FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jakub Hrnčíř

Projection method applied to modelling blood flow in cerebral aneurysm

Mathematical Institute of Charles University

Supervisor of the master thesis:	RNDr. Jaroslav Hron, Ph.D.
Study programme:	Mathematics
Study branch:	Mathematical Modelling in Physics and Technology

Prague 2016

Title: Projection method applied to modelling blood flow in cerebral aneurysm

Author: Jakub Hrnčíř

Institute: Mathematical Institute of Charles University

Supervisor: RNDr. Jaroslav Hron, Ph.D., Mathematical Institute of Charles University

Abstract: This thesis is motivated by a problem of cerebral aneurysms, which are abnormal bulges on the arteries which supply blood for our brain. These aneurysms can rupture and cause death or permanent neurological deficits. To study the evolution of aneurysms and assess the risk of rupture, mathematical modelling might be used to compute otherwise unobtainable information about blood flow inside the aneurysm. For this reason it is essential to be able to model blood flow in sufficiently high resolution. A goal of this thesis was to implement standard projection method for the solution of unsteady incompressible Navier-Stokes equations using the free finite element software FEniCS to create a working code adjusted to the need of this particular application. The incremental pressure correction scheme was chosen. Various shortcomings of this method are described and a proper choice of boundary conditions and other implementation issues are discussed. A comparison of computed important hemodynamic indicator wall shear stress using new and previously used solution approach are compared. A test of the new code for parallel efficiency and performance on finer meshes for a real medical case was conducted.

Keywords: projection method, incompressible Navier Stokes, pulsatile flow, cerebral aneurysm

First of all, I would like to thank my supervisor Jaroslav Hron for good guidance, numerous consultations and lots of time spent in my support. I would like to thank my colleague Helena Švihlová for providing me with real aneurysm meshes and introducing me to various topics concerning the medical application and some related technical issues. My thanks goes also to other colleagues and teacher for advice given, namely to Jan Blechta, who provided me useful advice about the FEniCS software and helped me to deal with several issues. And I also thank my parents for the support during the years of my studies.

Contents

List of abbreviations and used mathematical notation

(S1)-(S4)	four steps of projection method; defined in section2.2.4
IPCS	incremental pressure correction scheme
SUPG	streamline-upwind Petrov-Galerkin
BC	boundary conditions
WSS	wall shear stress
$\boldsymbol{u}$	velocity field
$\boldsymbol{u}_*$	tentative velocity
$\boldsymbol{u}_{\text{ext}}$	extrapolated velocity
$\boldsymbol{u}_D$	prescribed boundary velocity (where Dirichlet BC apply)
ρ	density
p	pressure field (possibly divided by ρ)
μ	molecular viscosity
ν	kinematic viscosity
$\mathbb{T}$	Cauchy stress tensor
δt	used time step
h_K	local mesh size
Ω	computational domain
$\boldsymbol{n}$	unit outer normal to the boundary of Ω
Γ_{in}	inflow part of the boundary of Ω
Γ_{wall}	vessel wall part of the boundary of Ω
Γ_{out}	outflow part of the boundary of Ω
Γ_D	part of the boundary of Ω with Dirichlet BC
$\mathbf{V}', Q$	infinite dimensional function spaces
$\mathbf{V}_h, Q_h$	finite dimensional function spaces
$\boldsymbol{v}, q$	test functions
a_n	bilinear form in weak formulation for projection step (Sn)
b_n	linear form in weak formulation for projection step (Sn)
$D(\boldsymbol{u})$	symmetric part of gradient of $\boldsymbol{u}$
$\boldsymbol{\varphi}, \psi$	finite element basis functions
$\boldsymbol{u}_h, p_h$	finite element velocity and pressure approximations
$\mathbf{x}, \mathbf{y}$	vectors
A_n	matrix for projection step (Sn)
$(\cdot, \cdot)$	inner product
τ	stabilization parameter

Introduction

A aneurysm is blood-filled bulge in wall of a blood vessel. An aneurysm can be formed on any vessel and location of the body, but usually they are formed on arteries at specific locations. This thesis is inspired by our cooperation with Neurosurgery clinic of Masaryk's Hospital in Ústí nad Labem, where brain aneurysms are studied.

About 2 percent of the general population has or will develop a cerebral aneurysm. These aneurysms may rupture, the incidence of rupture is approximately 1 in 10 000 people per year. Ruptured brain aneurysms are fatal in about 40 % of cases. Of those who survive, about 66 % suffer some permanent neurological deficit (NINDS; BAF).

Aneurysms can be detected using computational tomography (CT) or magnetic resonance imaging technologies. Detected aneurysms can be treated to prevent possible rupture, but treatment itself is not without risks. Therefore doctors need to evaluate the probability of rupture, so only aneurysms with serious risk of rupture are treated.

Every aneurysm is different, and the risk of rupture may depend on many factors. As current measurement methods and physical experiments are very limited and do not provide enough information to assess the risk, mathematical modelling can be used to provide more information about the blood flow. Typically, all we have is the geometry of veins and aneurysm provided by medical imaging technology.

Solving a pulsative blood flow in complex geometry is not an easy task. To improve the computational efficiency of our current Navier-Stokes solver based on direct fully coupled method, we try to find and implement a more efficient numerical strategy in order to be able to get reasonably accurate results for higher quality meshes in practical computational time.

We have chosen to use a projection method. We implement it using the FEniCS software. Projection methods can provide efficiency at the cost of creating additional "splitting errors". Despite the long usage of the projection methods, many issues concerning the nature of the splitting errors and use of different boundary conditions are still unclear or undecided. We tried to describe those errors and to explain some issues related to the boundary conditions and other implementation problems.

This thesis has a following structure: In the first chapter, we describe the mathematical model used and the context of our work. In the second chapter we review relevant projection methods, schemes and some theoretical results. In the third, main chapter, we describe the implementation of our chosen method and discuss the different options and various implementation issues. In the last chapter we provide results obtained from testing of the created code. We tested its performance on finer meshes and its parallel efficiency. We compared the new code with a previous approach.

1. Modelling blood flow

1.1 Chosen mathematical model

We want to model a blood flow in real geometry of an aneurysm and the adjacent vessels. The size of the geometry is usually around 2 or 3 cm. Because aneurysms are often formed near or directly on places of branching of vessels, the geometry can have one or more inflows as well as outflows.

The walls of human arteries and surrounding mass are viscoelastic and interact with the blood flow. This interaction can be quite strong, as is for example in the case of aorta, which expands and shrinks considerably during the blood flow cycle. But in the case of cerebral arteries, we assume that the movement of vessel wall is negligible so we will compute in fixed geometry. This is a simplification, but computing with viscoelastic vessel walls is much more difficult problem and also we lack information about material properties of mass around aneurysm. We will denote our computational domain Ω. Our Ω is bounded, Lipschitz, and obviously not convex. We will mark different parts of its boundary $\partial\Omega$ in following way: Γ_{wall}, Γ_{in}, Γ_{out} for vessel walls, all inflows and all outflows respectively.

To obtain mesh suitable for computation from usually inaccurate and coarse data from medical scanners is a problem for itself. In this work, we used meshes provided by my colleague Švihlová (2013, master thesis).

We will model blood flow as a time dependent pulsatile flow using Navier-Stokes equations:

$$\rho \boldsymbol{u}_t + \rho \left[\nabla \boldsymbol{u}\right] \boldsymbol{u} = \operatorname{div}\left(\mathbb{T}\right) \tag{1.1a}$$

$$\operatorname{div} \boldsymbol{u} = 0 \tag{1.1b}$$

These equations model a flow of incompressible fluid with velocity vector $\boldsymbol{u}$, time derivative $\boldsymbol{u}_t$ and density ρ. The $\mathbb{T}$ stands for stress tensor, which is given by constitutive relation

$$\mathbb{T} = -p\mathbb{I} + \mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right), \tag{1.2}$$

where p stands for pressure. In this work, we will use Newtonian fluid, that means molecular viscosity μ will be constant. Because blood is not a Newtonian fluid, we will want to leave open possibility for possible future modification of the model to non-constant viscosity.

We will compute with no external forces, as effect of gravity is negligible in comparison with inertial forces. Also, in the context of incompressible model and forcing inflow velocity profile, the effect of external force is often mainly a modification of pressure.

By substituting the constitutive relation (1.2) into the momentum equation (1.1a), dividing by the constant density ρ and rearranging the terms we get a form

$$\boldsymbol{u}_t + \left[\nabla \boldsymbol{u}\right] \boldsymbol{u} + \nabla p - \operatorname{div}\left(\nu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right)\right) = \boldsymbol{0} \tag{1.3a}$$

$$\operatorname{div} \boldsymbol{u} = 0, \tag{1.3b}$$

where $\nu = \mu/\rho$ is kinematic viscosity. We also used simplification $p \overset{\text{def}}{=} p/\rho$.

There are more possibilities for improving the model, such as using viscoelastic model or considering blood as a mixture of different components. But our concern in this work will be to apply more efficient and scalable numerical method.

Equations must be completed by adding suitable boundary and initial conditions. Pulsation of the flow is achieved by prescribing time-dependent oscillating velocity profiles on the inflows Γ_{in} simulating possible real inflow. For simplicity we assume that flow has a period 1 second (using average heartbeat 60 beats per minute). On walls we prescribe zero velocity (no-slip) condition. We will use notation $\Gamma_D = \Gamma_{\text{in}} \cup \Gamma_{\text{wall}}$ for part of the boundary with Dirichlet boundary conditions, and $\boldsymbol{u}_D$ will be our prescribed velocity on Γ_D.

On outflows we will use some sort of neutral condition. Discussion of its choice will be given in section 3.4.1. We will use zero initial condition. We will get pulsatile solution by computing over one or more blood flow cycles and than using data from the next cycle, so the effect of initial condition wears off.

1.2 Hemodynamic indicators

Velocity and pressure obtained from model can be used to compute various hemodynamic indicators. Many of different indicators are derived from spatial and temporal distribution of wall shear stress (WSS)

$$\tau_{wall} = (I - \boldsymbol{n} \otimes \boldsymbol{n})\, \mathbb{T}\boldsymbol{n},$$

defined as a tangential part of stress at vessel walls with outer unit normal $\boldsymbol{n}$. Another option is computing energy loss associated with an aneurysm (Qian et al., 2011). What are the right indicators and values for assessing probability of growth and rupture of different types of aneurysm is still subject of research (Can and Du, 2016).

1.3 Former approach in our group

One possible approach already implemented and used in our group is based on computing problem discretized by finite element method and solving coupled nonlinear Navier-Stokes equations using generalized Newton method. This leads to solving several linear systems in each time step. These linear system are solved by direct solvers (using MUMPS library).

Because of high changes of flow velocity during cycle, we have to use relatively small time step (0.01 s). We need to compute several cycles to make sure initial condition have negligible effect on solution. We also want to model flow using quite a fine mesh, so number of degrees of freedom is high (several million).

Although this approach gives quite reliable results when applicable, it has substantial drawbacks. If we want to compute on finer meshes, dimension of our problem becomes too high to use a direct method (due to amount of time and memory needed). Also efficiency of parallel computation is quite limited.

In this thesis, we will refer to this approach as to “the direct method”.

1.4 Considered alternatives

For such a large system it would be better if we could find a way to use iterative rather then direct methods. Iterative methods need less memory. Iterative methods can also lead to sufficiently accurate solution in lesser amounts of time, and are more effective when run in parallel.

We can consider using Krylov iterative methods (such as CG, GMRES) on the coupled Navier Stokes system, but their direct use is impractical without good preconditioning, which is not easily obtained for coupled Navier-Stokes equations. Another option is to decouple these equations and use some operator splitting technique such as projection method. This approach is easier to grasp and easier to implement. Therefore, we chose the latter for practical reasons.

1.5 Goals of thesis

In this work we tried to find and implement better computational strategy, that would allow us to compute our problem on finer meshes to get reasonably accurate solution in same or less time. We wanted to solve all technical problems associated with application of new numerical approach to complex geometries obtained from real medical data. We wanted to assess its inaccuracy, especially with respect to computation of wall shear stress.

2. Projection methods

In this chapter, we give a short review of basic idea and schemes of projection methods needed for understanding main part of the work. Projection methods are used from 1960s, so there are many works about them. Our main source was an overview by Guermond and Shen (2006).

2.1 Overview

2.1.1 Pressure and velocity correction methods

When solving incompressible Navier-Stokes equations, we want to find velocity field that satisfies momentum equation (1.3a) under a constraint of zero divergence (1.3b). The pressure in this context is not a thermodynamic variable. It is an implicit variable that adjust itself just so the zero divergence constraint is satisfied.

The incompressible Navier-Stokes equations are a saddle point problem, its corresponding matrix (arising from FEM discretization) is indefinite. Such problems are difficult to solve. Projection methods evade solving saddle point problem by splitting equations and computing in each time step more simpler problems, typically a convection-diffusion equation for velocity and a Poisson problem for pressure. We first compute a tentative solution of one equation, and then we compute a correction using a second equation. The choice which equation is “first” separates two types of projection methods. One type is “pressure correction method”, in which we compute a tentative compressible velocity field satisfying the momentum equation, and in the second step we compute “pressure” as a correction to an incompressible velocity field. A second type is “velocity correction method”, when we start by computing pressure and associated incompressible velocity field and then we correct it to satisfy the momentum equation.

We will use only pressure correction methods, so we restrict our review only to this type.

2.1.2 Idea of projection

The core idea of projection method rests on the Helmholtz decomposition.

Let us first state some function spaces. Let $L^2(\Omega)^3$ denote a space of Lebesgue square integrable vector functions,

$$\boldsymbol{H}_0(\mathrm{div},\Omega) = \left\{\boldsymbol{v} \in L^2(\Omega)^3, \boldsymbol{v} \text{ admits weak divergence}, \boldsymbol{v}\cdot\boldsymbol{n}|_\Gamma = 0\right\}$$

$$\boldsymbol{H} = \{\boldsymbol{v} \in \boldsymbol{H}_0(\mathrm{div},\Omega), \mathrm{div}\,\boldsymbol{v} = 0\}$$

The theorems 2.6 and 2.7 in Girault and Raviart (1986) combined gives the orthogonal decomposition of $L^2(\Omega)^3$:

Theorem 1 (The Helmholtz decomposition)**.** *Let Ω is connected. Then $L^2(\Omega)^3 = \boldsymbol{H} \oplus \boldsymbol{H}^\perp$, where $\boldsymbol{H}^\perp = \{\nabla q, q \in H^1(\Omega)\}$.*

This means that any vector field we can get (e. g. a weak solution from Finite element method) can be decomposed orthogonally into a field with zero divergence and a second field, which has a potential. The correction of the tentative to incompressible velocity is an orthogonal projection, hence the name of the operator splitting approach. Computed "pressure" is a potential of correction vector field.

The catch here is that our solution cannot satisfy condition $\boldsymbol{v} \cdot \boldsymbol{n}|_\Gamma = 0$ on the outflow boundary[1]. This is probably one of the reasons for complications concerning outflow boundary conditions that we will see later.

2.2 Basic schemes

Now we will demonstrate basic projection scheme as proposed by Chorin (1967) and Témam (1969). We will look how error introduced by splitting of equation is manifested. Then we will show two modifications used to reduce splitting error.

As the convective and diffusion terms does not change we will use following notation:

$$L(\boldsymbol{u}) = \operatorname{div}\left(\nu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\top\right)\right)$$
$$N(\boldsymbol{u}) = [\nabla \boldsymbol{u}]\, \boldsymbol{u}$$

We will write all schemes in this section using backward Euler time discretization for simplicity, but other discretizations (such as Crank-Nicholson, which we will use later) are also possible. We will assume δt is chosen constant size of time step.

2.2.1 Chorin's scheme

To get the first step of Chorin's pressure correction scheme, we simply drop the pressure term in the momentum equation (1.3a) and do not consider the zero divergence condition. We want to compute a tentative velocity $\boldsymbol{u}_*^{k+1}$ satisfying

$$\frac{\boldsymbol{u}_*^{k+1} - \boldsymbol{u}^k}{\delta t} + N\left(\boldsymbol{u}_*^{k+1}\right) - L\left(\boldsymbol{u}_*^{k+1}\right) = \boldsymbol{0} \text{ in } \Omega,\ \boldsymbol{u}_*^{k+1} = \boldsymbol{u}_D \text{ on } \Gamma_D. \tag{2.1}$$

In the second step we want to find $\left(\boldsymbol{u}^{k+1}, p^{k+1}\right)$ such that ∇p^{k+1} is correction of $\boldsymbol{u}_*^{k+1}$ to incompressible velocity field $\boldsymbol{u}^{k+1}$. We get decomposition

$$\boldsymbol{u}_*^{k+1} = \boldsymbol{u}^{k+1} + \delta t \nabla p^{k+1}, \tag{2.2}$$
$$\operatorname{div} \boldsymbol{u}^{k+1} = \boldsymbol{0}. \tag{2.3}$$

This is a saddle point problem, but much easier than (1.3). In this case we can separate the variables. We apply div to (2.2), thus eliminating $\boldsymbol{u}^{k+1}$ from the equation and getting Poisson problem for p^{k+1}

$$\Delta p^{k+1} = \frac{1}{\delta t} \operatorname{div} \boldsymbol{u}_*^{k+1}. \tag{2.4}$$

[1]It is not a problem for the Dirichlet part of the boundary, as we will seek our weak solution function in space satisfying the homogeneous Dirichlet boundary conditions. For explanation see section 3.4.1

After we compute p^{k+1}, we can get a corrected velocity by simple update:

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}_*^{k+1} - \delta t \nabla p^{k+1}. \tag{2.5}$$

Note that when we want to compute a weak solution p of (2.4), we have to use an implicit boundary condition $\frac{\partial p}{\partial \boldsymbol{n}} = ?$. As we do not know proper values for the pressure gradient, we should use

$$\frac{\partial p}{\partial \boldsymbol{n}} = 0. \tag{2.6}$$

This boundary condition is satisfied only weakly by numerical solution of (2.4). Note also that tangent derivatives of p are generally non-zero.

As we correct our tentative velocity $\boldsymbol{u}_*$ by ∇p satisfying (2.6), it means that the correction does not change normal velocity on the inflow and the outflow. But $\boldsymbol{u}_*$ is correct only on the inflow. The correction does change the tangential component of the velocity on the vessel walls (in general).

The splitting error manifests itself in two main ways. The first way is that $\boldsymbol{u}^{k+1}$ does not satisfy no-slip condition in tangential direction to the boundary. More problematic is the second error, a non-physical pressure gradient $\frac{\partial p}{\partial \boldsymbol{n}} = 0$ on outflow: it creates a boundary layer and in our setting it can lead to global error. Consider these circumstances: We use zero initial condition, so in the first time step, the tentative velocity on the outflow will be almost zero independently of the inflow velocity condition. Because there is almost no correction on the outflow, this error can propagate many time steps. While we might think that the corrected velocity should be close to incompressible, we see almost no outflow independently of the rate of inflow. That contradicts incompressibility. We will discuss more choices of boundary conditions and possible ways to solve this problem later (see section 3.4.2).

2.2.2 Incremental pressure correction scheme (IPCS)

To improve Chorin's scheme, we can use some extrapolation of previously computed pressure in the first step. The simplest option is to use just p^k. We can get better tentative velocity and lesser splitting error by modifying equation (2.1) to

$$\frac{\boldsymbol{u}_*^{k+1} - \boldsymbol{u}^k}{\delta t} + N\left(\boldsymbol{u}_*^{k+1}\right) + \nabla p^k - L\left(\boldsymbol{u}_*^{k+1}\right) = \boldsymbol{0} \text{ in } \Omega,\ \boldsymbol{u}_*^{k+1} = \boldsymbol{u}_D \text{ on } \Gamma_D. \tag{2.7}$$

The remaining steps must be modified accordingly. From the perspective of the projection method, we compute a partially projected tentative velocity in the first step, so in the second step we compute only a correction of that projection. After similar manipulation as in Chorin's scheme, we get

$$\Delta\left(p^{k+1} - p^k\right) = \frac{1}{\delta t} \operatorname{div} \boldsymbol{u}_*^{k+1} \tag{2.8}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}_*^{k+1} - \delta t\left(\nabla p^{k+1} - \nabla p^k\right). \tag{2.9}$$

Now the implicit boundary condition for pressure is $\frac{\partial p^{k+1}}{\partial \boldsymbol{n}} = \frac{\partial p^k}{\partial \boldsymbol{n}}$. The splitting error is therefore proportional only to the difference $\frac{\partial p^{k+1}}{\partial \boldsymbol{n}} - \frac{\partial p^k}{\partial \boldsymbol{n}}$, instead of being scaled by $\frac{\partial p^{k+1}}{\partial \boldsymbol{n}}$. To illustrate this effect, it means that improved formulation has zero splitting error for constant velocity flow. Also, using lower time step now significantly reduce splitting error.

This modification brings up following question: p^k was computed as a potential of correction of tentative velocity. Is it also the right "pressure" enforcing incompressibility of corrected velocity?

$\frac{\partial p^{k+1}}{\partial \boldsymbol{n}} = \frac{\partial p^k}{\partial \boldsymbol{n}} = 0$ on the inflow is right in the sense that the normal inflow velocity does not need correction (it is forced by the Dirichlet boundary condition). But it is inconsistent with the velocity field $\boldsymbol{u}^{k+1}$ in the momentum equation, as inflow velocity should be accompanied by non-zero normal pressure gradient. That might cause problems in next tentative velocity computation and cause some oscillations near the inflow boundary. This inconsistency can be removed by following modification.

2.2.3 Rotational scheme

In this modification proposed by Timmermans et al. (1996), we compute the potential of a correction, now denoted ϕ^{k+1}, in the same way as in previous scheme. But another correction of ϕ^{k+1} to p^{k+1} is added.

$$\Delta\left(\phi^{k+1} - p^k\right) = \frac{1}{\delta t}\operatorname{div}\boldsymbol{u}_*^{k+1} \tag{2.10}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}_*^{k+1} - \delta t\left(\nabla\phi^{k+1} - \nabla p^k\right) \tag{2.11}$$

$$p^{k+1} = \phi^{k+1} - \nu\operatorname{div}\boldsymbol{u}_*^{k+1} \tag{2.12}$$

Explanation of the scheme is given in Guermond and Shen (2006) or Timmermans et al. (1996). It is important to note that this correction does not mend the problem of loss of incompressibility due to unphysical pressure gradient on outflow boundary.

2.2.4 Summary

In our work, we will use incremental pressure projection scheme (IPCS) (2.7), (2.8), (2.9) and its rotational modification (2.7), (2.10), (2.11), (2.12).

The projection methods split each time step to three or four subproblems:

(S1) Compute a tentative velocity from modified momentum equation.

(S2) Compute the Poisson problem to get a correction potential (and pressure in case of IPCS).

(S3) Compute a corrected velocity.

(S4) Compute a pressure from the correction potential (only in the rotational scheme).

We will use the (S#) designation when we will need to refer to the projection method steps throughout the thesis.

In this section, we have shown three types of errors created by using projection scheme:

1. The corrected velocity does not satisfy tangential boundary conditions on the vessel walls. This error is scaled with the size of the time step for incremental scheme. Further discussion of its effect on wall shear stress (WSS) computation will be given in section 3.6.1.

2. The unphysical condition $\frac{\partial p^{k+1}}{\partial \boldsymbol{n}} = 0$ on the outflow is problem for all given schemes and it will be resolved in section 3.4.2.

3. Inconsistency of the correction potential and the pressure on the inflow due to $\frac{\partial p^{k+1}}{\partial \boldsymbol{n}} = 0$ condition is treated by an additional correction step in the rotational scheme.

2.3 Available theoretical results

Here we give some theorems concerning convergence of projection schemes. Assumptions common for all theorems are: Ω is Lipschitz, only homogeneous Dirichlet boundary conditions on velocity $\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{0}$ are applied, solution $(\boldsymbol{u}, p)$ of the original problem (1.3) is sufficiently smooth in time and space, solutions are acquired using exact arithmetic, multi-step schemes are properly initialized.

We will use $\boldsymbol{u}_*^{\delta t}$, $\boldsymbol{u}^{\delta t}$ and $p^{\delta t}$ to denote the sequences of solutions of tentative velocity, corrected velocity and pressure in discrete time steps using some constant time step δt.

Theorem 2. *(Shen, 1996) Under the assumptions above, a solution of incremental pressure scheme formulation* (2.7)*,* (2.8)*,* (2.9) *satisfies estimate:*
$||\boldsymbol{u} - \boldsymbol{u}_*^{\delta t}||_{\ell^\infty(H^1(\Omega)^3)} + ||p - p^{\delta t}||_{\ell^\infty(L^2(\Omega))} < C\delta t.$

Theorem 3. *(Guermond and Shen, 2004) Under assumptions above, a solution of rotational modification of incremental pressure scheme* (2.7)*,* (2.10)*,* (2.11)*,* (2.12) *satisfies estimate:*
$||\boldsymbol{u} - \boldsymbol{u}^{\delta t}||_{\ell^2(H^1(\Omega)^3)} + ||\boldsymbol{u} - \boldsymbol{u}_*^{\delta t}||_{\ell^2(H^1(\Omega)^3)} + ||p - p^{\delta t}||_{\ell^2(L^2(\Omega))} < C\delta t^{3/2}.$

From these theorems we can only deduce that corrected velocity, tentative velocity and pressure should all converge to the solution of original problem for $\delta t \to 0$. Limitations of significance of these results in our case are strong assumptions, some of which (especially the choice of boundary conditions and the exact computation of solutions) are unmet in our problem. Also the theorems only state asymptotic behaviour of the schemes, which might not be visible for our choice of meshes and time steps. And finally, we will use some stabilization techniques to solve our problem and that change of scheme is not accounted for in cited analysis.

3. Implementation

3.1 Problems

We used two main problems to test our implementation and different aspects and difficulties associated with the projection methods.

As a “test problem”, we have chosen Womersley flow in a cylindrical pipe. Womersley flow is analytic solution for laminar pulsatile flow of incompressible Newtonian liquid. It is the most difficult problem similar to our application that has an analytical solution, which allows us to monitor spatial distribution of the error. In analogy with setting of our main problem, we will enforce the flow by prescribing analytic velocity profile on inflow.

As a second, “real problem”, we take flow in geometry obtained from real medical data (see figure 3.1).

3.2 Implementation in FEniCS

To get an approximate solution to our chosen PDEs, we want to use Finite element method (FEM). To get actually numerically computable problem, we will follow these steps:

1. Choose time discretization of our schemes.
2. Derive weak formulation of semi-discrete scheme.
3. Choose finite-dimensional function spaces using FEM to discretize in space.

To implement the code for projection method, we use the (FEniCS, release 1.7.0), a collection of free software with an extensive list of features for automated, efficient solution of differential equations.

We chose FEniCS for a number of reasons. It is a tool which we can use to solve our problems numerically without having to implement every detail of FEM for our particular application from scratch. It is also suitable to build modular code: we can use the same code for solving both our test problem and our real problem, or we can return back to direct solver strategy by replacing only our specification of solvers. The problem itself is entered using symbolic notation of our weak formulation, and pre-programmed tricks and efficient coding techniques are generated by FEniCS automatically. More specifically, FEniCS allow us to use automated domain decomposition technique to solve our problem on parallel architectures with only minor changes in our code. FEniCS achieves this by using PETSc linear algebra backend. Unlike with commercial software, we still have quite high degree of control above what is going on in the computations.

For information about the access to the created code and example of basic usage see Appendix - the code.

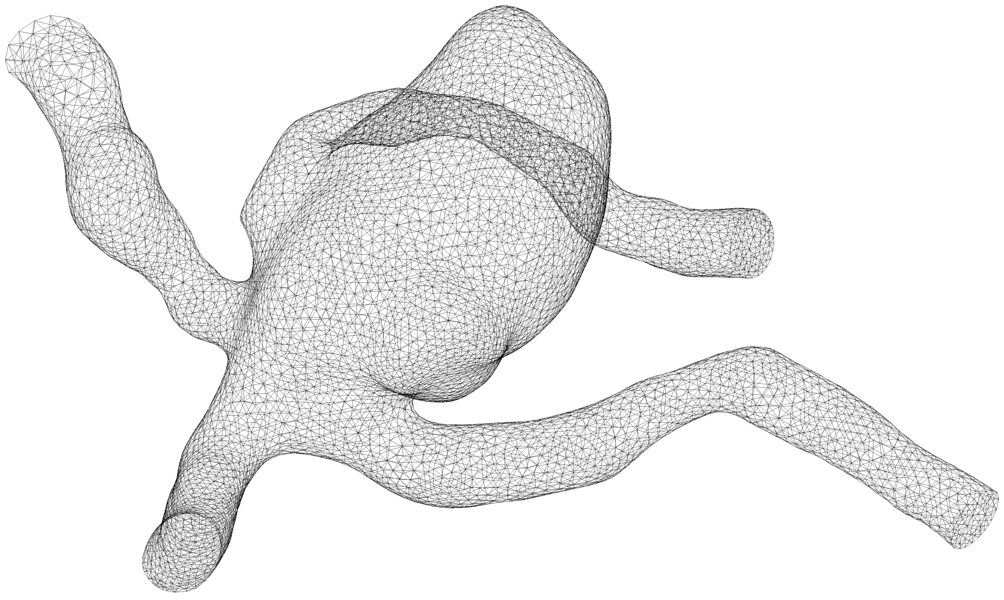

Figure 3.1: Real mesh. The two inflow vessels are on the left, the two outflow vessels are on the right.

3.3 Time discretization

Let δt be our chosen constant time step. We will show time discretization for incremental pressure correction scheme. In the first step of projection method, we will use Crank-Nicholson scheme.

As we are using projection method and we will need to compute with small time steps, we do not want to compute nonlinear problem. We have to choose some form of linearization of the convective term. The easiest way for solver would be to treat this term explicitly $\left(\left[\nabla \boldsymbol{u}^k\right] \boldsymbol{u}^k\right)$ but this approach require too small time step to be stable.

Therefore we will use following semi-explicit scheme

$$[\nabla \boldsymbol{u}]\, \boldsymbol{u} \approx \left[\nabla \boldsymbol{u}^{k+\frac{1}{2}}\right] \boldsymbol{u}_{\text{ext}},$$

where $\boldsymbol{u}_{\text{ext}}$ is velocity extrapolated from previous time step. We will use two step Adams-Bashfort scheme for the extrapolation

$$\boldsymbol{u}_{\text{ext}} = \frac{3\boldsymbol{u}^k - \boldsymbol{u}^{k-1}}{2}. \tag{3.1}$$

Using notation

$$\boldsymbol{u}_*^{k+\frac{1}{2}} = \frac{\boldsymbol{u}_*^{k+1} + \boldsymbol{u}^k}{2}$$

for Crank-Nicholson scheme and

$$D\left(\boldsymbol{u}\right) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \left(\nabla \boldsymbol{u}\right)^\top\right)$$

for the symmetric part of gradient, complete time-discretized scheme for the first step (S1) is

$$\frac{\boldsymbol{u}_*^{k+1} - \boldsymbol{u}^k}{\delta t} + \left[\nabla \boldsymbol{u}_*^{k+\frac{1}{2}}\right] \boldsymbol{u}_{\text{ext}} + \nabla p^k - \operatorname{div}\left(2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}}\right)\right) = \boldsymbol{0}. \tag{3.2}$$

Note that this is essentially a vector variant of convection-diffusion problem.

In the steps (S2), (S3), (S4) of projection method there is no time discretization. We will use schemes as they are stated in the previous chapter.

3.4 Weak formulation

In this section, we want to derive weak formulations of our problems. That means we want to find some Hilbert function spaces V and V' for solutions and test functions, and derive for each equation a bilinear form $a : V \times V' \to \mathbb{R}$ and linear form $b : V' \to \mathbb{R}$ in such a way that we can always find a weak solution as unique function $u \in V$ such that $a(u, v) = b(v) \quad \forall v \in V'$.

3.4.1 Weak formulation of tentative velocity step (S1)

The Dirichlet boundary conditions are "essential". That means they are satisfied exactly via choice of a solution space. However, space of functions that satisfy non-homogeneous Dirichlet boundary conditions is not linear (it does not contain zero function). Therefore we will seek a solution in form $\boldsymbol{u}_*^{k+1} = \boldsymbol{u}_{V'} + \boldsymbol{v}_D$, where $\boldsymbol{u}_{V'}$ lies in the test function space V', which is a linear space of functions with zero on part of boundary with Dirichlet boundary condition, and $\boldsymbol{v}_D \in V$ is some function satisfying our Dirichlet boundary conditions. For simplicity, we can from now on seek only $\boldsymbol{u}_{V'}$ denoted again by $\boldsymbol{u}_*^{k+1}$ and take $V = V'$.

We multiply the scheme (3.2) by a vector test function $\boldsymbol{v}$. We will denote

$$(A,\, B) = \int_\Omega A : B \mathrm{d}x, \qquad (\boldsymbol{u},\, \boldsymbol{v}) = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} \mathrm{d}x, \qquad (p,\, q) = \int_\Omega pq \mathrm{d}x$$

the inner products of tensor, vector and scalar functions. Then

$$\left(\frac{\boldsymbol{u}_*^{k+1} - \boldsymbol{u}^k}{\delta t} + \left[\nabla \boldsymbol{u}_*^{k+\frac{1}{2}} \right] \boldsymbol{u}_{\text{ext}},\, \boldsymbol{v} \right) + \left(\nabla p^k,\, \boldsymbol{v} \right) - \left(\operatorname{div} 2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}} \right),\, \boldsymbol{v} \right) = \boldsymbol{0}. \tag{3.3}$$

We need to redistribute function derivations evenly between solution and test functions. We formally use integration by parts on diffusion and pressure terms. We get

$$\begin{aligned} \left(\frac{\boldsymbol{u}_*^{k+1} - \boldsymbol{u}^k}{\delta t} + \left[\nabla \boldsymbol{u}_*^{k+\frac{1}{2}} \right] \boldsymbol{u}_{\text{ext}},\, \boldsymbol{v} \right) - \left(p^k,\, \operatorname{div} \boldsymbol{v} \right) + \left(p^k \boldsymbol{n},\, \boldsymbol{v} \right)_{\partial\Omega} + \\ + \left(2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}} \right),\, \nabla \boldsymbol{v} \right) - \left(2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}} \right) \boldsymbol{n},\, \boldsymbol{v} \right)_{\partial\Omega} = \boldsymbol{0}, \end{aligned} \tag{3.4}$$

where $(\cdot,\, \cdot)_\Gamma$ denotes integral over boundary Γ. As $\boldsymbol{v}$ is zero on Γ_{in} and Γ_{wall}, boundary integrals vanish outside of outflow boundary Γ_{out}.

Outflow boundary condition for velocity

To deal with the outflow boundary integrals, we have to specify some boundary conditions on the outflow. We cannot use Dirichlet conditions on outflow as the outflow velocity profile should be determined by the solution inside our computational domain. Therefore we want to prescribe some sort of "neutral" condition in the sense that it will minimally interfere or collide with the flow inside domain. To develop such a condition for the projection method, we started with a typical neutral conditions for the coupled equations.

The most simple neutral condition is the no-stress condition $\mathbb{T}\boldsymbol{n} = \boldsymbol{0}$. If we recall the formula for stress (1.2),

$$\mathbb{T}\boldsymbol{n} = -p\boldsymbol{n} + 2\mu D\left(\boldsymbol{u}\right)\boldsymbol{n} \tag{3.5}$$

we see this condition couples pressure and velocity together. As pressure is not an unknown in first step of projection method, we need to find a correct analogy for the projection scheme. A naive option would be

$$2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}}\right)\boldsymbol{n} = p^k\boldsymbol{n}, \tag{3.6}$$

but this means that the value of the computed pressure on the boundary determines the gradient of velocity on the outflow. As pressure is determined up to an arbitrary additive constant (see section 3.4.2), using this form of boundary condition leads to major oscillations. The correct outflow boundary condition is

$$2\nu D\left(\boldsymbol{u}_*^{k+\frac{1}{2}}\right)\boldsymbol{n} = \boldsymbol{0}. \tag{3.7}$$

Using this condition, only the boundary integral with pressure remains in the equation (3.4).

Now we move all terms without the unknown $\boldsymbol{u}_*^{k+1}$ to the right hand side. The remaining left hand side becomes our bilinear form

$$a_1(\boldsymbol{u}_*^{k+1},\, \boldsymbol{v}) = \left(\frac{\boldsymbol{u}_*^{k+1}}{\delta t} + \frac{1}{2}\left[\nabla\boldsymbol{u}_*^{k+1}\right]\boldsymbol{u}_{\text{ext}},\, \boldsymbol{v}\right) + \left(\nu D\left(\boldsymbol{u}_*^{k+1}\right),\, \nabla\boldsymbol{v}\right) \tag{3.8}$$

and the right hand side defines our linear form:

$$\begin{aligned} b_1(\boldsymbol{v}) = \left(\frac{\boldsymbol{u}^k}{\delta t} - \frac{1}{2}\left[\nabla\boldsymbol{u}^k\right]\boldsymbol{u}_{\text{ext}},\, \boldsymbol{v}\right) - \left(\nu D\left(\boldsymbol{u}^k\right),\, \nabla\boldsymbol{v}\right) + \\ + \left(p^k,\, \operatorname{div}\boldsymbol{v}\right) - \left(p^k\boldsymbol{n},\, \boldsymbol{v}\right)_{\Gamma_{\text{out}}}. \end{aligned} \tag{3.9}$$

To finish our weak formulation, we now choose the proper function spaces. Let

$$H^1(\Omega) = \left\{v \in L^2(\Omega), v \text{ admits weak gradient from } L^2(\Omega)\right\}, \tag{3.10}$$

$$\mathbf{V}' = \boldsymbol{H}_0^1(\Omega) = \left\{\boldsymbol{v} \in \left[H^1(\Omega)\right]^3, \boldsymbol{v}|_{\Gamma_D} = \boldsymbol{0}\right\}. \tag{3.11}$$

$$\mathbf{V} = \boldsymbol{H}_D^1(\Omega) = \left\{\boldsymbol{v} \in \left[H^1(\Omega)\right]^3, \boldsymbol{v}|_{\Gamma_D} = \boldsymbol{u}_D\right\}. \tag{3.12}$$

Here $\mathbf{V}'$ denotes Sobolev space of vector functions satisfying homogeneous Dirichlet boundary conditions. Now Lax-Milgram lemma gives existence and uniqueness of solution to our weak formulation:

Definition 1 (Weak formulation of the first step (S1) of the incremental pressure projection method.)**.** *Let* $\boldsymbol{u}^k$, $\boldsymbol{u}_{ext} \in \mathbf{V}$, $p^k \in H^1(\Omega)$. *Find* $\boldsymbol{u}_*^{k+1} \in \mathbf{V}'$ *such that*

$$a_1(\boldsymbol{u}_*^{k+1},\, \boldsymbol{v}) = b_1(\boldsymbol{v}) \quad \forall\boldsymbol{v} \in \mathbf{V}'. \tag{3.13}$$

3.4.2 Pressure boundary conditions for (S2)

Multiplying (2.8) by a test function q and using integration by parts, we get

$$-\left(\nabla\left(p^{k+1}-p^{k}\right),\,\nabla q\right)+\left(\nabla\left(p^{k+1}-p^{k}\right)\cdot\boldsymbol{n},\,q\right)_{\partial\Omega}=\frac{1}{\delta t}\left(\operatorname{div}\boldsymbol{u}_{*}^{k+1},\,q\right). \quad (3.14)$$

As we mentioned earlier, weak formulation of the Poisson problem for the pressure requires implicit use of a boundary condition[1]. Apart from loss of incompressibility, there is one more difficulty connected to the choice of homogeneous Neumann condition $\frac{\partial p}{\partial \boldsymbol{n}}=0$. Solution of this problem is unique up to an additive constant, so in the case of our FEM discretization, resulting problem matrix A_2 is singular (notation A_2 is used to be consistent with the rest of the thesis).

There are different possible approaches to treat this difficulty. We will now comment three of them in relation to the unphysical boundary condition.

Approach 1: Additional Dirichlet boundary condition on pressure

Theoretically, to remove ambiguity in the solution it would suffice to set pressure at some arbitrary point. But solving of such problem is not numerically stable, as the approximate solution tends to oscillate near the chosen point.

In the case of our test problem in cylindrical geometry, we can use the symmetry of the solution. As analytic pressure is a function of longitudinal coordinate (and time), it is always constant on any cross-section. Therefore we can chose to set $p=0$ on Γ_{out}. This approach is more stable and accurate for the test problem, because by using this condition we enforce the right symmetry on the solution. Replacement of Neumann boundary condition by Dirichlet type also releases any direct constraint on normal pressure gradient on this part of boundary. This approach effectively removes problem of unphysical boundary condition on outflow.

The question is, whether it is possible to use same approach in the case of real aneurysm geometry. An analogy of cylindrical cross-section for a real geometry is a section normal to an outflow centerline. By prescribing $p=0$ on the outflow, we enforce symmetry that need not to be present in the solution. But it seems that the difference is small, because use of this approach leads to a stable solution. And if the outflow vessel is sufficiently long, error near the boundary will not interfere with the solution inside the aneurysm.

A second problem appears if our geometry has multiple outflows. What constants should we prescribe on the outflows? A relative difference between these constants determines how much blood flows through each of outflow vessels. This may affect overall flow pattern in the aneurysm itself. Unfortunately, we have no reliable data available to answer this question. We also cannot compute it without modelling the flow in much larger part of vascular system. We have to use some heuristic here. We can choose both constants the same or we can compute them to maintain the flow rate appropriate to the sizes of the outflow vessels. As long as this problem is open, we always have to asses the actual effect of different choices on particular problem.

[1]Otherwise the boundary term would remain undefined for pressure functions from space $H^1(\Omega)$.

It is important to note that the uncertainty in choice of proper outflow boundary conditions is property of our problem setting, not a problem of the projection method itself. If we compute coupled Navier Stokes equations with no-stress ($\mathbb{T}\boldsymbol{n} = \mathbf{0}$) condition on outflow, then our velocity-pressure pair is uniquely determined, but we still do not know if the outflow rates of computed flow match reality or not[2].

Approach 2: Use known null space in PETSc

Another approach to solve singular Neumann problem is to use the PETSc Krylov solver ability to compute minimal norm solution to singular problems (PETSc).

To do this, we need to be able to express algebraic vector generating the null space of A_2. For this approach to work, we also need to orthogonalize the right hand side to the null space to ensure that it lies in range of A_2.

Problem with this approach is that it does not help with the unphysical boundary condition. If we could enforce the right outflow by prescribing pressure gradient correctly, than we could use it. In test problem with cylinder, we know that normal pressure gradient is constant on each cross-section. In each time step, we can compute right outflow pressure gradient to ensure that inflow and outflow rates for corrected velocity are equal. This approach gives good results on our test problem.

This approach is not practical for the real problem. As the enforced gradient determines quite directly the resulting corrected velocity profile on the outflow, we would either need to know the resulting shape of the outflow profile in advance to set the gradient consistent with the rest of the velocity field, or we would need to devise some more sophisticated pressure boundary condition. Even then we could have more problems with solvability of our Neumann problem[3]. Using constant gradient on whole cross section leads to numerical problems because of inconsistency between enforced constant velocity profile, no-slip boundary conditions on walls and velocity profile in outflow vessel.

Approach 3: Lagrange multipliers

Third possibility is to add condition on average pressure $\int_\Omega p \, \mathrm{d}x = 0$ using Lagrange multipliers. This again does not help solving the problem with unphysical boundary condition. Using constraint also makes solving the algebraic problem significantly harder, as resulting matrix is indefinite (saddle point) and is no longer diagonally dominant (see matrix properties at the end of section 3.5).

Chosen approach

We tested all three approaches, but only the first two are implemented in the final version of code. We use the first approach, as it is the only approach that provides us with a good solution for the real problem. We have chosen both Dirichlet conditions to be zero for our tests on real mesh.

[2]How to prescribe pressure difference on the outflows for the coupled Navier-Stokes equations is described in the work of Heywood et al. (1996)

[3]In the work of Gresho and Sani (1987) there seem to be a defence of use of such a complex pressure boundary condition, but we do not see it could be somehow applicable to our projection scheme

3.4.3 Weak formulation of the Poisson problem (S2)

Returning to equation (3.14), the boundary terms vanish for all considered cases: either $\frac{\partial p}{\partial \boldsymbol{n}}$ or q is zero on each part of boundary. We can now define bilinear and linear forms

$$a_2(p^{k+1}, q) = \left(\nabla p^{k+1}, \nabla q\right) \tag{3.15a}$$

$$b_2(q) = \left(\nabla p^{k}, \nabla q\right) - \frac{1}{\delta t}\left(\operatorname{div} \boldsymbol{u}_*^{k+1}, q\right). \tag{3.15b}$$

and choose the space

$$Q = \begin{cases} \{q \in H^1(\Omega),\, q|_{\Gamma_{\text{out}}} = 0\} & \text{for the first approach,} \\ \left\{q \in H^1(\Omega),\, \int_\Omega q \mathrm{d}x = 0\right\} & \text{for the second approach.} \end{cases} \tag{3.16}$$

for the solution and test functions of the Poisson problem to get our weak formulation:

Definition 2 (Weak formulation of the second step (S2) of the incremental pressure projection method.)**.** *Let* $\boldsymbol{u}_*^{k+1} \in \mathbf{V}$*,* $p^k \in Q$*. Find* $p^{k+1} \in Q$ *such that*

$$a_2(p^{k+1}, q) = b_2(q) \quad \forall q \in Q. \tag{3.17}$$

3.4.4 Weak formulation of third and fourth step

Weak formulations of the remaining steps are straightforward, as no integration by parts and no discussion of boundary conditions are needed. Resulting bilinear forms are:

$$a_3(\boldsymbol{u}^{k+1}, \boldsymbol{v}) = \left(\boldsymbol{u}^{k+1}, \boldsymbol{v}\right) \tag{3.18a}$$

$$a_4(p^{k+1}, q) = \left(p^{k+1}, q\right). \tag{3.18b}$$

Both steps can be interpreted as an L^2-orthogonal projections of corresponding right hand side to the velocity space $\left[H^1(\Omega)\right]^3$ or the pressure space $H^1(\Omega)$.

3.5 Space discretization by FEM

The weak formulation is defined with infinite-dimensional function spaces. To get a practically computable problem, we must perform a space discretization by choosing finite-dimensional subspaces $\mathbf{V}_h \subset \mathbf{V}'$ and $Q_h \subset Q$.

We are using Finite element method (FEM) as a way of choosing these subspaces. Our computational domain Ω is partitioned by tetrahedral mesh. Widely used choice for this type of problem are Taylor-Hood elements, which satisfy the inf-sup stability condition[4]. Velocity space $\mathbf{V}_h$ are continuous vector functions that are polynomials of degree two on each tetrahedron. Pressure space

[4]Analysis and numerical tests of different projection methods in relation to need of the choice of inf-sup condition stable elements can be found in the work of Liu et al. (2010). Our projection scheme falls into the category of "pressure update (PU)" methods in the article and was found unstable when using linear polynomial finite element spaces for velocity.

Q_h are continuous scalar functions that are polynomials of degree one on each tetrahedron.

Using typical nodal bases $\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_N\}$ of $\mathbf{V}_h$ and $\{\psi_1, \ldots, \psi_M\}$ of Q_h in our bilinear forms, we get algebraic problems

$$A_s\mathbf{x} = \mathbf{b}_s,\ (A_s)_{ij} = a_s(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j), (\mathbf{b}_s)_j = b_s(\boldsymbol{\varphi}_j),\ \text{for } s = 1 \text{ and } 3 \tag{3.19}$$

$$A_s\mathbf{y} = \mathbf{b}_s,\ (A_s)_{ij} = a_s(\psi_i, \psi_j), (\mathbf{b}_s)_j = b_s(\psi_j),\ \text{for } s = 2 \text{ and } 4 \tag{3.20}$$

where vectors $\mathbf{x}$ and $\mathbf{y}$ determines our finite element velocity and pressure approximations $\boldsymbol{u}_h$ and p_h by relations

$$\boldsymbol{u}_h = \sum_{i=1}^{N} \mathbf{x}_i\boldsymbol{\varphi}_i, \quad p_h = \sum_{i=1}^{M} \mathbf{y}_i\psi_i. \tag{3.21}$$

From the shape of the bilinear forms and the choice of finite element spaces we can see some properties of matrices A_s.

The matrices A_3 and A_4 are Gram matrices of corresponding finite element basis, usually called “mass matrices” in the context of FEM. Both are diagonally dominant. The matrix A_4 has non-negative entries due to use of finite elements of first polynomial degree. Matrix A_2 is symmetric positive definite and diagonally dominant.

The matrix A_1 has a structure $A_1 = A_3 + \delta t\,(A_{\text{convection}} + A_{\text{diffusion}})$ (up to the Dirichlet boundary conditions). The non-symmetry of matrix $A_{\text{convection}}$ imply non-symmetry of A_1.

3.6 Notes on boundary conditions

Now we return to other implementation issues concerning boundary condition that were not resolved in previous sections.

3.6.1 Splitting error and Dirichlet BC for (S3)

As we described in section 2.2, one effect of the splitting error of projection method is that corrected velocity does not satisfy no-slip condition on walls in tangential direction. What happens, if we enforce our Dirichlet boundary conditions on the corrected velocity? Boundary conditions will be satisfied exactly, but because they are inconsistent with the correction, the effect will probably be just a shift of error distribution, not necessarily a reduction of the error. But in practice this splitting error is negligible in comparison to error in the inside of the domain, and the same goes for the effect of this splitting error on the computed wall shear stress.

3.6.2 Inflow velocity profile for real problem

If we want to reproduce accurately true blood flow patterns in our simulation, we should choose inflow boundary conditions that are also patient specific. However, current techniques of non-invasive measuring of blood flow are not advanced enough to provide enough information to determine patient's inflow velocity field. We must make assumptions about the flow, which creates additional error in the computed flow. Quoting from the conclusion of Venugopal et al. (2007):

"Our study indicates that measurement of a patient's blood flow rate as well as the distribution of flow rates in the feeding arteries may be needed for numerical simulations to accurately reproduce the intraaneurysmal hemodynamic factors in a patient-specific aneurysm."

Our main goal in this work is to implement new numerical method, so we chose a hand picked polynomial approximation of possible flow rate distribution during the blood flow cycle (see figure 3.2), and we use parabolic velocity profiles on circles inscribed to the inflow cross-sections, with zero flow outside the circle.

3.6.3 Dirichlet boundary condition and initial condition compatibility

We start our computation from zero velocity and pressure initial conditions, but our inflow velocity profile is always non-zero. This incompatibility can cause serious problems during computation of first cycle, typically serious oscillations or divergence of iterative solver. Therefore we need to avoid this incompatibility. This can be easily done by multiplying our inflow velocity by suitable smoothing function. We used:

$$\boldsymbol{u}_D(\text{smoothed}) = \boldsymbol{u}_D \frac{1-\cos(2\pi t)}{2} \text{ for } t \in [0, 0.5] \tag{3.22}$$

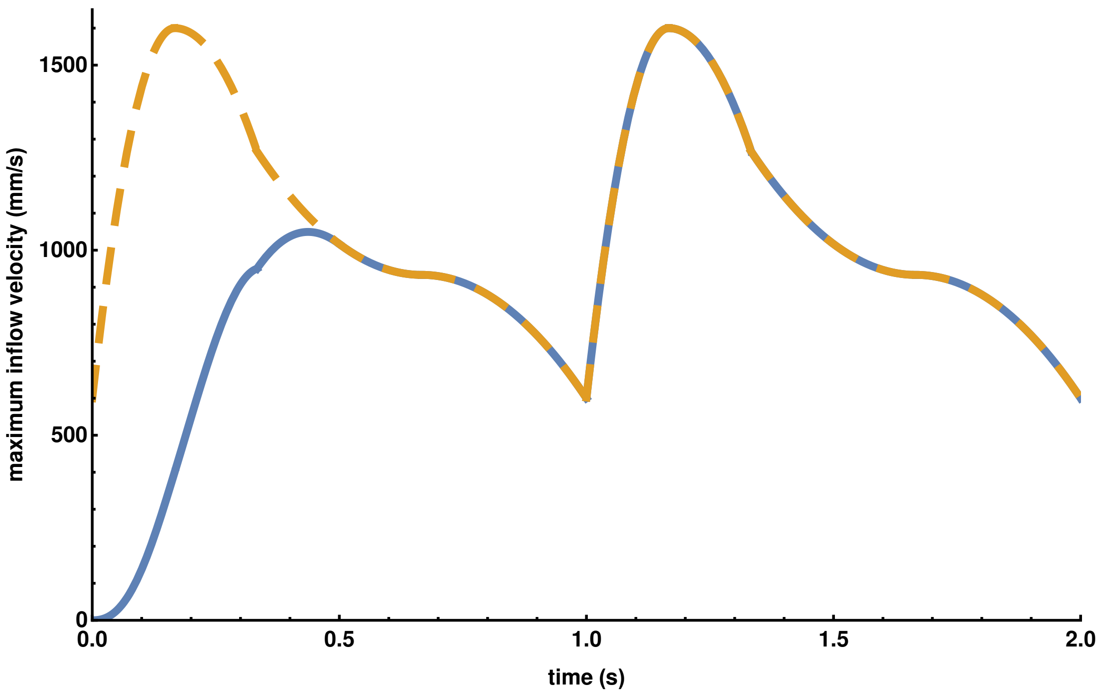

Figure 3.2: Shows chosen course of maximal inflow velocity with (solid line) and without (dashed line) used smoothing.

3.7 Stabilization

Stability of computation of tentative velocity is very sensitive to the magnitude of the velocity and chosen time step. To get results on tested real meshes without any oscillations, a 0.0001 s time step is needed. This is very demanding on computational resources, even with a projection method. Additional stabilization is required to use the projection method with larger time steps.

As the equations for tentative velocity are essentially a vector variant of convection-diffusion problem and the instability is caused by dominating convection term, we chose standard streamline-upwind Petrov-Galerkin (SUPG) technique, described for example in John and Schmeyer (2008). There are two main ways to implement this technique.

First one is "consistent" in the sense that a solution of the original weak formulation is a solution of the stabilized one. This technique modifies the test functions in following way:

$$\boldsymbol{v} \to \boldsymbol{v} + \tau \left[\nabla \boldsymbol{v}\right] \boldsymbol{u}_{\text{ext}} \tag{3.23}$$

where τ is chosen stabilization parameter. But our equations stabilized in this way are much more difficult to solve. We resorted to the second type: We only modify the test function in the convection term of our bilinear form. That means adding term

$$\frac{\tau}{2} \left(\left[\nabla \boldsymbol{u}_*^{k+1}\right] \boldsymbol{u}_{\text{ext}},\, \left[\nabla \boldsymbol{v}\right] \boldsymbol{u}_{\text{ext}} \right) \tag{3.24}$$

to our bilinear form. This leads to stabilization of the problem at the cost of changing the equations and thus introducing a new error.

To use SUPG successfully, the choice of parameter δ is crucial. According to proposal from Codina (2000) we used

$$\tau = \tau_M \frac{h_K^2 \delta t}{2\nu \delta t + h_K \delta t ||\boldsymbol{u}_{\text{ext}}||_K + h_K^2}, \tag{3.25}$$

where h_K is mesh cell size, $||\cdot||_K$ local L^2 norm.

Multiplicative parameter τ_M must be adjusted empirically for our problem. In our case, $\tau_M \in [1, 2]$ is the best choice for time step 1 ms for the real problem. For time steps 5 ms and above, the oscillations cannot be removed by this kind of stabilization.

3.8 Notes on the implementation

In this section we comment on some practical issues concerning implementation of projection methods.

3.8.1 Partial assembly of matrices

When we are using iterative solvers and small time steps, the efficiency of assembling problem matrices can have strong impact on use of the computational resources. As bilinear forms for steps (S2), (S3) and (S4) are independent of current time step, we can assemble the associated matrices only once.

If we recall the bilinear form for the first step (3.8) including stabilization term (3.24)

$$\begin{aligned} a_1(\boldsymbol{u},\, \boldsymbol{v}) =& \frac{1}{\delta t}\left(\boldsymbol{u}_*^{k+1},\, \boldsymbol{v}\right) + \frac{1}{2}\left(\left[\nabla \boldsymbol{u}_*^{k+1}\right] \boldsymbol{u}_{\text{ext}},\, \boldsymbol{v}\right) + \frac{1}{2}\left(2\nu D\left(\boldsymbol{u}_*^{k+1}\right),\, \nabla \boldsymbol{v}\right) + \\ &+ \frac{\tau}{2}\left(\left[\nabla \boldsymbol{u}_*^{k+1}\right] \boldsymbol{u}_{\text{ext}},\, \left[\nabla \boldsymbol{v}\right] \boldsymbol{u}_{\text{ext}}\right), \end{aligned} \tag{3.26}$$

we see that only convective and stabilization terms with extrapolated velocity $\boldsymbol{u}_{\text{ext}}$ change with time (note that δt is constant). We can save resources by assembling all other terms into matrix A_{const} only once and in each time steps assemble the changing terms and add the resulting matrix to A_{const}.

The FEniCS also provides us with additional optimization techniques for assembly code generation. This techniques simplify symbolic expression and reuse repeatedly used quantities to reduce number of redundantly repeated computations during matrix assembly. Technical details of these techniques are described in (Logg et al., 2012, Chapter 7).

3.8.2 Choice of solvers and preconditioners

We also need to choose solvers and preconditioners for each of the four projection method problems. We selected from options provided by library PETSc. We tried to choose solvers and preconditioners that scale well when running in parallel.

For the tentative velocity step (S1), which has nonsymmetric matrix, we used GMRES method. Algebraic multigrid preconditioner from the LNLL package hypre and the "SOR" preconditioners worked well for our problem. The PETSc "SOR" preconditioner with default settings is Jacobi block preconditioner with Gauss-Seidel on each block. We chose to use the "SOR" because it scales much better for higher numbers of processors.

For step (S2), (S3), (S4) with symmetric matrices we chose CG with "SOR" preconditioner again because of scaling.

To reduce use of computational resources we use last corrected velocity as the starting vector for both corrected (S3) and tentative (S1) velocity steps.

3.8.3 Computation of WSS

We found two solutions of the technical problem of extracting WSS from computed velocity (pressure is not needed, as it contributes only to the normal component of the stress). One uses FEniCS class BoundaryMesh to extract surface mesh, where the computation of WSS makes sense. We can get WSS as a vector field or its norm this way. Problem with this approach that extraction of BoundaryMesh does not work when higher numbers of processors are used in the computation (exact number depends on mesh). This problem can be circumvented by saving stress tensor for the whole domain and post-processing it later with lower number of processors. Other option is to compute WSS using an integral over all mesh facets. This works for any number of processors, but we are only able to get the norm of WSS using this technique.

4. Summary of results

With the implemented IPCS code, we are able to solve our problem on finer meshes than before. To test it, we computed our three meshes of the same geometry and different quality. Parameters of meshes are given in Table 4.1. The coarsest mesh is small enough to be computed with direct solver approach for comparison (the memory requirement for direct solver is about 40 GB RAM).

Mesh	cells	vertices	average edge length	DOF for velocity
1	$1.95 \cdot 10^5$	34,791	0.32	$8.18 \cdot 10^5$
2	$5.83 \cdot 10^5$	$1.05 \cdot 10^5$	0.23	$2.46 \cdot 10^6$
3	$2.15 \cdot 10^6$	$3.56 \cdot 10^5$	0.15	$8.71 \cdot 10^6$

Table 4.1: Properties of real meshes used in our tests. DOF stands for degrees of freedom.

Using the rotation scheme with stabilisation has negligible effect on the resulting solution in comparison with stabilized IPCS scheme. Without stabilisation, rotation scheme was less stable than IPCS scheme. In our setting, rotational modification does not bring any new quality and so there is no reason to use it. Therefore all result for the projection methods shown in this thesis are for the IPCS scheme.

4.1 Reliability of computed WSS

To assess an impact of stabilized and non-stabilized pressure correction method on computed solution we have run them using the coarsest grid, so we could compare the results with the direct approach. Because there were only negligible differences between results in the second blood flow cycle and any of computed following cycles, we used values of WSS and velocity from the time of peak inflow in the second cycle ($t = 1{}^1\!/\!{}_6\,\mathrm{s}$). Computed WSS norm averaged over cell facets is shown in Figure 4.1. Computed peak velocities are shown in Figure 4.2. The velocity fields for direct method and non-stabilized IPCS are almost the same, but the stabilised flow is different. The distribution and size o WSS differs for the three computational approaches. Assuming that the results from the direct approach are the most reliable, we can say that using non-stabilized IPCS method provides better solution. However, differences between stabilized IPCS and direct approach may be considered small enough for practical purposes. The decision what method to use depends on available resources and proper assessing of significance of the other error sources present in modelling, such is the inaccuracy of obtained geometry or used boundary conditions.

To assess an impact of using meshes of different quality, we have run stabilized IPCS method for all three meshes. All meshes are acquired from the same geometry, but using different sampling (meshes are not constructed by refining). Computed peak WSS norms are shown in Figure 4.3. Apart from the better resolution of computed quantities, small qualitative differences are visible between different meshes.

4.2 Scalability

To test performance of the code when running in parallel, simple scaling tests were conducted in the national supercomputing center IT4I in Ostrava (www.it4i.cz).

Two types of parallel scalability are usually distinguished. Strong scaling measures efficiency of code when increasing the number of computational units while computing a single problem. The ideal performance would be that for n-times more computational units the computation time is reduced n-times. Weak scalability tests the performance of the code when we increase problem size and computational resources accordingly. Here the ideal performance would be that the computational time remains constant for any size of problem. We cannot expect such behaviour in our setting because the more computation units we use the more time is spent on communication between the processes.

We have ran two seconds with 1 ms time step for all three meshes on different numbers of processors (used cluster provides us with 24 core nodes). We measured time spent on each of the individual solver steps and on the assembly of the non-constant part of the matrix A_1.

In all the test problems following stopping criteria were used for preconditioned residuum:

Step	Absolute tolerance	Relative tolerance
(S1)	10^{-4}	10^{-12}
(S2)	10^{-6}	10^{-10}
(S3)	10^{-4}	10^{-12}

Table 4.2: Tolerances used in stopping criteria for IPCS solvers

Stopping criterion is activated if at least one tolerance is satisfied.

Results for strong scalability for each of the meshes can be found in Figures 4.4, 4.5 and 4.6. Results for weak scalability (for more or less constant DOF per processor ratio) can be found in Figure 4.7.

From the results we see that the IPCS method scales well with the exception of Poisson problem solver on step (S2). This problem is smaller if we use finer meshes, but until better solver and preconditioner setup is implemented, this poses a limit for further scalability.

4.3 Comparison of the computational cost

To illustrate the difference in efficiency of direct and projection methods, we have computed the flow to the first 1.3 s of flow on the coarsest grid. We used eight processors, because the direct method does not scale for more than eight processors. Comparison with the stabilized IPCS scheme is in table 4.3. Even without using the advantage of scalability, the IPCS method is about 12 times faster.

Approach	Direct solver	IPCS with SUPG
Total time (h)	65.94	5.34
Avg. time/step (min)	30.43	0.25

Table 4.3: Comparison of computational cost

To compare the computational costs of the IPCS with and without the stabilisation, we have run two cycles (seconds) on 96 processors. The results are shown in table 4.4.

Approach	IPCS with SUPG	IPCS no SUPG
Total time (min)	36.09	248.28
Running solvers (min)	18.44	154.2
A1 assembly (min)	10.57	45
A1 assembly/step (s)	0.32	0.14
Avg time/step (s)	1.08	0.74

Table 4.4: Comparison of computational cost

Without the stabilisation, we have to compute ten times more steps, but the computation takes only about seven times more time. This has two reasons: without stabilization, we save on the time needed to assemble the stabilization matrix. Also, smaller time steps means our initial guess for velocity solvers is closer to the solution, resulting in faster convergence.

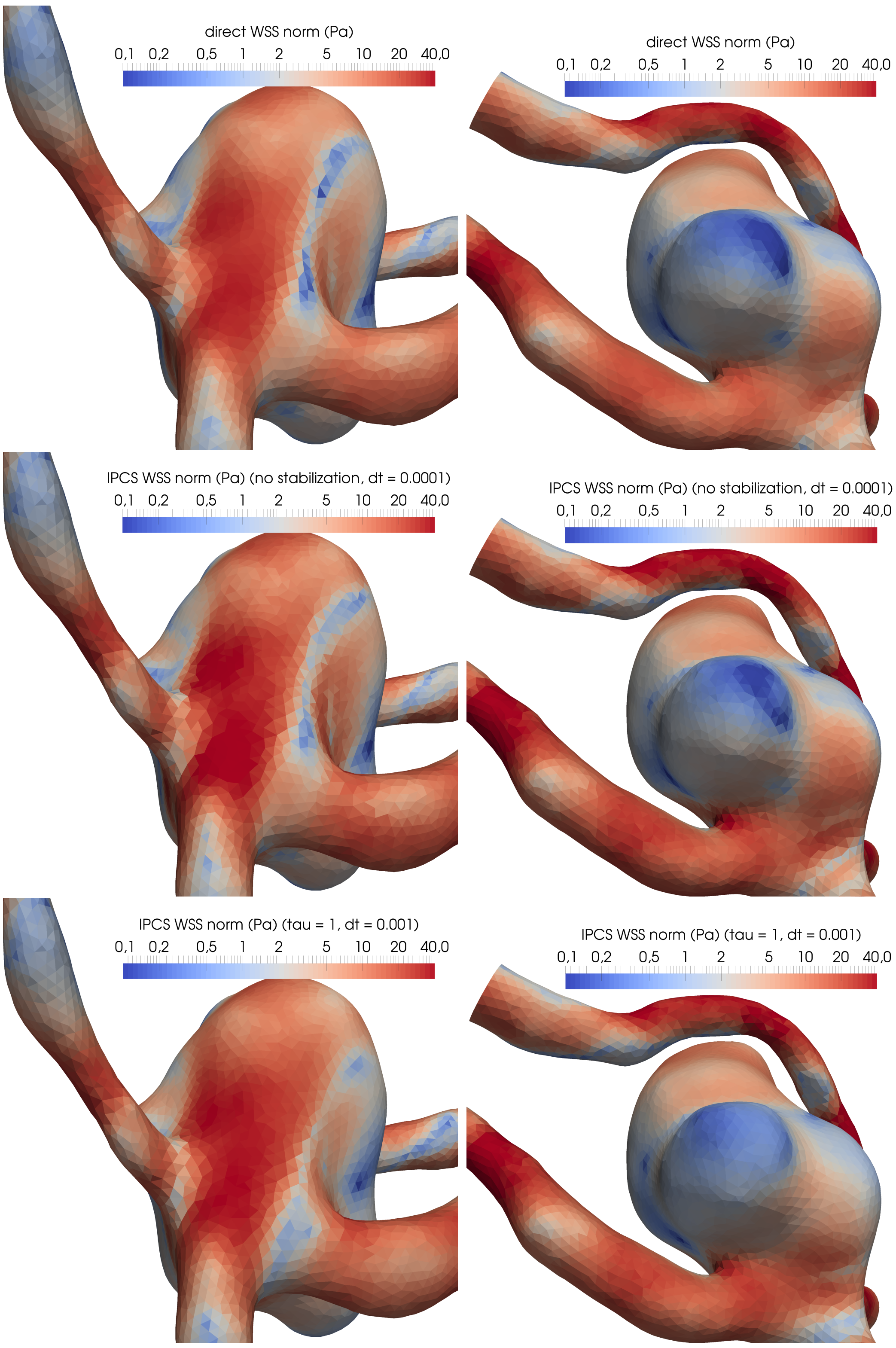

Figure 4.1: WSS comparison for the coarsest mesh.

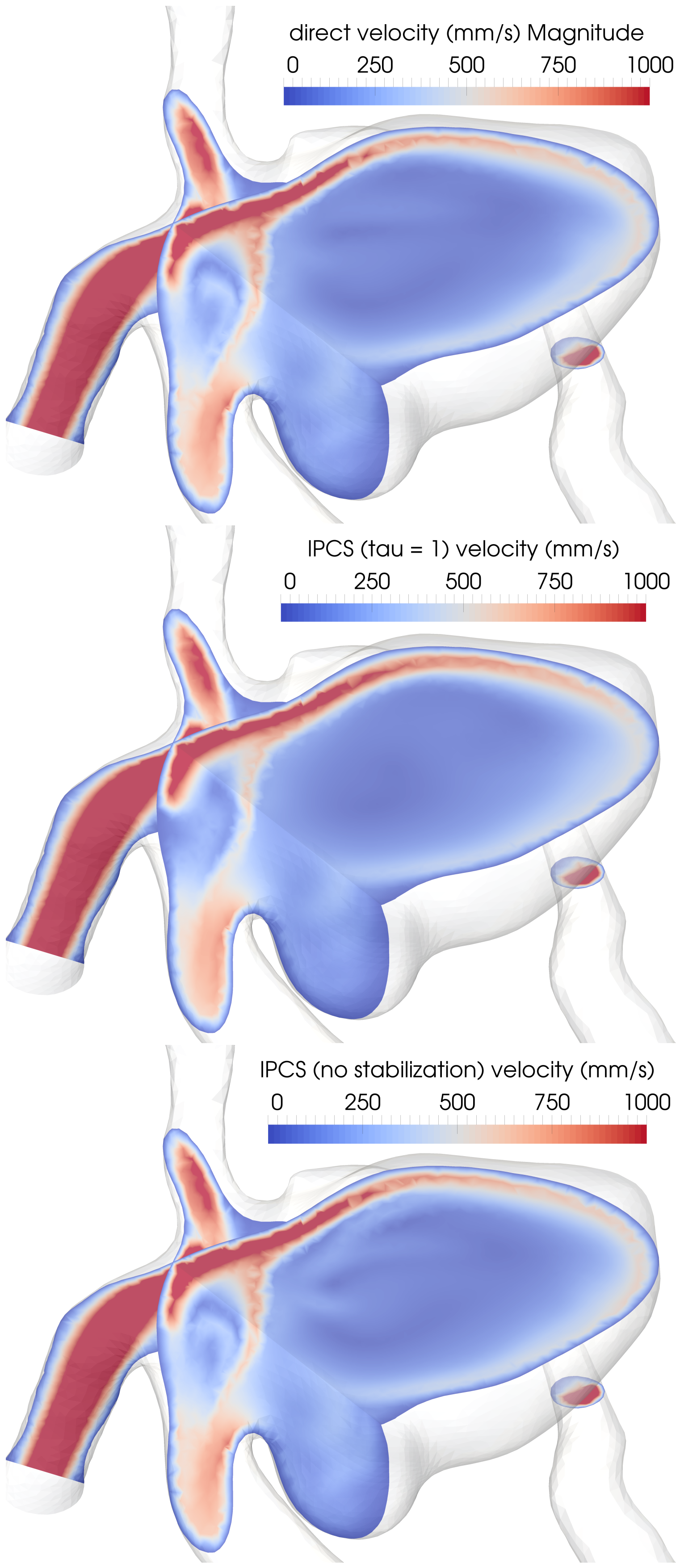
direct velocity (mm/s) Magnitude
0 250 500 750 1000
IPCS (tau = 1) velocity (mm/s)
0 250 500 750 1000
IPCS (no stabilization) velocity (mm/s)
0 250 500 750 1000

Figure 4.2: velocity comparison for the coarsest mesh.

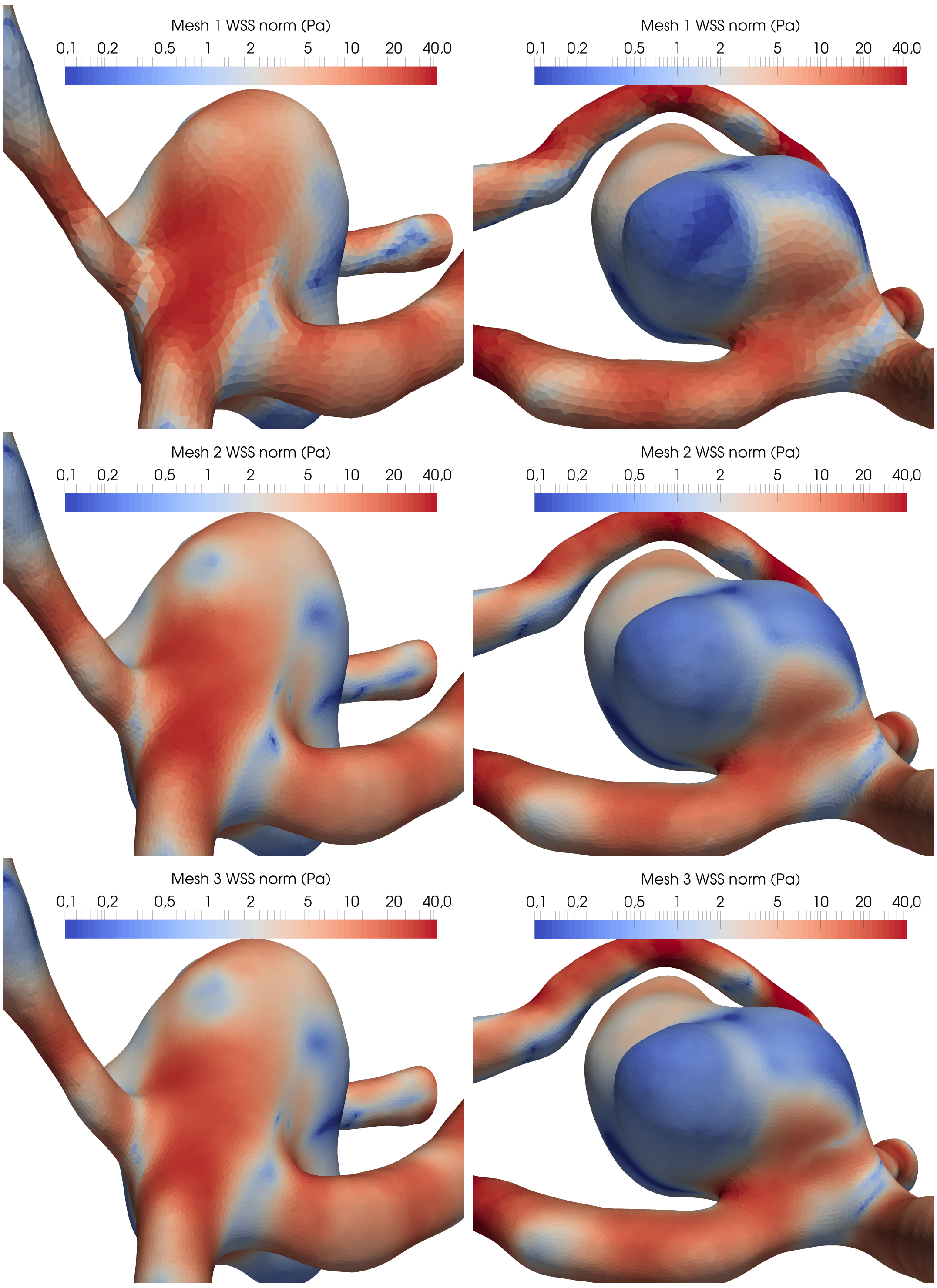

Figure 4.3: WSS comparison for the three different meshes computed by stabilized IPCS.

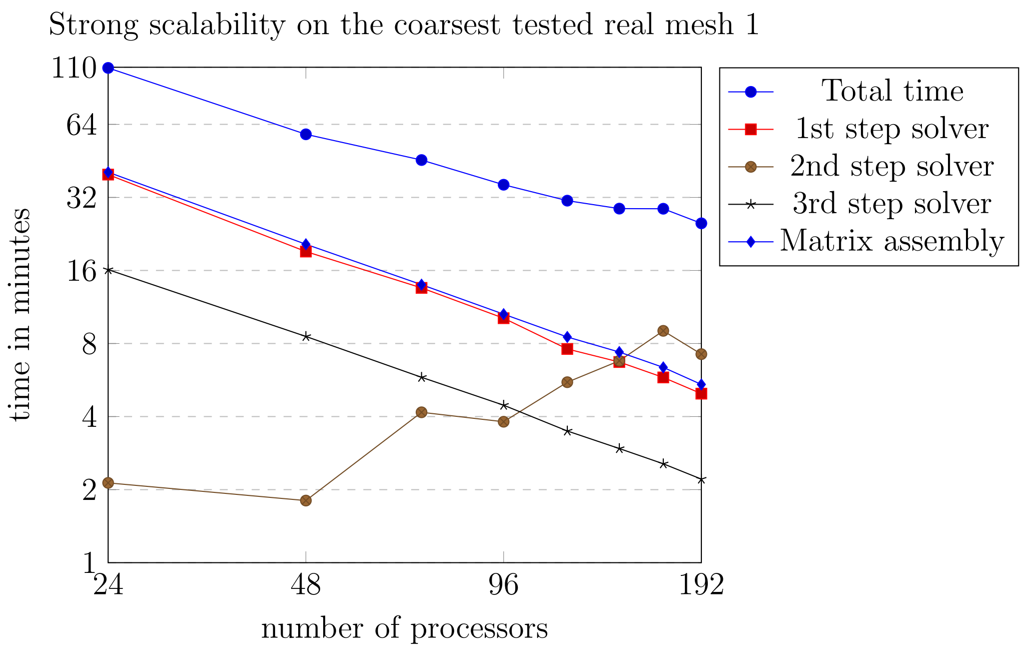

Figure 4.4: Strong scaling

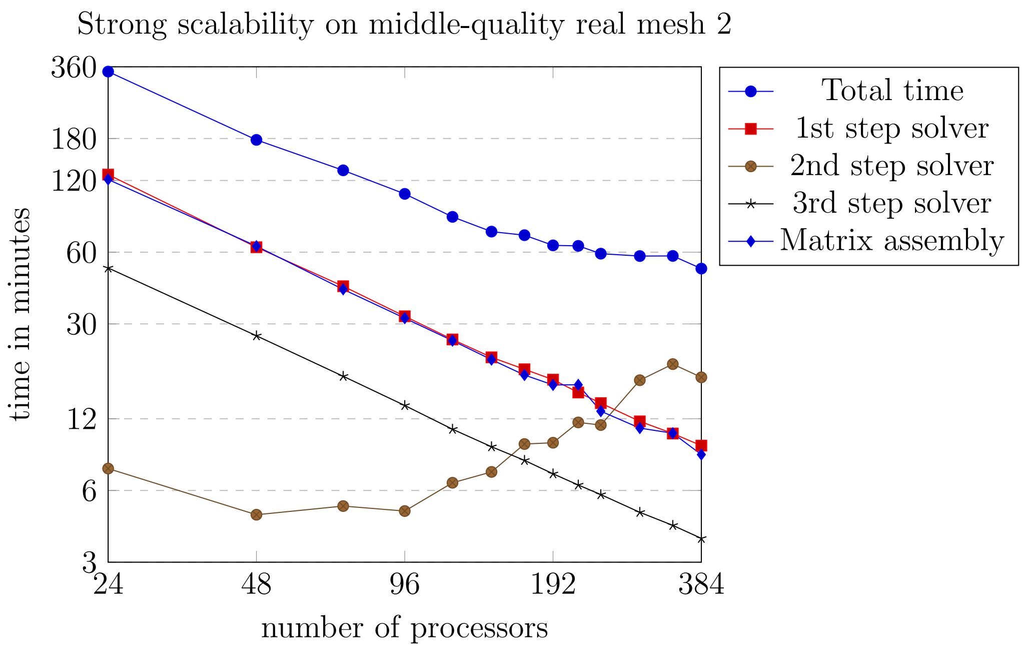

Figure 4.5: Strong scaling

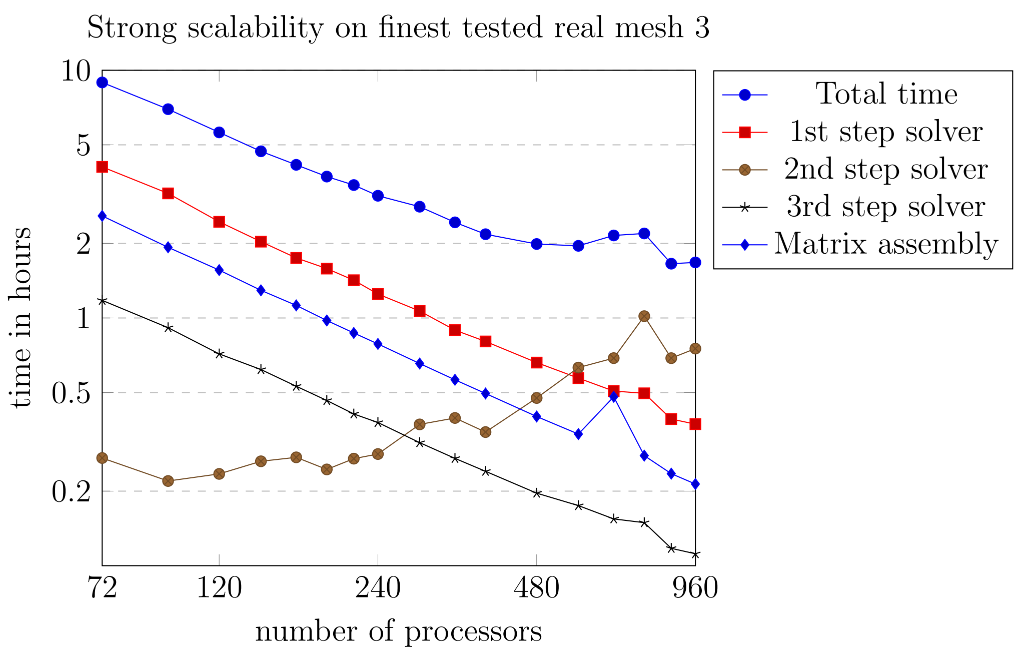

Figure 4.6: Strong scaling

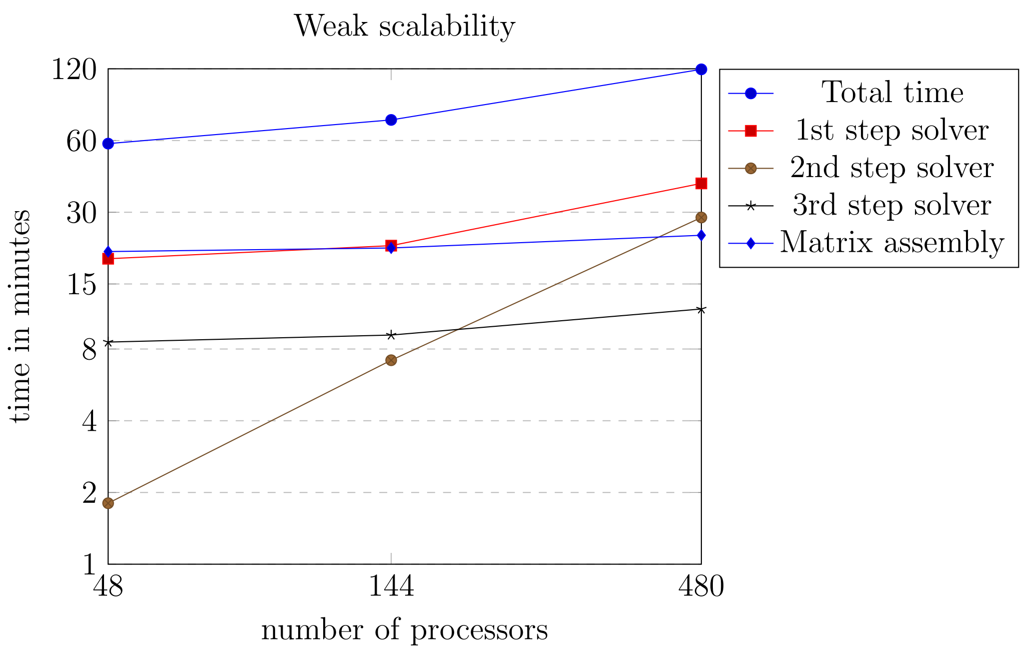

Figure 4.7: Weak scaling

Conclusion

During the work on this thesis, we successfully developed a working implementation of IPCS for the needs of our application. The code can now be used for generating blood flow and relevant hemodynamic indicators and can serve as a tool for further research.

For direct practical purposes, the results can be used as a clue for the expert neurosurgeon's guess at the risk factors of particular aneurysm. This is mainly because the question how to assess the risk of a rupture is still an open question and no definite working procedure is known. For this purpose, several technical issues are worth refining. The most important is implementing a better way to prescribe suitable inflow boundary conditions. The next would be implementing a possibility to compute with various rates of flow on individual inflows and outflows to assess the sensitivity of the results to such changes in boundary conditions.

As for further research, possible next steps for improving the physical model would be adapting the code for use of a non-Newtonian model of blood.

Ability of the code to scale in parallel computing is important because it allows us solving bigger problems when needed. Given enough computational resources, code can be used for conducting various sensitivity studies with bigger confidence in results due to higher achieved resolution of computation.

To further improve the efficiency of the code, possible improvements could be achieved by better and more involved adjustment of used solvers and preconditioners, especially the solver for the Poisson problem, which now does not scale well. This could be remedied by proper tuning of a multilevel type preconditioner. Another option would be implementing of an adaptive time stepping, as the extremely short time step is needed only in a part of the blood flow cycle.

Bibliography

BAF. Brain Aneurysm Foundation: Brain Aneurysm Statistics and Facts. [online]. URL `http://www.bafound.org/about-brain-aneurysms/brain-aneurysm-basics/brain-aneurysm-statistics-and-facts/`.

Anil Can and Rose Du. Association of Hemodynamic Factors With Intracranial Aneurysm Formation and Rupture: Systematic Review and Meta-analysis. *Neurosurgery*, 2016.

Alexandre Joel Chorin. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Amer. Math. Soc.*, 73(6):928–931, 11 1967. URL `http://projecteuclid.org/euclid.bams/1183529112`.

R. Codina. On stabilized finite element methods for linear systems of convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 188:61–82, July 2000. doi: 10.1016/S0045-7825(00)00177-8.

FEniCS. [online]. URL `https://fenicsproject.org/`.

Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations : theory and algorithms*. Springer series in computational mathematics. Springer-Verlag, Berlin, New York, 1986. ISBN 0-387-15796-4. Extended version of : Finite element approximation of the Navier-Stokes equations.

Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1111–1145, 1987. ISSN 1097-0363. doi: 10.1002/fld.1650071008. URL `http://dx.doi.org/10.1002/fld.1650071008`.

J. L. Guermond and Jie Shen. On the Error Estimates for the Rotational Pressure-Correction Projection Methods. *Mathematics of Computation*, 73(248):1719–1737, 2004. ISSN 00255718, 10886842.

J. L. Guermond and Jie Shen. An overview of projection methods for incompressible flows. *Comput. Methods Appl. Mech. Engrg*, 41:112–134, 2006.

J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokse equations. *International Journal for Numerical Methods in Fluids*, 22(5): 325–352, 1996. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19960315)22:5⟨325::AID-FLD307⟩3.0.CO;2-Y. URL `http://dx.doi.org/10.1002/(SICI)1097-0363(19960315)22:5<325::AID-FLD307>3.0.CO;2-Y`.

Volker John and Ellen Schmeyer. Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion. *Computer Methods in Applied Mechanics and Engineering*, 198(3–4):475 – 494, 2008. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2008.08.016. URL `http://www.sciencedirect.com/science/article/pii/S0045782508003150`.

Jian-Guo Liu, Jie Liu, and Robert L. Pego. Stable and Accurate Pressure Approximation for Unsteady Incompressible Viscous Flow. *J. Comput. Phys.*, 229(9):3428–3453, May 2010. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.01.010. URL `http://dx.doi.org/10.1016/j.jcp.2010.01.010`.

Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. ISBN 978-3-642-23098-1. doi: 10.1007/978-3-642-23099-8.

NINDS. National Institute of Neurological Disorders and Stroke: Cerebral Aneurysms Fact Sheet. [online], 2013. URL `http://www.ninds.nih.gov/disorders/cerebral_aneurysm/detail_cerebral_aneurysms.htm`.

PETSc. PETSc Documentation. [online], 2016. URL `http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/Mat/MatSetNullSpace.html`.

Y. Qian, H. Takao, M. Umezu, and Y. Murayama. Risk Analysis of Unruptured Aneurysms Using Computational Fluid Dynamics Technology: Preliminary Results. *American Journal of Neuroradiology*, 32(10):1948–1955, 2011. doi: 10.3174/ajnr.A2655. URL `http://www.ajnr.org/content/32/10/1948.abstract`.

Jie Shen. On Error Estimates of the Projection Methods for the Navier-Stokes Equations: Second-Order Schemes. *Mathematics of Computation*, 65(215): 1039–1065, 1996. ISSN 00255718, 10886842. URL `http://www.jstor.org/stable/2153791`.

Helena Švihlová. Aplikace metody konecnych prvku na realne problemy v hemodynamice. Master's thesis, Mathematical Institute of Charles University, 2013. URL `https://is.cuni.cz/webapps/zzp/detail/114225/`.

R. Témam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I). *Archive for Rational Mechanics and Analysis*, 32(2):135–153, 1969. ISSN 1432-0673. doi: 10.1007/BF00247678. URL `http://dx.doi.org/10.1007/BF00247678`.

L. J. P. Timmermans, P. D. Minev, and F. N. Van De Vosse. An Approximate Projection Scheme For Incompressible Flow Using Spectral Elements. *International Journal for Numerical Methods in Fluids*, 22(7): 673–688, 1996. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19960415)22: 7⟨673::AID-FLD373⟩3.0.CO;2-O. URL `http://dx.doi.org/10.1002/(SICI)1097-0363(19960415)22:7<673::AID-FLD373>3.0.CO;2-O`.

Prem Venugopal, Daniel Valentino, Holger Schmitt, J. Pablo Villablanca, Fernando Viñuela, and Gary Duckwiler. Sensitivity of patient-specific numerical simulation of cerebal aneurysm hemodynamics to inflow boundary conditions. *Journal of Neurosurgery*, 106(6):1051–1060, 2007. doi: 10.3171/jns.2007.106.6.1051. URL `http://dx.doi.org/10.3171/jns.2007.106.6.1051`. PMID: 17564178.

List of Figures

List of Tables

Appendix - the code

The code is accessible through GitHub (https://github.com/j-hr/projection).

You can download it there or clone it using the git - free version control software (recommended).

The code is operated using command line parameters. Basic structure is:

```
python main.py problem_name solver_name mesh total_time time_step
```

problem_name	“womersley_cylinder” for test problem or “real” for real aneurysm problem
solver_name	“ipcs1” for IPCS scheme or its modifications or “direct” for previous solving strategy
mesh	“cyl_c1”, “cyl_c2” or “cyl_c3” for test problem meshes (gradually increasing quality) or “HYK” for real mesh (coarsest)
total_time	computation will run from 0 to this time (e. g. “1”, or “0.25”)
time_step	e. g. “0.1”

All parameters are used without quotes. Many optional parameters can be appended.

To visualise the results you will need software compatible with XDMF standard, such as the Paraview software (the code generates scripts for more convenient visualisation using the Python interface of Paraview 4.4).

For up-to-date information about the code, its usage, software prerequisites etc. see the README file on GitHub.