A Unified Continuum Fluid-Structure Interaction Solver using an ALE Finite Element Method

An investigation on how to simulate blood flow

M I C H A E L   S T Ö C K L I

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2007

A Unified Continuum Fluid-Structure Interaction Solver using an ALE Finite Element Method

An investigation on how to simulate blood flow

M I C H A E L   S T Ö C K L I

Master's Thesis in Numerical Analysis (20 credits)
at the Scientific Computing International Master Program
Royal Institute of Technology year 2007
Supervisor at CSC was Johan Hoffman
Examiner was Axel Ruhe

TRITA-CSC-E 2007:010
ISRN-KTH/CSC/E--07/010--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

The goal of this thesis was to implement a Fluid-Structure Interaction solver using the Finite Element Method in an attempt to model blood flow. In comparison to other FSI solvers, this solver is singular and not split into a fluid solver and a solid solver, thus there is no communication taking place across the boundaries. In fact, the structure and the fluid are present in the same mesh. The resulting solver happens to be stable and easy to configure as well as flexible towards future applications.

Sammanfattning

Vi implementerar en lösare för fluid-struktur interaktion (FSI) som använder finite element metoden för att simulera en modell för blod flöde. Jämfört med andra FSI lösare är vår inte uppdelad i en solid och en fluid lösare som kommunicerar genom randvilkor, utan vi löser istället en ekvation som är formulerad för det kombinerade fluid-struktur området, diskretiserad på ett och samma nät.

Contents

Chapter 1

Introduction

This thesis is an extension to the research that has been going on for quite some time. It is the research of the flow of a fluid, be it liquid or gaseous, interacting with a structure. This is a very general description of the research field which reflects the reality quite well, it is diverse. This field as explained in the previous few sentences encompasses many problems that we would like to be able to solve numerically. Due to the nature of these problems, the fact that we have a fluid interacting with a structure, these problems have been categorized as FSI problems, Fluid Structure Interaction. FSI problems are interesting because a symbiotic relationship exists between the two materials, the properties of the fluid affect the properties of the structure and therefore the structure undergoes change as the fluid changes, and vice versa. Such a symbiotic relationship can be quite difficult to model for such reasons as having to account for differing governing equations, one for the structure and another for the fluid. This means we may, as an example have a Navier-Stokes flow interacting with an elastic membrane whose governing law is described by Lagrange coordinates, that acts as the container for the fluid. A perhaps rather simple problem at first glance, but proves to be quite a bit more intricate once we consider the necessary steps that must be implemented to give reliable answers. One thing we would have to consider is of course the interaction between the two materials. We cannot simply solve for the fluid and then for the structure without passing information between the two entities, this would be a huge mistake since they affect each other, thus the flow of information between the two is a two way street.

Before going further into the research that has already been conducted, I will give a few examples of FSI problems and also explain what we are considering in this thesis. This is to help the reader become aware of the differing ways we can have FSI problems, one being very different from the other yet sharing the same category. Firstly, I would like to introduce the reader to the main motivation of this thesis, that is the modeling of blood flow through a vessel; The blood, a rather complex non-Newtonian fluid in itself, flows within the vessels of almost all living breathing organisms. Here we have a fluid, namely blood, interacting with its container, the vessel walls which we consider as the structure. Both the vessel walls and blood are rather complex materials, the vessel walls usually consisting of several layers of material layered on top of each other much like the pages of a book, and blood which is made up of several different agents such as blood cells. All has been simplified to be able to construct something easily, hence our fluid is Newtonian and our vessel walls consist only of a single layer.

A different example of a FSI problem is the way that paper - a structure, interacts with the surrounding air, the fluid, when it undergoes some kind of transport operation. This is a real problem that is also undergoing research in numerical analysis which can then be applied to the paper making industry. Our problem on the other hand, applies to physiology and thus can be of significance in the health sector. As one can tell, two very different problems that share the same category, the main difference being that the problems are inverses of each other, in our problem, the fluid is contained by the structure, while in the second FSI problem the structure is flowing inside the fluid domain.

"What's the use of it Mr. Faraday" a question raised by Mr. Gladstone, the chancellor of the exchequer when Faraday presented his findings on electricity and electromagnetics. It seemed like fun magic with no real application. So, what is the use of being able to model blood flow? The answer to that is that it aides our understanding of how blood flow related problems arise in humans and animals. It hopes to shed light on the complex mechanics that affect the living physiology and thus being able to detect problems and carry out investigations before the need for evasive action is required. For a more thorough description of the applications of blood flow simulation, please read [1]. As with most models, others can take it and extend upon it to fit just their needs and use it for all kinds of flow investigations, not just limited to physiology. Faradays answer to the question was a much simpler "Why, sir, there's every probability that you'll soon be able to tax it." [1]

1.1 Previous work on FSI and blood flow simulation

One of the people interested in the research of blood flow simulation is Alfio Quarteroni who has published [1]. With the article he sets a foot print for the research of the simulation of blood and explains the possibilities of using mathematics to create models of the cardiovascular system. In short, it offers a rather comprehensive introduction into the world of FSI and blood flow simulation. Many difficulties are present that have to be overcome by means of simplifying without losing considerable physical behaviour, otherwise the model becomes pointless. Difficulties that arise are for instance the fact that the vessels do not only provide a transport infrastructure but also aid in the propagation of the blood due to the small contractions and relaxations caused by muscle fibre that surrounds the vessel.

Quite a lot of research has been done in the area of FSI with respect to blood flow. Most articles discuss an approach which chooses to use separate solvers for the structure and the fluid [3] [4] [5] [6]. This is a rather good idea since advances in either solvers can be directly applied to the FSI solver without additional work. However, there is

[1]Please note that our work is published under the GPL license.

a major drawback with this idea, and that is the fact that one has to somehow link the two solvers using boundary conditions, otherwise the solvers wouldn't know about each other and solve independent problems on their own. A clear example of how the boundary conditions are constructed is given in [10]. The need for boundary conditions to pass information between the two solvers brings forth a lot of extra work and can cause instability and problems with convergence. To counteract the convergence problem, [3] suggests the use of artificial compressibility near the fluid-structure interface. As a result, the method showed good convergence in strongly coupled cases but small time-steps were harder to resolve.

As mentioned above, our main motivation for constructing a FSI solver is to be able to apply it to blood flow problems. Blood flow is something that is rather complex, not just because of the many different layers that form the vessel walls and the non-Newtonian fluid that is blood, but also because blood flows in a system, the cardiovascular system. By focusing on just a small segment of the system, such as a single vessel of undefined length, we truncate a large extent of the information that that affects the simulation. Suggestions have been made that one could simulate the system using terminology from electronics [1]. Another problem that occurs when we choose to only focus on a small aspect is that we end up "clamping" the vessel at both ends when prescribing boundary conditions. This causes waves to reflect off the boundary and interact with our solution thus falsifying the results. This was found to be the case for [6], but the main goals of that article was to examine different meshing techniques and simulation of large deformations using a Jacobian-free Newton Krylov method. [4] shows how this problem can be minimized by letting the waves be absorbed by a 1D model thus effectively cancelling the reflected waves from the boundary.

Our method is different in a very significant way, instead of using separate solid fluid solvers and then coupling them using boundary conditions, we chose to merge the two governing equations into one, resulting in just a single solver that can handle both solids and fluids. This makes boundary conditions at the interface redundant and allows for great flexibility for future work, such as the solidification of a fluid or melting of the solid structure should be able to be handled quite easily.

1.2 Previous work on stabilized ALE-FEM

Since we are dealing with convection when we use the Navier-Stokes formulation, we have to find some way to stabilize the method due to the FE-method having some problems with convection dominated problems. In both [7] and [8] do we read about the G2 method, which is a weighted least-squares stabilized finite element method. Sometimes it is also referred to as the streamline diffusion method.

Even though turbulence is not a major concern for the calculation of blood flow, we should still take it into consideration. To model the mean field of turbulent flows, we can use the G2 method, with sufficient resolution, see [7].

Further detail on how the method is implemented will be given in the later moments of this writing.

Chapter 2

Fluid-Structure Interaction

2.1 Motion: Eulerian vs. Lagrangian description

The differences between the Eulerian world and the Lagrangian world are not significantly large. All that separates the two descriptions is basically the position of the observer. In the Lagrange description, also called the material coordinate system, we tend to attach ourselves to the material and move as the material moves recording any changes of just that same piece of material, considered a particle. As an example, we have an elastic ruler whereby we consider the velocity and stress of the ruler as we bend the right end downwards whilst keeping the other side fixed. For this example, we shall only consider one single point on the ruler as our observation point, let's say that this point is located in the middle of the right half of the ruler. Now, as we bend the ruler downwards slowly on the right side, our point of observation stays attached to that point and thus moves downwards with the ruler. At the same time, we record the velocity of this point as the ruler bends as well as the stress that the material of the ruler is undergoing. In this case, the material is a solid and has properties which make it favourable for using a Lagrangian description, the fact that it does not undergo excessive distortion. Had the material been a fluid, like the flow of water in a river bed, then the point of observation would have travelled along in the stream exhibiting a rather erratic behaviour which, in simulations, causes ill-shaped meshes and makes finding a solution a rather difficult task. For flow problems, it is in general favourable to use the Euler description.

For the Euler description, our point of observation is no longer fixed to the material, but fixed in space. To use the example of water flowing in a river bed again, using the Eulerian description means that we fix our point of observation not to the water, in this case better described as a water particle, but instead we fix our point in space, somewhere above the water flow, like a small boat anchored to a fixed point on the river. At this point, we record the material (water) velocity and temperature, purely to give an example. Note that we no longer record the properties of just one single material particle, but all of the stream which flows beneath our anchor point.

What follows is a more formal discussion taken from [2] describing the differences.

The notation has intentionally not been changed to that from [2] so that the reader who wants to indulge into the reference, can do so without experiencing a change in notation.

For the Lagrangian material description, we say that we have a domain $R_X \subset \mathbb{R}^d$ where d is the spatial dimension made up of material particles X. As mentioned above, we follow the particles, thus we attach our viewpoint to the material. In a more concrete manner, we say that we map a grid over the material whereby our degrees of freedom (dof's) are attached to the material particles. More formally, we follow the continuum in its motion. Conversely, the Eulerian configuration, we say that R_x is the spatial domain consisting of spatial points x. When deformation of the continuum occurs, the changes can be mapped to the spatial coordinates, written as a function φ which takes into account time t.

$$\varphi : R_X \times [t_0, t_{final}[\longrightarrow R_x \times [t_0, t_{final}[\tag{2.1}$$

or more simply $\varphi(X,t) = (x,t)$ and so we can write x as a function of time and material coordinate $x = x(X,t)$. In any case, we have a mapping φ which maps a particles position in R_X to a spatial position in R_x as the continuum moves over time t. And we can also do the opposite, namely $\varphi^{-1}(x,t) = (X,t)$.

For future purposes, it is useful to define the gradient of φ together with the material velocity of a particle X.

$$\frac{\partial \varphi}{\partial(X,t)} = \begin{pmatrix} \dfrac{\partial x}{\partial X} & v \\ 0^\top & 1 \end{pmatrix} \tag{2.2}$$

where v is the given velocity

$$v(X,t) = \left.\frac{\partial x}{\partial t}\right|_X$$

2.2 ALE explained

The ALE description is a culmination between the Lagrangian and Eulerian descriptions, much like its full name suggests; Arbitrary-Lagrangian Eulerian. ALE, in a sense, can be thought of as a bridge between the two descriptions. It provides a means for us to do calculations using both Lagrangian and Eulerian descriptions with the use of a third coordinate system, the reference coordinate system. In a more practical sense, it allows us to move our domain at a different velocity compared to our observed material without skewing our solution. This is because we can compensate for the movement of the domain. In practice this is the mesh we use when we calculate our solutions for the fluid domain where we would otherwise use a Eulerian description. This allows us to use the Lagrangian description for the structure part of our FSI solver so that we can use standard elasticity models. At the same time, we can use the ALE description, also referred to as Quasi-Eulerian for our fluid model.

For a more thorough understanding, I will once again refer to [2]. As mentioned above, we make use of a third coordinate system, the reference coordinate configuration denoted R_χ with reference points χ. Much like we used a mapping φ to get from the material configuration to the spatial configuration, we have two additional mappings Φ and Ψ which map from the referential configuration to the spatial and material configurations respectively. An illustration is given in Fig.2.1.

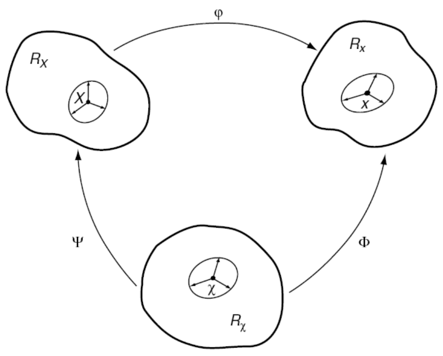


Figure 2.1. Illustration of the different configurations. Taken from [2].

From the illustration, we can easily derive some relationships between the different configurations such as $\varphi = \Phi \circ \Psi^{-1}$ which describes particle motion. The mappings are similar to (2.1)

$$\Phi(\chi, t) = (x, t) \tag{2.3}$$

$$\Psi^{-1}(X, t) = (\chi, t) \tag{2.4}$$

and also the respective gradients in matrix form

$$\frac{\partial \Phi}{\partial(\chi, t)} = \begin{pmatrix} \frac{\partial x}{\partial \chi} & \hat{v} \\ 0^\top & 1 \end{pmatrix} \tag{2.5}$$

$$\frac{\partial \Psi^{-1}}{\partial(X, t)} = \begin{pmatrix} \frac{\partial \chi}{\partial X} & w \\ 0^\top & 1 \end{pmatrix} \tag{2.6}$$

where

$$\hat{v}(\chi, t) = \left.\frac{\partial x}{\partial t}\right|_\chi$$

$$w(X, t) = \left.\frac{\partial \chi}{\partial t}\right|_X$$

the reference point velocity and the material particle velocity with respect to the reference system holding the reference point and the material particle fixed, respectively.

We can relate the different velocities by using the matrix forms of the various gradients from (2.2), (2.5) and (2.6).

$$\begin{pmatrix} \frac{\partial x}{\partial X} & v \\ 0^\intercal & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial \chi} & \hat{v} \\ 0^\intercal & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial \chi}{\partial X} & w \\ 0^\intercal & 1 \end{pmatrix} \tag{2.7}$$

After rearranging, we are left with

$$c = v - \hat{v} = \frac{\partial x}{\partial \chi} w \tag{2.8}$$

where c is the particle velocity relative to the mesh as seen from the spatial domain R_x, in other words, the actual velocity of a fluid after compensating for the movement of the mesh. $\hat{v}$ is the movement of the mesh and by subtracting it from v, we compensate for this manually induced deformation, which here, is caused by the mesh smoothing algorithm discussed below.

2.3 Mesh smoothing

In a FSI problem, we have a mesh that is divided into structure and fluid parts. For the structure part, we use the Lagrange description for calculations, this means that we move the mesh according to the solution of the calculations. The fluid and its behaviour however, are calculated using the ALE description where there is no mesh movement if we consider a pure fluid computation except for the movement specified by boundary operations. When we use a domain consisting of both fluid and solid domains, the mesh which belongs to the solid will move causing the mesh at the interface between the solid and the fluid to become ill conditioned. We are then required to redistribute the vertices of the mesh belonging to the fluid domain. This is done using the Laplace mesh smoothing technique.

The technique works by moving all the internal vertices to the centre of all its neighbours. A neighbour of a vertex can be called a neighbour when the two vertices are connected via an edge. Usually, in the 2D case, the number of connected neighbours is six. We can use a simple algorithm to smooth the mesh as written below.

$$v_i = (\sum_{j \neq i} v_j)/n \tag{2.9}$$

where

v_i is an internal vertex
v_j is a neighbouring vertex
n is the number of neighbours excluding oneself.

As one can see, we simply calculate the average location of all the neighbouring vertices, this gives us the centre of mass of the system of neighbouring vertices and thus

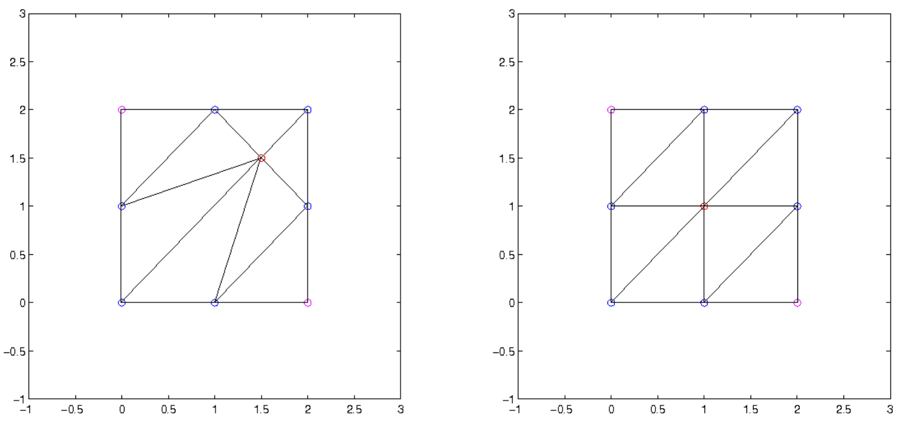


Figure 2.2. The plot on the left is before mesh smoothing, the plot on the right is after mesh smoothing. The blue vertices are neighbours to the red vertex which is being smoothed.

the mid-point for our current vertex, this can be seen in Fig.2.2. Since we move the vertices in our domain, we have to let our governing equation know about the change. We do this by communicating the changes via our ALE description. This is done by formulating the change in the position of the individual vertices in terms of the mesh velocity $\hat{v}$. Thus we have

$$\hat{v} = (x_{new} - x_{old})/k \tag{2.10}$$

where

x_{new}	is the new location of the vertex
x_{old}	is the previous location of the vertex before mesh smoothing
$\hat{v}$	is the mesh velocity
k	is the time step size.

Feeding $\hat{v}$ back into our ALE FSI solver lets the governing equation know about the movement which the domain underwent resulting in the correct simulation of the FSI problem.

2.4 Navier-Stokes in ALE

If we take a moment to look at the incompressible Navier-Stokes equation, we can easily spot that there is one convective term in the equations.

$$\dot{u} + (u \cdot \nabla)u - \nu\Delta u + \nabla p = f \tag{2.11}$$

$$\nabla \cdot u = 0 \tag{2.12}$$

Namely $(u \cdot \nabla)u$. What we have learned above in the previous paragraphs is that if we want to be able to calculate a reliable solution whilst moving the mesh, we have to implement the ALE description. As we have also learned above, this can quite easily be done by just taking into account the mesh movement velocity $\hat{v}$ alongside the existing velocity v from above, in this case u. To illustrate this small change, here are the new ALE friendly incompressible Navier-Stokes equations.

$$\dot{u} + ((u - \hat{v}) \cdot \nabla)u - \nu \Delta u + \nabla p = f \tag{2.13}$$

$$\nabla \cdot u = 0 \tag{2.14}$$

Another change that takes place is the definition of the boundary conditions. Since the mesh is moving all the time, there is no real way of defining the boundaries using spatial coordinates. Instead we choose to use the reference coordinate points since they do not change as $t \to \infty$ from a mesh point of view.

For this to be the case, we choose the reference domain Ω^R to be the spatial domain at time $t = 0$, where the system is at rest. This gives us a further relationship, namely that $\chi = x \in \Omega_{t=0}$ meaning that the reference coordinate points can be identified by the spatial coordinate points at $t = 0$. This is useful for us because it allows the programmer to specify the boundary conditions at $t = 0$ on the reference domain and thus they will be valid for $t \geq 0$

2.5 Elastic in Lagrange

For the computation of the structure, we use an elastic model. The equations that govern the state of our solid body are presented in Lagrangian coordinates, and thus the mesh is fixed to the body.

Much like we described in the previous section outlining the differences between Lagrangian and Eulerian descriptions, here again we use a mapping from the Lagrangian space to the Eulerian space.

$$x = \varphi(t, X) \tag{2.15}$$

$$\Omega(t) = \{x(t, X) : X \in \Omega^0\} \tag{2.16}$$

The above prescribes a mapping from X the material particle, to the spatial coordinate x. Furthermore, we describe the domain Ω where our solid lives within and that the initial configuration Ω^0 holds the material particles. Remember that we use the Lagrangian coordinate system to do our calculations and then map these to the spatial system. Formulating the equilibrium equation

$$\rho \dot{u}(t, x) - \nabla \cdot \sigma(t, x) = f(t, x) \quad \text{for } x \in \Omega(t) \tag{2.17}$$

where

$x = \varphi(t, X), X \in \Omega^0$	is the mapping of the particle
σ	is the stress tensor
u	is the velocity
ρ	is the density
p	is the pressure

Stress plays a large role in the calculation of the deformation of an elastic body. There exists a linear relationship between stress and strain, they relate via Hooke's law.

$$\sigma = E(\epsilon) \tag{2.18}$$
$$E(e) = 2\mu e + \lambda e_{kk} I \tag{2.19}$$
$$\mu = \frac{Y}{2(1+\eta)} \tag{2.20}$$
$$\lambda = \frac{Y\eta}{(1+\eta)(1-2\eta)} \tag{2.21}$$

where

μ and λ	are Lame's parameters
ϵ	is the strain
Y	is Young's modulus
η	is Poisson's ratio

Strain can be formulated in two ways, in terms of the displacement from the initial body at rest or in terms of the velocity. We also have two forms of stress, viscous and elastic. The two different stresses are combined to give the final true stress (Cauchy stress). Using the strain formulated using the displacement gives us a much more general strain tensor accounting for any type of deformation (2.23), whilst using velocity assumes small deformations (simple, linear) (2.22).

$$\epsilon_s = \frac{1}{2}(\nabla u + (\nabla u)^\top) \tag{2.22}$$
$$\epsilon_l = \frac{1}{2}(I - (FF^\top)^{-1}) \tag{2.23}$$

where F is the Jacobian matrix representing (2.2). We can see that (2.22) is also used below in (2.27) for the strain tensor for the Navier-Stokes equations. A relationship between the two strains appears when we try to find the stress rate tensor by taking the material time derivative of ϵ_l.

$$\dot{\sigma} = E(\dot{\epsilon_l}) \tag{2.24}$$
$$\dot{\epsilon_l} = \epsilon_s - \sigma\nabla u - \nabla u^\top \sigma \approx \epsilon_s \tag{2.25}$$
$$\dot{\sigma} = E(\dot{\epsilon_l}) \approx E(\epsilon_s) \tag{2.26}$$

The relation (2.26) holds for deformations without large rotations. This relationship proves to be quite useful when we formulate our FSI model.

2.6 Unified Continuum FSI

The changes required to merge the two models together are rather few and involve separating the stress into a deviatoric and a hydrostatic part. This allows us to isolate the deviatoric which happens to be calculated differently depending on the medium and at the same time allows the pressure to be shared between both materials. For the fluid, this is already done, all we do is identify how the deviatoric is being calculated.

Fluid: Newtonian

$$\sigma_D = 2\nu\epsilon(u) \tag{2.27}$$

$$\epsilon(u) = \frac{1}{2}(\nabla u + (\nabla u)^\top) \tag{2.28}$$

Note that $\nabla \cdot \epsilon(u) = \Delta u$ which is what we have in (2.11). The solid, on the other hand, requires a bit more massaging until the deviatoric comes to light. When it does, it will be in the form of a time derivative that needs to be integrated for the stress deviatoric to available.

Solid: Linear elasticity

$$\sigma = E(\epsilon_l) \tag{2.29}$$

$$\dot\sigma = E(\dot{\epsilon_l}) \approx E(\epsilon_s) \tag{2.30}$$

$$\dot\sigma = \underbrace{2\mu\epsilon_s}_{\dot\sigma_D} + \underbrace{\lambda tr(\epsilon_s)}_{-p} I \tag{2.31}$$

$$\sigma_D = \int 2\mu\epsilon_s dt = \int \dot\sigma_D dt \tag{2.32}$$

This leaves the pressure p. In the Navier-Stokes equation, the pressure is calculated as a result of the continuity formulation. This is also possible for the solid.

$$\sigma_v = E_v(\epsilon_s) \quad \text{viscous stress} \tag{2.33}$$

$$\sigma_e = E_e(\epsilon_l) \quad \text{elastic stress} \tag{2.34}$$

$$\sigma = \sigma_e + \sigma_v \tag{2.35}$$

$$= 2(\mu_v\epsilon_s + \mu_e\epsilon_l) \qquad \longrightarrow \sigma_D \tag{2.36}$$

$$+ (\lambda_v tr(\epsilon_s) + \lambda_e tr(\epsilon_l))I \qquad \longrightarrow -pI \tag{2.37}$$

$$\frac{p}{-\lambda_v} = tr(\epsilon_s) - \frac{\lambda_e tr(\epsilon_l)}{-\lambda_v} \tag{2.38}$$

We would like to preserve the volume of our elastic body. To do this, we say that η is close to 0.5. This leads to λ_v becoming rather large such that we can approximate terms that are divided by λ_v as 0.

$$tr(\epsilon_s) = \nabla \cdot u = 0 \tag{2.39}$$

Thus we have managed to isolate the stress deviatoric for both the fluid and the solid. We shall from here on use the following notation to avoid confusion.

$$\begin{aligned}
\epsilon &= \frac{1}{2}(\nabla u + (\nabla u)^\top) \\
\sigma_D^F &= 2\nu\epsilon(u) \\
\sigma_D^S &= \int 2\mu\epsilon dt = \int \dot{\sigma}_D^S dt
\end{aligned}$$

Other shared functions will distinguish between solid and fluid membership by their superscript, ^{S} is solid while ^{F} is fluid. The only addition required is the introduction of a marker function that tells us whether we are dealing with a fluid or a solid. Let us call this function ϕ.

$$\phi = \begin{cases} 1 & \text{on } \Omega^F\text{, the fluid domain} \\ 0 & \text{on } \Omega^S\text{, the solid structure domain} \end{cases} \tag{2.40}$$

So far, we have only seen the Navier-Stokes equation where density is implicitly present due to the fluids incompressibility, since we have used the assumption that density is constant. Since the solid and the fluid will be of different densities, it is time to reintroduce the function ρ in our equations. Below are the Navier-Stokes equations using density and with a separated stress factor and ALE.

$$\rho^F(\dot{u} + ((u - \hat{v}) \cdot \nabla)u) + \nabla p - \nabla \cdot \sigma_D^F = f \tag{2.41}$$
$$\nabla \cdot u = 0 \tag{2.42}$$

For comparison, the elastic model

$$\rho^S \dot{u} + \nabla p - \nabla \cdot \sigma_D^S = f \tag{2.43}$$
$$\nabla \cdot u = 0 \tag{2.44}$$
$$\dot{\sigma}_D^S = 2\mu\epsilon \tag{2.45}$$

It's easy to see that these two equations are very similar and it makes it rather obvious that when we want to calculate a solution, we should be able to combine these two models into one. By setting $\hat{v} = u$ allows us to work in the Lagrange coordinate system and thus another difference between the two equations falls away since $((u - \hat{v}) \cdot \nabla)u) = 0$. Finally all we have to do is use ϕ to help us distinguish which form of the stress deviatoric we are supposed to calculate with. Once that is done, we have a single formulation for a unified continuum for solving fluid-structure interaction problems. The resulting equations are shown below.

$$\rho(\dot{u} + ((u - \hat{v}) \cdot \nabla)u) + \nabla p - \nabla \cdot \gamma = f \tag{2.46}$$
$$\nabla \cdot u = 0 \tag{2.47}$$

$$\gamma = \phi\sigma_D^F + (1-\phi)\sigma_D^S \tag{2.48}$$
$$\rho = \phi\rho^F + (1-\phi)\rho^S \tag{2.49}$$
$$\sigma_D^S = \int \dot{\sigma}_D^S dt = \int 2\mu\epsilon dt \tag{2.50}$$

2.7 Discussion of model to blood flow

As mentioned above, blood is not a simple Newtonian fluid, so why choose to represent the fluid as an incompressible Navier-Stokes flow. The answer lies in the context of the problem. Blood is suspension of particles in a fluid mainly made up of water (90-92%) which is an incompressible fluid. For calculations where the diameter of the vessel is large, it is sufficient to use this kind of model. This presents a good platform for continuing the development of the method as we, in the future, shall take smaller diameters into consideration. It is at these small diameters that the effect of the bloods contents, such as red blood cells, affect the overall behaviour of the flow. So for now we have limited us to flows where the effect of the bloods non-Newtonian properties are indistinguishable from that of a Newtonian fluid.

Our elastic model provides us with the ability to have vessel walls that stretch when the pressure of the fluid passing through the body is high, and relax for when the pressure is low. See [9] for a discussion on what models have been used along with their advantages and disadvantages.

Chapter 3

FEM for FSI

3.1 Stabilized FEM: G2 method

As mentioned in the introduction, the finite element method is not good at handling convection dominated problems and therefore requires some kind of stabilization technique. The method used for this implementation is the Generalized Galerkin method (G2), which can also be used to compute turbulent flow, see [7].

3.2 Model problem

We started out with the Navier-Stokes equations which are very close to the resulting FSI equations formulated earlier (2.46, 2.47, 2.48, 2.49, 2.50). To illustrate the effects of the stabilization method, we shall examine the variational formulation of NS equations with stabilizing terms and corresponding parameters δ_1 and δ_2.

$$\begin{aligned}&(\dot{u} + u \cdot \nabla u, v) - (p, \nabla \cdot v) + (\nu \nabla u, \nabla v) + (\nabla \cdot u, q)\\&+ (\delta_1(u \cdot \nabla u + \nabla p), u \cdot \nabla v + \nabla q)\\&+ (\delta_2 \nabla \cdot u, \nabla \cdot v) = 0\end{aligned}$$

given that $f = 0$ meaning we do not feed any energy into the system, as in this case, as well as $\nabla \cdot u = 0$ as we are dealing with an incompressible fluid.

For the model problem, we set $v = u, q = p$, meaning that our test functions are equal to the actual functions, velocity and pressure. The result of this is shown below.

$$(\dot{u}, u) + (u \cdot \nabla u, u) + ||\sqrt{\nu}\nabla u||^2 + ||\sqrt{\delta_1}(u \cdot \nabla u + \nabla p)||^2 + ||\sqrt{\delta_2}\nabla \cdot u||^2 = 0$$

We can set $(u \cdot \nabla u, u) = 0$ since $\nabla \cdot u = 0$[1] assuming the existence of a pointwise solution to the Navier-Stokes equation, and also

$$(\dot{u}, u) = \frac{1}{2}\frac{d}{dt}||u||^2$$

[1] $\int_\Omega (u \cdot \nabla)u \cdot u)dx = -\int_\Omega (u \cdot \nabla)u \cdot u)dx - \int_\Omega \nabla \cdot u|u|^2 dx$ for vanishing boundary terms, this means that the integral on the lhs is equal to its negative. There is only one way for this to be possible and that is if $\int_\Omega (u \cdot \nabla)u \cdot u)dx = 0$

so our equation now looks like

$$\frac{1}{2}\frac{d}{dt}||u||^2 + \underbrace{||\sqrt{\delta_1}(u \cdot \nabla u + \nabla p)||^2 + ||\sqrt{\delta_2}\nabla \cdot u||^2}_{\geq 0} = 0$$

thus we have control over the velocities kinetic energy time derivative which can only be ≤ 0 using the positivity of the weighted least-squares terms. We can further investigate the stabilized equation by integrating it with respect to time.

$$\begin{aligned}
&\frac{1}{2}||u(T)||^2 + \int_0^T ||\sqrt{\nu}\nabla u||^2 dt \\
&+ \int_0^T ||\sqrt{\delta_1}(u \cdot \nabla u + \nabla p)||^2 dt \\
&+ \int_0^T ||\sqrt{\delta_2}\nabla \cdot u||^2 dt = \frac{1}{2}||u(0)||^2
\end{aligned}$$

Here we have bounded the solution at time T in terms of the initial condition at time 0. This also shows that if the viscosity is small, the gradient of u can potentially be quite large if we do not use stabilization. With stabilization we also have control of the residual terms $u \cdot \nabla u + \nabla p$ and $\nabla \cdot u$

3.3 G2 for Unified Continuum FSI

In our solver, G2 helps us by stabilizing the method, reducing spurious oscillations in the solution and also dealing with turbulence when it occurs. For the G2 method to work for our implementation, it had to be adjusted to the ALE formulation, which basically meant changing the convective terms in the equation to that given by (2.8). This adjustment was all that was required for the transition from the Navier-Stokes formulation of G2 to the Fluid-Structure Interaction formulation of this thesis. The end result is a stabilized method for the calculation of incompressible fluid interacting with an elastic body.

Chapter 4

Implementation

All the modules created for this thesis can be downloaded from [12].

4.1 FEniCS

The implementation of the method was done using the software provided by the FEniCS project [11]. FEniCS is a platform for the creation and maintenance of tools used for the computation of differential equations using the finite element method. At the heart of FEniCS is DOLFIN which is a set of libraries that enable the programmer to create fast and reliable code in a consistent problem solving environment for solving problems in the realm of scientific computing.

4.1.1 DOLFIN

The software package relies on well-used algebraic back-end software such as LAPACK and PETSC for calculations. DOLFIN itself provides an interface between the back-end and the programmer in addition providing the programmer with important classes for mesh generation, i/o, matrix/vector assembly and various finite element functions and operations.

4.1.2 FFC

FFC is a program which automates the generation of a chosen discretization of a differential equation using finite elements. It receives the variational form of the governing equations and compiles the equations to generate a header file for the inclusion into DOLFIN. FFC relies on the FIAT software package for generating finite element basis functions and can make use of the FErari package for the optimization of the evaluation of element tensors.

4.2 Modules

DOLFIN already has a few modules built around it. Modules are created for the purpose of simplifying some computational aspects for the programmer including predefined environments for the calculation of specific equations e.g., Navier-Stokes equation. The Navier-Stokes module provides the programmer with a solver for a Navier-Stokes problem. This includes the variational form of the equations as well as the solver methods, in this case the cG(1)cG(1) space-time method combined with fixed-point iteration.

4.3 Implementing the ALE module

Step one in the implementation of the thesis was to create an ALE module. A piece of code that would be able to handle a moving mesh and use mesh smoothing to ensure quality of the mesh. It should also be able to capture the mesh velocity $\hat{v}$ since we will want to use this in the ALE formulation. Since there already exists a class which handles boundary conditions, we will have to create a new class that 'sits' between the original boundary condition class and the new ALE adjusted boundary condition class. The new boundary condition class is necessary because we would like to define the conditions in terms of the reference coordinate system and not the spatial system as this will lead to inconsistencies when the mesh moves outside the given spatial boundary conditions. Summarizing all this, we need to change the existing Navier-Stokes module to accommodate for:

- ALE formulation
- Mesh moving + smoothing
- ALE based boundary conditions

4.3.1 ALE formulation

All that was required in this part was to replace the convective velocity with the convective velocity which compensates for the movement of the mesh (2.8). This had to be done for the stabilization as well.

4.3.2 Mesh moving and smoothing

This required having explicit access to the vertex values of the mesh. Once this was done, we could easily set the values of the vertices to any values we would like. Having this amount of control enables the use of mesh-smoothing techniques such as (2.9) which was successfully implemented.

4.3.3 ALE boundary conditions and external deformation function

Perhaps one of the more challenging aspects was to implement a new boundary condition class that would at the same time conserve the underlying framework. A solution was found by placing another class between that of the original boundary condition class and the end-users C++ implementation, see Fig.4.1. This way we could record the initial boundary point values at $t = 0$, as mentioned before, these points coincide with the reference frame points. After recording, we would cycle over these reference points again as the original boundary condition requests the boundary values for the spatial points.

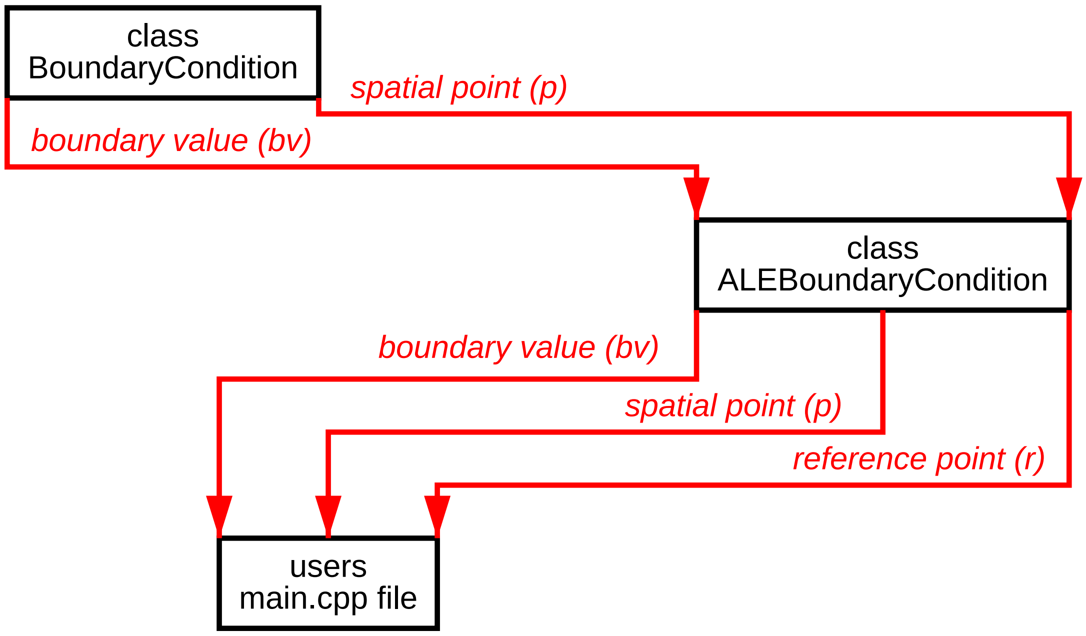


Figure 4.1. An illustration of how the ALEBoundaryCondition class is situated. BoundaryCondition does not know that there lies a layer between itself and the user.

To take advantage of the fact that we have an ALE configuration, it would be useful to have a function that could deform the boundaries of the domain. Such a function is now included and works similarly to that of the ALEBoundaryCondition class in the sense that the user is passed information about the spatial point as well as the reference point. All the user has to do is return a value and this value will then be used as the deformation velocity for the boundary.

4.4 Unified Continuum FSI implementation

4.4.1 Variational formulation

Since we are solving for a total of three variables u, p, σ_D, we need the corresponding number of equations. The basis for constructing a finite element method is to form the variational formulation of the equations. The type of trial and test functions used here are different for each of the functions u, p and σ_D. For a deeper understanding of FEM, see [8] and [7].

It is not necessary to construct a variational form for the stress deviatoric since we can simply update it by using the following relations.

$$\dot{\sigma}_D = E\epsilon(u)$$
$$\dot{\sigma}_D \approx \frac{\sigma_D^n - \sigma_D^{n-1}}{k}$$

where we approximate $\epsilon(u) \approx \epsilon(u^n)$. This then gives us our stress update equation as seen in (4.4).

While u is piecewise (p.w.) linear in space, the stress deviatoric σ_D is p.w. constant for each cell. Since we are dealing with a space-time discretization, the trial functions also vary in time. We use a cG(1)cG(1) discretization, meaning p.w. linear trial functions in space and time but with test functions being p.w. constant in time, see the table below.

Table 4.1. Table of test and trial functions

Function	**Variable**	**Space**	**Time**
velocity trial	u	p.w. linear	p.w. linear
pressure trial	p	p.w. linear	p.w. constant
velocity test	v	p.w. linear	p.w. constant
pressure test	q	p.w. linear	p.w. constant
stress deviatoric	σ_D	p.w. constant	p.w. linear

Velocity u is defined as a p.w. linear function in time. This means that the resulting integration of the variational form will result in a Crank-Nicolson type time stepping scheme whereby we take the average value of the velocity across one time step. So let us begin with the momentum equation.

$$\begin{aligned}&(\rho\dot{u}, v) + \phi((u-\hat{v})\cdot\nabla u, v) - (p, \nabla\cdot v)\\&+ (\alpha\epsilon(u), \epsilon(v)) + \phi(\delta_1(u-\hat{v})\cdot\nabla u, (u-\hat{v})\cdot\nabla v)\\&= (f, v) - (1-\phi)(\sigma_D^{n-1}, \nabla v)\end{aligned} \tag{4.1}$$

where v is the test function, δ_1 is the stabilization parameter and

$$\rho = \phi\rho^F + (1-\phi)\rho^S \tag{4.2}$$
$$\alpha = \phi 2\nu + (1-\phi)kE \tag{4.3}$$

The stress deviatoric for the structure part has to be calculated using the previous value of the stress deviatoric, σ_D^{n-1} where we use the approximation

$$\sigma_D^n = \sigma_D^{n-1} + kE\epsilon(u^n) \tag{4.4}$$

which is why σ_D is absent from the variational formulation and $\alpha(\phi = 0) = kE$. This means that we have to solve (4.4) separately in our solver keeping the value of σ_D^n until we need it in the formulation of the next time step as the value σ_D^{n-1}. The keen eye may notice that we do not really need a ϕ in the convection term since this will be 0 anyway because the structure is evaluated using the Lagrange coordinate system and thus the velocities cancel each other out.

Next up is the variational formulation for the pressure. We solve for the pressure using the continuity equation. This is exactly the same as is being done for the existing Navier-Stokes module.

$$(\delta_2 \nabla p, \nabla q) = (\nabla \cdot u, q) \tag{4.5}$$

Here we use q as our test function and δ_2 is the stabilization parameter.

4.4.2 Computing ϕ

Our marker function ϕ is calculated by letting the user specify a bisection function which divides the domain into two regions. We iterate over all cells and for every cell we iterate over all the vertices connected to the cell. For every vertex, we query the user's bisection function. If the bisection function returns 0 for all the vertices that belong to the cell, then the cell is part of the structure. If one or more vertices result in a returned value of 1, then the whole cell is automatically considered to be part of the fluid. This means that the bisection function is fluid biased. ϕ is then defined for all cells which means that our interface will be a union of edges which divide the domain into fluid and structure parts.

4.4.3 Solving the model

Once we have the necessary formulations written down, all we have to do is build our new solver module, the FSI module. Inside the FSI module, we will have the actual FSISolver class together with the FFC generated finite element header files. Instead of describing the process, included is a flow chart which outlines the procedures in a more holistic manner, see Fig.4.2.

The algebraic solvers that are used rely on the algebraic back-end that is attached to DOLFIN. Optimally, we use Algebraic to solve the continuity equation for the pressure. We use gmres with ILU preconditioning for the momentum equation.

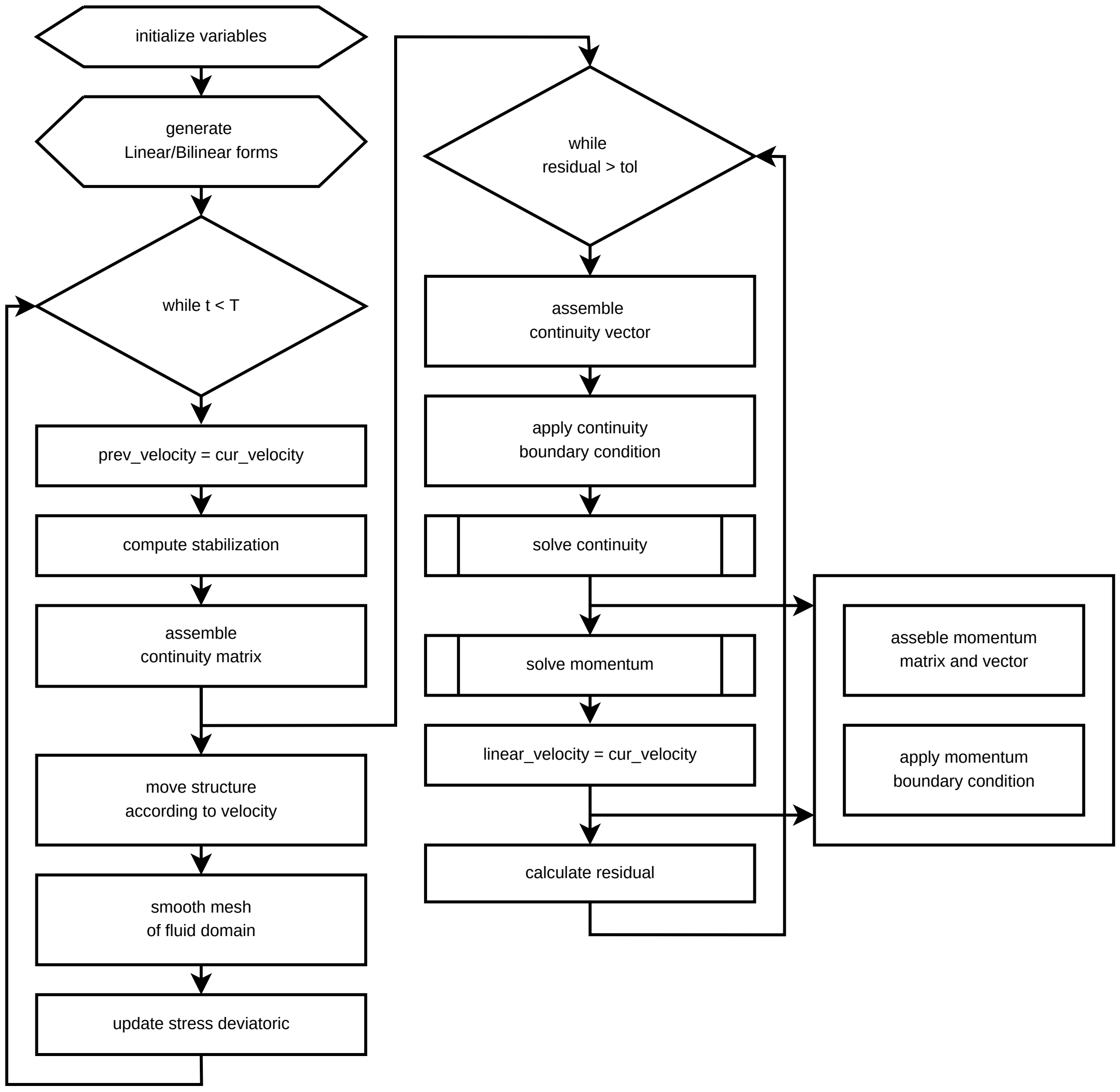

initialize variables
generate Linear/Bilinear forms
while t < T
prev_velocity = cur_velocity
compute stabilization
assemble continuity matrix
move structure according to velocity
smooth mesh of fluid domain
update stress deviatoric
while residual > tol
assemble continuity vector
apply continuity boundary condition
solve continuity
solve momentum
linear_velocity = cur_velocity
calculate residual
asseble momentum matrix and vector
apply momentum boundary condition

Figure 4.2. An illustration of the processes that occur inside the FSISolver class.

Chapter 5

Results from Computation

Images that are located here are stills of animations. These animations can be downloaded from [12].

The results are from a simulation whereby we have a fluid flowing through a channel that is initially at rest. For the flow channel, we use a parabolic inflow profile $u = 4y(1-y)$ on the left boundary and we set $p = 0$ at the outflow to model a free flow on the right boundary. We use no slip boundary conditions on the top and bottom boundaries. We then move the top and bottom boundaries and observe the behaviour of the fluid.

5.1 ALE images from computations

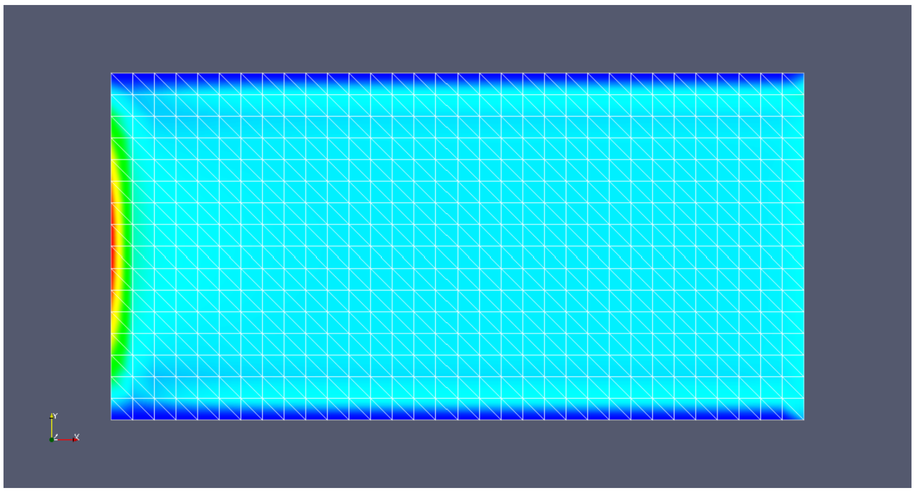


Figure 5.1. This figure shows the flow at $t = k, k$ is the time step size. Here the mesh is unchanged and in the *reference* configuration.

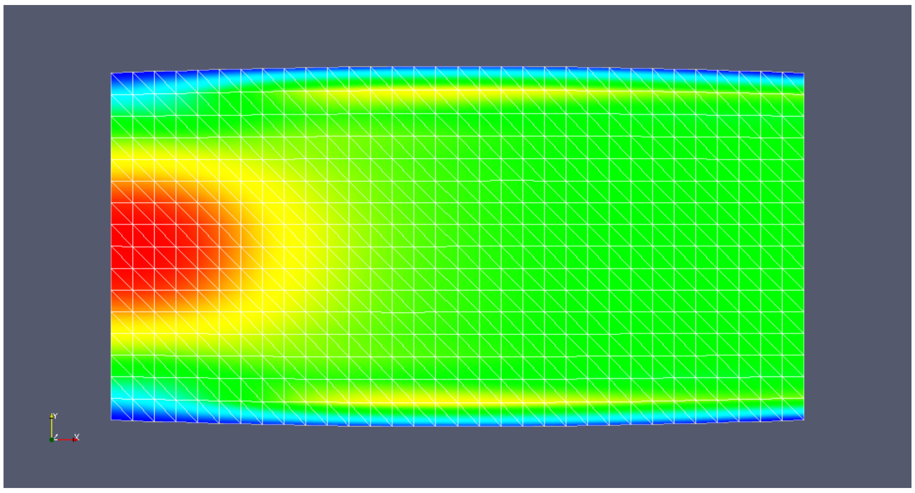


Figure 5.2. $t = 0.4$. The boundaries expand, and the mesh smoothing algorithm prevents the boundary triangles from distorting.

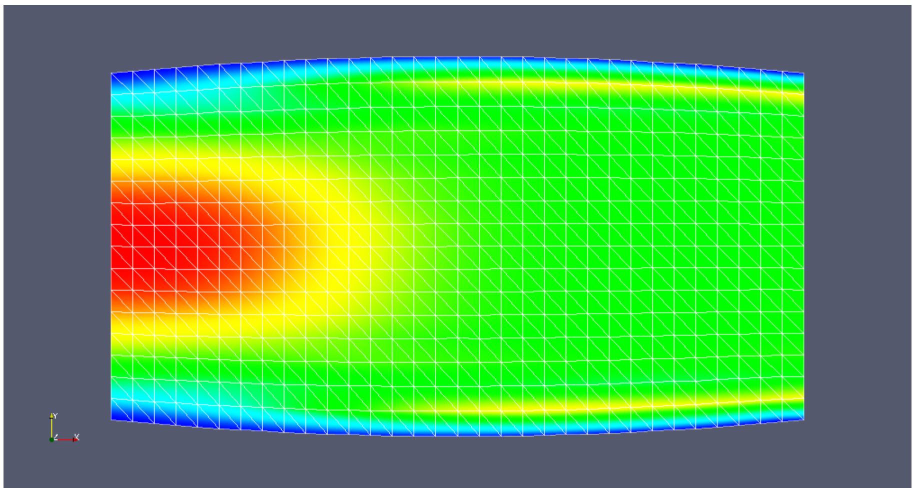


Figure 5.3. $t = 0.8$. Just a few fractions of a second later, another snapshot of the fluid domain.

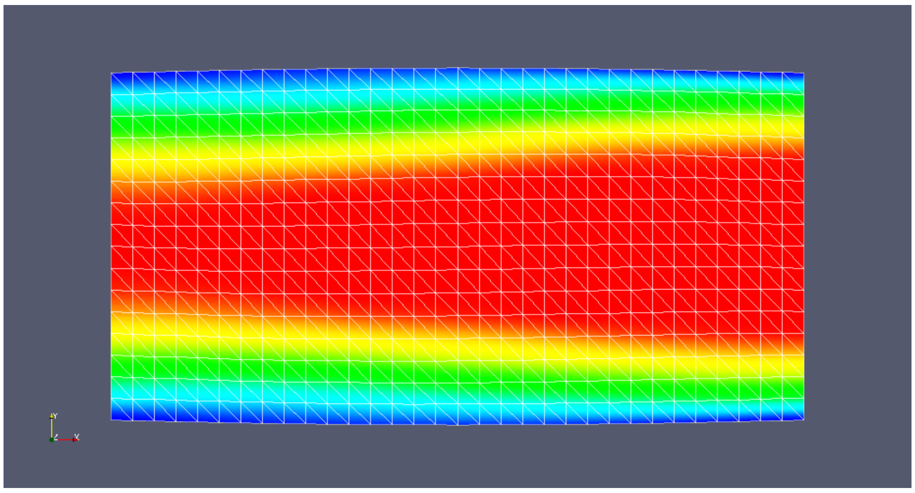


Figure 5.4. $t = 8.0$. The end of the simulation.

5.2 Fluid container

Initially, we did simulations using a vessel that was suspended in space by using homogeneous Neumann boundary conditions on the vessel walls. Mathematically speaking, we use homogeneous Neumann boundary conditions on the external vessel wall boundaries. A result of this was that our time step size was constrained to the order of 10^{-5}.

There are several ways to remedy this. One way would be to include surface tension boundary conditions on the outer surface of the walls, or to stabilize the boundary stresses. Another solution, that we then implemented, was to use a fluid container around the vessel walls. The fluid container then has fixed Dirichlet boundary conditions. Since our solver only uses one fluid definition, this *buffer* fluid has the same properties as what flows through the vessel. In our case we are trying to simulate blood, thus the buffer region contains blood.

A bonus from this fluid container is that we gain in realism since it is not really realistic to have a vessel floating in space with no external environmental effects. Using a fluid container is quite realistic since the body is mainly made up of fluid, if we consider muscle tissue and other surrounding physiological fibres.

5.3 FSI blood flow

5.3.1 Overview of simulation

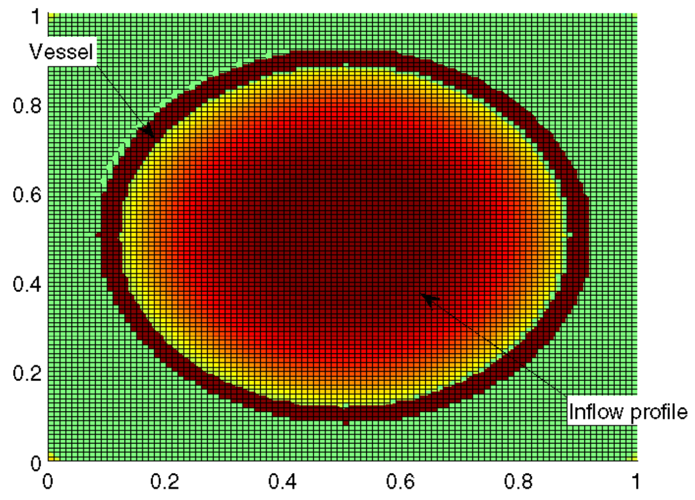
The boundary conditions used in the simulation of blood flow are the following:

- **Inflow boundary:** We use a parabolic inflow profile for the velocity with an amplitude that is time-dependent. The inflow profile is shown in Fig.5.5 while the amplitude corresponds to that of a pulse Fig.5.6.
- **Outflow boundary:** Here we just set $p = 0$. This means that we model a *free* outflow at the right boundary.
- **Fluid container:** The fluid container, as mentioned above, has fixed Dirichlet boundary conditions. This corresponds to the no-slip condition.
- **Vessel walls:** The vessel walls are fixed at both the left and the right hand side. Thus we use fixed Dirichlet boundary conditions to clamp them.

For these particular simulations, we use the following parameters:

Young's modulus	$E = 30$
Poisson's ratio	$\eta = 0.3$
Fluid density	$\rho^F = 1.0$
Solid density	$\rho^S = 1.0$
Viscosity	$\nu = 0.001$
Nr. of vertices	4843
Nr. of cells	9472
Time step	$k = 0.00552$

We wanted to be able to see how the vessel walls deform according to the pulsating inflow, this means that we used a somewhat more *relaxed* configuration for the solid material compared to the parameters used in the parameter study of this thesis.



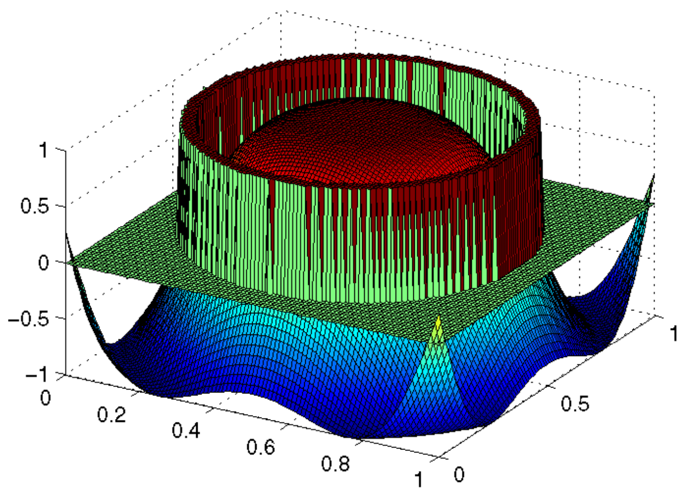


Figure 5.5. Here we see the inflow profile superimposed on the vessel. This allows us to check whether we have zero velocity at the interface boundaries between the fluid and the vessel wall and that everything looks like it should. Please note that we truncate the inflow profile function once we reach the inside of the vessel walls. What we see in these figures is the untruncated version where it looks like we have a negative inflow velocity.

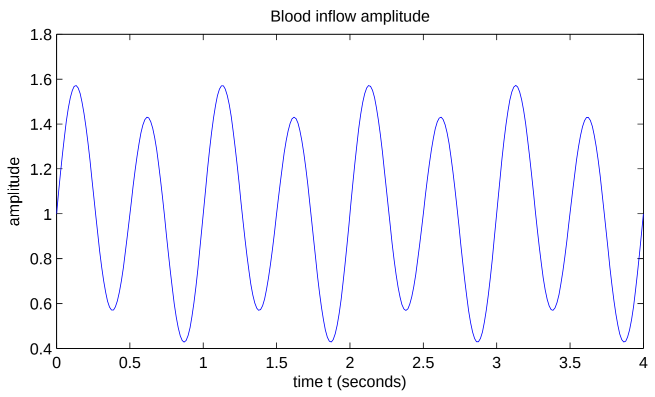


Figure 5.6. The amplitude that was used for the blood inflow profile. The period of the curve is 1 second.

5.3.2 Images from computation

Uniform mesh refinement

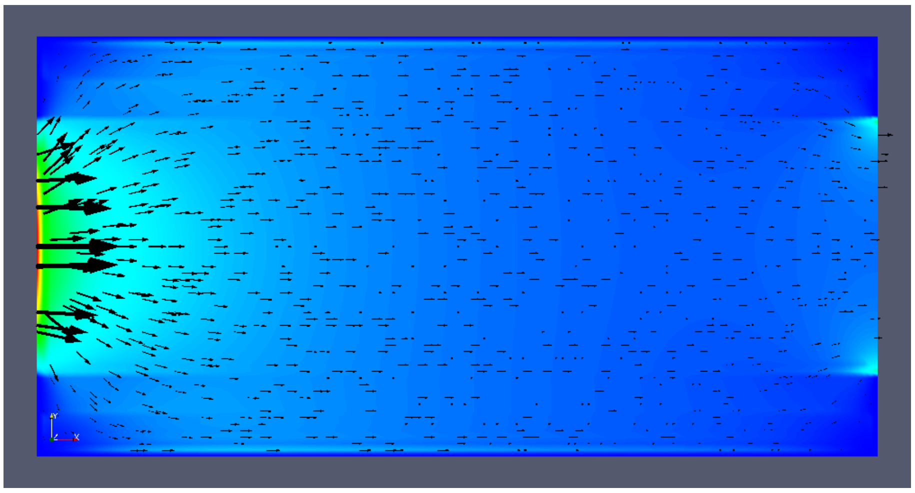


Figure 5.7. The domain at $t = k$. We can see where the vessel walls begin since there is a definite velocity (colour) difference between the flow channel (green/light blue) and the vessel wall (darker blue) near $x = 0$ and then another transition into an even darker shade of blue which is the fluid container surrounding the vessel.

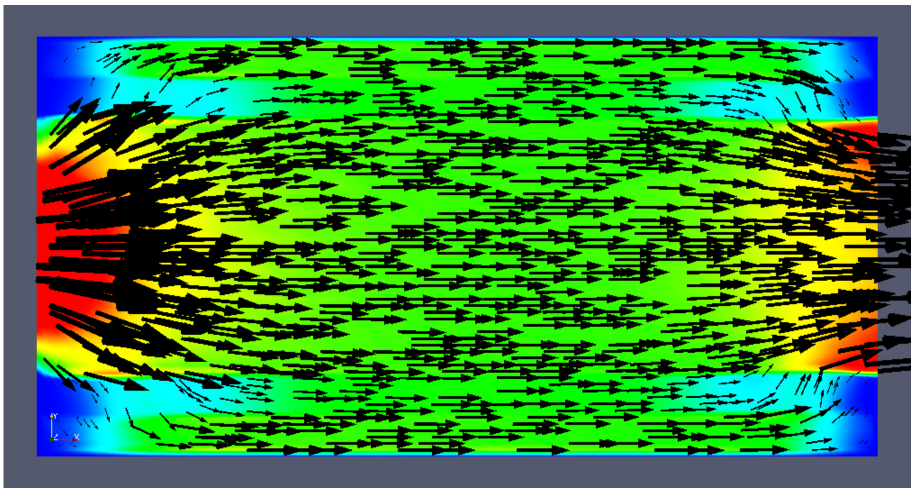


Figure 5.8. $t = 0.1$. The arrows indicate the velocity of the medium at that point. It sometimes looks like the fluid is going through the vessel walls but that is just the solid moving in the direction of the arrows.

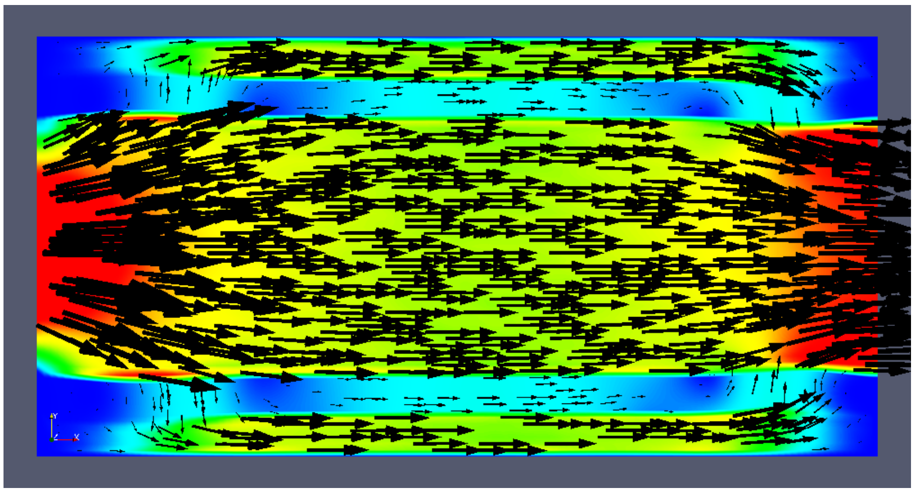


Figure 5.9. $t = 0.2$. This image makes it quite easy to see that we have a vessel wall that contains a fluid, and at the same time is surrounded by fluid.

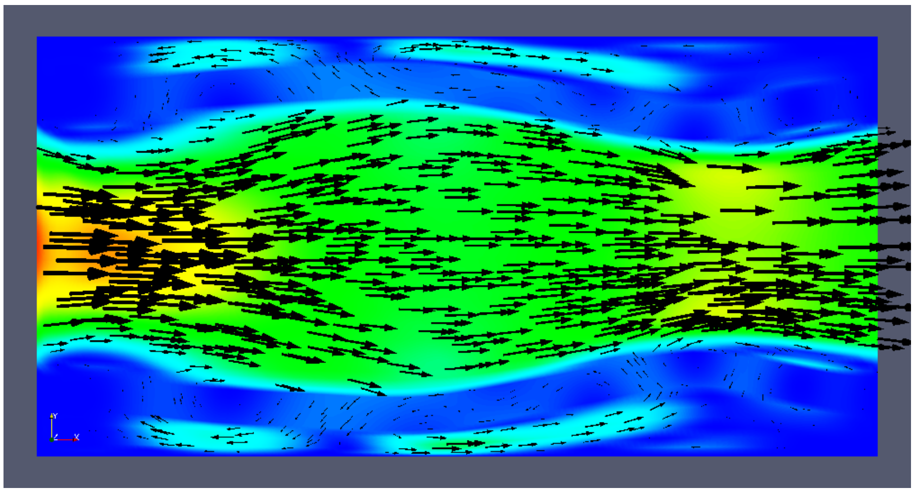


Figure 5.10. $t = 2.0$. At the end of the simulation.

Locally refined mesh

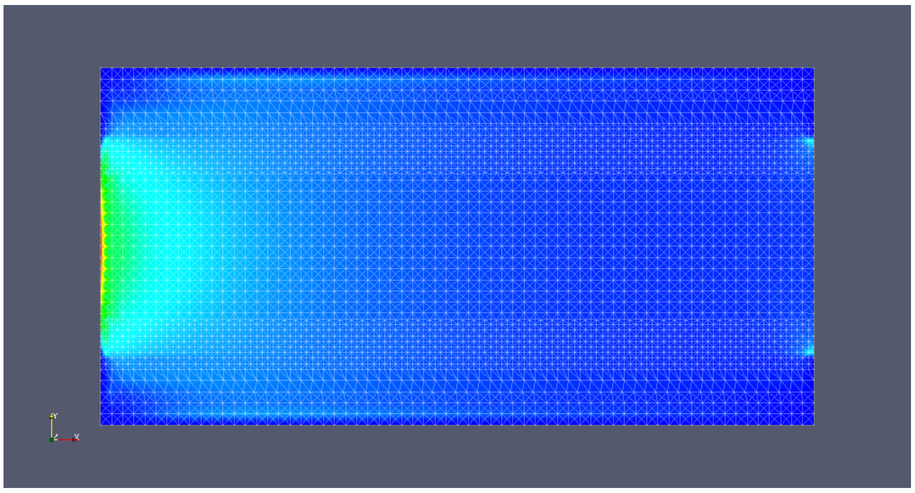


Figure 5.11. A figure of the initial mesh using local mesh refinement. We refined the interface to account for the large change in the velocity u in the boundary layer as we go from the fluid domain to the solid structure domain.

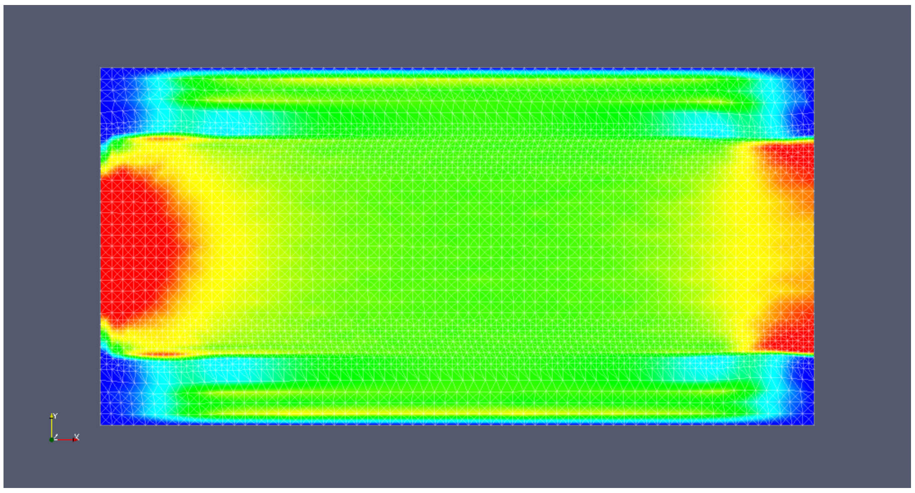


Figure 5.12. $t = 0.1$

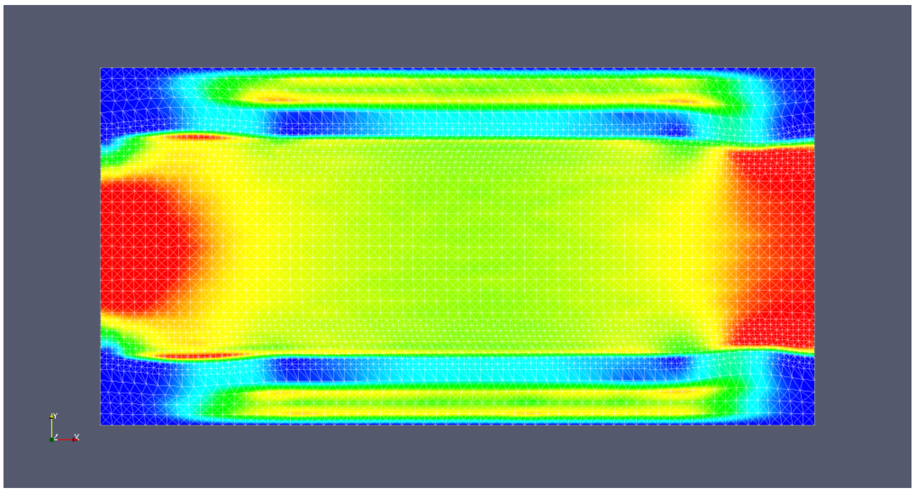


Figure 5.13. $t = 0.2$. It looks very similar to Fig.5.9 albeit a bit more detail along the interface.

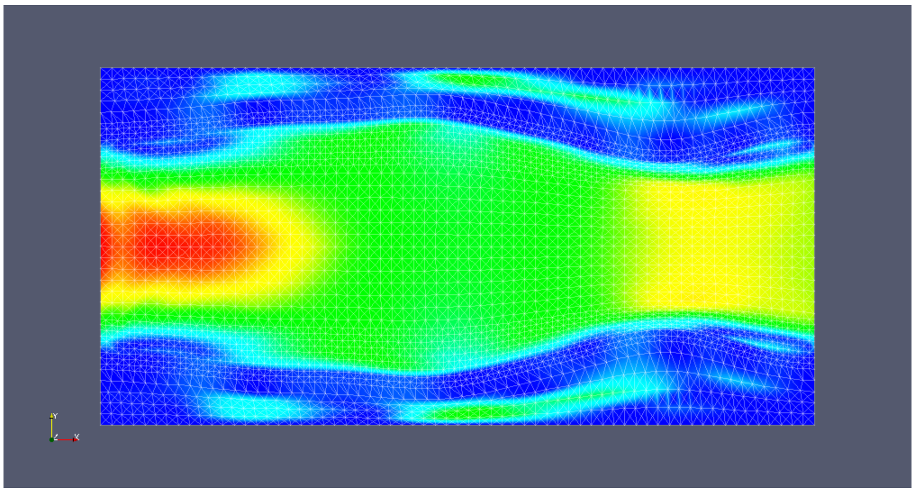


Figure 5.14. $t = 2.0$. The end of the simulation. It looks very similar to Fig.5.10.

3D Uniform mesh refinement

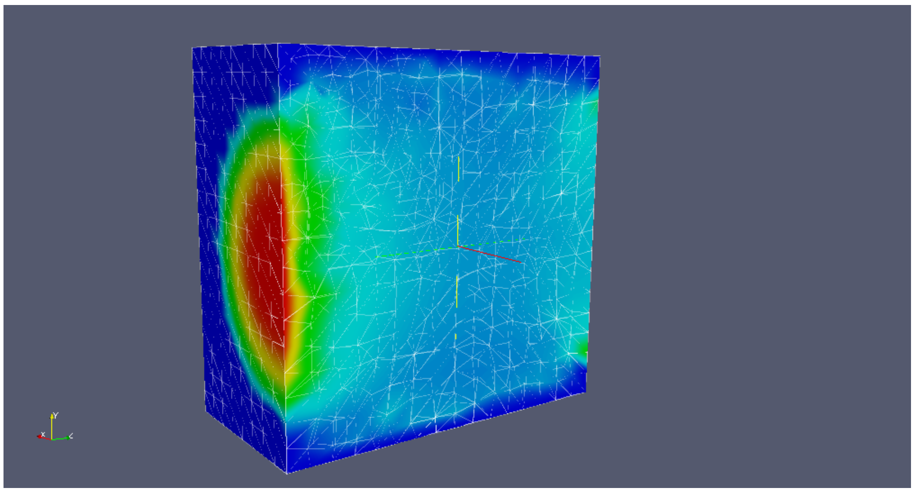


Figure 5.15. The initial configuration of the 3D cube at time $t = k$. Our cube acts as a container wherein a vessel is placed. Surrounding the vessel (cylindrical structure in the middle) is fluid. 3D simulations require a lot of resources and thus we cannot reach the same refinement levels as we did in the 2D models from the previous sections.

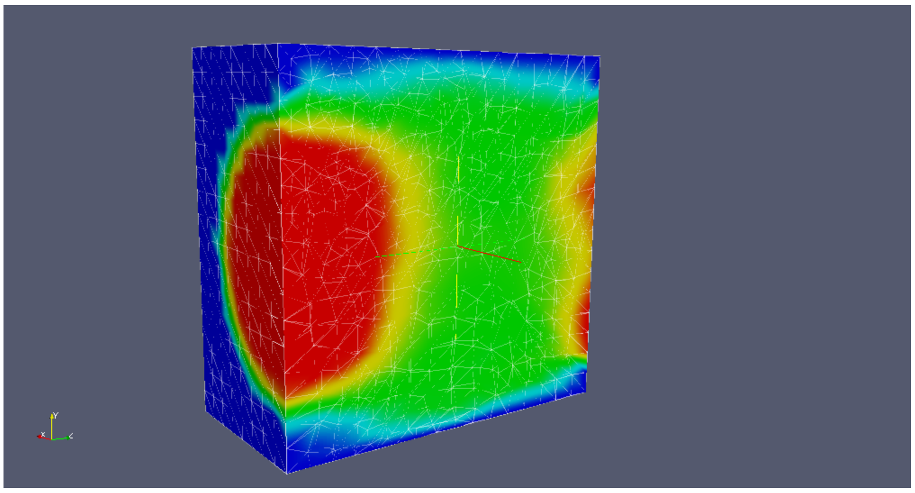


Figure 5.16. $t = 0.1$

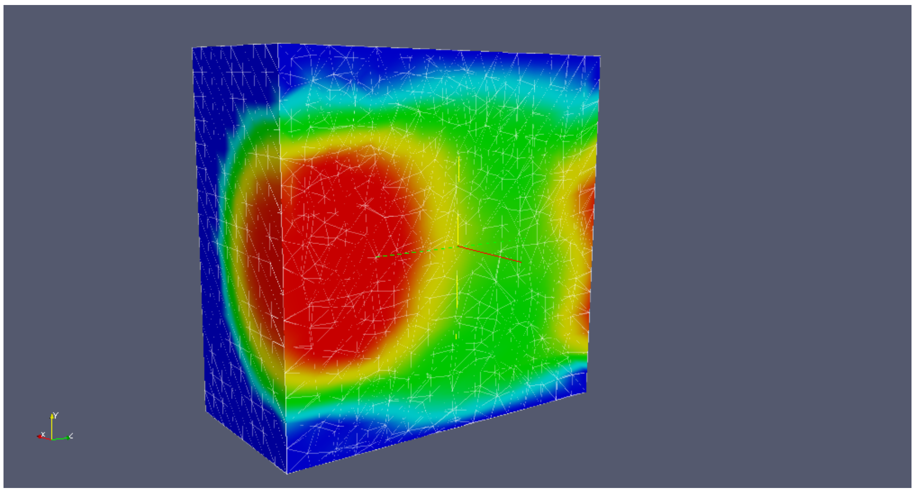


Figure 5.17. $t = 0.2$

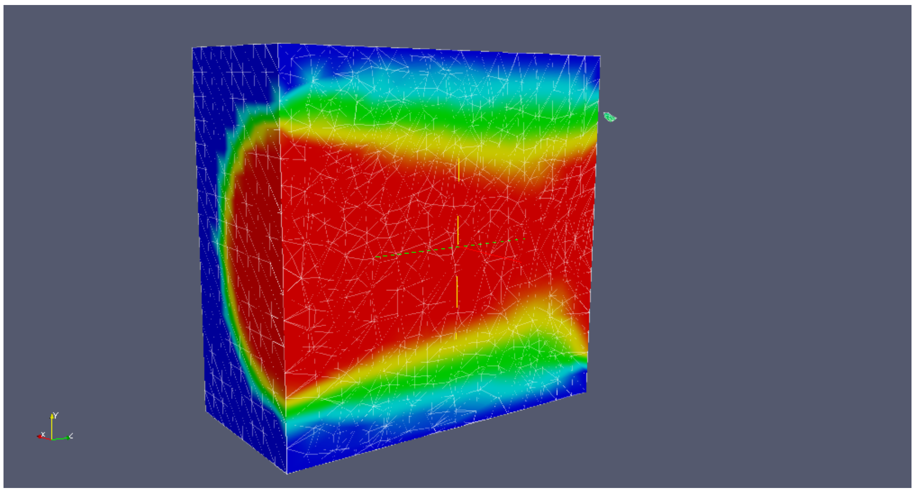


Figure 5.18. $t = 0.4$. The simulation would have taken a total of 10h on a laptop so we cut the simulation short by a few steps. This is just to indicate that it is indeed possible to use our method when we wish to calculate a solution in 3D.

Pressure visualization

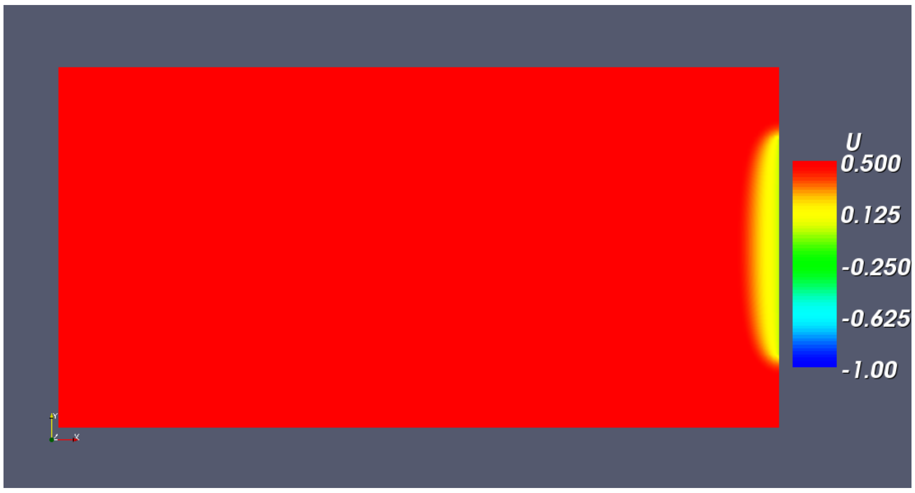


Figure 5.19. Here we see the pressure distribution at time $t = k$. One can compare these figures with the velocity plots to get a bigger picture. In the animation, we see how the pressure fluctuates along with the pulse of the inflow profile.

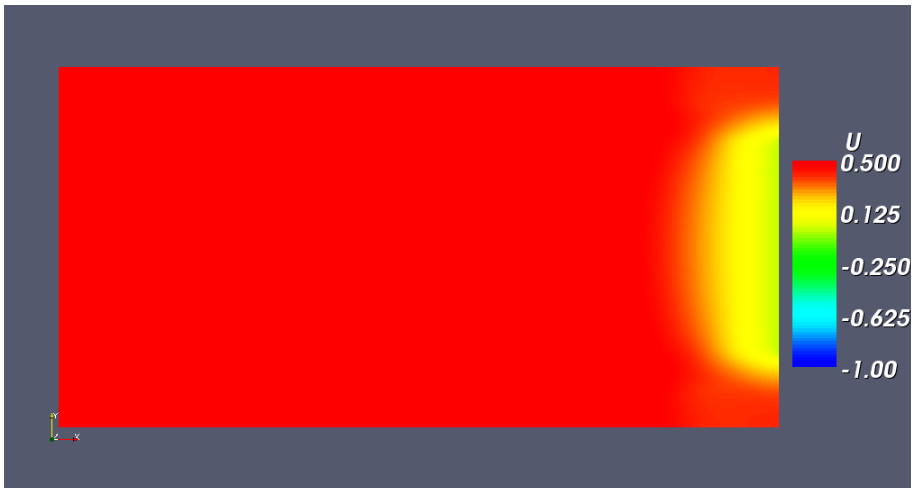


Figure 5.20. $t = 0.1$

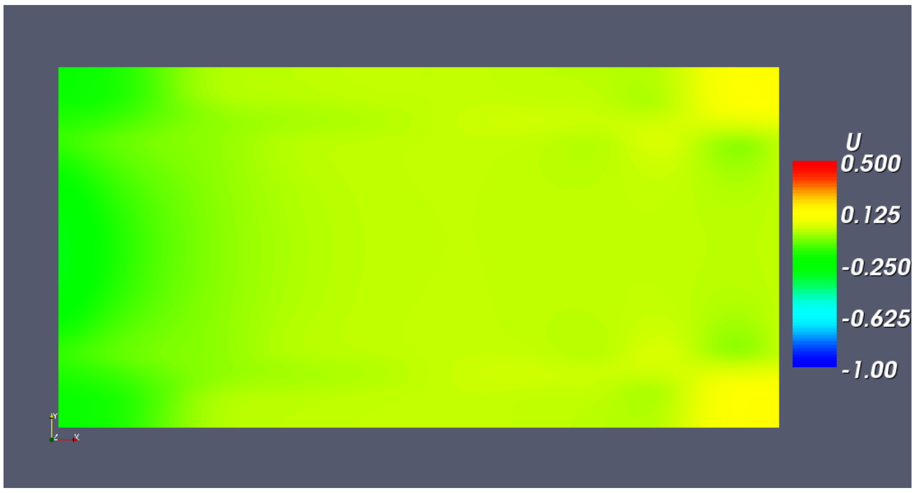


Figure 5.21. $t = 0.2$

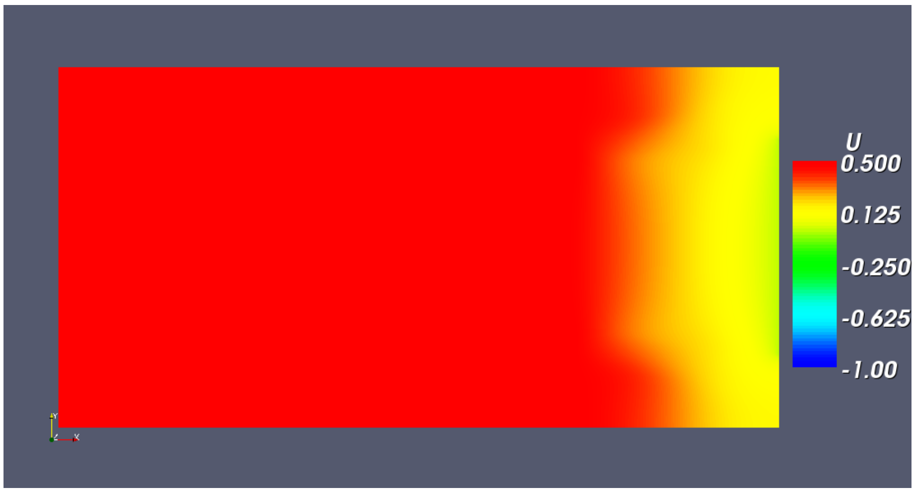


Figure 5.22. $t = 2.0$. End of the simulation.

Chapter 6

Parameter Study

6.1 Default parameters

We may presume from [6] that the solid/fluid density ratio for blood flow simulations is 1.2. At the same time, we would like to preserve the volume of our elastic body, thus $\eta = 0.5$[1]. Blood is largely made up of water and thus we use low viscosity in our computations. Since empirical data is sparse, we are free to choose our elastic parameter Young's modulus. From calculations, we believe that 50 is quite a reasonable value.

Young's modulus	$E = 50$
Poisson's ratio	$\eta = 0.5$
Fluid density	$\rho^F = 1.0$
Solid density	$\rho^S = 1.2$
Viscosity	$\nu = 0.001$
Nr. of vertices	4843
Nr. of cells	9472
Time step	$k = 0.00552$

In the figures that follow, the first entry in the legend always corresponds to the default parameters shown above. Solutions are calculated on a locally refined mesh as in Fig.5.11.

6.2 Stability testing

In this section, we use a time step size that is a quarter of the minimum cell diameter $k = 0.25 \cdot h_{min}$. This is the time step size that is used in the existing Navier-Stokes solver.

[1]Mathematically, this would be wrong due to our derivation in 2.38, but since we are in a numerical framework, the difference between 0.4999 and 0.5 is negligible.

6.2.1 Variation in viscosity

Time stepping statistics

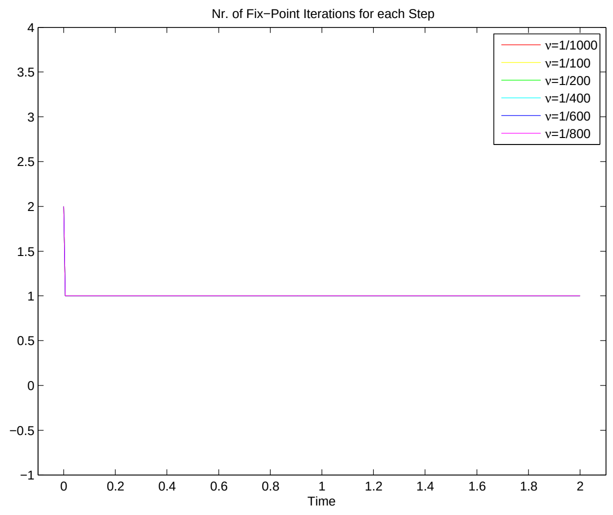


Figure 6.1. This figure shows the number of iterations of the fix-point iteration that were carried out for each time step. We see that it usually takes two iterations in the beginning and then just one making the method pseudo-explicit.

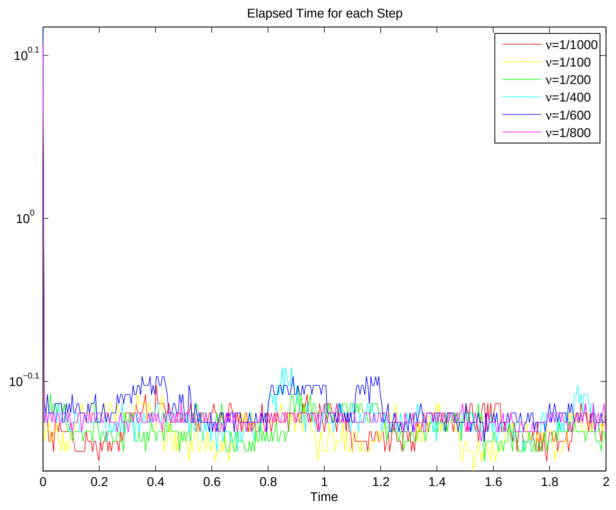


Figure 6.2. Here we see the time it took to calculate each time step. The values at $t = 0$ correlate with the two fix-point iterations that are done in the beginning of each trial.

Residual statistics

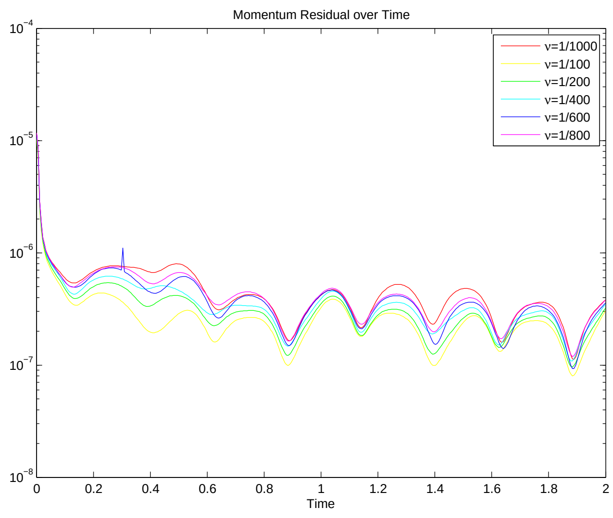


Figure 6.3. This is a plot of the residual from the momentum equation. Lower values of ν mean higher residuals. The spike for $\nu = 1/600$ is “odd”.

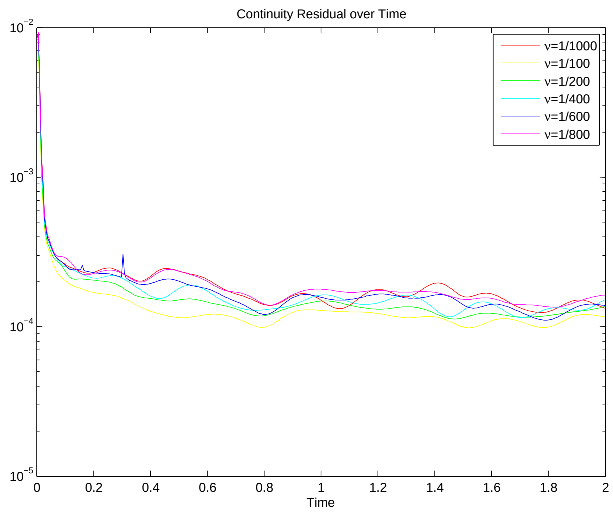


Figure 6.4. Residual from solving the continuity equation. This residual is always a bit higher since its equation is much harder to solve. It follows a similar trend to the momentum residual.

6.2.2 Variation in density

Time stepping statistics

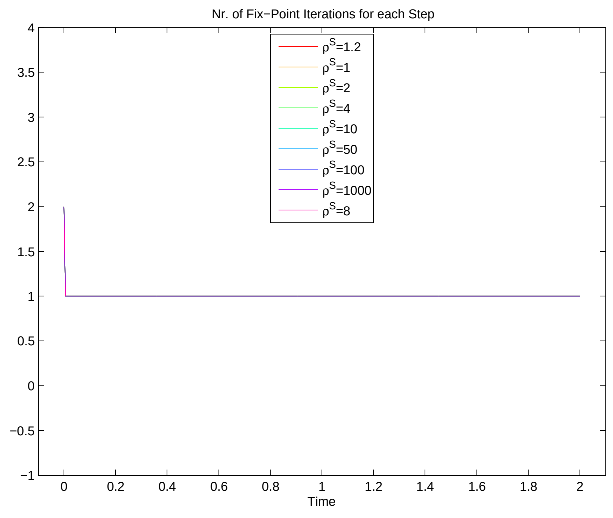


Figure 6.5. Much the same thing as we saw for the viscosity. First two iterations followed by just 1 fix-point iteration for each consecutive step.

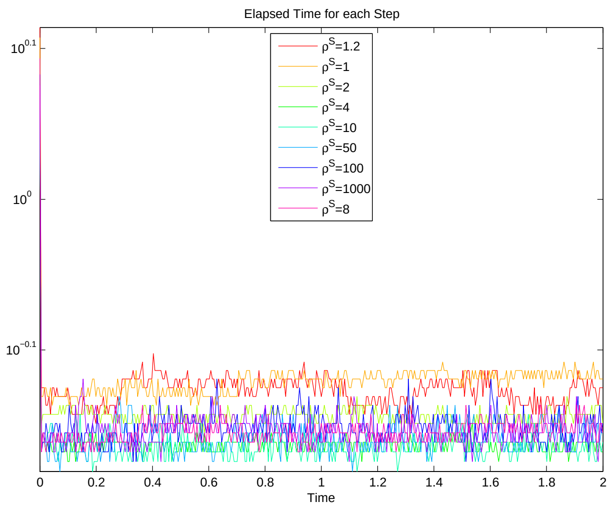


Figure 6.6. Here we see the time it took to calculate each time step. The density ratios close to 1 take a bit longer to calculate compared to the others.

Residual statistics

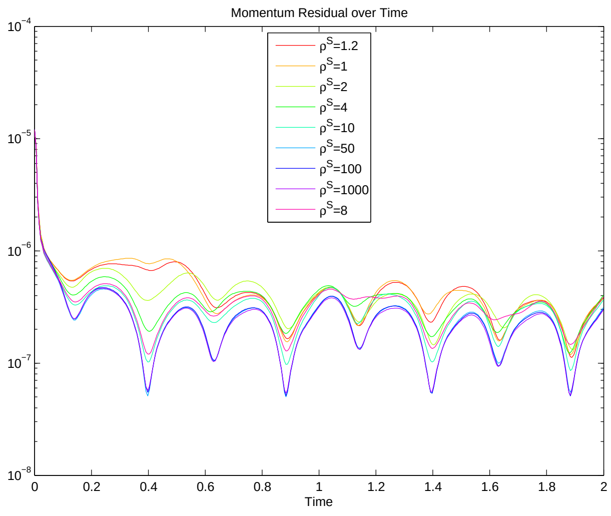


Figure 6.7. The solver has higher residuals for density ratios close to 1.

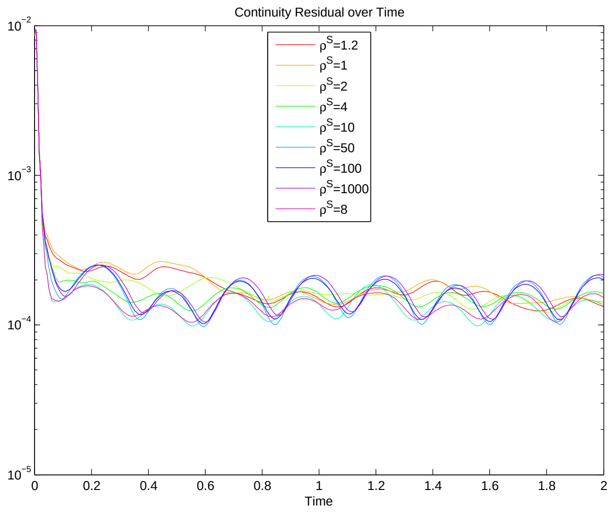


Figure 6.8. Residual from solving the continuity equation.

6.2.3 Variation in Young's modulus

Time stepping statisics

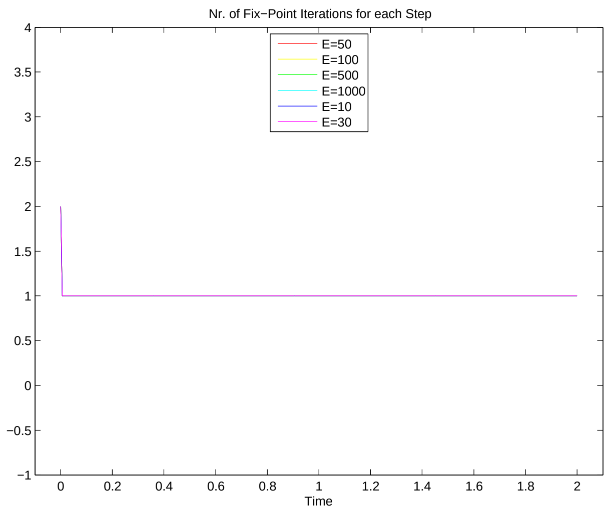


Figure 6.9. A, by now, quite familiar picture.

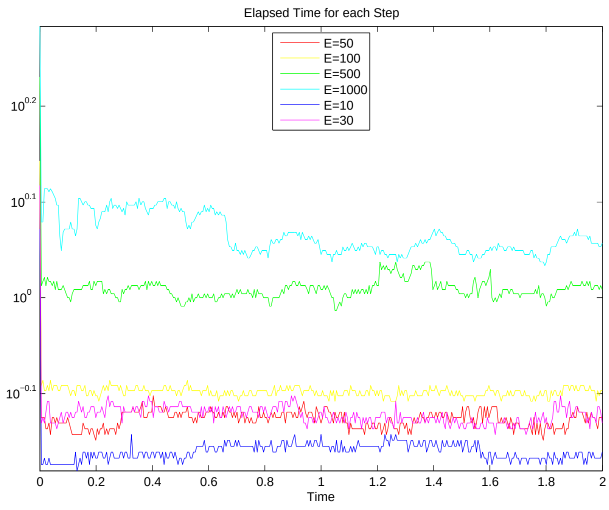


Figure 6.10. The higher the value of E, the longer each step takes to compute.

Residual statistics

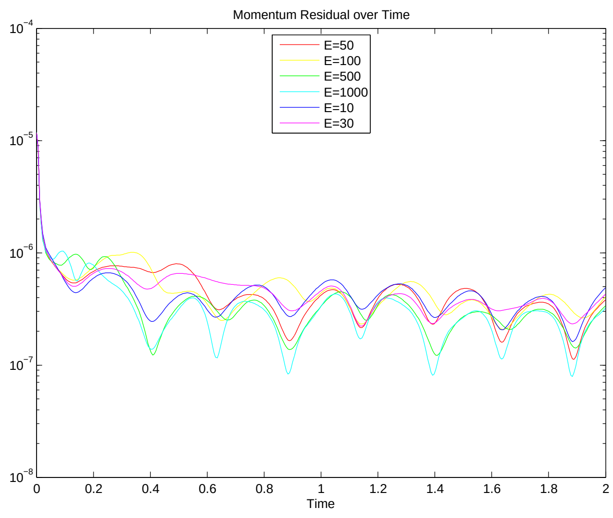


Figure 6.11. We can see that the residual is higher for lower values of E. Lower values of E mean that the solid behaves more elastic.

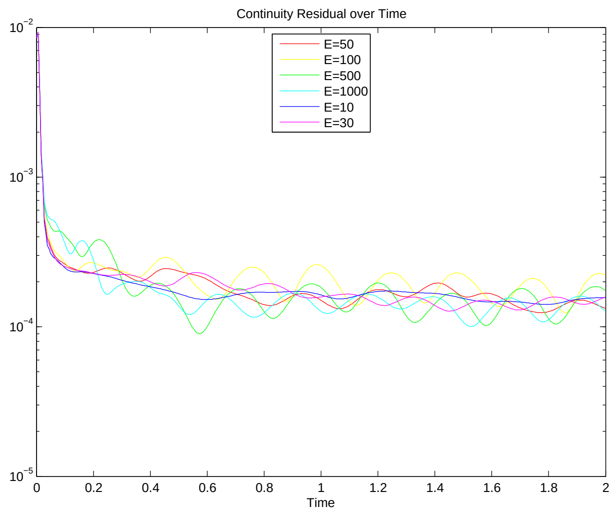


Figure 6.12. Residual from solving the continuity equation.

6.3 Summary

Throughout all the different trials, the solver has proven to be quite stable. Even for the density ratios that are close to 1 which apparently are somewhat more difficult to resolve for standard FSI solvers with a weak coupling.

In conclusion, we have been able to keep the relation $k \propto h$. This is by no means obvious since the solver is also dependent on how the domain deforms over time. If a small triangle where to collapse in the space of one time step, then the solver would not be able to resolve this and would thus be unable to compute a solution.

Chapter 7

Convergence Study

7.1 Methodology

In this chapter, we compare the solution gained from computation on a fine grid, called the *finest* solution, with 19155 degrees of freedom to other solutions calculated on grids with a lower number of degrees of freedom. This way we can calculate the errors for each approximation that is of inferior quality to the finest solution. Note that we keep the relationship $k = 0.25 \cdot h_{min}$ because we have a space-time discretization method. Since our simulation is time dependent, we chose to estimate the error at $t = 2$. The number of degrees of freedom are listed in the table below.

Table 7.1. Table of the solutions and their degrees of freedom

Solution	**Degrees of Freedom**
finest	19155
approx1	93
approx2	315
approx3	1271
approx4	4843

To be able to subtract the two solutions, we were required to project the approximate solutions onto the mesh of the finest solution. This was done using an $L2$ projection[8]. Since the coarser meshes are contained in the finest mesh, the corresponding finite element spaces are subspaces so that the $L2$ projection error is zero.

7.2 Max norm and L2 norm of the error

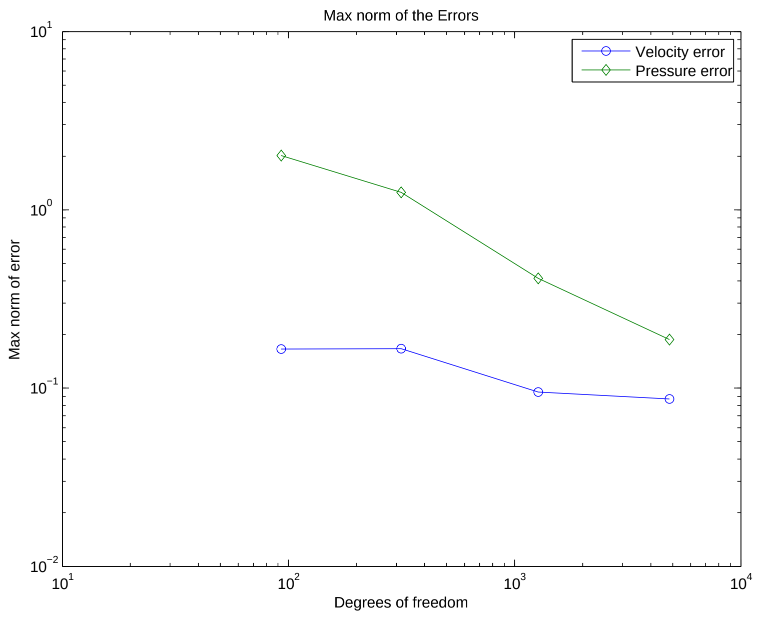


Figure 7.1. The max error, which is the largest singular error anywhere in the domain, shows signs of a downward trend as we increase the number of degrees of freedom.

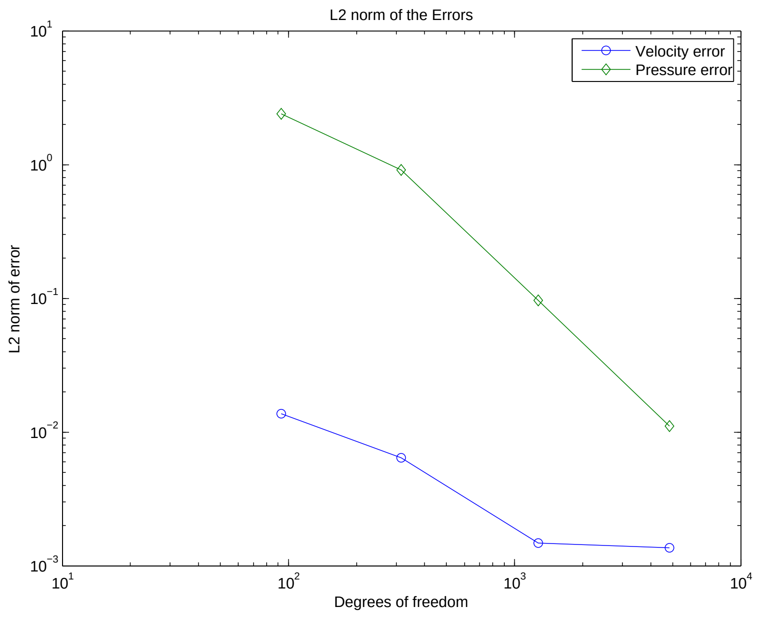


Figure 7.2. The pressure $L2$-error indicates 2nd order convergence whereas the velocity $L2$-error is rather 1st order but stagnates as we get closer to the refinement level of the finest mesh. Using more samples we will be able to investigate whether this is a true effect or not.

7.3 Visualization of errors

7.3.1 Velocity error

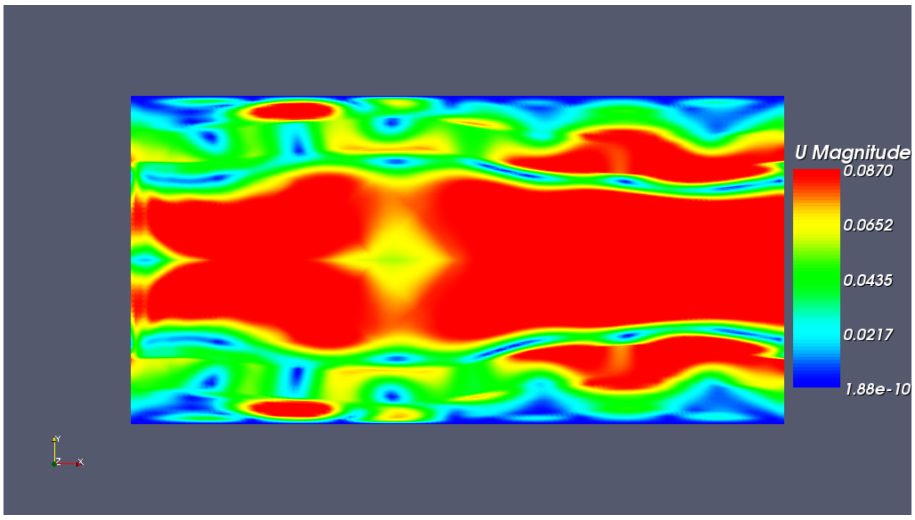


Figure 7.3. Approx1: Since all the solutions have the same boundary conditions, we shouldn't be able to see any error on the boundaries. However, since we are dealing with a space-time discretization, the solutions at $t = 2$ do not match up exactly. This means that some of the approximate solutions may be from $t = 1.985$ while the *finest* solution is from $t = 1.999$ instead.

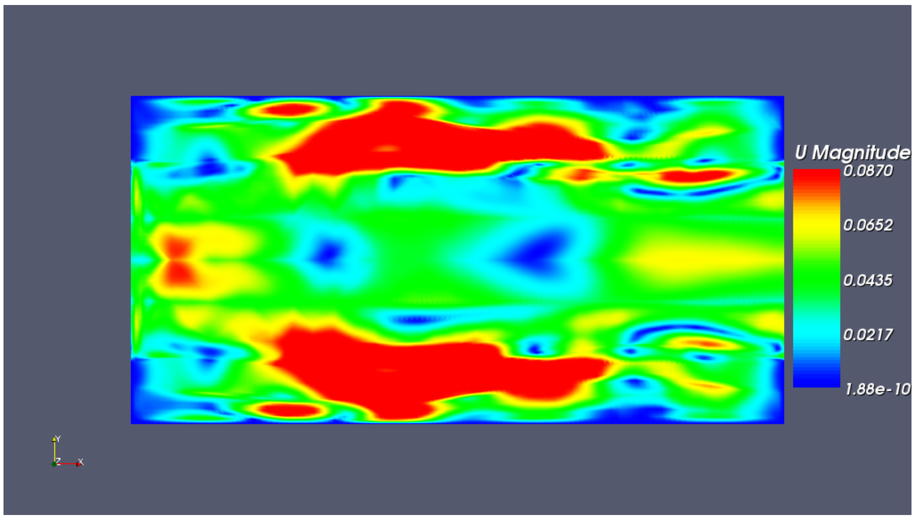


Figure 7.4. Approx2: We can see that the error from the previous solutions has become smaller and concentrated in other areas.

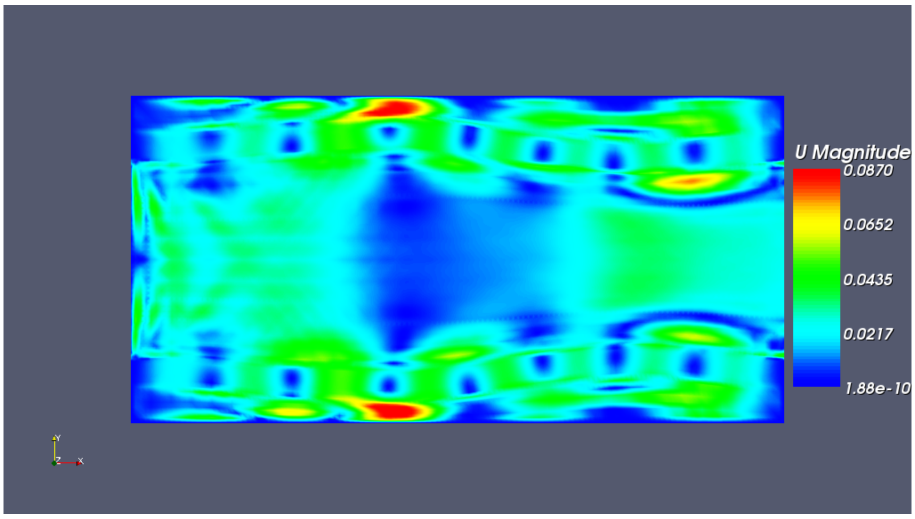


Figure 7.5. Approx3: A periodic structure shows itself from the error solutions that correspond to the solid material in the computations.

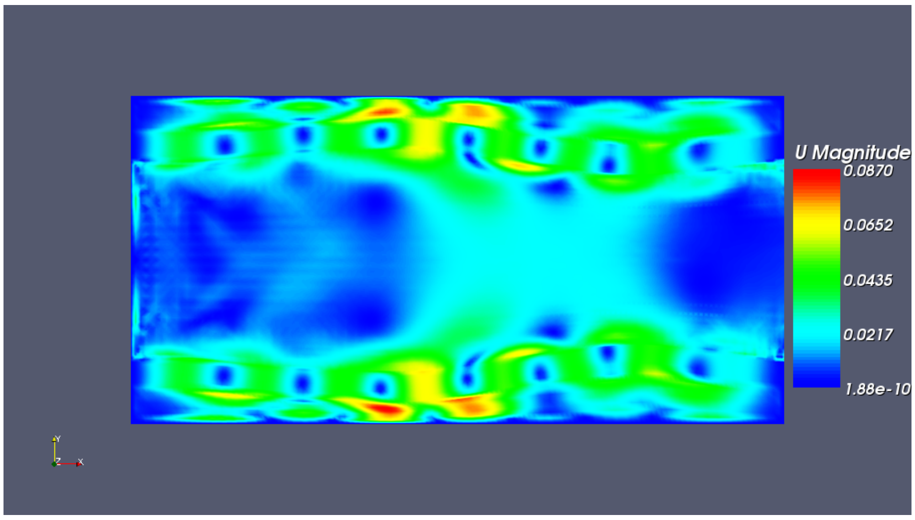


Figure 7.6. Approx4: The error is now largely present in the solid structure and is almost non-existent on the fluid domain except for in the fluid container around the outside of the solid vessel wall.

7.3.2 Pressure error

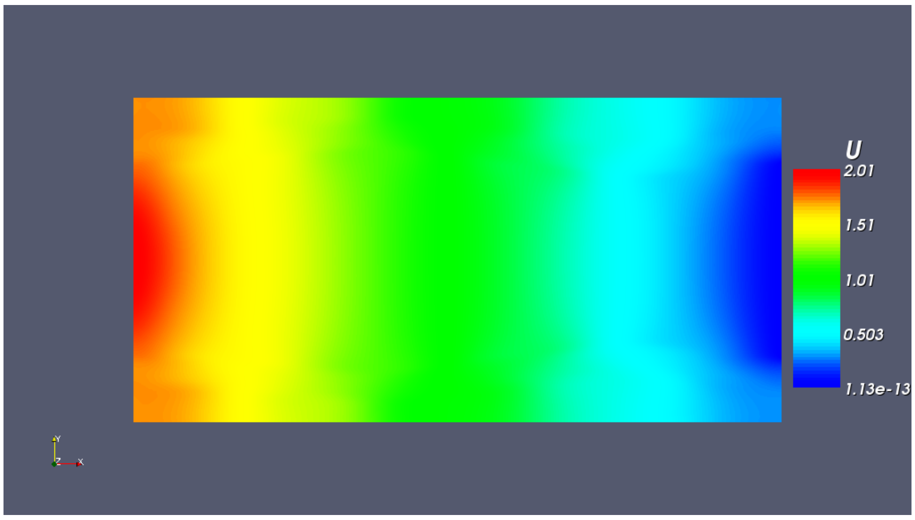


Figure 7.7. Approx1: The error is largest at the inflow boundary since we only define a boundary conditi. This is consistent with the velocity error from the same computation.

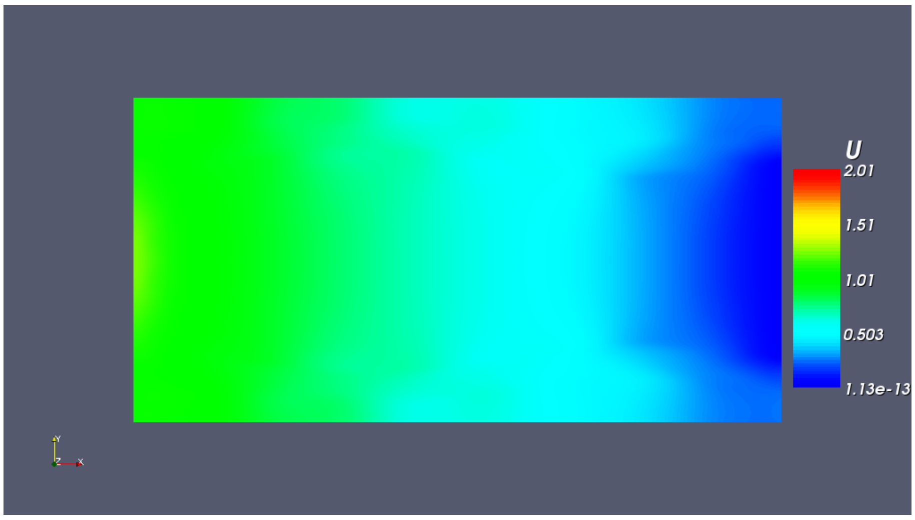


Figure 7.8. Approx2: As the number of degrees of freedom increases, the error is reduced nicely.

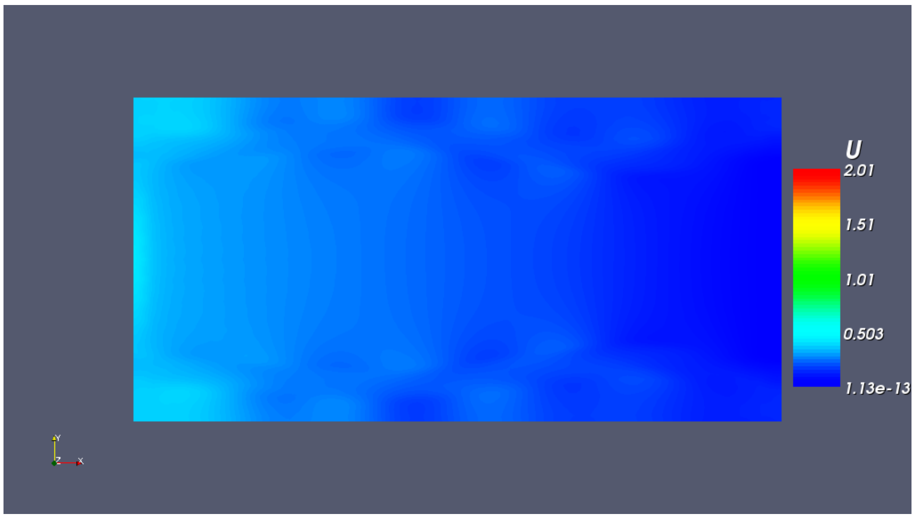


Figure 7.9. Approx3: We start to see the periodic pattern again which we saw before for the velocity solution on the solid domain. At this point, the error is already quite small compared to the previous approximations.

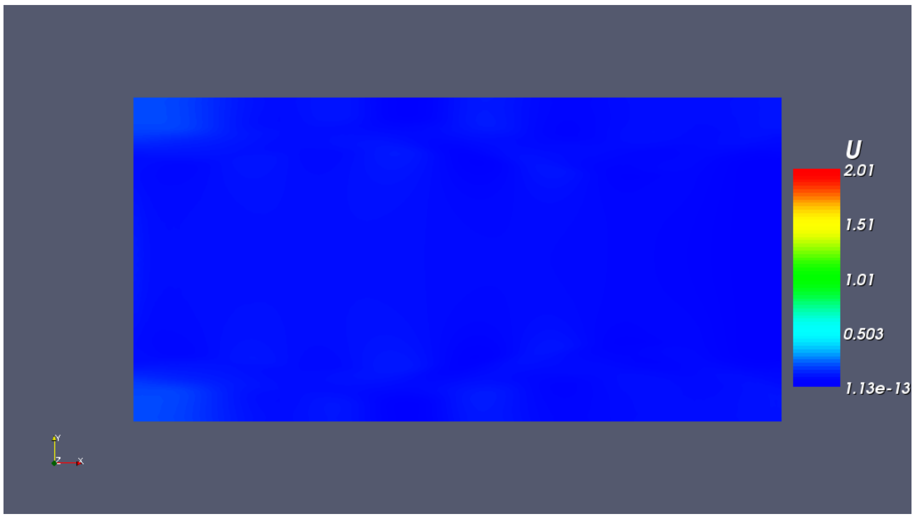


Figure 7.10. Approx4: It's hard to notice anything in this image but we can see that the left hand corners have an error that is slightly above 0.

7.4 Summary

As we could see from the previous figures, the error shows a generally decreasing trend for both the velocity and the pressure. The pressure solutions especially give us a nice indication of a 2nd order convergence method, while the velocity errors rather indicate 1st order. Perhaps if we increased the number of samples we would be able to say something more conclusive.

Chapter 8

Discussion

8.1 Improvements

Currently the solver is quite rigid in the way that we define parameters such as density, i.e., we cannot have structures with varying degrees of density but one. This could overall improve the versatility of the solver as well as the quality of the outcome due to being able to use more correct models. Right now, we use a fluid container to for the vessel to try to simulate the surrounding environment of the vessel. This is somewhat more realistic than not having anything around it, even better would be an extra structure layer surrounding the vessel that is similar to muscle tissue. As explained above, the inclusion of a fluid container has not only to do with accurately simulating blood flow, but also stabilizes the boundary stress.

One more inclusion that would make the solver perform better would be the ability to set stress boundary conditions of the surface of the tissue. Reason for this is similar to above, the vessel has to respond to something externally not just the fluid which flows inside the vessel.

A rather large drawback is the difficulty of defining the geometry. It is nearly impossible to use a manually created function to specify the regions of structure for a complex object. Within DOLFIN we have something called a MeshFunction where we can define a parameter for a mesh entity such as a cell or a vertex. Given the ability to draw the geometry using an external drawing program, one should be able to create a script which translates the different geometries from the external program into MeshFunction parameters. This would make it both easy and intuitive to define complex geometries. Another possibility would be to use what we already know about the geometry to define different structure regions. For example, we always know how far away we are from the boundary of an object, thus we could say that if the current region is within 2mm of the boundary, than this region is part of the structure.

Then there is also the question of being able to include compressible fluids. This is something that is being looked at right now for future problems in exhaust systems coupled to acoustics.

8.2 Conclusion

In this thesis, we have shown that one can combine the Navier-Stokes equation together with an elastic model such that one can solve a FSI problem by using just one solver. Our results have convinced us that this is an interesting pathway for new development and innovation in the area of fluid-structure interaction modeling and that the solver can be extended upon easily. So far we lack the information gained from empirical experiments and thus have not been able to fully test out our solver, but this should be done in the future.

Further, we seemed to have been able to sail past one of the troubling difficulties that arise in other FSI solvers, in that the solvers seem to become somewhat unstable when the density of the fluid is that of the structure.

References

[1] A. Quarteroni, *What mathematics can do for the simulation of blood circulation*, Dept. Math., Politecnico di Milano, January 16, 2006. Available at `http://mox.polimi.it/it/progetti/pubblicazioni/quaderni/mox77.pdf`.

[2] J. Donea, A. Huerta, J.-Ph. Ponthot, A. Rodriguez-Ferran, Arbitrary Lagrangian-Eulerian Methods, *Encyclopedia of Computational Mechanics - Volume 1: Fundamentals*, John Wiley and Sons Ltd, 2004.

[3] P. Råback, J. Ruokolainen, M. Lyly, E. Järvinen, *Fluid-Structure Interaction Boundary Conditions by Artificial Compressibility*, A paper presented in ECCOMAS CFD 2001, Swansea 4-7 September, 2001

[4] L. Formaggia, J. F. Gerbeau, F. Nobile, A. Quarteroni, On the coupling of 3D and 1D Navier-Stokes Equations for Flow Problems in Compliant Vessels, *Comput. Methods Appl. Mech. Engrg.*, 191:561-582, 2001

[5] J. Hron, S. Turek, *A Monolithic FEM Solver for ALE Formulation of Fluid Structure Interaction with Configurations for Numerical Benchmarking*, First International Conference on Computational Methods for Coupled Problems in Science and Engineering, Santorini 25-27 May, 2005

[6] J.-F. Gerbeau, M. Vidrascu, P. Frey, Fluid-Structure Interaction in Blood Flows on Geometries coming from Medical Imaging, *Computers and Structures*, 83:155-165, 2005

[7] J. Hoffman, C. Johnson, *Computational Turbulent Incompressible Flow*, Springer-Verlag, 2007.

[8] K. Eriksson, D. Estep, P. Hansbo, C. Johnson, *Computational Differential Equations*, Studentlitteratur, 1996.

[9] R. P. Vito, S. A. Dixon, Blood Vessel Constitutive Models - 1995-2002, *Annual Rev. Biomed. Eng.*, 5:413-439, 2003

[10] T. A. Dunn, *An Incompressible ALE Method for Fluid-Structure Interaction*, Proceedings from the NECDC 2004, Livermore 4-8 October, 2004.

[11] FEniCS Website, http://www.fenics.org visited February 3rd, 2007.

[12] FSI File Repository Website, http://www.mediamax.com/michael_stoeckli/Hosted/projects visited February 4th, 2007.

TRITA-CSC-E 2007:010
ISRN-KTH/CSC/E--07/010--SE
ISSN-1653-5715