Thesis Title

Institution Name

Author Name

Day Month Year

Innhold

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.

Before considering the full FSI problem will discuss the fluid and solid equations separately. These equations will be stated in different frameworks. The fluid is described in an Eulerian framework, where points in the domain are fixed and the fluid flow passes through these points. This reasoning can be understood if one were to stand by the side of river and watch the water flow down. The continuous deformation of fluid flow makes this the most sensible approach, the water becomes the cupso to speak.

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 $\mathcal{F}$:

$$\rho\frac{\partial u}{\partial t} + \rho u \cdot \nabla u = \nabla \cdot \sigma_f + f \tag{1}$$

$$\nabla \cdot u = 0 \tag{2}$$

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 [**?**]. But this does not stop us from discretizing and solving them numerically.
Before these equations can be solved we need to impose boundary conditions.

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 $\mathbf{f}$

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

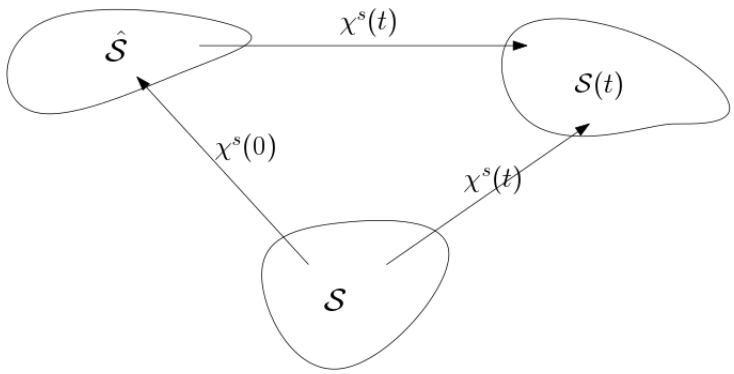
Solid Equations

In this chapter we will look briefly into the various parts that make up the solid equation. The solid equation is most commonly described in a Lagrangian description as it fits the solid problem. In a Lagrangian description the material particles are fixed with the grid points, this is a useful property when tracking the solid domain.

Reference domain

Mapping and identites

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:

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

Following the notation of [1],where $\mathbf{X}$ denote a material point in the reference domain and χ^s denotes the mapping from the reference configuration. $d^s(\mathbf{X}, t)$ denotes the displacement field and w($\mathbf{X}$,t) is the domain velocity. we set the mapping

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

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

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

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

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

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:

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

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 :

$$\begin{aligned} dv =& JdV \\ J =& \det(F) \end{aligned}$$

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:

$$dv = ds \cdot dx = JdSdX$$

This is used to get the Nanson‘s formula:

$$ds = JF^{-T}dS$$

which holds for an arbitrary line element in different configurations.

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. [Godboka]. 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.

Stress

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

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

Cauchys stress theorem states that if the traction vectors(force measured on surface area per unit) depends on $\mathbf{n}$ or $\mathbf{N}$ then they must be linear in $\mathbf{n}$ or $\mathbf{N}$. Giving:

$$\begin{aligned} t(x, t, n) =& \sigma_s(x, t)\mathbf{n} \\ T(x, t, n) =& P(X, t)\mathbf{N} \end{aligned}$$

where t is the traction vector and T is the first Piola-Kirchhoff traction vector. σ is the Cauchy stress tensor and P is the first Piola-Kirchhoff stress tensor. Using Nanson?s formula P can be written as:

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

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

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

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

$$P = FS$$

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 and theory from [Godboka]:

$$\rho_s \frac{\partial d^2}{\partial t^2} = \nabla \cdot (P) + \rho_s f \tag{3}$$

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]

Fluid Structure Interaction Problem in ALE

The ALE formulation stands for Arbitrary Lagrangian Eulerian. This entails formulating the fluid equations in an Eulerian and the solid in a Lagrangian framework. This is as previously stated a very nice approach as it describes both fluid velocity and displacement in known configurations. The ALE method can be solved by moving the mesh for each time step following the structure body movements, and calculating on a new"mesh every time. This approach gives advantages as we can explicitly represent the fluid-structure interface, but problems arise when there are large deformations in the solid giving large deformations to the fluid domain. We will therefore approach the ALE from a reference frame. That is we solve the equation on a initial, stress free domain, and use a series of mappings to account for the movements of the domain. From a technical point of view, both formulations are equivalent. But the ease of computing on fixed mesh does so that we chose fixed approach.
Since the domain is moving with the structure movements, we will need to state the fluid in a little different.

Notation

u - Velocity in fluid and structure.
w- Velocity in the domain. It is the velocity of the mesh in the calculations. This will also be the velocity in structure when defined in the Lagrangian formulation.
d - Displacement of the solid. The time derivative of the displacement will be the domain velocity.
p - Pressure in the fluid.
$\hat{\mathcal{S}}$ - Solid reference domain
$\mathcal{S}$ - Solid current domain
$\hat{\mathcal{F}}$ - Fluid reference domain
$\mathcal{F}$ - Fluid reference domain

Before we can state the full FSI problem we need to define the mappings to be used when computing from a reference frame.

Mapping

Let $\hat{\mathcal{V}}$ be a reference volume and $\mathcal{V}(t)$ be the current time volume using [Godboka]. Then using the determinant of the deformation gradient $F = I + \nabla d$, $J = det(F)$ we define a mapping between the volumes

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

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

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

Balance laws

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

Solid

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

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

Fluid

The fluid domain is moving, and therefore we need to redefine the velocity in the convective term in N-S to account for the moving domain

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

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 again following [godboka] using the forementioned mappings

$$\int_{\mathcal{V}(t)} \rho_f \frac{\partial u}{\partial t} dx = \int_{\hat{\mathcal{V}}} \rho_f J \frac{\partial u}{\partial t} dx \tag{4}$$

$$\int_{\mathcal{V}(t)} \nabla u (u - \frac{\partial d}{\partial t}) dx = \int_{\hat{\mathcal{V}}} ((\nabla u) F^{-1} (u - \frac{\partial d}{\partial t}) dx \tag{5}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot u dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (J F^{-1} u) dx \tag{6}$$

$$\int_{\mathcal{V}(t)} \nabla \cdot \sigma_f dx = \int_{\hat{\mathcal{V}}} \nabla \cdot (J F^{-1} \hat{\sigma_{\mathbf{f}}}) dx \tag{7}$$

$$\hat{\sigma_{\mathbf{f}}} = -pI + \mu(\nabla u F^{-1} + F^{-T} u^T) \tag{8}$$

Harmonic extension

To bind together the computation of fluid and structure domain, we need a harmonic extension to the boundary values. The solid deformation d is extended from the interface into the fluid domain and is done to help deal with big deformations in fluid domain. These big deformations can then cause several challenges to the ALE mapping. For this purpose define the following harmonic extension equation in the fluid domain:

$$\nabla^2 d^f = 0 \quad in \ \hat{\mathbf{F}}$$

This equation is chosen for its good regularity and smoothing properties.
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, that is of fourth order character, and thus will have a high computational cost. For now I will stick with the harmonic extension and maybe look at these in the future. [Godboka.]

Coupled Fluid Structure Interface conditions

This section is based on 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: $u_f = u_s \quad on \Gamma$. The fluid and structure velocities need to be equal on the boundary.
  We later realize this later by setting $u - \frac{\partial d}{\partial t} = 0$ 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.

Full FSI Problem in reference domain

Find $u \in \hat{\mathcal{F}}, p \in \hat{\mathcal{F}}$ and $d \in \hat{\mathcal{S}}$ such that :

$$\rho_f J \frac{\partial u}{\partial t} + (\nabla u) F^{-1}(u - \frac{\partial d}{\partial t}) + \nabla \cdot (J\hat{\sigma_{\mathbf{f}}} F^{-T}) = 0 \text{ on } \hat{\mathcal{F}} \tag{9}$$

$$\nabla \cdot (JuF^{-T})\big) = 0 \text{ on } \hat{\mathcal{F}} \tag{10}$$

$$\rho_s \frac{\partial u}{\partial t} + \nabla \cdot FS_s, = 0 \text{ on } \hat{\mathcal{S}} \tag{11}$$

$$\nabla^2 d = 0 \text{ on } \hat{\mathcal{F}} \tag{12}$$

$$u - \frac{\partial d}{\partial t} = 0 \text{ on } \hat{\mathcal{S}} \tag{13}$$

$$J\sigma_f F^{-T} n_f = \sigma_s n_s \text{ on } \Gamma \tag{14}$$

Finite Element method FSI in ALE

Variational formulation

Reference domain

We use 3 testfunctions, ϕ, ψ, γ. As mentioned before we use a global velocity function u for both the solid and fluid.

$$\rho_f J\big(\frac{\partial u}{\partial t} + (\nabla u) F^{-1}(u - \frac{\partial d}{\partial t}), \phi\big)_{\hat{\mathbf{F}}} + (J\sigma_f F^{-T}, \nabla\phi)_{\hat{\mathbf{F}}} = 0 \tag{15}$$

$$\big(\nabla \cdot (JuF^{-T}), \gamma\big)_{\hat{\mathbf{F}}} = 0 \tag{16}$$

$$\big(\rho_s \frac{\partial u}{\partial t}, \phi\big)_{\hat{\mathbf{S}}} + \big(FS_s, \nabla\phi\big)_{\hat{\mathbf{S}}} = 0 \tag{17}$$

$$\big(\nabla d, \nabla\psi\big)_{\hat{\mathbf{F}}} = 0 \tag{18}$$

$$\big(u - \frac{\partial d}{\partial t}, \psi\big)_{\hat{\mathbf{S}}} = 0 \tag{19}$$

Equation (5) has not been addressed and is added since we use a global function for velocity we need to force that the structure velocity is the time derivative of the deformation in the structure domain.

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.

Monolithic FSI Code

Here we will look at the implementation of the monolithic FSI Code in FENiCS.
All the mappings and stresstensors are made using functions, I will just show some to understand the code later on:

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

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

def sigma_f_new(u,p,d,mu_f):
        return -p*Identity(len(u)) + mu_f*(grad(u)*inv(F_(d)) + inv(F_(d)).T*grad(u).
```

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

```
F_fluid = (rho_f/k)*inner(J_(d_["n"])*(v_["n"] - v_["n-1"]), phi)*dx_f
F_fluid += rho_f*inner(J_(d_["n"])*grad(v_["n"])*inv(F_(d_["n"]))*(v_["n"] - (d_["n"]-
F_fluid += inner(J_(d_["n"])*sigma_f_new(v_["n"], p_["n"], d_["n"], mu_f)*inv(F_(d_["
F_fluid -= inner(div(J_(d_["n"])*inv(F_(d_["n"]))*v_["n"]), gamma)*dx_f
```

where dx_f is the fluid domain. Next is the solid variational form:

```
delta = 1E10
        F_solid = rho_s/k*inner(v_["n"] - v_["n-1"], phi)*dx_s

        F_solid += inner(Piola1(0.5*(d_["n"] + d_["n-1"]), lamda_s, mu_s), grad(phi))

        F_solid += delta*((1./k)*inner(d_["n"] - d_["n-1"],phi)*dx_s - inner(0.5*(v_[

        F_solid += inner(grad(d_["n"]), grad(phi))*dx_f + (1./k)*inner(d_["n"] - d_["
```

The condition $u = \frac{\partial d}{\partial t} in \mathcal{S}$, is weighted with a delta value. This is done since we compute everything together this is an important condition? vet ikke hva jeg skal skrive her... We have used a Crank-Nicholson scheme in the solid to preserve the energy, which will be discussed later in the Verification and Validation chapter.
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
        % (Iter, residual, atol, rel_res, rtol)
        Iter += 1
```

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
```

Introduction

Here we will look at the partitioned approach to solving the FSI problem. This means splitting our scheme into parts where we solve the fluid, structure and extension problem in different steps. This is to greatly reduce the size of the computational cost, and hopefully increase speed. So far the methods for coupling of the fluid and structure, has led to unconditional numerical instabilites and a large added-mass effect. Here we look at a new approach to explicit coupling, first proposed by Fernadez, which uses a Robin-Neumann explicit treatment of the interface first for thin walled structure [**?**] but later with an extension to thick walled structures[**?**]. This combined with a lumped mass approximation of the solid problem ensures added-mass free stability. [Generalized R-N explicit coupling schemes]

Robin-Neumann Interface

The Robin-Neumann treatment of the interface proposed by Fernandez uses a boundary operator $B_h : \Lambda_{\Sigma,h} \to \Lambda_{\Sigma,h}$ which is used together with the known coupling of stresses:

$$J^n \sigma^f(u^n, p^n)(F^n)^{-T} n^f + \frac{\rho^s}{\tau} B_h u^n = \frac{\rho^s}{\tau} B_h(\dot{d}^{n-1} + \tau \partial_t \dot{d}^*) - \Pi^* n^s$$

- The explicit treatment of the solid ensures uncoupling of the fluid and solid computations. Giving a genuine partitioned system.
- Treating the left hand side solid tensor implicitly ensures added-mass free stability

The fluid domain is computed using a generalized Robin condition on the interface, and the solid is computed with the familiar Neumann condition on the interface, equalling the stresses from the fluid and structure.
The general r-order extrapolation x^* is defined:

$$x^* = \begin{cases} 0, & \text{if } r = 0 \\ x^{n-1}, & \text{if } r = 1 \\ 2x^{n-1} - x^{n-2}, & \text{if } r = 2 \end{cases} \tag{20}$$

Boundary interface operator

Using the notation of [**?**] We denote $(.,.)_{\mathcal{S},h}$ as the lumped mass approximation of the inner product $(.,.)_{\mathcal{S}}$. We will consider a solid and fluid sided discrete lifting operator $\mathcal{L}_h^s : \Lambda_{\Sigma,h} \to \mathcal{S}$, lifting values from the interface into the solid domain. If $\xi_h, \lambda_h \in \Lambda_{\Sigma,h}$ then $\mathcal{L}_h^s|_\Sigma = \mathcal{L}_h^f|_\Sigma = \xi_h$. We use this to define the boundary operator: $B_h = (\mathcal{L}_h^s)^* \mathcal{L}_h^s$, mapping from interface to interface $B_h : \Lambda_{\Sigma,h} \to \Lambda_{\Sigma,h}$. Where stars stands for the adjoint operator of $\mathcal{L}_h^s$. We can then write:

$$(B_h \xi_h, \lambda_h)_\Sigma = (\mathcal{L}_h^s \xi_h, \mathcal{L}_h^s \lambda_h)_{\mathcal{S},h}$$

Explicit Robin-Neumann scheme:
Step 1: Fluid domain update

$$\begin{aligned} d^{f,n} &= Ext(d^{n-1}) \\ w^n &= \frac{\partial d^{f,n}}{\partial t} \\ with F &= I + \nabla d, J = det(F) \end{aligned}$$

Step 2: Fluid step: find u^n, p^n:

$$\begin{aligned} \rho^f \big(\frac{\partial u^n}{\partial t} + (u^{n-1} - w^n) \cdot \nabla u^n\big) - \nabla \cdot \sigma(u^n, p^n) &= 0 \in \mathcal{F} \\ \nabla \cdot u &= 0 \in \mathcal{F} \\ \sigma(u^n, p^n) n^f &= f \\ J^n \sigma(u^n, p^n)(F^n)^{-T} n^f + \frac{\rho^s}{\tau} B_h u^n &= \frac{\rho^s}{\tau} B_h(\dot{d}^{n-1} + \tau \partial_t \dot{d}) - \Pi^* n^s \end{aligned}$$

Step 3: Solid Step: find d^n

$$\begin{aligned} \rho^s \partial_t \dot{d}^n + \alpha \rho^s \dot{d}^n - \nabla \cdot \Pi^n &= 0 \in \mathcal{S} \\ \dot{d} &= \partial_t d^n \\ d^n = 0, \beta \dot{d}^n &= 0 \in \Gamma^d \\ \Pi^n n^s &= 0 \in \Gamma^n \\ \Pi^n n^s &= -J^n \sigma(u^n, p^n)(F^n)^{-T} n^f \in \Sigma \end{aligned}$$

The solid stress tensor is given as $\Pi^n = \pi(d^n) + \beta \pi^?(d^{n-1}) \dot{d}^n$

Kapittel 1

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 on a computer. We then use the solutions to extract data that will answer the questions of the problem we set out so solve. A problem then immediately arises, is this solution correct? To answer this we need to answer another question, is the problem defined correct mathematically, and if so are these equations solved correct numerically? Without answering these questions, being confident that your solutions are correct is difficult. [**?**] 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 or so called benchmark tests. 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. We will use a series of tests, in Verification we will use the Method of Manufactured solutions and in Validation we will use well known benchmarks.

1.1 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. In verifying the code, order of convergence tests will be the most rigorous. To do this test we will use the method of manufactured solutions (MMS) [**?**]. This method entails manufacturing an exact solution that is non trivial but analytical. The solution defines the boundary conditions and is passed through the equations giving a source term, named f. This source term is set to equalize the given equation, and then a solution is calculated. If the calculation is correct our calculated solution should equal the manufactured solution down to a give precision, computers are only precise to about 10^{-16}. We can then increase for instance the number of cells in our computational domain, and see if the difference between the manufactured and computed solution (eg. error) gets smaller. The rate at which the error reduces can be checked with mathematical theory, we can than be more confident that our computation is correct. This will also be done in time, by reducing the time steps and investigating the error. The manufactured solution does not have to have any physical relation, and this fact does not implicate a less accurate verification. The solution needs only be non-trivial.
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 or decrease timestep, we expect the error to get smaller. Its the rate of this error that lets us now wether the

solution is converging correctly and hence that we are computing in the right fashion. If we assume that the number of spatial points are equal in all directions we know that the error behaves like

$$E = C_1 \Delta x^k + C_2 \Delta t^l$$

where $k = m + 1$ and m is the polynomial degree of the spatial elements. This means that when we compute with Taylor-Hood elements (P2-P1) we should expect to get a convergence rate of 2 in space and 1 in time. The convergence rates are computed as:

$$\frac{E_{n+1}}{E_n} = \big(\frac{\Delta x_n + 1}{\Delta x_n}\big)^k \tag{1.1}$$

$$k = \frac{log(\frac{E_{n+1}}{E_n})}{log(\frac{\Delta x_n + 1}{\Delta x_n})} \tag{1.2}$$

1.1.1 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:

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

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 1.1: 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

1.1.2 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:

$$\left(\rho_s \frac{\partial u}{\partial t}, \phi\right)_{\hat{\mathbf{S}}} + \left(P, \nabla\phi\right)_{\hat{\mathbf{S}}} = f_s \tag{1.3}$$

$$\left(u - \frac{\partial d}{\partial t}, \psi\right)_{\hat{\mathbf{S}}} = 0 \tag{1.4}$$

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

$$d = (cos(y)sin(t), cos(x)sin(t))$$
$$u = (cos(y)cos(t), cos(x)cos(t))$$

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

Tabell 1.2: 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

Time:

Tabell 1.3: 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

1.2 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 [**?**]. A model is made for a specific purpose, its only valid in respect to that purpose [**?**]. 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 [**?**]: 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 [**?**] 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. [**?**]
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.

1.2.1 Taylor-Green vortex

The Taylor-Green vortex problem is used to examine if our N-S code has the ability to correctly simulate vortex decay and turbulence [**?**]. TGV is a non-trivial analytical solution to N-S. We will compute and compare the kinetic energy dissipation rate against a well known very good solver (smisk smisk). This will help us determine the solvers ability to handle turbulence.

Problem definition

Using a cube with sides 2π.
We have an initial distribution of velocity $\bar{u} = (u, v, w)$:

$$u(x, y, z) = V_0 sin(x)cos(y)cos(z) \tag{1.5}$$
$$v(x, y, z) = -V_0 cos(x)sin(y)cos(z) \tag{1.6}$$
$$w(x, y, z) = 0 \tag{1.7}$$

The Reynolds number is defined as: $Re = \frac{V_0 L}{\nu}$ where we set $V_0 = 1$

Something about calculating data

The method used to evaluate the TGV solutions will be by investigating the rate of which the fluid dissipates the kinetic energi. The kinetic energi will be calculated as:

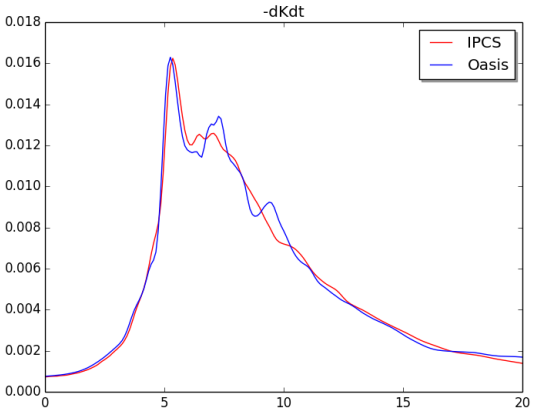
$$E_k = \frac{1}{\rho_0 \Omega} \int \rho \frac{uu}{2} = 0.5 \frac{u^2}{(2pi)^3}$$

We use this to calculated the dissipation rate by differentiating E_k w.r.t time:

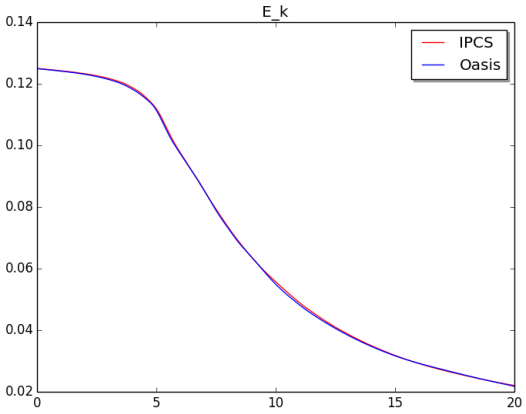
$$\epsilon(E_k) = -\frac{dE_k}{dt}$$

Results

To help validate the Taylor-Green solutions we used the Oasis solver [**?**] which is known to handle TGV and other turbulent flows. We look at a plot of the dissipation rate over time and the kinetic energi compared with Oasis. These test were run with: N = 32, $\Delta t = 0.001$, $\nu = 0.001$ giving Re = 1000:



Figur 1.1: $\epsilon(E_k)$ N = 32



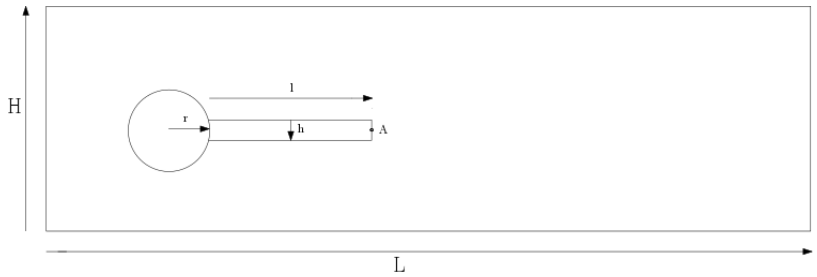
Figur 1.2: E_k N = 32

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 [**?**]. This benchmark is based on the older benchmark flow around cylinder with 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.

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 [?] 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)
```

1.2.2 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 1.4: 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 1.5: 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 1.6: 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 1.7: 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 1.8: 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 1.9: 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 1.10: CSM 3

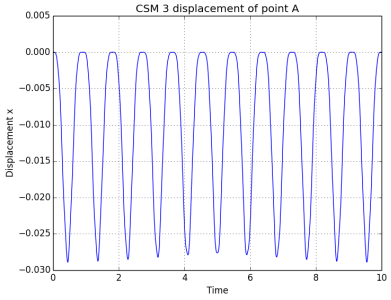
elements	dofs	ux $[10^3]$	uy $[10^3]$
725	1756	-11.743 ± 11.744	-57.952 ± 58.940
2900	6408	-13.558 ± 13.559	-61.968 ± 63.440
11600	24412	-14.128 ± 14.127	-63.216 ± 64.744
46400	95220	-14.182 ± 14.181	-63.305 ± 64.843
ref		-14.305 ± 14.305	-63.607 ± 65.160

FSI test

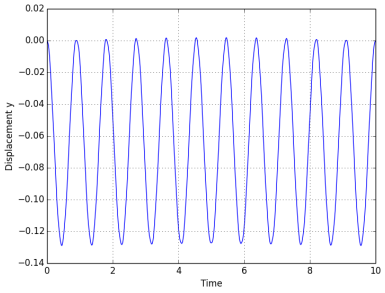
Tabell 1.11: FSI Parameters

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
$\mathrm{Re} = \frac{Ud}{\nu_f}$	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

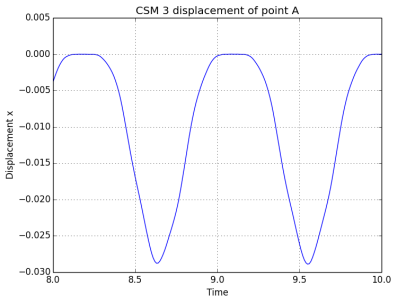
Results:



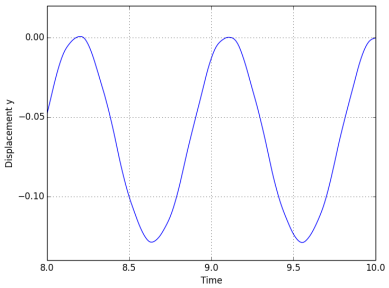
(a) Displacement in x-direction



(b) Displacement in x-direction



(c) Displacement in x-direction



(d) Displacement in x-direction

Figur 1.3: Displacement of point A

Tabell 1.12: 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**

OLD SHITZ FSI:

Tabell 1.13: 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	0.02271	0.80288	14.1736	0.787891	P2-P2-P1
2698	23563	0.00581116	0.000000738678	12.07	0.02345	P2-P2-P1 without weighting
10792	92992	0.0227341	0.808792	14.1855	0.801044	P2-P2-P1
43168	369448	0.227352	0.812595	14.227	0.797242	P2-P2-P1
ref	**ref**	**0.0227**	**0.8209**	**14.295**	**0.7638**	**ref**

Bibliografi

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