing of computational mesh, function spaces, functions and finite element assembly.

- UFL (The Unified Form Language) is a domain specific language, used for the discretization of mathematical abstractions of partial differential equations on a finite element form. Its implementation on top of Python, makes it excellent to define problems close to their mathematical notation without the use of more complex features. One uses the term *form* to define any representation of some mathematical problem defined by UFL.
- FFC (The form compiler) compiles the finite elements variation forms given by UFL, generating low-level efficient C++ code
- FIAT the finite element backend, covering a wide range of finite element basis functions used in the discretization of of the the finite-element forms. It covers a wide range of finite element basis functions for lines, triangles and tetrahedras.

DOLFIN also incorporate the necessary interfaces to external linear algebra solvers and data structures. Within FEniCS terminology these are called linear algebra backends. PETSc is the default setting in FEniCS, a powerful linear algebra library with a wide range of parallel linear and nonlinear solvers and efficient as matrix and vector operations for applications written in C, C++, Fortran and Python.

4.2 Implementation

As implementation of mathematics differ from the choices of programming languages and external libraries, a deep dive within the implementation in FEniCS will not be covered in this thesis. Only variational forms and solvers will be presented as to give the reader a general overview of the key concept and the interpretation of mathematics. Basic knowledge of coding is assumed of the reader.

4.2.1 Variational Form

Implementation of the code-blocks of the fluid variational form given in Chapter 3, and Newton solver will be presented. It is not the intention to give the reader a deep review of the total implementation, but rather briefly point out key ideas intended for efficient speedup of the calculation. These ideas have proven essential as for the reduction of computation time of the complex problem.

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

def J_(U):
        return det(F_(U))

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))

def A_E(J, v, d, rho_f, mu_f, psi, dx_f):
    return rho_f*inner(J*grad(v)*inv(F_(d))*v, psi)*dx_f \
        + inner(J*sigma_f_u(v, d, mu_f)*inv(F_(d)).T, grad(psi))*dx_f


def fluid_setup(v_, p_, d_, n, psi, gamma, dx_f, ds, mu_f, rho_f, k, dt, v_deg
    , theta, **semimp_namespace):

        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

        return dict(F_fluid_linear = F_fluid_linear, F_fluid_nonlinear =
    F_fluid_nonlinear)
```

Algorithm 4.1: thetaCN.py

Alorithm 1.1 presents the implementation of the fluid residue, used in the Newton iterations. Apart from the rather lengthy form of the fluid residual, the strength of Unified Form Language preserving the abstract formulation of the problem is clear. The overall representation of the problem is by now just a form, its a representation and does not yet define vectors or matrices.

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

    while rel_res > rtol and residual > atol and Iter < max_it:
        if Iter % 4  == 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
```

Algorithm 4.2: newtonsolver.py

4.3 Optimization of Newtonsolver

As for any program, the procedure of optimization involves finding the bottleneck of the implementation. Within computational science, this involves finding the area of code which is the primary consumer of computer resources.
As for many other applications, within computational science one can often assume the consummation of resources follows the *The Pareto principle*. Meaning that for different types of events, roughly 80% of the effects come from 20% of the causes. An analogy to computational sciences it that 80% of the computational demanding operations comes from 20% of the code. In our case, the bottleneck is the newtonsolver. The two main reasons for this is

- **Jacobian assembly**
  The construction of the Jacobian matrix for the total residue of the system, is the most time demanding operations within the whole computation.

- **Solver**.
  As iterative solvers are limited for the solving of fluid-structure interaction problems, direct solvers was implemented for this thesis. As such, the operation of solving a linear problem at each iteration is computational demanding, leading to less computational efficient operations. Mention order of iterations?

Facing these problems, several attempts was made to speed-up the implementation. The FEniCS project consist of several nonlinear solver backends, were fully user-customization option are available. However one main problem which we met was the fact that FEniCS assembles the matrix of the different variables over the whole mesh, even though the variable is only defined in one to the sub-domains of the system.In our case the pressure is only defined within the fluid domain, and therefore the matrix for the total residual consisted of several zero columns within the structure region. FEniCS provides a solution for such problems, but therefore we were forced to construct our own solver and not make use of the built-in nonlinear solvers.

The main effort of speed-up were explored around the Jacobian assembly, as this was within our control.
Of the speed-ups methods explored in this thesis we will specify that some of them were *consistent* while others were *nonconsistent*. Consistent methods are methods that always will work, involving smarter use of properties regarding the linear system to be solved. The non-consistent method presented involves altering the equation to be solved by some simplification of the system. As these simplifications will alter the expected convergence of the solver, one must take account for additional Newton iterations against cheaper Jacobi assembly. Therefore one also risk breakdown of the solver as the Newton iterations may not converge.

4.4 Consistent methods

4.4.1 Jacobi buffering

By inspection of the Jacobi matrix, some terms of the total residue is linear terms, and remain constant within each time step. By assembling these terms only in the first Newton iteration will save some assembly time for the additional iterations needed each time step. As consequence the convergence of the Newton method should be unaffected as we do not alter the system.

4.5 Non-consisten methods

4.5.1 Reuse of Jacobian

As the assembly of the Jacobian at each iteration is costly, one approach of reusing the Jacobian for the linear system was proposed. In other words, the LU-factorization of the system is reused until the Jacobi is re-assembled. This method greatly reduced the computational time for each time step. By a user defined parameter, the number of iterations before a new assembly of the Jacobian matrix can be controlled.

4.5.2 Quadrature reduce

The assemble time of the Jacobian greatly depends on the degree of polynomials used in the discretisation of the total residual. Within FEniCS this parameter can be controlled, and as such we can specify the order of polynomials representing the Jacobian. The use of lower order polynomials reduces assemble time of the matrix at each newton-iteration, however it leads to an inexact Jacobian which may results to additional iterations.

Kapittel 5

Numerical Results

In this chapter the main calculations of the proposed theories and will be presented.

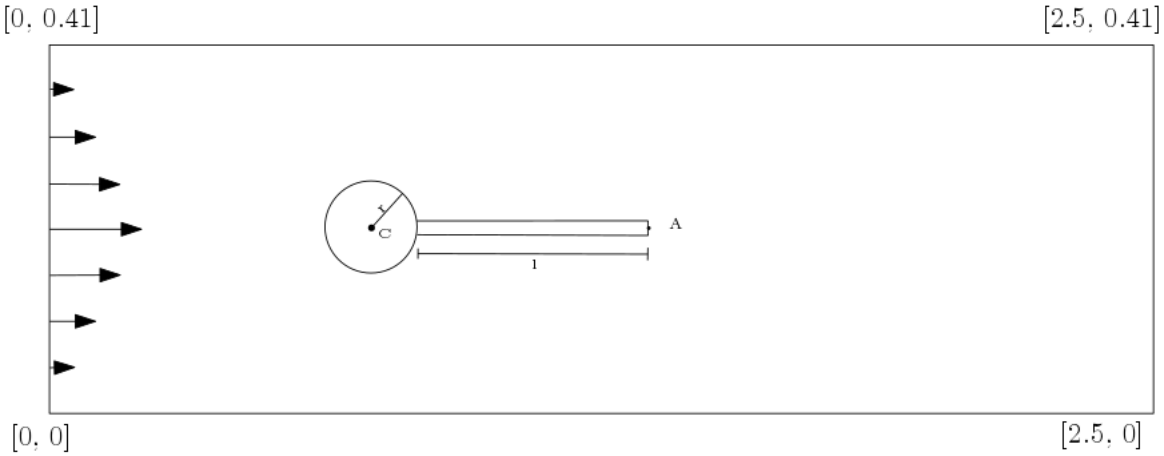
5.1 Verification

5.2 Validation

For verification purposes the numerical benchmark presented in [**?**] has been chosen for this thesis. This benchmark as been widely accepted throughout the fluid-structure interaction community as a rigidly validation benchmark. This is mainly due to its diversity of tests included, challenging all the main components of a FSI solver.
The benchmark is divided into three main testenvironments. In the first environment the purely fluid solver is tested for a range of different inflow parameters.
The second environment regards the purely structure implementation, regarding bending of the elastic flag. We will in this thesis consider the final environment, testing the total system in terms of a fluid-structure interaction problem. The others have been tested and proved to be an essential part of the development of the solver, but will be for brevity not reported.

The fluid-structure interaction validation benchmark is divided into three different problems with increasing difficulty, posing different challenges to the implementation. Each problem alters the fluid and solid parameters to provoke different behavior of the system.
Several quantites for comparion is presented in [**?**] for validation purposes. We will report

Figur 5.1: Domain configuration



- The position of point A(t) as the strucutre undergoes deformation.
- Drag and lift forces exerted on of the whole interior geometry in contact with the fluid, consisting of the rigid circle and the elastic beam.

$$(F_D, F_L) = \int_\Gamma \sigma \cdot \mathbf{n} dS$$

  Where **n** is the unit normal vector, pointing into the fluid domain.

The amplitude and mean values for the time dependent properties are calculated from the last period of oscillations, together with the period.

Tabell 5.1: Benchmark environment

Solid parameters			
parameter	FSI1	FSI2	FSI3
$\rho^s[10^3 \frac{kg}{m^3}]$	1	10	1
ν^s	0.4	0.4	0.4
$\mu^s[10^6 \frac{kg}{ms^2}]$	0.5	0.5	2.0
Fluid parameters			
$\rho^f[10^3 \frac{kg}{m^3}]$	1	1	1
$\nu^f[10^{-3} \frac{m^2}{s}]$	1	1	1
U	0.2	1	2
parameter	FSI1	FSI2	FSI3
Re	20	100	200

5.2.1 FSI1

The first environment yields a steady state solution for the system. It is meant as a basic implementation test as it applies small deformations to the system. Therefore is provides a test for the solving procedure, but doesn't excess large constrain of choice of mesh extrapolation operator.

Tabell 5.2: FSI 1

$\Delta t = 0.5$						
Model	nel	ndof	ux of A [x 10^3]	uy of A [x 10^3]	Drag	Lift
Biharmonic bc2	1	1	0.0228	0.7740	14.17280	0.7614
Biharmonic bc1	1	1	0.0228	0.7737	14.17281	0.7612
Elastic	1	1	0.0227	0.7960	14.17283	0.7607
Laplace	1	1	0.0227	0.7958	14.17285	0.7607
$\Delta t = 0.1$						
Model	nel	ndof	ux of A [x 10^3]	uy of A [x 10^3]	Drag	Lift
Biharmonic bc2	1	1	0.0228	0.7743	14.17279	0.76162
Biharmonic bc1	1	1	0.0228	0.7739	14.17280	0.76142
Elastic	1	1	0.0228	0.7962	14.17281	0.76092
Laplace	1	1	0.0228	0.7961	14.17284	0.76090

5.2.2 FSI2

The second environment results in a periodic solution. It proved to be one of the most demanding tests due to its large deformation, leading to the risk of entangled mesh cells. As such this raised the need for a high quality extrapolation of the solid deformation.

5.2.3 FSI3

The final environment does not induce deformation to the extent of the FSI2 benchmark. However a critical phase in the transition to the periodic solution was discovered, where the pressure oscillation induces a large deformation to the system.

Tabell 5.3: FSI 3

Test	x-comp A	y-comp A	drag	lift
bc1			442.9 +/ 17.26	2.83 +- 148.8
bc2			442.8 +/ 17.44	3.06 +-147.9
Ref	2.69 2.53	1.48 34.38 5.3	457. 22.66 10.9	2.22 149.78 5.3

5.3 Mesh movement

The final enviroment

Bibliografi

[1] Robert T Biedron and Elizabeth M Lee-Rausch. Rotor Airloads Prediction Using Unstructured Meshes and Loose CFD/CSD Coupling.

[2] J Donea, A Huerta, J.-Ph Ponthot, and A Rodríguez-Ferran. Arbitrary Lagrangian-Eulerian methods. (1969):1–38, 2004.

[3] Th Dunne. An Eulerian approach to uid – structure interaction and goal-oriented mesh adaptation. *International Journal for Numerical Methods in Fluids*, (December 2005):1017–1039, 2006.

[4] Thomas Dunne and Rolf Rannacher. Adaptive Finite Element Approximation of Fluid-Structure Interaction Based on an Eulerian Variational Formulation. *Fluid-Structure Interaction*, 53:110–145, 2006.

[5] Richard P Dwight. Robust Mesh Deformation using the Linear Elasticity Equations.

[6] Miguel A Fernández and Jean-Frédéric Gerbeau. Algorithms for fluid-structure interaction problems. 2009.

[7] Brian T. Helenbrook. Mesh deformation using the biharmonic operator. *International Journal for Numerical Methods in Engineering*, 2003.

[8] Su-Yuen Hsu, Chau-Lyan Chang, and Jamshid Samareh. A Simplified Mesh Deformation Method Using Commercial Structural Analysis Software.

[9] Hrvoje Jasak and Željko Tuković. Automatic mesh motion for the unstructured Finite Volume Method. *Transactions of Famena*, 30(2):1–20, 2006.

[10] V V Meleshko. Bending of an Elastic Rectangular Clamped Plate: Exact Versus 'Engineering' Solutions. *Journal of Elasticity*, 48(1):1–50, 1997.

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

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

[13] K Stein, T Tezduyar, and R Benney. Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements.

[14] T E Tezduyar, M Behr, S Mittal, and A A Johnson. COMPUTATION OF UNSTEADY INCOMPRESSIBLE FLOWS WITH THE STABILIZED FINITE ELEMENT METHODS: SPACE-TIME FORMULATIONS, ITERATIVE STRATEGIES AND MASSIVELY PARALLEL IMPLEMENTATIONSt. *New Methods in Transient Analysis ASME*, 246(143), 1992.

[15] Wolfgang A. Wall, Axel , Gerstenberger, Peter , Gamnitzer, Christiane , Förster, and Ekkehard , Ramm. Large Deformation Fluid-Structure Interaction – Advances in ALE Methods and New Fixed Grid Approaches. In *Fluid-Structure Interaction: Modelling, Simulation, Optimisation*, pages 195—-232. Springer Berlin Heidelberg, 2006.

[16] Thomas Wick. *Adaptive Finite Element Simulation of Fluid-Structure Interaction with Application to Heart-Valve*. PhD thesis, Heidelberg.

[17] Thomas Wick. Solving Monolithic Fluid-Structure Interaction Problems in Arbitrary Lagrangian Eulerian Coordinates with the deal.II Library.

[18] Thomas Wick. Fully Eulerian fluid-structure interaction for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 255:14–26, 2013.