

EINDHOVEN UNIVERSITY OF TECHNOLOGY (TU/e)

MASTER'S THESIS

Cardiovascular blood flow Comparing measurement and simulation

Author:
N.H.L.C. (Niels) DE HOON

Supervisors:
Roy VAN PELT
Andrei JALBA
Anna VILANOVA

*A thesis submitted in fulfilment of the requirements
for the degree of Master of Science*

Visualization
Mathematics and Computer Science

August 2013

Abstract

Mathematics and Computer Science

Cardiovascular blood flow Comparing measurement and simulation

N.H.L.C. (Niels) DE HOON

In the cardiovascular field, Magnetic Resonance Imaging (MRI) is increasingly used in healthcare, to obtain non-invasive anatomical patient-specific images. In addition to anatomical images, MRI can be used to evaluate the blood flow within the cardiovascular system. Time-resolved 3D phase-contrast (PC) MRI provides a technique to measure blood flow in three directions at multiple time steps during the cardiac cycle. The acquired PC-MRI blood-flow data is expected to be able to diagnose several cardiovascular diseases (CVDs), and is therefore of great relevance to physicians. With typically twenty measurements per cardiac cycle the data is relatively coarse in the temporal domain.

Due to this coarse temporal domain, linear temporal interpolation is currently used to obtain a velocity field between two measurements. In this study, a novel method is proposed, PC-MRI-measurement integrated (PCMI) simulation. PCMI combines PC-MRI measurements with a fluid simulation from the computer-graphics field, namely the fluid implicit particle (FLIP) method. Our measurement-coupling method was compared to existing techniques using synthetic data, and it was shown that our method was more similar to the measured velocity field, within the physical constraints. Furthermore, noise robustness was shown for noise typical for PC-MRI. A significant difference between linear temporal interpolation and the fluid simulation was demonstrated, however, our technique is likely beneficial due to its use of fluid mechanics. Furthermore, a visual analysis with PC-MRI data was done. This showed that typical blood-flow patterns, in a healthy volunteer and a patient suffering from an aortic dissection, was maintained.

To the best of our knowledge, we are the first to combine full field velocity measurements with fluid simulation instead of defining only the in- and outflow conditions for the simulation. Therefore, we think it is plausible, that our method is more in correspondence with patient's blood flow than conventional measurements coupling techniques, as PCMI is physically underpinned.

Acknowledgements

I could not have finished this master thesis without the help of many people, either directly or indirectly. Firstly, I would like to thank my project supervisor Andrei Jalba for the useful comments, remarks and help with the technical details. Furthermore, I would like to thank my project advisors Roy van Pelt and Anna Vilanova for introducing me to this topic as well for their great suggestions on the way. Also, I like to thank the reviewers of my report: Roy van Pelt, Andrei Jalba, Yoka van Eijk, Daniël Eisink, Rik van Roekel and Jorn van der Pol. Without their help this report would not have been in the state it is in now. Furthermore, I would like to thank Yoka van Eijk for her explanations of the medical aspects within this project. Moreover, I would like to thank my loved ones, who have supported me throughout the process, by keeping me calm and sane when things did not work out as expected.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	v
List of Tables	vi
Abbreviations	vii
Symbols	viii
1 Introduction	1
2 Data acquisition	6
2.1 Doppler Ultrasonography	6
2.2 Phase-Contrast Magnetic Resonance Imaging	7
3 Related work	11
3.1 Fluid simulation control	12
3.2 Blood-flow simulation	13
3.3 Measurement-integrated blood-flow simulations	14
4 Comparison study of fluid-simulation techniques in computer graphics	15
4.1 Eulerian approaches	17
4.2 Lagrangian approaches	19
4.3 Hybrid approaches	21
4.4 Comparison	22
5 Blood-flow visualization based on the FLIP method	24
5.1 The Navier-Stokes Equations	25
5.2 Domain discretization	26
5.2.1 Time step size	26
5.2.2 Space discretization - Grid	27
5.2.3 Space discretization - Particles	27
5.3 Operator computation	29
5.3.1 Body forces	29
5.3.2 Advection	29

5.3.3	Projection	30
5.3.4	Preconditioned conjugate gradient algorithm	31
5.4	Boundary conditions	33
5.5	Sweeping	35
6	Measurement-simulation coupling	38
6.1	The PC-MRI-measurement-integrated method	39
6.2	The Linear Feedback Control method.	41
7	Implementation	43
7.1	QFlowExplorer	44
7.2	The PC-MRI-measurement-integrated method	44
7.3	The linear feedback control method	45
8	Qualitative evaluation	46
8.1	Comparison of simulation methods	47
8.1.1	Sensitivity of parameter γ for the Linear Feedback Control method	51
8.2	Robustness	53
8.3	Simulation compared to interpolation	55
9	Results on PC-MRI data	58
9.1	Volunteer data	59
9.2	Patient data: aortic dissection	61
10	Discussion & conclusion	64
11	Future work	66
A	Algorithms	68
	Bibliography	73

List of Figures

1.1	Anatomical structure of the thorax	2
1.2	Anatomical location of the aorta	3
1.3	Blood flow in a normal heart	4
2.1	Phases of a PC-MRI measurement.	8
2.2	Blood-flow measurements using PC-MRI.	9
2.3	The three directions of velocity.	10
5.1	The 3D MAC-Grid.	28
5.2	An example of a two dimensional signed distance function.	33
5.3	An example of the velocity weight computation.	34
5.4	An example of Fast Sweeping in two dimensions.	37
8.1	Synthetic vortex data. Used for the evaluation	48
8.2	The error of the magnitude for different methods.	50
8.3	The error of the angle for different methods.	51
8.4	The influence of different values for γ on error of the magnitude for the LFC method.	52
8.5	The influence of different values for γ on error of the angle for the LFC method.	52
8.6	The influence of different amounts of noise on error of the angle.	54
8.7	The influence of different amounts of noise on error of the magnitude.	55
8.8	The error of the angle made by interpolation of simulated data.	56
8.9	The error of the magnitude made by interpolation of simulated data.	56
9.1	Near boundary artifacts of the measurements.	59
9.2	The simulation applied on the measurements.	60
9.3	Vorticity caused by a dissection.	62
9.4	Relative pressure map of a dissection.	63

List of Tables

4.1 Performance of different fluid simulation techniques	22
--	----

Abbreviations

PC-MRI	Phase Contrast Magnetic Resonance Imaging
FLIP	Fluid-Implicit-Particle
PIC	Particle In Cell
SPH	Smoothed Particle Hydrodynamics
CG	Conjugate Gradient
PCG	Preconditioned Conjugate Gradient
MAC	Marker-And-Cell
SDF	Signed Distance Function
MIC	Modified Incomplete Cholesky
CFL condition	Courant Friedrichs Lewy condition (named after three mathematicians)
SNR	Signal to Noise Ratio
CFD	Computational Fluid Dynamics
UMI	Ultrasonic-Measurement-Integrated
LFC	Linear Feedback Control
PCMI	PC-MRI-Measurement-Integrated
CPU	Central Processing Unit
GPU	Graphics Processing Unit

Symbols

x, y, z	position in 3D relative to the origin	mm
u, v, w	speed in respectively x, y, z direction	mm/s
\vec{x}	a position in the field	(x, y, z)
\vec{u}	fluid velocity field in three dimensions	(u, v, w)
ρ	fluid density	g/mm^3
\vec{g}	body force	mm/s^2
p	pressure of the fluid per unit area	Pa
ν	kinematic viscosity	$(\text{kg}/(\text{s}\cdot\text{m}))/\rho$
t	time	s
ω	vorticity	1/s
A	area	mm^2
dt	time step size	s
dx	cell width	mm
m	mass	g
\vec{f}	force	g mm/s^2

Chapter 1

Introduction

Cardiovascular diseases (CVDs) cause 47% of all deaths in Europe and 40% in the European Union. It takes the first place on the list of the main causes of death in women in all countries of Europe, and is the number one cause of death in men in all but six countries [1]. Many studies indicate that abnormal blood flow can cause cardiovascular anomalies. Moreover, abnormalities in the vessel structure can cause aberrant blood flow. Examples include atherosclerosis [2], valve-related diseases [3] and aortic dissections [4]. Furthermore, age dependent blood-flow differences can be measured with techniques such as phase-contrast magnetic resonance imaging (PC-MRI) [5]. This study focuses on aortic blood flow.

In order to understand, prevent, or cure CVDs, a thorough understanding of the cardiovascular system and its blood flow is required. Research and cardiac investigations in clinical practice utilizes various types of non-invasive measurement techniques, such as Doppler ultrasonography, computed tomography angiography (CTA), magnetic resonance angiography (MRA), phase-contrast X-Ray imaging (PCI) and phase-contrast magnetic resonance imaging (PC-MRI). Moreover, computer simulations are used to model patient-specific blood flow under all kinds of circumstances. Both measurements and simulations have contributed to a better understanding of cardiovascular blood flow, as well as cardiovascular anomalies. Although, some hospitals opt for Doppler ultrasonography as the first choice of investigation when, for example, aortic dissection is suspected, MRI is increasingly used [6]. Doppler ultrasonography costs less, however, MRI is less noisy. Figure 1.1 shows a healthy anatomical structure of the thorax seen

from the left hand side. In Figure 1.2 the anatomical location of a healthy aorta is depicted from posterior direction.

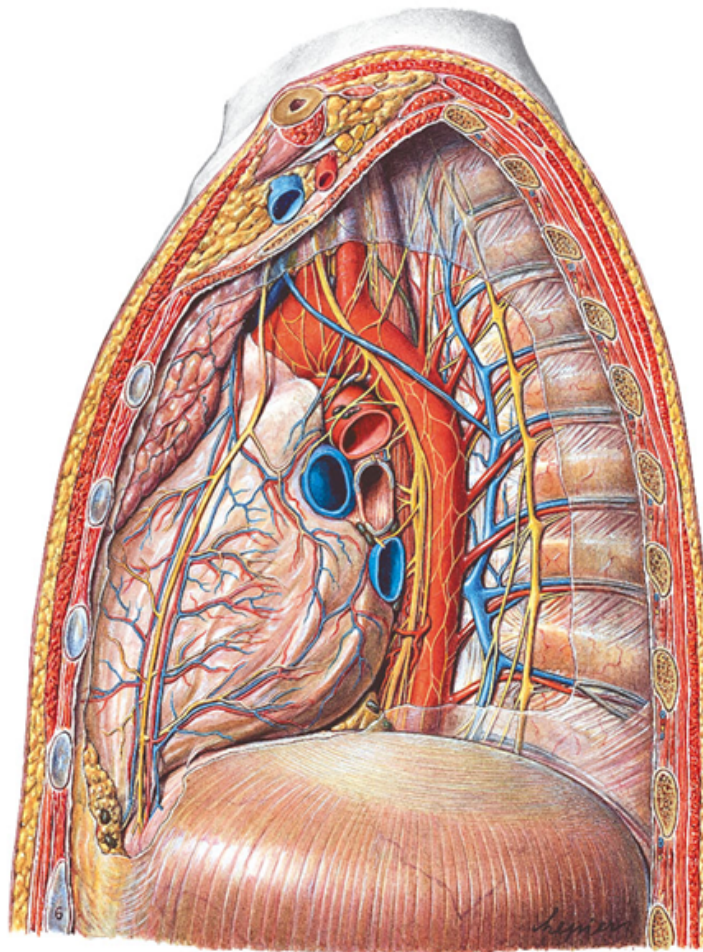


Figure 1.1: The anatomical structure of the thorax, seen from the left hand side. The heart is clearly visible, as well as the aorta coming from the upper side of the heart moving towards the the back and then downwards. Image taken from Sobotta edited by R. Putz and R. Pabst [7].

Within the heart, blood flow is generated by alternating blood pressures. These alternating blood pressures are formed due to varying degrees of contraction of the heart. One heartbeat, or one cardiac cycle as depicted in Figure 1.3, consists of the systolic phase, followed by the diastolic phase. During the systolic phase, as a result of the contracting ventricles, the heart will pump blood into the aorta. Approximately halfway the systolic phase we find the peak systemic arterial blood pressure, which is produced by the transmission of the pressure generated by the contracting ventricles. Subsequently, the systolic pressure will decrease, which gives rise to the diastolic phase, in which the heart will fill itself with blood again. For the specific moment during the heartbeat, where we speak of peak systemic arterial blood pressure, the term peak systole is commonly used. To maintain diastolic blood pressure, and therefore optimal functioning of the

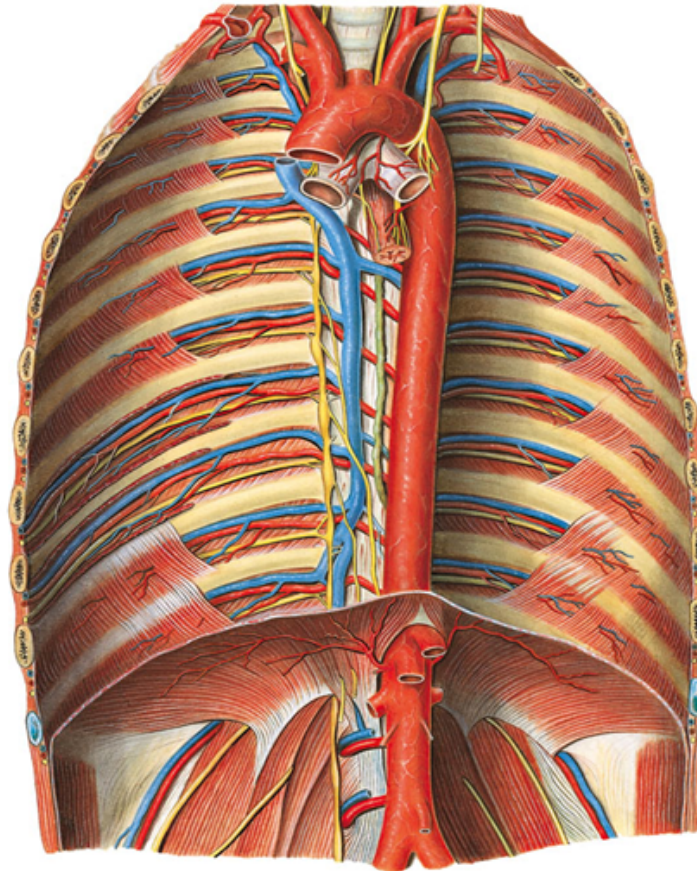


Figure 1.2: The anatomical location of the aorta in the thorax, depicted from posterior direction. Image taken from Sobotta edited by R. Putz and R. Pabst [7].

heart, vascular tone and the presence of intact valves are crucial [8]. By vascular tone is meant the degree of constriction experienced by a blood vessel relative to its maximally dilated state. When abnormalities in blood flow are detected by using techniques such as Doppler ultrasonography or PC-MRI, the chance of finding CVDs increases significantly.

These measurement techniques all have a limited spatial-temporal resolution. To ensure that the temporal interpolation between time points of the measurement is based on the physics of fluid mechanics, this project focuses on combining PC-MRI measurements with simulation techniques from the computer graphics field. The main advantage of simulation techniques in computer graphics is that they are generally real-time, and therefore can be incorporated interactively. This makes it possible to change the settings of the simulation during runtime. Furthermore, it saves processing time. This in contrast to the current more complex blood-flow simulations.

Most of these blood-flow simulations are initialized with a patient-specific mesh of the vessel wall obtained from measurements. Furthermore, the in- and outflow conditions of

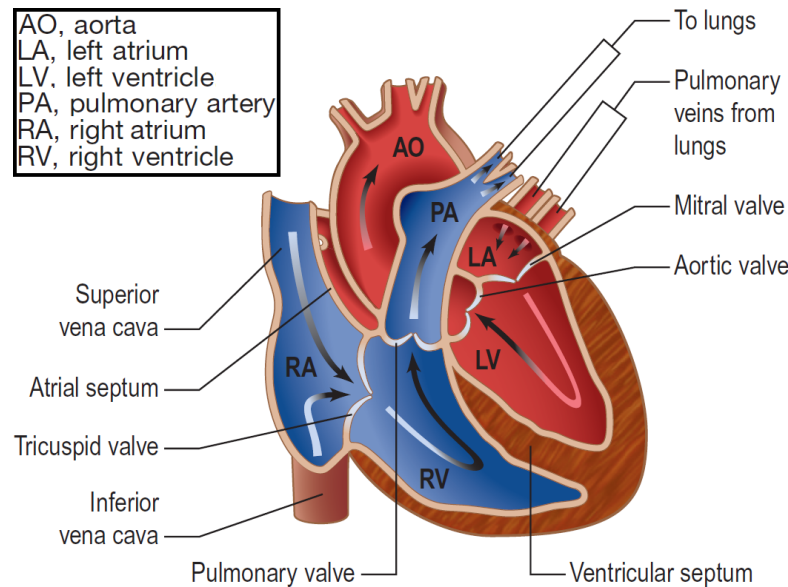


Figure 1.3: The blood flow in a normal heart. Oxygen-poor blood (blue) from the inferior and superior vena cava streams into right atrium during the diastolic phase, to subsequently stream into the right ventricle. The blood flow will then continue to stream into the truncus pulmonalis, or pulmonary artery, for the blood to be oxygenated in the lungs. Oxygen-rich blood (red) will then flow, via the pulmonary veins, into the left atrium and then the left ventricle, to be pumped into the aorta during the systolic phase. Image taken from P. Kumar and M. Clark [6].

the mesh are determined using measurements, however, these only specify the in- and outflow. This means no specification of the flow is given for the rest of the simulation. Moreover, these methods take a lot of computations, and thus are slow compared to the simulations used in computer graphics.

In this thesis, a novel technique is proposed to combine PC-MRI measurements with a simulation. This technique, called the the PC-MRI-measurement-integrated method (PCMI), is applied on our fluid simulation technique from the computer graphics field. To the best of our knowledge, this has not been done before. We can improve the temporal interpolation within physical constraints. Also, we see opportunities for prognosis and interactive treatment assessment in the future.

This thesis is structured as follows: first the data acquisition is explained in Chapter 2, then the related work in the field will be discussed in Chapter 3. Subsequently, in Chapter 4, a comparison study on the different fluid simulation techniques in computer graphics is conducted. In this chapter a suitable simulation technique is selected. Chapter 5 describes the selected fluid simulation in detail. In Chapter 6, four different measurement

coupling techniques are explained; no method, velocity field replacement, the linear feedback control (LFC) method by Kim [9] and our novel PC-MRI-measurement-integrated (PCMI) method based on the ultrasound-measurement-integrated (UMI) method by Funamoto [10]. In Chapter 7, the implementation details and used tools are described. The techniques, described in Chapter 6, are evaluated and the best performing technique is selected in Chapter 8. In this chapter also robustness is tested and a comparison with conventional linear interpolation is done. Chapter 9 shows the results, presenting the performance of the simulation and the selected coupling method on both volunteer and patient data. The conclusion is given in Chapter 10. Lastly, future work is discussed in Chapter 11.

Chapter 2

Data acquisition

2.1 Doppler Ultrasonography

The most common method to measure blood-flow velocity data is by using Doppler ultrasonography. The main reason for Doppler ultrasonography to be so commonly used, despite the relatively high amount of noise, is that it has a high spatial-temporal resolution. Moreover, Doppler ultrasonography is relatively inexpensive. In Doppler ultrasonography, ultrasound is used to measure the Doppler velocity of the blood flow. Ultrasonography measures the reflected frequencies of the ultrasound signal. However, moving particles, in this case blood cells, cause a shift in frequency when moving in the direction of the ultrasound beam. Doppler ultrasonography measures this frequency shift. Due to the restriction of the beam direction it is rather difficult to recognize the exact three-dimensional velocity field of the blood flow. Therefore, a complete velocity field is not directly available. This also causes a limited field of view. Moreover, nearby vessels can produce a so-called shadow in the measurements hiding other vessels.

Currently, several diseases can be diagnosed using echocardiography, which is a form of ultrasonography particularly concerning the heart. Inspected diseases include valvular stenosis, valvular regurgitation, aortic aneurysms and aortic dissections. For example, in case of mitral stenosis, peak, mean and end-diastolic pressure gradients can be obtained using continuous-wave (CW) Doppler, a form of echocardiography that collects all the velocity data and analyzes it. Also, dilatation of the aorta, as seen in aneurysms

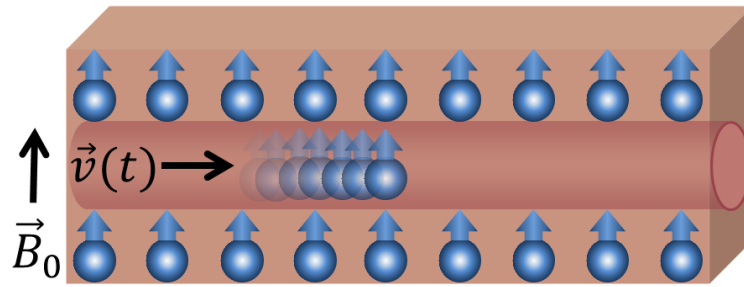
and dissections, and which is strongly suggestive for the diagnosis, can be measured accurately with the aid of echocardiography [6].

2.2 Phase-Contrast Magnetic Resonance Imaging

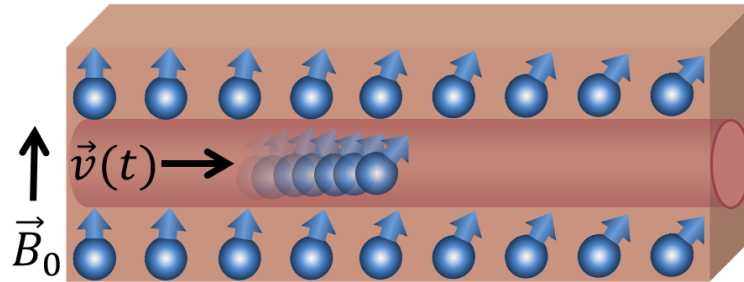
The data used in this project is obtained by Phase-Contrast Magnetic Resonance Imaging (PC-MRI). This is a technique where Magnetic Resonance Imaging (MRI) is used to measure blood-flow velocities in arteries. It provides 3D images of the blood flow over time, and therefore this data is often referred to as 4D blood-flow data. However, clinically, 2D images of the blood flow over time are used. Visual inspection of healthy individuals and patients with anomalies yields better understanding of hemodynamics in general, and potentially leads to better diagnosis and treatment.

MRI uses the fact that nuclei of atoms have a physical property called spin. This spin has a direction that can be influenced by magnetic fields as produced by MRI scanners. This magnetic field aligns the spin of the hydrogen nuclei in the magnetic field direction. By superimposing small spatially varying magnetic fields, called gradient fields, it is possible to make the spin rotate around a specified axis at specified locations. This rotation can be measured, and the amount of measurements in a region corresponds to the density of hydrogen nuclei of the region. Also, each kind of tissue has a certain amount of hydrogen nuclei, and each type of tissue thus has its own specific density. Therefore, a measurement yields a slice of the density of the measured tissue [11]. This process can be repeated to get multiple slices, and also a three dimensional image of the anatomical structure. However, MRI is sensitive to flow as well as movement, when flow or movement is measured a shift in spin occurs, and thus a low or no density is measured at that point.

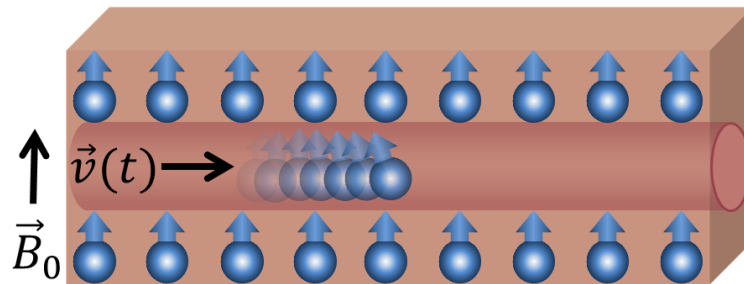
PC-MRI applies a constant magnetic field to align all spins, as depicted by Figure 2.1a. Then a linear gradient field is applied, so spins get a location-specific direction, while moving nuclei get a spin that varies, as depicted by Figure 2.1b. Finally the negative linear gradient is applied to ensure all atoms, which do not have any velocity owing to the flow, have a spin equal to the situation beforehand, while moving atoms have a spin that is linearly depending on their velocity, as depicted by Figure 2.1c. By measuring this spin the component in the direction of the linear gradient of velocity can be derived. By



a) Constant magnetic field



b) Linear positive gradient magnetic field



c) Linear negative gradient magnetic field

Figure 2.1: Different phases (a, b, c) of a PC-MRI measurement with different magnetic fields. Figure based on Van Pelt [11] and J. Lotz et al.[12].

using the magnetic field with a linear gradient in all three dimensions, a three-directional velocity field can be made. Typically a measurement of cardiac blood-flow has a spatial resolution of $2.0 \times 2.0 \times 2.5\text{mm}$, for $128 \times 128 \times 50$ voxels. The temporal resolution is typically 50ms, yielding 20 to 25 measurements throughout the cardiac cycle [11]. In Figure 2.2 a three-dimensional PC-MRI measurement of the aorta is depicted. It shows the magnitude of the measured velocities. Figure 2.3 depicts the velocity in the three measured directions, X, Y and Z.

Using PC-MRI it is possible to diagnose CVDs due to an altered or aberrant flow, including dissections [4], atherosclerosis [2], stenosis [13] and valve-related diseases [3].

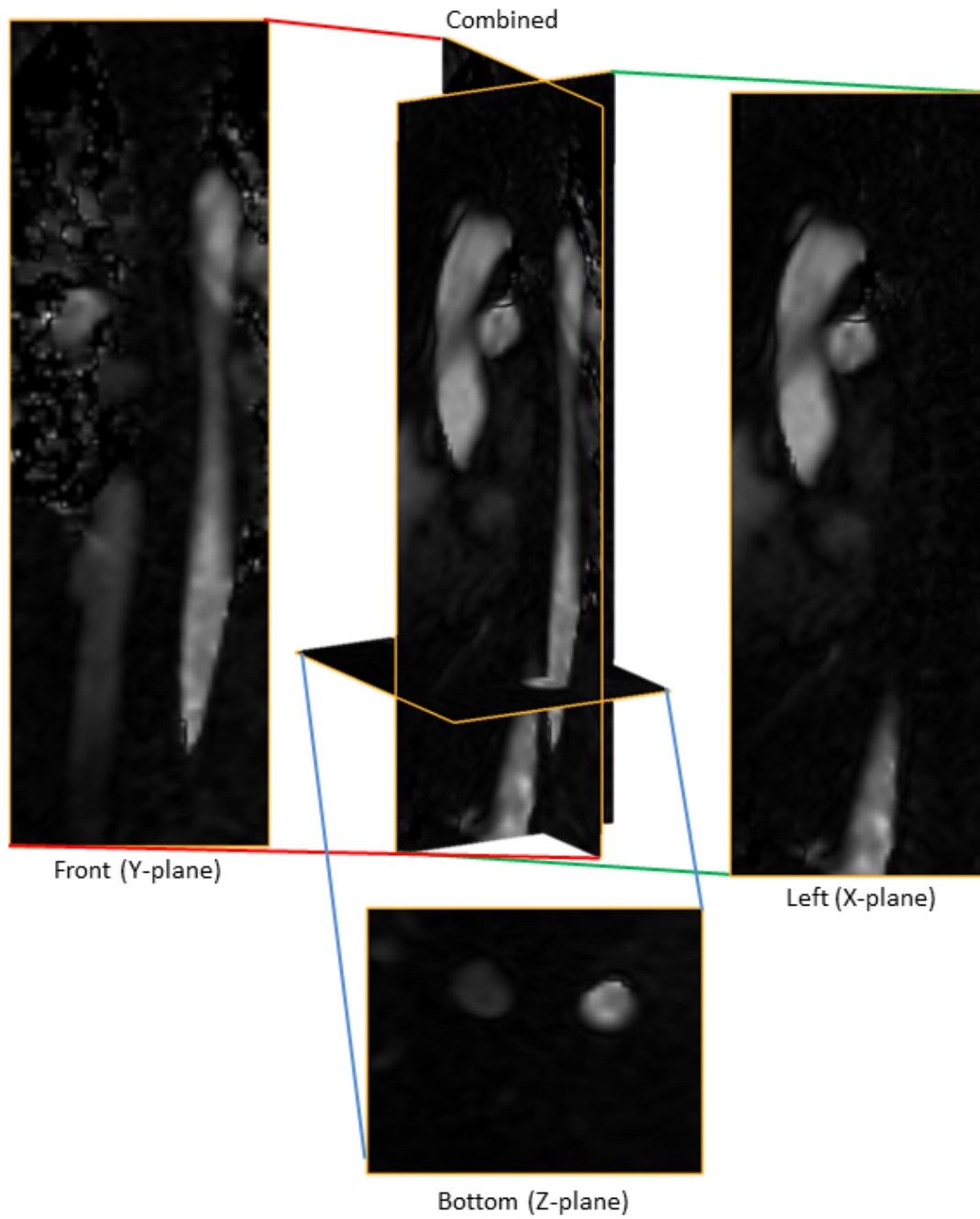


Figure 2.2: 3D time-resolved blood-flow measurements using PC-MRI. Here the velocity is shown by means of the magnitude. Black represents a low velocity and white is used for high velocities.

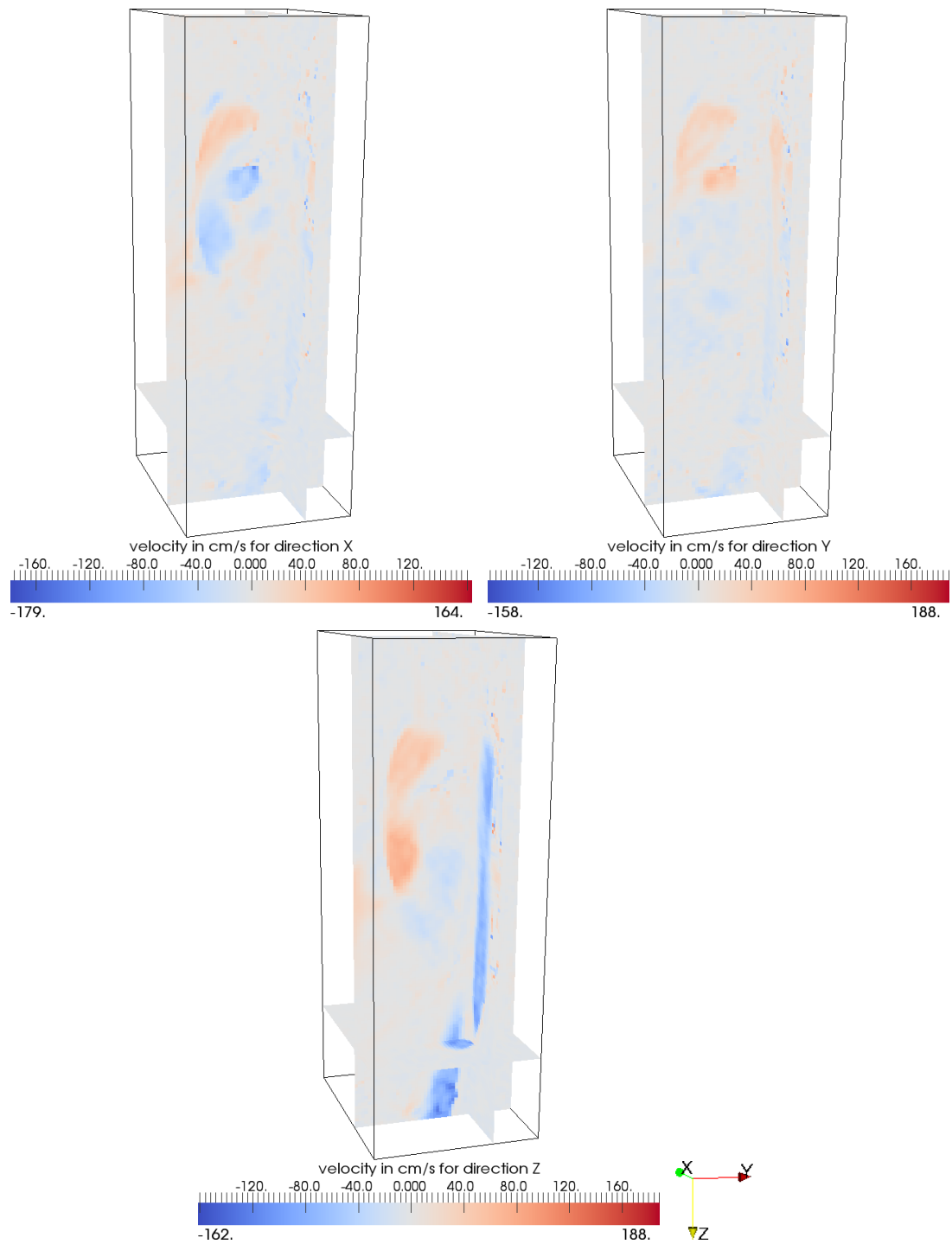


Figure 2.3: The velocity in the X, Y and Z directions as measured is shown. Blue represents the negative velocity and red is used for the positive velocity relative to the normal plane of the direction. Gray represents zero velocity. The orientation is shown by the arrows.

Chapter 3

Related work

4D PC-MRI of blood flow, as described in Chapter 2, helps doctors to identify anomalies and get a better understanding of blood flow inside intracranial vessels [14] and the cardiac system [15, 16]. To overcome the coarse temporal resolution of PC-MRI, temporal interpolation is often used. Various interpolation techniques are used, such as temporal interpolation for particle path tracing with Runge-Kutta 4 [11]. Schwenke et al. [17] present a technique propagates the velocity field from the measurements over time using the so-called Fast Sweeping technique, however, this method is not based on fluid mechanics. This project tries to improve the temporal interpolation by using a simulation technique from the computer graphics field. This technique should be driven by the given PC-MRI measurements, while maintaining a physically-correct model of the blood flow. That is, the fluid should be *incompressible*, and based on the fluid mechanics, i.e. the so-called Navier-Stokes equations. For this project, a fluid simulation from the computer-graphics field is selected, since such simulations are fast and physically correct. However, they often are less accurate compared to contemporary blood-flow simulations. However, this inaccuracy of fluid simulations from the computer-graphics field is depending on the amount of calculations allowed per time step. This chapter will discuss the related work of respectively control methods for fluid simulations in computer graphics, subsequently, we look at blood-flow simulations and control methods for blood-flow simulations.

3.1 Fluid simulation control

For this project, controllability of the fluid simulation is required to ensure that the blood-flow simulation incorporates the given measurements. Simulations in computer graphics often require controllability to allow animators to produce various effects, such as maelstroms and characters made of fluid. To be of use for animators, the simulations should be interactive, easy to use and controllable, also the fluid should look realistic.

In this section, different methods of controllability are discussed. Foster and Metaxas [18] were one of the first to enable control of the animated fluid for animators that do not have a full understanding of the underlying physics, while maintaining a relatively low number of computations. They provide different control mechanisms to allow the animator to set an external and internal pressure field, a velocity field and (moving) boundary conditions. These controls directly influence the properties of the modeled fluid. However, the control for the animator is limited, and details of the flow are hard to model. Therefore, different methods were developed that allow more control and more detailed effects. Such a method is given by Hong et al. [19]. They define a potential field, which can be regarded as an extra dimension acting as a sort of height map. That is, the fluid flows down, to regions with a lower potential. This allows specific control of the shape of the fluid, while the incompressibility is maintained. The potential is applied in the fluid simulation by taking the negative gradient, resulting in a force that directs the fluid to flow to the regions with low potential. With this method it is relatively complex to make, for example, the fluid follow a certain path defined by the animator.

The method by Kim [9] makes it possible to let the fluid follow a defined line or shape. To do so, a linear control force is applied that directs the fluid towards the line. This force is derived from a given target velocity field, and the velocity of the simulation. This allows the simulation to match with the target velocity field after a number of iterations, while maintaining the incompressibility of the fluid. For this method, the animator only has to draw the line or shape to be followed over time and is thus easy to use. However none of these control method allow the use of a target velocity field provided in our case by the PC-MRI measurements, which are only available at certain times during the simulation.

3.2 Blood-flow simulation

Blood-flow simulations are generally performed using computational fluid dynamics (CFD) methods, because of their scalability and their high temporal and spatial resolutions, compared to measurements. Another advantage is that minimal patient involvement is required; this is in contrast to flow measurements such as PC-MRI, which require a large scan time and the patient has to be motionless during these scans. Another advantage of CFD over measurements is that one can use models of any anomaly and experiment with treatments, without the need of multiple of patients. Furthermore complex operations can be planned using the simulations.

While CFD methods only provide a model of the actual blood flow, this model is accurate as shown in the papers by Kong et al. [20], Theresia et al. [21] and Ooij [22]. In these papers, PC-MRI measurements of flow in phantoms and patients are compared with a CFD simulation, yielding comparable results.

Ivankovic et al. [23] developed a method that aims for an early diagnosis of atherosclerosis. For this they use a technique called the finite volume method, which divides the simulated space in small discrete control volumes, and solves the partial derivative equalities of fluid mechanics for these volumes using restrictions on the boundaries of the volumes. A similar technique, the finite element method (FEM), discretizes the simulated space into so-called elements e.g. triangles in 2D and tetrahedra in 3D. FEM then connects the many relatively simple local element equations over many small domains within the simulation, named finite elements, to approximate the more complex equation of the full domain. By using this kind of discretization it is possible to set the level of detail by using smaller volumes or elements. FEM is used successfully to model blood flow in the brain [24, 25], the aorta and hearth [26, 27].

A different technique is used by Figueroa et al. [28]. Their technique Coupled Momentum Method for Fluid-Solid Interaction (CMM-FSI) models the elasticity of the vessel walls. It models the elasticity of the tissue of the vessel wall by using the physical laws for elasticity. This method can also be used to model large scale blood flow of all major arteries as well as detailed blood flow within single arteries, as shown by Xiao et al. [29].

3.3 Measurement-integrated blood-flow simulations

This study tries to combine simulation techniques from the computer graphics fields with PC-MRI measurements. This should result in a physics based temporal interpolation between measurements. Furthermore, the fluid simulation is based on the physical laws of fluid mechanics, and therefore, yields a temporal interpolation that is based on the fluid mechanics laws.

The combination of blood-flow measurements and simulations exists in the Doppler ultrasonography by means of ultrasound-measurement-integrated (UMI) simulation. This method, first described by Funamoto et al. [10], uses the difference in the projected measured velocity and the velocity field of the simulation, Section 6.1. From this difference, feedback signals are generated. These signals define a local force, which is used to direct the simulation towards the measurements. However, in contrast to PC-MRI, ultrasound provides a high spatial and temporal resolution. Therefore, this technique cannot be applied on PC-MRI measurements directly.

To the best of our knowledge no temporal interpolation method exists for PC-MRI measurements based on the fluid mechanic laws. Therefore, this study proposes a novel coupling method for a computer-graphics simulations with PC-MRI measurements. This method is based on the UMI method described in this section.

Chapter 4

Comparison study of fluid-simulation techniques in computer graphics

Several fluid simulation techniques exist in computer graphics. This chapter provides a brief comparison between these techniques. From the discussed techniques one is chosen to be used as basis for our approach. The technique should model physically-correct blood flow, that is, within the made assumptions. The fluid simulation should model blood as incompressible, without viscosity and with low computational costs. By using the Navier-Stokes equations, which describe the behavior of fluid in physics, this is approximated. When modeling small arteries the compressibility of blood is important [23], however, blood-flow in larger structures is usually modeled using incompressible fluid [28]. The viscosity can be dropped since blood is not highly viscous nor do we model small scale fluid behavior [30]. In this chapter, first the physics behind fluid mechanics are explained. Then, different approaches for fluid simulations in computer graphics applying these physics are explained and compared.

Every physically-based fluid simulation is based on the Navier-Stokes equations, the physical equations that describe fluid behavior. These equations consist of a momentum equation given by Equation 4.1, and an incompressibility condition given by Equation 4.2. The latter is optional, depending on whether the modeled fluid is incompressible or

not.

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \cdot \nabla p = \vec{g} + \nu \nabla^2 \vec{u}, \quad (4.1)$$

$$\nabla \cdot \vec{u} = 0. \quad (4.2)$$

In these equations the velocity field $\vec{u}(\vec{x}, t)$ of the fluid in three dimensions is defined by three components of velocity, namely (u, v, w) . A position in the field \vec{x} is given by the three directions (x, y, z) . Time is denoted by t . The density of the fluid is given by ρ , which may vary within the fluid. The pressure p denotes the force per unit area that the fluid exerts. To take body forces, e.g. gravity, into account, the force \vec{g} is used, exerting a uniform force on the fluid as a whole. The kinematic viscosity is denoted by ν and is given by the dynamic viscosity η of the fluid divided by pressure. Note that Equation 4.1 does not directly relate to fluid mechanics. Rewriting Equation 4.1 and filling in the formula for kinematic viscosity makes it possible to assign fluid mechanical properties to different terms as shown in Equation 4.3.

$$\rho \left(\underbrace{\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}}_{\text{inertia}} \right) = \underbrace{-\nabla p}_{\text{pressure force}} + \underbrace{\eta \nabla^2 \vec{u}}_{\text{viscosity}} + \underbrace{\vec{g}}_{\text{body forces}} \cdot \rho \quad (4.3)$$

Here advection is the process of transport of a quantity through a fluid, such as velocity, heat, density, etc. The properties at a point A are transported over time to point B. Another type of transport, namely diffusion, also carries the property of a point directly to all neighboring points. The viscosity of a fluid describes the resistance to deformation. A highly-viscous fluid, such as honey or lava, tries to maintain its shape more than for example water. The divergence of a velocity field (so also of a fluid) measures the magnitude of sources and sinks at a given point. A divergence-free liquid therefore does not contain such sources, nor sinks, and thus can be said to be *incompressible*. By using the Helmholtz-Hodge decomposition, any velocity field can be made divergence-free by subtracting the pressure gradient from the velocity field [30]. This means the pressure should be calculated from the velocity field. Another important property of fluid is *vorticity*. Vorticity is the rotational movement around an axis inside a fluid. Every axis with such a rotation is called a vortex.

In fluid simulation for computer graphics, two distinct approaches are used to model fluid, namely the Eulerian, grid-based, approach and the Lagrangian, particle-based, approach. Each method tries to mimic the continuity of fluids by solving the fluid equations on a finite number of fixed “measurement” points, spread out in the space continuum in which the fluid exists. This discretizes the computations such that only a finite number of calculations have to be done. The main difference between these methods is given by these measurement points, for the Eulerian approach these are positioned on a fixed grid, and thus have a static location, whereas for the Lagrangian approach, the measurement points evolve along with the fluid. Both methods fill up the continuum by using interpolation between the measuring points. The following properties are in most cases important, with respect to fluid simulation in computer graphics:

- Physically correct
- Low computational cost
- Boundary conditions
- Incompressibility

Physical correctness means that the Navier-Stokes equations should be approximated as close as possible, this includes advection and vorticity; furthermore, the fluid should look realistic. Low computational cost is required to make the simulation run fast and thus more interactive; this requirement, however, hampers the physical correctness by allowing less detail. It is therefore important to make a trade off between physical correctness and required time per frame. The boundary conditions of the fluid should be obeyed in a natural manner, such that the boundaries of solids are taken into account in a realistic manner. Divergence-freeness is required for (almost) incompressible fluids, such as water and oil.

4.1 Eulerian approaches

Stam [31] proposed one of the first stable fluid simulation based on the Navier-Stokes equations. The paper introduced a semi-Lagrangian approach for the advection: the

value of a quantity at a grid point is derived by backtracking through the velocity field from the current grid point p_g to find a point p_i . The value of this point p_i is then interpolated to retrieve the value at point p_g , which is stored in a new grid. This method is semi-Lagrangian because the backtracking can be regarded as if a virtual particle's previous location was traced. However, no real particles are used in the system. By using this approach, the Eulerian approach is unconditionally-stable with regard to advection. Unfortunately, this method yields numerical dissipation over time, due to rounding errors in the interpolation.

Diffusion is relatively easy as we can spread out the value stored at a grid cell to the neighboring grid points. For viscosity an additional term of the Navier-Stokes equation must be solved to model the resistance of the fluid to shape changes.

An advantage of this method is that boundary conditions, such as walls, can be modeled easily by defining some cells to be solid, and alter the fluid at the boundaries of this solid cell. This is done mainly by setting the velocity in the direction of the normal of the solid to the velocity of the solid in that direction, thus zero if the solid does not move, as done by Bridson [30]. This way, one enforces that no fluid leaks inside the solid. To allow more curved, non-grid aligned boundaries, additional tricks are needed as applied by Chentanez et al. [32], also enabling deformable bodies. Zhu and Bridson [33] and Batty et al. [34] define the solids on a grid using a "Signed Distance Function" (SDF), which results in some inaccuracy of the actual location of the boundary, but enables relatively simple coupling between solids and fluids.

The strength of the Eulerian approach mainly lies in the fact that it can enforce incompressibility, since the velocity field is stored on a grid, as well as the pressure. By using, for example, the preconditioned conjugate gradient (PCG) algorithm (preconditioned by a Poisson matrix), the pressure of the velocity field can be computed. This pressure then can be used to make the velocity field divergence-free; for a detailed overview of the PCG algorithm see the book by Bridson [30] and Appendix A.

A main drawback of the Eulerian approach is the lack of details such as vorticity, which can be added using a technique called vorticity confinement. Vorticity confinement, introduced by Fedkiw et al. [35], adds small scale detail to the flow. This additional vorticity is not based on any physical equivalence, but is needed to enforce that vortices in the fluid do not damp out too fast, which is caused by undesired smoothing due to the

interpolations done on the grid. Another disadvantage is that the computations need to solve one time step are related to the size of the grid instead of the amount of fluid.

4.2 Lagrangian approaches

For almost every Lagrangian approach it is hard to make the fluid incompressible. This is mainly due the lack of structure between the measurement points. No advanced mathematical algorithms, such as available for grids, can be applied (directly) as these require a matrix with information and thus a grid. However two methods have been developed that attempt to overcome this problem. Smoothed Particle Hydrodynamics (SPH) uses particles that represent the fluid, while the vortex method uses particles that represent vorticity, so-called vortons, to model the velocity field of the fluid.

SPH– Smoothed Particle Hydrodynamics (SPH), introduced in the field of astronomy by Gingold and Monaghan [36], is a technique that uses Equation 4.4 for interpolating a scalar quantity A at a position \vec{x} :

$$A(\vec{x}) = \sum_j m_j \frac{A_j}{\rho_j} W(\vec{x} - \vec{x}_j, h), \quad (4.4)$$

where j iterates over all particles, m_j is the mass, \vec{x}_j is the location, ρ_j the density and A_j the quantity of A of the particle j , and W is a smoothing kernel with core radius h . Computing the gradient (∇) or the Laplacian (∇^2) of A , is relatively simple.

$$\nabla A(\vec{x}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\vec{x} - \vec{x}_j, h), \quad (4.5)$$

$$\nabla^2 A(\vec{x}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\vec{x} - \vec{x}_j, h), \quad (4.6)$$

This also makes the calculations of the pressure easy, which can be used to make the fluid incompressible, since the pressure is a scalar value. The technique was first used by Müller et al. [37] for fluid simulations, in a computer-graphics setting.

In SPH the velocity of a particle has a certain range of influence; typically this range is determined using a kernel function W as described above, which takes the distance to the particle as input. Its value decreases with increasing distance. This means a particle only influences nearby particles, which increases the performance, if particles are stored

such that the neighboring particles are known to a particle, as introduced by Harada et al. [38] for Graphics Processing Units (GPUs).

Due to the fact that particles store their own velocity, the advection step becomes implicit. Every time step they simply move using their stored velocity. Viscosity is modeled in a natural fashion by reducing the neighboring particles' velocity, depending on viscosity, and thus making the fluid resist deformation. However, it is very hard to make SPH incompressible, as stated by Hong et al. [39].

The main advantages are its high performance due to the fact that only particles are needed where fluid exists, and the relative simple per particle computation. As in the paper by Harada et al. [38], boundary conditions are taken into account by placing particles on the boundary of solids, and take these particles with infinity mass into account when solving the equations. This, however, increases the computation time due to the many additional particles.

Vortex particle method– The vortex particle method, introduced by Rosenhead [40], models the Navier-Stokes formulas by means of vorticity. Like SPH, the vortex particle method uses particles to represent parcels of fluid. SPH, however, directly solves the momentum equation, whereas the vortex particle methods solve the vorticity equation. This equation rewrites the Navier-Stokes equations 4.3 by amending the velocity with the vorticity $\vec{\omega}$ defined as $\vec{\omega} = \nabla \times \vec{u}$. This results in Equation 4.7, where $\vec{\tau}$ is an external torque, which can be added to model external forces. This torque is defined as $\vec{\tau} = \vec{r} \times \vec{F}$, where \vec{r} is the vector from which the torque is measured to the point where to force \vec{F} is applied. The buoyancy is used to ensure incompressibility, as it contains the pressure. Buoyancy is defined as the force exerted by a displaced volume inside the fluid, and thus it depends only on the density and pressure.

$$\underbrace{\frac{\delta \vec{\omega}}{\delta t}}_{\text{Change in vorticity}} = \underbrace{\vec{\omega} \cdot \nabla \vec{u}}_{\text{stretching/tilting}} + \underbrace{\nu \nabla^2 \vec{\omega}}_{\text{viscous diffusion}} + \underbrace{\frac{\nabla \rho \times \nabla p}{\rho^2}}_{\text{buoyancy}} + \underbrace{\vec{\tau}}_{\text{torque}} \quad (4.7)$$

The vortex method uses vorticity of vortons, i.e. particles that represent a vortex in the fluid. For advection and viscosity, a technique as in SPH is used. This kernel function states the effect of a vorton on neighboring vortons. This is done by translating the vorticity of a vorton to a local velocity field.

Due to the use of particles, incompressibility is hard to model according to the same argument as why SPH is hard to make incompressible. Furthermore, advection is not trivial since the velocity field is only stored implicitly in the vortons, and thus more computations are needed for the advection compared to SPH. Like in SPH, the boundaries are normally modeled by using fixed vortons on the boundary as done by Park et al. [41]. By modeling the fluid using vortons, the simulation exhibits more small-scale rotational features, due to the vorticity common in explosions and smoke, as shown by Selle et al. [42].

4.3 Hybrid approaches

Since every of these techniques has its own advantages, hybrid approaches between Eulerian and Lagrangian methods exist. For making Lagrangian methods incompressible, often grids are used to solve the incompressibility term, while maintaining the high details of the particles. This combination of a Lagrangian approach with an additional grid for incompressibility, is done for SPH by Hong [39], Losasso [43] and for the vortex method by Selle [42]. This, however, has the disadvantage that two techniques should be solved every time step.

Other hybrid approaches are the Particle-In-Cell (PIC) method, introduced by Harlow in 1963 [44], and FLuid Implicit Particle (FLIP) method, introduced by Brackbill et al. in 1965 [45]. These methods use both particles and grids. However the particles are used in the advection step to have small scale features, but the velocity of the particles is projected on a grid, which is then used to ensure incompressibility. The differences between PIC and FLIP is in the translation from the divergence-free grid velocity to the particles. In the PIC method the velocity of the particles is substituted, while in the FLIP method only the difference in velocity is added to the particles velocity. This allows FLIP to have nearly no numerical dissipation, and thus no vorticity confinement is needed. FLIP is nowadays used by many fluid simulations, for examples see Zhu et al. [33] and Batty et al. [34].

TABLE 4.1: Performance of different fluid simulation techniques from the computer graphics field with respect to the requirements. Here a “-” means the technique does not perform well for the given property. An “o” means the technique can handle the property up to a certain level. The “+” means the technique performs good on the specified property

Properties	Eulerian	SPH	Vortex particle method
Computation time	-	+	+
Advection	-	+	o
Diffusion	+	+	+
Boundary conditions	o	+	+
Enforce incompressibility	+	-	-
Vorticity	-	o	+

4.4 Comparison

Table 4.1 shows an overview of the properties of the different fluid simulation techniques described in Sections 4.1 and 4.2. The score is based on the known weaknesses, as described and referenced in these sections. Note that hybrid methods are not taken into account. Overall the Lagrangian methods, SPH and the vortex particle method provide higher detail, compared to the Eulerian approaches, by means of vorticity and advection, and exact boundary conditions, while having a lower computation time. Furthermore, Lagrangian approaches allow a per-particle implementation, which reduces the overall complexity. Also scaling is straightforward by adding additional particles. The main disadvantage of these methods is the fact that it is hard to make these simulations incompressible.

The Eulerian approaches can guarantee incompressibility. However, the methods are hard to scale, e.g., many grid cells might be empty and thus not contain any fluid, however, they are still taken into account by the computations. Also modeling the boundaries is not very precise and requires a high density grid when highly detailed boundaries should be taken into account. Furthermore, due to numerical dissipation, many small scale features will smoothen away.

For our application the fluid must be incompressible, and hence a grid-based solution is required. However, the small scale features might also be of interest for physicians, which are given by the use of particles. Therefore, hybrid techniques come to mind. Using both a grid and particles solves the incompressibility issues, while maintaining the high amount of details. However, implementing a hybrid that is Lagrangian based, requires a full implementation of this Lagrangian method. Moreover, also an implementation of

an Eulerian approach is required to get the incompressibility. This incompressibility of the Eulerian method is the most computation-intensive step of all Eulerian methods. By using such hybrids, the computations of both the Eulerian and the Lagrangian method are summed up.

The hybrid methods remaining are PIC and FLIP. Both these methods also use an Eulerian grid for the incompressibility. However, the implemented particles are much simpler than the particles used by the other Lagrangian methods. These particles therefore use less computations compared to the Lagrangian methods. Due to this, the advantages of both Eulerian methods and Lagrangian methods are exploited without a big increase in terms of computations. To ensure that the numerical dissipation is minimal, FLIP is preferred over PIC. By above reasoning FLIP is selected as the best suiting fluid simulation technique for this project. Therefore, a more detailed explanation of FLIP will be provided in Chapter 5.

Chapter 5

Blood-flow visualization based on the FLIP method

Due to the outcome of the comparison study in Chapter 4, here we explain the fluid implicit particle (FLIP) method, used on our approach. Despite when modeling small arteries the compressibility of blood is important [23], blood-flow in larger structures is usually modeled using incompressible fluid [28]. This is because blood is only minimally compressible. Therefore, when modeling on a large scale, this compressibility is so minimal that the fluid is nearly incompressible. The viscosity, as stated by Bridson [30], can be dropped, since blood is not highly viscous nor do we model small-scale fluid flows. Furthermore, the aorta is not a static boundary for the blood. However modeling elasticity is complex [32]. Also the real boundary is unknown but a good approximation can be made at peak-systole. Therefore only the boundary at peak systole is used in our approach. In conclusion the modeled blood should be incompressible, is inviscid and the aorta is modeled using a static solid boundary. The method described in this chapter, the FLIP method, allows modeling blood-flow under these assumptions.

FLIP is a hybrid approach as it uses both the Eulerian (grid-based) and the Lagrangian (particle-based) approaches. The FLIP method (first introduced by Brackbil et al. [45]) described in this chapter uses a grid structure based on the book by Bridson [30]. The particle structure is based on the method by Zhu et al. [33], while the solid-fluid coupling and solver is done using the method described by Batty et al. [34]. The coupling between the grid and particles is made to overcome the weaknesses of both the Eulerian and the

Lagrangian methods, as explained in Chapter 4.4. The main purpose of the grid is to ensure a divergence-free velocity field, while the particles have no numerical dissipation in the advection step when using FLIP.

5.1 The Navier-Stokes Equations

In Chapter 4, we have described the Navier-Stokes equations, given by Equations 4.1 and 4.2. These equations govern the physics of fluid mechanics. Assuming the viscosity $\eta = 0$ as explained, these more general equations are reduced to:

$$\frac{\partial \vec{u}}{\partial t} = \vec{g} - \vec{u} \cdot \nabla \vec{u} - \frac{1}{\rho} \nabla p, \quad (5.1)$$

$$\frac{\partial \rho}{\partial t} = -\nabla \vec{u} \cdot \nabla \rho, \quad (5.2)$$

$$\nabla \cdot \vec{u} = 0, \quad (5.3)$$

where the velocity field \vec{u} is an abbreviation for $\vec{u}(\vec{x}, t)$, the velocity at location \vec{x} at time t . \vec{g} is the body force, ρ the density and p the pressure. To solve Equations 5.1, 5.2 and 5.3, a discretization is required. This discretization separates the continuous spatial and temporal domain into discrete spatial and temporal domains. These discrete spatial and temporal domains require a finite number of calculations, such that it can be solved in a finite number of steps.

Due to the complexity of the equations, they are first divided into smaller, simpler-to-solve equations. To do so *operator splitting* is used, so that every part is efficiently solved using dedicated methods. Equation 5.1 defines the derivative of the velocity field over time. This derivative should be integrated for \vec{u} to get the current solution \vec{u}_{new} at time t_{new} , given $dt = t_{new} - t_{old}$. It is important to notice that the order in which the terms of the equations are solved matters. The body force should be added before the advection step to ensure it is taken into account. The advection itself needs a divergence-free velocity field to be correct, but may yield a non-divergence-free velocity field. The density is required for the pressure. The pressure is used to make the velocity field divergence-free again for the next step. Note that the velocity \vec{u} and pressure p are coupled. The idea is to decouple \vec{u} and p , such that the incompressibility (pressure) term is separated from the rest of the equation. Furthermore, the body force term can

be decoupled. Splitting the equations in the right order yields the following partial differential equations (PDEs):

$$BF(\vec{u}, dt) : \frac{\partial \vec{u}}{\partial t} = \vec{g} : \text{body force}, \quad (5.4)$$

$$ADV(\vec{u}, dt) : \frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} : \text{advection}, \quad (5.5)$$

$$DENS(\vec{u}, dt) : \frac{\partial \rho}{\partial t} = -\vec{u} \cdot \nabla \rho : \text{density advection}, \quad (5.6)$$

$$PROJ(\vec{u}, dt) : \frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho} \nabla p : \text{projection} \quad (5.7)$$

$$\text{where } \nabla \cdot \vec{u} = 0 : \text{incompressibility}. \quad (5.8)$$

Thus the original Navier-Stokes equation is rewritten as:

$$u_{new}^{\vec{}} = PROJ(DENS(ADV(BF(\vec{u}, dt), dt), dt), dt). \quad (5.9)$$

5.2 Domain discretization

Note that the above equations are still a specification, and require a discretization in time and space. To discretize in time, a proper time step dt should be derived. This is described in Section 5.2.1. Then, for the spatial discretization both a grid and particles are used, which is discussed in Sections 5.2.2 and 5.2.3 respectively. The algorithms $BF(\vec{u}, dt)$, $ADV(\vec{u}, dt)$ and $PROJ(\vec{u}, dt)$ are explained in Sections 5.3.1, 5.3.2 and 5.3.3 respectively. The advection of density, computed by $DENS(\vec{u}, dt)$, is derived from the particles, and therefore discussed in Section 5.2.3. The boundary conditions of the simulation are discussed in Section 5.4.

5.2.1 Time step size

To guarantee convergence of the simulation, a necessary condition for the time step is given by the CFL condition, named after Courant, Friedrichs and Lewy. Basically, it describes a relation between the cell dimension dx , the maximal velocity in the system c and the timestep size dt . Equation 5.10 is the general CFL condition. Here α is a user-defined variable, stating the distance, expressed in the number of cells the fluid can flow through in one timestep. By using $\alpha = 1$ the fluid can only move one cell per

time step, and thus the interpolation errors made by advecting the particles are small, therefore $\alpha = 1$ is used. The CFL condition is defined as:

$$dt = \alpha \frac{dx}{c}. \quad (5.10)$$

5.2.2 Space discretization - Grid

The grid in the FLIP method is an auxiliary data structure. No information is stored longer than one time step, i.e., all necessary data is stored in the particles. Every time step, the grid is cleared and refilled with the data from the particles.

The grid data is stored on a so-called staggered marker-and-cell (MAC) grid, first used by Harlow and Welch [46]. On this MAC-grid, the pressure is stored in the center of the cell and the velocity is stored on the cell boundaries, as depicted in Figure 5.1. In the implementation non-cubical cells are used, with dimensions dx , dy and dz . For simplicity, however, here each cell is a cube with edges of length dx . This grid is used to derive the unbiased second-order accurate central difference to calculate the derivative of the velocity in the center of the cell. This is required to calculate the gradient of the velocity at the center point of a cell, needed for the pressure solve. For example, the first-order difference in the x direction of u component of $\vec{u} = (u, v, w)$ at a point $q(i, j, k)$ is

$$\left(\frac{\partial u(q)}{\partial x} \right) = \frac{u(x + 1/2, y, z) - u(x - 1/2, y, z)}{dx}. \quad (5.11)$$

To get the velocity at locations that are not stored within the cell, trilinear interpolation is used on individual components of velocity \vec{u} . A small interpolation error occurs, but only the difference between the particle's velocity and the grid velocity is used, therefore, minimizing the numerical dissipation caused by the interpolation.

5.2.3 Space discretization - Particles

Each particle stores its own velocity. Every time step this velocity is split in its components, and accumulated to the grid cell containing the particle using a one-dimensional kernel function K , which used for weighting in every direction. This weighting is related to the distance of the particle to the grid location, where the velocity should be stored.

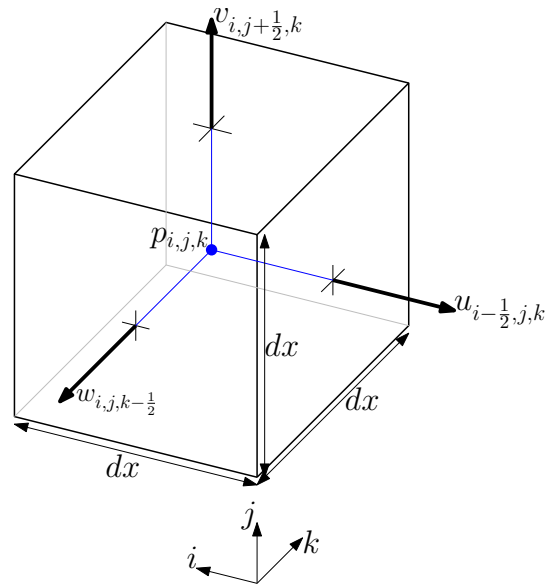


Figure 5.1: A cell from the 3D-MAC-grid

Here, the following linear kernel function is used:

$$K = dx - x. \quad (5.12)$$

The return value of this function is always between $\frac{1}{2}dx$ and dx , where dx is the cell dimension and x is the distance of the particle to the center of the cell. A copy of the grid is made of this grid to be used after the grid is made divergence-free.

When the grid is made divergence-free, the translation from grid to particles uses the difference between the stored velocity on the grid \vec{u}_{old} as stored and the new velocity as stored on the on the divergence-free grid \vec{u}_{new} via interpolation. Then the velocity of the particle is given by $\vec{u}_{new} - \vec{u}_{old}$. By using this difference, a minimal undesired smoothing, due to numerical dissipation, of the particles' velocity occurs, as stated by Batty et al. [34]. Due to this property, no vorticity confinement is needed to get small scale features as the velocity does not fade out at an unnaturally fast pace. As was explained in Chapter 4.1, this occurs in pure Eulerian approaches.

The density advection is implicitly given by the particles. Here it is important that density ρ is defined as $\rho = \frac{m}{V}$, mass divided by volume. Now by assigning a mass to every particle we can set the density. For example, if we initially have 16 particles per cell and want a density of 0.00106 g/mm^3 , the density of blood [47], and the dx of the cell is 2mm the mass in such an initial cell is given by $m = \rho \cdot V = 0.00106 \cdot (2 \cdot 2 \cdot 2) = 0.00848$.

From this the mass of a particle is given by $\frac{0.00848}{16} = 0.00053\text{g}$. When the mass of a particle is known it is straightforward to compute the density of a cell after the particles are advected. Using the example of above we get for a cell with 12 particles a density of $\rho = \frac{m}{V} = \frac{0.00053 \cdot 12}{2 \cdot 2 \cdot 2} = 0.000795 \text{ g/mm}^3$.

5.3 Operator computation

5.3.1 Body forces

The body force algorithm $BF(\vec{u}, dt)$ applies a given force \vec{g} to the whole velocity field. Meaning that for all the velocities on a grid cell $\vec{g} \cdot dt$ should be added to the velocity \vec{u} on the grid. The function $BF(\vec{u}, dt)$ was specified by:

$$\left(\frac{\partial \vec{u}}{\partial t} = \vec{g} \right),$$

by using forward differencing we get:

$$u(\vec{x}, t + dt) = u(\vec{x}, t) + dt \cdot g. \quad (5.13)$$

5.3.2 Advection

The advection algorithm $ADV(\vec{u}, dt)$ transports the particles on the grid storing the velocity \vec{u} . For advection of a particle p , we have to calculate the new location $px_{new}^{\vec{}}$ from its old location $px_{old}^{\vec{}}$. To advect the particles we use the second second order Runge-Kutta method. By using Runge-Kutta 2 the new location of the particle $x_{new}^{\vec{}}$ is defined as:

$$\begin{aligned} x_{mid}^{\vec{}} &= x_{old}^{\vec{}} - \frac{1}{2} dt \cdot \vec{u}(x_{old}^{\vec{}}), \\ x_{new}^{\vec{}} &= x_{old}^{\vec{}} - dt \cdot \vec{u}(x_{mid}^{\vec{}}), \end{aligned} \quad (5.14)$$

where $\vec{u}(\vec{x})$ is the velocity stored on the grid for position \vec{x} . Once every particle is advected, their velocity can be stored on the grid again for the next step. Outside the fluid the velocity field is extended using the method described in Section 5.5. This extension is required so that the fluid can advance into regions in which there is, and

thus the velocity could be 0 there. For this Fast Sweeping method extends the PDE $\nabla \vec{u} \cdot \nabla \phi = 0$ is extended, here ϕ is the distance to the nearest solid stored on the grid, this ϕ is also described in Section 5.5.

5.3.3 Projection

In this section the pressure solver is explained. The pressure is used to make the velocity field divergence-free, and thus incompressible. A different notation is used in order to make a clear distinction between big matrices, vectors with a dimension higher than 3 and scalars needed by the solver, this to distinguish them with the vectors in the simulation. These matrices are represented with bold capital letters, and vector names are represented using lower-case bold letters, while scalars and indices are non-bold letters.

Pressure projection uses the Helmholtz-Hodge decomposition [30], which states that every vector field \vec{u} can be decomposed in a divergence-free vector field \vec{u}_{df} and a curl-free scalar field p . This curl field describes the rotation of the three-dimensional velocity field at given points. Thus,

$$\vec{u} = \vec{u}_{df} + \nabla p. \quad (5.15)$$

Note that by definition $\nabla \cdot \vec{u}_{df} = 0$. This is the case because for a given region, the amount of inflow and outflow for the region must be equal, and therefore the divergence must be zero. By applying the divergence to every term in Equation 5.15 we get:

$$\nabla \cdot \vec{u} = \nabla \cdot \vec{u}_{df} + \nabla^2 p = \nabla^2 p. \quad (5.16)$$

This is a Poisson equation, and by solving it, the pressure p can be obtained. This Poisson equation can be solved for p using the method described in the next section. From this, the divergence-free velocity field \vec{u}_{df} can be computed:

$$\vec{u}_{df} = \vec{u} - \nabla p. \quad (5.17)$$

5.3.4 Preconditioned conjugate gradient algorithm

The Poisson equation presented in the previous section can be solved using the Conjugate Gradient (CG) algorithm, of which a detailed explanation is given in Appendix A. The Poisson equation can be broken down in a linear equation for every grid point [48]. This yields a set of linear equations.

An example, without weighting irregular boundaries, shows why the pressure can be solved by an algorithm that solves linear equations, can be shown using the following function:

$$A \cdot p = f.$$

In this equation, A is the area of the cells boundary being fluid in mm^2 , p is the pressure in Pa and f is the magnitude of force acting in the normal direction of the surface A in $\text{g} \cdot \frac{\text{mm}}{\text{s}}$. By multiplying with time step dt we get

$$A \cdot p \cdot dt = f \cdot dt = m \cdot v.$$

Here m is the mass in g and v is the speed in the normal direction of A in $\frac{\text{mm}}{\text{s}}$. We can get rid of the mass term using $m = \rho \cdot V$ where ρ is the density and V the volume of the cell, assuming that the density is equal to one leaves a division by V , which in our example would be dx in mm. Note that three spatial components, x , y and z , are treated separately, hence the scalar-valued volume. The resulting formula is given by:

$$\frac{A \cdot p \cdot dt}{dx} = \frac{s}{dx}. \quad (5.18)$$

This is a linear equation that relates the pressure to the velocity for every point on the grid. If these equations are solved for p at every grid point, the pressure field is obtained.

These linear equations can be solved by the CG algorithm. This algorithm solves these linear equations by iteratively solving the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, where \mathbf{x} is an unknown vector, \mathbf{A} is a symmetric and positive-definite square matrix, and \mathbf{b} is a given vector. Because \mathbf{A} must be positive-definite the property given by equation 5.19 must hold.

$$y^T A y > 0, \text{ for any non-zero vector } y \quad (5.19)$$

Instead of CG, the Preconditioned Conjugate Gradient (PCG) solver of Bridson was used, similar to the paper by Batty et al. [34]. In this paper a pressure equation is introduced that reduces the kinetic energy, while taking arbitrary solids into account. The solver defined in the paper ensures incompressibility by using the PCG algorithm, which is a faster variant of the conjugate gradient (CG) algorithm.

To solve the pressure with the PCG algorithm, vector **div** is initialized with the divergence, and matrix **A** is used to define the constant relation of the linear equation between neighboring cells. To get the index of a cell from the 3D environment with coordinates (i, j, k) with respect to the one dimensional vector (and two dimensional matrix) the vector index q is calculated using $q = i + n_i \cdot (j + n_j \cdot k)$, where n_i is the number of cells in the i -direction and correspondingly n_j is the number of cells in the j -direction. Note that, this is an unique mapping from every cell to an index in the vector and to row and column indexes of the matrix. This is used to initialize the divergence vector **div**. For every cell with index i, j, k and corresponding mapped index q the divergence div_q is calculated and stored in **div** at index q . In the continuous case divergence is defined as $\nabla \cdot \vec{u}$. Therefore, on a MAC-grid, the discrete divergence is calculated as follows:

$$\begin{aligned}
 -div_q = & \frac{u_w(i - \frac{1}{2}, j, k) \cdot u(i - \frac{1}{2}, j, k) - u_w(i + \frac{1}{2}, j, k) \cdot u(i + \frac{1}{2}, j, k)}{dx} + \\
 & \frac{v_w(i, j - \frac{1}{2}, k) \cdot v(i, j - \frac{1}{2}, k) - v_w(i, j + \frac{1}{2}, k) \cdot v(i, j + \frac{1}{2}, k)}{dx} + \\
 & \frac{w_w(i, j, k - \frac{1}{2}) \cdot w(i, j, k - \frac{1}{2}) - w_w(i, j, k + \frac{1}{2}) \cdot w(i, j, k + \frac{1}{2})}{dx}, \quad (5.20)
 \end{aligned}$$

where both the weights (u_w , v_w and w_w) and the corresponding velocity fields (u , v and w) are defined on a MAC-grid with cell width dx . The weights are computed as in Section 5.4. Note that the **div** vector corresponds with the $\frac{s}{dx}$ term in Equation 5.18.

Filling in the matrix **A** is a bit more complex. Note that this matrix should correspond with the area term, $\frac{A \cdot dt}{dx}$, of Equation 5.18. If the cell fl contains fluid, and a neighboring cell nb contains fluid we add the term $term = nb_w \cdot \frac{dt}{dx^2}$ to the entry (fl_t, fl_t) for every neighboring fluid cell, and we subtract $term$ from the entry (fl_t, nb_t) . If the cell fl contains fluid but the neighbor does not, we add $term$ divided by the fraction of fluid between the neighbor and fl . These subtracted and added values are used to ensure the area of a cell is not taken into account twice, and to enforce the boundary

conditions. Now by using the PCG algorithm the pressure can be found by calculating $\mathbf{A} \cdot \mathbf{pressure} = \mathbf{div}$, where the input is the matrix \mathbf{A} and the vector \mathbf{div} and the resulting vector $\mathbf{pressure}$ is returned. More details on the PCG algorithm can be found in Appendix A. Then according to Equation 5.17, we can make the velocity field divergence-free.

5.4 Boundary conditions

To use irregular boundaries, i.e., boundaries that do not have to be aligned with the grid, the technique by Batty et al. [34] is used. They use a signed distance function (SDF) with distance to solid objects stored on the MAC-grid, which is used to compute the weights needed by the pressure solve explained in Section 5.3.3. This weights ensure that no fluid can flow inside the solids.

The SDF is negative inside the solid and positive outside the solid. On the boundary of the solid the function is 0, as depicted by Figure 5.2. Calculating the signed distance

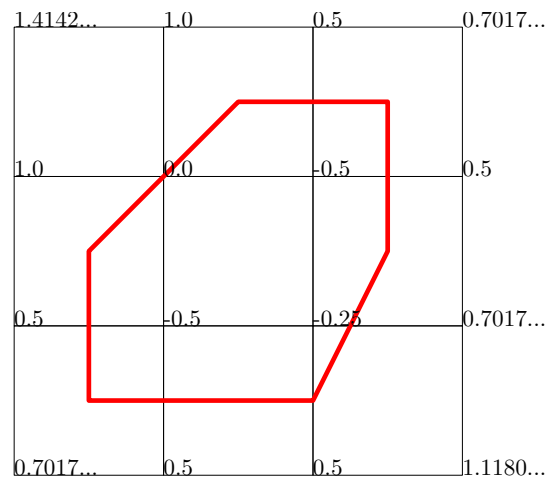


Figure 5.2: The two-dimensional signed distance function of a polygon assuming cell spacing of 1. The given values are the nearest signed distance to the polygon boundary, negative values are inside the polygon and positive values lay outside the polygon.

function of an implicit function, e.g., a sphere is trivial, as this function already defines the distance to the spheres surface. Furthermore the surface has distance value 0 by definition.

For meshes, however, a more complex method is required. Only the points on the boundary of the mesh have a known distance. To assign distance values to other locations,

the Fast Sweeping technique by Zhao is used [49]. This technique extends the known distance by sweeping over the domain multiple times in every direction, a thorough explanation of the sweeping algorithm can be found in Section 5.5.

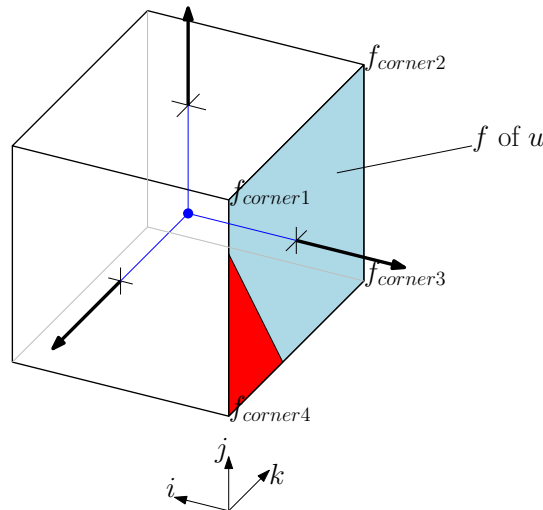


Figure 5.3: An example of the computation of the velocity weight in the u direction. The weight is computed on the blue face, the red area is inside the solid.

The scalar-valued distance to the solid, like the pressure, is stored in the center of the MAC-grid. From this, the fraction inside the solid for every faces f of every MAC-grid cell is computed and stored as a the weight of this face. Note that the area of the complete face is defined as being 1 to get a $0 \leq \text{weight} \leq 1$. Figure 5.3 depicts an example of such a face. Here face in the u direction is shown in blue. The corners, $f_{corner1}$, $f_{corner2}$, $f_{corner3}$ and $f_{corner4}$, are used to compute the area of the face that is inside the solid, shown in red. For example, the weight u_w of the face in the u direction is calculated as follows:

$$u_w = 1 - \text{Area}(f_{corner1}, f_{corner2}, f_{corner3}, f_{corner4}). \quad (5.21)$$

Here the function Area calculates the area of the face that is inside the solid (has a SDF value smaller than 0). Note that the SDF values are stored on the corners of the MAC-grid cells. The area it self is calculated using the Marching Squares algorithm to find the boundaries of the region, a polygon, inside the solid. The area of this polygon is then calculated.

This weight is later on used to alter the velocity around the solid and is therefore used in the projection step, explained in Section 5.3.3. To ensure no fluid flows inside the solid, the velocity in the direction of the normal should be zero (or for moving solids

equal to the solids velocity). This requires a normal at any position of the SDF. This normal \vec{n} can be calculated using the gradient of the SDF at the point q :

$$\vec{n}(q) = \left(\frac{\nabla SDF(q)}{\|\nabla SDF(q)\|} \right). \quad (5.22)$$

Also the velocity of the fluid at the given point is derived and the normal component is calculated. This normal component is then used to set the velocity of the fluid in the direction of the solid to zero (or to the solids velocity).

For particles the same method is used when a particle ends up on a grid location with a negative SDF value. First the location of the collision of the particle with the solid is derived using bisection search on the particles trajectory. The particle is clamped to that position. Then, the normal of the SDF is calculated and is used to remove the particles velocity in the normal direction. Note that the gradient gives the normal pointing away from the solid.

In this project only static solids are taken into account. Therefore, only when initializing the simulation, the SDF and its corresponding weights are computed. When considering moving solids, one has to calculate these every step, which yields a higher computation time per time step. To avoid having to calculate the whole SDF again, one can also store every distance larger than a constant c as being equal to c . Now when a solid moves only the region nearby the surface should be updated, i.e., only the region around the solids boundary with absolute SDF values $< c$. For the weight function this c value can be used to derive whether a solid is inside the face; just check the corners of the face (or line) and check if these are all $\geq c$, if so no solid intersects this face (or line). This, however, requires an appropriate value for c .

5.5 Sweeping

In this section we present the approach to calculate the SDF for solid objects as mentioned in Section 5.4 and the extension of the velocity field as explained in Section 5.3.2. We use the Fast Sweeping for Eikonal equations method by Zhao [49] in 3D. An Eikonal equation is of the form

$$|\nabla q(x)| = F(x), x \in \Omega, \quad (5.23)$$

and it describes the propagation of a function $q(x)$ through the domain Ω . $F(x)$ is given a function with positive values. In case $F = 1$, the signed distance function (SDF) is the solution of the Eikonal equation. In general this method is used to extend a given wave equation to define the wave propagation.

The Fast Sweeping algorithm combines upwind differencing with Gauss-Seidel iterations in all spatial directions. To use the algorithm by Zhao, first the known values should be set to initialize the system. For example, the points in the mesh have a known distance of 0 to the mesh or the velocity of the fluid. Before the first iteration all unknown values are set to the highest distance in the system (for example the width of the total grid) plus one. These values are later updated in an iterative manner. Namely, the algorithm sweeps the grid in all three directions, back and forth, per iteration. Below we assume $F(x) = 1$ to compute the SDF.

For the three dimensional case we want to calculate the new value $q_{i,j,k}$ for the cell with index (i, j, k) . For this, the upwind differencing given by Equation 5.24 is used to discretize Equation 5.23:

$$(q_{i,j,k} - a)^2 + (q_{i,j,k} - b)^2 + (q_{i,j,k} - c)^2 = F(x) = 1. \quad (5.24)$$

Here, the values $a = \min(q(x_{i-1,j,k}), q(x_{i+1,j,k}))$, $b = \min(q(x_{i,j-1,k}), q(x_{i,j+1,k}))$ and $c = \min(q(x_{i,j,k-1}), q(x_{i,j,k+1}))$ are defined. They store the minimal value of x for the neighboring cells. Now, without loss of generality we assume $a \leq b \leq c$. A new value for $q_{i,j,k}$ is only assigned if this new value is smaller than the current value. The new value for $q_{i,j,k}$ is defined as:

$$q_{i,j,k} = \begin{cases} d_1, & \text{if } d_1 \leq b \\ d_2, & \text{if } d_2 \leq c \\ d_3, & \text{otherwise} \end{cases} \quad (5.25)$$

where d_1 , d_2 and d_3 are defined by:

$$\begin{aligned} d_1 &= a + 1, \\ d_2 &= \frac{\sqrt{-a^2 + 2 \cdot a \cdot b - b^2 + 2} + a + b}{2}, \\ d_3 &= \frac{\sqrt{(-2 \cdot a - 2 \cdot b - 2 \cdot c)^2 - 12 \cdot (a^2 + b^2 + c^2 - 1)} + 2 \cdot a + 2 \cdot b + 2 \cdot c}{6}. \end{aligned}$$

Here, the function for d_1 is given by solving $(q_{i,j,k} - a)^2 = 1$, thus $d_1 = q_{i,j,k} = a + 1$ for q , likewise d_2 is given by solving $(q_{i,j,k} - a)^2 + (q_{i,j,k} - b)^2 = 1$ and d_3 by solving $(q_{i,j,k} - a)^2 + (q_{i,j,k} - b)^2 + (q_{i,j,k} - c)^2 = 1$.

$dim + 1$	$dim + 1$	$dim + 1$
2.3	l	$dim + 1$
$dim + 1$	3	$dim + 1$

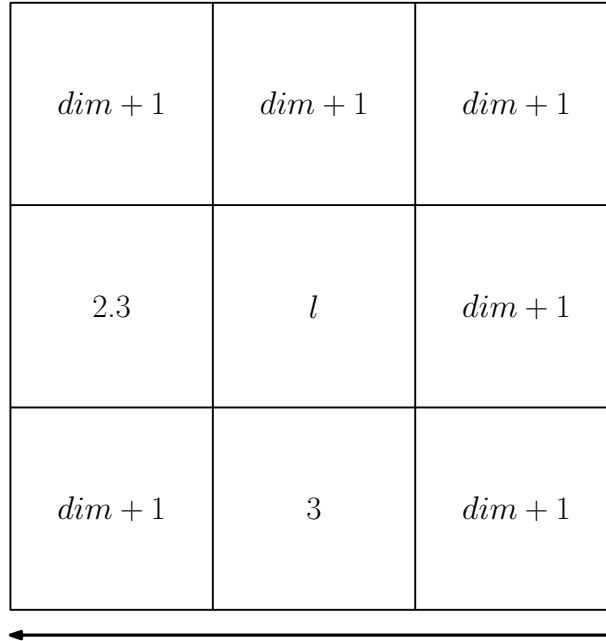


Figure 5.4: An example of 2 dimensional Fast Sweeping. The arrow indicates the sweeping direction. A value of approximately 3.264 for l is found. $dim + 1$ is the largest grid dimension plus one.

Figure 5.4 shows a two dimensional example of Fast Sweeping. The arrow indicates the sweeping direction. Here $a = \min(2.3, dim + 1) = 2.3$ and $b = \min(dim + 1, 3)$. The value for l is now given by computing first $d_1 = a + 1 = 3.3$. This is larger than 3, so also d_2 has to be computed: $d_2 = \frac{\sqrt{-a^2 + 2 \cdot a \cdot b - b^2 + 2 + a + b}}{2} = \frac{\sqrt{-2.3^2 + 2 \cdot (2.3) \cdot (3) - 3^2 + 2 + 2.3 + 3}}{2} \approx 3.264$. No value for c exists in this two dimensional example, and thus $l = 3.264$.

When sweeping for the SDF of the solid objects, only two iterations are used as this is a simple scalar field. To extend the velocity of the fluid in the regions without fluid three iterations per component are done in.

Chapter 6

Measurement-simulation coupling

This project tries to combine the PC-MRI measurements, introduced in Section 2.2, and the simulation introduced in Chapter 5. Both have their own limitations and benefits. The goal of this project is to exploit the advantages of both, with a minimum of disadvantages. The advantages of the PC-MRI measurements are the fact that they provide patient-specific and real world values, while the simulation is a model of the blood-flow in the patient. This model strives to be a physically correct representation of the physiology of the patient.

The cardiac 4D PC-MRI data yields patient-specific data in relatively short period. From this, the velocity field of the flow can be reconstructed retrospectively with a temporal resolution of approximately 50ms with a spatial resolution of about $2\text{ mm} \times 2\text{ mm} \times 2.5\text{ mm}$ [11]. Moreover, measurements allow physicians to do patient specific research as well as diagnosis. For simulations, however, the amount of computation time available determines the amount of detail incorporated in the simulation. A high resolution requires more computations. Another advantage of the simulation is that it yields incompressible velocity fields, while the measurements suffer from divergence due to noise and artifacts.

By combining both methods, we aim to increase the temporal and spatial resolution of the measured data in a physically based manner by using the simulation coupled with the measurements.

In this chapter two methods will be presented that couple the measurements with the simulation. A coupling method should maintain the incompressibility of the fluid simulation, but should ensure the measurements are clearly represented by the simulation. Furthermore, the small scale features of the flow as measured should remain intact after the coupling is done. However, the smallest scale spatial features will not occur in the measurements, but can be introduced by the simulation. The first method we will discuss alters the velocity field of the simulation, while the second uses so-called control forces. These control forces are local forces that steer the simulation towards a measurement. The quality of the described methods will be evaluated in Chapter 8.

6.1 The PC-MRI-measurement-integrated method

A simple control method would be to replace the velocity field of the simulation by the velocity field of the measurement at the point in time. This, however, is likely to result into divergence in the simulation, since the measurements are not guaranteed to be divergence-free, and the artifacts and noise may influence the simulation. Furthermore all previous steps of the simulation are not taken into account. This is where our new method, PC-MRI-measurement-integrated (PCMI) simulation, is used.

PCMI simulation is inspired by ultrasound-measurement-integrated simulation (UMI). The UMI simulation technique is described in the paper by Funamoto et al. [10]. This technique combines Doppler ultrasonography measurements with a fluid simulation. Section 2.1 provides more information about Doppler ultrasonography. UMI uses the difference between the velocity fields of the measurements and the simulation. From this difference, feedback signals are generated to control the simulation. These signals are then applied by using a local force \vec{f}_v on the simulation. This force is given by:

$$\vec{f}_v = -K_v \rho (\vec{V}_c - \vec{V}_s) u_{max} \Delta S, \quad (6.1)$$

where K_v is the feedback gain, which is a control parameter, and u_{max} is the maximum flow speed of the measurements, and ΔS is the control volume of pressure. \vec{V}_c and \vec{V}_s are respectively the projections of the velocity field of the simulation and the velocity field of the measurement in the ultrasonic beam direction. The force is used to direct the simulation towards the measurements.

UMI, however, is geared towards ultrasound, which provides high spatial and temporal resolution. In the PC-MRI case, the temporal resolution is much lower, but a complete velocity field is given, in contrast with Doppler ultrasonography. While the temporal resolution of the measurements can be lower than the resolution of the simulation [50], a relatively high temporal resolution is required for to get the best results for UMI.

In contrast to UMI, for the PCMI method a complete velocity field is available for both the simulation \vec{u}_{sim} and the measurements \vec{u}_{meas} at a certain point in time. Therefore, the difference of the two velocity fields yields another velocity field \vec{u}_{diff} . Unfortunately, compared to Doppler ultrasonography, less measurements in the temporal domain are available. Therefore, no control force is used, since such a force would only be defined when a PC-MRI measurement is available.

Let $\vec{u}_{diff} = \vec{u}_{sim} - \vec{u}_{meas}$ be the difference between the simulation and measurement velocities. By using the Helmholtz-Hodge decomposition we can compute a divergence-free \vec{u}_{new} by computing the pressure p_{diff} of \vec{u}_{diff} :

$$\vec{u}_{diff} = \vec{u}_{new} + \nabla p_{diff},$$

When applying the divergence operator, we get:

$$\nabla \cdot \vec{u}_{diff} = \nabla \cdot \vec{u}_{new} + \nabla \cdot \nabla p_{diff},$$

\vec{u}_{new} is by definition divergence-free and thus equal to 0, so the result is

$$\nabla \cdot \vec{u}_{diff} = \nabla^2 p_{diff}. \quad (6.2)$$

By solving Equation 6.2 a divergence-free velocity field \vec{u}_{new} can be computed. By adding this velocity field to the velocity field of the simulation incompressibility is maintained. Furthermore, the velocity field will be close to the measurements, since \vec{u}_{new} is still close to \vec{u}_{diff} . Advantages of this method are that it is fast, since it requires only one update. A disadvantage is that the amount of control is not directly configurable, as is possible when using forces. In some cases it might be desirable to have the simulation loosely coupled to the measurements. For this method that is not trivial. This is because the divergence-freeness of \vec{u}_{new} must be maintained.

6.2 The Linear Feedback Control method.

The linear feedback control (LFC) method introduced by Kim [9] is a control force method that allows the usage of a target velocity field \vec{u}_{meas} , and thus this method uses a force to steer the simulation towards the measurements. The incompressibility of the simulation is maintained by the solver of the simulation, because the forces are applied before the velocity field is made divergence-free. Convergence is shown by Kim [9] when the control force is applied iteratively for every time step. UMI adds a control force \vec{f}_l to the force term \vec{g} in the Navier-Stokes Equation 5.1 to control the flow of the fluid. The goal is to converge the absolute difference between the simulation velocity field \vec{u}_{sim} and the target velocity field \vec{u}_{meas} to be zero, thus $|\vec{u}_{sim} - \vec{u}_{meas}| \rightarrow 0$. However, if the target field is not divergence-free the method still converges to a positive value, only the converged value of $|\vec{u}_{sim} - \vec{u}_{meas}|$ will not be 0.

The force applied is given by the following linear feedback law for the force \vec{f}_l :

$$\vec{f}_l = \vec{F}_b - \gamma(\vec{u}_{sim} - \vec{u}_{meas}) \text{ with } \gamma > u_{max}, \quad (6.3)$$

where γ is a positive constant parameter with a value larger than the maximum speed u_{max} in the system, and \vec{F}_b is a measure for the body force that is generated by the target field. It represents the force within the target field. The $\gamma(\vec{u}_{sim} - \vec{u}_{meas})$ term is used to parameterize the strength of the force applied. A lower γ value means less control, and thus, γ can be used to tune the measurement-simulation balance. \vec{F}_B is defined by:

$$\vec{F}_b = \frac{\partial \vec{u}_{meas}}{\partial t} + \vec{u}_{meas} \cdot \nabla \vec{u}_{meas} - \nu \nabla^2 \vec{u}_{meas}. \quad (6.4)$$

This equation corresponds with the Navier-Stokes equation, without the incompressibility condition. Therefore, it can be solved using the same methods as used for the fluid simulation.

Due to the lack of enough measurements in the temporal domain, an iterative procedure has to be applied at the time the measurement is available. This is because no target field is known in the intermediate times between measurements; the temporal difference between the measurements is large. This requires that when a measurement is available at time t the algorithm must converge. Therefore, no temporal good approximation differentiation of the the target field exists. Thus we assume the target field is constant

during the iterations. Moreover, in our case the viscosity term is not modeled so $\nu = 0$. In our case Equation 6.4 thus reduces to:

$$\vec{F}_b = \vec{u}_{meas} \cdot \nabla \vec{u}_{meas}. \quad (6.5)$$

Due to the iterative behavior of the procedure, multiple updates of the velocity field of the simulation are done when applying the method to maintain a divergence-free velocity field.

An advantage of this method that it gradually steers the simulation towards the measurements. Another advantage is that the amount of control can be set using γ . A disadvantage is that it requires multiple iterations to converge, which all need a pressure solve for the simulations.

Chapter 7

Implementation

All methods described in the previous chapters allow the implementation of a fluid simulation that is controlled by PC-MRI measurements. This chapter explains the developing details, aiming for an efficient and robust implementation of the presented algorithms. For clarity, some of the details of the fluid simulation were given along with the theory in Chapter 5.

The QFlowExplorer tool by Van Pelt [11] visualizes 4D PC-MRI blood-flow measurements. This tool was extended with a module to combine the measurements with the simulation. Both QFlowExplorer and the module are implemented using the C++ programming language. This module uses the OpenGL library for the visualization. The CLAPACK [51] and the CBLAS [52] libraries are used to improve the performance for vector and matrix operations used in the fluid simulation. The CLAPACK is the C++ version of the Linear Algebra PACKage (LAPACK) and the CBLAS is the C++ version of the Basic Linear Algebra Subprograms (BLAS). Both the CLAPACK and the BLAS are originally written in the Fortran programming language. For the processing of measured blood-flow data and the anatomical meshes, the Visualization Tool Kit (VTK) was used [53]. This toolkit implements functions for 3D computer graphics, image processing and visualization. The toolkit also provides reading and writing functionality for the VTI and VTK file formats that are used to store 3D image data in the XML format. QT was used for the graphical user interface. All programming was done in Visual C++ Studio 2008 Express Edition.

The implementation was running on a system with the 64-bit Windows 7 operating system. An Intel Core i7-3770 Quad-core processor with hyper-threading at 3.40GHz and 8GB of RAM. The video card used was a NVIDIA GeForce GTX 660 with 4GB memory.

7.1 QFlowExplorer

The QFlowExplorer tool is developed by Van Pelt [11]. In this tool, various visualizations of PC-MRI blood-flow measurements were implemented. It has the ability to load PC-MRI measurement data, stored in an application specific the VTI-data format. It also allows the loading of meshes in the VTK-data format. For this project, QFlowExplorer was extended with an additional module. This module implements the fluid simulation described in Chapter 5 and the PC-MRI-measurement-integrated method presented in Chapter 6.1, as well as the linear feedback control method in Chapter 6.2. By using QFlowExplorer, the module has direct access to loaded measurements and meshes.

7.2 The PC-MRI-measurement-integrated method

The particles of the fluid simulation represent a more detailed velocity field. This is because each cell contains multiple particles, which all store their own velocity. The PC-MRI-measurement-integrated method, PCMI, which was described Chapter 6.1, therefore applies the velocity update on the particles of the fluid simulation. For this purpose, another fluid simulation SIM_{new} is initialized with the same dimensions and solids as the current simulation $SIM_{original}$. Then, for every particle in $SIM_{original}$ a new particle is made for SIM_{new} . This new particle has the same location as the original particle. The velocity of the new particle is set to the velocity of the original particle minus the velocity of the measurement at the position of the particle. Now SIM_{new} is made divergence-free, using the tools available for the fluid simulation. Finally, for every particle p_s in $SIM_{original}$ the velocity of the corresponding particle p_n in SIM_{new} is subtracted from the velocity of p_s . Here it is important that the particles are not advected, so the location of the particles of both $SIM_{original}$ and SIM_{old} are equal. Moreover, both have particles that represent a divergence-free velocity field. Thus, by

subtracting the velocity field of SIM_{new} from $SIM_{original}$ yields another divergence free velocity field, which is close to the measurements velocity field.

7.3 The linear feedback control method

To apply the linear feedback control method (LFC) explained in Chapter 6.2, the velocity field of the simulation \vec{u} and the measurement \vec{U} are used. Both \vec{u} and \vec{U} are stored on a MAC-grid to enable second order accuracy on differential computations, as explained in Chapter 5.2.2. The required gradient computations can be easily calculated with a second order accuracy using the MAC-grid. Note that \vec{F}_b only depends on the target velocity field U , and therefore is constant for a specific measurement does not change.

When \vec{F}_b is known, we can start the iterative procedure. Since U is most likely not divergence-free, this procedure has to continue until it converges. Convergence is obtained when the error of the current iteration is equal to the error of the previous iteration. This error measure is $\max |u - U|$, which specifies the distance of u to U .

Every iteration, the linear feedback force defined in Equation 6.3 has to be computed. Which is defined as the previously computed (constant) F_b minus γ times the difference between u and U . This yields a force field \vec{f}_l , which is then applied by adding $\vec{f}_l \cdot dt$ to u , where dt is the time step currently used in the simulation. Note that during the whole procedure dt is constant. After the force is applied, u is made divergence-free using the pressure solver of the simulation, which was explained in Chapter 5.3.3. Finally the error is computed again, and if needed another iteration is executed.

Chapter 8

Qualitative evaluation

This section evaluates the presented measurement-simulation coupling. To do so, three experiments were conducted. In the first experiment the quality of the different coupling methods, given in Chapter 6, is compared. To make sure the expected velocity is known at every moment in time, synthetic data is used. The second experiment tests the noise robustness of the simulation. It shows the ability of the simulation to deal with various amounts of noise, as occurs in the measurements. To quantify this, two simulations are initialized with the same synthetic data. However, the second simulation has noise added to the data. These two simulations are then compared, where the simulation without noise is the expected result. The third experiment compares temporal linear interpolation of velocity fields to the fluid simulation. In this experiment the quality of the interpolation is compared with the use of fluid simulation for interpolation.

For all experiments two error measures are used for the comparison of the synthetic ground truth of an experiment with the found value: the difference in angle and magnitude of the velocity vectors. Assume that \vec{a} is the velocity vector given by the ground truth and \vec{b} is the velocity vector found in the experiment. For the difference in angle the dot product between the two normalized velocity vectors is calculated. This value is then converted to degrees using the arc cosine to represent the error in degrees. So the error in angle err_{angle} is defined by:

$$err_{angle} = \arccos \left(\frac{\vec{a}}{|\vec{a}|} \cdot \frac{\vec{b}}{|\vec{b}|} \right) \quad (8.1)$$

For the magnitude, a relative error is used. The length of found velocity vector \vec{b} is divided by the expected ground truth velocity vector \vec{a} . The result of this division is then subtracted from 1 to get an error of zero if $||\vec{a}|| = ||\vec{b}||$. Subsequently, the result is multiplied with 100 to get an error in terms of percentage. For the magnitude, this relative error err_{magn} is defined by:

$$err_{magn} = \left| 1 - \frac{||\vec{b}||}{||\vec{a}||} \right| \cdot 100. \quad (8.2)$$

Both these metrics linearly correlate the difference between the expected value and the found value.

8.1 Comparison of simulation methods

This section tries to determine the most suitable method from the methods described in Chapter 6. To quantify the error made by the methods, the previously defined error metrics are used. However, actual PC-MRI data do not define a ground truth; there is no ground truth for the description of cardiovascular blood flow, independent of measurement technique. Therefore, to generate an expected ground truth for a flow field, synthetic data is used.

For this evaluation study, a synthetic flow field is generated to mimic the measured data. This way, the ground truth of the data set is known. The synthetic data consists of a parametric flow field, describing a rotational vortex. The velocity (u, v, w) at a position (x, y, z) , where $0 \leq x, y, z \leq 1$, is defined as:

$$\begin{aligned} u &= (10 \cdot time + 1) \cdot 2 \cdot (y - 0.5) \\ v &= (10 \cdot time + 1) \cdot 2 \cdot (x - 0.5) \\ w &= 0, \end{aligned} \quad (8.3)$$

where, *time* is an integer value that increases from 0 to 7. This ensures a time dependent velocity that increases linearly in time, as depicted by Figure 8.1. The actual time between two outputs is 40ms, the velocities ranges from -72cm/s to 72cm/s. A boundary mesh, a cylinder, that matches the vortex is also generated. Now the velocities at a position are known throughout time, which is used as a ground truth for comparing the

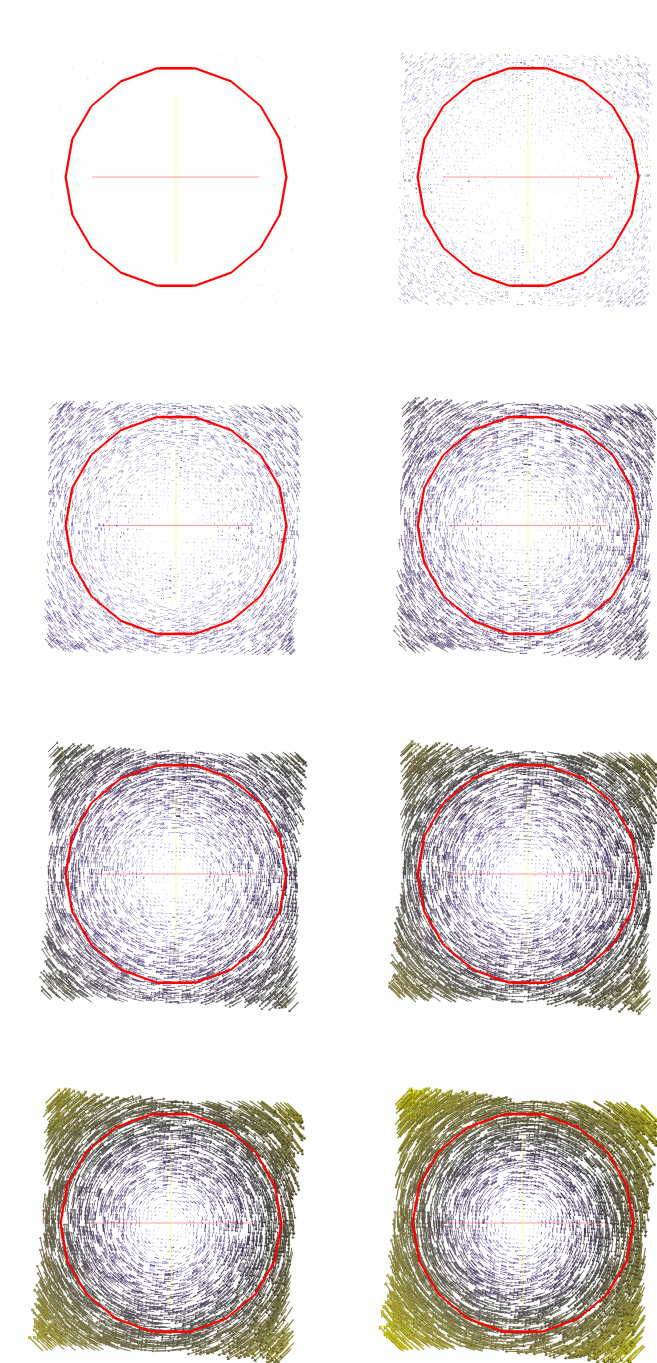


Figure 8.1: The synthetic data, a vortex with linearly increasing velocity. Blue indicates a low velocity and yellow a high velocity. The red circle indicates the cross section of the cylindrical mesh.

velocities produced by the simulation. This allows for a validation of the coupling, and a comparison between the approaches suggested in this thesis.

It is important to note that the simulation cannot mimic the synthetic data, due to fact that the velocity in the synthetic data increases over time. The simulation, however, dampens out over time. Therefore, the error, when no method is applied, will increase. However, if a coupling method is applied, the error should be as low as possible.

Four different methods are tested:

- No method: only the first measurement is used to initialize the simulation. It represents the lack of control.
- Replacement with measurement: replaces the velocity of every particle with the velocity of the measurement at the position of the particle. This is the best case, since it uses the measurement perfectly. However the result are not guaranteed divergence-free, and are hence physically incorrect.
- Linear Feedback Control (LFC) method: the method by Kim [9] explained in Section 6.2.
- PC-MRI-measurement-integrated method: our newly introduced method, explained in Section 6.1.

Note that, the “no method” and “replacement with measurement” are respectively a worst-case and best-case scenario, and are given for comparison. The “no method” case uses no coupling between measurements and simulation. Therefore, no coupling technique should get a comparable error. The ideal method would yield the same error as the replacement with measurements method, while maintaining an incompressible fluid, which is not possible, since the replaced velocity field is not incompressible.

Figures 8.2 and 8.3 show respectively the error in magnitude and angle of the chosen methods. The average error is represented by lines, the standard deviation is represented by the shades in the same color. Note that the time label represents the time of the synthetic data. The methods are applied where t has an integer value, as represented by the vertical lines. The error is measured every 0.10 time steps. For the LFC method, the parameter γ is set to $200 \cdot \vec{u}_{max}$, where \vec{u}_{max} is the maximum velocity in the system. How this γ value was determined is explained in Section 8.1.1.

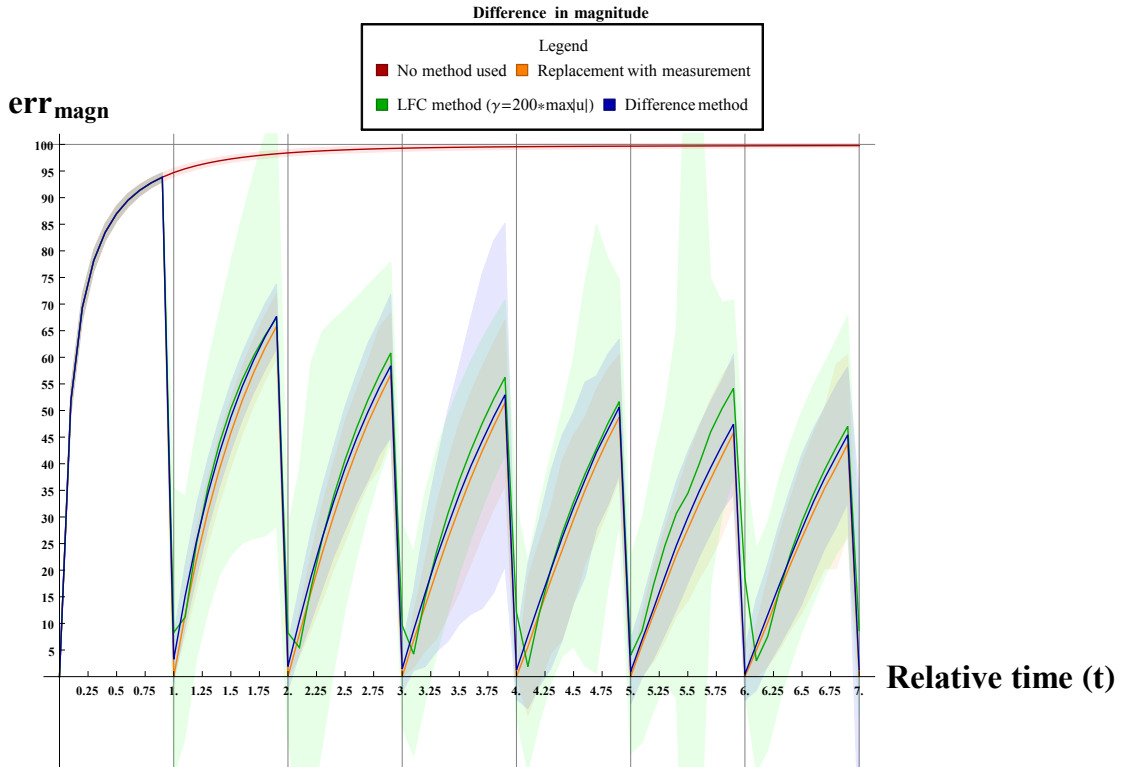


Figure 8.2: The error of the magnitude of velocity for different methods. The lines show the average error, the shades represent the standard deviation. Note that $\max|\vec{u}|$ is the maximum velocity in the system.

As expected, in both figures all methods have a comparable error in the first time steps until the simulation methods are applied at time $t = 1$. All methods are initialized with the same velocity field. The no method line represents the error when the simulation is not affected by the measurements, and therefore the error increases. At time $t = 1$, the methods are applied and the differences become clear. Note that between the vertical lines no method is applied and the error increases.

For the error in magnitude, both the LFC method and PCMI method yield a velocity magnitude comparable to the replacement of velocity. That is, an error of only a few percent is found.

For the error in angle, however, the LFC method has a high error compared to the PCMI method. This error can be explained by the fact that the LFC method is a method that tries to control the shape and flow of the fluid. The method is intended to specify the shape and general direction of the fluid, while locally the direction might differ from the target field. The PCMI method has an error in angle close to zero when applied.

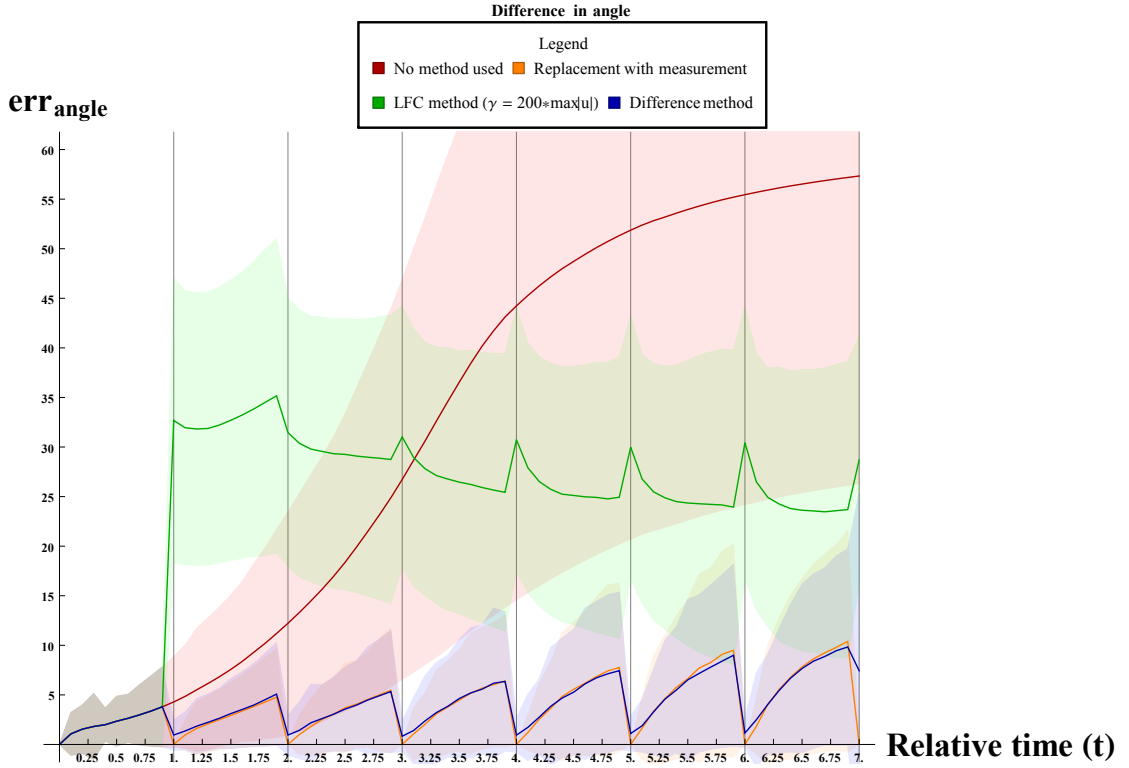


Figure 8.3: The error of the angle of velocity for different methods. The lines show the average error, the shades represent the standard deviation. Note that $\max|\vec{u}|$ is the maximum velocity in the system.

Both the LFC method and the PCMI maintain a divergence-free velocity field. However, while the magnitude is comparable to the replacement of velocity for the LFC method, the angle might differ a lot. In contrast, the PCMI method behaves similar to the replacement of the velocities with respect to both magnitude and angle. In general the PCMI method seems to perform best by maintaining the fluid divergence-free and having a small error with respect to the measurements when available. Note that, in the actual measurements the velocity has a less steep increase in velocity between two measurements. Therefore, the error in between two measurements will be smaller in practice.

8.1.1 Sensitivity of parameter γ for the Linear Feedback Control method

To make a fair comparison, the best performing γ value for the LFC method should be found. The parameter γ is used to set the amount of control for LFC. As stated by Kim [9], γ should be larger than the maximum velocity in the system, therefore the maximum velocity \vec{u}_{max} is calculated. This maximum velocity was then used to define

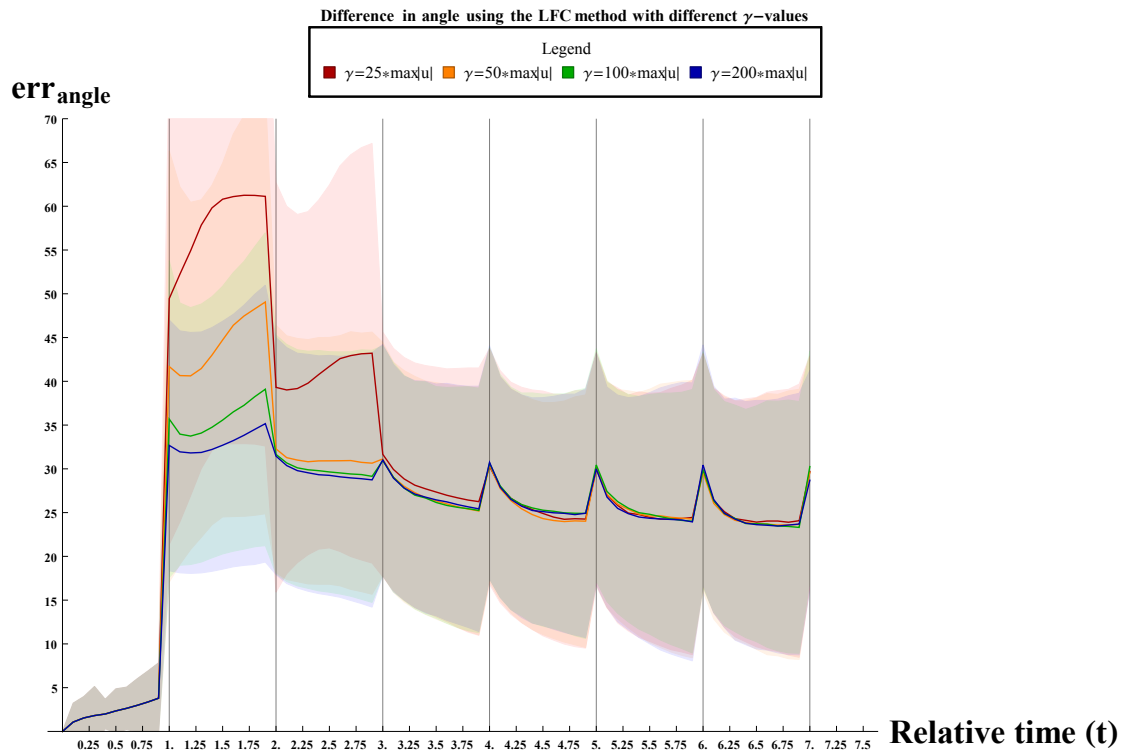


Figure 8.4: The error of the angle of velocity for different γ values for the Linear Feedback Control method.

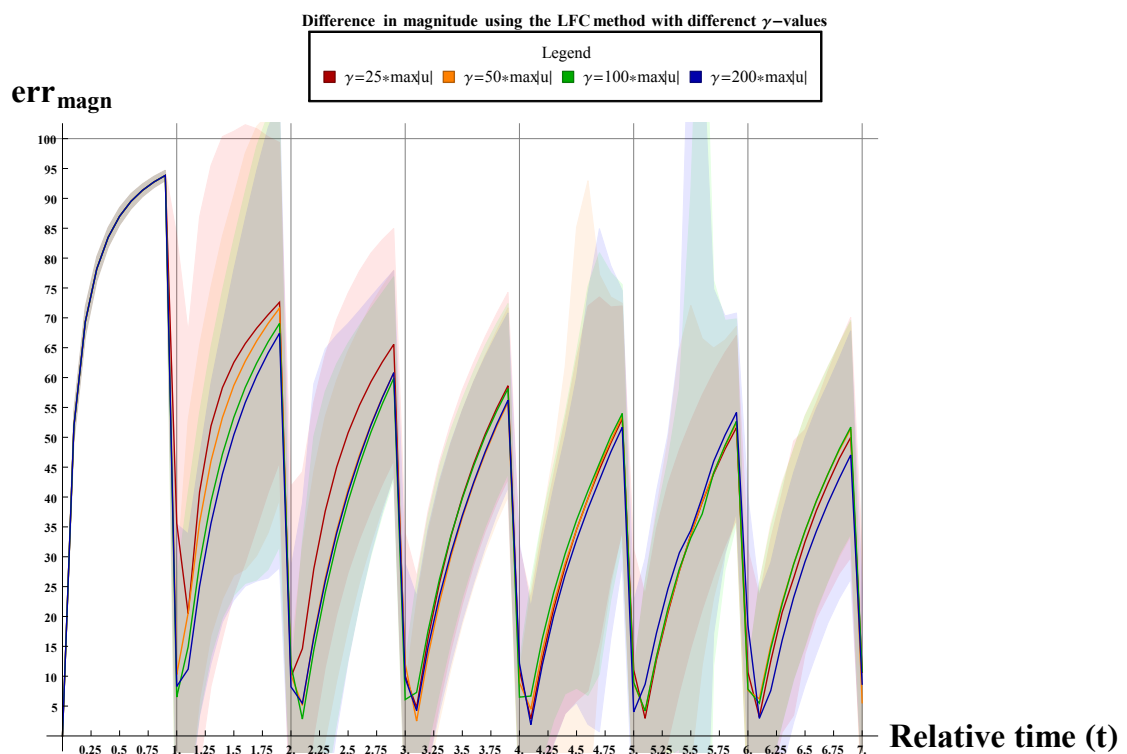


Figure 8.5: The error of the magnitude of velocity for different γ values for the Linear Feedback Control method.

four γ values: $\gamma \leftarrow 25 \cdot \vec{u}_{max}$, $\gamma \leftarrow 50 \cdot \vec{u}_{max}$, $\gamma \leftarrow 100 \cdot \vec{u}_{max}$ and $\gamma \leftarrow 200 \cdot \vec{u}_{max}$. For higher γ values the method produced too strong forces to stabilize. The four selected values for γ fall within a range of \vec{u}_{max} to $200 \cdot \vec{u}_{max}$. Note that, γ only depends on the maximum velocity, therefore the same value can be used once the best value is found.

Figures 8.4 and 8.5 show respectively the error in angle and magnitude after applying the LFC method. The errors for the magnitude are comparable. For the angle however decreases by using a large value for γ , which justifies the use of $\gamma = 200 \cdot \vec{u}_{max}$ for this experiment.

8.2 Robustness

PC-MRI is subject to so-called Rician noise [54]. This noise influences the measurements, and results in uncertainty of the measured values. The simulation, when given this noisy data, should produce useful results, despite the introduced noise. In this section, the robustness of the simulation to noise is tested.

The Rician noise of the PC-MRI can be approximated with Gaussian noise [54, 55], given that the signal-to noise-ratio (SNR) is higher than two. For the experiment, different SNRs are used. The SNR is defined as $SNR = \frac{\text{signal power}}{\text{noise power}}$, where the signal power is given by the average velocity and the noise power can be set using the variance of the normal distribution [14].

Using PC-MRI, the SNR for in vitro measurements is around 28 using contrast agent [22]. For in vivo measurements, depending on the scanner and the scanned region, the SNR lies between 21 and 56, as indicated by Bammer et al. [14].

To obtain Gaussian noise, a random value from a normal distribution is drawn. These numbers are generated using the Box-Muller algorithm [56], which generates $X \sim N(\mu, \sigma^2)$, with mean $\mu = 0$ and standard deviation $\sigma = 1$, where X is a value drawn from the normal distribution. To generate noise with a certain SNR, the resulting value should be multiplied with \vec{u}_{avg}/SNR to modify the variance, σ^2 . Here \vec{u}_{avg} is the average of the velocities in the system. Note that the mean is not altered by the multiplication, since it was defined to be zero.

For this experiment, two simulations are running in parallel. Both simulations are initialized with the previously defined synthetic data. From this data, the set with the highest velocity, $-72\text{cm/s} \leq \vec{u} \leq 72\text{cm/s}$, is selected to initialize the simulations. One of these simulations, however, has additive Gaussian noise applied to the initial velocity of the particle. Note that, the particles are unrelated when initialized, and thus the additional noise is unrelated. The simulations are run and compared until time $t = 1$, since after this, another noisy input measurement will arrive in practice.

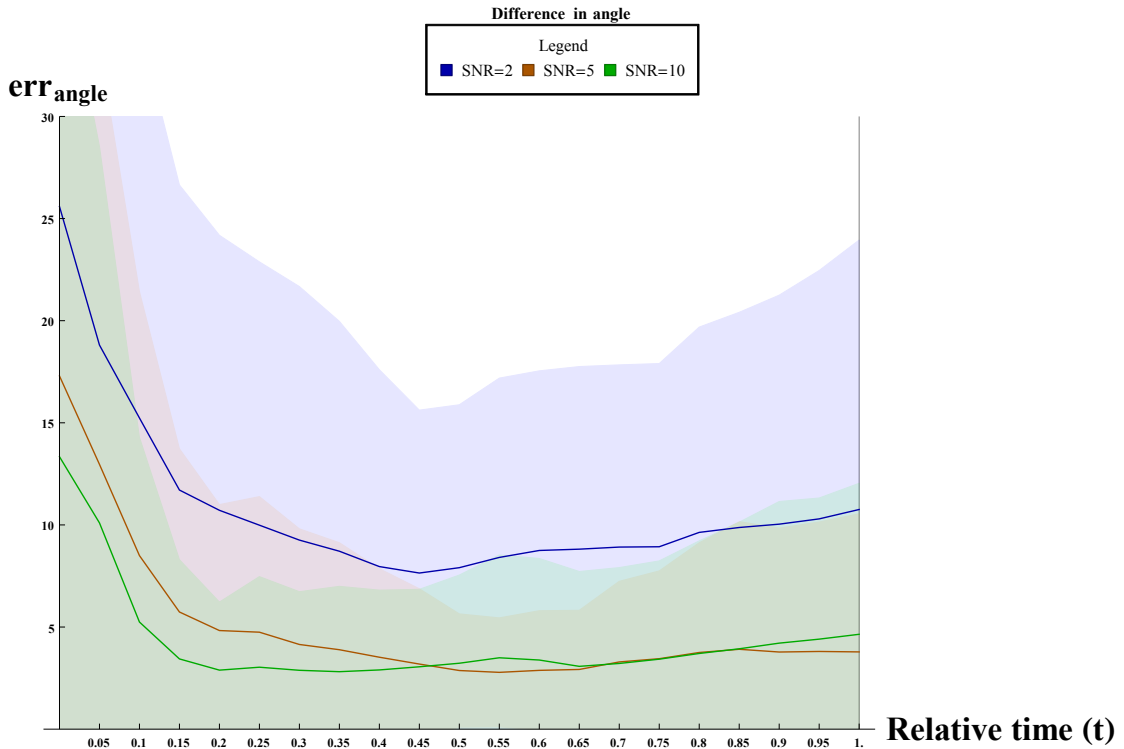


Figure 8.6: The error of the angle of velocity using different amounts of initial noise. The lines show the average error. The corresponding coloring represents the standard deviation.

Figures 8.6 and 8.7 show respectively the error in angle and magnitude of the velocities with different SNRs, namely 2, 5 and 10. These values clearly induce more noise than normally in PC-MRI measurements. A time step of 0.05 time units is used. As can be seen in both figures, the influence of noise is reduced. However, when more initial noise is used, the error remains higher compared to less initial noise.

The lowering of the error due to noise is implicitly given by the translation of the particles with a noisy velocity on the grid, causing smoothing of the data. This smoothing occurs, since the noise has a mean value of 0. By taking the averages of the noisy velocity of multiple particles over one grid cell, the overall noise of the grid cell is closer to the mean. Also, the noise is reduced by the solver, which removes the divergence left by the

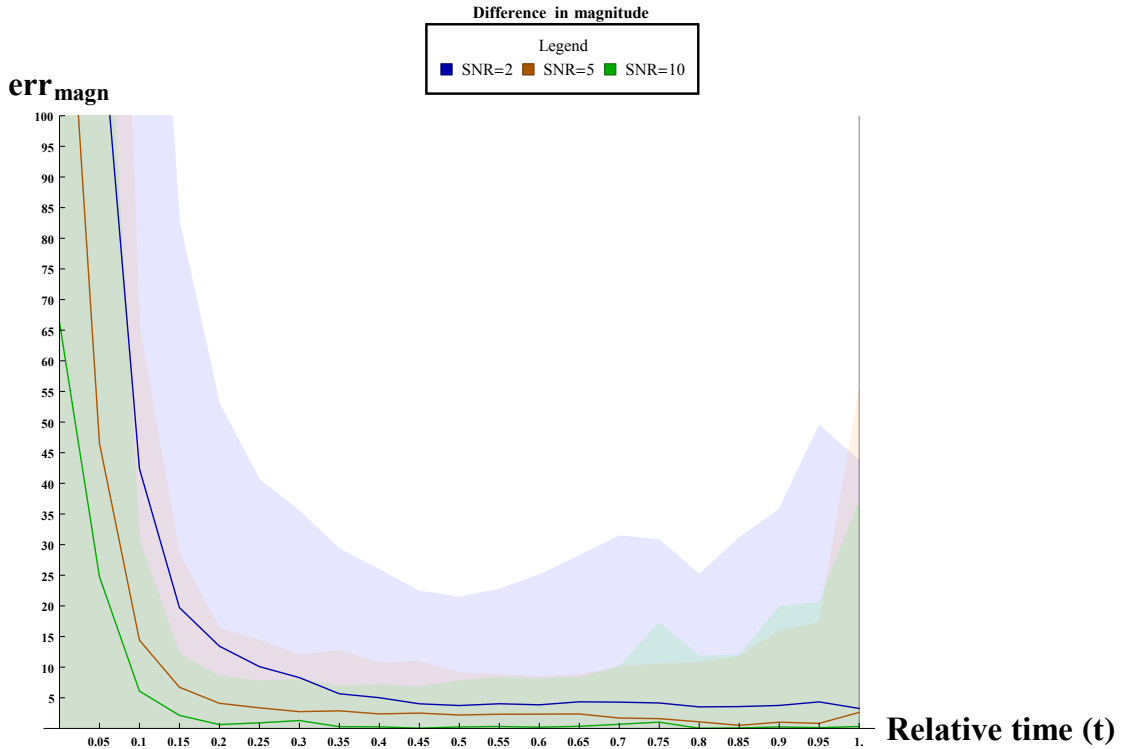


Figure 8.7: The error of the magnitude of velocity using different amounts of initial noise. The line show the average error. The corresponding coloring represents the standard deviation.

noise. The error due to noise for the angle is a lot higher compared to the error of the magnitude. For the angle, the three components of the velocity vector are important, while for the magnitude the three components are averaged again, and hence smoothing the noise.

8.3 Simulation compared to interpolation

Currently, spatial-temporal linear interpolation of the measurements is used. In this section the benefits of using simulation are compared to normal linear interpolation. For this purpose, another experiment was conducted.

In this experiment the fluid simulation is initialized using the high velocity synthetic data described above. The simulation is started, and every 0.10 time units the velocity field is exported until the time $t = 1$. At this point, the velocity at time $t = 0$ and $t = 1$ is interpolated and the results are compared with the stored intermediate velocity fields. The results are shown in Figures 8.8 and 8.9. As shown, the linear interpolation has a big error of both the angle and the magnitude when used at high velocities, right

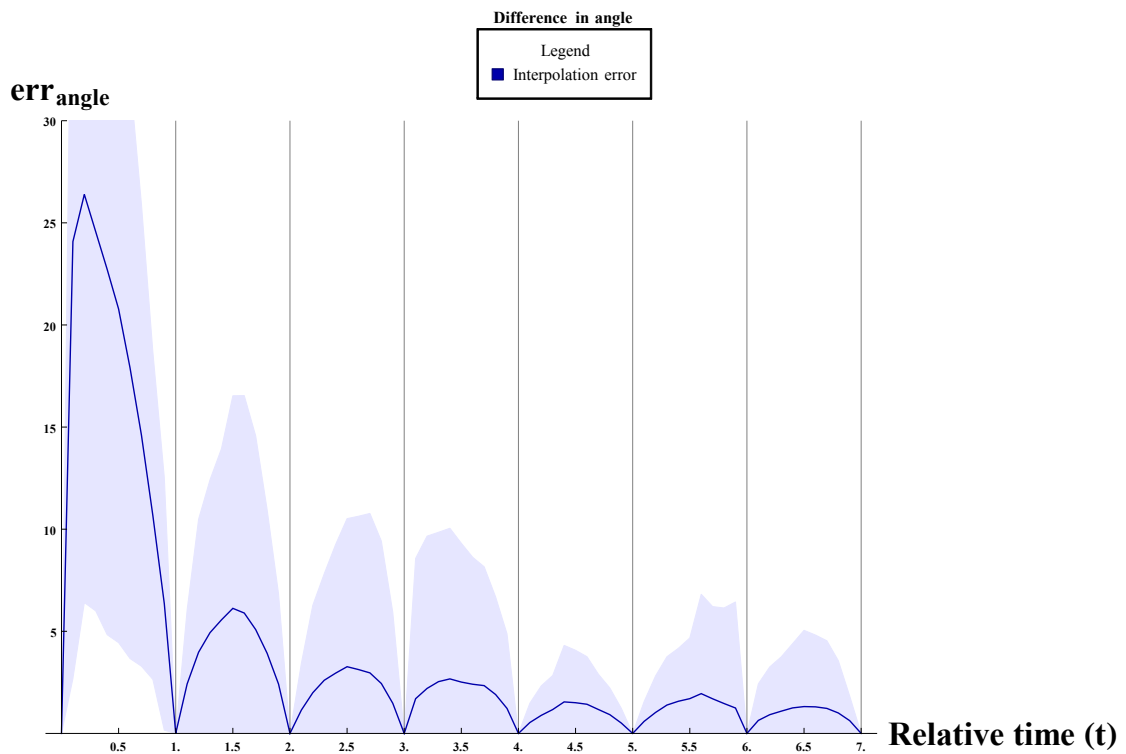


Figure 8.8: The error of the angle made by interpolation of simulated data, the coloring represents the standard deviation.

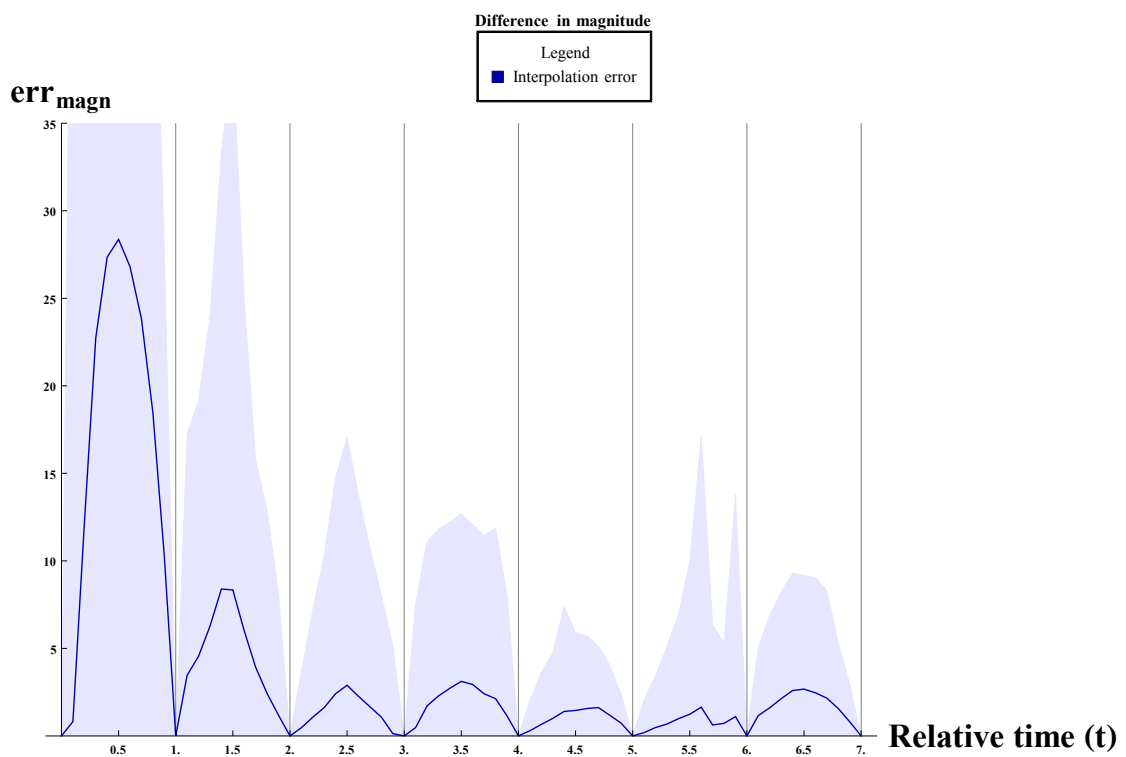


Figure 8.9: The error of the magnitude made by interpolation of simulated data, the coloring represents the standard deviation.

after the initialization. After this the simulation dampens out, and the overall velocity decreases, as well as the relative errors. However, the error seems to stabilize for both the angle and magnitude. This is as expected, because when the vortex barely moves the difference in angle and magnitude will be minimal. Overall these results show that interpolation of high velocity flow fields yields large errors.

Chapter 9

Results on PC-MRI data

Since no ground truth is known for the PC-MRI data, no quantitative comparison can be made for the PC-MRI blood-flow data. Therefore, we rely on a visual inspection of our data. As described in Section 8.1, the PCMI method yields a simulated velocity closest to the measurements, and maintains a divergence-free velocity field. Therefore, this method is used.

When applying the PCMI method on actual data, no unexpected sudden increases of the velocity of the simulation are visible. The blood flow looks realistic and plausible. Furthermore, by using the simulation the artifacts near the vessel wall due to respiration, movement of the patient and the averaging over multiple heartbeats are reduced. This reduction of artifacts is shown in Figure 9.1. Here, yellow is used to draw the velocity vector of the particles of the simulation and purple represents the measured velocity.

When using PC-MRI measurements, no exact boundary is known. The anatomical mesh is an approximation of the vessel boundary at maximum dilation, derived from the measurements. Therefore, fluid can be simulated near the mesh boundary, while the actual vessel boundary in the patient might be at a different location. Thus, the actual vessel might be more narrow than it appears to be compared to the mesh boundary. The flow in these regions, however, is more natural than given by the velocity measured at these locations.

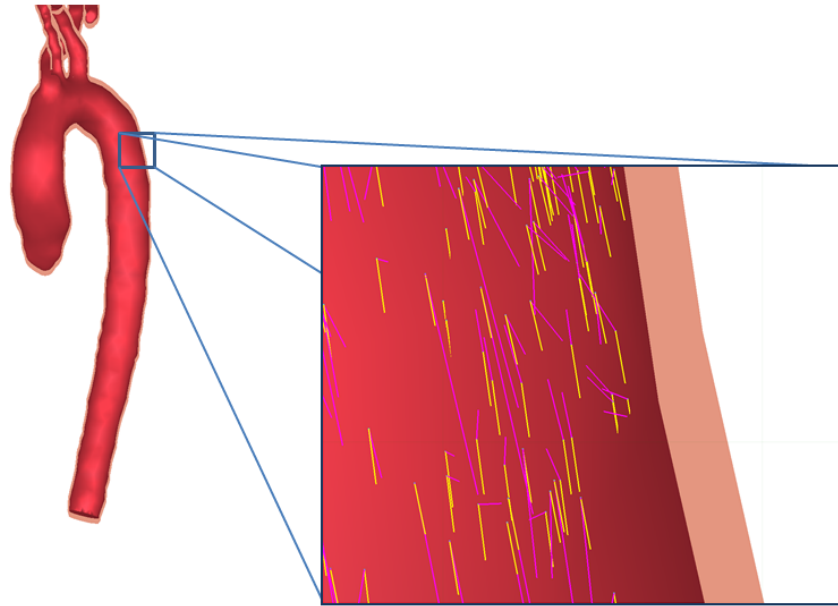


Figure 9.1: The artifacts of the measurements is mainly present at the boundaries. Yellow represents the simulations velocity of the particles while purple shows the velocity of the measurement as interpolated.

In the next sections, a visual inspection of the simulation coupled with the measurements is given for both a healthy volunteer, as well as a patient suffering from an aortic dissection. In these sections, we show that the typical expected blood-flow characteristics are maintained.

9.1 Volunteer data

The images in Figure 9.2 show the simulation just before peak systole, and at peak systole, respectively. The simulation is running in real-time on a coarse grid of $12 \times 34 \times 12$, and uses the PCMI method described in Section 6.1 for the coupling. Note that the grid of the simulation is much sparser than the measured grid, which is $51 \times 144 \times 50$. In the images, the white lines indicate the particle (blue) trajectories over time.

As can be seen in the second image, the right-handed helical flow pattern in the aortic arch, described by Kilner et al. [15], is preserved even on this sparse grid. That is, the

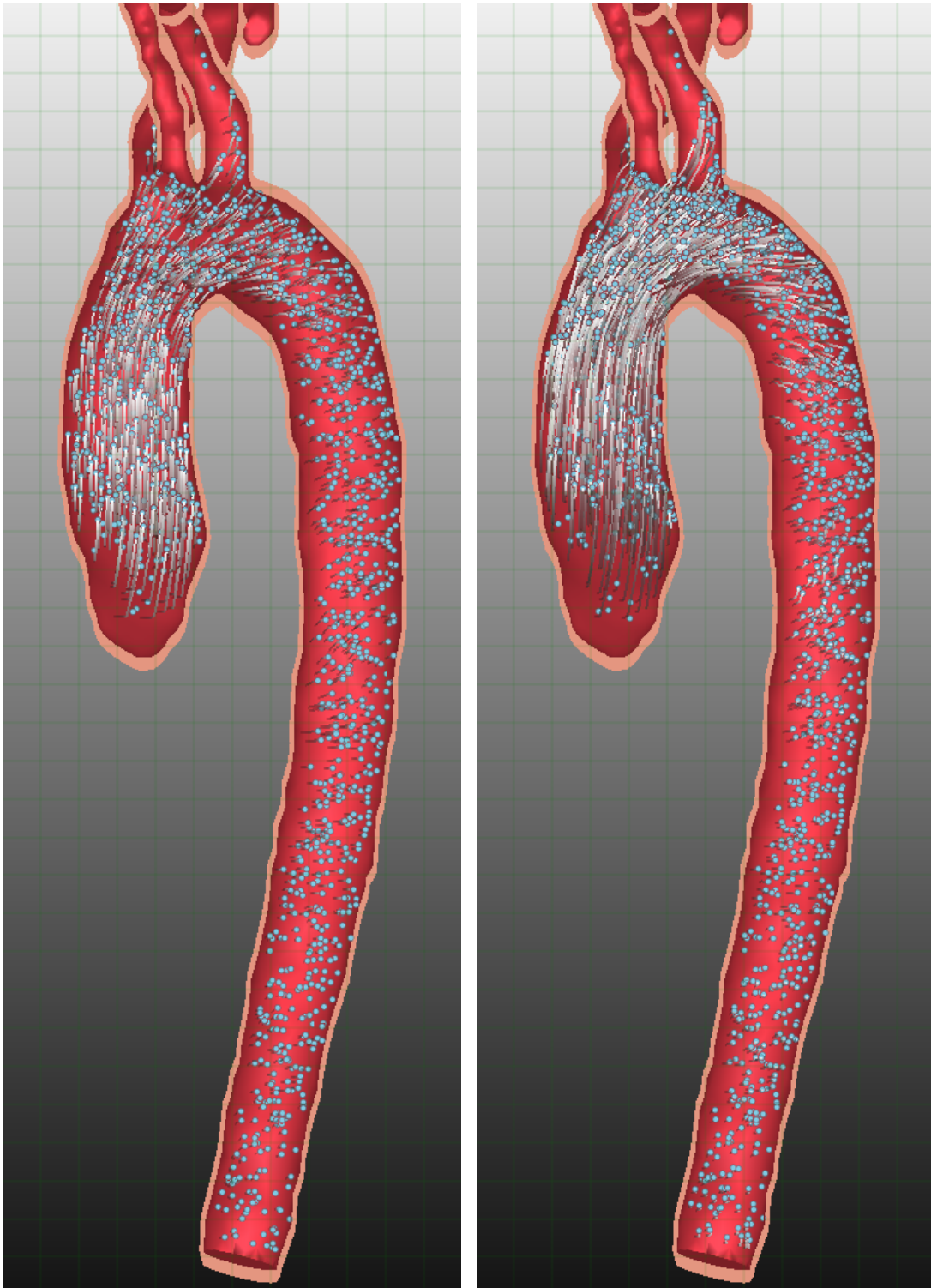


Figure 9.2: The simulation applied on the measurements using the PCMI method, just before peak-systole (left) and at peak-systole (right).

particles shown have a helical movement in the curvature of the arch. This helical flow is typical for healthy subjects at peak systole.

9.2 Patient data: aortic dissection

When a tear in the inner wall of the aorta causes blood to flow between the layers of the wall, we speak of an aortic dissection. This blood between the layers will form a so-called false lumen, which can also be seen in the image data. An aortic dissection normally occurs in a weakened area of the aortic wall. It can be caused by chronic high blood pressure that may stress the aortic tissue, which then has a higher risk of tearing. An aortic dissection leads to a malformation of the aorta, and causes aberrant blood flow. Furthermore, the blood pressure might increase due to narrowing of the aorta.

Figure 9.3 shows that, the existence of vorticity and jets in the blood flow is maintained by the simulation with a $26 \times 47 \times 20$ grid size. By jets is meant high velocity flow. In the figure, only the velocity of the particles is shown for clarity. This vorticity corresponds with the results found by Bogren et al. [57].

By using the simulation, also a relative pressure field within the flow was calculated. Physicians are interested in this pressure, because high pressure regions have a higher risk of tissue damage. Figure 9.4 shows relative pressure maps. As expected, the pressure is high where jets are generated, due to the narrowing of the aorta. Moreover, a high pressure is expected behind the aortic valve, because of the pressure that results from the high velocity, caused by the high velocity of the blood pumped out of the heart.

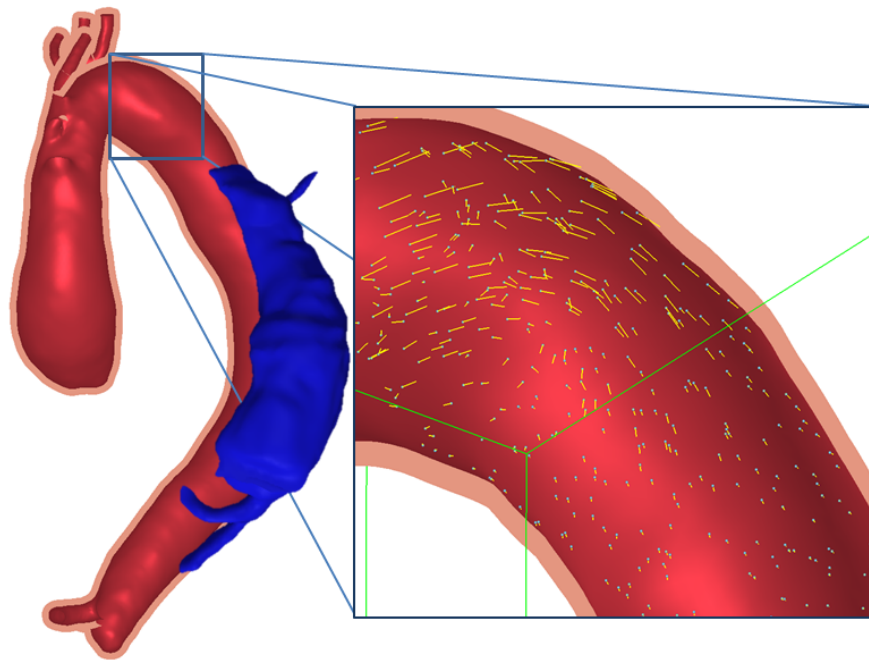


Figure 9.3: Vorticity in the aorta caused by a dissection. The zoom-in shows the vortex in the aortic arch due to the narrowing before the arch. In the image, the false lumen is shown in blue. The yellow lines indicate the velocity of the shown particles, the length indicates the speed.

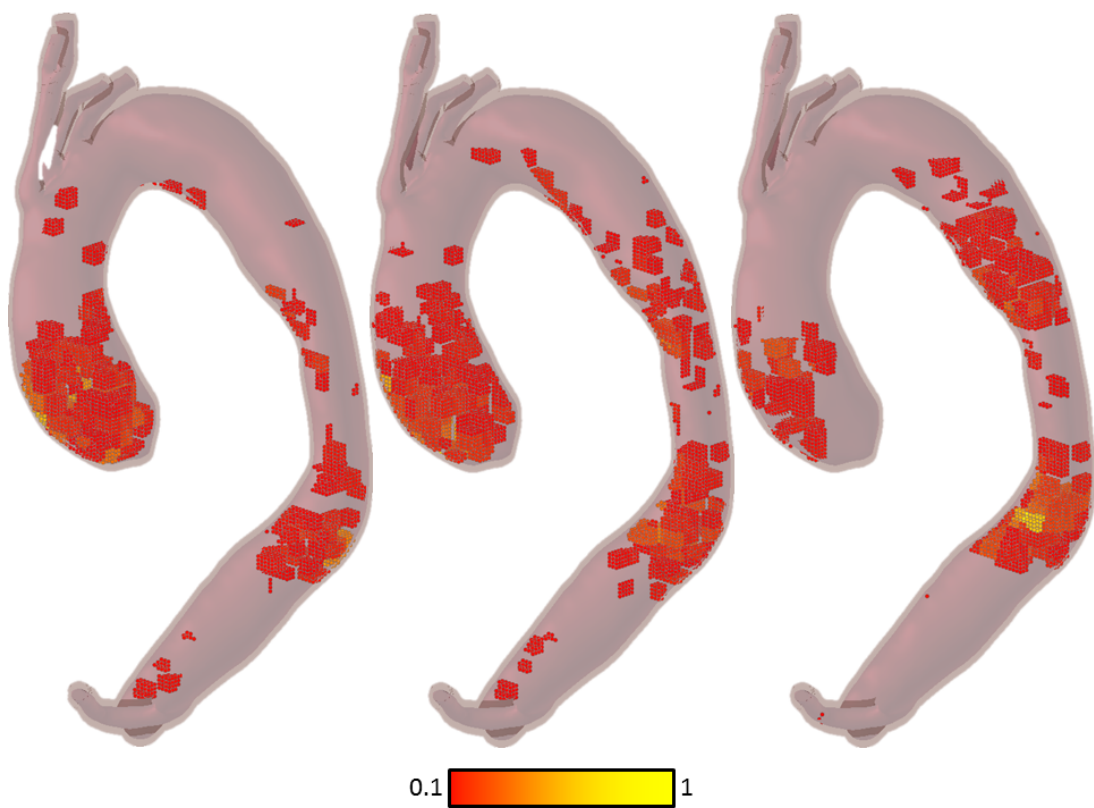


Figure 9.4: Relative pressure maps of a dissection. The pressure is normalized and shown by a dot per location, yellow is used for the highest relative pressure, while red relates to low relative pressure. The mesh is shown as transparent, yet the false lumen is not rendered. For clarity, relative pressures below 0.1 are not rendered.

Chapter 10

Discussion & conclusion

We have presented a new control method to combine time-resolved three-dimensional PC-MRI measurements with a fluid simulation from the computer graphics field. These measurements are defined as a three-directional velocity field. The measured velocity fields are used to be combined with the velocity field from the simulation. The goal of this study was to temporally interpolate between measurements, based on the physics of fluid mechanics, with limited computation time. That is, the resulting velocity field should be incompressible and close to the measured velocity field, even if the measured velocity field is not necessarily divergence-free. Moreover, limited computations reduce processing time and allow incorporated interactively.

A linear feedback control technique and a novel measurement-integrated technique are compared. The linear feedback control (LFC) technique is an iterative process. Every step a control force field is determined by a linear feedback law, based on the difference between the simulated velocity field and the measured velocity field. The control force is then applied to direct the fluid simulation to the measured velocity field. The measurement-integrated technique also determines the difference between the simulated velocity field and the measured velocity field. In contrast to the LFC method, this difference field is then made incompressible, and is added to the simulated velocity field. Both methods are fast and an easy extension given an implementation of a fluid simulation. Furthermore, they can be used for any numerical fluid simulation.

In this thesis, we have evaluated the mentioned techniques for measurement simulation coupling. To evaluate these techniques, the difference in angle and magnitude of the

vectors of the velocity fields of the simulation and the measurement were used as error measures. The comparison revealed that the measurement-integrated method yields a simulated velocity field that is more similar to the measured velocity field, compared to the linear feedback control method. Especially the direction of the vectors of the velocity field have a higher error using the linear feedback control method.

Furthermore, we performed a robustness evaluation. This evaluation compares two simulated velocity fields. Both simulations were initialized with the same data. However, noise is added to the input of one of the simulations. By comparing the two resulting velocity fields for different amounts of noise, the robustness to noise is demonstrated. The simulation is robust to the level of noise typically encountered in PC-MRI measurements.

This study also compared conventional linear interpolation and Runge-Kutta 4 approaches with the interpolation carried-out by our simulation. From this experiment a significant difference between the current interpolation and the simulated interpolation was found. This demonstrates a significant difference between interpolation and simulation exist. It is important to keep in mind that, the simulation is physically underpinned.

Finally, a visual inspection of the measurement-integrated technique applied to PC-MRI measurements was conducted. It was shown that typical blood-flow characteristics are maintained, also in pathological flow. Furthermore, artifacts near the vessel wall are suppressed by using the simulation, and thus have less influence on the blood flow.

Therefore, we conclude that the presented measurement-integrated method, PCMI, provides a significant contribution for combining measurements and simulation techniques. The method enforces a divergence-free flow field and deals with PC-MRI noise, and is likely to perform better than conventional interpolation schemes.

To the best of our knowledge, we are the first to combine full field velocity measurements with fluid simulation instead of defining only the in- and outflow conditions. Therefore, we think it is plausible, that our method is more in correspondence with patient's blood flow than conventional measurements coupling techniques, as PCMI is physically underpinned.

Chapter 11

Future work

For future work, an interesting problem is to handle the non-static boundaries of the aorta. It will be challenging, since the patient-specific time-resolved vessel boundaries cannot be derived directly from PC-MRI measurements. The solution for this problem may be found via two ways; either by MRI acquisition of the anatomical structure, which is expensive and time consuming, or using post-processing by modeling an fluid-penetrable boundary. By using the potential field by Hong et al. [19] it is possible to define such a less strict boundary, compared to a mesh. This potential field can be computed by using the available mesh, such that the regions inside the mesh have a low potential. This way, the fluid will try to flow in the regions with low potential. Furthermore, at the boundaries of the vessel, the strength of the potential can be varied to mimic the wall movement. The attainability of this should be investigated.

Treatment planning and prognosis are also interesting prospectives. For example, by altering the mesh, it is possible to determine variations in blood flow and pressure. This can be used by physicians to experiment with approaches before actually applying them on the patient. Furthermore, it can be used to predict malfunctions in the blood flow and cardiovascular structure. You could, for example, predict beforehand in what way the blood flow will respond to a stent, or what will be the result of a fenestration in a vessel wall.

Due to the increase of information derived from the combination of measurements and

simulation, previously unsuitable visualization techniques can become useful. For example visualizations of the influence of pressure over time on the vessel wall, such visualizations could help to predict vessel damage within a patient. This can help physicians to obtain a better understanding of the blood flow, as well as early diagnosis of CVDs.

More practically, the performance of the simulation can be increased. Due to the fact that most of the cells are non-fluid cells, the performance of the solver of the fluid simulation has a lot of overhead. By using a more efficient data structure, such as a k-d tree, the amount of these empty cells can be reduced. However, the computations become more complex. Still the amount of computations will be reduced enough to reduce the overall computation time. Another improvement of the performance can be made when all matrix operations, needed for the pressure solve, can be done on the Graphics Processing Unit (GPU). This, in the ideal case, can reduce $O(n)$ Central Processing Unit (CPU) operations to $O(1)$ where n is the number of cells in the matrix. This, however, requires zero overhead and n processing units on the GPU. Still, the parallel behavior of the GPU can be exploited to reduce overall computation time. When combined with a sparse matrix library for the GPU, such as Cusp, cuBLAS or cuSparse for CUDA, this ideal case can be approximated. CUDA is the Compute Unified Device Architecture that allows the programmer to use the GPU.

Our method is, to the best of our knowledge, the first to combine full field velocity measurements with fluid simulation. Several aspects for future work, as described in the chapter, seem promising to implement, as they could be of great assistance to physicians. Therefore, future research is needed to determine the attainability of these promising extensions.

Appendix A

Algorithms

The Preconditioned Conjugate Gradient (PCG) algorithm is used to solve $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ for the vector \mathbf{x} , given a positive definite symmetric matrix \mathbf{A} and a vector \mathbf{b} . Preconditioning is used to decrease the number of iterations needed to solve the system. To generate a suitable preconditioner the Modified Incomplete Cholesky (MIC) factorization algorithm is used. In this appendix all matrices and vectors have a dimension greater than 3, and therefore they are shown in bold face for clarity.

The technical report by Shewchuk [58] gives a detailed and intuitive explanation of the (preconditioned) conjugate gradient method. Basically, we want to solve a set of linear equations such that $\mathbf{A} \cdot \mathbf{x} - \mathbf{b} = \mathbf{s}$. Ideally the so-called search vector \mathbf{s} becomes $\mathbf{0}$ when the system is solved. So the goal is to minimize \mathbf{s} . Assume we have the scalar function $f(\mathbf{x})$:

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + c, \quad (\text{A.1})$$

where c is a scalar constant. The derivative of this function when \mathbf{A} is symmetric is given by:

$$f'(\mathbf{x}) = \mathbf{A} \mathbf{x} - \mathbf{b}, \quad (\text{A.2})$$

which is exactly our system. This means that we solve the system (minimize \mathbf{s}), when $f'(\mathbf{x}) = 0$. It is important to note here that for every positive-definite matrix the minimum is unique and no other (local) minima exist. Therefore, \mathbf{A} must be positive-definite to converge to a suitable solution.

Clearly \mathbf{s} relates to $f'(\mathbf{x})$. To solve $f'(\mathbf{x}) = 0$, an initial guess can be seen as the starting point within the solution space in which we have to search. Typically, this search vector is set to the null vector, however in some other cases a previously found \mathbf{x} vector might be useful to decrease the number of iterations.

The number of iterations needed by the algorithm corresponds to the number of steps that are needed to move from the starting point towards the minimum. The direction and step size taken per step greatly influences the total number of steps needed. Naturally taking the direction with the steepest decrease seems a promising method, which is conveniently called the 'Steepest Descent' method. This method does converge but the algorithm might decent in one direction multiple times before converging, and thus these steps chosen according to the steepest descent rule are not optimal.

A better method is given by the observation that we have \mathbf{A} -orthogonal directions to move in. For an $n \times n$ matrix n such directions exist. Two vectors \mathbf{i} and \mathbf{j} are \mathbf{A} -orthogonal, or conjugate, if and only if $\mathbf{i}^T \mathbf{A} \mathbf{j} = 0$. This means that \mathbf{i} and \mathbf{j} are orthogonal with respect to the space defined by \mathbf{A} . By using the residual vector $\mathbf{r} = -f'(\mathbf{x}_{current})$ we can approximate how big a step in a certain direction should be. To make this step the step size α is calculated. Then \mathbf{x} the result is updated to $\alpha \cdot \mathbf{s}$. The residual vector then is update by $\mathbf{r} = -\alpha \cdot \mathbf{A} \cdot \mathbf{s}$ since the algorithm made a step with size α in the $\mathbf{A} \cdot \mathbf{s}$ direction. From that definition it is clear that \mathbf{r} is the direction of steepest descent. So if every direction is used once the step sizes, α , of each step is stored in \mathbf{x} . This is exactly what the Conjugate Gradient method does.

By using a preconditioner \mathbf{M} that approximates \mathbf{A} , the number of iterations, and thus the computation time, can be decreased by computing \mathbf{x} for

$$\mathbf{M}^{-1} \mathbf{A} \mathbf{x} = \mathbf{M}^{-1} \mathbf{b}. \quad (\text{A.3})$$

This Equation A.3 has the same solution for \mathbf{x} as without preconditioning. Here $\mathbf{M}^{-1} \mathbf{A}$ is almost the identity matrix since \mathbf{M} is just an approximation of \mathbf{A} . This makes the whole computation easier when \mathbf{M} is invertible.

Incomplete Cholesky factorization of a symmetric positive definite matrix generates a sparse approximation of the Cholesky factorization of the given matrix. The Cholesky

```

Data: Positive-definite symmetric matrix  $\mathbf{A}$ , vector  $\mathbf{b}$ , tolerance factor
         toleranceFactor, maximum number of iterations maxIterations
Result: Solves  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$  for  $\mathbf{x}$ 
set  $\mathbf{x}$  to zero;
/* initialize auxiliary data                                     */
set  $\mathbf{z}$  to zero;
/*  $\mathbf{z}$  is an additional vector for storing results           */
 $\mathbf{r} \leftarrow \mathbf{b}$ ;
/*  $\mathbf{r}$  is the residual vector                                 */
residual  $\leftarrow |\max(\mathbf{r})|$ ;
if residual = 0 then
    | iterations  $\leftarrow$  0;
    | return  $\mathbf{x}$ ;
end
tol  $\leftarrow$  toleranceFactor  $\cdot$  residual;
FormPreconditioner( $\mathbf{A}$ );
ApplyPreconditioner( $\mathbf{r}$ ,  $\mathbf{z}$ );
 $\rho \leftarrow \mathbf{z} \cdot \mathbf{r}$ ;
if  $\rho = 0$  then
    | /* Invalid  $\rho$  value                                     */
    | return null;
end
 $\mathbf{s} \leftarrow \mathbf{z}$ ;
/*  $\mathbf{s}$  is the search vector                                 */
for iteration  $\leftarrow$  0 to maxIterations do
    |  $\mathbf{z} \leftarrow \mathbf{A} \cdot \mathbf{s}$ ;
    |  $\alpha \leftarrow \frac{\rho}{\mathbf{s} \cdot \mathbf{z}}$ ;
    |  $\mathbf{x} \leftarrow \alpha \cdot \mathbf{s}$ ;
    |  $\mathbf{r} \leftarrow -\alpha \cdot \mathbf{z}$ ;
    | residual  $\leftarrow |\max(\mathbf{r})|$ ;
    | if residual  $\leq$  tol then
    | | return  $\mathbf{x}$ ;
    | end
    | ApplyPreconditioner( $\mathbf{r}$ ,  $\mathbf{z}$ );
    |  $\rho_{new} \leftarrow \mathbf{z} \cdot \mathbf{r}$ ;
    |  $\beta \leftarrow \frac{\rho_{new}}{\rho}$ ;
    |  $\mathbf{s} \leftarrow \beta \cdot \mathbf{s} + \mathbf{z}$ ;
    |  $\rho \leftarrow \rho_{new}$ ;
end
/* Maximum number of iterations exceeded                       */
return null;

```

Algorithm 1: The PCG-algorithm

factorization of a matrix \mathbf{A} generates a triangular matrix \mathbf{L} such that $\mathbf{A} = \mathbf{L} \cdot \mathbf{L}^T$, where \mathbf{L}^T is the transpose of \mathbf{L} . The Modified Incomplete Cholesky (MIC) factorization however takes the sparseness of \mathbf{A} into account by ensuring that every 0 value in \mathbf{A} is also 0 for the corresponding value in \mathbf{L} . Therefore $\mathbf{L} \cdot \mathbf{L}^T$ is as sparse as \mathbf{A} , enabling the

use of sparse matrix libraries.

Moreover, solving $\mathbf{M}\mathbf{x} = \mathbf{b}$ should be done efficiently, since it is done every time the preconditioner is applied in the PCG algorithm, the function *ApplyPreconditioner*. The pseudo code of the PCG algorithm is given in Algorithm 1. An efficient solving of the system is possible by solving $\mathbf{L}(\mathbf{L}^T\mathbf{p}) = \mathbf{b}$. Recall that \mathbf{L} is triangular, and thus the number of variables per linear equation increases linearly. So by using substitution of variables $\mathbf{L}(\mathbf{L}^T\mathbf{p}) = \mathbf{b}$ can be solved efficiently. Since the preconditioner is not necessary for the solver to work, no detailed explanation of the algorithm is given, but details can be found in the book by Bridson [30]. For completeness, Algorithm 2 gives pseudo code to compute the MIC-preconditioner.

Data: Positive-definite symmetric matrix \mathbf{A} with dimension n , amount of MIC compared to Incomplete Cholesky Ω

Result: Gives an approximation for \mathbf{L} where $\mathbf{L}\mathbf{L}^T = \mathbf{A}$

$\mathbf{L} \leftarrow$ the lower triangle of \mathbf{A} ;

for $k \leftarrow 0$ **to** n **do**

$\mathbf{L}_{k,k} \leftarrow \sqrt{\mathbf{L}_{k,k}}$; **for** $i \leftarrow k + 1$ **to** n **do**

$\mathbf{L}_{i,k} = \mathbf{L}_{i,k} - \mathbf{L}_{k,k}$;

end

for $j \leftarrow k + 1$ **to** n **do**

for $r \leftarrow 0$ **to** n **do**

$\mathbf{v}_r \leftarrow \mathbf{L}_{r,k} \cdot \mathbf{L}_{j,k}$;

if $\mathbf{A}_{r,j} \neq 0$ **then**

$\mathbf{w}_r \leftarrow \mathbf{v}_r$;

else

$\mathbf{w}_r \leftarrow 0$;

end

end

$missing \leftarrow \sum (\mathbf{v} - \mathbf{w})$;

for $t \leftarrow j$ **to** n **do**

$\mathbf{L}_{t,j} \leftarrow \mathbf{L}_{t,j} - \mathbf{w}_t$;

end

$\mathbf{L}_{j,j} = \mathbf{L}_{j,j} - \Omega \cdot missing$;

end

end

Algorithm 2: MIC Cholesky factorization for generating the preconditioner for PCG

Bibliography

- [1] M. Nichols, N. Townsend, P. Scarborough, R. Luengo-Fernandez, J. Leal, A. Gray, and M. Rayner. European cardiovascular disease statistics 2012. *European Heart Network, Brussels, European Society of Cardiology, Sophia Antipolis*, 2012.
- [2] H.G. Bogren, M.H. Buonocore, and R.J. Valente. Four-dimensional magnetic resonance velocity mapping of blood flow patterns in the aorta in patients with atherosclerotic coronary artery disease compared to age-matched normal subjects. *Journal of Magnetic Resonance Imaging*, 19(4):417–427, 2004.
- [3] M.D. Hope, T.A. Hope, S.E. Crook, K.G. Ordovas, T.H. Urbania, M.T. Alley, and C.B Higgins. 4D flow CMR in assessment of valve-related ascending aortic disease. *JACC: Cardiovascular Imaging*, 4(7):781–787, 2011.
- [4] R.E. Clough, M. Waltham, D. Giese, P.R. Taylor, and T. Schaeffter. A new imaging method for assessment of aortic dissection using four-dimensional phase contrast magnetic resonance imaging. *Journal of Vascular Surgery*, 55(4):914–923, 2012.
- [5] H.G. Bogren and M.H. Buonocore. 4D magnetic resonance velocity mapping of blood flow patterns in the aorta in young vs. elderly normal subjects. *Journal of Magnetic Resonance Imaging*, 10(5):861–869, 1999.
- [6] M.L. Clark and P. Kumar. *Kumar & Clark’s clinical medicine*. Saunders Elsevier.
- [7] R. Putz and R Pabst. *Sobotta atlas of human anatomy*. Williams & Wilkins, 2009.
- [8] T.O.H. de Jongh, J Buis, H.E.M. Daelmans, E de Jong, W.L.M. Kramer, R. Remmen, P.M. Verhoeff, G.N. Verwijnen, and R Zietse. *Fysische diagnostiek*. Bohn Stafleu van Loghum, 2010.
- [9] Y. Kim. *Control of physics-based fluid animation using a velocity-matching method*. PhD thesis, The Ohio State University, 2006.

- [10] K. Funamoto, T. Hayase, A. Shirai, Y. Saijo, and T. Yambe. Fundamental study of ultrasonic-measurement-integrated simulation of real blood flow in the aorta. *Annals of Biomedical Engineering*, 33(4):415–428, 2005.
- [11] R. van Pelt. *Real-time illustrative Visualization of Cardiovascular Hemodynamics*. PhD thesis, TU/e, 2012.
- [12] J. Lotz, C. Meier, A. Leppert, and M. Galanski. Cardiovascular flow measurement with phase-contrast MR imaging: basic facts and implementation. *Radiographics*, 22(3):651–671, 2002.
- [13] K.R. O’Brien, B.R. Cowan, M. Jain, R.A. Stewart, A.J. Kerr, and A.A. Young. MRI phase contrast velocity and flow errors in turbulent stenotic jets. *Journal of Magnetic Resonance Imaging*, 28(1):210–218, 2008.
- [14] R. Bammer, T.A. Hope, M. Aksoy, and M.T. Alley. Time-resolved 3D quantitative flow MRI of the major intracranial vessels: Initial experience and comparative evaluation at 1.5T and 3.0T in combination with parallel imaging. *Magnetic Resonance in Medicine*, 57(1):127–140, 2007.
- [15] P. J. Kilner, G. Z. Yang, R. H. Mohiaddin, D. N. Firmin, and D. B. Longmore. Helical and retrograde secondary flow patterns in the aortic arch studied by three-directional magnetic resonance velocity mapping. *Circulation*, 88(5):2235–2247, 1993.
- [16] M. Markl, F.P. Chan, M.T. Alley, K.L. Wedding, M.T. Draney, C.J. Elkins, D.W. Parker, R. Wicker, C.A. Taylor, R.J. Herfkens, and N.J. Pelc. Time-resolved three-dimensional phase-contrast MRI. *Journal of Magnetic Resonance Imaging*, 17(4):499–506, 2003.
- [17] M. Schwenke, A. Hennemuth, B. Fischer, and O. Friman. Blood flow computation in phase-contrast MRI by minimal paths in anisotropic media. In G. Fichtinger, A. Martel, and T. Peters, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2011*, volume 6891 of *Lecture Notes in Computer Science*, pages 436–443. 2011.
- [18] N. Foster and D. Metaxas. Controlling fluid animation. In *Computer Graphics International, 1997. Proceedings*, pages 178–188, 1997.

- [19] J.M. Hong and C.H. Kim. Controlling fluid animation with geometric potential: Research articles. *Computer Animation and Virtual Worlds*, 15(3-4):147–157, 2004.
- [20] E.O. Kung, A.S. Les, C.A. Figueroa, F. Medina, K. Arcaute, R.B. Wicker, M.V. McConnell, and C.A. Taylor. In vitro validation of finite element analysis of blood flow in deformable models. *Annals of Biomedical Engineering*, 39(7):1947–1960, 2011.
- [21] T.I. Yiallourou, J.R. Krger, N. Stergiopoulos, D. Maintz, B.A. Martin, and A.C. Bunck. Comparison of 4D phase-contrast MRI flow measurements to computational fluid dynamics simulations of cerebrospinal fluid motion in the cervical spine, 2012.
- [22] P. van Ooij, J.J. Schneiders, H.A. Marquering, C.B. Majoie, E. van Bavel, and A.J. Nederveen. 3D cine phase-contrast mri at 3T in intracranial aneurysms compared with patient-specific computational fluid dynamics. *American Journal of Neuroradiology*, 2013.
- [23] A. Ivankovic, A. Karac, E. Dendrinou, and K. Parker. Towards early diagnosis of atherosclerosis: The finite volume method for fluid-structure interaction. *Biorheology*, 39(3):401–407, 2002.
- [24] M. Oshima, R. Torii, T. Kobayashi, N. Taniguchi, and K. Takagi. Finite element simulation of blood flow in the cerebral artery. *Computer Methods in Applied Mechanics and Engineering*, 191(6-7):661–671, 2001.
- [25] H.M. Huang, M.C. Lee, S.Y. Lee, W.T. Chiu, L.C. Pan, and C.T. Chen. Finite element analysis of brain contusion: An indirect impact study. *Medical and Biological Engineering and Computing*, 38(3):253–259, 2000.
- [26] P. Pratt, F. Bello, E. Edwards, and D. Rueckert. Interactive finite element simulation of the beating heart for image-guided robotic cardiac surgery. *Studies in Health Technology and Informatics*, 132, 2008.
- [27] D. P. Nathan, C. Xu, J. H. Gorman, R. M. Fairman, J. E. Bavaria, R. C. Gorman, K. B. Chandran, and B. M. Jackson. Pathogenesis of acute aortic dissection: a finite element stress analysis. *The Annals of Thoracic Surgery*, 91(2):458–463, 2011.
- [28] C.A. Figueroa, I.E. Vignon-Clementel, K.E. Jansen, T.J.R. Hughes, and C.A. Taylor. A coupled momentum method for modeling blood flow in three-dimensional

- deformable arteries. *Computer Methods in Applied Mechanics and Engineering*, 195 (4143):5685–5706, 2006.
- [29] N. Xiao, J.D. Humphrey, and C.A. Figueroa. Multi-scale computational model of three-dimensional hemodynamics within a deformable full-body arterial network.
- [30] R. Bridson. *Fluid Simulation For Computer Graphics*. A. K. Peters, 2008.
- [31] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, 1999.
- [32] N Chentanez, T.G. Goktekin, B.E. Feldman, and J.F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–89, 2006.
- [33] Y Zhu and R Bridson. Animating sand as a fluid. *ACM Transactions on Graphics*, 24(3):965–972, 2005. ISSN 0730-0301.
- [34] C Batty, F Bertails, and R Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics*, 26(3), 2007.
- [35] R. Fedkiw, J. Stam, and H.W. Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 15–22, 2001.
- [36] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.
- [37] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 154–159, 2003. ISBN 1-58113-659-5.
- [38] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International*, pages 63–70, 2007.
- [39] W Hong, D.H. House, and J. Keyser. Adaptive particles for incompressible fluid simulation. *Image and Vision Computing*, 24(7):535–543, 2008.

-
- [40] L. Rosenhead. The formation of vortices from a surface of discontinuity. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 134(823):170–192, 1931.
- [41] S.I. Park and M.J. Kim. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 261–270, 2005.
- [42] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24(3):910–914, 2005.
- [43] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.
- [44] F.H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. *Experimental arithmetic, high-speed computations and mathematics*, 1963.
- [45] J.U. Brackbill, D.B. Kothe, and H.M. Ruppel. FLIP: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38, 1988.
- [46] F.H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.
- [47] G. Elert. The physics factbook – density of blood, 2004.
- [48] P. Concus, G.H. Golub, and D.P. O’Leary. *A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations*, pages 309 – 332. Academic Press, 1976.
- [49] H. Zhao. A fast sweeping method for Eikonal equations. *Mathematics of Computation*, 74:603–627, 2005.
- [50] K. Funamoto, T. Hayase, Y. Saijo, and T. Yambe. Numerical experiment for ultrasonic-measurement-integrated simulation of three-dimensional unsteady blood flow. *Annals of Biomedical Engineering*, 36(8):1383–1397, 2008.

-
- [51] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, third edition, 1999.
- [52] S.P. Datarina, J.J. Du Croz, S.J. Hammarling, and M.W. Pont. A proposed specification of BLAS routines in C. *The Journal of C Language Translation*, 3(4): 295–309, 1992.
- [53] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, Third Edition*. Kitware Inc., 2004.
- [54] H. Gudbjartsson and S. Patz. The Rician distribution of noisy MRI data. *Magnetic Resonance in Medicine*, 34(6):910–914, 1995.
- [55] O. Friman, A. Hennemuth, A. Harloff, J. Bock, M. Markl, and H. Peitgen. Probabilistic 4D blood flow mapping. In T. Jiang, N. Navab, J.P.W. Pluim, and M.A. Viergever, editors, *Medical Image Computing and Computer Assisted Intervention (3)*, volume 6363 of *Lecture Notes in Computer Science*, pages 416–423, 2010.
- [56] G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.
- [57] H.G. Bogren, M.H. Buonocore, and D.M. Follette. Four-dimensional aortic blood flow patterns in thoracic aortic grafts. *Journal of Cardiovascular Magnetic Resonance*, 2(3):201–208, 2000.
- [58] J. R. Shewchuk. An introduction to the Conjugate Gradient method without the agonizing pain. Technical report, 1994.