Development of an In-Silico Model of the Arterial Epicardial Vasculature

Entwicklung eines in-silico Modells der arteriellen epikardialenVaskulatur

**Doctoral thesis for a doctoral degree
at the Graduate School of Life Sciences,
Julius-Maximilians-Universität Würzburg,
Section Clinical Sciences**

submitted by

Johannes Martens

from

Köln

Würzburg, 2019

Submitted on:
Office stamp

Members of the Promotionskomitee:

Chairperson: Prof. Peter Jakob

Primary Supervisor: Prof. Laura Schreiber

Supervisor (Second): Prof. Wolfgang Bauer

Supervisor (Third): Prof. Herbert Köstler

Date of Public Defence:

Date of Receipt of Certificates:

Contents

Summary

In dynamic contrast-enhanced (CE) magnetic resonance (MR) perfusion imaging the passage of an intravenously injected contrast agent (CA) bolus through tissue is monitored to assess the myocardial pefusion state. To enable this, knowledge of the shape of CA wash-in through upstream epicardial vessels is required, the so-called arterial input function (AIF). For technical reasons this cannot be quantified directly in the supplying vessels and is thus measured in the left ventricle, which introduces the risk of systematic errors in quantification of myocardial blood flow (MBF) due to bolus dispersion in coronary vessels. This means occuring CA dispersion must be accounted in the quantification process in order to produce reliable and reproducible results. In order to do this, Computational Fluid Dynamics (CFD) simulations are performed to analyze and approximate these errors and deepen insights and knowledge gained from previous CFD analyses on both idealized as well as realistic and pathologically altered 3D geometries.

In a first step, several different procedures and approaches are undertaken in order to accelerate the performed workflow, however, maintaining a sufficient degree of numerical accuracy. In the end, the implementation of these steps makes the analysis of the cardiovascular 3D model of unprecedented detail including vessels at pre-arteriolar level feasible at all. The findings of the Navier-Stokes simulations are thus validated with regard to different aspects of cardiac blood flow. These include the distribution of volume blood flow (VBF) into the different myocardial regions, the areals, which can be associated to the large coronary arteries as well as the fragmentation of VBF into vessels of different diameters.

The subsequently performed CA transport simulations yield results on the one hand confirming previous studies. On the other hand, interesting additional knowledge about the behavior of CA dispersion in coronary arteries is obtained both regarding travelled distance as well as vessel diameters. The relative dispersion of the so-called vascular transport function, a characterizing feature of vascular networks, shows a linear decrease with vessel diameter. This results in asymptotically decreased additional dispersion of the CA time curve towards smaller and more distal vessels. Nonetheless, perfusion quantification errors are subject to strong regional variability and reach an average value of (-28 ± 16) % at rest across the whole myocardium. Depending on the distance from the inlet and the considered coronary tree, MBF errors up to 62 % are observed.

Zusammenfassung

Bei der Messung der myokardialen Perfusion mittels Kontrastmittel (KM)-gestützter Magnet Resonanz Tomographie (MRT) wird die zeitliche Entwicklung der Anströmung eines intravenös injizierten Kontrastmittel-Bolus im Herzmuskelgewebe gemessen und hinsichtlich des Myokardialen Blutflusses (MBF) ausgewertet. Zusätzlich zum Signal des KM im Gewebe ist für eine Quantifizierung des MBF außerdem aber die sogenannte arterielle Eingangsfunktion (AEF) notwendig. Diese Funktion beschreibt, wie das KM durch das versorgende koronare Blutgefäß ins Gewebe einströmt. Aus technischen Gründen ist die AEF nicht direkt messbar, weshalb sie in der Regel im großen Blutvolumen des linken Ventrikels bestimmt wird. Da der KM-Bolus auf dem Weg vom linken Ventrikel aufgrund der Strömungsverhältnisse in den Herzkranzgefäßen einer zeitlichen und räumlichen Veränderung, sogenannter Dispersion unterliegt, birgt die Verwendent der AEF aus dem linken Ventrikel das Risiko systematischer Fehler bei der MBF-Quantifizierung mit dieser Methode. Für eine reproduzierbare und verlässliche Perfusionsmessung mittels KM-gestützter MRT ist daher die Berücksichtigung der Bolus-Dispersion zwingend erforderlich. Um diese Effekte zu untersuchen und eine Fehlerabschätzung zu ermöglichen, werden in dieser Arbeit fluid-mechanische Berechnungen (Computational Fluid Dynamics, CFD) des Blutflusses und KM-Transports in 3D Geometrien von Herzkranzgefäßen durchgeführt.

CFD Simulationen auf realistischen Gefäßgeometrien, wie sie hier präsentiert werden, gehen mit speziellen Anforderungen einher. Dies betrifft sowohl die technische Durchführbarkeit als auch die Wahl der Randbedingungen um physiologisch korrekte Ergebnisse zu erhalten. In dieser Arbeit werden daher verschiedene Ansätze angewendet und validiert, um die zeitlich effiziente aber dennoch numerisch akkurate Berechnung des Blutflusses und KM-Transports in hochdetaillierten 3D Geometrien der kardialen Vaskulatur unter Verwendung realistischer Randbedingungen überhaupt erst zu ermöglichen. Anschließend werden die Ergebnisse der Blutfluss-Simulationen hinsichtlich verschiedener Aspekte validiert. Dies beinhaltet sowohl die Verteilung des gesamten Blutvolumens in die verschiedenen Regionen des Myokards, die Zuordnung dieser Regionen zu den großen epikardialen Gefäßen (die rechte und die Hauptadern der linken Koronarie) sowie das Verhältnis des Volumen-Blut-Flusses zur Größe des betrachteten Gefäßes.

Anschließend werden die Ergebnisse der Blutfluss-Simulationen verwendet, um die CFD-Analyse des KM-Transports in den kleinen koronaren Gefäßen durchzuführen. Diese Analysen bestätigen zum Einen die Erkenntnisse aus vorherigen Arbeiten, liefern jedoch wichtige neue Erkenntnisse über auftretende KM-Dispersion in Abhängigkeit sowohl des Gefäßdurchmessers als auch des zurückgelegten Wegs des KM. Hierfür wird die sogenannte vaskuläre Transportfunktion (VTF) verwendet. Sie beschreibt die Veränderung der AEF auf dem Weg vom linken Ventrikel durch die Herzkranzgefäße aufgrund der Gefäßbeschaffenheit. Die relative Dispersion (RD) der VTF kann als charakteristische Größe eines vaskulären Netzwerks betrachtet werden und die Ergebnisse dieser Arbeit zeigen eine lineare Abnahme der RD mit dem Gefäßdurchmesser. Dies hat zur Folge, dass die zeitliche Verbreiterung der AEF selbst eine asmptotische Sättigung in kleineren distalen Gefäßen aufweist. Nichtsdestotrotz zeigen die Untersuchungen dieser Arbeit, dass die Fehler der Perfusionsquantifizierung mit Bolus-basierten MRT-Messungen starker regionaler Variabilität unterliegen. Im Ruhezustand beträgt dieser Fehler (-28 ± 16) % im Durchschnitt über den gesamten Herzmuskel. In Abhängigkeit vom Abstand zum Inlet des Gefäßmodells und des betrachteten Koronarbaums wird eine maximale Unterschätzung von bis zu 62 % beobachtet.

Chapter 1

Introduction

Despite continually improving prevention and therapy, cardiovascular diseases and particularly coronary artery disease (CAD) represent the most frequent cause of death in the European Union [1] and industrial countries in general ($\sim 17\%$) [2]. CAD is caused by arteriosclerosis of coronary arteries, which is a chronically progressing degeneration of vessel walls. Due to formation of arteriosclerotic plaques in the arteries, vessel constriction (stenosis) or even complete occlusion as well as increased stiffening of the affected vessels can occur. As a consequence, blood flow into the tissue (perfusion) of the heart muscle (myocardium) is constrained. This causes a lack of oxygen, also called *ischemia*, in the tissue. Chest pain (angina pectoris) or cardiac arrythmia (irregular heart beat) can occur as symptoms. If these plaques detach from the vessel wall, they can clog smaller downstream vessels and even cause heart failure or sudden cardiac [3].

It is for these reasons that an early diagnosis of CAD is of fundamental importance for successful treatment. In order to assess pathologically induced changes in myocardial blood flow (MBF), different imaging techniques can be used. One method is Computed Tomography (CT) coronary angiography, where an X-ray contrast agent (CA) is injected into a peripheral vene and visualized by X-ray imaging. By visual analysis, the coronary vasculature can then be examined for the existence of stenoses and arteriosclerotic plaques. Further methods to scan the status of the coronary arteries and MBF are Positron Emission Tomography (PET) and Single-Photon Emission Computed Tomography (SPECT). In exceptional cases even ultrasound examinations can help visualizing the coronary arteries [4]. In contrast to PET, SPECT and CT, which are dependent on the use of ionizing radiation, Magnetic Resonance Imaging (MRI) represents an attractive, largely non-invasive alternative technique [5–7]. In comparison to the nuclear medical methods PET and SPECT, MRI offers better spatial resolution not using radioactive tracers [8–10]. Moreover, in addition to the morphological analysis of the cardiac vasculature, MRI perfusion measurements also give information about the hemodynamic relevance of visually identified pathological alterations [11, 12].

In contrast-enhanced (CE) myocardial MRI perfusion measurements, the passage of an intravenously injected CA bolus through tissue is monitored by rapid dynamic imaging of the myocardium. The subsequent analysis of the obtained concentration time curves can then be performed in a purely qualitative (i.e., visually), a semi-quantitative or a quantitative approach. While the qualitative method only relies on the time evolution of CA flow in the myocardial tissue, both quantitative methods also require the shape of bolus wash-in through the upstream epicardial arteries, the so-called arterial input function (AIF). However, coronary arteries have very small radii and they move due to contraction and relaxation of the surrounding cardiac tissue. Therefore, direct measurement of the AIF in the coronary arteries is technically not feasible. The AIF is thus measured in the left ventricle [13–15]. On the way from the left ventricle through the coronary arteries, the

CA bolus underlies temporal broadening, so-called dispersion, which introduces the risk of systematic errors in MBF quantification [16, 17].

Since these bolus broadening effects cannot be analyzed by use of imaging techniques (for the reasons mentioned above: resolution and movement of the vessels), the analysis is often performed with the help of Computational Fluid Dynamics (CFD) simulations [18], an approach applied widely to generally assess and improve MRI measurements [19–21]. The influence of epicardial vessels on CA bolus dispersion and resulting systematic errors in MBF have been investigated in several studies using both idealized [22, 23] as well as realistic vessel geometries including pathophysiologic changes of the vasculature [24, 25]. Even in the healthy vasculature, these CFD analyses all show systematic underestimation of MBF in MRI measurements due to various parameters (e.g., flow velocity, length, curvature, stenosis shape and degree).

The objective of the work presented here is the CFD analysis of CA transport in large parts of the coronary arterial network. By use of an imaging cryomicrotome dataset [26–28], which allows high resolution of the cardiovascular morphology, these analyses include vessels down to the pre-arteriolar level.

Previous analyses have shown the crucial importance of the choice of boundary condition (BC)s [24, 29–31] in CFD simulations of not only blood flow but also CA transport. In order to model the hemodynamic conditions in the coronary vasculature as realistically and time-efficiently as possible, several preparatory steps are usually taken. One of these consists of the implementation of an advanced BC [32–34], which allows estimation of volume flow curves through a cardiovascular 3D geometry by inclusion of pressure working on the microvasculature, due to contraction and relaxation of the surrounding myocardium (Chapter 3).

In Chapter 4 an approach to analyze the so-called Fractional Flow Reserve (FFR) in cardiac arteries with mild stenoses with the help of CFD simulations is presented. Th FFR is used to assess the severity of coronary stenoses and is of significant clinical importance for decision making of operational intervention. Based on the BC introduced in Chapter 3, the results show good agreement with invasive measurements of the FFR.

Subsequently, the analysis is extended to include CA transport in the coronary vasculature. It comprises a methodological analysis of CA dispersion in a segment of a coronary artery by means of bolus broadening and arrival times at different points along the branches of the segment (Chapter 5) The investigation underlines the complex interplay of several different factors (branching angles, CA concentration gradients) affecting CA transport.

Subsequently, this analysis is expanded onto the cardiovascular imaging cryomicrotome dataset (Chapter 6) including both the orifices of the left and right coronary arteries as well as vessels at pre-arteriolar level. First, blood flow through the left and right coronary arterial networks is examined in detail and previously established scaling laws of vascular volume [35, 36] are benchmarked. Moreover, volume blood flow into different myocardial regions is analyzed. Secondly, CA dispersion is investigated by means of bolus widths and arrival times invessels of different diameters and at varying distances. An asymptotic relationship between the obtained dispersion and the analyzed vessel size is observed. The analysis is completed by an estimation of the subsequent errors in perfusion quantification measurements due to neglected CA bolus dispersion.

Altogether, in this thesis an in-silico model of the arterial epicardial vasculature is presented. In order to make this feasible in the first place and guarantee physiological relevance of the analysis, several technical requirements are considered. The investigation is benchmarked with regard to different physiological aspects. Finally, a detailed and systematic analysis of CA transport in small coronary arteries is performed and the consequences of bolus dispersion on perfusion quantification are analyzed.

Chapter 2

Theory

2.1 Computational Fluid Dynamics

In this section, the governing equations of fluid flow and CA transport are derived, which are the Navier-Stokes equation and the advection-diffusion equation. Furthermore, the finite volume method, which is used to solve these equations in a discretized way, is presented. The derivations described here are mainly based on [37] and [38].

2.1.1 Navier-Stokes Equation

By definition, matter is called a fluid if it deforms without limit under the influence of shear forces, where the dynamics of the resulting deformation are governed by the fluid's viscosity. In particular, the shear force required to deform a fluid body goes to zero when the deformation rate goes to zero.

In addition to this convention, in fluid dynamics, the assumption of continuum physics is made. This means that for a continuous, extensive quantity ϕ the average is considered over a time-varying control volume $V(t)$:

$$\Phi = \int_{V(t)} \rho\phi \,\mathrm{d}V. \tag{2.1}$$

Here ρ denotes the fluid's density and ϕ the corresponding intensive quantity. For a fixed control volume V_0, which coincides with $V(t)$ at time t_0, the time derivative of this expression can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \rho\phi \,\mathrm{d}V = \frac{\partial}{\partial t}\int_{V_0} \rho\phi \,\mathrm{d}V + \int_{\partial V_0} \rho\phi\, \mathbf{u}\cdot\mathbf{n}\,\mathrm{d}S, \tag{2.2}$$

also referred to as *Reynolds Transport theorem.* The time derivative of Eq. 2.1 is given by the change of ϕ within the fixed volume V_0 and the flux of the quantity ϕ through V_0's surface ∂V_0. The vector $\mathbf{u}$ denotes the fluid velocity and $\mathbf{n}$ is the normal of the surface element $\mathrm{d}S$. Using conservation of mass m within the control volume $V(t)$,

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \rho\phi \,\mathrm{d}V = 0, \tag{2.3}$$

and setting $\phi = 1$, Eq. 2.2 becomes

$$\frac{\partial}{\partial t}\int_{V_0} \rho \,\mathrm{d}V + \int_{\partial V_0} \rho\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}S = 0. \tag{2.4}$$

By use of Gauss' divergence theorem on the second term on the left hand side (LHS), the differential form of mass conservation is obtained:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \mathbf{u}) = 0, \tag{2.5}$$

which holds for an arbitrary volume $V(t)$. Under the assumption of incompressible fluids (i.e. $\partial\rho/\partial t = 0$), which is applicable for modelling blood flow, this simplifies to

$$\nabla \mathbf{u} = 0. \tag{2.6}$$

According to Newton's second law, the forces acting on the control volume's surface are given by

$$\frac{\mathrm{d}(m\mathbf{u})}{\mathrm{d}t} = \mathbf{F}. \tag{2.7}$$

Setting $\phi = \mathbf{u}$ in Eq. 2.2, this can be written as

$$\frac{\partial}{\partial t}\int_{V_0} \rho \mathbf{u}\,\mathrm{d}V + \int_{\partial V_0} \rho\,\mathbf{u}\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}S = \mathbf{f}, \tag{2.8}$$

where $\mathbf{f}$ denotes mass-specific forces. The first integral on the LHS thus comprises local changes of momentum within the control volume, whereas the second integral represents momentum changes due to flux across the surface of the considered volume. Furthermore, forces $\mathbf{t}$ acting directly on the surface need to be distinguished from external forces $\mathbf{k}$, which can be considered by writing Eq. 2.8 as

$$\frac{\partial}{\partial t}\int_{V_0} \rho \mathbf{u}\,\mathrm{d}V + \int_{\partial V_0} \rho\mathbf{u}\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}S = \int_{V_0} \rho\mathbf{k}\mathrm{d}V + \int_{\partial V_0} \mathbf{t}\,\mathrm{d}S. \tag{2.9}$$

The surface forces in the second term on the right hand side (RHS) are defined as

$$\mathbf{t} \equiv \underline{T}\mathbf{n}, \tag{2.10}$$

where $\underline{T}$ denotes the stress tensor, consisting of contributions normal (diagonal elements) and tangential (non-diagonal elements) to the surface element $\mathrm{d}S$ with normal vector $\mathbf{n}$. These diagonal and non-diagonal elements can be identified as pressure (normal stress) and viscous (shear) stress, leading to

$$\underline{T} = p\mathbb{1} + \underline{\sigma}. \tag{2.11}$$

Here, p denotes pressure, $\mathbb{1}$ the unity matrix, and $\underline{\sigma}$ the viscous stress tensor.

Transforming the surface integrals in Eq. 2.9, using Gauss' integral theorem, the differential form of impulse conservation is obtained:

$$\frac{\partial(\rho\,\mathbf{u})}{\partial t} + \nabla(\rho\,\mathbf{u}\mathbf{u}) = \nabla\mathbf{t} + \mathbf{k}. \tag{2.12}$$

Using the product rules for differentiation on the LHS,

$$\frac{\partial(\rho\mathbf{u})}{\partial t} = \mathbf{u}\frac{\partial\rho}{\partial t} + \rho\frac{\partial\mathbf{u}}{\partial t}, \text{ and} \tag{2.13}$$

$$\nabla(\rho\mathbf{u}\mathbf{u}) = \mathbf{u}\mathbf{u}\nabla\rho + \rho\mathbf{u}\nabla\mathbf{u} + \rho\mathbf{u}\nabla\mathbf{u}, \tag{2.14}$$

and Eq. 2.12 becomes

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\nabla\mathbf{u}\right) + \mathbf{u}\left(\underbrace{\frac{\partial\rho}{\partial t} + \underbrace{\mathbf{u}\nabla\rho + \rho\nabla\mathbf{u}}_{1.\ =\nabla(\rho\mathbf{u})}}_{2.\ =0,\text{ by Eq. 2.5}}\right) = \nabla\mathbf{t} + \mathbf{k}, \tag{2.15}$$

where (1.) results from the product rules of differentiation and (2.) from mass conservation. Thus,

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\nabla\mathbf{u}\right)=\nabla\mathbf{t}+\mathbf{k} \tag{2.16}$$

is obtained, the general form of the Navier-Stokes equation.

For an incompressible fluid, the viscous stress tensor $\underline{\sigma}$ is defined as

$$\underline{\sigma}=2\mu\underline{E} \tag{2.17}$$

with the fluid's viscosity μ, and Cauchy's strain rate tensor $\underline{E}$, which is defined as

$$\underline{E}=\frac{1}{2}\left((\nabla\mathbf{u})+(\nabla\mathbf{u})^{\mathrm{T}}\right). \tag{2.18}$$

Here, $(\nabla\mathbf{u})$ denotes the velocity gradient and is given by the Jacobian matrix

$$(\nabla u)_{ij}\equiv\frac{\partial u_j}{\partial x_i}. \tag{2.19}$$

Replacing this in Eq. 2.16 requires calculation of the divergence of the stress tensor $\underline{\sigma}$ as follows,

$$(\nabla\underline{\sigma})_i=\frac{\sigma_{ij}}{\partial x_j}=\mu\left(\frac{\partial^2 u_i}{\partial x_j x_j}+\underbrace{\frac{\partial^2 u_j}{\partial x_j x_i}}_{=\frac{\partial^2 u_j}{\partial x_i x_j}=0}\right)=\mu(\Delta u)_i, \tag{2.20}$$

where Einstein's summation convention and Schwarz's theorem of symmetry of second derivatives are used, and $\Delta\equiv\nabla^2$. Furthermore, Eq. 2.6 is applied to set $\partial u_i/\partial x_i=\nabla\mathbf{u}=0$

Finally, this yields the Navier-Stokes equation for an incompressible Newtonian fluid in differential form,

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\nabla\mathbf{u}\right)=-\nabla p+\mu\Delta\mathbf{u}+\mathbf{k}. \tag{2.21}$$

2.1.2 Advection Diffusion Equation

In order to compute the transport of CA (or more generally any substance) through the coronary vessels, an equation that describes convective and diffusive transport is required. This can be derived based on CA mass conservation across the control volume $V(t)$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)}\rho c\,\mathrm{d}V=\sum f_c, \tag{2.22}$$

where c denotes the CA mass fraction in the blood and $\sum f_c$ comprises sources and sinks as well as diffusive transport of CA [39]. According to Fick's law [40], in the absence of sources and sinks, diffusive transport is given by

$$\sum f_c=\int_{\partial V_0}\rho D\,\nabla\cdot c\mathbf{n}\,\mathrm{d}S, \tag{2.23}$$

where D represents the the CA diffusion coefficient. By use of Eq. 2.23 and Eq. 2.2 (setting $\Phi=c$), Eq. 2.22 becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_0}\rho c\,\mathrm{d}V+\int_{\partial V_0}\rho c\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}S=\int_{\partial V_0}\rho D\,\nabla c\cdot\mathbf{n}\,\mathrm{d}S. \tag{2.24}$$

Applying Gauss' integral theorem to replace the surface by volume integrals yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_0}\rho c\,\mathrm{d}V+\int_{V_0}\nabla(\rho c\,\mathbf{u})\,\mathrm{d}V=\int_{V_0}\nabla(\rho D\,\nabla c)\,\mathrm{d}V,\ \text{or} \tag{2.25}$$

$$\frac{\mathrm{d}c}{\mathrm{d}t}=\nabla(D\nabla c)-\nabla(c\,\mathbf{u}), \tag{2.26}$$

which is the advection diffusion equation in differential form. For a constant diffusion coefficient D and incompressible flow, this becomes

$$\frac{\mathrm{d}c}{\mathrm{d}t}=D\Delta c-\mathbf{u}\nabla c. \tag{2.27}$$

In order to solve the governing equations 2.21 and 2.27 numerically, the contained continuous variables need to be described in a discretized way, which is done at fixed points in space and time. Based on [38], this discretization in both time and space is explained in the following sections.

2.1.3 Time Discretization

Discretization of time means that the time axis is divided into fixed sampling points, given by

$$t^{\mathrm{n}}=\mathrm{n}\Delta t, \tag{2.28}$$

where n denotes a time index and Δt the time step size. In the following, an arbitrary function $\Phi(t)$ is considered,

$$\frac{\mathrm{d}\Phi}{\mathrm{d}t}=\mathrm{f}(\Phi(t))\ \text{and}\ \Phi(t=0)=\Phi_0, \tag{2.29}$$

where Φ_0 denotes the starting value of Φ at $t=0$. The term $\mathrm{f}(\Phi(t))$ denotes the operator of the differential equation, which is given by $\nabla(D\nabla c)-\nabla(c\mathbf{u})$ in the case of $\Phi=c$ (Eq. 2.26). Equation 2.29 represents a so-called *initial value problem*, where Φ is a variable dependent on the t,

$$\Phi(t^{\mathrm{n}})\equiv\Phi^{\mathrm{n}},\ \text{and}\ \mathrm{f}(\Phi(t))=\mathrm{f}(\Phi^{\mathrm{n}})\equiv\Phi^{\mathrm{n}}. \tag{2.30}$$

$\Phi^{\mathrm{n}+1}$ at $t^{\mathrm{n}+1}$ is to be computed from Φ^{n} at t^{n} (or even $\Phi^{\mathrm{n}-1},\Phi^{\mathrm{n}-2}$), which are known.

In the *implicit Euler method* the differential quotient of the time derivative is replaced by a difference quotient

$$\frac{\Phi^{\mathrm{n}+1}-\Phi^{\mathrm{n}}}{\Delta t}=\mathrm{f}^{\mathrm{n}+1}, \tag{2.31}$$

which yields

$$\Phi^{\mathrm{n}+1}=\Phi^{\mathrm{n}}+\Delta t\,\mathrm{f}(\Phi^{\mathrm{n}+1}) \tag{2.32}$$

depending on the yet unknown $\Phi^{\mathrm{n}+1}$ at time $t^{\mathrm{n}+1}$. This method is called implicit because it cannot simply be solved for the desired variable $\Phi^{\mathrm{n}+1}$ and requires the iterative solution of a set of algebraic equations. This renders this method computationally expensive; however, in contrast to *explicit* methods, where

$$\Phi^{\mathrm{n}+1}=\Phi^{\mathrm{n}}+\Delta t\,\mathrm{f}(\Phi^{\mathrm{n}}), \tag{2.33}$$

is approximated, the *implicit Euler method* benefits from favourable stability attributes.

Since this method of time discretization is applied in all analyses presented in this work, the discussion is limited to this approach. For more detail and alternative time discretization methods, please see [38, 39].

2.1.4 Finite Volume Method

Analogously to discretization in time, the considered system requires spatial discretization. Therefore, an n-dimensional computational grid is defined within the treated n-dimensional domain. Although several different methods exist to numerically compute solutions of the governing equations on such a computational grid, merely the so-called Finite Volume Method (FVM) will be described here, because it is used throughout this thesis. The description given here is largely based on [39].

In general, the conservation equation for a quantity ϕ is given by

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\left(\rho\phi\,\mathbf{u} - \Gamma\,\nabla\phi\right) = q_\Phi, \tag{2.34}$$

where Γ denotes ϕ's diffusivity and q_ϕ contains sources and sinks. Inserting $\phi = 1$ yields the conservation of mass (Eq. 2.5) and $\phi = \mathbf{u}$ the Navier-Stokes equation (Eq. 2.21). Since the conservation equation 2.34 applies to each cell volume V of the computational grid, by use of Gauss' integral theorem, the following is obtained:

$$\int_V \frac{\partial(\rho\phi)}{\partial t}\,\mathrm{d}V + \int_S \left(\rho\phi\,\mathbf{u} - \Gamma\,\nabla\phi\right)\cdot\mathbf{n}\mathrm{d}S = \int_V q_\Phi\,\mathrm{d}V. \tag{2.35}$$

The integrals can then be approximated by different approaches, which are outlined in the following.

The surface integrals are written as the sum of the integrals on all cell faces S_k,

$$\int_S f\,\mathrm{d}S \approx \sum_k \int_{S_k} f\,\mathrm{d}S = \sum_k F_k, \tag{2.36}$$

where the index k runs over all cell faces. S_k denotes the face area and f represents either the convective ($\rho\phi\,\mathbf{u}\cdot\mathbf{n}$) or the diffusive term ($\Gamma\,\nabla\phi\cdot\mathbf{n}$) in Eq. 2.35. Using the simplest approximation, the so-called midpoint rule, the integral across across each cell face is approximated by

$$F_k = \int_{S_k} f\,\mathrm{d}S = \overline{\phi_k} S_k \approx \phi_f\,S_k, \tag{2.37}$$

with $\overline{\phi_k} S_k$ the mean value of the variable ϕ across the face k and ϕ_f the value at the geometric center of face k. Since the value of ϕ is not available at the face center, it is interpolated from the values that ϕ takes at the centers of the adjacent cells of the considered face. Applying this to all cell faces thus allows the approximate calculation of the integral in Eq. 2.36.

A similar approximation is used in Eq. 2.35 to calculate the occuring volume integrals,

$$Q_\phi = \int_V q_\phi\,\mathrm{d}V = \overline{q_\phi}\Delta V \approx q_{\phi,c}\,\Delta V, \tag{2.38}$$

where $\overline{q_\phi}$ is the mean value of q_ϕ over the cell volume ΔV, which is approximated by $q_{\phi,c}$, the value at the midpoint of the considered grid cell. In the case of constant q_ϕ or linear behavior across the cell volume, this approximation is exact and otherwise of second order.

As mentioned above, evaluation of Eq. 2.37 requires interpolation based on the values of ϕ at the cell centers of two adjacent cells to approximate ϕ_f on the common face. One way to do this is by application of the so-called upwind differencing scheme (UDS). In this method ϕ_f is approximated by the upstream cell center value,

$$\phi_f = \begin{cases} \phi_P \text{ if } (\mathbf{u}\cdot\mathbf{n})_f > 0; \\ \phi_{P-1} \text{ if } (\mathbf{u}\cdot\mathbf{n})_f < 0, \end{cases} \tag{2.39}$$

where $P, P-1$ denote the indices of two adjacent grid cells. This approximation is numerically stable; however, this comes at the cost of it being highly susceptible to so-called numerical diffusion. The reason for this is that in the UDS only the first term of a Taylor series expansion about cell i (for $\mathbf{u} \cdot \mathbf{n} > 0$) is considered,

$$\phi_f = \phi_P + (\mathbf{x}_P - \mathbf{x}_f)(\nabla\phi)_P + \frac{(\mathbf{x}_P - \mathbf{x}_f)^2}{2}(\Delta\phi)_P + \mathcal{O}(3), \tag{2.40}$$

the UDS thus being of first order accuracy. Correspondingly, the leading truncation error is of the form

$$(\mathbf{x}_P - \mathbf{x}_f)(\nabla\phi)_P \tag{2.41}$$

resembling the diffusion term in Eq. 2.21, hence the name numerical diffusion. In the case of the flow direction being oblique to the cell faces, this error is magnified. This can be tackled by increased grid refinement, which comes with higher computational costs.

A more accurate scheme is the linear upwind differencing scheme (LUDS), where the first two terms of the Taylor series (Eq. 2.40) are considered,

$$\phi_f = \begin{cases} \phi_P + (\mathbf{x}_P - \mathbf{x}_f)(\nabla\phi)_P & \text{if } (\mathbf{u}\cdot\mathbf{n})_f > 0; \\ \phi_{P-1} + (\mathbf{x}_{P-1} - \mathbf{x}_f)(\nabla\phi)_{P-1} & \text{if } (\mathbf{u}\cdot\mathbf{n})_f < 0. \end{cases} \tag{2.42}$$

The leading truncation error being of $\mathcal{O}(2)$, the LUDS is thus second order accurate. Both interpolation schemes presented here require compliance with the so-called Courant-Friedrichs-Lewy (CFL) condition,

$$CFL = \left|\frac{u\Delta t}{\Delta x}\right| \leq 1, \tag{2.43}$$

where u denotes the magnitude of the velocity $\mathbf{u}$, and Δx the distance between two adjacent cell centers P and $P-1$. Since all simulations presented in this work use the LUDS, the discussion is confined to these two methods. For more information on other approximations, please see [39].

Having defined an algebraic equation for each cell volume, a full set of algebraic equations is obtained for the whole computational grid, which relates variable values at the center of each cell to the values of neighboring cells,

$$A_P\phi_P + \sum_k A_k\phi_k = Q_P, \tag{2.44}$$

where P is the considered cell and k runs over all cells involved in the approximation. Correspondingly, A_P and A_k denote the approximation's coefficients. Q_P is presumed known and does not depend on the variable ϕ. In case of cell faces at the geometry's boundaries (e.g. walls, inlets, outlets), conditions reflecting the physical properties need to be defined.

This procedure thus yields a well-posed system with an equal number of equations and unknowns, which can be written as a matrix equation

$$\underline{A}\phi = \mathbf{Q}, \tag{2.45}$$

with the matrix $\underline{A}$ being sparse, since by definition each equation shall only contain few unknowns. The vectors ϕ and $\mathbf{Q}$ represent the values in the cell centers and the RHS from Eq. 2.44, respectively. The length of these vectors is given by the total number of grid cells. This system of equations can be solved in several different ways. For non-linear partial differential equations, as is the case here (Eq. 2.21), to do this an iterative approach must be chosen. This involves guessing of an initial solution, linearizing of the solution and improving it until a result is obtained that satisfies a pre-defined convergence criterion [39]. The method used in this work is described in detail in the following section.

2.1.5 Numerical Solution of the Navier-Stokes Equation

In the following, an approach to solve the Navier-Stokes equations (Eq. 2.21) with the approximations described in the previous sections is presented. This derivation is largely based on [39]. Rewriting Eq. 2.45 for the case of the Navier-Stokes equation and using an implicit method for time discretization (Section 2.1.3) yields

$$A_P\, u_{i,P}^{n+1} + \sum_k A_k\, u_{i,k}^{n+1} = Q^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i}\right)_P, \tag{2.46}$$

with the same definitions as before. Thus, P is the cell index and k runs over all neighboring cells with A_P, A_k the corresponding matrix entries and the source term Q^{n+1}. Accordingly, $n+1$ denotes that the variables are computed for the advanced timestep t_{n+1} as defined in Section 2.1.3. The pressure gradient ${\delta p^{n+1}}/{\delta x_i}_P$ at cell P is independent of the chosen spatial discretization scheme.

Due to the dependency of Q^{n+1} and A on u_i^{n+1}, an iterative approach is required to solve Eq. 2.46. In each iteration, the equations to be solved are of the following type

$$A_P\, u_{i,P}^{\kappa} + \sum_k A_k\, u_{i,k}^{\kappa} = Q^{\kappa-1} - \left(\frac{\delta p^{\kappa-1}}{\delta x_i}\right)_P, \tag{2.47}$$

where the index κ is introduced to replace the timestep $n+1$ from the time discretization. This new index denotes the current iteration, which yields an adjusted (improved) estimation of u_i^{n+1}. The values on the RHS result from the previous iteration $\kappa-1$, which means that the obtained velocities $\tilde{u}_i^{\kappa}$ do not fulfill the continuity equation.

To adjust this, a common approach is the correction of the pressure and velocity fields by small modulations u' and p',

$$u_i^{\kappa} = \tilde{u}_i^{\kappa} + u_i' \text{ and } p^{\kappa} = p^{\kappa-1} + p'. \tag{2.48}$$

Introducing this into Eq. 2.46 yields the following correlation between the corrections

$$u_{i,P}' = \hat{u}_{i,P}' - \frac{1}{A_P}\frac{\delta p'}{\delta x_i}_P, \text{ where} \tag{2.49}$$

$$\hat{u}_{i,P}' = \frac{\sum_k A_k\, u_{i,k}'}{A_P} \text{ is defined.} \tag{2.50}$$

Under the assumption that the continuity equation ${\delta u_i^{\kappa}}/{\delta x_i} = 0$ is fulfilled,

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_P}\frac{\delta p'}{\delta x_i}\right]_P = \left[\frac{\delta \tilde{u}_i^{\kappa}}{\delta x_i}\right]_P + \left[\frac{\delta \hat{u}_i'}{\delta x_i}\right]_P \tag{2.51}$$

is obtained, which represents the necessary pressure correction. The first term on the RHS, containing the unknown velocity correction $\tilde{u}_i^{\kappa}$, is ignored in the approach described here. As a consequence, this solution method, also known as the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm [41], converges rather slowly and requires a small timestep size Δt in order to fulfill the CFL condition (Eq. 2.43). As soon as the pressure correction Eq. 2.51 is solved, the velocities can be corrected by use of Eq. 2.48 and 2.49.

By use of so-called relaxation factors, convergence of the SIMPLE algorithm can be accelerated

$$p^{\kappa} = p^{\kappa-1} + \alpha_p p',\ 0 \le \alpha_p \le 1, \tag{2.52}$$

$$u_i^{\kappa} = \tilde{u}_i^{\kappa-1} + \alpha_u u_i',\ 0 \le \alpha_u \le 1, \tag{2.53}$$

and larger timesteps can be applied. Nonetheless, this algorithm is mainly used for steady-state computations.

In order to solve unstationary problems, this approach can be adapted by adding a second corrector step

$$u''_{i,P} = \hat{u}'_{i,P} - \frac{1}{A_P}\frac{\delta p''}{\delta x_i}_P, \tag{2.54}$$

where $\hat{u}'_{i,P}$ is calculated by Eqs. 2.48 and 2.50 as described above (neglecting $\tilde{u}'_i$ in Eq. 2.51). Subsequently, by use of the continuity equation (${\delta u^\kappa_i}/{\delta x_i} = 0$) as above, a second pressure-correction can be applied

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_P}\frac{\delta p''}{\delta x_i}\right]_P = \left[\frac{\delta \hat{u}'_i}{\delta x_i}\right]_P . \tag{2.55}$$

This iterative method to solve Eq. 2.51 is known as the Pressure-Implicit with Splitting of Operators (PISO) algorithm [42].

The above considerations about the PISO algorithm can be summarized by the following steps:

1. Setting up of the discretized momentum equation (Eq. 2.47) with the solutions from previous step u^n_i and p^n (or the initial boundary conditions) for an estimation of the required velocity fields.
2. Solution of Eq. 2.47 for $\tilde{u}^\kappa_i$.
3. Solution of pressure correction (Eq. 2.51) to obtain p'.
4. Calculate velocity correction (Eq. 2.50, 2.48) u'_i.
5. Solution of second pressure correction (Eq. 2.55) for p''.
6. Calculate velocity correction (Eq. 2.50, 2.48).
7. Repeat steps 2.-6. (PISO loop) until convergence criterion is fulfilled and advance to next timestep.

Analogously to the SIMPLE algorithm, the PISO algorithm is only stable if the CFL condition (Eq. 2.43) is fulfilled for every timestep in the computation. In the case of highly refined computational grids (small Δx), this reduces the maximum allowed timestep size Δt and can thus lead to prohibitively long computation times in non-steady CFD simulations.

This can be tackled with the so-called Combination of PI(SO) and (SI)MPLE algorithm (PIMPLE) approach, which combines the SIMPLE with the PISO algorithm [43].[1] By use of relaxation factors (Eq. 2.52, 2.53) and repetition of steps 1.-6. from the above list in so-called PIMPLE loops, the maximum timestep size Δt can be increased. In other words, each timestep of the transient problem is treated as a steady-state case, where relaxation factors are applied in order to reach faster convergence.

Adept choice of relaxation factors and appropriate numbers of inner (PISO) and outer (PIMPLE) loops, allows stable execution of the above calculations with increased timestep size Δt and thus larger CFL number. However, the used CFL number should not be increased too much, in order to retain all temporal information required to correctly describe the physical problem in question.

[1]This algorithm is implemented in the software package *OpenFOAM*, which is used for all simulations of this thesis.

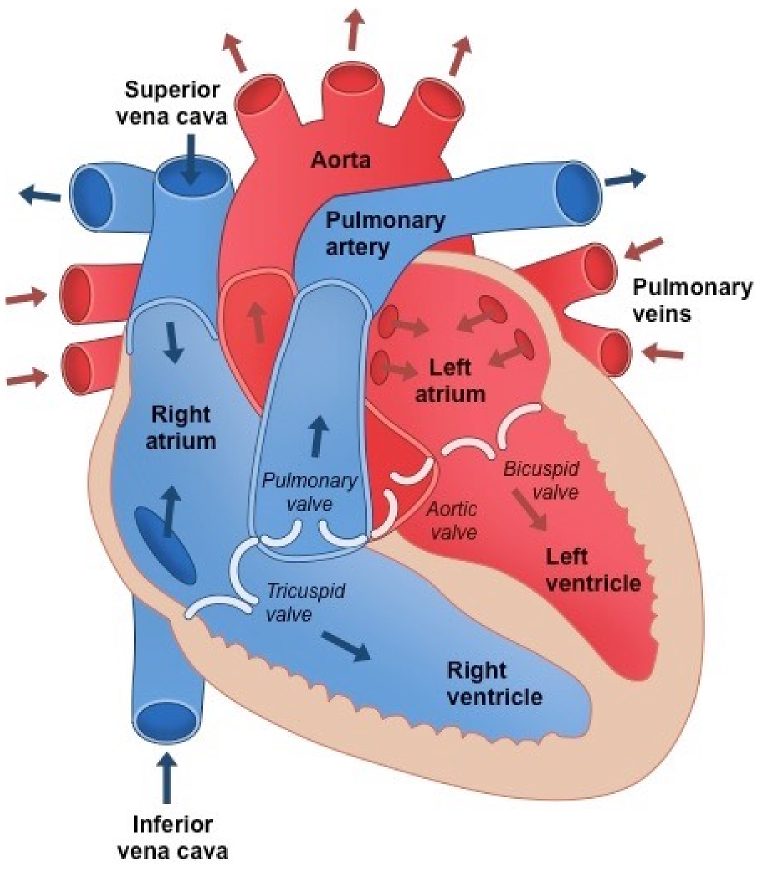


Figure 2.1: Structure of the human heart [47]

2.2 Coronary Blood Flow

This chapter contains a general description of blood circulation in the human organism, with a focus on the role of the heart, regarding its functionality, structure and blood supply. For a more detailed explanation, the interested reader is referred to [44–46].

2.2.1 The Circulatory System

Circulation of blood in the human organism is regulated by the circulatory or cardiovascular system. It consists of the heart that serves as the system's driving pump, and the blood vessels, in which blood is transported and directed to the various organs. This system can be subdivided into the pulmonary and the systemic circuit, which are passed by the bloodstream one after the other.

The heart itself consists of four chambers, left and right atrium and ventricle, respectively, that are associated with the two circuits as depicted in Figure 2.1. Deoxygenated blood coming from the organs enters the right atrium through the Venae Cavae and is pumped towards the lungs through the pulmonary arteries by the right ventricle for oxygenation. This part of the circulatory system represents the pulmonary circuit.

Coming from the lungs, the reoxygenated blood flows through the pulmonary veins into the left atrium and enters the systemic circuit. The left ventricle then pumps the blood through the aortic valve into the Aorta, which distributes the oxygenated blood in the organism supplying organs and tissue with nutrients. Afterwards, the deoxygenated blood is carried back to the heart by the venous system and the whole process is repeated.

The vascular system, which regulates the transport of blood from the heart and back can be considered to consist of seven types of vessels grouped by size and function spanning several orders of magnitude. As described above, the largest vessels, the Aorta (2.5-3.5 cm) and the Venae Cavae (2-3 cm), connect the heart with the systemic circuit. From the Aorta, the arteries branch off and direct the blood towards the different organs and tissues. The arterioles that follow exert the largest share of hemodynamic resistance to blood flow at rest ($\sim$ 60 % [48]). However, by vaso-dilation and -constriction (i.e., increasing and decreasing lumen diameter), the arterioles also regulate the amount of blood that passes, thus adapting to the recipient's tissue requirements. In case of increased blood demands of the respective organ, reductions of arteriolar flow resistance of up to $\sim$ 80 % [48] are possible. The transition from arteries to arterioles is a fluent transition, where different ranges defining arterioles exist. A common range is $20 - 100\,\mu$m [49, 50]; however, some works also count larger vessels and even distinguish pre-arterioles from arterioles [48].

Draining the arterioles, the major share of exchange of oxygen and nutrients between tissue and blood takes place in the capillaries with a diameter of $5 - 10\,\mu$m. Depleting the capillaries, the oxygen-deficient blood passes the venous system through vessels of increasing size, the venules, the veins and finally the Venae Cave until it reaches the right atrium.

As described above, blood flow to the different organs is governed by the vascular system. The total cardiac output at rest is ($\sim$ 2000 ml/min [51]), with the heart itself requiring $\sim$ 5 % [52] of this, corresponding to $\sim$ 200 ml/min [52–54]. The blood flow to the entire heart muscle (myocardium) can be put into relation with the weight of the heart ($\sim 300 - 350$ g in a healthy adult human [55]). However, since this value can vary on the mm scale within the myocardium [56, 57] (and other organs as well), MBF is usually quantified in separate myocardial segments,

$$MBF = \frac{F_{\text{segment}}}{m_{\text{segment}}} \text{ in ml/min/g,} \tag{2.56}$$

where F_{segment} represents blood volume flow into a fixed myocardial segment of weight m_{segment}. Blood is transported to the heart muscle through the coronary arteries, which branch off right at the beginning of the Aorta. The particular properties of blood flow through the coronary vasculature is treated in detail in the following section.

2.2.2 The Coronary Vasculature

After emanating from the Aorta just above the aortic valve, the left and right coronary artery largely follow paths as outlined in Figure 2.2. The two arteries are named after the part of the heart (left and right) they mainly supply with blood. After a few cm, the left main coronary artery (LMCA) (sometimes the term main is omitted as in Figure 2.2) bifurcates into Left Circumflex (LCX) and Left Anterior Descending (LAD).

From the base (the "top") of the heart, the LAD runs towards the apex of the heart (i.e., the tip of the left ventricle) mainly supplying the Septum (the partitioning wall between right and left ventricle) and anterior regions of the left ventricle. The other major branch of the LMCA, the LCX supplies the lateral regions of the heart. The right coronary artery (RCA) mainly perfuses the right ventricle; however, inferior regions of the left ventricle are also reliant on right coronary blood flow. As described in Section 2.2.1, both coronary arterial systems bifurcate into ever smaller vessels down to the capillary level to supply the myocardium with blood. This bifurcating system is also referred to as the coronary tree and represents a so-called end artery. For an end artery, a portion of tissue associated with one of these arteries is supplied exclusively by this specific vessel. As a consequence, pathologic obstruction (e.g. constriction or even complete occlusion of vessels) can lead to reduced (or total stop) of myocardial tissue oxygenation, which can result in myocardial infarction.

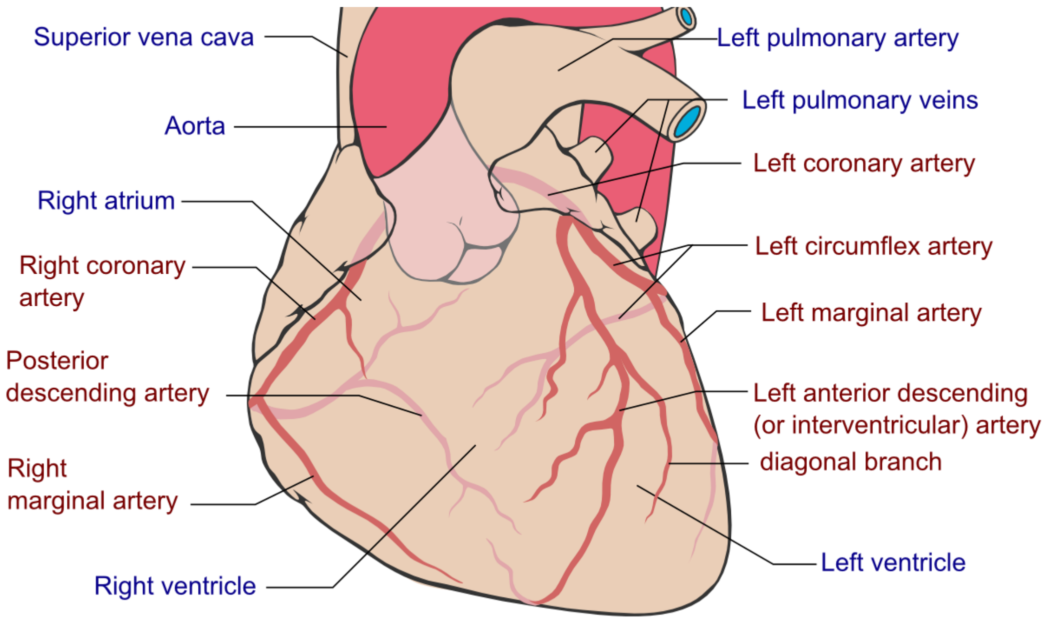


Figure 2.2: Anterior view of the coronary arteries [58]. This image represents exemplary branching of the cardiac vasculature, which can vary significantly between different individuals. E.g. the left main coronary artery (LMCA) can branch into three large vessels, or arteries emanating from the Left Anterior Descending (LAD) towards lateral regions, like the diagonal branch, can be more numerous.

In a healthy human organism, total coronary blood flow is distributed unevenly into the two main coronary arteries (usually $^{\text{right}}/_{\text{left}} \approx {}^{1}/_{4}$ [59]), which results in very different flow patterns and velocities. Exemplary volume flow curves for both arteries are shown in Figure 2.3. The reasons for this are the different requirements of the left and right ventricle. Supplying the whole systemic circuit (cf. Section 2.2.1) the left ventricle has to muster significantly higher pressures in order to meet the organism's demands. In general, pressures between 80-120 mmHg (cf. Figure 2.3) occur at the orifice of the Aorta. On the other hand, the right ventricle pumps the blood coming from the systemic veins to the right atrium (central venous pressures of ~3-8 mmHg) into the lungs, where much lower pressures (<25 mmHg [60]) than in the Aorta prevail. Because of these different requirements of the left and right ventricle, the myocardium of the left ventricle is usually distinctly thicker than that of the right ventricle. As Fig. 2.3 shows, blood circulates through the heart muscle mostly during diastole in contrast to all other organs where blood flow is increased in systole.

At rest, oxygen extraction in the myocardium amounts to ~ 70 % [51]. This means that under hyperemic conditions (at stress, e.g., exercise), MBF (cf. Eq. 2.56) must be increased in order to meet the increased demands of the myocardium [61]. This enhancement of MBF is called the myocardial perfusion reserve (MPR)

$$MPR = \frac{MBF_{\text{Stress}}}{MBF_{\text{Rest}}}, \tag{2.57}$$

and can get as high as a factor of 4-5. It represents an important indicator of the functionality of the heart. In general, both MBF and MPR are strongly heterogeneous across the myocardium and between different individuals [56, 62].

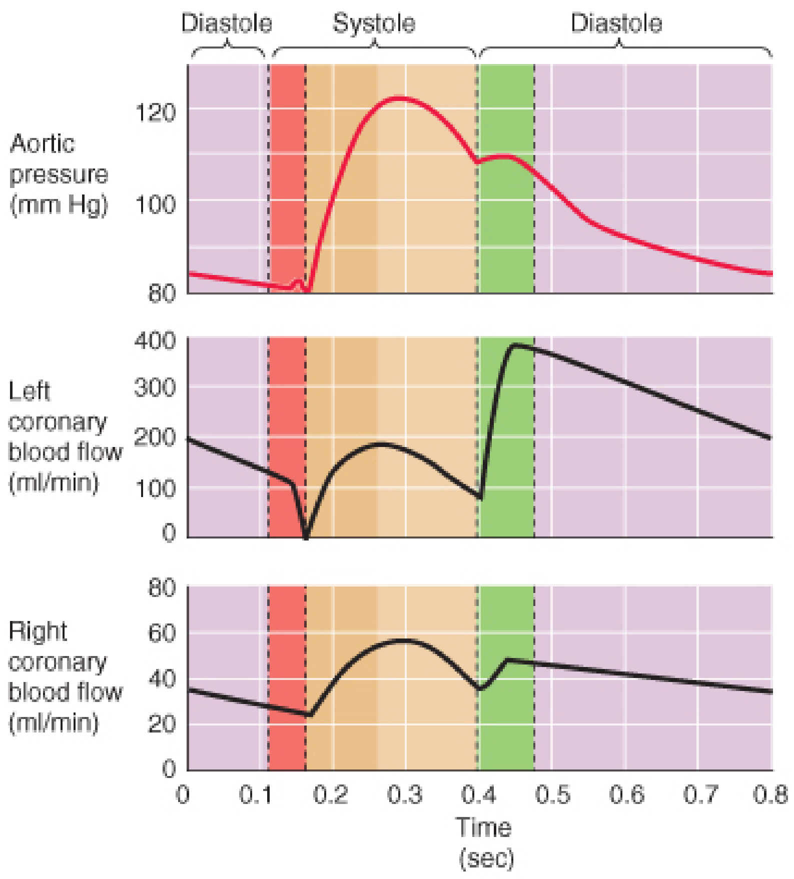


Figure 2.3: Aortic pressure and left and right coronary volume flow curves [59]. During systole, the heart muscle contracts (red section in the plots) and blood is expelled towards the systemic and pulmonary circuits (yellow section). During diastole, the heart muscle relaxes (green) and, subsequently, the ventricles are filled with blood again (violet section). Due to the embedding of the cardiac vessels directly within the contracting and relaxing myocardial tissue, coronary flow resistance varies resulting in decreased flow during systole and increased flow in diastole.

2.2.3 The Viscosity of Blood

Blood, as a composition of non-cellular components, in particular blood plasma and hematocrit (volume percentage of red blood cells (erythrocytes) in blood), represents a *non-Newtonian* fluid [37, 39]. This means its viscosity is dependent on the shear rate $\dot{\gamma}$ within the fluid,

$$\dot{\gamma} = \frac{\Delta u}{\Delta z} = \frac{\mathrm{d}u}{\mathrm{d}z}, \text{ for } \Delta z \to 0, \tag{2.58}$$

where Δu represents the difference in the magnitude of the flow velocity in a fluid between two layers, which are Δz apart. Moreover, due to the presence of particles of finite size in the blood, its viscosity is increased, e.g., in comparison to water ($\mu_{\text{Water}} =\sim 1\,{}^{\text{kg}}/_{\text{m s}}$). Depending on the amount of solid components, μ_{Blood} can be as high as $\sim 4\,{}^{\text{kg}}/_{\text{m s}}$. The erythrocytes in the blood stream possess varying characteristics resulting in different shear rates dependent on the velocity of the blood stream. At low flow velocities, the erythrocytes cluster in stacks (also called *rouleaux*), which form due to their discoid shape [46], resulting in flow perturbations and subsequently increased viscosity. At high flow velocities, on the other hand, the erythrocytes deform and orient themselves with the flow direction, consequently decreasing perturbances between adjacent flow layers. This leads to decreased apparent viscosity [46], down to as low as $\sim 2.5\,{}^{\text{kg}}/_{\text{m s}}$.

However, the non-Newtonian properties of blood are particularly notable in the case of flow through vessels with small diameter ($< 300\,\mu\text{m}$) and low shear rates (Eq. 2.58). Here, the so-called Fahraeus-Lindqvist-effect [63] leads to axial migration of erythrocytes towards the center lines of the vessels and formation of low-viscous layers at the vessel walls [9, 46].

2.3 Cardiac Magnetic Resonance Imaging

In this section, a short introduction to nuclear magnetic resonance (NMR), the foundation of MRI and followingly MR perfusion measurements, is given. It is followed by a basic overview of the MRI operation principles, with a focus on its use in cardiac measurements. More detailed explanations, can be found in [64–66].

2.3.1 Nuclear Magnetic Resonance

The NMR effect is only observed in chemical elements whose nuclei possess a magnetic moment $\mu \neq 0$, which is only the case for nuclei with uneven numbers of nucleons. The nucleus' angular momentum $\mathbf{J}$ determines the magnetic moment as

$$\mu = \gamma \mathbf{J}, \tag{2.59}$$

where $\gamma = {}^{(q\hbar)}/_{(2m)}$ is the element-specific gyromagnetic ratio, with q and m being the particle's electrical charge and mass, respectively, and $\hbar = 1.055 \cdot 10^{-34}\text{kg}\,\text{m}^2\text{s}^{-1}$ Planck's constant.

From quantum mechanics, it follows that the nucleon angular momentum $\mathbf{J}$ (also called *spin*) can only take discrete values:

$$|\mathbf{J}| = \sqrt{\hbar(l(l+1))}, \tag{2.60}$$

$$\text{with the } z\text{-component: } J_z = m_z\hbar. \tag{2.61}$$

Here m_z denotes the spin quantum number, which can only take $(2l+1)$ different values between $\{-l, -l+1, ..., l\}$, with l either half-integer or integer. For simplicity, only the case of hydrogen ${}^1H$ (i.e., only one proton spin to consider, $l = {}^1/_2$ and $m_z = \pm{}^1/_2$) will be treated in this section.[2] However, transfer to particles with nucleon spins $l > {}^1/_2$ is accessible in the literature, e.g., [69].

Due to this intrinsic magnetic moment of the hydrogen nucleons, it is possible to influence the energetic state of the spins by applying an external magnetic field

$$\mathbf{B}_0 = \begin{pmatrix} 0 \\ 0 \\ B_0 \end{pmatrix}. \tag{2.62}$$

As a consequence, the formerly degenerate energy eigenstates split up into two non-degenerate energy eigenstates:

$$E_\pm = \mp\frac{\gamma_p \hbar B_0}{2}, \tag{2.63}$$

Here, γ_p is introduced to denote the gyromagnetic ratio of protons (as in ${}^1H$). The splitting of the energy eigenstates is called the Zeeman-effect and is schematically illustrated in Fig. 2.4. The two states "+" and "–" are also called *spin-up* and *spin-down* and reflect the orientation of the spins relative to the applied magnetic field.

[2]Being the element with the highest occurrence in the human organism (atomic ratio: $\sim 62\,\%$ [67, 68]), it is the hydrogen spins that are most frequently used for medical MRI imaging. This is also the case in MR perfusion imaging, which is addressed in this thesis.

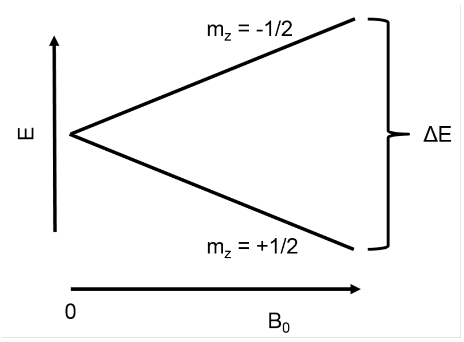


Figure 2.4: Zeeman Splitting. The size of the energy gap ΔE between the two non-degenerate energy levels $E_\pm$ is proportional to the applied magnetic field strength B_0. The parallel spin-up state ($m_z = +1/2$) is energetically more favorable, which is reflected in the reduced energy eigenvalue. Accordingly the anti-parallel spin-down state ($m_z = -1/2$) has a higher energy eigenvalue.

The energy difference between the two energy levels E_- and E_+ thus amounts to

$$\Delta E = \gamma_p \hbar B_0 = \hbar\omega_L, \tag{2.64}$$

where the Larmor-frequency $\omega_L = \gamma B_0$ is introduced. Transitions between the energy levels can be induced by absorption or emission of photons of the frequency ω_L.

If a system with n nucleons is exposed to a magnetic $\mathbf{B}_0$ field (Eq. 2.62), in thermodynamic equilibrium, the occupation numbers $n_\pm$ of the now separated energy levels $E_\pm$ are given by the Boltzmann-distribution

$$\frac{n_+}{n_-} = \mathrm{e}^{\frac{\Delta E}{kT}}, \tag{2.65}$$

with ΔE as in Eq. 2.64, the Boltzmann constant $k = 1.381 \cdot 10^{-23}\mathrm{m}^2\mathrm{kg\,s}^{-2}\mathrm{K}^{-1}$ and the absolute temperature T (in Kelvin), respectively. With $\gamma_p = 2.675 \cdot 10^8\,\mathrm{rad\,s}^{-1}\,\mathrm{T}^{-1}$ the gyromagnetic momentum of ${}^1H$ [70, 71], a common magnetic field strength of $B_0 = 3\mathrm{T}$ in medical settings, and a room temperature of $T = 295\,\mathrm{K} \approx 22°\mathrm{C}$, the energy difference ΔE (Eq. 2.64) can be calculated. Using $\Delta E \ll kT$, a Taylor series expansion of Eq. 2.65 yields

$$\frac{n_+}{n_-} \approx 1.000021, \tag{2.66}$$

reflecting a disbalance in the occupation numbers of the different energy levels of ~ 21 parts per million. Due to the abundance of water molecules in the human body (> 99 % of $\sim 10^{25-27}$ cells, depending on several factors such as age or weight [67, 68]), the resulting effective magnetization $\mathbf{M}$ in direction of the applied magnetic field $\mathbf{B}_0 = B_0\mathbf{e}_z$ is of macroscopic size:

$$M_{0,z} = \frac{\pi N_s \gamma_p^2 \hbar^2 B_0^2}{2kT}. \tag{2.67}$$

Here, N_s represents the number of spins per unit volume and $M_{0,z}$ is the component of $\mathbf{M}$ in z-direction in equilibrium.

In order to stimulate resonant absorption and emission of photonts with energy ΔE [72] (cf. Eq. 2.64) a time-periodic radio frequency (RF) magnetic $\mathbf{B}_1(t)$ field can be applied:

$$\mathbf{B}_1(t) = B_1 \begin{pmatrix} \sin(\omega_L t) \\ \cos(\omega_L t) \\ 0 \end{pmatrix}, \tag{2.68}$$

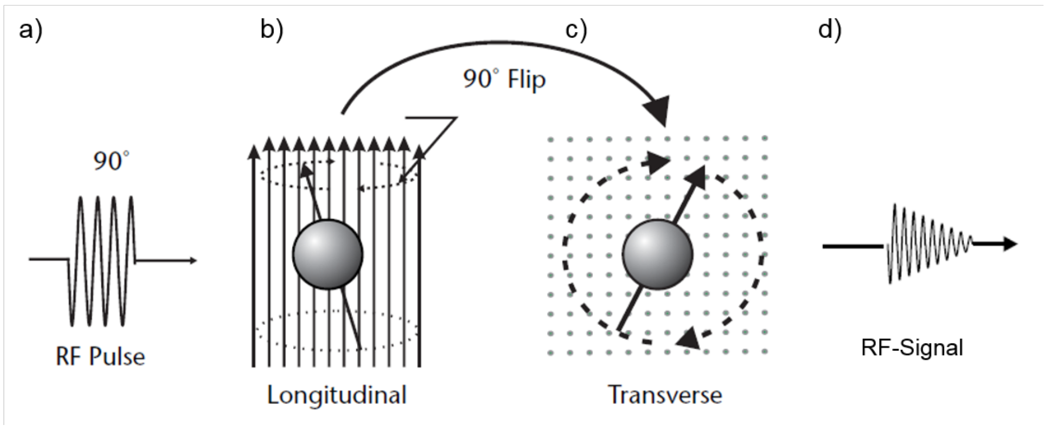


Figure 2.5: Excitation with a 90° radio frequency (RF) pulse (adapted from [66]). In the beginning, the spins are oriented parallel to the direction of the magnetic field. After application of the 90° RF pulse (a,b), the orientation of the spins is deflected until they rotate in the plane transversal to the magnetic field direction (c). As soon as the RF pulse is turned off again, the magnetization tends back to its original orientation. By specifically designed receive coils, this signal change can be measured (d).

As a result of the applied $\mathbf{B}_1$ field, the magnetization $\mathbf{M}$ is subject to a deflection from the direction of the constant magnetic $\mathbf{B}_0$ field by the so-called flip angle [73] towards the x, y-plane. Depending on the pulse duration τ of the applied RF field, the flip angle

$$\alpha = \int_0^\tau \gamma_p \frac{B_1}{2} \mathrm{d}t \tag{2.69}$$

can be adjusted to become as large as 180°, which is equivalent to total inversion of the magnetization $\mathbf{M}$. The effect of a 90°-pulse on a nucleon spin is graphically illustrated in Fig. 2.5. The deflection leads to precession of the effective magnetization around the axis of the static $\mathbf{B}_0$ field (i.e., a rotation in the x, y-plane) with the applied Larmor frequency ω_L.

After application of the RF-pulse of duration τ, the magnetization tends back to the initial thermodynamic equilibrium state $M_{0,z}$ (Eq. 2.67) in a process called spin relaxation. This decay can be described by

$$M_z = M_{0,z} - M_{\tau,z} e^{\frac{t-\tau}{T_1}}, \tag{2.70}$$

$$M_{xy} = M_{\tau,xy} e^{\frac{t-\tau}{T_2}}, \tag{2.71}$$

where M_z and M_{xy} represent the current magnetization components in z-direction and in the x, y-plane, respectively. Accordingly, $M_{\tau,z}$ and $M_{\tau,xy}$ are the magnitudes of the exponentially decaying deflections. The parameters T_1 and T_2 denote the time constants of the two principally responsible relaxation processes [74].

The energy absorbed from the $\mathbf{B}_1$ field, which is released through interactions with the atoms of the surrounding tissue, is called longitudinal (in z-direction) or *spin-lattice* relaxation and is accounted for by the time constant T_1. Additionally, interactions of the spins with each other, so-called transverse (in the x, y-plane) or *spin-spin* relaxation is comprised in T_2. At magnetic B_0-field strengths of 3T, relaxation times lie in ranges between $T_1 =\sim 800 - 2000\,ms$ and $T_2 =\sim 30 - 300\,ms$ depending on the type of biological tissue [75], where always $T_2 < T_1$ for a specific tissue [76, 77].

2.3.2 Magnetic Resonance Imaging

The main principle of MRI is based on the derivations of the previous section. In order to obtain spatially dependent information, the external magnetic $\mathbf{B}_0$ field can be superposed

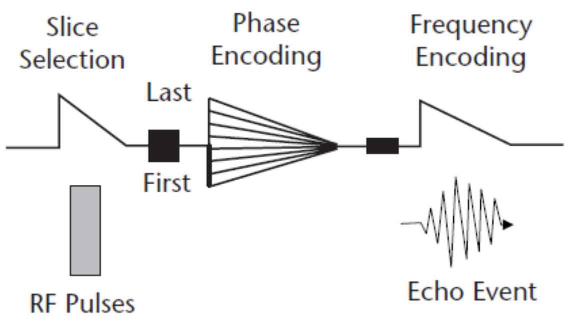


Figure 2.6: Imaging Protocol [66]. The three gradients are switched on one after the other, forming an imaging cycle. The horizontal axis represents time. The slice selection gradient is turned on simultaneously with the switched RF pulses. The phase encoding gradient is changed slightly between each imaging cycle. The frequency encoding gradient is applied during image acquisition. The whole process is repeated in a so-called pulse sequence.

with a spatially varying and time-dependent magnetic field gradient $\mathbf{G}(t)$ of the general form:

$$\mathbf{G}(t) = \begin{pmatrix} x\,G_x(t) \\ y\,G_y(t) \\ z\,G_z(t) \end{pmatrix} \tag{2.72}$$

In the z, y, x-directions, the so-called *slice selection, phase* and *frequency encoding* gradients G_z, G_y and G_x are applied one after the other (cf. Fig. 2.6).[3]

Spatial Encoding

The slice selection gradient is switched on at the same time as the RF excitation pulse and yields a z-dependency of the resonance frequency ω_L (Eq. 2.64). This allows stimulation of spins in defined frequency intervals, i.e., z-layers. The subsequent application of the phase encoding gradient is performed before the final data acquisition (cf. Fig. 2.6). It causes spatially varying angular velocities of spin precession along the y-axis. The variation of $G_y(t)$ between the subsequent imaging cycles determines the spatial resolution of the MR-image in the end. The magnetic field gradient $G_x(t)$ is applied during the actual data readout and provides an x-dependent shift of the temporal variation of the phase differences between the spins from to the previously applied gradients.

Image Reconstruction

After spatial encoding, the spin distribution is thus encoded in an RF-signal, which is read out by a dedicated receive coil. The spatially dependent phase modulations of proton spins from phase and frequency encoding are comprised in the measured signal intensity, which can be represented in the so-called k-space

$$S\left(k_x, k_y\right) \propto \int N_{x,y}\, \mathrm{e}^{-i(k_x\,x + k_y\,y)} \mathrm{d}x \mathrm{d}y. \tag{2.73}$$

Here $N_{x,y}$ represents the spatial distribution of spins at the spatial coordinates x, y and k_x, k_y are the correspoding spatial frequencies. The brightness of each coordinate in k-space thus contains information about the contribution of its unique spatial frequency to the actual MR-image. Accordingly, in each point in k-space frequency and phase information

[3]The choice of the succesion of the three spatial directions is arbitrary; however, the conventionally used order is chosen here.

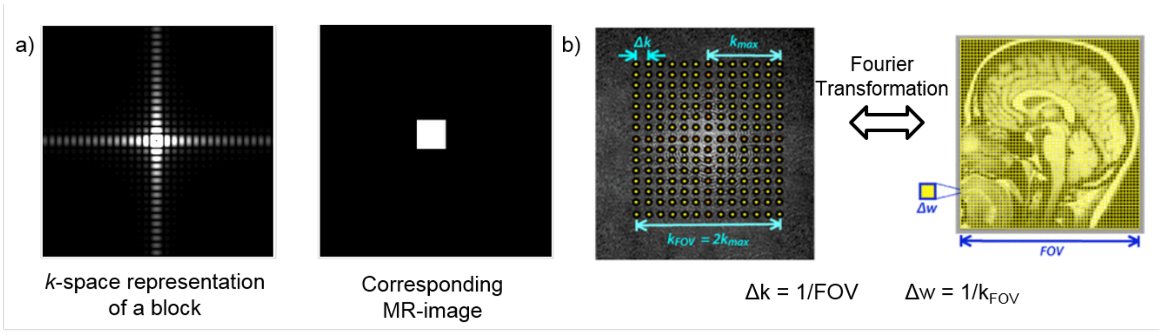


Figure 2.7: Representations in k-space and MR-image (adapted from [77]). Panel a) k-space representation and MR-image of a block. Panel b) In k-space, the image is sampled with the frequency interval Δk. Data near the center of k-space comprises information about low spatial frequences (i.e., general contours) and data from the periphery contains high spatial frequency information (details).

(corresponding to the real and imaginary parts of the oscillations) about every pixel of the final image are present. In Fig. 2.7 a) an example of the k-space representation of a simple block is shown. In order to convert the k-space representation into a spatial representation, an inverse Fourier transformation has to be applied

$$N_{x,y} \propto \int S\left(k_x, k_y\right) \mathrm{e}^{i(k_x\, x + k_y\, y)} \mathrm{d}k_x \mathrm{d}k_y. \tag{2.74}$$

Image Resolution

The resolution of the obtained image is determined by the number of sampling points that is taken in k-space. During the acquisition, n_x points (Δk_x apart) are recorded. Repeating the measurement n_y times, using equidistant G_y ($\rightarrow \Delta k_y$) thus corresponds to filling a cartesian grid in k-space with $n_x \cdot n_y$ measurement points in line.[4] The spatial resolution with which $N_{x,y}$ can be represented is limited by the largest frequency in k-space, $k^{\max} = {}^{(n\Delta k)}/_{2}$.

On the other hand, the size of the so-called field of view (FOV) is determined by the spacing Δk in k-space, $FOV = {}^{2\pi}/_{\Delta k}$. In Fig. 2.7 examples of representations in k-space and corresponding MR-images are shown to give a better impression of the determining factors of MR image resolution.

Image Contrast

MRI does not only allow to assess the spin-density $N_{x,y}$ in an examined tissue volume. The signal also contains information about various tissue parameters such as T_1 and T_2 relaxation times, which vary with the type of tissue that is present. The varying proton densities in these tissue types then yield differences in image contrast (brightness). Depending on the clinical or technical question that is analyzed, adept choice of pulse sequence parameters then allows highlighting of different magnetic properties via the image contrast, which is also known as T_1-, T_2- or spin-weighted imaging [73, 78].

Moreover, the image contrast can be improved by use of specifically designed contrast agents (CAs), enhancing the visualization of myocardial perfusion, where contrast obtained by tissue properties alone is not sufficient. Using a paramagnetic substance, the dipole-dipole interactions between the spins of the substance's unpaired electrons and those of

[4]This is a very common method to fill the k-space, however, different methods (e.g. using radial or spiral trajectories [73]) exist as well.

the protons in the surrounding tissue can be manipulated, resulting in a reduction of the tissue-specific relaxation times as [17]

$$\frac{1}{T_i} = \frac{1}{T_{i,0}} + R_i \cdot c, \text{ for } i = \{1, 2\}. \tag{2.75}$$

Here, c denotes the CA concentration and R_i its relaxivity. By shortening of the relaxation times, a signal increase in T_1- and a decrease in T_2-weighted images is obtained. Common CAs, which are often used in cardiac MRI, are Gadolinium-diethylenetriaminepentacetate (Gd-DTPA) or Gd-DOTA [73]. Recent investigations [79, 80] have shown that adverse effects from Gadolinium retention cannot be completely excluded. For this reason, use of Gd-based CAs is restricted to patients and may not be applied in studies of healthy volunteers.

2.3.3 Contrast-Enhanced Myocardial Perfusion Imaging

Assessment of myocardial perfusion is generally performed by use of a T_1-weighted dynamic contrast-enhanced (CE) MRI measurement. In this method, the first passage of an intravenously injected CA bolus through the myocardial tissue is monitored with a set of structured images. The CA shortens the blood's T_1 relaxation time and a distinct signal rise compared to surrounding tissue can be observed. The principle of perfusion MRI is suppressing the signal within the heart tissue to intensify the signal from the CA. Subsequently, based on the signal increase in the myocardium, tissue blood flow can then be evaluated. Pure visual inspection already allows for a qualitative analysis, where differences in the signal intensities across the myocardium are used to identify hypoperfused regions, i.e., regions with decreased tissue blood flow (cf. Fig. 2.8).

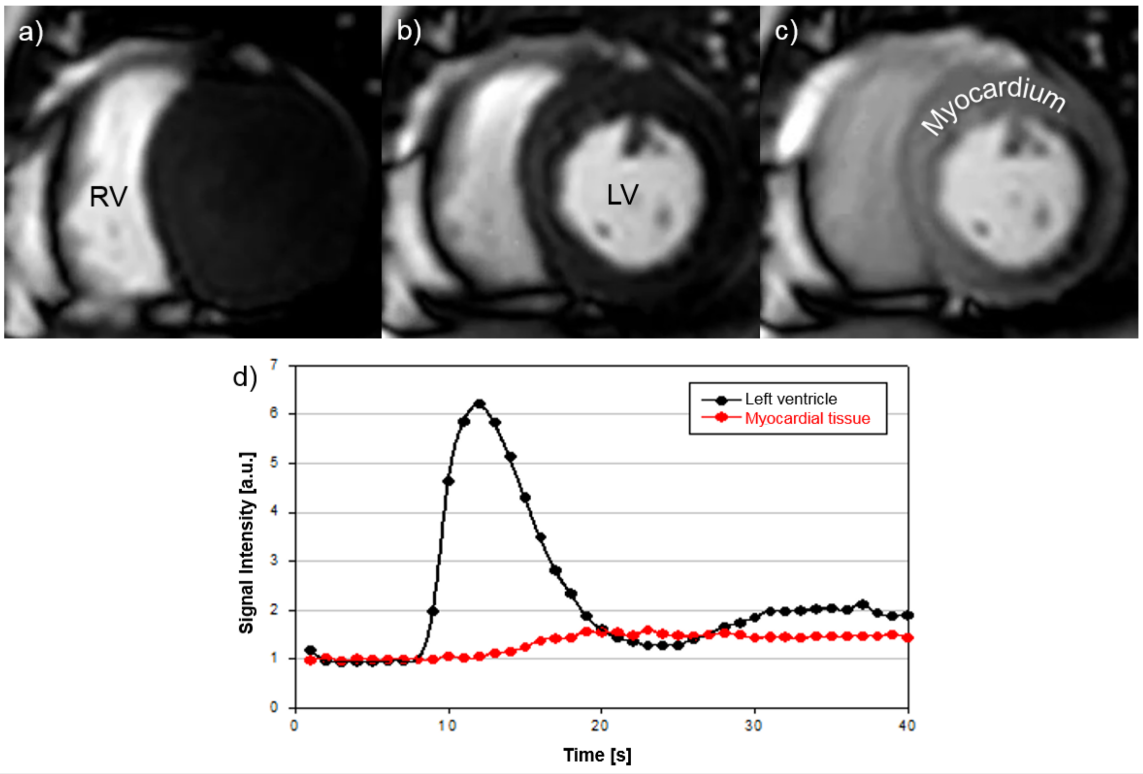


Figure 2.8: CE MRI Perfusion Measurement. Panel a-c) CA flowing through the RV (a), reaching the LV (b) and finally the left ventricular myocardium (c) is made visible by use of a T_1-weighted MRI measurement. Panel d) (adapted from [81]) the obtained signal curves in the LV and the myocardium can be used for perfusion quantification.

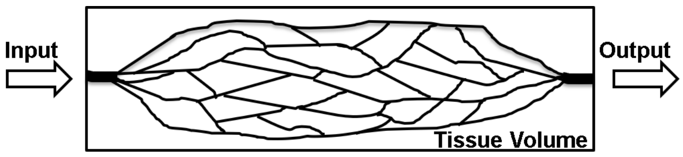


Figure 2.9: Schematic illustration of possible pathways of a flow indicator through a tissue volume. It is supplied and drained by a single in- and output, between which CA can take multiple different paths (i.e., capillaries).

On the other hand, semi-quantitative methods permit relative evaluation of regional myocardial blood flow on the basis of specific parameters of the temporal evolution of the measured tissue signal. For example, the ratio of the signal increase under stress and at rest can be used to calculate the MPR [82]. Similarly, conclusions can be drawn from the signal's maximum value or the time between start of CA wash-in and the measured maximum value (time-to-peak) [83]. Moreover, this evaluation method requires conversion of the signal intesities to actual CA concentration time curves [84]. Since proportionality between these two quantities is only fulfilled for small concentrations and solely in a first approximation, semi-quantitative methods require the usage of low CA doses. Nonetheless, neither qualitative nor semi-quantitative analyses yield absolute values of tissue blood flow and must always be evaluated in relation to other patient-specific data.

For quantitative MBF measurements, in addition to the CA concentration time curve in the myocardial tissue[5], the shape of CA inflow in the nearest upstream vessel, the arterial input function (AIF), is also required. Due to the small size and the movement of the cardiac arteries (relaxing and contracting heart, cf. Section 2.2), direct measurement of the AIF close to the examined tissue volume is in practice not possible. For this reason, the well measurable CA signal in the left ventricle (LV) is used as AIF in quantitative myocardial perfusion measurements. On the CA's path from the LV through the coronary arteries into the myocardium this function changes its shape and underlies spatial and temporal broadening, so-called dispersion. As a consequence, usage of the AIF from the LV introduces the risk of systematic errors in this method to quantitatively assess MBF. In order to calculate absolute MBF values in ml/min/g, so-called tracer-kinetic modelling [17] can be used in the quantitative evaluation of CE myocardial MRI perfusion measurements.

MBF Quantification

Depending on the type of CA that is used for the measurement, it must be differentiated which so-called compartments of the cardiovascular tissue it penetrates. Usually, intravascular, interstitial and intracellular compartments are distinguished [11]. Frequently used MRI CAs, such as Gadolinium-chelates, are confined to the intravascular and interstitial space. Thus, these two compartments are sufficient in the analysis of MRI perfusion measurements.

The principal assumption that is made in this analysis stems from *indicator-dilution* theory and is known as the *central volume principle* [85, 86]. This fundamental theorem states that a vascular system in a tissue volume can be considered consisting of one input and one output, between which CA can take multiple different paths, as depicted in Fig. 2.9. It is assumed that all the considered pathways are in-line, no sinks or sources exist between in- and output and that the volume flow F (e.g. in m^3/s) is constant. These three properties make the vascular system in the tissue volume a so-called *linear* system. After injection of an indicator (CA), the total mass fraction of indicator within the tissue volume

[5]As for semi-quantitative mehtods, only low CA concentrations satisfy linearity between signal rise and concentration time curve.

at time t can be described by the difference between the CA mass fraction time curves at the input $c_{in}(t)$ and the output $c_{out}(t)$ in the following way

$$\begin{aligned} c_{tot}(t) &= \frac{m_{\text{CA}}(t)}{m_{\text{TissueVolume}}} && (2.76)\\ &= F \int_0^t \left(c_{in}(\tau) - c_{out}(\tau)\right) \mathrm{d}\tau, && (2.77)\end{aligned}$$

where $m_{\text{CA}}(t)$ denotes the total CA mass entering the tissue volume of mass $m_{\text{TissueVolume}}$ at time t. The different accessible pathways within the tissue volume (cf. Fig. 2.9) can be described with the help of the so-called residue function

$$R(t) = 1 - \int_0^t h(\tau)\mathrm{d}\tau, \tag{2.78}$$

where $h(t)$ denotes the probability density of transit times, with which a CA particle entering the tissue volume at time 0 is still present in the tissue volume at time t.

The output concentration $c_{out}(t)$ can be rewritten as a convolution of the input concentration $c_{in}(t)$ and the probability density,

$$c_{out}(t) = h(t) \otimes c_{in}(t) = \int_0^t c_{in}(\tau) h(t-\tau)\mathrm{d}\tau. \tag{2.79}$$

With this definition, Eq. 2.77 can be reformulated as follows

$$\begin{aligned} c_{tot}(t) &= F \int_0^t c_{in}(\tau) \left\{1 - \int_0^t h(\tau - t')\mathrm{d}t'\right\} \mathrm{d}\tau && (2.80)\\ &= F \int_0^t \left\{\int_{-\infty}^{\infty} c_{in}(t') \left[\delta(\tau - t') - h(\tau - t')\right] \mathrm{d}t'\right\} \mathrm{d}\tau && (2.81)\\ &= F \int_{-\infty}^{\infty} c_{in}(t') \left\{\int_0^t \left[\delta(\tau - t') - h(\tau - t')\right] \mathrm{d}\tau\right\} \mathrm{d}t', \text{ setting } \tilde{t} = \tau - t' && (2.82)\\ &= F \int_{-\infty}^{\infty} c_{in}(t') \left\{\left[1 - \int_{-\tau}^{t-\tau} h(\tilde{t})\right] \mathrm{d}\tilde{t}\right\} \mathrm{d}t'. && (2.83)\end{aligned}$$

Using the definition of the residue function (Eq. 2.78), this becomes

$$c_{tot}(t) = F \int_{-\infty}^{\infty} c_{in}(t') R(t - t')\mathrm{d}t'. \tag{2.84}$$

Including the definition of blood flow per myocardial tissue mass, $MBF = {}^{F}/_{m}$ (Eq. 2.56) this yields

$$\begin{aligned} c_{tot}(t) &= MBF \cdot m \cdot \int_{-\infty}^{\infty} c_{in}(t') R(t - t')\mathrm{d}t' && (2.85)\\ &= MBF \cdot m \cdot c_{in}(t) \otimes R(t), && (2.86)\end{aligned}$$

the essential equation required for quantification of MBF.

In order to solve this equation for MBF, measurement of $c_{tot}(t)$ and $c_{in}(t)$ is required, because $R(t)$ is not directly quantifiable. A model-independent approach can be taken, where deconvolution of Eq. 2.86 is performed, e.g. the so-called Fermi model approach where $R(t)$ is assumed to have a Fermi shape [8]. However, in this work the chosen approach is by assignment of specific parameters to the different compartments of the vascular system. These describe their volumes accessible for CA transport as well as the permeability between compartments for CA transfer. The general concept of this so-called *Tracer Kinetic Modelling*, which is used in this work is described in the following section.[6]

[6]For the choice of parameters for MBF quantification, please see Section 3.3.

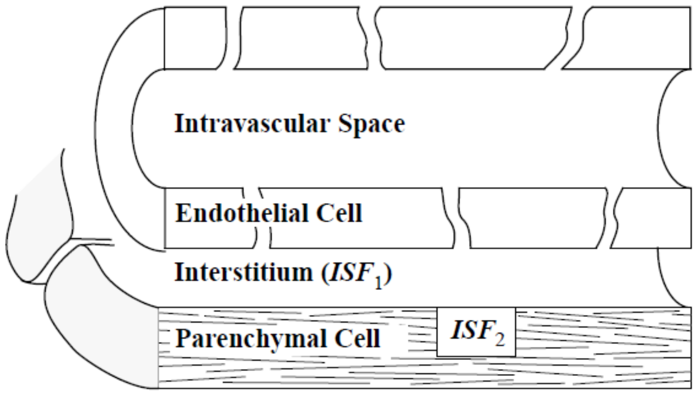


Figure 2.10: Schematic illustration of four regions included in MMID4 [88]. For intravascular tracers, as they are solely considered in this work, only the intravascular and interstitial space are used.

Tracer-Kinetic Modelling with MMID4

In order to realistically model the pharmacokinetic behavior of the vascular system, MMID4 (*Multiple Path, Multiple Tracer, Indicator Dilution, 4 Region Model*, National Simulation Resource, University of Washington, Seattle, USA) [87, 88] allows simulation of CA exchange between interstitial and intravascular space. It has previously been used in several different studies [22–24, 89]. Up to four different compartments can be included in the procedure, namely the intravascular and interstitial space as well as the endothelial and parenchymal cells (cf. Fig. 2.10).

MMID4 permits a simulation analysis of delay and dispersion of CA between the injection site and the organ, where the tissue concentration curve is measured. Analogous to Fig. 2.9, the modelled vascular system is made up of an input artery and an output vene. Between these two major components, up to 20 parallel microvascular paths can be considered, between which blood supply is distributed heterogeneously by use of a probability density function. Each path consists of an arteriole, a tissue exchange unit and a venule, all draining into a single venous output. Exchange between vasculature and tissue is confined to the tissue exchange units, which correspond to the capillary network between arterioles and venules (cf. Section 2.2.1).

2.3.4 Contrast Agent Dispersion in the Coronary Vasculature

Eq. 2.86 reveals that $c_{in}(t)$ is required for MBF quantification, which corresponds to the AIF describing how CA is washed into the tissue. Due to its measurement in the LV and subsequent bolus broadening between the LV and the vessels providing the myocardium, it causes the risk of systematic errors in perfusion quantification. In order to analyze the leading question of this work, the following approaches are used to quantify the occurring CA dispersion.

It is assumed that the coronary vascular system can be treated as a linear system [90] in which conservation of mass is fulfilled. In this case, similar to Eq. 2.79, the shape of the dispersed AIF_{disp} in the vessels providing the myocardial tissue can be expressed as

$$AIF_{\text{disp}}(t-t_0) = AIF_{\text{LV}}(t) \otimes VTF(t-t_0). \tag{2.87}$$

By a convolution of AIF_{LV}, the measured CA signal in the LV, and the vascular transport function (VTF), which includes dispersion effects, the dispersed AIF_{disp} is obtained. As a measure for bolus dispersion, the mean vascular transit time (MVTT) and the VTF's standard deviation SD_{VTF} can be calculated without deconvolution of Eq. 2.87. This is done by calculating the 0th, 1st and 2nd moments of the dispersed and undispersed AIFs as follows:

$$MVTT = \frac{AIF^{(1)}_{\text{disp}}}{AIF^{(0)}_{\text{disp}}} - \frac{AIF^{(1)}_{\text{LV}}}{AIF^{(0)}_{\text{LV}}}, \tag{2.88}$$

$$SD_{VTF} = \sqrt{\frac{AIF^{(2)}_{\text{disp}}}{AIF^{(0)}_{\text{disp}}} - \frac{AIF^{(2)}_{\text{LV}}}{AIF^{(0)}_{\text{LV}}} + \left(\frac{AIF^{(1)}_{\text{LV}}}{AIF^{(0)}_{\text{LV}}}\right)^2 - \left(\frac{AIF^{(1)}_{\text{disp}}}{AIF^{(0)}_{\text{disp}}}\right)^2}. \tag{2.89}$$

The integral momenta $f^{(i)}$ are approximated by a Riemann-sum over all N computed time steps t_k with distance Δt:

$$f^{(i)} = \int_0^\infty t^i f(t) \mathrm{d}t = \sum_{k=0}^{N} (t_k)^i f(t_k) \Delta t. \tag{2.90}$$

According to the central volume theorem for indicator-dilution theory for infinitely short bolus lengths under conditions of a well-mixed single compartment distribution of the CA [86], tissue blood flow (TBF) is given by the ratio of tissue blood volume and mean transit time (MTT) of CA within the tissue volume under consideration

$$TBF = \frac{TBV}{MTT} = \frac{TBV}{\displaystyle\int_0^\infty R(t)\mathrm{d}t}. \tag{2.91}$$

As in Eq. 2.78, $R(t)$ represents the residue function, which describes the proportion of CA remaining in the tissue volume at time t, following the injection of a true, i.e., δ-shaped CA bolus. It only depends on the structure of the vascular bed and is related to the time-dependent concentration of CA in tissue by $R_{\text{measured}} = R \otimes AIF$. MTT can be considered a function of SD_{VTF} as defined in Eq. 2.89 and thus proves highly relevant in the evaluation of vascular dispersion effects.

In addition to this, the standard deviation SD_{AIF} as well as the mean value of the dispersed AIF_{disp} itself can also be used to observe and assess CA dispersion. The standard deviation is given by

$$SD_{AIF} = \sqrt{\frac{\int (t-\bar{t})^2 \cdot c(t)\mathrm{d}t}{\int c(t)\mathrm{d}t}} = \sqrt{\frac{\sum (t_i - \bar{t})^2 \cdot c(t_i)}{\sum c(t_i)}}, \tag{2.92}$$

where $c(t), c(t_i)$ denote the tracer mass fraction at times t, t_i. Accordingly, the mean bolus arrival time $\overline{T}$ is defined by:

$$\overline{T} = \frac{\int t \cdot c(t)\mathrm{d}t}{\int c(t)} = \frac{\sum t_i \cdot c(t_i)}{\sum c(t_i)}, \tag{2.93}$$

where $c(t), c(t_i), t, t_i$ as above. Please note, $\overline{T}$ thus corresponds to the dispersed AIF's first moment, merely shifted from the MVTT by ${}^{AIF^{(1)}_{LV}}/_{AIF^{(0)}_{LV}}$ (cf. Eq. 2.88).

Chapter 3

Preparatory Work

3.1 Motivation

The general workflow of the CFD simulations presented in this thesis consists in several subsequent steps. First, a 3D volume is required in which the blood flow and CA transport computations are to be conducted. This can be a realistic cardiovascular geometry as in [24] or an idealized tube-like model without bifurcation [18] or one bifurcation as in [23]. In the next step, in the utilized 3D geometry, an inlet where blood and CA enter the domain as well as one or multiple outlets are defined where the volume is drained. The remaining parts of the geometry represent the vessel walls where blood and CA cannot penetrate. As soon as these prerequisites are fulfilled, the 3D model is provided with a computational grid to accomplish volume discretization, which is required as described in Section 2.1.4. In addition to these geometrical constraints, BCs at the model inlet and outlets as well as the vessel walls need to be defined for the physical quantities described by the governing equations of blood flow (Eq. 2.21) and CA transport (Eq. 2.26). In this chapter, several steps to assure relevance and accuracy of the performed simulations as well as measures to optimize their computational speed are described.

As explained in Section 2.2 cardiac circulation must be treated differently than systemic circulation, due to compression and relaxation of the myocardial tissue, in which the coronary arteries are embedded. In Section 3.2 a model, which takes account of this effect, will be presented.

In Section 3.3 an alternative volume discretization method will be introduced. It is designed to accelerate the workflow's and simulation's efficiency. One of the major difficulties during preparation of a cardiovascular model for CFD simulations arises from the need for a high quality computational grid. This is necessary to reduce errors due to numerical diffusion. In Section 3.3 a fully automatic meshing procedure is tested and validated on a simple bifurcation model.

In Section 3.4 the influence of the degree of discretization on numerical accuracy is investigated by comparing the simulation results obtained on three grids of different cell numbers. This is performed on the same simple model with one bifurcation as in Section 3.3.

In this work, simulations on highly detailed cardiovascular models are performed. In order to minimize the computational duration, they can be performed on so-called *High Performance Computing* (HPC) clusters. However, this poses specific requirements on the file management of the employed software codes. In Section 3.5 results of a validation of an approach to reduce file numbers during computations with a sub-time stepping method are presented.

Finally, before computationally expensive software applications can be run on a very large HPC cluster (i.e., a*Supercomputer*) their behavior during highly parallelized execution needs to be verified. Section 3.6 addresses this by so-called *scalability testing.*

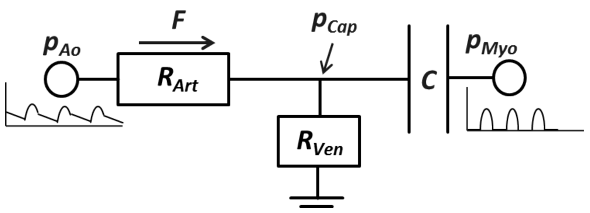


Figure 3.1: Electrical circuit analogon of cardiac blood flow. p_{Ao} and p_{Myo} represent the pressures in Aorta and myocardium, respectively. Relation between flow resistances in arteries and veins, R_{Art}, R_{Ven}, and compliance C of the vessels determine how blood flow F and pressure at capillary level p_{Cap} adjust.

3.2 Improved Boundary Condition

Unlike systemic blood flow, cardiac circulation is increased in diastole compared to systole. This is due to the inherent embedding of the cardiovascular network directly within the myocardial tissue, which it supplies. The compression (systole) and relaxation (diastole) of the myocardium results in decreased and increased flow in cardiac vessels. This effect can be pronounced to an extent that even retrograde flow is observed in large epicardial arteries during early systole [91].

In order to correctly reflect this property characteristic of cardiovascular blood flow in the CFD simulations, dedicated boundary conditions need to be conceived and implemented. A rather sophisticated model makes use of the analogy of cardiac and electrical circuit. It was first proposed by Westerhof et al, [32] and enhanced and further developed in [33, 34, 92, 93].

The Electrical Analog of Coronary Circulation

In this work an adapted version of this model is used to describe this behavior physiologically accurate. In Fig. 3.1 the circuit with the symbols from the electrical analogy is depicted and Table 3.1 lists the corresponding units. The circuit consists of three branches with two external power supplies, p_{Ao} and p_{Myo} for the pressures in the Aorta and the myocardium, respectively. It can be considered as an RC-circuit, where the total resistance is composed of two parallel resistances (left and bottom branches).[1] The capacitor C in the right branch is thus charged and discharged across the parallel resistances depending on its own charge U_C as well as the applied voltages (i.e., pressures p_{Myo} and p_{Ao}). In an electrical RC-circuit the voltage across a capacitor can be written as [95]

$$U_C(t) = U_C(t-\Delta t) + \frac{\Delta t}{C} \cdot (I(t) - I(t-\Delta t)), \tag{3.1}$$

where $I(t), I(t-\Delta t)$ represent the current and $U_C(t), U_C(t-\Delta t)$ the voltage at times t and $t-\Delta t$ across the capacitor. For the physiological analogon this means the pressure "stored" across a vessel's compliance C is given by:

$$p_C(t) = p_C(t-\Delta t) + \frac{\Delta t}{C} \cdot (F(t) - F(t-\Delta t)). \tag{3.2}$$

$F(t), F(t-\Delta t)$ denote the flow through the vessel and C the vessel's compliance with the units listed in Table 3.1.

As described in Chapter 2 CFD simulations require a well-defined set of boundary conditions. p_{Ao} in Fig. 3.1 represents a pressure curve measured in the Aorta. p_{Myo} correspondingly describes the associated pressure curve within the myocardium. Since myocardial pressure is not easily measurable it is approximated by the left ventricular pressure curve

[1]In an electrical RC-circuit a resistance R and a capacitor C are arranged in parallel[94, 95].

Table 3.1: Analogy of electrical circuit and cardiac circulation.

Quantity	Symbol	Unit	Quantity	Symbol	Unit
Current	I	A	Volume Flow	F	$\mathrm{m^3\,s^{-1}}$
Voltage	U	V	Pressure	p	$\mathrm{kg\,m^{-1}\,s^{-2}}$
Electr. Resistance	R	Ω	Flow Resistance	R	$\mathrm{kg\,m^{-4}\,s^{-1}}$
Capacitance	C	F	Compliance	C	$\mathrm{kg^{-1}\,m^4\,s^2}$

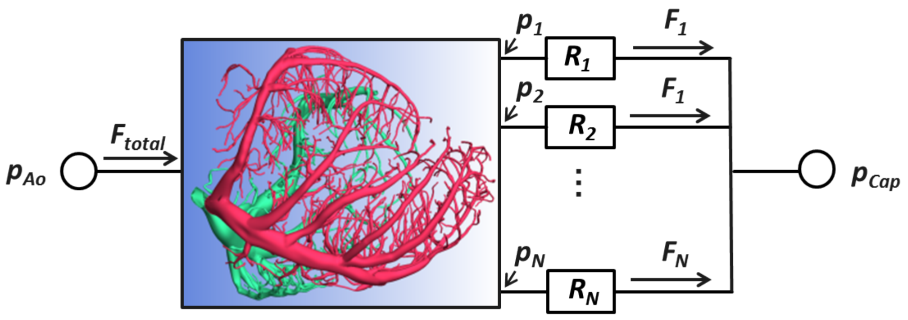


Figure 3.2: Division of arterial resistance into 3D-model with several parallel arranged outlets and associated downstream blood flow resistances. The arterial resistance R_{Art} from Fig. 3.1 is subdivided into a contribution from the model (CFD-simulations) and estimated corresponding outlet resistances R_i. Accordingly, total Flow F splits into F_i, which distribute over all N model outlets. p_i denotes the pressure at outlet i.

multiplied by 0.95, based on [96]. In Fig. 3.1 p_{Cap} is to be understood as the pressure in the middle of the capillaries, thus, the resistances R_{Art} and R_{Ven} (cf. Fig. 3.1) comprise capillary contributions each.

In order to use this electrical analogy for the simulation of blood flow in 3D-models of coronary arteries, the arterial resistance R_{Art} must be split up as sketched in Fig. 3.2.

If the inlet pressure p_{Ao}, the outlet resistances R_i, and the pressure at capillary level p_{Cap} are known, the pressure at the outlets is thus given by

$$p_i = F_i \cdot R_i + p_{Cap}. \tag{3.3}$$

This approach guarantees, that the flows to the different outlets F_i can adapt according to the conditions within the 3D-model. At the same time, intramyocardial pressure p_{Cap} and flow resistance R_i of the remaining vasculature behind the outlets are taken into account, as well.

Estimation of Vascular Resistances

It follows that this method requires knowledge of the outlet resistances R_i. These are approximated on the basis of the self-similarity of the coronary arterial tree [35, 36, 97, 98]. Depending on the outlet radius and the contribution of arteries ($\sim 0\,\%$), pre-/arterioles ($\sim 60\,\%$), capillaries ($\sim 25\,\%$), veins and venules ($\sim 15\,\%$) [48, 99] to total vascular flow resistance, an estimation of R_i downstream of an outlet can be made. According to [35]

$$R_i = \mathrm{const}_R \cdot \frac{L_{\text{crown}}}{{D_{\text{outlet}}}^4} \tag{3.4}$$

applies, where L_{crown} is the cumulative length of all vessels downstream of the outlet with diameter D_{outlet} and $\text{const}_R = R_{\max} \cdot \frac{D_{\max}{}^4}{L_{total}}$, with $D_{\max}$ the most proximal stem diameter (i.e., inlet diameter at the orifice of the LMCA at the Aorta), L_{total} total cumulative crown length and $R_{\max}$ total resistance of full vascular tree starting at $D_{\max}$, respectively. The cumulative length of an outlet crown is given by [36]

$$L_{\text{crown}} = L_{\max} \cdot \left(\frac{D_{\text{outlet}}}{D_{\max}} \right)^{7/3}, \tag{3.5}$$

where $L_{\max}$ is estimated with the help of [97][2]. With the electrical analogy of Fig. 3.1, the total resistance can be defined as

$$R_{\max} = \frac{\Delta p}{F_{\text{total}}}, \tag{3.6}$$

where $\Delta_p = p_{Ao} - p_{preCap}$ represents the pressure drop across the arterial and arteriolar vessels before the capillaries (thus the naming p_{preCap})[3]. F_{total} denotes the total blood flow through the entire coronary tree. This requires knowledge of F_{total}, which is estimated by

$$F_{\text{total}} = \text{const}_F \cdot L_{\max} \text{ [36]}, \tag{3.7}$$

with the stem flow-crown length relationship $\text{const}_F = 6 \cdot 10^{-9} \text{m}^2\text{s}^{-1}$ from [98].

The final parameter from the analogy of cardiac circulation and electric current to be determined is the compliance of the vessels within the myocardium, C. Measurement of this parameter is performed in [33]; however, due to the difficulty to assess this parameter, estimations based on physiologically relevant parameters are made in [34, 93, 100].

Estimation of the Myocardial Compliance

A similar approach to obtain a reasonable value for C is chosen in this work. With the help of the open source software package *QUCS*, the electrical circuit depicted in Fig. 3.1 is set up (see Fig. 3.3) and the influence of C is analyzed.

The units given for R_{Art}, R_{Cap1}, R_{Cap2} and R_{Ven} as well as C in Fig. 3.3 translate one-to-one to their physiologic analogies as given in Table 3.1. The values in Fig. 3.3 are chosen such that they provide typical physiologic flow values for the LMCA, which can be seen in Fig. 3.4. However, it should be pointed out that the plotted graphs represent only the first two cardiac cycles. The system will take some time until it reaches a stationary state, where charging and discharging of the capacitor balance each other.

In equilibrium, an average flow value of $2.4 \cdot 10^{-6}\,\text{m}^3\,\text{s}^{-1} = 144\,\text{ml}\,\text{min}^{-1}$, which is in a realistic range for left coronary blood flow [53, 54, 102, 103], and a ratio of diastolic to systolic flow of ~ 3 is obtained with the parameters chosen as in Fig. 3.3.

Figure 3.5 shows (stationary) results for different values of compliance C. Accordingly, larger values of C lead to an increased ratio of diastolic to systolic flow, whereas lower values have the opposed effect. This can easily be explained by looking at the electrical analogon of the blood flow circuit. A smaller capacitance in an RC-circuit leads to faster charge and discharge of the capacitor and thus smaller flow across the right branch of the circuit. This means the system has a shorter time constant $\tau = R \cdot C$ and pressure differences between the applied pressures p_{Ao} and p_{Myo} have a reduced effect. Both during diastole and systole the equlibrium state of no current (i.e., volume) flowing through the capacitor is reached faster and only the pressure difference between $p_{Ao} - p_{Atrium}$[4] is of interest, which compared

[2]Data is based on measurements performed in pig coronary trees. Since equivalent data for humans is not available, further approximations for CFD-simulations in human datasets are pointed out where applicable.

[3]Please note, that according to the definition in [35], this does not include the capillary contribution to flow resistance.

[3]*Quite Unified Circuit Simulator*, http://qucs.sourceforge.net/.

[4]p_{Atrium} corresponds to U_{atrium} in Fig. 3.3 and represents the lowest pressure in the cardiac circulation.

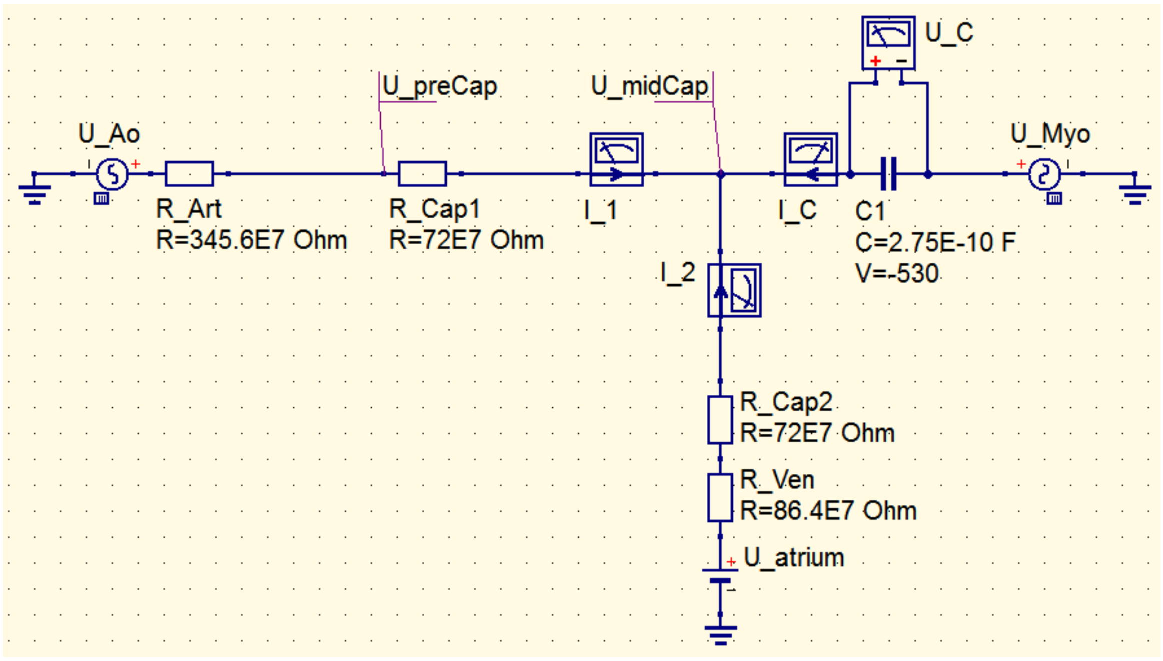


Figure 3.3: Electrical circuit set up with QUCS. The values for R_{Art}, R_{Cap1}, R_{Cap2} and R_{Ven} are estimates of the resistances for the total vascular tree. Together with the assigned value of the capacitor $C1$ (with charge V at $t = 0$) these give physiologic values for total flow (cf. Fig. 3.4a)), where I_1, I_2 and I_C are plotted. The pressure generators U_{Ao} and U_{Myo} are given by read-in of time-dependent csv-files, for simplicity U_{atrium} (the circuit's "earthing") is set constant to $\approx 500\,\text{Pa}$ (estimated as mean value from [101]). The obtained pressure-time curves for U_{preCap}, U_{midCap} and U_C are plotted in Fig. 3.4 b).

to $p_{Ao} - p_{Myo}$ only changes slowly. This means with smaller capacitance C, the currents in systole and diastole even out to similar values faster, thus reducing the ratio of diastolic to systolic flow. This becomes visible in the sketches of the current in the QUCS-model with different values for capacitance/compliance C (cf. Fig. 3.4a), 3.5).

With this knowledge, the approximations for the flow resistances (Eqs. 3.4–3.7) in the circuit and the time dependency of a charging and discharging capacitor in an RC-circuit (Eq. 3.2) the periodic problem can be solved. This is done using several `C++`-programs (cf. Appendix A):

- *scalingLawResistance.cpp* computes the resistance of the downstream lying vasculature depending on the outlet diameter. Viscosity in the small vessels of the coronary tree is approximated by a radius-dependent empirical formula, [104, 105].

- *capillaryPressure.cpp* calculates the pressure at capillary level based on the derivations above. As soon as periodicity is reached, pressure time curves at pre-capillary level are saved to disk for later use in the CFD-solver (cf. Fig. 3.2).

It must be borne in mind that the explanations above treat the "simple" case, where both bottom and left branches consist of one serial resistance each in figs. 3.1 and 3.3. For the CFD-simulations on the actual 3D-geometries with multiple (parallel) outlets as indicated in Fig. 3.2 the situation is more complex. This means, not only one constant compliance C needs to be considered in this case, but each model outlet i with individual outlet resistance R_i requires a separately assigned compliance C_i. In accordance with [34] the above relationship $R_{Art} \cdot C = 1.14$ (cf. Fig. 3.5), which yields physiologic flow values in

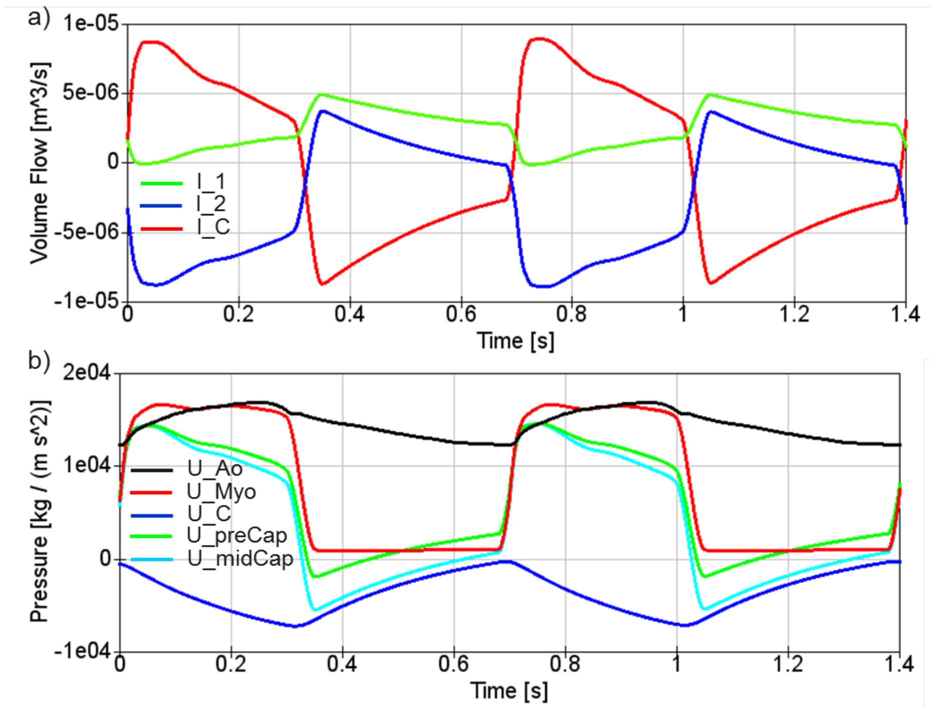


Figure 3.4: First two cycles for volume flow and pressure curves obtained for circuit depicted in Fig. 3.3. a) The obtained currents I_1, I_2 and I_C add up to zero at all times. The combination of the values chosen for the resistances and the compliance depicted in the circuit gives a ratio of diastolic to systolic flow of ~ 3 for I_1 (cf. Fig. 3.5). b) The obtained pressures at the indicated points in Fig. 3.3 reflect the behavior shown in a), $U_{Ao} - U_{preCap}$ is large during diastole and small during systole. The pressure time curve U_{preCap} is relevant for use in the outlet BC.

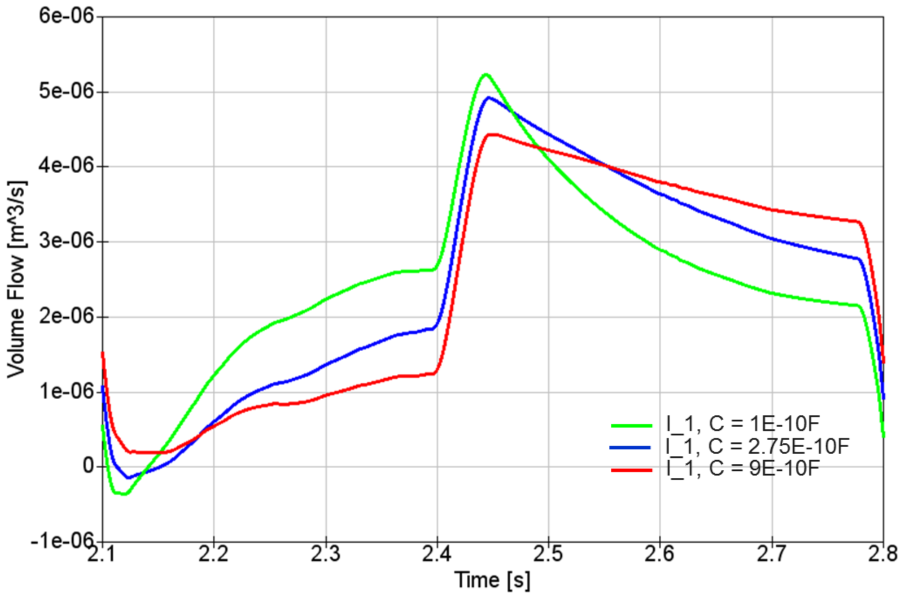


Figure 3.5: Model volume flow I_1 for different compliances C and fixed resistance values (as in Fig. 3.3). The charging time constant $\tau = R \cdot C$ differs for the three values for $C = 1 \cdot 10^{-10}\,\mathrm{F}, 2.75 \cdot 10^{-10}\,\mathrm{F}, 9 \cdot 10^{-10}\,\mathrm{F}$. Assuming $R \approx R_{Art}$, τ evaluates to 0.4 s, 1.14 s and 3.75 s respectively.

the simplified circuit as depicted in figs. 3.1 and 3.3, the individual outlet compliances are defined as

$$C_i = \frac{1.14}{R_{Art}}. \tag{3.8}$$

The implementation of Eq. 3.3 as a boundary condition in a dedicated solver is described in detail in appendix A.3. All simulations in this chapter are performed with this advanced solver.

Conclusion

The described BC represents an approach to enable reliable CFD simulations of blood flow and subsequently CA transport in the coronary vasculature. Based on physiological pressure curves from the Aorta as well as the morphometric characteristics of the dataset (inlet and outlet radii), realistic approximations of total coronary VBF can be made. To calcuate vascular resistances, the BC makes use of several volume scaling laws [35, 36, 97] that have been validated on multiple datasets from different organs and species. The sensitivity analysis performed to assess the unquantifiable compliance of the vessels embedded in the myocardial tissue is in convincing agreement with the results from [34] where a lumped parameter model has been used to estimate arterial and venous resistances.

The framework derived here allows the application of several different BCs in the actual CFD simulations. It is possible to directly apply either volume flow or pressure curves at the model inlet and outlets. Moreover, in an even more sophisticated approach, also making use of the calculated capillary pressure curves, the downstream vasculature behind the model outlets can be integrated in the simulations. As a consequence, this approach takes account of the actual flow resistances of the used 3D model itself. The consideration of varying capillary pressure curves in dependence of the location of the model outlets between endo- and epicardium represents one of the major adaptations of the solver as it is presented here. In [25] a similar resistance BC is used, however, only including the downstream vascular resistances not including capillary pressure at all. On the other hand, in [34], intramyocardial pressure is considered using the slightly simplified assumption of the same pressure working on the whole downstream microvasculature (equal to LV pressure). This is adequate for the cardiovascular geometries that are simulated in [34] or Chapter 4 of this thesis; however, in the case of highly detailed 3D models as in Chapter 6 the approach presented here also allows consideration of different outlet locations with respect to their depth within the myocardium.

3.3 Mesh Creation with *cfMesh* and *ANSYS ICEM*

3.3.1 Introduction

As mentioned in Section 2 CFD-simulations require discretization of the considered 3D volume into small grid cells. The smaller the cells, the higher is the numerical accuracy of the obtained results. However, not only the size of the grid cells plays an important role, their shape is of importance as well. Hexahedral grid cells for one can provide more accurate as well as more stable results than tetrahedral cells [18, 22, 23, 25, 106–108]; however, purely hexahedral mesh creation comes along with much more effort.

In this section, an approach to automatically generate a mainly hexahedral computational grid in vessel geometries is compared to an existing procedure, which allowed creation of purely hexahedral grids, but required massive user intervention.

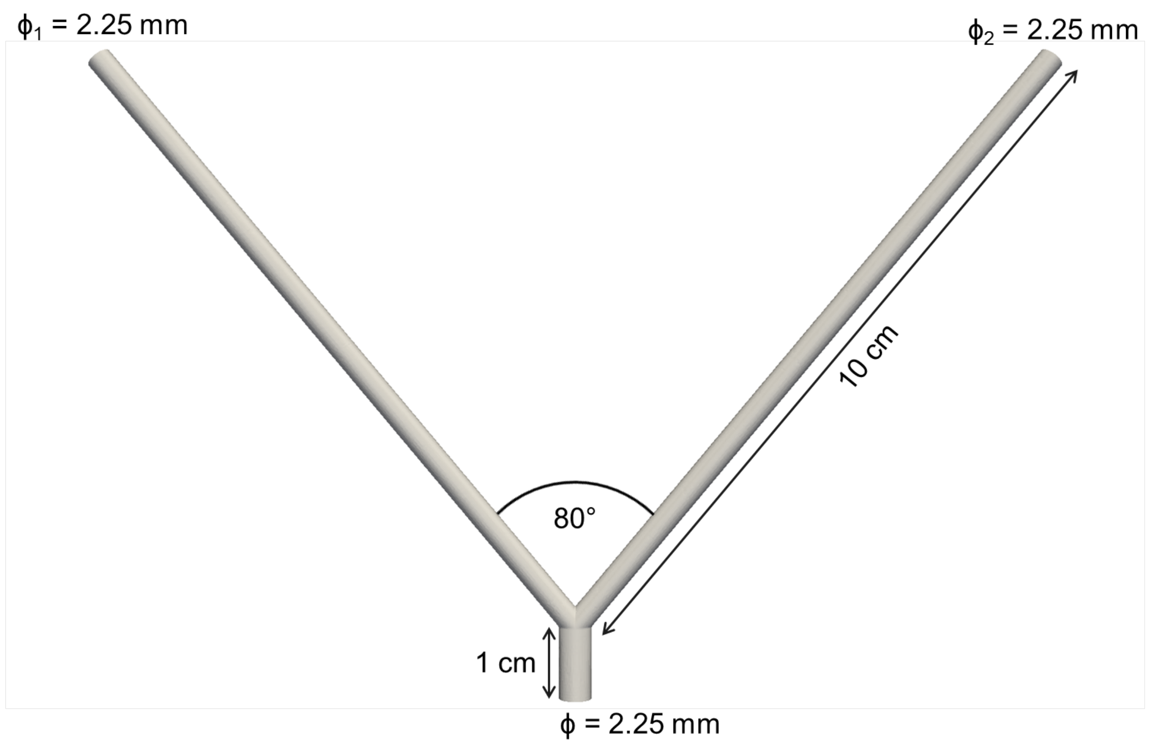


Figure 3.6: Idealized model of LMCA branching into LAD ($\phi1$) and LCX ($\phi2$) [23].

3.3.2 Methods

Model Creation and Computational Framework

The analysis presented here is performed on an idealized geometry of a bifurcation of the LMCA into LAD and LCX. The 3D model of the bifurcating vessels (cf. Fig. 3.6, outlet 1 $\rightarrow$ LAD, outlet 2 $\rightarrow$ LCX) is created with open source software *freeCAD*[5]. The following model parameters are selected analogous to [23]. The radius of the inlet vessel (LMCA) amounts to 2.25 mm and the chosen radii of LAD and LCX are 1.78 mm each, corresponding to average values from literature data for normal right dominant males ($\sim$ 85 %)[109]. The increase of the cross-sectional area thus amounts to a factor of 1.25, well within the reported range in [110]. According to [111] the angle between the two bifurcating vessels is set to 80°. The length of the inlet segment is 10 mm and the distance between the bifurcation and the outlets is 100 mm.

The solution of the Navier-Stokes equations for blood flow and the advection diffusion equation for transport in the geometry is performed with the open-source software package OpenFOAM (OpenFOAM-2.3.0, OpenCFD Ltd., ESI Group, Bracknell, United Kingdom) with the finite volume method. The computations are performed in parallel on eight cores of a designated compute server (2x Intel©Xeon E5-2630V3, 8-Core, 16 Threads, 512 GB RAM).

Blood Flow Simulation

The Navier-Stokes equations for blood flow (Eq. 2.16) are solved for two cardiac cycles, which is sufficient for the system to reach periodicity. This is controlled by comparison of the outlet flow values of the two subsequent cycles. The simulations are started with a time step size of 0.1 ms, and the Courant number (cf. Section 2) is chosen such that the time step

[5]www.freecadweb.org

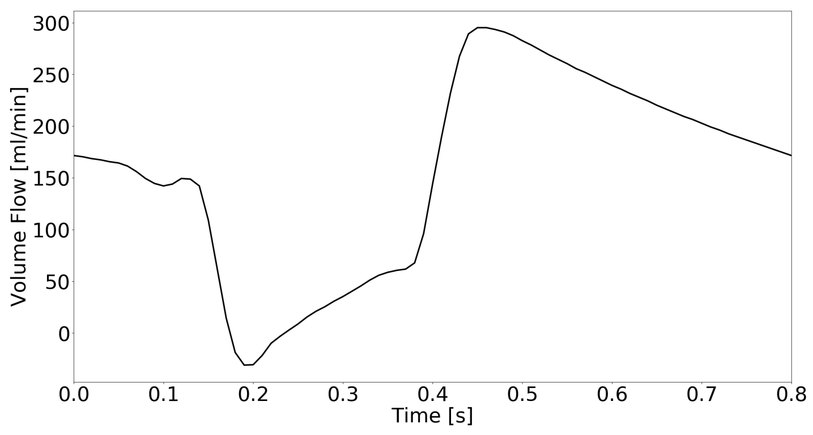


Figure 3.7: Volume flow at the inlet of an idealized model of the LMCA with two outlets (LAD, LCX). The curve is computed according to the BC as described in Section 3.2 (cf. Section A.2).

size adapts to values between 0.1 and 1 ms during the entire simulation. The convective term is discretized by use of a linear upwind interpolation, which is recommended as a stable and accurate discretization scheme [112, 113]. Discretization of time is done using an implicit Euler scheme.

At the model inlet the volume flow profile as depicted in Fig. 3.7 is assigned. The curve is computed with the `C++`-program *outletFlow.cpp*, presented in Appendix A.2 and total mean VBF into the LMCA amounts to 155 ml/min. The time curves used for aortic and ventricular pressure are taken from [101]. At the vessel walls a "no-slip" BC is assumed. At the outlets the pressure is calculated by Eq. 3.3, based on the outlet volume flow from the previous iteration. This is implemented in a custom solver, which is described in detail in Section A.3. The resulting physical fields (pressure p, velocity U, diffusion coefficient D, volume flow ϕ, etc.) are stored on disk for the entire cardiac cycle with a time resolution of 0.01 s.

General Assumptions

Blood viscosity is modeled using Ballyk's generalized power law [114]

$$\mu(\dot\gamma) = \lambda(\dot\gamma)\dot\gamma^{n(\dot\gamma-1)}, \text{ where} \tag{3.9}$$

$$\lambda(\dot\gamma) = \mu_\infty + \mu \exp\left[-\left(1+\frac{\dot\gamma}{a}\right)\exp\left(-\frac{b}{\dot\gamma}\right)\right] \text{ and} \tag{3.10}$$

$$n(\dot\gamma) = n_\infty + n \exp\left[-\left(1+\frac{\dot\gamma}{c}\right)\exp\left(-\frac{d}{\dot\gamma}\right)\right]. \tag{3.11}$$

μ denotes the viscosity and $\dot\gamma$ the shear rate and $\mu_\infty = 0.0035\,\mathrm{Pa\,s}$, $\mu = 0.025\,\mathrm{Pa\,s}$, $n_\infty = 1.0$, $n = 0.45$, $a = 50$, $b = 3$, $c = 50$ and $d = 50$.

In order to incorporate turbulent blood flow in the simulations, the Large Eddy Simulation (LES) approach, *Dynamic Smagorinsky* is chosen, in which the Smagorinsky coefficients are calculated dynamically from the smallest resolved scales [115–118].

The diffusion coefficient of the CA is chosen as $D_{\mathrm{CA}} = 2.92 \cdot 10^{-10}\,\mathrm{m^2s^{-1}}$ from [119]. According to the Einstein-Stokes equation, the diffusion coefficient is dependent on the fluid's viscosity μ

$$D = \frac{k_{\mathrm{B}}\,T}{6\pi\,\mu\,R}, \tag{3.12}$$

where k_B is the Boltzmann constant, T the temperature and R the radius of the diffusing particle. In a non-Newtonian fluid, where the viscosity is spatially dependent ($\mu \to \mu(r)$), this means the diffusion coefficient varies locally. In the simulations this is taken account of by

$$D(r) = \frac{D_\mathrm{CA}\,\mu_0}{\mu(r)}, \tag{3.13}$$

where $\mu_0 = 0.0046\,\mathrm{Pa\,s}$[120] for blood viscosity at the characteristic shear rate.

CA Transport Simulation

By repeatedly reading in the stored volume flow and diffusion coefficient fields ϕ and D, the advection-diffusion equation is solved for transport in the geometry. This is done with a second custom solver that is described in detail in Section A.3. By use of a sub-time stepping technique (cf. Section 3.5), transport of CA through the geometry is computed with an effective time step size of 0.1 ms. Discretization of the convection term and time is performed analogous to the solution of the Navier-Stokes equation. As inlet boundary condition a gamma-variate function is used $C_\mathrm{CA}(t) = a(t-t_0)^b \mathrm{e}^{-c(t-t_0)}$, with $a = 1.013 \cdot 10^{-4}, b = 2.142, c = 0.454\mathrm{s}^{-1}$. According to [22, 121] this function represents a good approximation of the CA concentration time curve in the LV. The diffusion coefficient is set to $D = 2.98 \cdot 10^{-10}\mathrm{m}^2\mathrm{s}^{-1}$ for Gd-DOTA [119]. In order to make sure that CA has largely left the geometry at the end of the transport simulations, this is performed over 50 cardiac cycles. The concentration time curves obtained at the model outlets then represent the real dispersed $AIF_\mathrm{disp}(t)$ in the region of interest to be used for MBF quantification by use of an adequate perfusion model.

Used Meshing Softwares

Previously, proprietary software *ICEM-CFD* by *ANSYS* has been used for purely hexahedral meshing of cardiovascular models[6]. This software allows orientation of grid cells with the expected flow direction. However, it requires a high degree of user intervention and is extremely time-consuming before a computational grid of sufficient quality can be created. On the CD that is handed in with this thesis a manual of 44 pages for computational grid creation with ICEM is attached, which gives a glimpse of the required steps. Depending on the model's complexity, this can take days to weeks.

The open source software package *cfMesh*[7] by *creativeFields* (London, United Kingdom) is a meshing library particularly built for OpenFOAM, the CFD software with which all computations in this thesis have been performed. Requiring solely prior naming of vessel wall, inlet and outlets in the cardiovascular model file of type *stl*[8], this software automatically creates high quality, mainly hexahedral computational grids ($< 1\,\%$ cells of non-hexahedral type), which are not aligned with the predominant flow direction. Depending on the level of detail of the cardiovascular geometry this automatic meshing process takes between 1 min and several hours. Mesh creation on the geometry depicted in Fig. 3.6 takes $\sim$ 10 min. The mesh created with *ICEM* consists of 1 367 722 grid cells whereas the mesh automatically generated with *cfMesh* possesses a total of 1 471 454 grid cells. In Table 3.2 several mesh parameters of the analyzed case are listed.[9]

[6]As described in Chapter 2 hexahedral grid cells are an optimal choice for CFD transport simulations.
[7]cfmesh.com.
[8]Abbreviation for "stereolithography", a file format for Computer-aided design (CAD) software.
[9]Full *OpenFOAM-checkMesh*-logs can be accessed on CD, which is submitted with this thesis.

Table 3.2: Grid data, *ICEM* vs *cfMesh*.

	Total # of cells	# of faces outlet 1	# of faces outlet 2
ICEM	1 367 722	1 208	1 260
cfMesh	1 471 454 (> 99% hex)	940	1 214

Both grids are of similar size. Due to the oblique angle between branch 2 and the general orientation of the grid cells, the number of faces on outlet 1 and outlet 2 in *cfMesh*-model differs. Please also see Fig. 3.8.

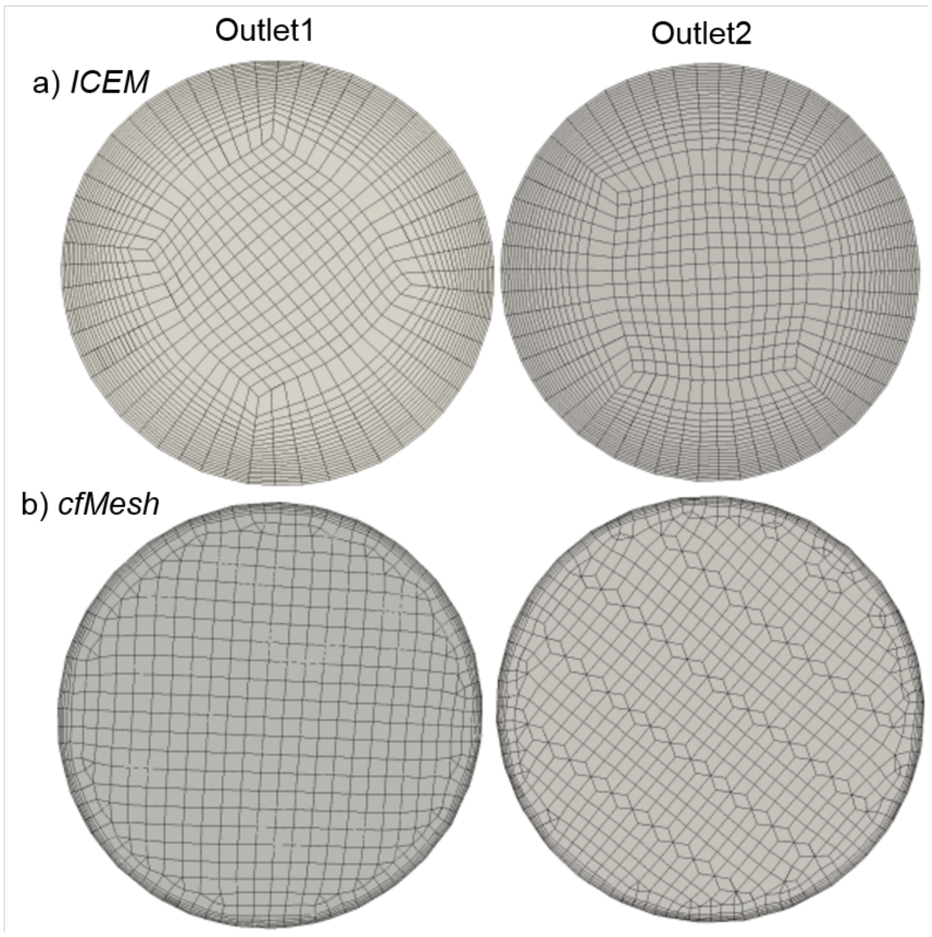


Figure 3.8: Orientation of grid cells in LAD (outlet 1) and LCX (outlet 2) of the simplified LMCA model. Panel a) In *ICEM*, by manual intervention the orientation of the grid cells can be aligned to the presumed flow direction in both branches. Panel b) While meshing with *cfMesh*, the principal orientation of the mesh is chosen such that one branch (outlet 1) has cells aligned with the assumed flow direction and the obliqueness between the flow direction and the cell faces is maximized in the other (outlet 2). Manual eshing with *ICEM* in this rather simple geometry takes a several hours, whereas the automatic meshing procedure with *cfMesh* takes ~ 7 minutes.

Both the mesh created with *ICEM* as well as the mesh created with *cfMesh* are refined at the vessel walls (cf. Fig. 3.8) and the grid sizes are chosen similar to [23]. [10]

Comparison of the Mesh Structures

With OpenFOAM's built-in command `checkMesh`, quality parameters of the meshes can be assessed and a warning or even an error is given, for meshes not fulfilling minimum requirements on grid quality. The most important parameters for stable and accurate execution of OpenFOAM simulations are *mesh non-orthogonality*, which should be below 70° and *skewness*, which should not exceed a value of 4 for the entire grid [112, 113]. *Non-orthogonality* measures the angle between the line connecting two cell centers and the normal of their common face. *Skewness* measures the distance between the intersection of the line connecting two cell centers with their common face and the center of that face.[11] The *cfMesh*-grid exhibits a maximum *mesh non-orthogonality* of 57.6° and a maximum *skewness* of 0.7. On the *ICEM*-grid these parameters amount to 55.8° and 1.6. Both meshes fulfill all requirements on a computational grid for simulations with OpenFOAM.

The principal orientation of the hexahedral cells in the *cfMesh*-grid is chosen such that in one branch (LAD) the grid cells are aligned with the predominant flow direction and in the other (LCX) the angle between cell faces and flow direction is maximally oblique. This is depicted in Fig. 3.8. With this setup, information about the influence of grid orientation on flow and transport can already be obtained by analysis of this grid alone. Nonetheless, comparison between both mesh creation methodologies is performed.

As it is uncertain which of the meshing formalisms gives the "true" solution, the following approach is used to compare the results of the blood flow simulations. The two branches of the 3D geometry being of identical size and length, the simulated values should also be identical for both outlets on both grids. With regard to VBF this is analyzed using the maximum deviation between VBF into the outlets ΔVBF_{max} as well as the difference of total blood volume (BV) reaching each outlet per cardiac cycle, ΔBV. For the analysis, these reference quantities are defined as follows:

$$\Delta VBF^{i,j}_{\max} = \max\left(VBF^i(t) - VBF^j(t)\right), \text{ with } i,j = \begin{cases} cfMesh1,\ cfMesh2 \\ ICEM1,\ ICEM2 \\ cfMesh1,\ ICEM1 \\ cfMesh2,\ ICEM2 \end{cases} \tag{3.14}$$

Here $cfMesh1$, $cfMesh2$, $ICEM1$ and $ICEM2$ denote the outlets 1 and 2 in the $cfMesh$ and $ICEM$ grid, respectively. The quantity ΔVBF_{max} is defined analogously:

$$\Delta BV^{i,j} = \left(BV^i - BV^j\right), \text{ with } i,j \text{ as in Eq. 3.14} \tag{3.15}$$

The parameters are calculated by $BV^i = \int_0^T F_i(t)\mathrm{d}t \approx \sum_k F_i(t_k)\Delta t$, with T the duration of one cardiac cycle. In order to mediate a better impression of the quantities defined in Eqs. 3.14 and 3.15, they are also calculated relative to half mean total VBF and BV entering the whole geometry.

For the transport simulations the analysis is performed using the parameters defined in Eqs. 2.92 and 2.93, i.e., the standard deviation and the temporal mean of the CA concentration time curves at the model outlets. Moreover, the integrated amount Q_{tot} of contrast

[10]Nonetheless, a mesh convergence study is performed to define an ideal mesh and cell size. This is presented in Section 3.4.

[11]https://openfoamwiki.net/index.php/CheckMesh, August 17, 2018.

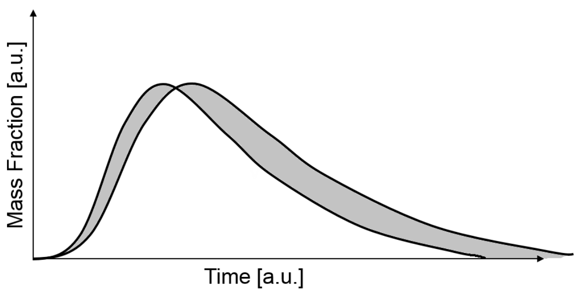


Figure 3.9: Schematic illustration of the area between the CA concentration time curves.

agent leaving the model through the considered outlet is also used to assess the validity of the two meshing procedures. This quantity is computed as

$$Q_{tot} = \int c(t) \cdot F(t)\mathrm{d}t \cdot \rho = \sum_i c(t_i) \cdot F(t_i)\Delta t \cdot \rho, \tag{3.16}$$

where $c(t)$ denotes the mass fraction and $F(t)$ the volume flow through the outlet at times t, t_i. For this calculation the blood density ρ is assumed constant at $\rho = 1060\mathrm{kg/m}^3$ [122] and unchanged by the present tracer.

Finally, the area between the CA concentration time curves at the model outlets (cf. Fig. 3.9) is used as an additional parameter to compare the results from the transport simulations on the grids created by the two meshing softwares *cfMesh* and *ICEM*. The shaded area in Fig. 3.9 is approximated by

$$\Delta Area^{i,j} = \mathrm{abs}\left(c^i(t) - c^j(t)\right)\Delta t, \text{ with } i,j = \begin{cases} cfMesh1,\ cfMesh2 \\ ICEM1,\ ICEM2 \\ cfMesh1,\ ICEM1 \\ cfMesh2,\ ICEM2 \end{cases} \tag{3.17}$$

Here “abs” denotes the absolute value of the difference in CA concentration between outlets i and j.

Estimation of Perfusion Quantification Errors

To complete the analysis the obtained concentration time curves are used within a tissue perfusion model *MMID4* to allow for an estimation of MBF-quantification due to bolus dispersion. Since these simulations are not based on true MRI perfusion measurements, the workflow depicted in Fig. 3.10 is used to make an approximation. The estimation is done in a two-step-procedure.

After the CFD-simulations with $\mathrm{AIF}_{\mathrm{LV}}$ as input boundary condiction are finished and the dispersed $\mathrm{AIF}_{\mathrm{disp}}$ are obtained (on the left in Fig. 3.10), these concentration time curves are used as real AIF within *MMID4*. In combination with a generic value of $\mathrm{MBF}_{\mathrm{LV}}$. this yields the myocardial tissue concentration curve $\mathrm{C}_{\mathrm{Myo}}$ (bottom in Fig. 3.10). Subsequently, together with $\mathrm{AIF}_{\mathrm{LV}}$ this is inserted into *MMID4* (bottom in Fig. 3.10) to compute $\mathrm{MBF}_{\mathrm{Fit}}$ in the conventional way. The difference

$$\Delta\mathrm{MBF} = \frac{\mathrm{MBF}_{\mathrm{Fit}} - \mathrm{MBF}_{\mathrm{Gen}}}{\mathrm{MBF}_{\mathrm{Gen}}} \tag{3.18}$$

is then used to estimate the error in MBF quantification with MRI perfusion measurements.

The parameters used to generate $\mathrm{C}_{\mathrm{Myo}}$ with *MMID4* can be set within the software *JSim* [88] by National Simulation Resource (University of Washington, Seattle, USA) and

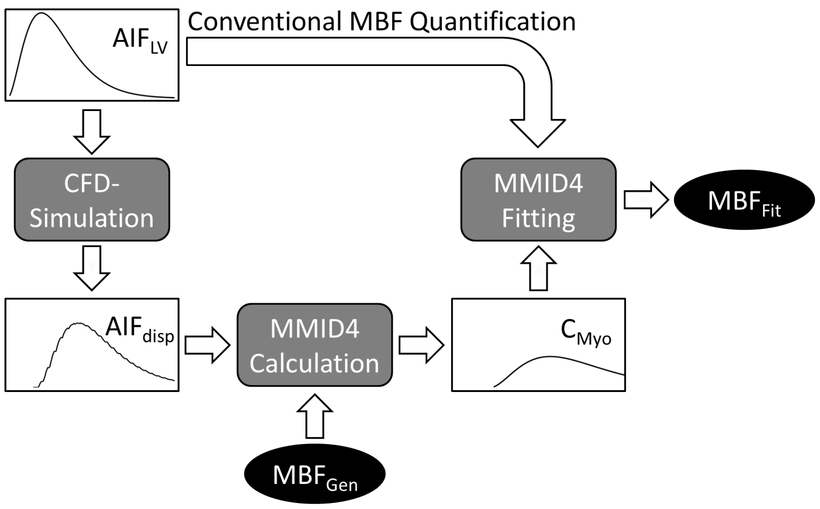


Figure 3.10: Workflow to estimate ΔMBF. AIF$_{\text{LV}}$ from the left ventricle is used as input boundary condition in CFD simulations and delivers the dispersed (real) AIF$_{\text{disp}}$ in the providing vessel. Combined with a generic MBF$_{\text{Gen}}$ and *MMID4*, this yields a myocardial tissue concentration curve C$_{\text{Myo}}$. Considered as the myocardial signal in an MRI perfusion measurement, this allows calculation of MBF$_{\text{Fit}}$ with AIF$_{\text{LV}}$ and fitting of *MMID4* in the conventional way. Comparison of MBF$_{\text{Fit}}$ and MBF$_{\text{Gen}}$ is then used to estimate ΔMBF.

are described in the following. A slightly right-skewed (skewness = 0.3) lagged function [123] is used as density function, with a relative dispersion RD = 0.55 [124].

The required hemodynamic parameters are defined in accordance with typical literature data [8, 18, 22–25, 87, 89, 124, 125]: Plasma volume in the capillaries (V_{p}) is set to 0.04 ml/g, relative dispersion of all vessels (arteries, arterioles, veins, venules) to $\text{RD}_{\text{art,artl,ven,venl}} = 0.48$, arterial and venous volume are set to V_{art}, $V_{\text{ven}} = 0.02$ ml/g and BV in arterioles and venules to V_{artl}, $V_{\text{venl}} = 0.03$ ml/g. Axial diffusion in the capillaries is chosen as $v_{\text{Dp}} = 1 \cdot 10^{-5}$ cm^2/s, the permeability surface of the capillaries (endothelial gap) $v_{\text{PSg}} = 1$ ml/min/g and the total distribution volume in interstitial space $v_{\text{Visfp}} = 0.35$ ml/g.

The methodic analyses in this chapter are all performed for rest state only, plasma flow (i.e., MBF) is thus set to $F_{\text{p}} = 1$ ml/min/g. For analyses of the hyperemic (stress) state, this value (among others) needs to be adapted, which is pointed out, where applicable in this thesis. The applied tissue perfusion model *MMID4* allows consideration of up to 20 different pathways to account for tissue heterogeneity. The parameter $\text{Npath}_{\text{int}}$ is thus set to 20 with $\text{Nseg}_{\text{int}} = 30$, the number of axial segments.

The following calculation of MBF (on the right in Fig. 3.10) with the generated tissue concentration curve C$_{\text{Myo}}$ is performed with the fitting algorithm SENSOP [126] within *JSim*, an application of the well-known Levenberg-Marquardt algorithm [127, 128]. Two different fitting procedures are performed in this chapter, where one respectively four parameters are left free to vary within predefined ranges.

Parameter fitting boundaries for MBF are set to 0 and 7 ml/g/min in both cases. When fitting with four-parameters, additionally permeability surface, plasma volume and delay between inflow of contrast agent in the LV and myocardial tissue are varied between v_{PSg}: 0.25-8 ml/min/g, V_{p}: 0.05-0.09 ml/g and delay: -1-3 s.[12] Starting values for the fitting procedure are chosen similar but not equal to the values set for the generation of C$_{\text{Myo}}$. All other parameters are kept constant as above.

[12] *JSim* does not allow direct access to time delay, this is acchieved by variation of Vtube_{i} between 0-0.36 ml/g with $\text{RDtube}_{\text{i}} = 0$. For generation of C$_{\text{Myo}}$, $\text{Vtube}_{\text{i}} = 0.01$ ml/g.

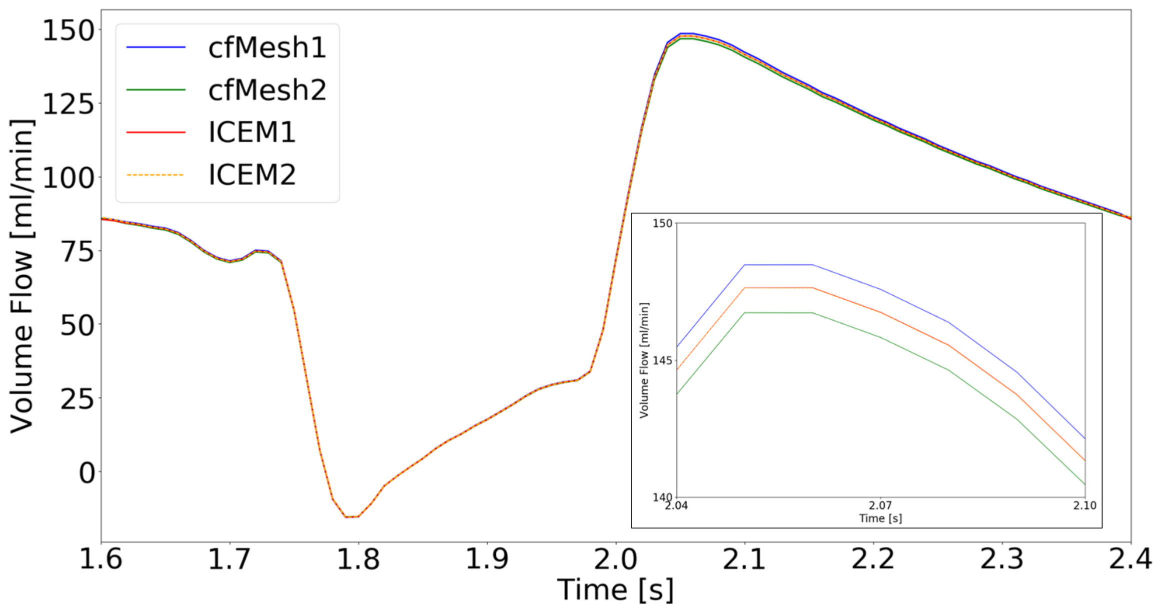


Figure 3.11: Outlet volume flows *cfMesh* vs *ICEM*. In both grids outlet 1, which is divided into fewer faces than outlet 2, shows slightly higher volume flow.

3.3.3 Results

Analysis of VBF

In Fig. 3.11 the obtained volume flow curves at the two model outlets for both grids are depicted. The computation time for the simulations on both meshes is ~ 6.5 d.

The curves in the mesh created with *cfMesh* deviate stronger from one another (blue, green) than the curves stemming from the *ICEM*-grid, which almost perfectly lie on top of each other (red, orange). In both grids outlet 1 that consists of less cell faces in both cases shows both higher maximum and mean volume flow. Mean volume flow for outlets 1 and 2 in *cfMesh* amounts to ~ 78 ml/min and ~ 77.2 ml/min. On the *ICEM*-grid this is distributed very evenly with ~ 77.59 and ~ 77.61 ml/min for outlets 1 and 2 respectively.

The maximum differences between the two curves for outlet 1 and outlet 2 in the meshes created with *cfMesh* and *ICEM* are listed in Table 3.3 together with the deviations obtained between the grids (cf. Eq. 3.14). In addition, the table shows the deviations in total BV at each outlet leaving the geometry comparing both outlets within each grid as well as the grids with each other (cf. Eq. 3.15).

Analysis of CA Transport

The transport simulations running from $t = 2.4$ to 50 s and the obtained concentration time curves at the two model outlets are depicted for both meshes in Fig. 3.12 for times between $t = 0$ to 25 s. Similar to the previous Navier-Stokes simulation, the curves in the mesh created with *cfMesh* deviate stronger from one another (blue, green) than the curves stemming from the *ICEM*-grid (red, orange), which show no difference at all. The concentration time curves at the two outlets of the *cfMesh*-grid show different behaviour as it seems the curve *cfMesh1* exhibits a lag compared to *cfMesh2*. At outlet 1 with fewer faces the concentration time curve overall shows faster feedback to flow increase in systole and decrease in diastole than outlet 2 (top right box in Fig. 3.12). The general shape of the blue curve of outlet *cfMesh1* resembling more the curves from the *ICEM*-grid suggests that it is closer to the “true” situation than *cfMesh2*.

Table 3.3: Deviations of VBF and total BV between outlets and meshes.

	cfMesh1, cfMesh2	*ICEM1, ICEM2*	*cfMesh1, ICEM1*	*cfMesh2, ICEM2*
$\Delta VBF_{\max}$ [ml/min]	1.75	0.68	0.83	0.9
	2.3 %	0.9 %	1.1 %	1.2 %
ΔBV [ml]	0.01	0.0003	0.005	0.005
	0.9 %	0.0003 %	0.47 %	0.51 %

Maximum deviation $\Delta VBF_{\max}$ of and the area between the flow curves into outlets 1 and 2 in the *cfMesh*- and the *ICEM*-grid are shown on the left. The deviations between the same outlets from both meshes are shown on the right. The relative values are scaled by half total mean VBF for $\Delta VBF_{\max}$ and half total BV for ΔBV.

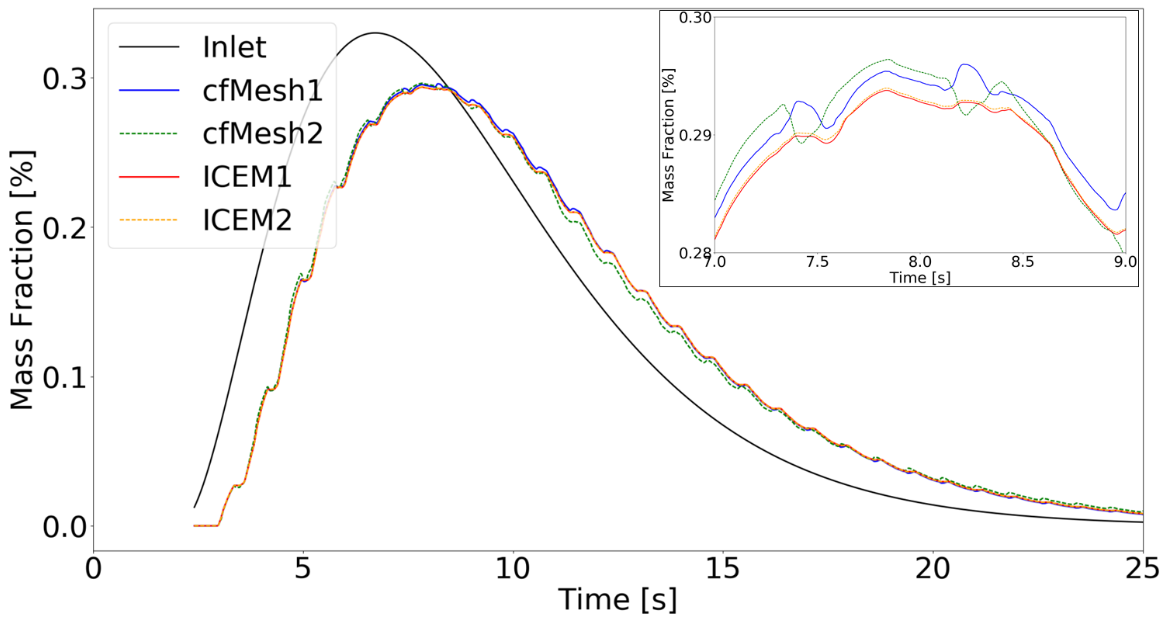


Figure 3.12: Outlet concentration time curves *cfMesh* vs *ICEM*. All outlet concentration time curves are dispersed in comparison to the gamma-variate function at the model inlet (black, cf. Section 3.3.2). In the *cfMesh*-grid outlet 2 with more faces appears to show a faster reaction to increased and decreased flow in diastole and systole. The same but much less pronounced behaviour is observed for the *ICEM*-outlets (box, top right). The steps in the outlet concentration time curves reflect the pulsatility of the underlying blood flow simulation.

Table 3.4: Mean bolus arrival times $\overline{T}$, standard deviation SD_{AIF} of concentration time curves and total tracer amount per outlet for varying mesh resolution.

Outlet	*cfMesh1*	*cfMesh2*	*ICEM1*	*ICEM2*
$\overline{T}$ [s]	8.16	8.20	8.20	8.20
SD_{AIF} [s]	4.50	4.65	4.55	4.55
Q_{tot} [mg]	38.20	37.74	37.96	37.97
	50.3 %	49.7 %	50 %	50 %

The obtained values show good accordance between the outlets in each mesh, with stronger conformity in the *ICEM*-mesh.

Table 3.5: Deviations of concentration time curves between outlets and meshes.

	cfMesh1, cfMesh2	*ICEM1, ICEM2*	*cfMesh1, ICEM1*	*cfMesh2, ICEM2*
$\Delta Area$	$6 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$6 \cdot 10^{-4}$
	2.4 %	0.1 %	0.8 %	2.1 %

As in Table 3.4 the *ICEM*-grid shows better accordance; however, the deviation within the *cfMesh*-grid as well as between the meshes is small.

In Table 3.4 the obtained values for $\overline{T}$ (Eq. 2.93) and SD_{AIF} (Eq. 2.92) are listed. It also contains the integrated amount of contrast agent leaving the model through the considered outlet (cf. Eq. 3.16). Ideally, the total amount of tracer flowing into the LMCA at the model inlet over the whole simulation of $\sim$ 75.92 mg should divide equally into both outlets on both grids. The values of $\overline{T}$ and SD_{AIF} are in a similar range, deviating stronger within the *cfMesh*-grid and showing very good accordance in the *ICEM*-grid.

The total amount of tracer reaching the outlets again shows slight differences for inner-mesh comparison. Where in the *ICEM*-grid, tracer is almost equally distributed (50 % each outlet), in the *cfMesh*-grid accordance is slightly poorer. The values of the calculated areas between the CA concentration time curves are listed in Table 3.5. These numbers also reflect the higher numerical accuracy in the *ICEM*-grid; however, deviations in the *cfMesh*-grid are small, as well.

Perfusion Quantification Errors

Subsequently, based on these dispersed arterial input functions, MBF-values are computed by fitting as described in Section 3.3.2 to further compare these results. In Fig. 3.13 the obtained values for ΔMBF are depicted in a bar chart for both one - and four-parameter fitting.

For one-parameter fitting ΔMBF amounts to $(-16.4 \pm 0.5)\%$ at outlet 1 and $(-16.9 \pm 0.5)\%$ at outlet 2 on the *cfMesh*-grid, and for four-parameter fitting to $(-11.9 \pm 0.2)\%$ and $(-12.1 \pm 0.2)\%$ respectively. On the grid generated with *ICEM* ΔMBF equals $(-17.1 \pm 0.5)\%$ at both outlets for one-parameter fitting and for four-parameter fitting $(-12.5 \pm 0.2)\%$ and $(-12.4 \pm 0.2)\%$ at outlet 1 and 2, respectively. For both fitting procedures MBF is thus underestimated. On both meshes stronger MBF underestimation is expected for one-parameter fitting.

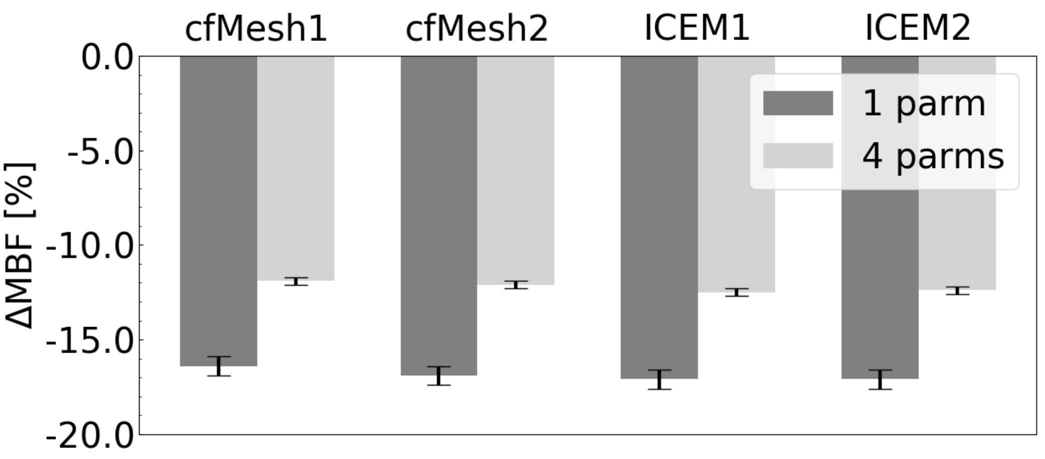


Figure 3.13: ΔMBF for *cfMesh* and *ICEM*. cfMesh1/2 stands for outlet 1/2 on the *cfMesh*-grid and accordingly ICEM1/2 on the *ICEM*-grid. Underestimation of MBF reaches ∼ 17 % for one-parameter and ∼ 12 % for four-parameter estimation.

The results for one- and four-parameter fitting do not lie within each other's error margins on both grids. However, comparing the obtained results for one-parameter fitting, they lie well within the error margins for each mesh-type separately (comparison of outlet 1 and 2 on *cfMesh* and *ICEM* each) but also between the meshes (comparison of outlet 1, 2 from one mesh with outlet 1, 2 from the other mesh). For four-parameter fitting this only applies for intra-mesh comparison and comparison of outlet 2 from both grids, but outlet 1 from *cfMesh* with less cell faces does not overlap with the result from *ICEM*1.

3.3.4 Discussion

The mean flow values obtained in the simulations show stronger deviations between the outlets in the grid generated with *cfMesh*, ∼ 78 and ∼ 77.2 ml/min, wheras the results on the *ICEM*-grid are nearly identical, ∼ 77.61 and ∼ 77.59 ml/min. Considering differences of the simulated flow curves, intra-grid deviations of 0.9 % and 2.3 % for the maximum difference and 0.9 % and 0.0003 % for total flow into the outlets are obtained for the meshes created with *cfMesh* and *ICEM*, respectively. The inter-grid comparison shows deviations of 1.1 % and 1.2 % for maximum difference for outlets 1 and 2, respectively, and ∼ 0.5 % difference in total flow for both outlets.

The obtained concentration time curves show that the total amount of tracer distributes evenly across both outlets in both meshes, 49.7 % to 50.3 % for *cfMesh* and perfectly halving total flow for *ICEM*. Furthermore, the results show preservation of contrast agent on both grids. No contrast agent is "lost" due to numerical inaccuracies, e.g., numerical diffusion. The comparison of mean bolus arrival times at the outlets (8.16 and 8.20 s for *cfMesh* and 8.20 s for both outlets in the *ICEM*-grid) and their standard deviations (4.50 s, 4.65 s and 4.55 s, respectively) also shows small variation within each and between the grids.

The comparison of the areas between the curves again shows the better conformity of the *ICEM*-grid with 0.1 % deviation in contrast to 2.4 % for *cfMesh*. The difference between the grids amounts to 0.8 % and 2.1 % for outlet 1 and 2, respectively. For the *cfMesh*-grid it stands out, that the concentration time curves seem to show qualitatively different behavior. In the top right box in Fig. 3.12, it appears as if the tracer mass fraction at outlet 1 with fewer faces increases in systole and decreases during diastole, as opposed to outlet 2.

The final and most important parameter used for comparison of the two grid types is presented in Fig. 3.13. The values for ΔMBF equal ∼ 17 % for one-parameter and ∼ 12 % for four-parameter estimation.

These results show the power of a high quality grid with grid cells oriented in predominant flow direction produced with the software package *ICEM*. The flow is evenly distributed into both model outlets, and concentration time curves are almost identical. The conformity of the obtained results for the *ICEM*-grid is striking. The results for *cfMesh* show seemingly less accurate results; however, they are also in a physiological range and do not deviate strongly from the results obtained with *ICEM*. Not only do the error margins for the quantity of interest, ΔMBF, overlap within the mesh itself, but also for inter-mesh conformity with the exception of outlet 2 for four-parameter fitting, where error margins are too small. Generally speaking it can be recorded that the differences between the grids presented in tables 3.3, 3.4 and 3.5 do not have a significant influence on the calculated value for ΔMBF.

In the inlaid box in Fig. 3.12 outlet 1 with fewer faces shows an increase in tracer mass fraction during systole and outlet 2 with more faces a simultaneous drop. The reduced resolution of the grid in the branch leading to outlet 1 results in a faster reaction of the outlet concentration time curve to flow changes through the vessel. Outlet 1 simply runs ahead of outlet 2 during the entire simulation. In the end, this results in reduced dispersion (less broadening) of the curve in this branch, which is in accordance with [18, 129] for reduced grid resolution due to numerical diffusion. Similarly to the small effects of pulsatility (compared to constant flow) on contrast agent dispersion [22] this slightly different behaviour of transport to the outlets has a vanishing influence. Overall, the effect of this behaviour is of reduced importance, since its influence on MBF-quantification and the error therein can be neglected, as can be deduced from Fig. 3.13. It is for these reasons, that most simulations presented in this thesis are conducted using the meshing software package *cfMesh*.

Conclusion

With regard to the manual actions (cf. the guide for grid creation with *ICEM* on the enclosed CD) required to produce grids of acceptable quality the slightly higher numerical accuracy of *ICEM* cannot outweigh its heavy disadvantages in mesh creation compared to *cfMesh*. This particularly applies when it comes to the mesh creation in highly detailed vascular models with multiple (tens to hundreds) bifurcations and outlets.

Furthermore, in these complex 3D models it cannot be guaranteed that manual meshing with *ICEM* can even produce meshes with sufficient cell quality everywhere in the domain. Due to vessel branching, tapering and curvature this becomes impossible not mentioning the prohibitively long times required to accomplish such a task - if possible at all.

3.4 Mesh Convergence

3.4.1 Introduction

As described in Section 2 the discretization of the volume, in which the CFD simulation is performed, is of crucial importance for the numerical accuracy of the results. However, not only the shape of the grid cells plays a role in this regard (cf. Section 3.3, but also the resolution of the grid, namely the number of grid cells, in which the 3D volume is discretized must be considered.

To investigate this, in this section a mesh convergence study is presented in order to reduce numerical inaccuracies due to insufficient resolution of the computational grid within the 3D vessel geometry.

Table 3.6: Grid data of the meshes used to assess mesh convergence.

	Total # of cells	# of faces outlet 1	# of faces outlet 2
Fine	1 471 454	940	1 214
Medium	900 152	716	893
Raw	522 212	513	654

3.4.2 Methods

With the software package *cfMesh*, two additional meshes of reduced resolution are created on the same geometry, which is used in Section 3.3 (cf. Fig. 3.6). Together with the mesh generated with *cfMesh* in Section 3.3, these meshes will be used to assess differences in numerical accuracy due to mesh resolution. In Table 3.6 several mesh parameters are listed. The mesh quality parameters maximum *mesh non-orthogonality* and maximum *skewness* (cf. Section 3.3) amount to 53.3° and 0.9 for the raw, 54.3° and 0.7 for the medium as well as 57.6° and 0.7 for the fine grid. All three meshes fulfill all requirements on a computational grid for simulations with OpenFOAM. Apart from the varying mesh resolution all boundary conditions are chosen identically for the simulations on all three grids to guarantee maximal comparability. These conditions are described in detail in Section 3.3.2. Analogously, the same two-step-procedure to simulate blood flow and CA transport in the geometry as well as the workflow to quantify ΔMBF with *MMID4* (cf. Fig. 3.10) is applied.

3.4.3 Results

Analysis of Blood Flow

In Fig. 3.14 the obtained volume flow curves for both outlets on all three grids are presented. As in Section 3.3 outlet 1 with the lower number of faces shows increased volume flow. The outlet volume flow curves approach each other for higher mesh resolution.

Mean volume flows for outlet 1 amount to $\sim$ 78.3 ml/min, $\sim$ 78.1 ml/min and $\sim$ 78 ml/min for the raw, medium and fine mesh, respectively. The corresponding values for outlet 2 are $\sim$ 76.9 ml/min, $\sim$ 77.1 ml/min and $\sim$ 77.2 ml/min. With increased mesh resolution, the values approach each other with closest accordance in the finest mesh. The same parameters as defined in Section 3.3.3 are used to quantify intra- and inter-mesh deviations of the obtained curves. The values are listed in tables 3.7 a) and b). Both the maximum deviation between the VBF curves as well as the the total outlet BV within the grids decrease from the raw and the medium to the fine grid. Comparison of each outlet of the raw and medium grid to the fine grid shows the same behavior.

Analysis of CA Transport

In Fig. 3.15 the concentration time curves for the transport simulation in all three grids are depicted. Table 3.8 shows the results for $\overline{T}$, SD_{AIF} and Q_{tot} as defined in Section 3.3.3 of the outlet concentration time curves. The mean bolus arrival time $\overline{T}$ almost does not change at all for higher mesh resolution. Similarly, bolus width and total tracer amount per outlet remain is barely influenced by higher discretization.

Perfusion Quantification Errors

Subsequently, based on the dispersed arterial input functions, MBF-values are again computed by fitting as described in Section 3.3.2 to further compare these results. In Fig. 3.16

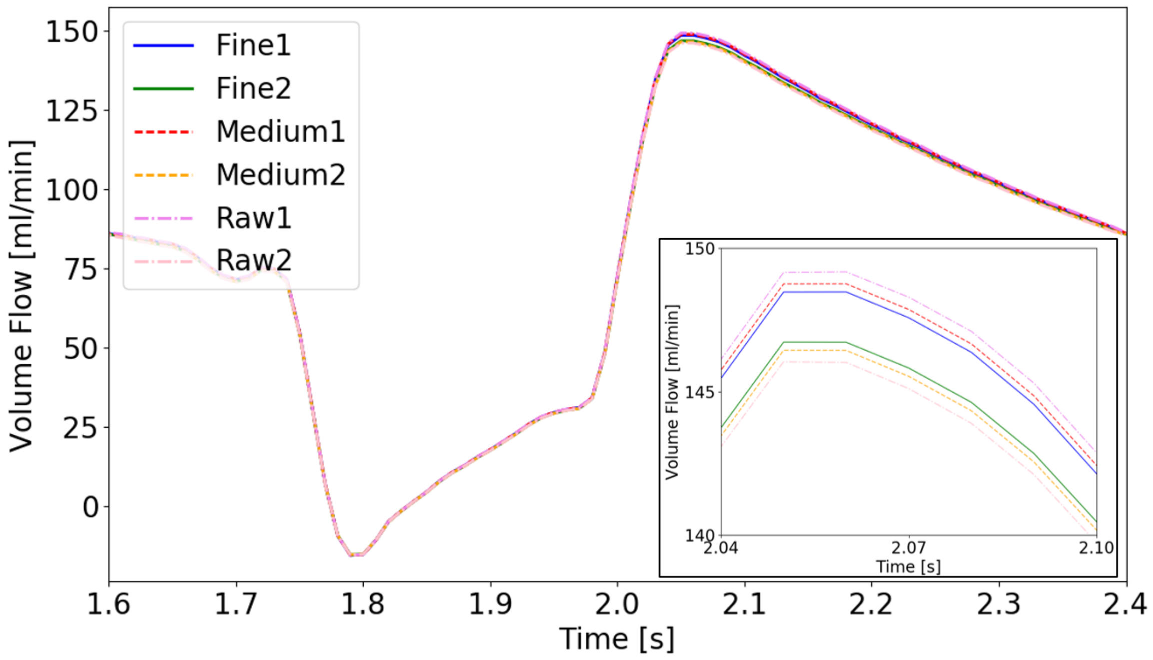


Figure 3.14: Outlet volume flows of all three grids for mesh convergence analysis. In all three grids outlet 1, which is divided into fewer faces than outlet 2, shows slightly higher volume flow. With higher grid resolution the deviation between the volume flow curves decreases. The effect is reduced for higher mesh resolution.

Table 3.7: Deviations of volume flow between outlets and meshes of different resolution.

a)	Raw1, Raw2	Medium1, Medium2	Fine1, Fine2
$\Delta VBF_{\max}$ [ml/min]	3.21	2.33	1.75
	4.1 %	3 %	2.3 %
ΔBV [ml]	0.02	0.014	0.01
	1.8 %	1.3 %	0.9 %

b)	Raw1, Fine1	Raw2, Fine2	Medium1, Fine1	Medium2, Fine2
$\Delta VBF_{\max}$ [ml/min]	0.74	0.74	0.29	0.29
	0.9 %	0.9 %	0.4 %	0.4 %
ΔBV [ml]	0.005	0.005	0.002	0.002
	4.4 %	4.4 %	1.8 %	1.8 %

a) Maximum deviation $\Delta VBF_{\max}$ of and the area between the flow curves into outlets 1 and 2 in the all three grids is shown. b) The same data is shown comparing the raw and medium grid with the fine grid. In both tables it becomes obvious, how increasing resolution improves the results' accuracy. The effect is reduced for higher discretization.

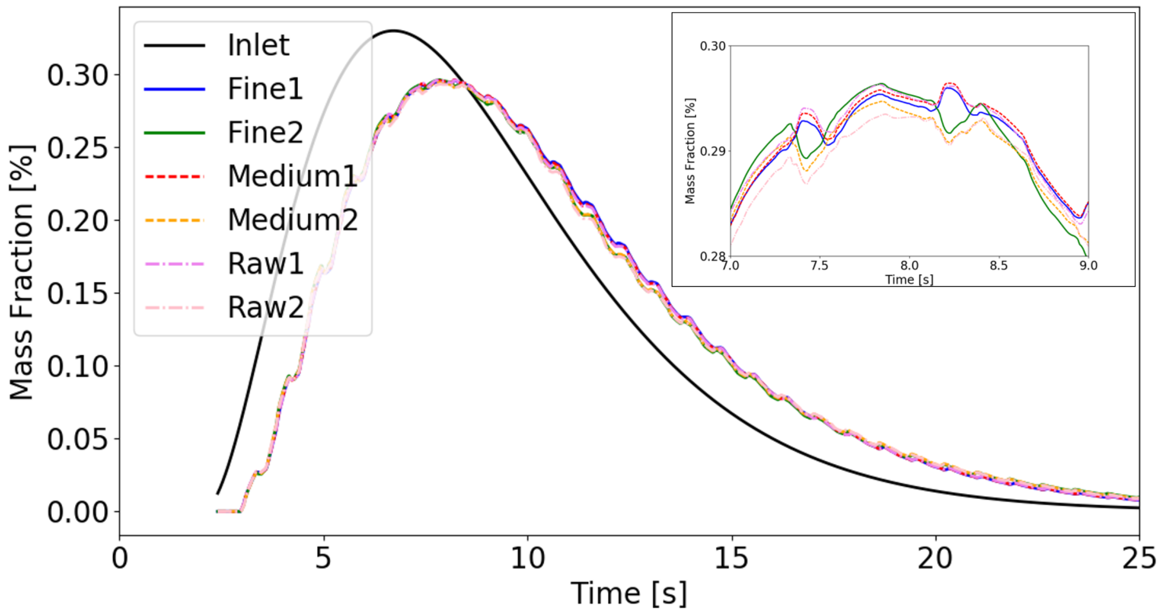


Figure 3.15: Outlet concentration time curves for mesh convergence analysis. All outlet concentration time curves are dispersed in comparison to the gamma-variate function at the model inlet (black, cf. Section 3.3.2). In all three grids outlet 2 with more faces appears to show a faster reaction to increased and decreased flow in diastole and systole. The behaviour is reduced on grids with higher resolution. Overall, the deviation between the outlet curves is decreased for higher discretization.

the obtained values for ΔMBF are depicted in a bar chart for both one- and four-parameter fitting, both outlets and all three meshes.

The results are in good accordance with those obtained in Section 3.3. Stronger MBF underestimation is obtained for one-parameter fitting than for four-parameter fitting. For one-parameter fitting, the results lie within the error margins for all outlets and all three grids. Additionally, the results approach each other with increased mesh resolution. However, for four-parameter fitting, the results only show accordance within the error margins for the highest resolution. As for one-parameter fitting, the values lie closer together for higher mesh discretization. As above, the differences of the areas between the curves is analyzed in tables 3.9 a) and b). Analogously, the numerical accuracy appears to be increased for higher mesh refinement.

Table 3.8: Mean bolus arrival times $\overline{T}$, standard deviation SD_{AIF} of concentration time curves and total tracer amount per outlet for varying mesh resolution.

	Raw1	Raw2	Medium1	Medium2	Fine1	Fine2
$\overline{T}$ [s]	8.17	8.22	8.16	8.21	8.16	8.20
SD_{AIF} [s]	4.54	4.63	4.52	4.63	4.50	4.65
Q_{tot} [mg]	38.34	37.47	38.25	37.73	38.2	37.74
	50.5 %	49.4 %	50.4 %	49.7 %	50.3 %	49.7 %

The obtained values show good accordance between the outlets in each mesh, with increasing conformity for higher mesh refinement.

Table 3.9: Deviations of concentration time curves between outlets and meshes.

a)	Raw1, Raw2	Medium1, Medium2	Fine1, Fine2
$\Delta Area$	$6 \cdot 10^{-4}$	$6 \cdot 10^{-4}$	$7 \cdot 10^{-4}$
	2 %	2 %	2.4 %

b)	Raw1, Fine1	Raw2, Fine2	Medium1, Fine1	Medium2, Fine2
$\Delta Area$	$2 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$8 \cdot 10^{-5}$	$2 \cdot 10^{-4}$
	0.6 %	1 %	0.3 %	0.6 %

a) Intra mesh comparison of the area betwen the outlet concentration time curves. b) Comparison of outlet concentration time curves by area difference between outlet specific curves on all three meshes.

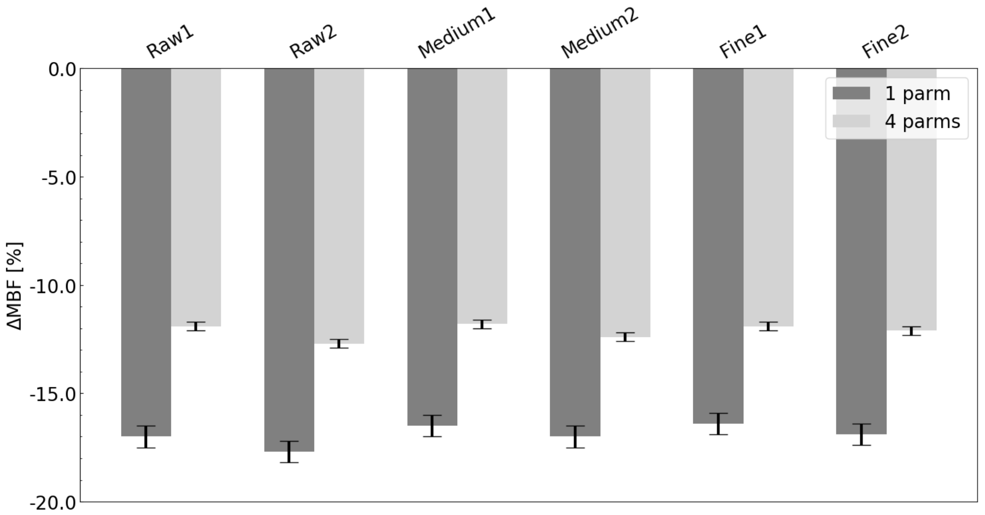


Figure 3.16: ΔMBF for mesh convergence analysis. Raw/Medium/Fine1/2 denote the outlets on the three grids, respectively. Underestimation of MBF generally is less pronounced on both outlets with increasing mesh resolution for both one- and four-parameter fitting. On all three grids, outlet 2 with more cell faces shows stronger MBF-underestimation than outlet 1.

3.4.4 Discussion

Mean volume flow for outlets 1 and 2 amounts to $\sim$ 78.3 ml/min and $\sim$ 76.9 ml/min, $\sim$ 78.1 ml/min and $\sim$ 77.1 ml/min and $\sim$ 78 ml/min and $\sim$ 77.2 ml/min in the raw, medium and fine mesh, respectively. Considering differences of the simulated flow curves, intra-grid deviations of 4.1 %, 3 % and 2.3 % for the maximum difference and 1.8 %, 1.3 % and 0.9 % for total flow into the outlets are obtained for the raw, medium and fine mesh, respectively.

The inter-grid comparison shows deviations of 0.9 % and 0.4 % as maximum differences for both outlets for raw-fine and medium-fine comparison, respectively. The corresponding differences in total flow amount to $\sim$ 4.4 % and $\sim$ 1.8 %, respectively.

The obtained concentration time curves show that the total amount of tracer distributes evenly across both outlets in all three meshes, with improving values from 50.5 %, 49.4 % (raw), 50.4 %, 49. 7% (medium) and 50.3 %, 49.7 % (fine). Furthermore, the results show good preservation of contrast agent on all three grids with the best balance on the finest grid. The comparison of mean bolus arrival times to the outlets, 8.17 and 8.22 s (raw), 8.16 and 8.21 s (medium) and 8.16 and 8.20 s (fine) and their standard deviations (4.54 s and 4.63 s, 4.52 s and 4.63 s as well as 4.50 s and 4.65 s, respectively) do not show much improvement for higher mesh resolutions.

The comparison of the areas between the curves even shows reduced conformity on the finest grid (2.4 % deviation in contrast to 2 % on both raw and medium resolution). However, comparing both outlets of the fine grid with its raw and medium counterpart, deviations decrease (0.6 % to 0.3 % and 1 % to 0.6 % for raw-fine and medium-fine comparison, respectively) as Fig. 3.15 suggests.

Similarly to the analysis in Section 3.3, the outlet concentration time curves in all three grids show the behaviour of opposing increase and decrease of CA concentration time curves at the outlets in dependence of diastole and systole (cf. upper right box in Fig. 3.15). However, it becomes clear that increasing mesh resolution mitigates this effect.

The final and most important parameter used for comparison of the two grid types is presented in Fig. 3.16. The values for ΔMBF are in the range of $\sim 17\,\%$ for one-parameter and of $\sim 12\,\%$ for four-parameter estimation.

These results show the importance of a mesh convergence study to ensure relevance of the obtained results. The flow is evenly distributed into both outlets in all three grids with increased accuracy for higher mesh resolution. The same applies for the distribution of total tracer in both outlets. The mesh convergence study shows improving results for higher mesh densities.

The conformity of the obtained results is not as strong as for the *ICEM*-grid of Section 3.3; however, differences between the three mesh densities are all in an acceptable range (particularly considering the good accordance between the obtained ΔMBF). As described in Section 3.4.3, the error margins for the quantity of interest, ΔMBF, overlap for all three meshes for one-parameter fitting and on the finest grid for four-parameter fitting.

Conclusion

Based on the results of this mesh convergence analysis, the simulations in this work will be performed on a grid of similar resolution as the finest grid in this section. This analysis allows the assumption that further mesh refinement does not significantly improve numerical accuracy. On the contrary, further mesh refinement would unnecessarily increase computation times.[13]

[13] However, special cases, with outlet diameters much smaller than in this idealized geometry require even further mesh refinement.

3.5 Subtimestepping

3.5.1 Introduction

Conducting CFD simulations on highly detailed vascular geometries as presented in this work poses high demands on the computational power available. To keep computation times in an acceptable timeframe, the simulations can be performed on HPC clusters, where high parallelization (i.e., distribution and simultaneous execution of calculations on multiple processors) enables significant speed-up. Ideally, computation times halve, when the number of parallel processes is doubled (cf. Fig. 3.17). However, as a side effect of higher parallelization, the structure of OpenFOAM induces that simultaneously file numbers double. This is due to the fact that in OpenFOAM, for each physical field, time step and processor a separate file is created.

Running applications on HPC clusters imposes certain conditions on the used software package, which includes its file management. Strict limits on file quota apply, which usually allow no more than several ten thousands of files per user. Looking at Fig. 3.17 it becomes clear, that this limit is reached soon.

In this section, an approach to reduce the number of files generated during a CFD-simulation with OpenFOAM is integrated and validated in order to fulfil these restrictions.

3.5.2 Methods

During a computation run with OpenFOAM a separate file in a separate folder is created for all physical fields per computed time step. Similarly, each processor writes calculated fields in a single associated folder. This means file numbers in a computed case multiply with the number of processors N_p, the total number of written time steps N_s and the number of fields necessary to fully describe the physical processes, N_f (e.g., pressure p, velocity U, etc.).

The total number of files for a computed case thus amounts to $N_t = N_p \times N_s \times N_f$. To solve a transport problem as treated in this work, $N_f = 8$. For a time step size of 0.1 ms as is sufficiently small to obtain accurate results for the advection-diffusion equation [18, 23, 25, 108, 129] and a cardiac cycle duration of 0.8 s (cf. Fig. 3.7), the number of time steps for one cycle is 8 000. In order to ensure periodicity of the solution, at least two cycles must be computed, which leads to $N_s = 16\,000$. As explained above, file numbers multiply accordingly as outlined in Fig. 3.17 and soon cluster limits on maximal file numbers apply.

In order to reduce the number of written files by decreasing time resolution while still ensuring correct solution of the advection-diffusion equation a sub-time stepping technique is applied. In OpenFOAM this can be realized by using the integrated `subCycles` class [130, 131]. This means results of the Navier-Stokes equation are stored with reduced time resolution (10 ms vs 0.1 ms) and during the following solution of the advection-diffusion equation the sub-time stepping technique with 1 000 sub-cycles is applied to achieve an effective time step size of 0.1 ms.

The validation of this method is performed on the same geometry (cf. Fig. 3.6) and with the same boundary conditions as described in Section 3.3. The computational grid in the geometry is the one that is also used in Section 3.3 for comparison with *ICEM* and is identical to the finest grid in Section 3.4.[14] Analogously, the obtained results for the concentrations time curves are compared by means of the same parameters as in sections 3.3 and 3.4.

[14]For simplicity, in the following referrals to results from these two sections will only be made with regard to Section 3.3.

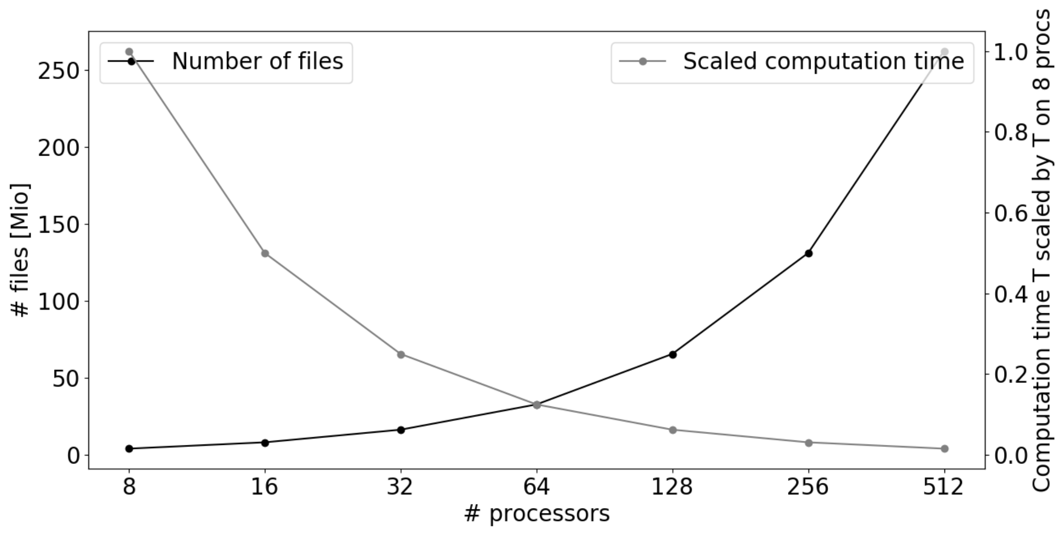


Figure 3.17: Dependence of file number and desired behavior of computation times on parallelization with OpenFOAM. Ideally, computation times decrease, when parallelization is increased. However, with high parallelization the structure of OpenFOAM yields prohibitively high file numbers.

3.5.3 Results

The obtained volume flow curves are identical for both methods, since only the time step size with which the fields are written is varied. The volume flow curves thus correspond to the ones depicted in Fig. 3.11 for the *cfMesh*-grid. In Table 3.10 the results obtained for $\overline{T}$, SD_{AIF} and total tracer per outlet are listed. Deviations between the two methods are presented with higher precision than in Section 3.3 and 3.4 since they are otherwise not detectable. The results are practically identical.

The area between the curves obtained at the model outlets is again computed both for the results obtained for each method as well as between the methods. The values are listed in Table 3.11. The ratio of the area between the outlet concentration time curves to the total area under the inlet concentration time curve with sub-time stepping thus amounts to 2.36 % (result from Section 3.3.3 with higher precision). Without sub-time stepping this

Table 3.10: Mean bolus arrival times $\overline{T}$, standard deviation SD_{AIF} of concentration time curves and total tracer amount per outlet for transport simulation with and without sub-time stepping method.

sub-time stepping:	ON1	ON2	OFF1	OFF2
$\overline{T}$ [s]	8.16	8.20	8.16	8.21
SD_{AIF} [s]	4.50	4.65	4.50	4.65
Q_{tot} [mg]	38.20	37.74	38.20	37.74
	50.3 %	49.7 %	50.3 %	49.7 %

ON1, ON2, OFF1 and OFF2 denote outlets 1 and 2 with the subteimstepping method applied (ON) or not (OFF). Regarding the numerical accuracy of the simulations, the obtained results are practically identical.

Table 3.11: Deviations of concentration time curves between outlets and methods.

	ON1, ON2	OFF1, OFF2	ON1, OFF1	ON2, OFF2
$\Delta Area$	$6 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$6 \cdot 10^{-4}$
	2.36 %	2.37 %	0.07 %	0.1 %

The differences between the outlet concentration time curves of each simulation method are nearly identical. Deviations between the methods are also negligible.

yields 2.37 %. Analogously, between the methods it amounts to 0.07 % and 0.1 % for outlet 1 and 2, respectively.

The MBF-values obtained with these outlet concentration time curves show no significant deviations either. For one-parameter fitting ΔMBF amounts to $(-16.4 \pm 0.5)\%$ at outlet 1 and $(-16.9 \pm 0.5)\%$ at outlet 2 with sub-time stepping and to $(-16.4 \pm 0.5)\%$ at outlet 1 and $(-17.1 \pm 0.5)\%$ at outlet 2 without sub-time stepping.

For four-parameter fitting, the accordance between the methods is even better, with $\Delta\text{MBF} = (-11.9 \pm 0.2)\%$ and $(-12.1 \pm 0.2)\%$ for outlets 1 and 2 in both cases. The results thus lie within the error margins obtained by fitting with *MMID4*.

3.5.4 Discussion

Regarding the good accordance between the two methods, the sub-time stepping technique is deemed not to have adverse effects on numerical accuracy of the simulation. Since it represents a good measure to significantly reduce the number of files written in a simulation run (factor $1/n$ for n sub-cycles), it is applied in this work to enable execution of the simulations on small HPC clusters (parallelization on 16 – 140 processors).

3.6 Scalability Testing

3.6.1 Introduction

As visible in Fig. 3.17, higher parallelization leads to higher file numbers. However, this also enables computational speed-up since computation times are reduced, if processes are executed in parallel on high numbers of processors. In order to run applications on *Supercomputers*, it is required to prove that they scale according to the gray line plotted in Fig. 3.17 even if distributed on thousands to ten thousands of processors. *Supercomputers* are specifically designed for this degree of parallelization.

Verification of this linear scaling behavior is done by executing the same computational task on different numbers of processors (e.g., 128, 256, 512, 1 024,...), i.e., different degrees of parallelization, and comparison of the time required. Regardless of the simultaneously increased file numbers treated in Section 3.5, a scaling study of a custom OpenFOAM solver is performed in this section.

3.6.2 Methods

A 3D model of the LCX (Fig. 3.18) is extracted from a high-resolution cryomicrotome imaging dataset of an ex-vivo pig heart [26, 28, 132, 133]. This is done with a dedicated software package (SimVascular, SimTK[15]).

[15]simvascular.github.io.

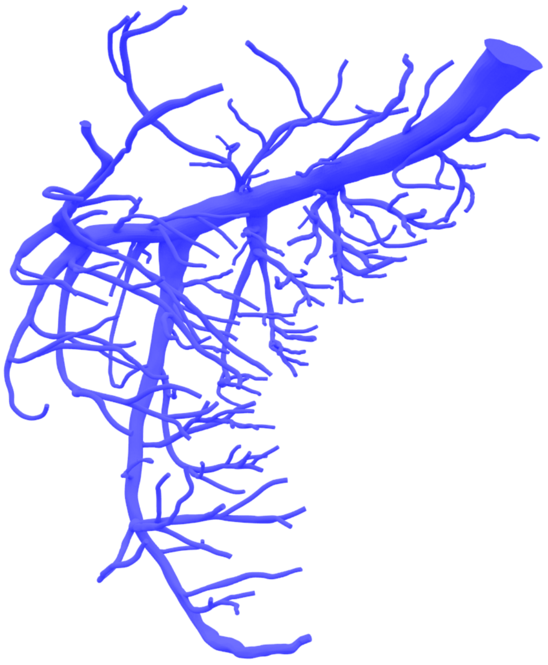


Figure 3.18: 3D model used for scalability testing of OpenFOAM. The model includes vessels of radii down to 160 µm.

As inlet boundary condition time periodic pressure data from [25, 108] is used and at the outlets a resistance boundary condition based on the structured tree [134–136] as implemented in [108] is applied. However, since this chapter addresses scaling behavior, physiological parameters of the simulations will not be discussed in detail.

Subsequently, the model is discretized with a 3D computational grid of predominantly hexahedral type with software *cfMesh* (cf. Section 3.3). Two computational grids consisting of 13 Mio and 46 Mio grid cells are generated. The created computational grids are then decomposed for parallel execution on 128, 256, 512, 1 024 (13 Mio cells) and 512, 1024, 2 048 (46 Mio cells) processors.

The speed-up of the computation runs on n_i cores is then calculated by T_{n_0}/T_{n_i}, where T_{n_i} is the time required to compute a fixed simulated time. n_0 denotes the minimal number of cores used in the scalability test, i.e., 128 for 13 Mio cells and 512 for 46 Mio cells. In order to perform these tests PRACE[16] preparatory access grants admission to *Supercomputers* throughout Europe.

3.6.3 Results

The results presented in Fig. 3.19 stem from simulation runs on the *Supercomputer HazelHen* at HLRS[17] during PRACE preparatory access and an extended HLRS test project.

It becomes obvious how the scaling behavior improves on the grid with more cells, compared to the smaller mesh. Information about the absolute computationan durations and simulated times is given in Table 3.12.

Both meshes scale nearly linearly up to $\sim 20\,000 - 250\,000$ cells/processor. The larger grid shows improved scaling behavior beyond 512 processors.[18] OpenFOAM's better performance

[16]Partnership for advanced computing in Europe, prace-ri.eu.

[17]Höchstleistungs-Rechenzentrum Stuttgart, www.hlrs.de.

[18]Results of additional simulations on several *Supercomputers* are available. However, since this technical aspect is not main subject of this thesis, only this example will be given here.

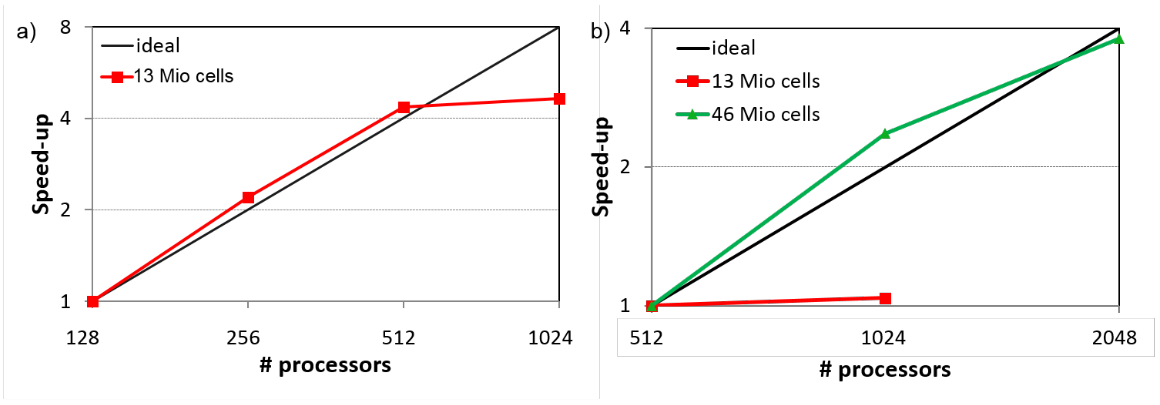


Figure 3.19: Computational speed-up on 13 Mio and 46 Mio grids for different degrees of parallelization. The simulated run time on 13 Mio grid cells is 10 ms and on 46 grid cells 0.8 ms. Corresponding computation times are listed in Table 3.12. Both grids show an even better scaling behavior for the first increase in parallelization, followed by a subsequent drop in performance.

Table 3.12: Absolute compute times on 13 Mio and 46 Mio grid for different degrees of parallelization.

	Computate time [s] and # cells/processor			
# processors	13 Mio (Run time: 10 ms)		46 Mio (Run time: 0.8 ms)	
128	12 087 s	#∼ 100 000	–	
256	5 784 s	#∼ 50 000	–	
512	2 649 s	#∼ 25 000	10 695 s	#∼ 90 000
1024	2 549 s	#∼ 12 500	4 526 s	#∼ 45 000
2048	–		2 819 s	#∼ 22 500

The compute time halves for double parallelization on the 13 Mio grid up to 512 processors. For the 46 Mio cells grid a speed-up is observed beyond 512 grid cells up to 2048 cells. However, minimal computation times are similar for parallelization on 512 and 2048 processors, respectively, where # cells/processor match approximately.

in this range of cells per processor is also underlined by the steeper increase of the scaling behavior as it is visible in Fig. 3.19.

3.6.4 Discussion

The presented results show the ranges in which good scaling behavior close to the ideal line can be obtained for the two analyzed mesh sizes. Both meshes scale linearly up to $\sim 20\,000-250\,000$ cells/processor. Accordingly, the larger grid shows better scaling behavior for parallelization beyond 512 processor, where the smaller grid does not scale well anymore. However, since the larger grid is subdivided into more and thus smaller grid cells, the computational amount increases. As a consequence, no acceleration in computation time is gained in comparison to the coarser grid parallelized on less processors (46 Mio: 2 819 s on 2 048 cores vs 13Mio: 2 649 s on 512 cores). In fact, the number of cells per core is even smaller for the highest parallelization on the 46 Mio grid ($\sim 22\,500$) than on the 13 Mio cells grid ($\sim 25\,000$), but still the computation time is not reduced. Moreover, the simulated time is shorter on the larger grid ($0.8\,\text{ms} < 1\,\text{ms}$). The improved scaling behavior on the larger grid can thus not compensate the increased computational amount.

The obtained results show, how OpenFOAM's scaling behavior is decisively dependent on the number of cells, which are computed per core. The number obtained in this work (20000-25000) is in accordance with [137–139]. Results from scalability testing on several other meshes and *Supercomputers* confirm these findings, which is why it can be deduced, that higher parallelization on the large grid will not lead to further acceleration. Since the ideal number of cells per core is already reached no further speed-up is expected.

On the contrary, distribution of the computations on even more cores will soon even slow down the simulations. This is due to an increased amount of input-output-operations, which are limited by the interconnect between the processors.

This behavior lies founded within OpenFOAM's structure and implementation and requires thorough adaptations, which are not subject or aim of this thesis. Fortunately, model sizes investigated in this work do not make parallelization above ~ 150 cores necessary and can still be approached in a reasonable time frame with OpenFOAM's capabilities.

Of course, from a medical point of view, the duration of the simulations is still too long. If transfering parts of the applications performed in this work into medical and diagnostic surroundings, this issue must be dealt with.

3.7 Summary and Conclusion

In order to guarantee physiological relevance of the simulations, in Section 3.2 a novel BC based on the analogy between a specific electrical circuit and the coronary circulation [32] is implemented. This BC can be used to calculate volume flow curves, which are applied at the inlets of the investigated 3D geometries.

With the increased detail of the analyzed models come increasing demands on computational power as well as the required preparatory steps. The volume discretization procedure used in [18, 23, 24] cannot be applied in this work due to its high degree of required user intervention. The complexity of the segmented models would thus lead to prohibitively long working times in order to create a computational grid of sufficient quality. Moreover, opposed to [18, 23, 24] where the vessels are modelled perfectly circular, in the segmentation process used in this work also non-circular vessels occur, which makes creation of a high-quality mesh with the former software even more difficult, if possible at all. To overcome this, an alternative meshing procedure is thus benchmarked in Section 3.3, yielding results of equivalent accuracy. In addition, being highly automatable, this meshing pipeline allows for a substantial accelaration of the whole workflow.

In Section 3.4, a mesh convergence study is performed to further assure the numerical correctness of the CFD simulations. The high computational demands of the conducted simulations make the execution on High Performance Computing (HPC) clusters necessary, which comes with specific challenges and limitations for the application. One of these is the number of files in a simulation case, which must not exceed HPC cluster specific limits. Since this is a critical issue with the used software *OpenFOAM*, in Section 3.5, a method to reduce the created file numbers, while retaining numerical precision, is introduced. In Section 3.6, so-called scalability testing is performed in order to fathom the possibilities of execution of the simulations on even larger HPC clusters, so-called *Supercomputers*, where parallel computation on up to ten thousands of processors is possible. However, the obtained results suggest that usage of OpenFOAM is more recommendable on medium sized HPC clusters.

Hence, it can be stated that the applied methods make the execution of CFD simulations on the high-resolution cardiovascular models used in this work feasible in the first place.

Chapter 4

Prediction of Fractional Flow Reserve by Computational Fluid Dynamics Simulations[1]

In this chapter the BC that is derived in Section 3.2 is used to analyze blood flow simulations in mildly stenosed epicardial coronary arteries. The aim of the analysis performed here is twofold. For one thing, it allows benchmarking of the physiological relevance of the presumed conditions. And for another thing, it is an approach to explore and fathom the possibilities, which in-silico modelling of blood flow in the coronary vasculature offers in a clinical perspective. For this purpose, data from real CT coronary angiographies and associated pressure measurements from patients with coronary stenoses are used to perform CFD simulations of blodo flow in large epicardial arteries. The obtained results are then compared to distal pressure measurements, which were performed beforehand during clinical examinations.

4.1 Introduction

The FFR is an indicator to assess the severity of pathological alterations (e.g. a stenosis) in epicardial coronary arteries. As part of cardiac catheter examinations, proximal (p_a, usually in the Aorta) and distal (p_d, intracoronary) pressures are measured. The ratio, $FFR \equiv \overline{p_d}/\overline{p_a}$, of the two obtained average pressures serves as an important clinical parameter for the assessment of the severity of a stenosis, and for decision making of intervention. In the healthy individuum, this ratio should be 100 % because the contribution of epicardial arteries to flow resistance can be neglected [48].

During cardiac catheterization, a pressure wire system is inserted into the coronary arteries and placed to encompass the coronary stenosis, which is to be examined. Subsequently, by administration of adenosine or other vasodilatory drugs, hyperemia is induced in order to assess the functional capacity of the vessel of interest at maximum achievable flow. This is necessary since myocardial perfusion pressure is only proportional to blood flow into the myocardium when vasodilation is at its maximum [140, 141].

Under the assumption that in normal coronary arteries, epicardial (conductance) vessels offer negligible resistance to coronary perfusion an FFR value of 1.0 means, that blood flow in the considered artery is not hampered. If an epicardial artery is subject to a stenotic lesion, the micro-vascular resistance at rest decreases in order to maintain the basal demands of myocardial tissue by "borrowing" from the coronary flow reserve (CFR) (i.e. increase of

[1]The work presented in this chapter was performed in cooperation with Prof. Dr. S. Achenbach from Cardiology and Angiology, Medical Clinics 2, University Hospital Erlangen.

Table 4.1: Invasively measured FFR values in the two used patient datasets.

Dataset	Stenosis Position	Stenosis Degree	Measured FFR
Model 1	LAD-branch	50%	0.83
Model 2	LAD	40%	0.87

coronary blood flow at stress) [142]. As a consequence, the actual CFR is reduced, due to the decreased hyperemic flow capacity. Nonetheless, according to [142, 143], a measured FFR in the range between 0.8-1, is still considered non-significant and ischemia is very unlikely. However, if the obtained value is below 0.75, the blood flow disorder due to the coronary stenosis is identified as the cause of the patient's ischemia (with a specificity of 100 % [140]) and the patient will benefit from surgical intervention. The range between 0.75-0.8 represents a gray area, in which no definite statement about the relevance of the stenosis can be made. The cut-off value of 0.75 for revascularization is subject of debate (in [144–146] an FFR<0.8 is proposed) and the range of the gray area can also vary [147].

On top of these uncertainties and the inherent variability of subsequent measurements, the large invasiveness of the method is a reason that, despite the recommendation to integrate this technique in clinical practice [145], it is not widely used. Furthermore, the administration of adenosine for induction of hyperemia can cause side effects, such as tachycardia or arrythmia, chest pain or hypertension, [48] and is often not well tolerated by patients. To remedy this, an alternative technique, the so-called instantaneous wave-free ratio (iFR), [148–150], which allows induction of hyperemia without use of adenosine, is currently developed.

Another completely different and non-invasive approach to assess the FFR is chosen in [151–155], where CFD analyses are performed, based on coronary CT angiographies. Since this technique already finds wide-spread application, commercial providers, such as *HeartFlow*[2] [156], offer dedicated services to extract the FFR from individual patient CT angiographies. In order to assess the feasibility and possible problems of this approch, in this chapter, the methods and BCs described and implemented in Chapter 3 are used on two clinical datasets in order to determine the FFR by CFD simulations.

4.2 Methods

Two CT datasets from patients with medium stenoses, obtained from the Department for Cardiology and Angiology at the University Hospital Erlangen, are used. The diagnosed degrees of stenosis as well as the invasively measured FFRs are listed in Table 4.1. Both patients were diagnosed having mild stenoses (40 % and 50 %) and the invasively measured FFR value (0.83 and 0.87, respectively) does not lie in the range where a surgical intervention to revascularize the affected vessels is recommended.

From the CT coronary angiographies 3D models of the epicardial arteries are extracted, which are used as a basis for the CFD simulations. Model segmentation is perfomed as described in Section 6, where the size of the models is chosen such that only the relevant parts, i.e., vessels upstream of the considered stenoses are segmented, in order to keep computation times low. Furthermore, this reduces the influence of vessels, which are omitted during the segmentation process due to both the dataset's resolution as well as human factors. Each model terminates at the distal position of the pressure measurement, which

[2]www.heartflow.com, Redwood City, California, USA.

is identified by the catheter visible in the regular representations of the coronaries. This is shown in Fig. 4.1 along with the extracted vascular 3D models.

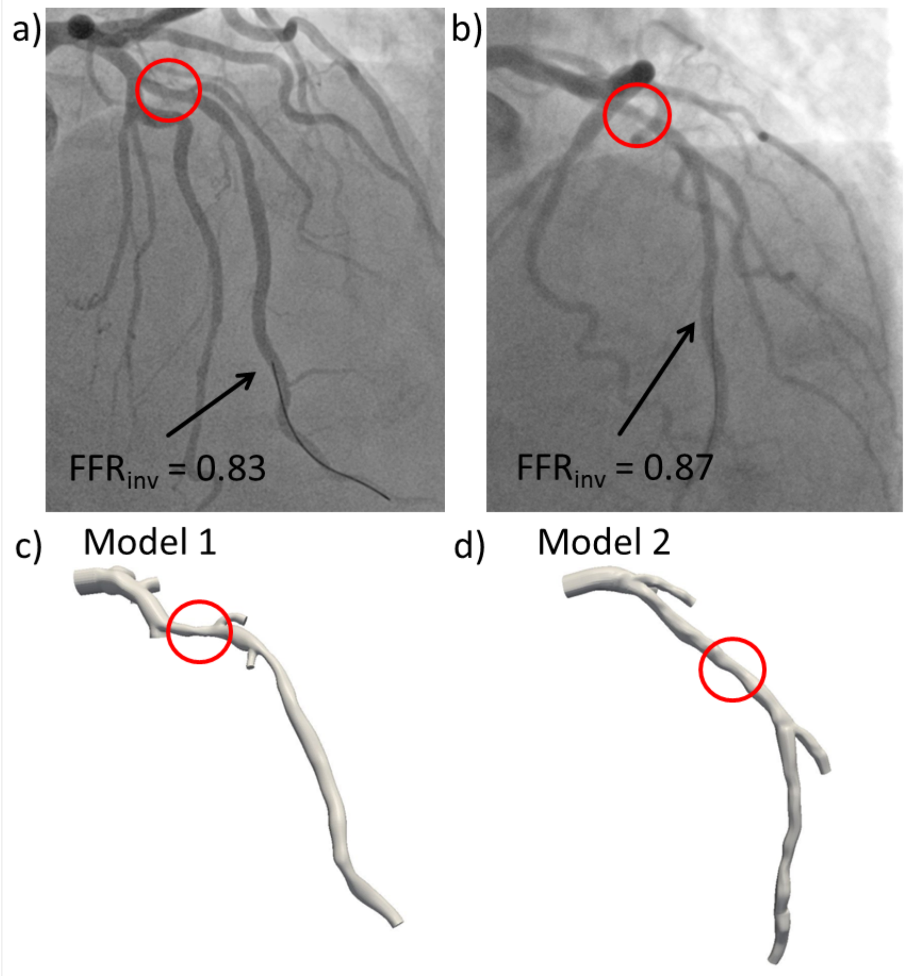


Figure 4.1: Catheter position and segmented models. Based on the regular representation (a),b)) of the coronaries, 3D models of the vasculature are segmented until the position of the pressure measurement with a catheter (c),d)). In both datasets and models, the position of the stenosis is indicated by a red circle.

Subsequent volume discretization is again performed with the software *cfMesh* with the same settings benchmarked in Section 3.3, resulting in mesh sizes of 281 748 (94 % hexahedral) and 361 419 (95 % hexahedral) grid cells for model 1 and 2, respectively. Mesh non-orthogonality and maximum skewness do not exceed 70° and 4, respectively, in each computational mesh.

The boundary conditions for the following CFD simulations are extracted from the accompanying pressure measurements in the aorta. Analogous to the procedure described in Section 3.2, the ventricular pressure curve is estimated from the shape of the measured aortic pressure curve. Similarly, the resistances of the vasculature lying downstream of the model outlets are calculated as described in Section 3.2 with the `C++`-program presented in Section A.2 under the assumption of fully dilated vessels due to administration of adenosine.

The value for the FFR is computed in four different ways. In all four simulations, a pressure boundary condition is applied at the inlet. Simulation I is the only dynamic simulation, and it is performed over the cardiac cycles, during which the invasive measurement is performed, i.e. when the effects of adenosine are observable. To compute the FFR, the mean pressure values of the exact same cycle of the invasive measurement at the distal end and in the aorta are used (cf. Fig. 4.2). At the outlets, volume flow time curves are assigned, which have been computed beforehand (cf. Section A.2).

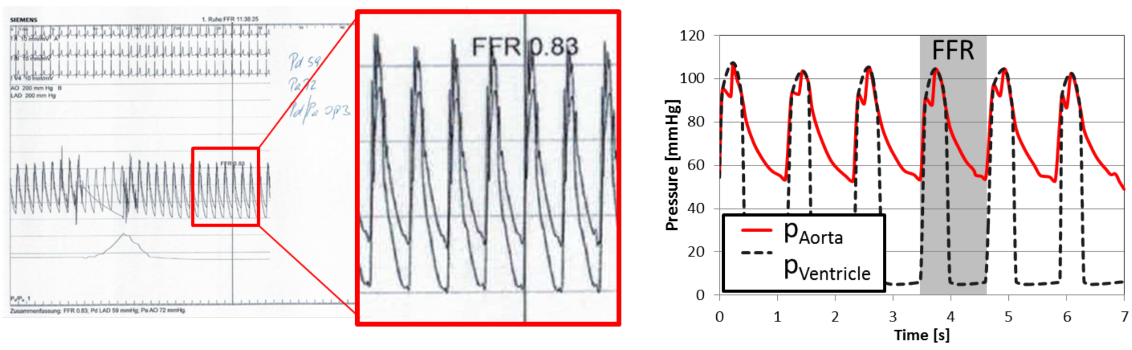


Figure 4.2: Invasive FFR pressure measurement and digitalization for model 1 in Fig. 4.1. For the dynamic simulation (Simulation I), 7 seconds around the invasive FFR measurement are simulated and the mean pressure values obtained in the cycle marked by the gray area on the right (marked by the vertical line on the left) are used for FFR computation.

The second simulation (Simulation II) is static. At the inlet the mean pressure over the cardiac cycle of the FFR measurement is applied and at the outlets the mean flow value of the same cycle is used. As soon as the outlet pressures have reached a stationary state, the simulation is stopped and the obtained value at the distal end is compared to the inlet pressure value.

The third and fourth simulations (Simulations III and IV) are also performed with static BCs. The mean pressure values applied at the inlet in these simulations are estimated from the rest state before administration of adenosine. Simulation III is performed with a fixed mean outlet flow value, and Simulation IV with the outlet resistance BC as described in Section 3.2 by $p_i = F_i \cdot R_i + \overline{p}_{Cap}$. As before, the outlet pressure p_i is computed based on the estimated resistance R_i of the fully vasodilated vascular tree downstream of the outlet, the flow F_i through the outlet and the mean value of the pressure at capillary level $\overline{p}_{Cap}$ at stress.

4.3 Results

The results obtained for the FFR with the four different simulations and the invasive measurement are listed in Table 4.2. The dynamic dynamic simulation over the cardiac cycles of invasive FFR measurement shows an overestimation of the FFR in comparison to the measurement. On the other hand, the static simulation with the outlet flow BC averaged over the exact cardiac cycle invasive FFR quantification shows a lower value of the FFR.

The simulation with averaged stress outlet flow, which is estimated from the rest pressure measurements in the Aorta and the geometry of the coronary tree, yields a slightly reduced value of the FFR in model 1 and an increased FFR in model 2. The last simulation, where the BC as described in Section 3.2 is used to compute average pressures at the model outlets yields better accordance between invasive measurement and simulation for model 1 and a slightly higher FFR for model 2.

The duration of the simulations is by far longest for the dynamic simulation of several cycles around the point of invasive measurement. It is substantially reduced for the static simulations, which are stopped as soon as the stationary state is reached. The simulations, where constant flow values are assigned to the outlets, are slightly faster than the last simulation with the more complex outlet BC from Section 3.2.

In Fig. 4.3 the FFR values obtained from the simulations with the averaged outlet BC from Section 3.2 (Simulation IV in Table 4.2) are shown with a specific color coding. The

Table 4.2: Measured and simulated values for FFR.

Simulation	I	II	III	IV
	Dynamic	Static	Static	Static
	(Cycles of Measurement)		(Estimate from rest)	
BC	Outlet Flow	Outlet Flow	Outlet Flow	Section 3.2
Model 1	0.86	0.80	0.81	0.83
Model 2	0.91	0.86	0.90	0.90
Compute Time	3-4 d	30 min	30 min	1 h

The invasively measured FFR values are obtained at the distal positions indicated in Fig. 4.1. Simulation I is a dynamic simulation of 6-7 s around the point of invasive measurement as indicated in Fig. 4.2. The FFR value is calculated averaged over the same cycle as the invasive measurement. In Simulation II a static simulation averaged over the cycle of FFR measurement is used to assess the FFR. Simulations III and IV the hyperemic state is estimated from the measurements at rest before adenosine administration to perform the invasive FFR measurement. In Simulation III a pre-calculated fixed outlet flow value is assigned in in Simulation IV the outlet BC as described in Section 3.2 is used.

vessel is colored according to the ratio of the pressure along the vessel's path over the inlet pressure (definition of the FFR) in the range between 0.8 and 1. This corresponds to the range where a patient would not benefit from surgical intervention. In both models, the vessel coloring clearly shows, how the FFR is reduced at the positon of the identified stenoses.

4.4 Discussion

Overall, the results presented in Table 4.2 show good agreement between the invasive FFR measurements and the CFD simulations. Absolute deviations between the FFR values are smaller than 5 % with regard to the measured pressure in the Aorta.

The results from the dynamic simulations (Simulation I) yield higher FFR values (0.86 and 0.91 on model 1 and 2, respectively) than the invasive measurements on both models (0.83 and 0.87). On the other hand, the static simulation with averaged outlet flow values from the exact cardiac cycle of the invasive FFR measurement (Simulation II) shows opposing behavior, with reduced simulated FFR of 0.8 and 0.86 on model 1 and 2, respectively.

The reason for the reduced FFR from the static (Simulation II) in comparison to the dynamic simulation (Simulation I) can be explained by the fact that fast administration of intracoronary adenosine has a decreasing effect on global blood pressure, [48]. This results in a general pressure drop of the pressure curves obtained from the invasive measurement. This effect is slightly visible in Fig. 4.2, but can be more pronounced.

In the dynamic simulation, this pressure reduction is (over-)compensated by the pre-calculated flow curves that are assigned to the model outlets. Due to the reduced pressure differences between each applied pressure and the ground in the corresponding RC-circuit (cf. Fig. 3.1 in Section 3.2) the myocardial compliance reaches the "fully charged" state faster and, thus, the computed flow curves run into "saturation" faster. This results in reduced flow and eventually leads to smaller pressure drops between the inlet and the outlets, yielding larger FFR values.

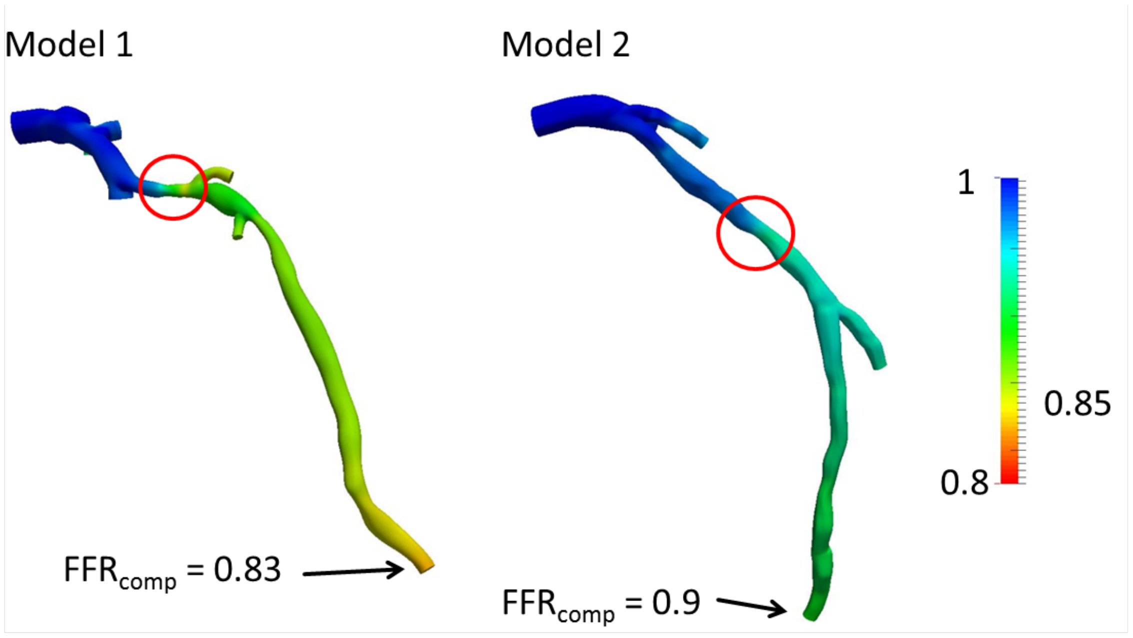


Figure 4.3: Simulated FFR within full 3D models (Simulation IV in Table 4.2). Color coding is chosen according to the calculated FFR value along the path of the vessel. In each model, it is clearly visible, how at the point of the identified stenosis, a pressure drop occurs.

In the static simulation, this effect is somewhat attenuated, resulting in reduced FFR. Nonetheless, both methods yield results, which do not deviate more from the invasive assessment than the magnitude of the grey area (5 % [157]) in such measurements. However, it should be pointed out, that the static simulation shows a considerable speed-up in comparison to the dynamic simulation with results of comparable quality.

The remaining two simulations (Simulations III and IV) are an approach to compute the FFR based on estimations from the rest state, as in [151, 153, 154, 158]. The results are of the same order of magnitude as the invasive measurement and the previous two simulations (Simulation III: 0.81 and Simulation IV: 0.83 for model 1, respectively, and 0.9 in both simulations for model 2), and the duration of the simulations is comparable to Simulation II.

At this point, it should be pointed out, that the last simulation is considered the most ralistic, since it is the only simulation, where effects of the segmented 3D geometry are actually considered in the simulation. This is also reflected in the slightly longer computation time. Instead of a fixed constant volume flow, at the outlets, the BC $p_i = F_i \cdot R_i + \overline{p}_{Cap}$ as described in Section 3.2 is applied. As a result of the integration of F_i therein, this guarantees that changes in the vasculature upstream of the outlet are incorporated in the pressure that adjusts at the outlets. Yet, the simulated FFR values are all in the same range, which can be ascribed to the fact, that the stenoses analyzed here are of mild degrees (40 % and 50 %). Typically, this does not reduce flow (and subsequently pressure) through the considered vessel enough to have significant effects, [157, 159, 160]. However, a verification that higher degree stenoses (area reduction: >50 % up to 95-99 %) do indeed reduce flow and pressure through the altered vessel in the CFD simulations would require similar analyses on such patient datasets, which is not performed here.

Moreover, even though a considerable acceleration of the CFD simulations (factor 8) is obtained by performing static instead of dynamic computations, the overall duration of the analysis still shows room for improvement. In similar studies, this kind of CFD analysis is performed in seconds or minutes [154, 158]. With dedicated work on the algorithms used for the numerical solution of the Navier-Stokes equations in the 3D geometry, this step could well be accelerated to reach a comparable timeframe.

So far, the discussion only included the last step of the analysis, the CFD simulations, yet, the previous steps (model creation as well as volume discretization) should not be left out in order to assess the time management of the analysis. Volume discretization is performed with the software package *cfMesh*, which is described and benchmarked in Section 3.3. After creation of the 3D geometry, this is an automatic endeavour of a 2-3 minutes.

On the other hand, the process of model creation itself is performed manually with the software package *SimVascular* (cf. Section 6). Models 1 and 2 analyzed in this section are not of very high detail (4 and 7 outlets), and it should take an experienced worker no more than 1 hour to segment comparable 3D vascular geometries, if it is clear, which vessels are required for the analysis. Undoubtedly, this first step (3D model creation) represents the time factor with the largest potential for improvement in the whole workflow. In comparable studies, to achieve a non-invasive FFR analysis, this is performed by application of neural networks [154, 155, 161, 162].

Conclusion and Limitations

In the end, it can be concluded that the results from this attempt to reproduce and assess invasive FFR measurements lie in a confidence-inspiring range. Using different BCs (regarding the outlets, the time dependence and the parameter estimation to consider hyperemia), the results from the CFD simulations and the invasive measurement all lie within the same region. Nonetheless, the outlet BC as described in Section 3.2 (Simulation IV) is identified as the most suitable, since contrarily to fixed outlet flow BCs, it takes into account backlash of possible vessel constrictions in the path of the vessels on volume blood flow and thereby pressure loss across the vessel. Despite this disadvantage of the simulations with fixed outflow values (Simulations I-III), the results still coincide well with the measurement and the obtained FFR value from Simulation IV. This is due to the fact that mild degree stenoses, as analyzed here, do not yield notable effects on volume blood flow. However, this underlines the necessity to expand the analysis onto more datasets for a thorough validation of the findings and the implementation as described here. Nevertheless, the simulations on the two datasets presented here still confirm the feasibility of this kind of analysis and help in identifying bottlenecks therein. As stated above, the most time-consuming factor retarding the workflow used in this section, is the manual segmentation of the cardiovascular 3D geometries.

In comparison to Chapter 3, the blood flow simulations are performed on a more complex vascular geometry. By comparison to medical measurements the simulations are benchmarked from a different perspective yielding results of equal quality. Overall, the results from the work presented in this chapter can be seen as a confirmation of the physiological relevance of the chosen settings (particularly the BC from Section 3.2).

Chapter 5

CA-Transport in Vessel Generations Six and Seven[1]

In this chapter, the analysis is extended to real cardiovascular geometries including smaller vessels. In contrast to Chapter 4, the investigation performed here focusses on the evolution of the CA concentration time curve with the distance from the model inlet. However, other than in Chapter 3, an estimation of perfusion quantification errors is spared here in favor of a dedicated analysis of CA bolus dispersion in smaller coronary arteries and at vessel bifurcations.

5.1 Introduction

The CFD simulations presented in this chapter are based on the results from previous studies [18, 24, 25, 90, 164] where both idealized (cf. Chapter 3) as well as realistic cardiovascular geometries have been investigated, also with regard to subsequent systematic errors in MBF quantification. These analyses show systematic underestimation of MBF values due to various parameters (e.g. flow velocity, length, curvature). In order to further investigate the influence of ever smaller vessels on CA dispersion, in this chapter, CFD simulations are performed on porcine vascular 3D models starting at the first diagonal artery branching off from the LAD (i.e. vessel generation three) and including vessels until generation seven.

5.2 Methods

In the following, several steps required in advance of the CFD simulations are outlined (cf. Section 3.3.2).

Model Creation

"The analysis is based on an imaging data set of a healthy *ex-vivo* porcine heart at a resolution of 160 µm, generated by an imaging cryomicrotome at Amsterdam Medical Center [26, 28]. With the help of the dedicated software package VMTK[2], 3D-models of epicardial vessels are extracted from a high resolution imaging data set. For a systematic study of CA

[1]Large parts of this chapter have previously been published in [163]. Literal quotations are marked by " ". The usage here is in accordance with the associated *Consent to Publish* from *Springer – Lecture Notes in Computer Science*: "Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgement is given to the original source of publication". Partially, content and wording have been altered in order to fit the overall structure of the thesis, e.g., here, the analysis is peformed with regard to SD_{VTF} instead of SD^2_{VTF} as it is done in [163].

[2]www.vmtk.org

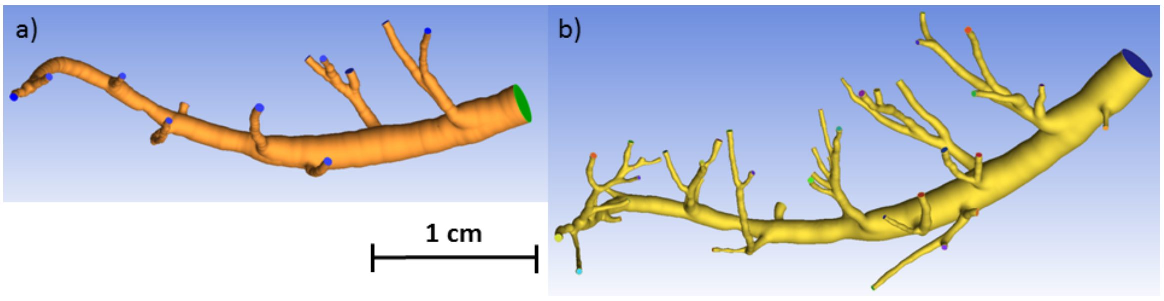


Figure 5.1: "Models to study vessel dependence of CA bolus dispersion, starting at first diagonal artery (vessel generation 3) branching off from LAD (inlet diameter ∼ 2mm). Model a) includes generations 3-5 and b) generations 3-7." (Figure taken from [163].)

bolus dispersion dependence on vessel generation, two models with varying level of detail are (cf. Fig. 5.1) created." The models start at the first diagonal branch of the LAD, with an inlet diameter of $\sim 2\,\mathrm{mm}$. The outlet diameters range between $350 - 500\,\mu\mathrm{m}$ in Fig. 5.1 a) and $160 - 300\,\mu\mathrm{m}$ in Fig. 5.1 b).

Volume Discretization

These geometries are meshed with the two software packages described in Section 3.3, *cfMesh* (for Fig. 5.1 a)) and *ICEM* (for Fig. 5.1 b)). Meshing time for Fig. 5.1 a) amounts to ∼ 10min, while the time required for creating the computational grid in Fig. 5.1 b) with *ICEM* is several weeks, before all required manual steps are performed and a mesh of comparable quality is obtained (cf. Section 3.3).

Mesh sizes are 2 999 543 for the purely hexahedral *ICEM*-grid and 799 625 for the mainly hexahedral ($\sim 99.5\,\%$) *cfMesh*-grid. The mesh used in Fig. 5.1 a) exhibits a maximum *mesh non-orthogonality* of 67.4° and a maximum *skewness* of 2. On the *ICEM*-grid (Fig. 5.1 b) OpenFOAM's mesh quality parameter *mesh non-orthogonality* counts 227 ($< 10^{-2}\,\%$) grid cells with values above 70° (max: 78.8°) and *skewness* measures 2.8. These values reflect the difficulty to produce high quality meshes with *ICEM* on complex geometries as depicted in Fig. 5.1.

CFD simulations

The governing equations (Eqs. 2.21 and 2.26) are solved in the same two-step-procedure as described in Section 3.3.2. As BC at the model inlet, the pressure-time curve depicted in Fig. 5.2 is applied. "It is taken from [25] and scaled by 50 % according to [165] to account for pressure decrease between the LAD in [25] and the smaller diagonal artery in this work."

"The BC at the outlets is the resistance model as used in [24, 25], where the outlet resistances are computed according to the so-called structured tree [135, 136]."[3] In this approach, a linear one-dimensional hydrodynamic model is applied, which predicts flow and pressure by balancing forces of the elastic wall with forces acting on the fluid, thus including effects of arteriolar tone in the computed outlet resistances. The applied outlet resistances are calculated as a zero frequency structured tree impedance with the root radius corresponding to the model outlet in question and a minimal radius of $50\,\mu\mathrm{m}$ at arteriolar level. In this model it is assumed, that an asymmetrical binary structured tree can be constructed where the radii of branching vessels are scaled by set constant factors until at the determined minimal radius the branching terminates. The number of vessels within the

[3]The BC from Section 3.2 is not used here, because the required associated capillary pressure curve was not available.

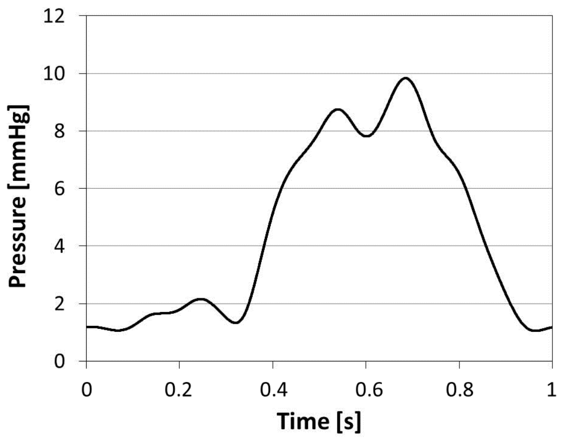


Figure 5.2: “Applied inlet pressure time curve from [25] scaled by 50%.” (Figure taken from [163].)

structured tree is derived depending on this minimal radius and the respective model outlet radius [166]. Combined with the outflowing blood volumes the above resistances provide outlet pressures needed for the solution of the Navier-Stokes-equations:

$$p_i = F_i \cdot R_i, \tag{5.1}$$

where p_i is the pressure at outlet i, F_i the outlet flow and R_i the outlet resistance.

Please note the missing capillary pressure p_{Cap} in comparison to Eq. 3.3. This is left out here, because the inlet pressure curve as depicted in Fig. 5.2 is obtained by a simulation based on an inlet flow BC in [25]. The relation of diastolic and systolic flow is already comprised in this pressure time curve.

“Since the solutions of the Navier-Stokes equations only depend on the difference between the applied inlet pressure and the correspondingly computed outlet pressures, the physiologically unrealistic pressure range in Fig. 5.2 can be ignored.” All other variables and the discretization parameters are chosen analogous to Section 3.3.

The simulations are performed on a local server as the simulations in Chapter 3 as well as the high performance computing cluster *elwetritsch* at University Kaiserslautern, parallelized on 128 processors. In this setup, the compute time for the simulation of blood flow and CA transport amounts up to two weeks per computed model.

The subsequent analysis of the obtained CA dispersion is performed using the quantities described in Section 2.3.4. With regard to the VTF, the SD_{VTF} (Eq. 2.89) and the $MVTT$ (Eq. 2.88) are calculated. Furthermore, an approach to analyze bolus dispersion with the help of a local AIF quantified on specifically defined intraluminal segments is made. This is done comparing the changes in SD_{AIF} (Eq. 2.92) and the mean bolus arrival time $\overline{T}$ (Eq. 2.93) between an intraluminal segment and a downstream branching vessel, which can be associated to the considered segment.

5.3 Results

“Figure 5.3 a) shows the computed results for CA bolus dispersion for all 12 outlets of the model shown in Fig. 5.1 in comparison to the AIF applied at the inlet.’ Figure 5.3 b) shows the obtained values for” SD_{VTF} “as a function of $MVTT$, calculated as described in” Section 2.3.4 “for both considered models (Fig. 5.1).” “The values of $MVTT$ and” SD_{VTF} range between $0.34 - 0.97$ s (mean value (0.63 ± 0.20) s) and $0.26 - 1.06$ s (mean value (0.67 ± 0.25) s) respectively in model Fig. 5.1 a), and between $0.39 - 1.63$ s (mean value (0.83 ± 0.27) s) and $0.19 - 1.34$ s (mean value (0.68 ± 0.31) s) respectively in model Fig. 5.1

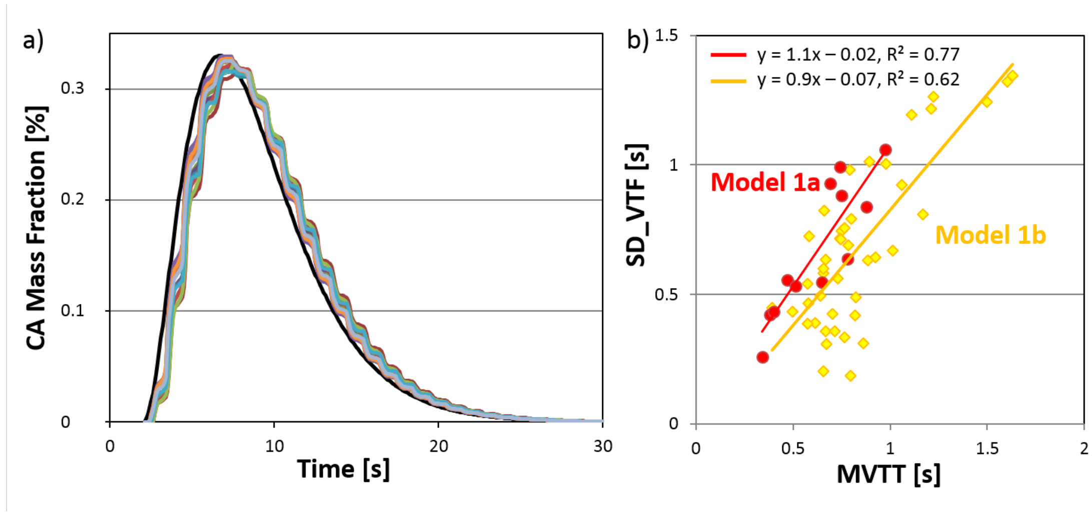


Figure 5.3: Results for dispersion, $MVTT$ and SD_{VTF}. “a) shows AIF_{disp} at all outlets of model Fig. 5.1 a), with the applied inlet concentration time curve AIF_{LV} (black).” b) depicts SD_{VTF} as a function of MVTT for both models. “The results at the outlets of model Fig. 5.1 a) are shown in red, with a linear fit (red line, standard errors for slope and intercept:” ±0.19, ±0.12, resp.). “Analogously, the results for model Fig. 5.1 b) are depicted in yellow (standard errors:” ±0.11, ±0.10). “Error margins of the graphs overlap, however, they do not yield information on vessel generation dependence of dispersion.” (Figure adapted from [163].)

b). “The obtained results for AIF_{disp} at the model outlets as well as the computed” SD_{VTF} “and $MVTT$ show reduced dispersion effects on CA bolus disperson than what is observed in [25] and [90] alike.

The data presented in Fig. 5.3 give information about the shape of the AIF at the different model outlets and their respective” SD_{VTF} and $MVTT$. “However, a clear understanding of the influence of different vessel segments and generations on CA bolus dispersion remains unclear. Figure 5.4 a) shows locations in the large model Fig. 5.1 b) between inlet and exemplary outlets, where additional information about CA concentration time curves are extracted. These allow for the analysis of” SD_{VTF} “and $MVTT$ of VTF depending on the distance from the model inlet as a measure for CA bolus dispersion in the epicardial vessels.”

“Figure 5.4 b) and c) show the dependence of $MVTT$ and SD_{VTF} at discrete positions in the pathway of the CA bolus to the outlets marked by arrows in Fig. 5.4 a). The different branches individually show very different behavior. Branch 8, the most unidirectional branch of the analyzed bifurcations in the model, roughly shows the expected saturation effects in bolus broadening with increasing vessel generation until the CA bolus reaches the region marked in blue in Fig. 5.4. Depending on bifurcation angles and vessel curvature the other branches show both dispersion as well as narrowing effects alike. In particular, results obtained in branches 2, 6 and 7 show that CA bolus broadening can be reduced until no additional dispersion actually remains at the distal ends. Branch 1, 2, 3, 4 and 5 show bolus dispersion effects just behind the bifurcation with 2, 3 and 4 showing strongly reduced dispersion later in the pathway. The remaining branches 6 and 7 differ in their behaviour, overall reducing obtained bolus dispersion.

To further investigate the behaviour of CA bolus dispersion in the vessels branching off from the central vessel, Fig. 5.5 shows CA flow through a model segment including branches 2 and 3 at different points in time. These two vessels branch off in different angles from the central branch 8; both draw bends of different curvature downstream; and each branches into further smaller vessels. Branch 2, with a steeper bifurcation angle, shows more dispersion

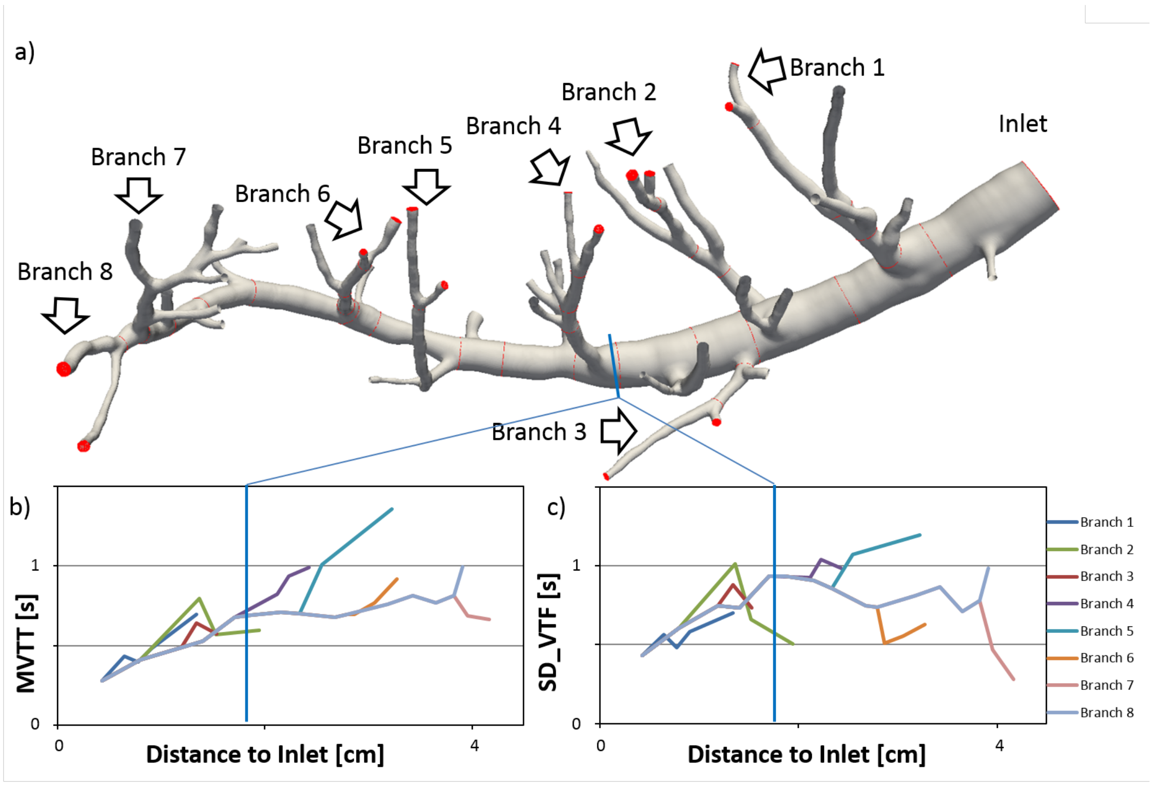


Figure 5.4: “Position of cross sectional planes and analysis of $MVTT$ and” SD_{VTF} “dependence on distance from model inlet.” “The vessel branches numbered in a) are analyzed in more detail. b) and c) show the evolution of $MVTT$ and” SD_{VTF} “in the course of the numbered branches at discrete positions (red cross sections marked in a)). The straight lines connecting the values solely serve to enhance visual presentation and do not hold any physical meaning. Close to the inlet both quantities increase. Travelling further along the considered vessels the behaviour varies strongly comprising dispersion as well as narrowing effects. Until the region marked in blue in the middle of the vascular model both quantities describe the expected monotonously increasing behavior.” (Figure adapted from [163].)

than branch 3 in the first segment; however, also more dispersion reduction is observed afterwards in its more pronounced bend.” For a better impression of flow of the shortened bolus through the whole geometry and the particular bifurcation from Fig. 5.5, please also see Videos 4-1 and 4-2 on the attached CD.

“Figure 5.6 shows velocity streamlines in the two branches analyzed in detail in Fig. 5.5.”[4] “The two images allow for a more vivid picture of how CA is transported by convection from the main branch through bifurcations into downstream vessels. Streamlines indicate that depending on their orientation daughter vessels are ‘fed’ by different cross sectional areals of the mother vessel. Accordingly the varying distribution of CA in the main branch will have an impact on observed dispersion in branching vessels.”

Figure 5.7 shows an attempt to identify intraluminal segments within the main branch, on the basis of streamlines leading into the branching vessels (branches 2 and 3 in Fig. 5.4). Due to the pulsatility of blood flow, the streamlines are not enclosed by the defined segment at all times. For a better impression please also see Video 4-3 on the enclosed CD.

In Fig. 5.8 the obtained values for $\overline{T}$ and SD_{AIF} of the CA concentration time curves on the intraluminal segments from Fig. 5.7 are shown in comparison to the concentration

[4]A streamline is a curve, which is tangent to the velocity of flow at a given point in space and time. It shows the direction in which a massless particle suspended in the fluid would travel.

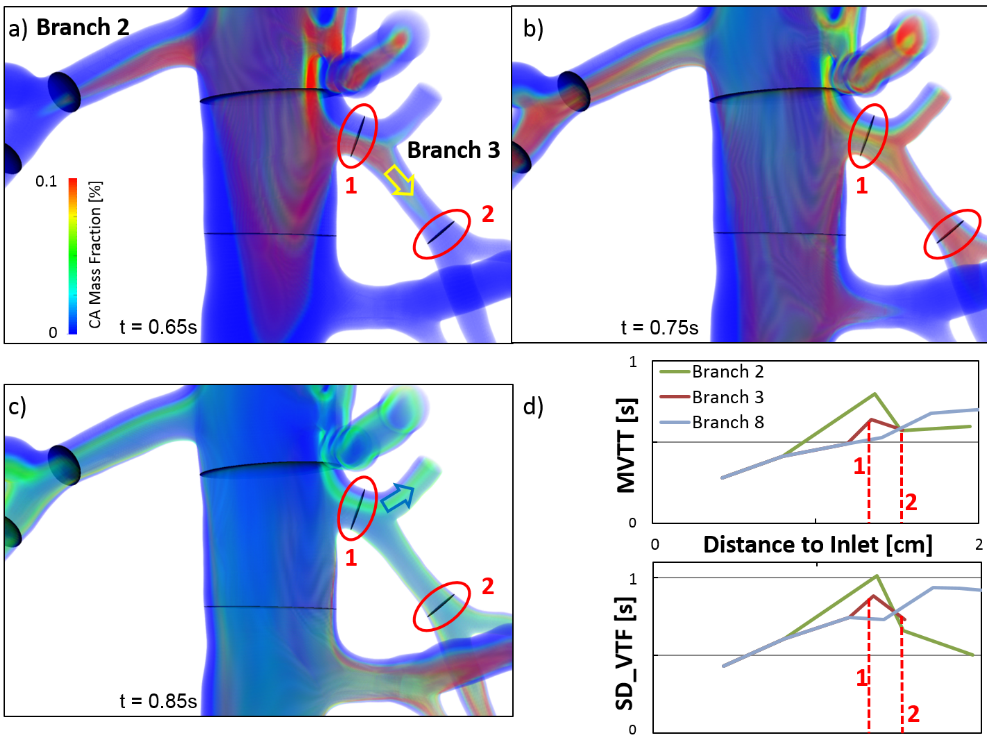


Figure 5.5: “CA flow through model segment and corresponding $MVTT$ and” SD_{VTF}. “a,b,c) show CA mass fraction (in %) of a shortened bolus (factor 100, for better visualization) at different points in time. In d) $MVTT$ and” SD_{VTF} “in branches 2 and 3 are plotted depending on the travelled distance. As can be seen in a,b,c) the CA bolus takes long to completely pass region 1, resulting in increased $MVTT$ and” SD_{VTF} “(cf. d)). While in a) and b) most CA passing region 1 enters region 2 (yellow arrow), in c) CA mostly flows into the steep bifurcation to the side (blue arrow), thus reducing $MVTT$ and” SD_{VTF} “in region 2 (cf. d)). Similar but more pronounced behaviour is observed in branch 2, with a bifurcation angle of nearly 90°.” (Figure adapted from [163].)

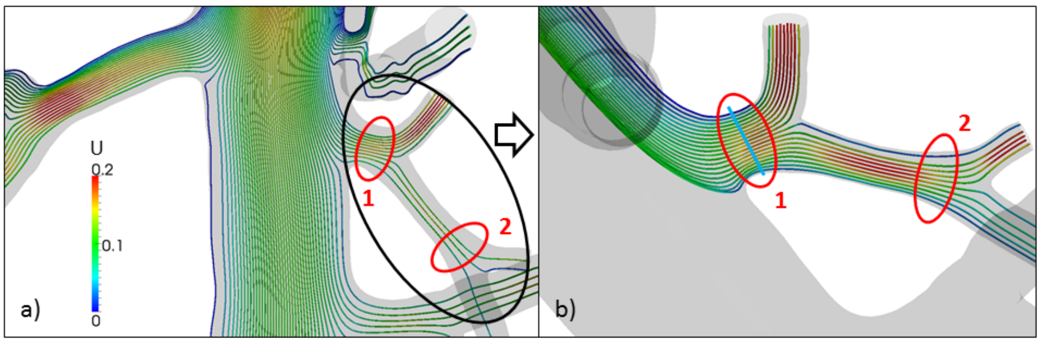


Figure 5.6: “Streamlines in branches 2 and 3 at 0.65s of one cardiac cycle (cf. Fig. 5.2), same timestep as Fig. 5.5 a). a) Seed of the plotted streamlines is a horizontal 2D line source in the image plane ahead of the two branches at the upper end of the depicted model section. The sector encircled in black is represented enlarged in b). The velocity scale (units ms^{-1}) in a) applies for both graphics. b) The 2D line source used as seed is indicated in blue. The two branches and subsequent daughter vessels are ‘fed’ from different cross sectional areals of the main branch. CA flow and velocities in the main branch thus strongly influence CA concentration time curves in branching vessels.” (Figure adapted from [163].)

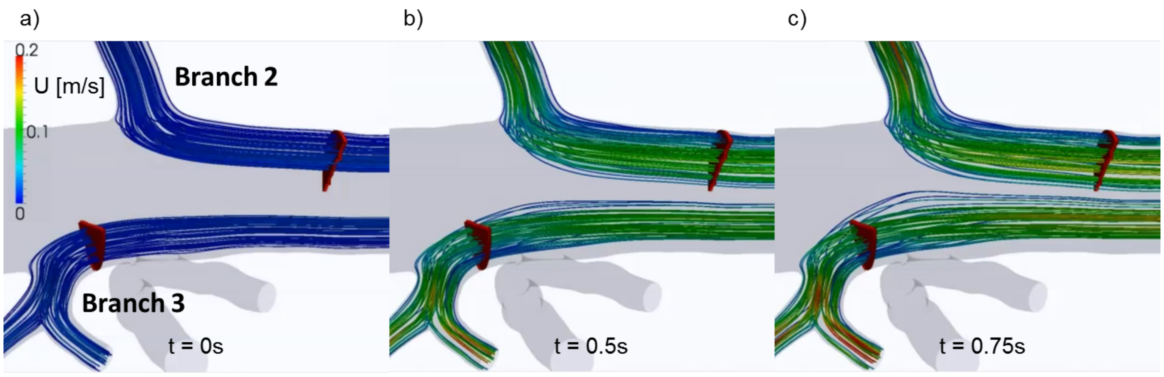


Figure 5.7: Definition of intraluminal segments for assessment of local AIF. Based on streamlines leading into the branching vessels (branches 2 and 3 from Fig. 5.4, cross sectional segments (red) are defined. a), b) and c) show the shapes of the streamlines at times 0s, 0.5s and 0.75s of the cardiac cycle, respectively.

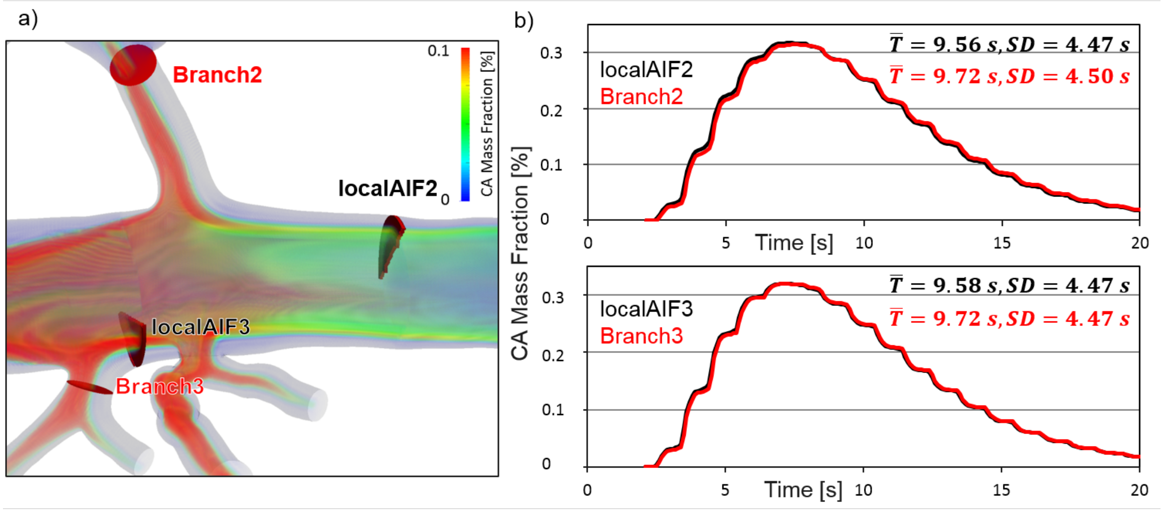


Figure 5.8: Local AIFs on intraluminal segments. a) shows CA tranport into branches 2 and 3 as well as intraluminal sections and cross-sections where CA curves in b) are obtained. $\overline{T}$ and SD_{AIF} of local AIFs show increased bolus broadening in branching vessels similar to Fig. 5.5.

time curve obtained in the branches. Analogous to the behavior obtained for $MVTT$ and SD_{VTF} of the VTF, both $\overline{T}$ and SD_{AIF} show a slight increase.

5.4 Discussion

"The computed AIF_{disp} depicted in Fig. 5.3 a) and the corresponding values for $MVTT$ and" SD_{VTF} "suggest reduced influence of increasingly smaller vessels on CA dispersion in coronary arteries as compared to previous studies [18, 22–25, 90, 164]. Both vascular models show a similar correlation between $MVTT$ and SD_{VTF} (cf. Fig. 5.3 b)). However, only considering CA concentration time curves at the model outlets does not yield specific information about the influence of vessels of different generations and their length on CA bolus dispersion. This requires a more detailed analysis with regard to the covered distance within the considered vessel sections.

The calculated results for" SD_{VTF} "and $MVTT$ along the different branches marked in Fig. 5.4 a) vary strongly from one another. Effects both leading to increasing dispersion (broadening of the VTF, i.e., increase of" SD_{VTF}) "as well as reducing dispersion (i.e., decrease of" SD_{VTF}) "are observed together with distinct fluctuations in $MVTT$ (cf. Fig. 5.4). The only branch showing the expected reduced influence of increasingly smaller vessels on CA dispersion in coronary arteries is the central (most unidirectional) branch 8 of the model. However, this branch also underlies fluctuations in $MVTT$ and" SD_{VTF}.

"The observation of an apparently reduced dispersion in the curved bifurcating vessels is in fact a consequence of the inhomogeneous distribution of CA across the branching vessel. In earlier studies similar observations are made in curved vessels, where CA distribution is also severely inhomogeneous across the vessel lumen, and where the variance increases in and behind the stenosis [18, 23, 81]. A few millimeters after the stenosis the variance decreases significantly but towards a value higher than before the stenosis. In analogy to those observations, we see a transitional increase of" SD_{VTF} "in the branching vessels before it returns to a lower value."

"The exact amount and temporal variation of CA which enters the branching vessel depends on a number of different factors. First, CA is transported by convective flow as suggested by the streamlines in Fig. 5.6. The amount of CA entering the branching vessel depends strongly on the (inhomogeneous) CA distribution within the main vessel, and the exact position of the ostium of the branching vessel. Moreover, diffusion of the contrast agent modifies this transport." The attempt to filter out the effect of the inhomogeneous CA distribution across the vessel lumen by definition of local AIFs as in Fig. 5.7 and Fig. 5.8 proves difficult due to the blood flow's pulsatility. This renders an analysis by local AIFs of the entire geometry infeasible. Furthermore, the analysis of bolus broadening in the exemplary vessel segments and the corresponding branching vessels reveals no supplementary information about the effects observed in these simulations. Quite the opposite, it confirms the findings obtained with the analysis of $MVTT$ and the SD_{VTF}.

"Strictly speaking, the concept of defining a mean vascular transit time in Eq. 2.87 may not be completely adequate: a general assumption of indicator dilution theory is that the intraluminal vessel space is a well-mixed single compartment. This assumption is not fulfilled because of the inhomogeneity of CA across the vessel lumen. Therefore, quantification of MBF on the basis of Eq. 2.91 is not feasible. Conservation of mass is only given if there is no other sink between the inlet and the vessel positions investigated. As the CA travels along the vessels, this requirement is increasingly not fulfilled. In our study the mean vascular transit time ($MVTT$ as above) is therefore considered a parameter to easily calculate the width of the bolus, but it is not supposed to reflect the true mean transit. It is mainly a parameter for simple analysis with a yet to be determined error.

Furthermore, the vascular system in question needs to be a linear system as defined in [167]" (cf. Section 2.3.4), "where system input and output are linearly related with regard to the amount of CA injected at the main vessel inlet. Deviations from this hypothesis may be present because of the non-negligible concentration dependent diffusion effects of the CA resulting in appearingly unphysical findings of reduced $MVTT$ and" SD_{VTF} "with increasing vessel generation.

However, the evolution of both quantities in the main branch of the considered model shows the expected monotonous increase, until CA reaches the region marked by the blue line in Fig. 5.4 a). This suggests that in the upstream region before branch 4 the above conditions for a linear system are satisfied at least in a first approximation. Behind this point distinct stenoses in the main vessel can be identified just after branch 4 and 5, and furthermore branch 8 itself becomes smaller and also underlies more curvature. In addition, more CA leaves the main branch at bifurcations and as a result the requirements for a linear system may not hold perfectly.

Verification of these results with the help of an alternative and more generally valid definition of $MVTT$ and SD_{VTF} as a measure for dispersion is still necessary. Nevertheless, the observations made in this work underline the complexity of CA transport and diffusion due to vessel curvature, tapering and branching."

Conclusion

The methodological analysis of CA transport performed in this chapter reveals a complex interplay of several different factors, which determines the degree of bolus dispersion. It is observed that due to vessel bifurcations and the orientation of branching vessels relative to the upstream mother vessels bolus broadening in smaller arteries can be less pronounced. In fact, by reason of inhomogeneous CA concentrations across the vessel lumen, leading to heterogeneous CA transport into the branching vessels, even reduced CA dispersion is obtained locally. Since smallest vessels generally lie at the end of a branching network with myriads of upstream bifurcations, it follows that additional dispersion in these vessels can be reduced.

This represents the major observation of this chapter, and it puts into question the assumption that the 3D model can be considered a linear system, as postulated by [85, 86]. Not only is the total amount of tracer not conserved between the model inlet and each specific outlet, but also, the analyzed vessel segment cannot be considered a homogeneously mixed compartment.

Chapter 6

Blood Flow and CA Transport in the Porcine Vasculature Including Small Vessels at Pre-Arteriolar Level

6.1 Introduction

In order to allow for a comprehensive understanding of blood flow and CA transport in the coronary vasculature, in this chapter, the modifications from Chapter 3, which have been benchmarked in Chapter 4, are used to perform the analysis from Chapter 5 in highly detailed cardiovascular 3D geometries of the coronary arteries. The used 3D models include the left and right coronary trees as well as small vessels of pre-arteriolar level. The results from the blood flow simulations are examined with regard to volume blood flow (VBF) into different myocardial regions both at rest and under stress as well as VBF in dependence of the vessel diameter. The evaluation of the transport simulations is performed with regard to the observed CA dispersion in vessels of different size and with respect to the obtained perfusion quantification errors.

6.2 Methods

In the following, the main preparatory steps are shortly outlined. Since the used procedure and BCs are generally identical to what is described in Section 3.3.2, only explicit differences to the previously performed steps are pointed out.

6.2.1 Model Generation and Preparation

Model Creation

The 3D models are extracted from the same high-resolution cryomicrotome imaging dataset of an ex-vivo porcine heart [26, 28, 132, 133], which was also used in Chapter 5. However, here the software package *SimVascular* is used for model creation as in Section 3.6, opposed to Chapter 5. Vessels of radii as close as possible to the resolution of the imaging data set of 160 μm are included in this procedure. Subsequent surface smoothing and file preparation is performed as described in Section 3.3.2. Due to the high complexity of the models, this is a manual endeavor of several months. The generated models are depicted in Fig. 6.1. For additional visualizatoin, Video 6-1 on the enclosed CD shows the rotating coronary trees.

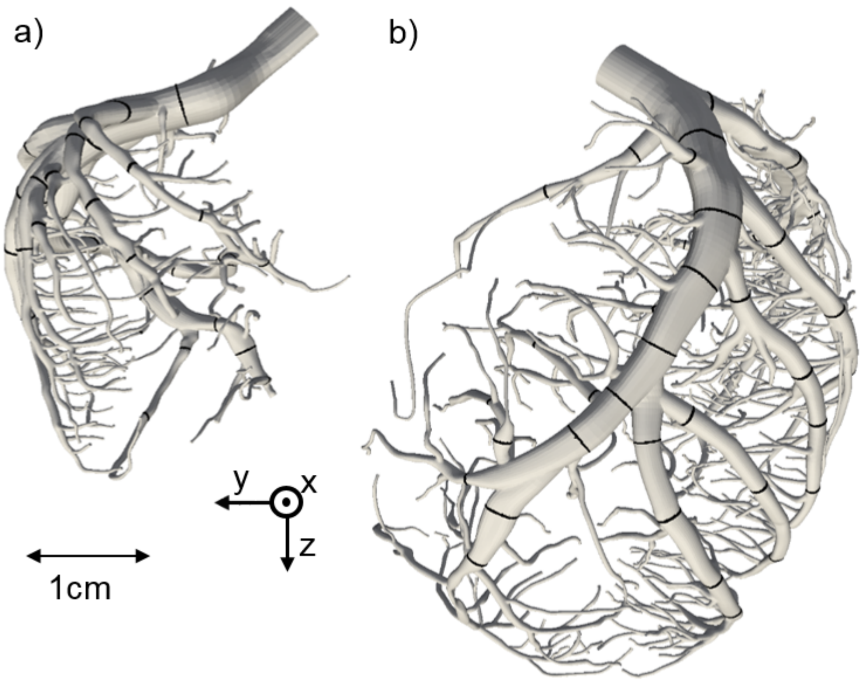


Figure 6.1: Anterior view of the generated models of the right (a) and the left coronary tree (b). Along the paths of the large epicardial arteries in both coronary trees, cross sections are defined, which are used in the CA dispersion analysis. A rotating visualization of both models is available on the CD enclosed with this thesis (Video 6-1).

Volume Discretization

In the following step, the vascular geometries are automatically discretized with the help of the software package *cfMesh* (cf. Section 3.3) taking approximately 1.5 hours and 20 minutes for the models of the left coronary tree (LCT) and the right coronary tree (RCT), respectively. Based on the mesh convergence analysis presented in Section 3.4, the generated computational grids consist of 13 830 835 and 6 975 193 largely hexahedral (LCT: 96 %, RCT 95 %) cells, respectively. In order to best resolve and take account of velocity gradients near to the boundaries, the meshes are automatically refined towards the vessel walls.

6.2.2 CFD Simulations

The CFD simulations of blood flow and CA transport are performed with the software *OpenFOAM* (Vs. 2.3.1) in the two-step procedure described in Section 3.3.2 and used throughout this thesis (1. blood flow simulation, 2. transport simulation), using identical BCs as introduced in Section 3.2.

At the model inlets, the volume flow curves as depicted in Fig. 6.2 are applied. These are calculated based on the model described in detail in Section 3.2. The initial aortic pressure curve is taken from [168] and the associated ventricular pressure curve is estimated in a physiologic range according to [101]. The cardiac cycle duration at rest is scaled to 0.7 s, yielding a typical heart rate for pigs weighing 25–35 kg [169], which corresponds to the size of the animals from which the used dataset was extracted [133]. As in Chapter 3 and 4, pressure at the model outlets is computed by Eq. 3.3 (cf. Fig. 3.2) and the required flow resistances describing the downstream vasculature are calculated as explained in Section 3.2. The simulations are performed with the solver described in appendix A.3 and discretization in space and time is performed as described in Section 3.3.2 on page 36. Analogously, the subsequent solution of the advection-diffusion equation (cf. Eq. 2.26) is

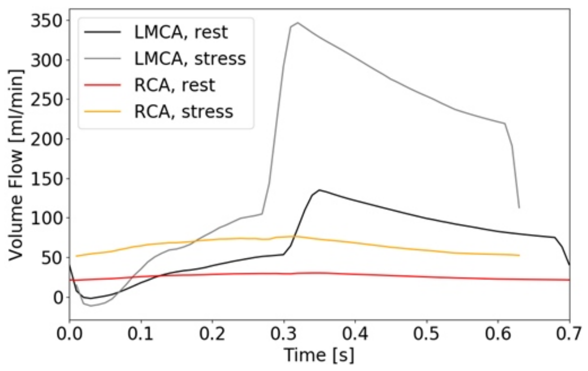


Figure 6.2: Rest and stress inlet volume flow curves for LMCA and RCA. In order to account for an increased heart rate at stress, the duration of the cardiac cycle is scaled by 90 % during computation of the hyperemic flow curves using the BC described in Section 3.2.

performed as described on page 38, i.e., a γ-variate function is used to approximate the real CA concentration time curve at the model inlet [22, 121].

Computational Facilities

All simulations are performed on the HPC cluster *CoolMUC-2* at the *Leibniz-Rechenzentrum* (Munich, Germany) parallelized on 140 and 56 cores for the LCT and the RCT model, respectively. To complete a simulation with this setup takes approximately two days, generating a total of $\sim$2 Mio files amounting to $\sim$1-2 TB for each case (LCT and RCT both at rest and under stress). Rest and stress simulations are conducted for each model.

6.2.3 Analysis of Blood Flow

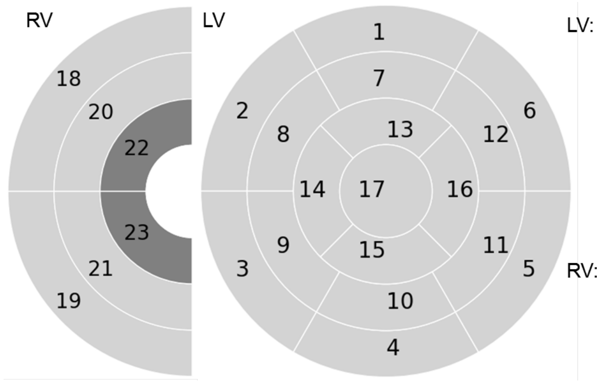
The results of the blood flow simulations are analyzed with regard to different parameters, which are presented in the following.

Volume Flow into the Myocardial Segments and MBF Heterogeneity

The solutions of the Navier-Stokes simulations are evaluated with regard to the 17 myocardial segments of the left ventricular myocardium as defined in [170]. In order to perform a similar analysis of blood flow into the musculature of the right ventricle (RV), an analogous segmentation into six segments is defined. The partitions of both ventricles as they are used here are shown in Fig. 6.3.

Based on their spatial coordinates, the model outlets can be associated with the defined myocardial segments. The fractions of mean total VBF into the segments is compared both at rest and under stress, also allowing for an estimation of the MPR in the different segments.

The orientation of the dataset is not perfectly aligned with the MRI planes as defined in [171]. For this reason, different rotations around the coordinate axes (cf. Fig. 6.1) of the cardiovascular 3D models are tested in order to obtain physiologically realistic VBF values in the myocardial segments. The distance between heart base and apex is subdivided into 4 segments (basal, mid, apical and apex cf. Fig. 6.3), corresponding to fractions of 32, 27, 27 and 14 % of the distance between the most basal and the most apical model outlet in the LCT-dataset. This division is chosen in order to best fit the standards defined in [170] without knowledge of the distribution of myocardial tissue in the segments or landmarks that are usually utilized to define the segments.



LV:
1. Basal Anterior
2. Basal Anteroseptal
3. Basal Inferoseptal
4. Basal Inferior
5. Basal Inferolateral
6. Basal Anterolateral
7. Mid Anterior
8. Mid Anteroseptal
9. Mid Inferoseptal
10. Mid Inferior
11. Mid Inferolateral
12. Mid Anterolateral
13. Apical Anterior
14. Apical Septal
15. Apical Inferior
16. Apical Lateral
17. Apex

RV:
18. Basal Septal
19. Basal Inferior
20. Mid Septal
21. Mid Inferior
22. Apical Septal
23. Apical Inferior

Figure 6.3: Right and left ventricular segments [170]. From outside to inside, the segments run from base to apex (cf. Section 2.2.2), i.e., from the top to the tip of the heart. The dark grey segments 22 and 23 are not considered in the analysis because the imaging cryomicrotome dataset did not allow segmentation of further vessels in the apical region of the right ventricle. The segments 1–17 encompass the whole left ventricle including the septum dividing the two ventricles (segments 2, 3, 8, 9 and 14). Therefore the right ventricle only consists of a “semicircle”.

In the next step, the mid and basal regions are further sectioned into six parts and the apical region into four parts. The Apex is not subdivided and fully constitutes segment 17 from Fig. 6.3. Analogously, the outlets of the RCT are associated with the RV-segments 18–21 by halving the maximum distance between most basal and most apical as well as most inferior and most septal outlet coordinates.

Here, it should be noted that this association of model outlets to the myocardial segments can merely serve as an approximation for segment blood flow. Moreover, the chosen rotation angles and subdivisions strongly affect the obtained results for volume flow value into the segments.

Dependence of Volume Flow on Vessel Diameter

To analyze if the computations are in agreement with morphometric analyses on vascular volume, diameter and volume flow [36, 172], the obtained VBF F_i in vascular segments are analyzed with regard to the following relationship

$$\frac{F_i}{F_{max}} = A \cdot \left(\frac{D_i}{D_{max}} \right)^B . \tag{6.1}$$

Here, D_i denotes the stem diameter (i.e., the beginning) of the considered vascular tree. The variable F_{max} is the volume flow through a larger upstream vessel, representing the beginning of the considered vascular network (i.e., the total vascular tree) and D_{max} the associated stem diameter. Following [36], the constants A and B are expected to be 1 and $7/3$, respectively (cf. Fig. 6.4).

The performed CFD simulations are investigated with regard to Eq. 6.1. This is done by non-linear regression of the mean volume flow values and the associated diameters at the model outlets as well as the cross sections shown in Fig. 6.1. The analysis is performed separately for the three large coronary trees starting at the RCA, the LAD and the LCX. Non-linear regression is performed in two different fitting procedures. First, only the proportionality constant A is left free to vary and B is fixed to $7/3$ as it is done in [36]. Secondly, both parameters A and B are varied in a curve fitting procedure to examine the general distribution of volume flow across all vessels. For each coronary tree (RCA, LAD, LCX), this is performed considering the corresponding model outlets and cross sections separately as well as both together.

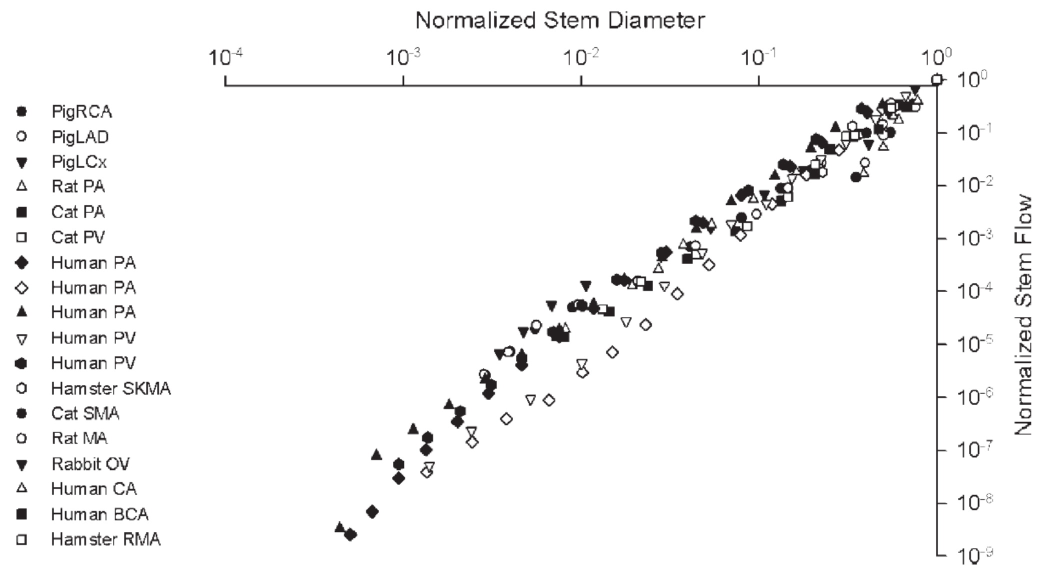


Figure 6.4: Expected stem flow diameter relationship from [172]. A power law relation corresponding to Eq. 6.1 is obtained from the analysis of morphological data from several organs of different species [36, 172]. RCA, right coronary artery; LAD, left anterior descending; LCX, left circumflex artery; PA, pulmonary artery; PV, pulmonary vein; SKMA, skin muscle arteries; SMA, sartorius muscle arteries; MA, mesentery arteries; OV, omentum veins; BCA, bulbular conjunctiva arteries; RMA, retractor muscle artery.

6.2.4 Analysis of CA Transport

CA dispersion is investigated on the basis of the $MVTT$ and the SD_{VTF} of the VTF (Eqs. 2.88 and 2.89) as well as $\overline{T}$ and SD_{AIF} of the AIF (Eqs. 2.92 and 2.93). The evolution of these parameters along the large epicardial arteries is observed in order to examine the obtained CA dispersion.

Moreover, the relative dispersion (RD) as defined in [167, 173, 174] is calculated at the model outlets and the cross sections:

$$RD = \frac{SD}{\bar{t}}, \tag{6.2}$$

corresponding to the coefficient of variation of the VTF and AIF, respectively. Here, SD denotes either the standard deviation of the AIF (SD_{AIF}) or the VTF (SD_{VTF}). Accordingly, $\bar{t}$ represents the $\overline{T}$, the mean bolus arrival time or MVTT at the specific point in the model, respectively. The parameter RD is included in the analysis because it is believed to be a characteristic parameter of a particular vascular bed [173], since

$$SD \propto \bar{t} = \frac{V}{F}, \tag{6.3}$$

where F represents volume flow through a vascular volume V, i.e. the central volume theorem [85, 86]. This proportionality makes RD as defined in Eq. 6.2 a dimensionless constant, which in contrast to $MVTT$ and SD_{VTF} as well as $\overline{T}$ and SD_{AIF}, is independent of changes in dispersion due to variations of flow or vascular volume. It represents a measure for the spread of the VTF and the AIF that comprises influences on dispersion due to properties of the particular considered vascular bed. The parameter RD corresponds to the quantities $\text{RD}_{\text{art,artl,ven,venl}}$ that are used in the tissue perfusion model MMID4 described on page 42. Here, RD is analyzed in dependence of the travelled distance from the inlet and the vessel diameter where AIF_{disp} is assessed. This analysis is performed separately for the large coronary trees starting at the RCA, LAD and LCX as well as the full LCT.

6.2.5 Estimation of Perfusion Quantification Errors

The analysis of the errors in quantitative CE MRI perfusion measurements is conducted as outlined in Fig. 3.10, applying the tissue perfusion model MMID4. For the "at rest" simulations, the parameters for both tissue curve generation as well as MBF fitting with *Jsim* are chosen to be identical to what is described in Section 3.3.2 (page 42). Resting MBF (i.e., plasma flowF_p) is set homogeneous at 1 ml/min/g since no information about the total heartweight as well as the distribution of the myocardial mass to the different segments of the used dataset or pigs in general is available. From the segmentation of the myocardium (cf. Fig. 6.3, [170]) it cannot be concluded that all segments are equal in mass. On the contrary, they can strongly differ from each other also due to the variable thickness of the myocardial wall [175]. It is thus assumed, that a homogeneous MBF distribution across the myocardium best suits the datasets since no pathologies of the porcine dataset are recorded.

For the hyperemic state, two different approaches are used to assess the perfusion quantification errors. First, plasma flow F_p is adjusted to the mean overall increase of VBF in the LV and the RV under stress in comparison to the rest state (1 ml/min/g). This procedure assumes a healthy myocardium and allows for better comparability with previous studies, e.g., [23, 24]. During the second fitting with MMID4, F_p is set to the simulated MPR values of the associated myocardial segment in order to integrate the results from the Navier-Stokes simulations in the estimation of the perfusion quantification errors. Nonetheless, it must be borne in mind that the association of the model outlets to the different segments is critical in this regard. In both cases, the volume of the arterioles is doubled to $V_{artl} = 0.06$ ml/g to account for vasodilation of the microvasculature under stress compared to the resting state (cf. Section 2.2.1 [23, 87]). In contrast to Chapter 3, only fitting with four parameters is performed here.

6.3 Results

6.3.1 Analysis of Blood Flow

Volume Flow into the Myocardial Segments

In Fig. 6.5 the obtained mean absolute VBF values at rest and under stress as well as the resulting MPR per myocardial segment (cf. Fig. 6.3) are shown for three slightly different orientations of the cardiovascular 3D models from Fig. 6.1. Total mean VBF into the full left and right ventricular musculature amounts to $\sim$ 78 ml/min and $\sim$ 206 ml/min in the LV, and $\sim$ 20 ml/min and $\sim$ 45 ml/min in the RV, at rest and under stress, respectively.

From Fig. 6.5 and Table 6.1 it becomes obvious that the orientation of the 3D dataset is critical for the analysis of MBF heterogeneity. In particular, segments 3 and 19 show strong variability in dependence of the chosen orientation. This concerns the general amounts of VBF with which the segments are perfused as well as the obtained MPR and the supplying vessels. For example, in the original orientation of the dataset, the basal inferoseptal segment 3 is perfused solely by the RCA both at rest and under stress. The obtained MPR is comparatively low in comparison to the neighboring segments. If the dataset is rotated as it is shown in Fig. 6.5 e-g) and h-j), respectively, the major fraction of VBF comes from the LAD and not from the RCA anymore. Moreover, the obtained MPR value in segment 3 increases distinctly in the rotated datasets.

In the basal inferior RV segment 19, the changes due to the varying rotations are even more dramatic. In all three orientations, it is the segment with the lowest VBF values in the RV. In the original position, an unrealistic MPR < 1 (cf. Fig. 6.5 d) is obtained. With the applied rotations, this behavior is "improved" to 1.3 and 1.7, respectively. Overall, the rotation angles of the cardiovascular 3D models are adapted in order to yield a realistic

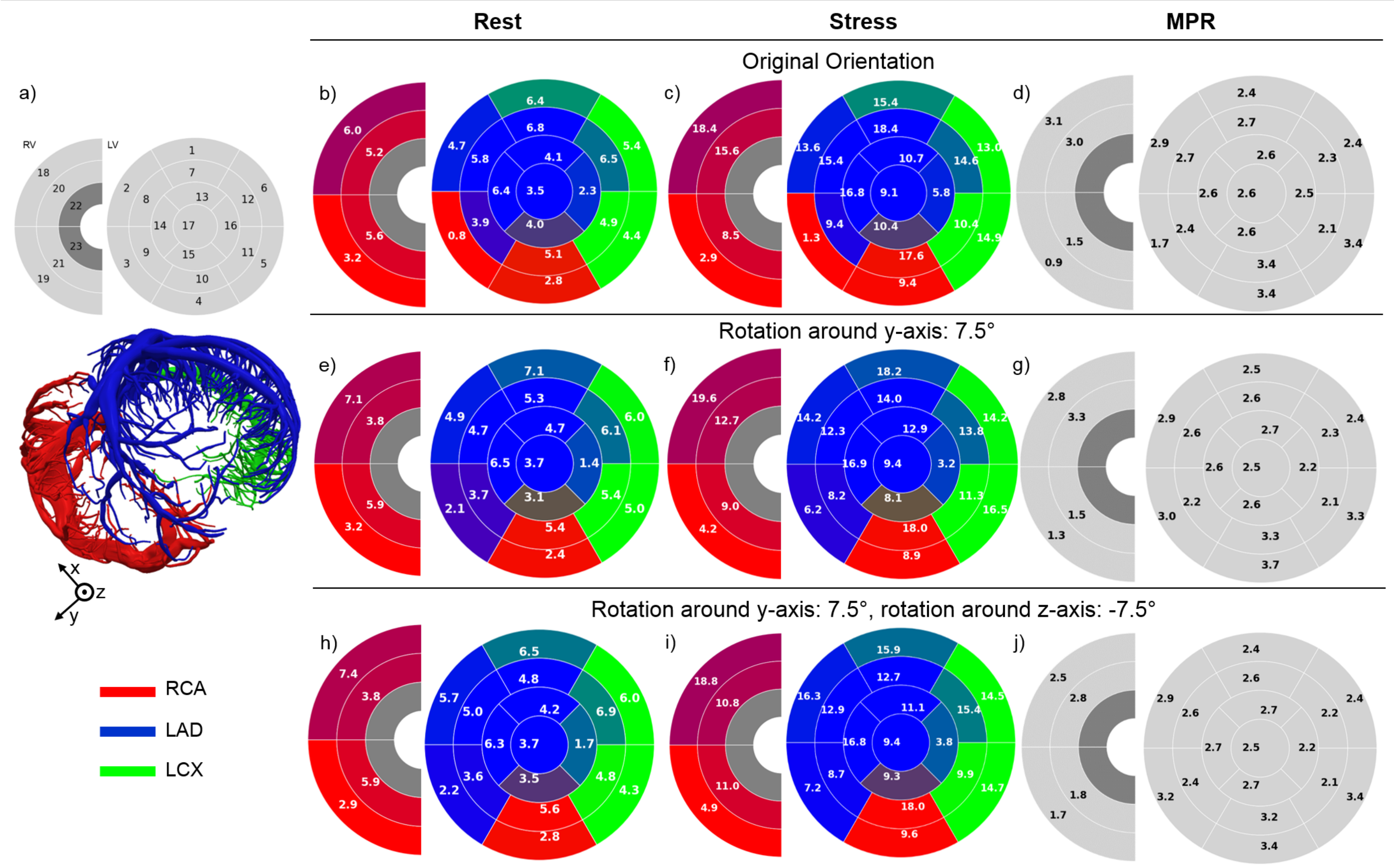


Figure 6.5: Mean rest (b,e,h) and stress (c,f,i) VBF (in ml/min) and MPR per myocardial segment for different rotations of the cardiovascular trees. Panel a) The numbering of the segments and the segmented 3D models. Panels b-d) Resulting resting and hyperemic VBF and MPR in the myocardial segments, if the original orientation of the dataset is retained. Panels e-g) The model outlets are assigned to the segments after rotation of the 3D models around the y-axis by 7.5°. Panels h-j) Resulting resting and hyperemic VBF and MPR in the segments after further rotation around the z-axis by $-7.5°$. The rotation angles are chosen to give physiologically realistic values of VBF and MPR. The coloring of the segments is chosen corresponding to the fraction of VBF coming from the LCX, RCA and LAD (cf. Table 6.1). In relation to the mass of the considered myocardial segment, VBF can be used to calculate MBF as defined by Eq. 2.56.

Table 6.1: Fraction of VBF per segment and vessel.

	No Rotation						y-Rotation						y, z-Rotation					
	Rest			Stress			Rest			Stress			Rest			Stress		
Segment	RCA	LCX	LAD	RCA	LCX	LAD	RCA	LCX	LAD	RCA	LCX	LAD	RCA	LCX	LAD	RCA	LCX	LAD
1	0	57	43	0	53	47	0	35	65	0	31	69	0	45	55	0	42	58
2	0	11	89	0	11	89	0	10	90	0	11	89	0	9	91	0	9	91
3	100	0	0	100	0	0	27	0	73	15	0	85	7	0	93	3	0	97
4	92	8	0	98	2	0	96	4	0	99	1	0	96	4	0	99	1	0
5	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0
6	0	100	0	0	100	0	0	97	3	0	98	2	0	100	0	0	100	0
7	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100
8	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100
9	23	0	77	11	0	89	24	0	76	12	0	88	8	0	92	3	0	97
10	91	9	0	95	5	0	94	6	0	97	3	0	99	1	0	99	1	0
11	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	2	98	0
12	0	43	57	0	40	60	0	42	58	39	61	0	52	48	0	48	52	0
13	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100
14	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100
15	29	22	49	31	24	45	38	33	29	40	34	26	33	20	47	35	22	43
16	0	17	83	0	13	87	0	27	73	0	23	77	0	40	60	0	35	65
17	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100
18	63	0	37	63	0	37	69	0	31	65	0	35	70	0	30	64	0	36
19	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
20	80	0	20	75	0	25	73	0	27	70	0	30	73	0	27	69	0	31
21	85	0	15	83	0	17	86	0	14	84	0	16	86	0	14	83	0	17

All values are in %, and they represent the fraction of VBF per segment, which comes from the LCX, the LAD or the RCT, respectively. The presented values are obtained for the different orientations of the dataset as shown in Fig. 6.5 for the rest an stress simulations. On the basis of these percentages, the color coding in Fig. 6.5 is defined. RCA→red; LCX→green; LAD→blue.

and physiologic distribution of VBF at rest and under stress thus also "optimizing" the obtained MPR. In all three orientations the results from the Navier-Stokes simulation show a heterogeneous distribution of VBF across the LV as well as the RV both at rest and under stress. Regional variations are similar for both hemodynamic states.

The VBF fractions coming from the different coronary trees (RCA, LCX and LD) reflect the location and orientation of the 3D cardiovascular models shown in Fig. 6.5 a) well. Furthermore, the general distribution of the perfusion areals of the three large coronary arteries (RCA, LAD, LCX) is in accordance with what is expected [170, 176, 177] in all three orientations of the dataset. However, with only two segments, the contribution of the RCA to LV perfusion is considerably smaller in orientations e-g) and h-j) than in the literature. However, these rotations yield a more balanced overall distribution of VBF than the original dataset orientation.

Generally, in all three dataset orientations, left ventricular segments, which could possibly be perfused by an overlap of two different vessels are largely perfused by the LAD. For example, segment 17 (Apex), which according to [177], would also receive blood from the RCT is solely perfused by the LAD in the results presented here. Similarly, segment 11 (mid inferolateral) would be expected to show a share of blood supply coming from the RCT, but is basically fully provided by the LCX in all three cases (cf. Table 6.1 and Fig. 6.5). In general, the LAD delivers the major share of VBF to the left ventricular musculature in the analyzed dataset. It should be noted that the arterial perfusion areals in the myocardium can vary strongly between different individuals and the distribution as it is obtained here is indeed physiologically reasonable. Nonetheless, it is pointed out once more that the association of the model outlets to the myocardial segments is chosen in order to best reflect physiological properties (a homogeneous distribution of MBF and MPR across both ventricles). As explained in Section 2.2.2, the LAD is oriented along the partition wall of left and right ventricle. Thus, vessels branching towards the right ventricle (cf. Fig. 6.5 c) perfuse the RV, which is reflected in the VBF fractions listed in Table 6.1 (cf. Fig. 6.5 b, e and h).

In general, the spread and the regional variability of the obtained VBF fractions across the LV is in the range of what is found in the literature for MBF measurements in pigs for all three orientations [178, 179]. In accordance with [56], the inferior segments (except for segment 10) show lower VBF fractions than the anterior or lateral segments (with the exception of the apical lateral segment 16). When considering all these values, it should be kept in mind, that they are highly influenced by the manual segmentation procedure, which is described in Section 6.2.1. With increasing distance from the model inlet, vessel sizes continuously decrease, which makes the segmentation process more and more susceptible to errors because vessels are simply overseen. Moreover, despite its very high quality and resolution, the used imaging cryomicrotome dataset is subject to local flaws making it impossible to accurately segment all vessels in the coronary trees. Explicitly, this is the case in the apical regions of the RCT where model segmentation is suspended, thus, most likely distorting VBF values in all segments, which are perfused by the RCT. Similarly, the comparatively small VBF fraction in the segment 16 can be explained by its very distal position in the pathways of both the LAD and the LCX.

Looking at the 3D model of the RCT in Fig. 6.5 c), it becomes obvious that generally fewer vessels are segmented in the RCT than in the LCT. This renders the obtained VBF values in the RV segments (as well as segments 4, 10 and 15 in the LV) less reliable. This is due to the above described weaknesses of the imaging cryomicrotome dataset, which are more pronounced in the region of the RV also resulting in a somewhat unphysiological shape of the large arteries (cf. Fig. 6.1 and Video 6-1 on the attached CD). It is uncertain, if these correspond to the real morphology of the dataset or if they are an artifact of the preparation procedure. Presumably, the reason for this is the smaller thickness of the RV (cf. Section

2.2.1), which makes it more vulnerable to changes of external influences during preparation. At the same time, the thicker myocardium of the LV is more robust.

Considering the calculated MPR, no measurements showing its regional heterogeneity in pigs exist; however, similarly strong variations across the LV are obtained in human MPR measurements obtained from MR [180–183] or PET [184, 185]. Nonetheless, the values obtained in segments 19 and 21 are below 2, which according to [180] is already pathologic and due to CAD. This could be a consequence of the reduced dataset quality in the regions of the RV, which result in seemingly pathologically altered models during the segmentation process. For this reason and since no real data is availabe about the perfusion reserve in the RV, above all for pigs, these values thus remain questionable.

A final circumstance that should be kept in mind when considering the obtained MPR values is the vasodilation of the arteries at stress (cf. Section 2.2.1). In the VBF curves assigned at the model inlets during the blood flow simulations this is considered as well as in the hemodynamic resistances that are assigned at the model outlets (cf. Eq. 3.3). However, since the 3D geometries of the coronary trees are modelled rigid and include vessels down to the pre-arteriolar level, vasodilation under stress is neglected in the cardiovascular models themselves. Since the outlet diameters vary between $\phi \sim 300 - 1000\,\mu\text{m}$, this means the effects of hyperemic vessel broadening distort the results at the model outlets and within the segments to different degrees.

Dependence of Volume Flow on Vessel Diameter

In Fig. 6.6, the diameter and volume flow ratios from Eq. 6.1 are visualized. On the abscissas, the ratio of the vessel diameter to the stem diameter of the considered coronary tree (LAD, LCX, RCT) is plotted. The obtained fitting parameters are listed in Table 6.2 for both fitting procedures for resting and hyperemic states.

The results show different behavior compared to what is found in the literature [36, 172] as well as when compared with each other. Fitting a function of type $f(x) = A \cdot x^B$ (cf. Eq. 6.1) with fixed B between the flow and diameter ratios ($f(x)$ and x, respectively) as it is done in [36] yields values $A < 1$ when both all vessels or solely the cross sections are considered for non-linear regression. However, this value more than doubles for all three coronary trees (RCT, LCX and LAD) if only the outlets are considered in the fitting procedure. Due to the smaller range of the considered diameters, the standard error of A in the fitting procedure when considering only the outlets is also considerably larger than for the other regressions. This increase suggests that in the CFD simulations presented here, VBF is distributed differently in the smaller vessels than what is predicted by Eq. 6.1. Namely, with increasing diameter, model outlets exhibit larger VBF than assumed from the hypothesis.

On the other hand, the fitting constant A is smaller than 1, if during non-linear regression with fixed B, only the cross sections or all vessels are considered. Due to the larger diameter range of cross sections, the obtained standard errors are distinctly smaller than for fitting with the outlet values only. Opposed to the behavior observed at the outlets, this means larger VBF in smaller vessels than the prediction from [36]. Generally, in this case, the obtained fitting parameter A is closer to the expected value of 1 in all three coronary trees; however, best accordance is obtained in the LCX ($A = 0.9$).

Considering the fitting procedure with both parameters of Eq. 6.1 left free to vary, the interpretation is more difficult. Overall, the behavior of A is similar to that when the regression is performed with fixed B. Higher values are obtained for fits using only the model outlets than for consideration of either all vessels or only the cross sections from Fig. 6.1. Generally, this fitting procedure thus suggests a better accordance of the simulations with the power law relationship from Eq. 6.1 for the smaller cross sectional and outlet diameter ranges than for the larger range when all vessels are used.

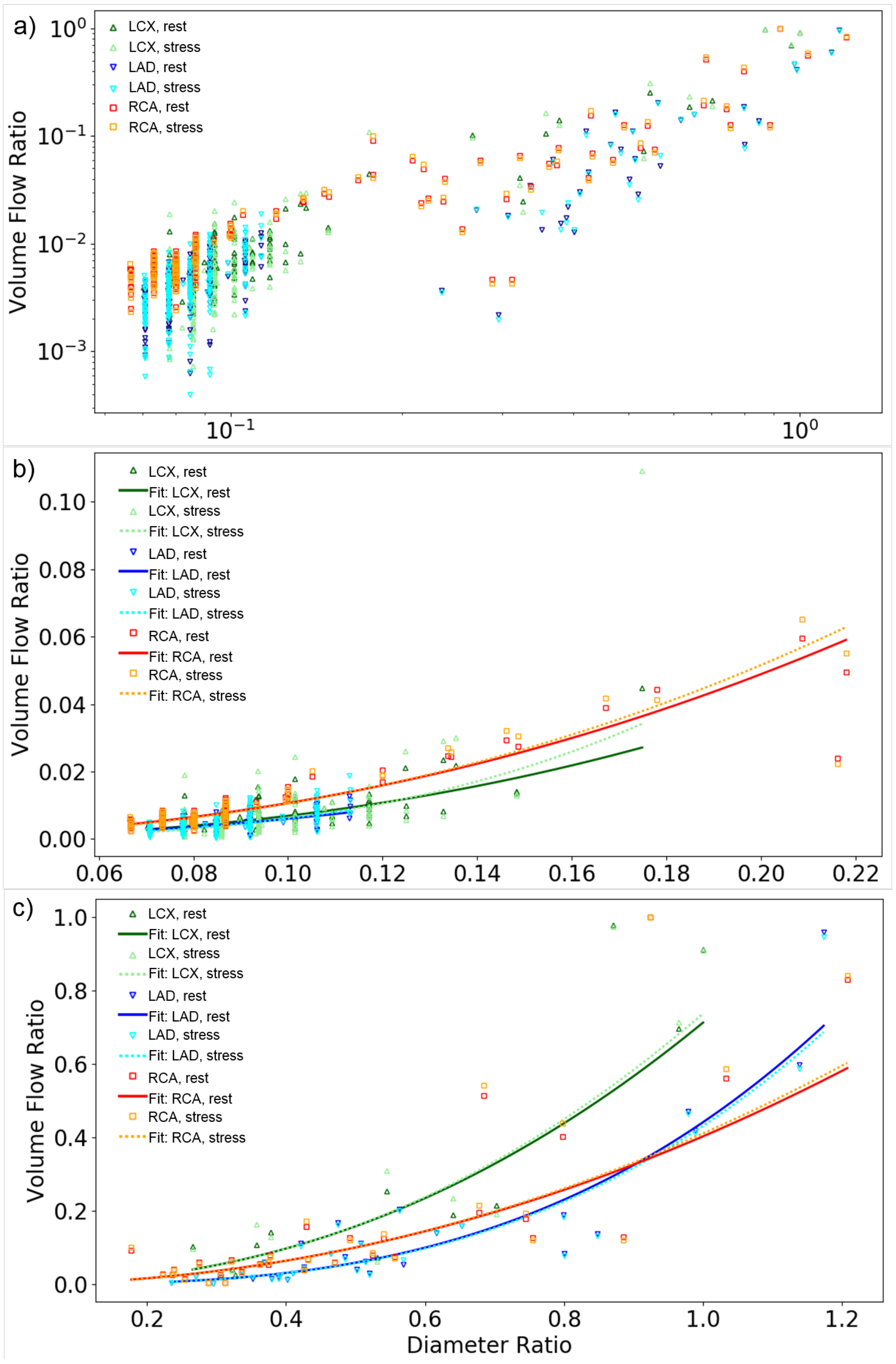


Figure 6.6: Flow diameter relation as determined from the simulation data (Eq. 6.1). Panel a) shows the obtained volume flow ratios including all cross sections and diameters of the models (cf. Fig. 6.1). Since the considered vessel diameters span several orders of magnitude, a logarithmic representation is chosen. In b), only the values obtained at the model outlets are plotted together with the curves from fitting both A and B. Accordingly, panel c) shows the relation obtained at the cross sections, together with the curves from fitting both A and B to the data (the fit for fixed B is not shown). All fitting parameters for both fitting procedures (A free to vary, B fixed; both A and B free to vary) are listed in Table 6.2.

Table 6.2: Obtained fitting parameters from regression for Eq. 6.1.

			A free, $B = 7/3$		A and B free		
			$A(SE)$	R^2	$A(SE)$	$B(SE)$	R^2
LCX	All	R	0.9 (0.02)	0.96	0.7 (0.2)	2.0 (0.1)	0.84
		S	0.9 (0.02)	0.95	0.8 (0.2)	2.2 (0.1)	0.75
	Outlets	R	1.9 (0.1)	0.76	2.0 (0.6)	2.5 (0.2)	0.44
		S	3.5 (0.3)	0.71	7.9 (0.8)	3.1 (0.3)	0.38
	CS	R	0.9 (0.1)	0.92	0.7 (0.3)	2.2 (0.4)	0.74
		S	0.9 (0.1)	0.92	0.7 (0.4)	2.2 (0.5)	0.69
LAD	All	R	0.5 (0.01)	0.94	0.2 (0.1)	1.6 (0.1)	0.80
		S	0.5 (0.01)	0.94	0.2 (0.1)	1.6 (0.1)	0.74
	Outlets	R	1.4 (0.1)	0.66	1.1 (0.6)	2.3 (0.2)	0.32
		S	1.8 (0.1)	0.64	3.4 (0.7)	2.7 (0.3)	0.29
	CS	R	0.5 (0.04)	0.93	0.4 (0.2)	2.9 (0.3)	0.77
		S	0.5 (0.04)	0.93	0.4 (0.2)	2.9 (0.3)	0.76
RCT	All	R	0.6 (0.03)	0.88	0.3 (0.1)	1.5 (0.1)	0.85
		S	0.6 (0.03)	0.88	0.3 (0.1)	1.6 (0.1)	0.84
	Outlets	R	1.8 (0.1)	0.90	1.7 (0.3)	2.2 (0.1)	0.79
		S	1.9 (0.1)	0.90	2.1 (0.1)	2.3 (0.1)	0.79
	CS	R	0.6 (0.1)	0.84	0.4 (0.3)	2.0 (0.3)	0.61
		S	0.6 (0.1)	0.84	0.4 (0.3)	2.0 (0.3)	0.60

"All" stands for all vessels in the considered coronary tree (LCX, LAD, RCT) and CS for the cross sections. The letters R and S denote resting or hyperemic state. The numbers in brackets denote the standard error (SE) of the fitting values A and B, respectively.

Comparing the values obtained at rest and under stress, it stands out that they do not deviate much from one another except for fitting when considering only the model outlets. In all three coronary trees the stress simulation yields higher fitting parameters A, or A and B, respectively. This behavior is most pronounced in the LCX. As explained above, this means that when only considering the model outlets in the fitting procedure, larger model outlets exert higher VBF in comparison to smaller outlets than what is predicted by [36, 172] (cf. Fig. 6.4). In all these considerations, it should be borne in mind that the fitting parameters obtained for fitting with only the model outlets, is possibly less reliable due the naturally smaller diameter range they span.

Finally, for the analysis of the relationship between VBF at stress and the vessel diameter, the same applies as what is described in Section 6.3.1. Due to the stiff modelling of the vessel walls, vasodilation of pre-arteriolar vessels wihin the 3D geometry is not considered. Presumably, this could be the cause for the steeper slopes of the fitting parameters under stress compared to the resting condition in Table 6.2.

6.3.2 Analysis of CA Transport

In this section, the question is investigated how the CA bolus changes its shape along the coronary arteries from Fig. 6.1. The analysis is performed with the help of the variables $MVTT$, SD_{VTF} and SD_{AIF} (Eqs. 2.88, 2.89 and 2.92, respectively). These de-

Table 6.3: Average $MVTT$, SD_{VTF} and SD_{AIF} along the large epicardial arteries from Fig. 6.7.

Artery	$MVTT\,(\sigma)$	$SD_{VTF}\,(\sigma)$	$SD_{AIF}\,(\sigma)$
LCT	2.5 (1.6) s	2.3 (1.2) s	4.6 (0.7) s
LCX	1.4 (0.7) s	1.3 (0.3) s	4.1 (0.1) s
LAD	2.8 (1.7) s	2.4 (1.3) s	4.7 (0.7) s
RCT	4.9 (3.3) s	3.7 (1.1) s	5.5 (0.8) s

All values are in seconds and the numbers in brackets denote the standard deviation σ around the average.

scribe the mean transit time of the CA starting from the model inlet, and the width of the VTF, which determines the change of shape of the AIF along the cardiovascular models (cf. Eq. 2.87). For better discrimination, the main branches of both models are named as shown in Fig. 6.7 a) and b).

In Fig. 6.7 c-h), the evolution of $MVTT$ and SD_{VTF} and SD_{AIF} is shown at the positions marked in Fig. 6.7 a) and b). $MVTT$ increases monotonously in all branches of both models. However, several vessels (LCT: diag2, diag3 and the terminal branches of the LAD; RCT: branch 1, 4 and 7) show increased $MVTT$ in comparison to the remaining branches. Similar behavior is observed, when SD_{VTF} and SD_{AIF} are analyzed along the different vessel paths. The branches with higher $MVTT$ increase also show higher SD increase at the same distances from the inlet.

However, SD_{VTF} and SD_{AIF} show qualitatively different behavior to the $MVTT$, as they decrease in some branches. For example, in the LCT, the second and fourth diagonal branch of the LAD (diag2, diag4) show distinct decreases in SD_{VTF} after ~ 35 mm and ~ 50 mm (red and brown arrows in Fig. 6.7 a,e,g), respectively. These distances coincide with the first cross sections after the bifurcations of the two branches from the LAD. Afterwards, SD_{VTF} increases again. On the other hand, the behavior in the main stem of the LAD (orange and blue data points in Fig. 6.7 c,e) at these points is opposing to this and shows stronger increase in SD_{VTF} just after the bifurcation. This is particularly apparent in the LAD after the bifurcation of diag4.

In the RCT, similar behavior is observed for SD_{VTF} in the second branch at ~ 30 mm, at the bifurcation of branches 1 and 2 (orange arrows in Fig. 6.7 b,f,h), with subsequent SD_{VTF} increase. This downstream increase of SD_{VTF} in branch 2 after the bifurcation is paralleled by a decrease of SD_{VTF} in branch 1. In both vessels, similar behavior is observed, when the SDs of the concentration time curves (SD_{AIF}) at the respective points are considered. However, the changes are not as pronounced.

Average values of $MVTT$ and $SD_{VTF,AIF}$ are listed in Table 6.3. In the LCT, maximum values of $MVTT$ = 6.6 s, SD_{VTF}=5.1 s, SD_{AIF}=6.4 s are obtained in the most distal LAD branch leading to the RV (cf. Fig. 6.7 c,e,g). It is observed that the maximum values in the branches of the LCX are smaller than in the LAD. In the RCT, maximum values are observed in branch 6 ($MVTT$=14 s, SD_{VTF}=7.1 s, SD_{AIF}=8.1 s, cf. Fig. 6.7 d,f,h). In other words, dispersion in the RCT is elevated in comparison to the LCT. This is due to the fact that in general, flow velocities in the LCT are markedly higher than in the RCT (cf. Fig. 6.2), and in the LCX tree higher than in the LAD tree. Higher flow velocities have effects similar as an increased diffusion coefficient, causing reduced dispersion [18, 23].

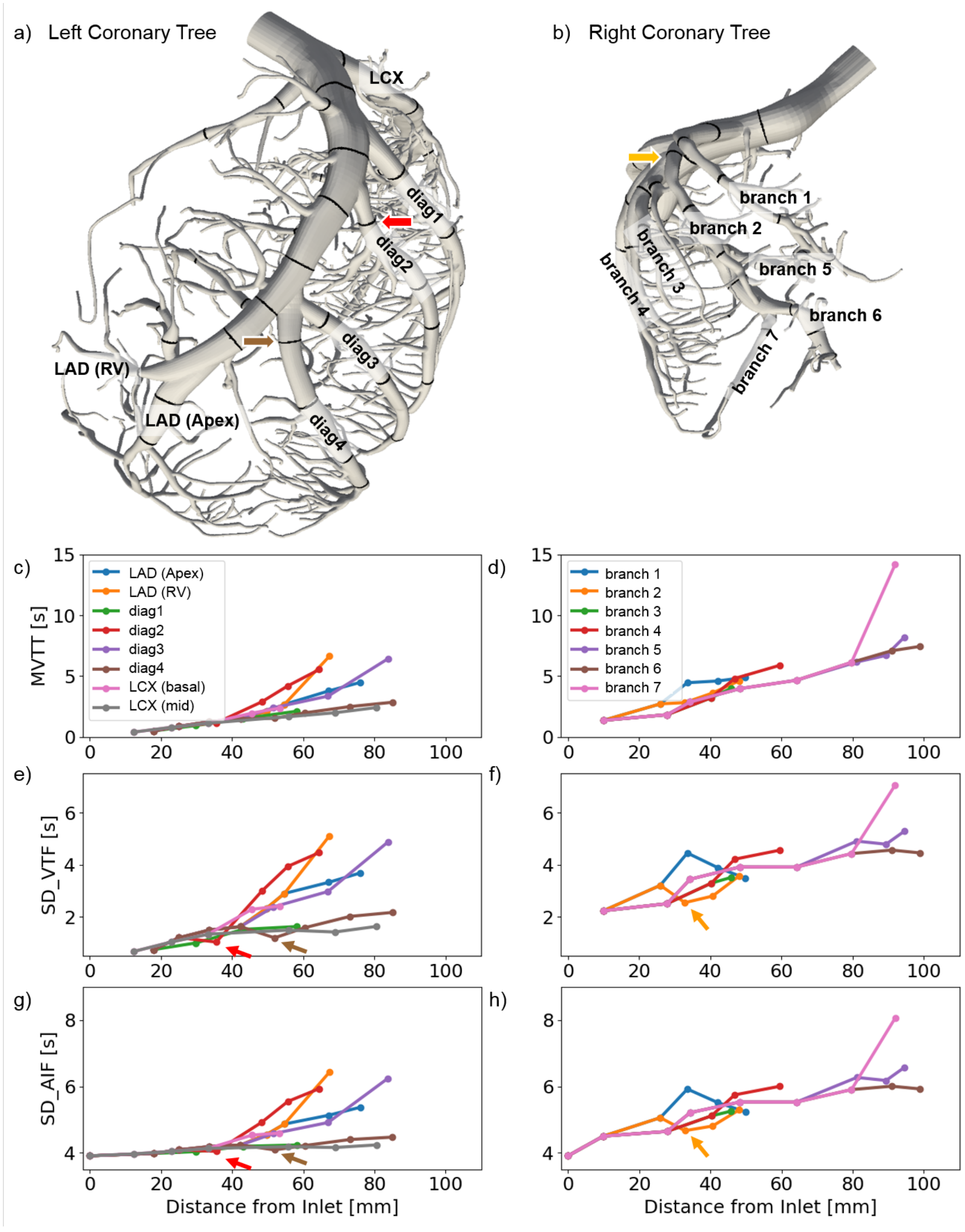


Figure 6.7: Evolution of $MVTT$, SD_{VTF} and SD_{AIF} along single branches of the 3D models. The data points represent the values obtained from Eqs. 2.88, 2.89 and 2.92 at the cross sections marked in panels a,b). Panels c,d) In both coronary trees, $MVTT$ increases monotonously. Panels e-h) Just behind bifurcations, at the positions marked by the red, brown and orange arrows SD_{VTF} (e,f) and SD_{AIF} (g,h) show distinct decreases in comparison with the remaining vessels forming the bifurcation.

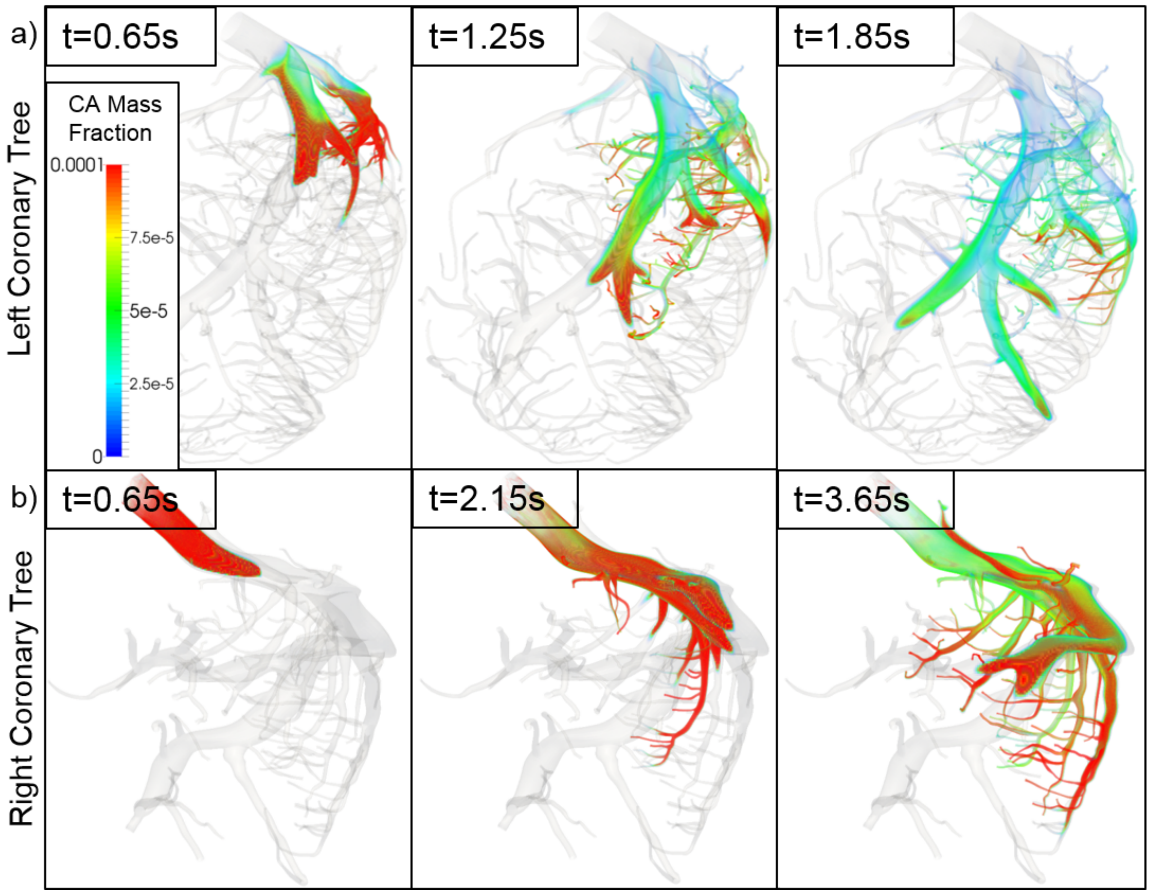


Figure 6.8: CA transport in the 3D models of the left and right coronary tree. Panel a) Anterior view of CA flow through the LCT. It becomes obvious, how CA distributes heterogeneously into the different branches of the LAD. Panel b) Inferior view of CA flow in the RCT (180° rotated from Fig. 6.1). Please note the slower CA transport in the RCT.

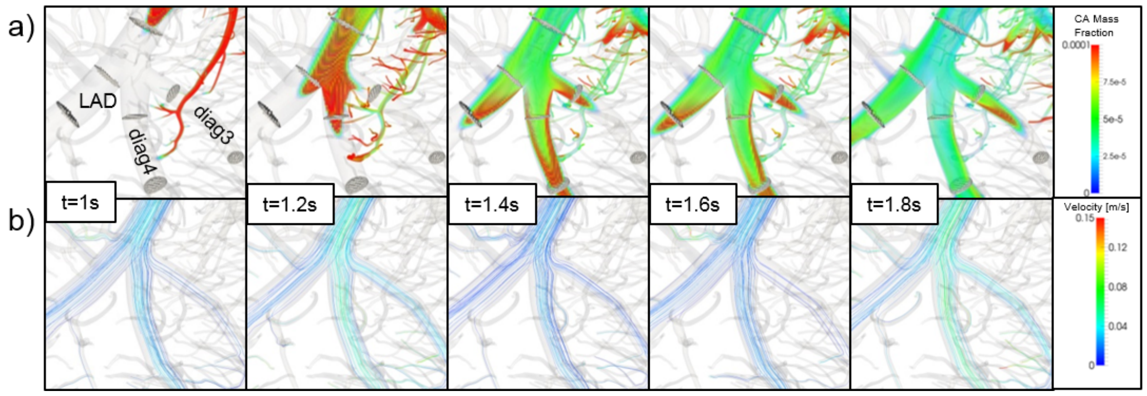


Figure 6.9: Anterior view of CA transport and velocity streamlines into diagonal branches. Panel a) CA first enters diag4, which bifurcates at a more beneficial angle from the LAD than diag3 with a steeper branching angle. Due to the curvature of the upstream LAD, CA assimilates at the side where the ostium of the branching vessels are located. The grey tiles in the path of the vessels represent the relvant cross sections from Fig. 6.1. Stripe patterns in the CA flow are artefacts from the underlying computational grid. Panel b) the seed of the plotted streamlines is a sphere at the center of the LAD at the upper end of the depicted model section. The streamlines show, how different intraluminal segments "feed" into the branching vessels.

CA Transport Mechanisms

In Fig. 6.8, CA transport through the two geometries is shown at different time steps. For better visualization, a shortened bolus (factor 100) is used here, compared to the real bolus, which is used as inlet BC in the transport simulations (cf. page 38). In the LCT, CA passes large parts of the LCX tree, before it reaches distal parts in the LAD tree. In the RCT, transport is considerably slower. Please also see the corresponding videos 6-2 and 6-3 of CA transport in the LCT and the RCT, respectively, on the CD that is attached to the thesis. In the LCT, after a short time, maximum concentrations are well below those, which are observed in the RCT (cf. Fig. 6.8, time step 1.85 s). In other words, the bolus front is spatially more spread in the LCT than in the RCT where it is more “focused”. This behavior is also a result of the higher flow velocities in the LCT, causing increased longitudinal diffusion-like effects [186].

In Fig. 6.9 a), CA transport at the bifurcations to diag3 and diag4 from the main branch of the LAD is shown. CA is transported faster into the fourth diagonal branch, while CA transport in the third diagonal branch is delayed. This behavior is also reflected in the decreased $MVTT$, SD_{VTF} and SD_{AIF} in diag4, compared to diag3 (cf. Fig. 6.7 a,c,e). Dispersion along the main branch of the LAD, the most unidirectional vessel, is also moderately increased, comparable to what is observed in diag3. For a better impression, please also see the corresponding Video 6-4 on the attached CD. In Fig. 6.9 b), velocity streamlines in the bifurcation are shown. Highest flow velocities are obtained in diag4 where the smallest increases of $MVTT$ and $SD_{VTF,AIF}$ are observed. Furthermore, streamlines feeding into this branch originate from central luminal segments of the mother vessel (LAD). The vessel diag3 is fed only by streamlines near the vessel wall of the LAD where flow velocities are smaller than at the vessel center.

The results depicted in Figs. 6.7, 6.8 and 6.9 reflect the influence of several different factors on CA dispersion, which were also observed and analyzed in detail in Chapter 5. Due to the inhomogeneous CA distribution across the vessel lumen, CA transport at bifurcations is strongly influenced by branching angles and the orientation of daughter vessels with regard to the curvature of the upstream vessel (cf. Fig. 6.8). As indicated by the streamlines shown in Fig. 6.9, CA from more central (i.e., faster flowing) intraluminal segments of the LAD is transported into diag4, which results in reduced SD_{VTF} and SD_{AIF} in this branch. In that respect, diag4 in the LCT and branch2 in the RCT have an “opportunist” position where CA transport takes place at a faster rate. Furthermore, flow velocities in the neighboring branches are also smaller than in the opportunist vessels. In accordance with [18], this results in additional bolus dispersion.

Relative Dispersion

The results depicted in Fig. 6.10 show different dependencies of $RD_{VTF,AIF}$ on vessel diameter and distance travelled by the CA in the coronaries. Qualitatively, the behavior does not differ strongly between the two hemodynamic states; however, the slopes of the linear relationships vary depending on rest and stress (cf. Table 6.4) and do not show agreement within the obtained error margins except for the LAD tree. For the full LCT as well as the LCX and the LAD trees, the slope of RD_{VTF} against the vessel diameter is steeper for the rest than for the stress simulation (cf. Table 6.4 a). In the RCT, this is reversed. Generally, the slopes are of the same order for all coronary trees, except for linear fitting only in the LAD where a much steeper slope is obtained. However, the correlation coefficient is also much weaker in the LAD tree, as well. Considering linear fitting of RD_{AIF} against the travelled distance, the behavior is largely equivalent in all coronary trees with steeper slope at rest (cf. Table 6.4 b).

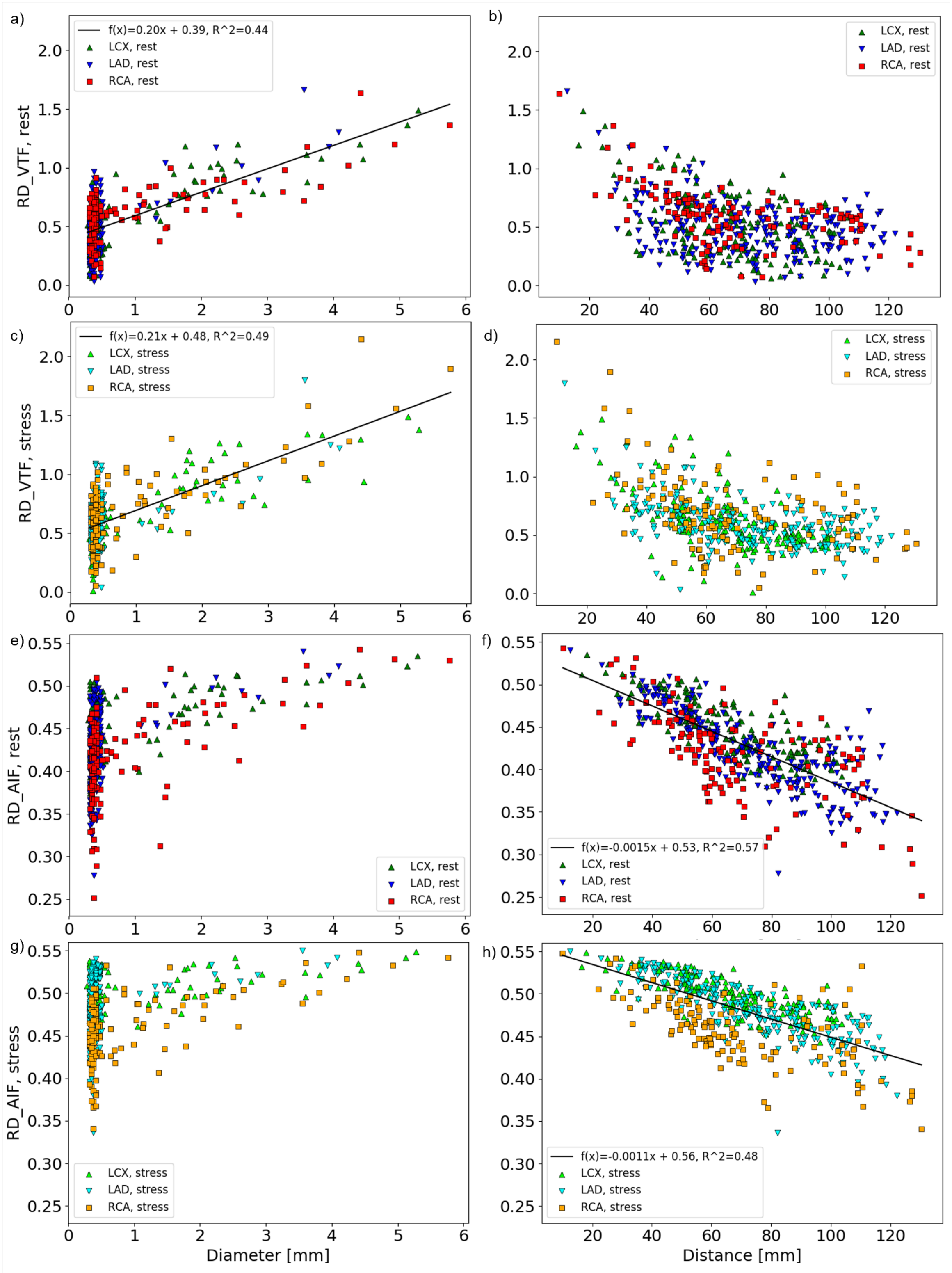


Figure 6.10: Dependence of RD_{VTF} and RD_{AIF} on vessel diameter and travelled distance at rest and under stress. Panel a-d) RD_{VTF} shows linearly increasing behavior in dependence of the vessel diameter for both hemodynamic states. Asymptotically decreasing behavior is obtained for RD_{VTF} dependence on the travelled distance. Under stress, the obtained RD_{VTF} appears to be generally higher. Panel e-h) RD_{AIF} spans a large range at small diameters ($\sim 0.25 - 0.50$ at rest and $\sim 0.35 - 0.55$ under stress) and asymptotically increases for larger vessels. A linear decrease of RD_{AIF} with the travelled distance is observed, with a steeper slope at rest than under stress. Results for linear fitting within each coronary tree are listed in Table 6.4 a) and b).

Table 6.4: Linear regression parameters and mean relative dispersion in the coronary trees.

a) Parameters for linear fitting of RD_{VTF} against the vessel diameter.

		Slope (SE)	Intercept	R^2
LCT	R	0.22 (0.01)	0.36	0.66
LCT	S	0.20 (0.01)	0.49	0.66
LCX	R	0.23 (0.01)	0.34	0.79
LCX	S	0.20 (0.01)	0.46	0.80
LAD	R	0.71 (0.25)	0.20	0.19
LAD	S	0.67 (0.22)	0.32	0.20
RCT	R	0.15 (0.01)	0.46	0.67
RCT	S	0.23 (0.02)	0.48	0.74

b) Parameters for linear fitting of RD_{AIF} against the travelled distance.

		Slope (SE)	Intercept	R^2
LCT	R	–0.0015 (0.6E–4)	0.54	0.65
LCT	S	–0.0011 (0.4E–4)	0.57	0.65
LCX	R	–0.0014 (0.8E–4)	0.55	0.65
LCX	S	–0.0009 (0.5E–4)	0.56	0.61
LAD	R	–0.0015 (0.7E–4)	0.53	0.66
LAD	S	–0.0012 (0.6E–4)	0.57	0.68
RCT	R	–0.0013 (1.3E–4)	0.51	0.44
RCT	S	–0.0010 (1.0E–4)	0.52	0.42

c) Mean relative dispersion of VTF and AIF.

		$RD_{VTF}\,(\sigma)$	$RD_{AIF}\,(\sigma)$
LCT	R	0.50 (0.25)	0.43 (0.04)
LCT	S	0.61 (0.22)	0.49 (0.03)
LCX	R	0.54 (0.30)	0.45 (0.04)
LCX	S	0.64 (0.26)	0.50 (0.02)
LAD	R	0.46 (0.19)	0.42 (0.04)
LAD	S	0.58 (0.17)	0.48 (0.03)
RCT	R	0.60 (0.23)	0.42 (0.05)
RCT	S	0.68 (0.32)	0.46 (0.04)

R and S stand for rest and stress simulation. In panels a,b), the values in brackets denote the standard errors of the fitted gradients (i.e., the standard deviations of the prediction errors) and the arithmetic standard deviations (σ) of the averages (c), respectively.

From a mathematical point of view, the general behavior of all six scatter plots in Fig. 6.10 can be analyzed using the definition of RD_{VTF} and RD_{AIF}, Eq. 6.2. With increasing distance from the inlet, $MVTT$ and $\overline{T}$ (cf. Eqs. 2.88 and 2.93) increase monotonously (cf. Fig. 6.7) and the plots suggest that the associated $SD_{VTF,AIF}$ (cf. Eqs. 2.89 and 2.92) does not increase with the same rate. It follows that RD_{VTF} and RD_{AIF} are reduced with travelled distance. Since naturally larger vessel diameters will always be found more proximally, both $MVTT$ and $\overline{T}$ are generally smaller there, yielding larger RD_{VTF} and RD_{AIF} at smaller distances and larger diameters. Similarly, under hyperemic conditions, higher flow velocities in the vessels result in reduced $MVTT$ and $\overline{T}$, which further cause increased relative dispersion (cf. Table 6.4 a,b).

The overall mean value (all vessels, rest and stress) of $RD_{VTF} = (0.57 \pm 0.25)$ is in accordance with available literature values of the cardiovascular bed, which show great variability, e.g. values (0.70 ± 0.07)[187] and (0.38 ± 0.05) [173] have been found. Similarly, the mean values for each coronary tree also correlate nicely with these literature values (cf. Table 6.4 c). However, while in those works, RD_{VTF} was calculated for the whole coronary system starting at the left atrium until the coronary sinus, in this work, a specific analysis of the RD_{VTF} for separate vessel segments and vessels of different sizes is performed. The effects of decreasing and increasing $SD_{VTF,AIF}$ in the vessel paths seem to even out and yield a mean RD_{VTF} of the same order as in [173, 187].

Under the assumption that RD_{VTF} is constant for a considered vascular bed, it can be deduced from Fig. 6.10 that larger vessels result in more dispersion than small vessels.[1] The decreased relative dispersion of both the VTF and the AIF (cf. Eq. 2.87) implies that $SD_{VTF,AIF}$ is not proportional to $MVTT$ (or $\overline{T}$, respectively) for varying vessel diameters.

The plots shown in Fig. 6.10 b) and d) are also a consequence of this non-proportionality between $SD_{VTF,AIF}$ and $MVTT$ and $\overline{T}$, respectively. With increased distance where vessel diameters become smaller and smaller, $MVTT$ (and $\overline{T}$) monotonously increase (cf. Fig. 6.7 a,b). However, due to the complexity of CA transport within the geometry, (cf. Fig. 6.8, 6.9 and the videos on the enclosed CD) SD_{VTF} and SD_{AIF} underlie strong heterogeneous influences along the vessel paths (cf. Fig. 6.7 e,f).

Overall, the linear relationships between RD_{VTF} and the vessel diameter as well as RD_{AIF} and the covered distance that are observed here can be interpreted to comprise of the degree of dispersion, which is obtained at the different points in the coronary trees. The linear increase of RD_{VTF} with vessel diameter (cf. Fig. 6.10 a,c) suggests more increase of CA dispersion in larger than in smaller arteries. Partially, this behavior is also the reason for the asymptotically decreased relationship between RD_{VTF} and the travelled distance, which is observed in Figs. 6.10 b,d). With longer travelled distances, the vessels naturally become smaller, which results in reduced additional dispersion, i.e., asymptotically decreasing RD_{VTF}. As analyzed in Figs. 6.8 and 6.9, this reduced additional dispersion in smaller vessels is by no means exclusively due to the reduced vessel sizes. On the contrary, these findings confirm the observations from Chapter 5, i.e., that CA bolus dispersion is strongly heterogeneous and depends on several different factors, such as vessel bifurcations, branching angles as well as intraluminal CA concentration heterogeneities due to the general shape variations of the vessels in the coronary trees

As the shape of the VTF determines the shape of the AIF (cf. Eq. 2.87), changes in RD_{VTF} should manifest in the behavior of RD_{AIF}. In Fig. 6.10 e) and g), RD_{AIF} shows a large spread for small vessel diameters. This can be explained by the high variability of the travelled distances in the small arteries, from which the data points stem. On the other hand, in the larger arteries RD_{AIF} is generally larger. As explained above, this is due to the naturally more proximal location of larger vessels within the coronary tree and thus smaller

[1]It should be kept in mind that the RD_{VTF} values obtained in the smaller arteries comprise the contribution from the larger upstream vessels.

Table 6.5: Obtained $\Delta\text{MBF}_{\text{Rest}}$, $\Delta\text{MBF}_{\text{Stress}}$ and ΔMPR averaged over all outlets and cross sections in the large coronary arteries.

Artery	$\Delta\text{MBF}_{\text{Rest}}$ [%]	$\Delta\text{MBF}_{\text{Stress}}$ [%]	ΔMPR [%]
All	-28 ± 16, (-0.7)	-11 ± 12, (-2.0)	-26 ± 22, (1.6)
LCT	-22 ± 14, (-1.1)	-7.5 ± 7.6, (-2.4)	-24 ± 20, (1.9)
LCX	-17 ± 8, (-0.3)	-4.7 ± 3.5, (-1.5)	-16 ± 8.8, (0.8)
LAD	-27 ± 16, (-0.8)	-9.3 ± 8.8, (-2.0)	-29 ± 24, (1.4)
RCT	-41 ± 14, (-0.2)	-23 ± 14, (-1.1)	-34 ± 22, (1.0)

Errors are the arithmetic standard deviations around the mean value. The value in brackets represents the skewness (i.e., the third momentum of the distributions), which is an indicator for the asymmetry of the distribution (unitless quantity).

$\overline{T}$. Differences between rest and stress are also less pronounced in the larger than in the smaller arteries. Generally, the increase of RD_{AIF} at larger diameters correlates nicely with the behavior of RD_{VTF} observed in Fig. 6.10 a) and c). As expected from the convolution of the AIF in the LV and the VTF (Eq. 2.87), which determines CA transport in a vascular system, stronger VTF dispersion (i.e., increased RD_{VTF}) yields larger CA bolus broadening (increase of RD_{AIF}). The decrease of RD_{AIF} with the travelled distance also reflects this general relationship between the VTF and the AIF.

6.3.3 Estimation of Perfusion Quantification Errors

As described in Section 6.2.5, two separate approaches to estimate the errors in perfusion quantification are taken. The results from the approach where homogeneous MBF and MPR distributions across the myocardium are assumed is described in the following.

Homogeneous MBF and MPR

On average, the resulting values of $\Delta\text{MBF}_{\text{Rest}}$ and $\Delta\text{MBF}_{\text{Stress}}$ spread over a large range in the analyzed geometries (cf. Table 6.5). Since $\Delta\text{MBF}_{\text{Stress}}$ values are smaller than $\Delta\text{MBF}_{\text{Rest}}$, the resulting ΔMPR values are also subject to great variability. Considering the three large coronary territories, errors are largest in the RCT where the smallest flow velocities occur, and smallest in the LCX, which is passed fastest by the CA (cf. Fig. 6.8 and Videos 6-2 and 6-3). In all vessels, ΔMBF is slightly left skewed (skewness < 0) with increased skewness at stress in all vessels. Correspondingly, the obtained MPR distributions are all right skewed, with the weakest skewness in the LCX.

In order to analyze the quantifcation errors with regard to the travelled distance, the maximum distance range is subdivided into 6 segments of ~ 2 cm length each. The results for mean $\Delta\text{MBF}_{\text{Rest,Stress}}$ and ΔMPR in these segments are shown in Fig. 6.11. A clear trend to larger perfusion quantification errors with increasing travelled distance is obvious. As a consequence of the decreasing relationship between RD_{AIF} and the travelled distance (cf. Fig. 6.10 d), one would expect asymptotic behavior of $\Delta\text{MBF}_{\text{Rest,Stress}}$. As can be seen in Fig. 6.11, this behavior can roughly be made out if the last two distance segments of the coronary trees are considered (segments 6-8 and 8-10 for the LCX tree, and 8-10 and 10-12 for the remaining trees) for the simulations at the resting state. Weaker additional MBF underestimation is obtained between these two segments. However, looking at the results for the stress simulations, the relative increase between the last two distance segments is comparably higher. This results in what can be interpreted as a starting asymptotic behavior of ΔMPR towards larger travelled distances, in accordance with [24]. Comparing the

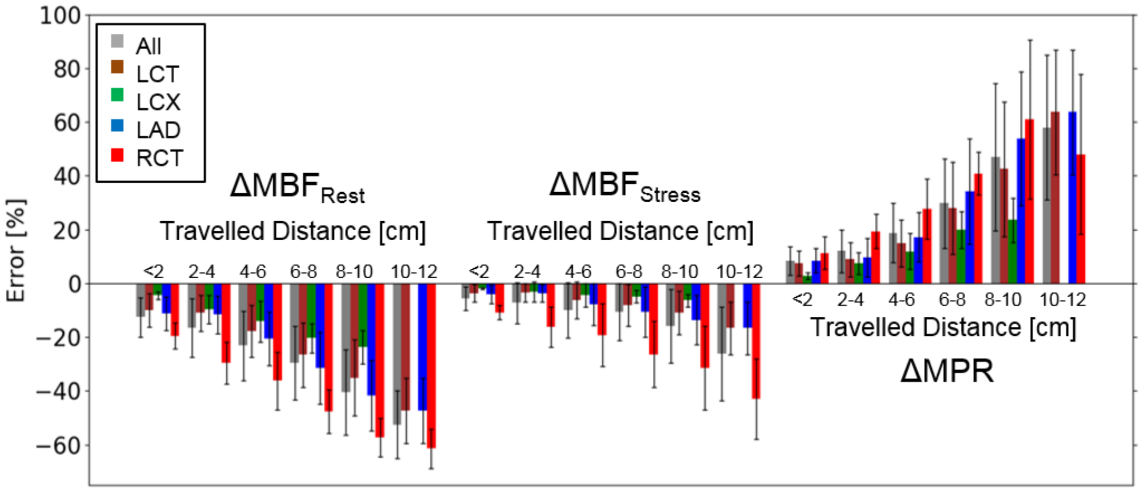


Figure 6.11: Increasing mean ΔMBF and ΔMPR at different distances from the model inlet for the homogeneous MPR. No value for the LCX lies in the last distance segment.

behavior of the rest and hyperemic perfusion quantification errors with the results obtained for RD_{AIF} from Fig. 6.10 f) and h), this is expected. The flatter slope of RD_{VTF} with regard to the travelled distance means that in the stress simulation, the relative contribution of distal vessels to CA dispersion is increased in comparison to the rest simulation, even though in general, CA dispersion and subsequent MBF quantification errors are smaller for higher flow velocities (i.e., under stress).

Heterogeneous MPR

After the estimation of the perfusion quantification errors under the assumption of a homogeneous MPR distribution, here follows the second approach described in Section 6.2.5. Perfusion quantification errors are analyzed using the heterogeneous MPR distribution from Fig. 6.5 j).

In Fig. 6.12 the evolution of mean ΔMBF and ΔMPR is shown in dependence of the distance to the model inlet in the three large coronary trees. The errors of resting MBF are identical to those in Fig. 6.11 since the same homogeneous resting plasma flow $F_p = 1$ ml/min/g is assumed. On the other hand, the errors of MBF quantification under stress and following MPR quantification show larger variability. This particularly concerns the mean values obtained in the RCT and the LAD tree in the first three distance segments. Generally, error margins are much larger than for the homogeneous analysis, suggesting an increased influence on the perfusion quantification errors due to the underlying varying MPR values as in the homogeneous analysis. Nonetheless, an overall increasing behavior of $\Delta \text{MBF}_{\text{Stress}}$ and ΔMPR with the travelled distance is obtained.

In order to separate the influence of the heterogeneous MPR distribution, Fig. 6.13 shows the mean MBF and MPR quantification errors obtained in each myocardial segment. Strong regional variations are obtained and segments with higher simulated MPR also show larger ΔMPR. Accordingly, segments 19 and 21, the segments with the lowest simulated MPR values show the strongest MBF underestimation under stress. If compared to segments 1 and 3 or 5, respectively, which lie at similar distances from the model inlets, it becomes obvious how smaller assumed values of F_p yield increased underestimation of perfusion. In the basal inferior segment 19, $\Delta \text{MBF}_{\text{Stress}}$ is even lower than $\Delta \text{MBF}_{\text{Rest}}$, resulting in a negative ΔMPR. This is exceptional for all results obtained in this work as well as previous studies.

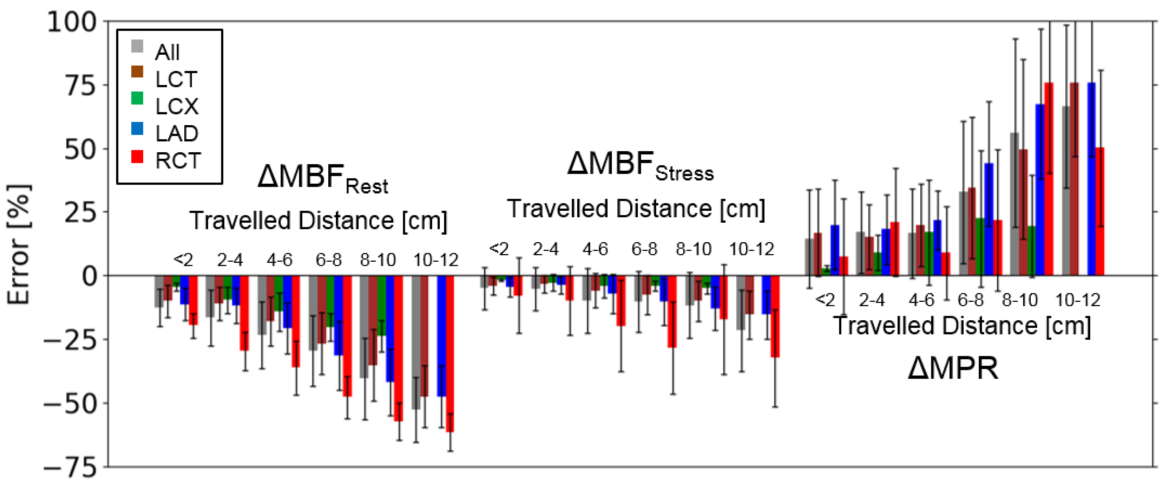
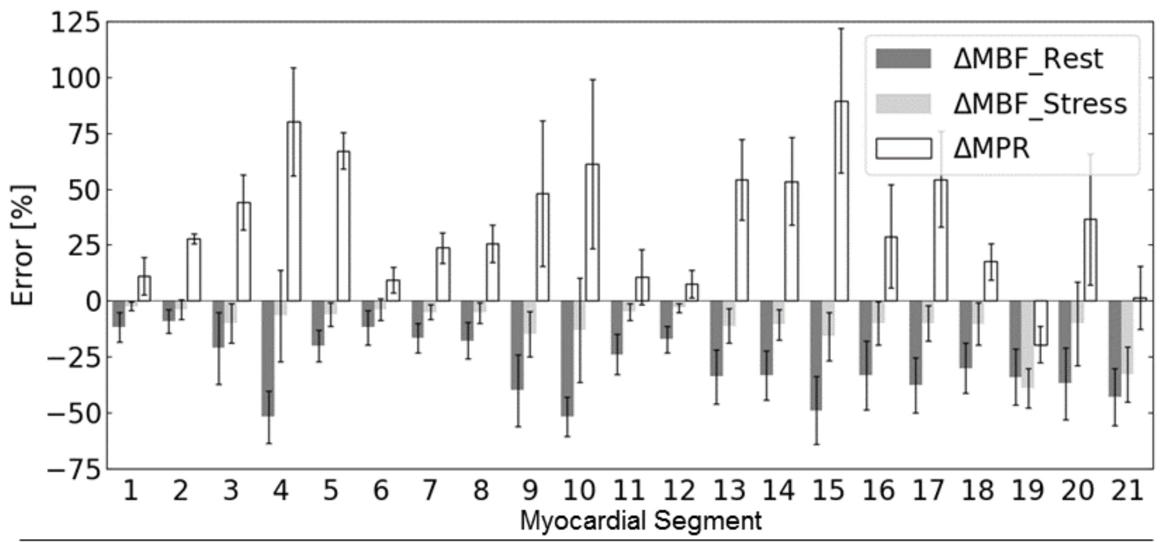


Figure 6.12: Increasing mean ΔMBF and ΔMPR at different distances from the model inlet for the heterogeneous MPR. As in Fig. 6.11, no value for the LCX lies in the last distance segment.



MPR	2.4	2.9	3.2	3.4	3.4	2.4	2.6	2.6	2.4	3.2	2.1	2.2	2.7	2.7	2.7	2.2	2.5	2.5	1.7	2.8	1.8
MD	4.6	4.0	4.8	9.8	7.2	5.3	6.0	5.8	8.1	10.5	8.8	7.1	8.2	8.9	10.8	9.5	10.8	5.2	4.8	7.0	6.7
σ	0.8	0.5	2.9	1.2	1.0	0.7	0.7	0.8	1.9	0.7	0.9	1.0	0.8	1.0	0.7	0.7	0.7	1.1	1.2	1.7	1.3

Figure 6.13: Mean ΔMBF and ΔMPR in the myocardial segments based on the heterogeneous MPR. Below the numbering of the myocardial segments, the MPR values from Fig. 6.5 j) are listed. MD represents the mean distance to the outlets in the segment and σ the standard deviation around MD, both in cm. Perfusion quantification errors are strongly heterogeneous between the myocardial segments.

The results in Fig. 6.13 also confirm the observations from Fig. 6.12 that increased travelled distances lead to increased perfusion quantification errors. Comparing $\Delta\text{MBF}_{\text{Stress}}$ and ΔMPR obtained in segments 7 and 15, with a similar MPR (2.6 and 2.7, respectively), however, different distances (6.0 and 10.8 cm, respectively), a distinct increase of the perfusion quantification errors is obtained in segment 15.

6.4 Discussion

In summary, the work discussed in this chapter presents for the first time a detailed analysis of blood flow and CA transport through both left and right coronary artery down to the pre-arteriolar level using an advanced coronary flow BC.

The examination of VBF in the different myocardial segments yields regional variations, which are in good accordance with literature values [170, 176–179]. This applies both for the territories being supplied by the large coronary arteries (RCA, LAD and LCX) as well as the fractions of VBF flowing into the segments. Even though distortions from segmentation errors can be recognized, overall, physiologically realistic results are obtained. However, it must be kept in mind that the analysis presented here is solely based upon the rheology of blood flow. Tissue demands, which also play a decisive role in blood flow distribution, are not considered.

Regarding the dependence of VBF on vessel diameter, an acceptable agreement with the hypothesized volume scaling law from [36, 172] is found. Overall, it can be stated that it is best fulfilled in the LCX for both fitting procedures if either only the cross sections or all vessels are included. However, it must be noted that the observations from the CFD simulations of blood flow presented here cannot fully confirm the expected behavior. A major difference between this work and [36, 172] is the relative diameter range that is considered in the analysis. While here only 2 orders of magnitude are considered, the analyses in [36, 172] are based on morphological data down to diameter ratios of 10^{-4}. The results from the hemodynamic analysis in [172] are based on a network flow analysis in which a simple symmetric model was used where all the vessel elements in any order are assumed to be of equal diameter and length, and arranged in parallel. Vessel resistances in [172] are approximated by Poiseuille's law $R = {}^{(\mu l)}/_{(\pi D^4)}$ for laminar stationary flow [188], where μ is the fluid's viscosity, l the length of the vessel segment and D its diameter. Furthermore, blood pressures at all of the junctions between different vessel orders are assumed equal. In comparison, the CFD approach used in this work allows for an analysis in a highly asymmetric cardiovascular tree (cf. Fig. 6.1) per se taking account of flow effects in the curved and tapering vessels. Considering this, the findings from this work are highly relevant and the accordance with [36] is promising, nonetheless. However, in order to fully understand the underlying mechanisms of blood flow in the microvasculature, a more profound and dedicated analysis towards even smaller vessels might be useful.

The results presented in Section 6.3.2 show how CA dispersion in the coronary trees is influenced by several different factors. All three quantities used to assess and quantify CA dispersion ($MVTT$, SD_{VTF}, SD_{AIF}) reflect considerably stronger dispersion in the RCT than in the LCT where more dispersion is observed in the LAD than in the LCX tree (cf. Table 6.5). In accordance with [18, 23], higher flow velocities have similar effects as an increased diffusion coefficient, which causes reduced dispersion. In all three coronary trees, $MVTT$ increases monotonously, however, rising at distinctly higher rates behind several bifurcations. Analogous to the analysis presented in Chapter 5, strongly heterogeneous behavior (increasing and decreasing) of SD_{VTF} and SD_{AIF} is observed at bifurcations of different angles and orientations due to the inhomogeneous CA distribution across the vessel lumen.

The findings show that RD_{VTF} can be considered a characteristic value of the considered vascular bed, depending on the travelled distance and the vessel diameter. From the analysis of RD_{VTF} and RD_{AIF}, a decreased influence of smaller vessels on CA dispersion can be assumed; however, this is not solely due to the vessel size. It is rather the complex transport phenomena due to the general shaping of vessels and bifurcations, which lead to the reduced additional dispersion observed in smaller arteries. Overall, the mean values obtained for RD_{VTF} (0.57 ± 0.25, Table 6.4 c) in this work are in good agreement with the literature [173, 187].

The subsequent estimation of perfusion quantification errors yields values that are in the range of what is found in previous studies[18, 22–25]. A clear tendency to increasing errors with higher travelled distances is observed for both homogeneous as well as heterogeneous perfusion reserve. Due to the reduced additional dispersion in smaller vessels under stress in comparison to the resting state, MPR quantification errors appear to be less pronounced at more distal positions. However, this observation requires further validation on even more detailed cardiovascular geometries. Moreover, it must be born in mind that the 3D geometries used in this work are not vasodilated in the stress simulations.

The results from the estimation of the perfusion quantification errors using the heterogeneous MPR distribution from the blood flow simulations yield results, which only partly corroborate the findings from the previous analyses with a homogeneous MPR. They reveal important information that the assumed values of MBF (i.e., plasma flow F_p) when fitting with MMID4 (*Jsim*, cf. page 42) play a decisive role for the estimation of perfusion quantification errors. A detailed analysis of this particular parameter as well as the remaining constraints in the fitting procedure should be integrated in future studies.

Consequences for MRI Perfusion Measurements

An indirect validation of the CFD simulations is possible by comparison of the estimated MBF errors to experimental data obtained by use of the microsphere method in animal measurements [189] as well as clinical data from PET and CT measurements of myocardial perfusion in humans [8, 9]. Particularly, the microsphere method is assumed to give correct account of MBF distribution since it is not dependent on the acquisition of dynamic data and, thus, dispersion-related quantification errors can be excluded. Hence, this technique is also referred to as the experimental gold-standard for perfusion quantification, despite known limits in the technique's spatial resolution [190]. In fact, in several studies an underestimation of MBF by MRI perfusion measurements was observed [5, 13, 191–193]. However, obtained values were dependent on different factors, such as the utilized quantification algorithms [194] or the applied MRI signal intensity corrections [195]. Moreover, most authors have also observed stronger underestimation of MBF by MRI quantification in comparison to microsphere perfusion measurements for increased MBF [195, 196]. Yet, this could also be a consequence of systematic MBF overestimation by microspheres in regions of high flow and underestimation in regions of low flow as it was observed by Bassingthwaighte et al [197]. In addition to this, it should be pointed out that the observed deviations between microsphere and MRI measurements varied considerably between the different studies and were partially even found to be negligible [89]. The same applies for studies where a comparison of MBF quantification by MRI and PET was performed. For example, Miller et al [15] compared several MRI quantification algorithms and obtained an increased MBF underestimation by MRI in comparison to PET for larger MBF values for all the applied quantification methodologies. Overall, in [15, 198] a general underestimation of MBF by MRI is observed. On the other hand, Fritz-Hansen et al (85) only found MBF underestimation by MRI at stress in comparison to PET quantification and good agreement for resting conditions.

A reason for these variations could lie in the applied measurement setups, regarding MRI protocols, CA injection technique and site as well as the location of AIF quantification. As discussed in the presented work, CA dispersion is also strongly influenced by the physiological (e.g., resting or stress conditions, i.e., the regime of blood flow velocities) as well as the morphological conditions (e.g., vessel branching and sizes), not to mention pathological alterations (e.g., stenoses and occlusions). In the studies mentioned above, the influence of these parameters was not systematically investigated. Moreover, they did not provide a detailed comparison of MBF quantification by MRI and microspheres in dependence of the proximity of the analyzed segment to the coronaries' orifices. In other words, the distance the CA travelled between the locations of AIF and MBF quantification, which is identified as one of the major determinants of CA dispersion in the presented study, is not considered. Accordingly, future studies should comprise of a possibility for direct validation of the CFD simulations. This could be achieved by parallel MBF quantification by MRI and microspheres in addition to the acquisition of morphological information of the coronary vasculature as well as precise monitoring of the hemodynamic and physiological conditions during the measurements.

Limitations and Outlook

The volume flow BC applied at the model inlets is computed by the model described in Section 3.2 based on generalized pressure curves. Precise measurements of pressure and flow curves (e.g., by Doppler or MRI measurements [199, 200]) to perform CFD simulations with intraindividual BCs would allow validation and benchmarking of the used BC. The assumption of rigid vessel walls is a common simplification, and thus, elastic reactions of the coronary arteries on the fluid pressure are not considered [201, 202]. Though myocardial compression and relaxation is comprised in the assigned inlet volume flow curves, tissue pressure acting from the outside on the vessels is not included. Particularly in the smaller endocardial arteries, this can lead to complete vessel collapse [203]. In future studies, these fluid structure interactions (FSI) should be integrated [204, 205].

At the model inlets, a γ-variate distribution as it is measured in the LV is assumed, neglecting bolus dispersion between the LV and the coronaries' orificies in the aorta. Moreover, with this approach, a homogeneous CA distribution is applied across the whole inlet lumen. As a consequence, effects of heterogeneous CA distribution across the aortic lumen on CA inflow at the inlets of the 3D models of the LCT and he RCT are not considered (similarly to what is analyzed here in the coronaries). In order to correct and recover this, the simulations could be extended by integrating a segment of the aorta.

Besides this, the used CA time curve describes the CA mass concentration in blood as a soluble not changing its density. Thus, influence of effects such as friction, size and shape of the transported particles on the blood stream and subsequently CA transport should be subject of future analysis.

The findings from this chapter contain important implication for all bolus-based methods of perfusion quantification (e.g., MRI, PET, CT) as well as particle or mass transport in the blood stream in general. The above approach shows that CFD analysis is capable of showing and predicting how mass is transported through the coronary arteries. However, the complexity of this whole process is emphasized, making a general statement about CA dispersion in individual coronary trees impossible.

Conclusion

The results presented in this chapter suggest three factors being the major determinants for prediction of CA bolus dispersion and subsequent estimation of perfusion quantification errors:

- The distance travelled by the CA bolus,

- The regime of flow velocities in the considered vascular tree, and
- The assumed plasma flow into the tissue.

Similar to previous CFD studies [18, 23, 24], the first two parameters appear to be the best indicators to be used for benchmarking of CE MRI perfusion measurements. The analysis presented here suggests that these two quantities should be central to any error correction scheme. Such a framework would, however, still be subject to large uncertainties, which increase for lower flow velocities and at larger distances as well as in dependence of the underlying MBF and MPR values.

Chapter 7

Summary and Outlook

In this thesis, a CFD analysis of CA transport in a cardiovascular model of unprecedented detail down to vessels at the pre-arteriolar level is performed in order to verify and validate insights from previous studies [18, 23, 24]. For this purpose, an extended and improved boundary condition is devised, which allows for an extensive investigation of blood flow and CA transport in the coronary arteries, ultimately providing an in-silico model of the arterial epicardial vasculature. The results from these analyses reveal several different aspects influencing the observed dispersion, including the type of CA, flow velocities and vessel curvature. Moreover, they underline the importance of the careful choice of BCs in the simulations, which are validated with regard to several physiological aspects. These include the territories perfused by the large coronary arteries or the volume flow in dependence of the vessel diameter. However, to make the analysis in the highly detailed models treated in this work feasible, several different aspects (e.g., model preparation, file management, computation times) need to be considered.

7.1 Summary

In order to guarantee the physiological relevance of the simulations, in Section 3.2, an innovative dedicated BC is implemented, which has not been used previously in this context. For the purpose of time-efficient execution of the CFD computations, an alternative volume discretization procedure is introduced (cf. Section 3.3), which is subsequently benchmarked in a mesh convergence analysis (cf. Section 3.4). The high computational demands of the numerical calculations make their execution on High Performance Computing (HPC) clusters necessary, which poses specific requirements on file management and parallelization. These issues are tackled in Sections 3.5 and 3.6. Overall, the preparatory tasks presented in Chapter 3 make the analysis of this thesis possible in the first place.

In Chapter 4, the validity of the chosen BCs of the blood flow simulations (Section 3.2 and page 36 ff) is benchmarked by a CFD analysis of the Fractional Flow Reserve (FFR), a clinical parameter to assess the severity of coronary stenoses. In comparison to the invasive measurement of the pressure drop across a coronary stenosis, the CFD simulations yield results, which lie in the same range.

In the following chapters, the analysis is complemented by a methodological study of CA transport in coronary arteries of vessel generations 3 to 7 (Chapter 5) and a comprehensive analysis of the results of both blood flow and CA dispersion in the left and right coronary tree (Chapter 6). This analysis starts at the orifices of the LMCA and the RCA at the Aorta and includes vessels at pre-arteriolar level (diameter $\sim$ 300μm). The physiological correctness of the blood flow analysis can be seen as a confirmation of the proposed approach from Section 3.2 and its ability to realistically model coronary blood flow. Both with regard to a previously hypothesized volume-diameter relationship [36]as well as the myocardial territories, which can be associated to the coronary trees, the results are in good agreement with the literature [170, 176, 177]. Overall, the general shapes of the inlet volume flow curves are in good agreement with normal coronary flow curves [101, 158]. However, variations on shorter timescales do not occur as in [206, 207] where the coronary vasculature is represented by a simplified 1D approach, not specifically resolving blood flow within the vessels and, thus, preventing the simulation of CA transport as in this thesis.

One of the most important insights about CA transport processes that is drawn from the analysis in this thesis is the fact that intraluminal inhomogeneities appear to represent the major influence on CA dispersion in the coronary vascular network. Nonetheless, the definition of the relative dispersion (Eq. 6.2 as defined in [167, 173, 174]) shows reduced additional CA dispersion in ever smaller arteries. However, the reason for this are not the reduced vessel diameters as suggested by previous CFD-studies [18, 22**?** –24] but rather the complex transport phenomena. Being dependent on the hemodynamic conditions, this behavior is less prounounced under stress than at rest, resulting in different amounts of MBF quantification errors in the resting and hyperemic state.

The analysis indicates several factors as the major determinants of the influence of CA dispersion on bolus-based perfusion quantification measurements. These are the distance the bolus has travelled as well as the regime of the flow velocities in the considered vascular tree (cf. Section 6.3.2). In addition to this, the importance of the careful choice of the parameters in the MBF fitting procedure with MMID4 is emphasized.

Consequences for MRI Perfusion Measurements

Even though common Gadolinium-based MRI tracers are suspected to cause the risk of adverse reactions such as nephrogenic systemic fibrosis [79, 80, 208, 209], its usage in clinical settings is still of high interest due to the high resolution of MRI. Validation of MRI perfusion measurements on the basis of microsphere measurements (the experimental gold standard) shows good correlation with slight underestimation of MBF [5, 13, 89, 193, 194, 196]. The analyses performed in this work hint at the fact that this could be ascribed to the influence of the neglected CA bolus dispersion. Comparing results of myocardial MRI to PET measurements, the clinical gold standard for absolute perfusion quantification [8, 9], good correlations are found for MPR-values, however, only weak correlation for MBF [198]. It should be kept in mind that different tracers possess varying diffusion coefficients, also changing the observed dispersion effects [18, 129]. Accordingly, the effects described and analyzed here apply to the different imaging techniques, however, to varying degree depending on the applied CA's diffusion coefficient. In general, CA dispersion should thus be considered in all kinds of bolus-based perfusion measurements in the heart although also regarding other organs such as the kidney [210] or the prostate [211]. However, the detailed CFD analysis performed in [108] suggested that perfusion quantification errors were not nearly as pronounced in the brain.

These studies all give a good impression of the accuracy and validity of CE MRI perfusion measurements in comparison to other bolus-based techniques to assess MBF. It can be assumed that all these methods underlie the same bolus dispersion effects to varying degree. In order to fully understand and benchmark the findings of this as well as previous CFD analyses, a profound study is required, in which all data necessary for the simulations (e.g., several pressure and/or volume flow curves, AIF measurement) are retrieved in combination with an actual CE MRI perfusion measurement as well as a coronary angiography. In this regard, an additional perfusion measurement with a different technique (e.g., PET or microspheres) in the same individual (human or animal, respectively) would even allow for counterchecking of the simulations with different measurement techniques as well as between the imaging modalities with varying diffusion coefficients.

To ensure the relevance of future comparable CFD studies, it should be considered to extend the CFD simulations to integrate the *arterial spin labelling* technique [212–214] due to the adverse nephrological risks when CA is used. A first analysis of this kind was already performed in [108], yielding errors up to 100 %, further emphasizing the importance of a deep understanding of the observed dispersion effects, if bolus-based perfusion measurements are to be corrected for these distortions.

7.2 Outlook and Limitations

In order to carry on the analysis presented in this work and further improve its relevance, several different aspects could be integrated. The variability of the errors depending on the underlying MBF values might help to further optimize the fitting procedure itself. In addition, comparison with different procedures such as the Fermi model approach [8, 23, 215, 216], so-called *singular value decomposition* [217] or *Tikhonov regularization* as it is applied in cerebral imaging [218, 219] could be included in order to benchmark the observed perfusion quantification errors.

The complexity of the results from this work raises doubts that an error correction scheme accounting for CA dispersion effects, which contort dynamic bolus-based perfusion measurements is feasible. An error correction scheme based on the travelled distance as well as the flow velocities in the vessels would merely be usable for a rough approximation of these quantification errors. Moreover, these errors will vary between individual coronary vascular trees, further making it difficult to validate such an error correction framework in general. However, under the assumption that in general CA dispersion and subsequent errors in MBF and MPR quantification underlie the same phenomena in all vascular networks, the results from this work can well be used as a starting point for a machine or even a deep learning approach for the prediction of perfusion quantification errors in bolus-based measurements. Similar to its application in FFR prediction [154, 155, 161, 162] or simply for the segmentation of cardiovascular 3D geometries [220–222] this represents a promising approach to tackle an error correction.

Owing to the comparable settings of the performed CFD simulations (cf. Chapter 3), all results presented here underlie similar limitations. One of these is the assumption of rigid walls in the simulations neglecting FSI due to fluid pressure inside the vessels as well as tissue pressure working from the outside on the vessel walls. Even though in [223, 224] no substantial influence of vessel wall motion on species transport was observed, this should be integrated in future studies. Particularly, if even smaller vessels are to be included, where effects of outside tissue pressure can even lead to complete collapse of the arteries (cf. Section 2.2.2). Moroever, in [225] effects of motion of the cardiac arteries on the coronary hemodynamics are observed, especially in the setting of different degrees of stenosis. This implies that, in the case of pathologically altered vessels, influences on CA transport cannot be excluded beforehand, either.

Another simplification that is made in the simulations presented here is that CA is described by a mass concentration time curve within the blood. This implies that the tracer is considered not changing the blood's density, hence, neglecting the influence of effects such as friction, size and shape of the transported particles on the blood stream. Similar to [226], where the transport of emboli (blood clots, several millimeters in diameter) is modelled directly, the much smaller CA particles (diameters in the nano- and micrometer regime [227]) could be simulated as proper particles. Yet, this would require further assumptions and precise definition of shape, size and friction parameters within the blood stream as well as careful implementation of additional suitable transport mechanisms. Nonetheless, an analysis of transport of tracers with different diffusion coefficients on the vascular geometries from this work, as it was performed on an idealized geometry in [129], would also allow for further interesting insights about the effects of dispersion in different bolus-based methods for quantification of perfusion.

A comparative study modelling the tracer as a soluble as well as separate particles could, hence, give further insights about the validity of the simulations where a tracer mass concentration in blood is assumed. Moreover, this approach would also shed some light about the transport of particles in the blood stream in general. Particularly concerning the application of drugs, the processes analyzed and described in this work are of high interest. This could help answering the question if administered medication in fact reaches the location where it is required, and furthermore, if the given dose is sufficient.

In this regard, a supplementary interesting addition to the matter treated in this work would consist in modelling of the downstream microvasculature behind the model outlets. This could be performed by an approach using Darcy's law to model diffusion of the transported particles in question through the tissue, as it can be done for interstitial microvascular flow modelling in tumors [228, 229]. In the case that this could be extended to additionally model tissue demands at the model outlets, this would represent an even more intriguing approach.

Finally, another conclusion to be drawn with regard to the analysis performed here concerns the long duration of the different steps and the high amount of user intervention. This concerns the preparation of the CFD simulations, including the manual segmentation of the 3D models as well as the preparation of the different files required for the BCs at the model inlets and outlets (volume flow and pressure curves, outlet resistances; cf. Section 3.2 and appendices A.2 and A.3). These steps may be acceptable for a scientific study with the goal of gaining general insights about blood flow and particle transport processes in the coronary vasculature, as presented in this work. However, they are in their current form not feasible to be performed in parallel to an actual CE MRI perfusion measurement and thus be integrated into the clinical routine because they are too time-consuming. Moreover, the additional time for conduction of the actual simulations also has to be taken into account. In comparison to previous works, here, a considerable acceleration of the workflow is obtained by usage of the highly automatable volume discretization procedure presented in Section 3.3. Nonetheless, the workflow additionally needs further speed up in several aspects before it can be considered for clinical applications. Also here, approaches by use of deep or machine learning appear to be most promising in this regard.

Bibliography

[1] Eurostat. Cardiovascular diseases statistics, 2017, 2018-08-23. URL `http://ec.europa.eu/eurostat/statistics-explained/index.php?title=Cardiovascular_diseases_statistics`.

[2] World Health Organization. The top 10 causes of death, 2016, 2018-08-23. URL `http://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death`.

[3] P Libby and P Theroux. Pathophysiology of coronary artery disease. *Circulation*, 111 (25):8, 2005.

[4] B Ibanez, JJ Badimon and MJ Garcia. Diagnosis of atherosclerosis by imaging. *Am J Med*, 122(1A):S15–25, 2009.

[5] LY Hsu, DW Groves, AH Aletras, P Kellman and AE Arai. A quantitative pixel-wise measurement of myocardial blood flow by contrast-enhanced first-pass CMR perfusion imaging: microsphere validation in dogs and feasibility study in humans. *JACC Cardiovasc Imaging*, 5(2):154–66, 2012.

[6] K Bratis and E Nagel. Variability in quantitative cardiac magnetic resonance perfusion analysis. *J Thorac. Dis.*, 5(3):357–359, 2013.

[7] T D'Angelo, C Grigoratos, S Mazziotti, K Bratis, F Pathan, A Blandino, E Elen, VO Puntmann and E Nagel. High-throughput gadobutrol-enhanced MMR: a time and dose optimization study. *J Cardiovasc Magn Reson*, 19:83, 2017.

[8] M Jerosch-Herold, N Wilke, AE Stillman and RF Wilson. Magnetic Resonance Quantification of the Myocardial Perfusion Reserve with a Fermi Function Model for Constrained Deconvolution. *Medical Physics*, 25(1):73–84, 1998.

[9] AA Qayyum and J Kastrump. Measuring myocardial perfusion: the role of PET, MRI and CT. *Clin Radiol*, 70(6):576–584, 2015.

[10] M Sturm. *Semiquantitative myokardiale Perfusionsmessung in der Kardialen Magnetresonanztomographie im Vergleich zur Quantitativen Koronarangiographie.* PhD thesis, Universitätsklinikum Ulm, Klinik für Innere Medizin II, 2010.

[11] DC Lee and NP Johnson. Quantification of absolute myocardial blood flow by magnetic resonance perfusion imaging. *JACC Cardiovasc Imaging*, 2(6):761–70, 2009.

[12] LAE Brown, SC Onciul, DA Broadbent, K Johnson, GJ Fent, JRJ Foley, P Garg, PG Chew, K Knott, E Dall'Armellina, PP Swoboda, H Xue, JP Greenwood, JC Moon, P Kellman and S Plein. Fully automated, inline quantification of myocardial blood flow with cardiovascular magnetic resonance: repeatability of measurements in healthy subjects. *J Cardiovasc Magn Reson*, 20(1):48, 2018.

[13] TF Christian, AH Aletras and AE Arai. Estimation of absolute myocardial blood flow during first-pass MR perfusion imaging using a dual-bolus injection technique: Comparison to single-bolus injection method. *J. Magn. Reson. Imaging*, 27(6):1271–1277, 2008.

[14] AR Patel. Assessment of advance coronary artery disease: Advantages of quantitative cardiac magnetic resonance perfusion analysis. *J. Am. Coll. Cardiol*, 56(7):561–569, 2010.

[15] CA Miller. Voxel-wise quantification of myocardial blood flow with cardiovascular magnetic resonance: Effect of variations in methodlogy and validation with positron emission tomography. *J Cardiovasc Magn Reson*, 16(11):1–15, 2014.

[16] F Calamante. Arterial input function in perfusion MRI: a comprehensive review. *Prog Nucl Magn Reson Spectrosc*, 74:1–32, 2013.

[17] M Jerosch-Herold. Quantification of myocardial perfusion by cardioavscular magnetic resonance. *J. Cardiovasc. Magn. Reson.*, 12(1):57, 2010.

[18] D Graafen, K Münnemann, S Weber, K-F Kreitner and LM Schreiber. Quantitative contrast-enhanced myocardial perfusion magnetic resonance imaging: Simulation of bolus dispersion in constricted vessels. *Medical Physics*, 36(7):3099–3106, 2009.

[19] A Fortin, S Salmon, J Baruthio, M Delbany and E Durand. Flow MRI simulation in complex 3D geometries: Application to the cerebral venous network. *Magn Reson Med*, 80(4):1655–1665, 2018.

[20] A Boccadifuoco, A Mariotti, K Capellini, S Celi and MV Salvetti. Validation of Numerical Simulations of Thoracic Aorta Hemodynamics: Comparison with In Vivo Measurements and Stochastic Sensitivity Analysis. *Cardiovasc Eng Technol*, 2018.

[21] F Condemi, S Campisi, M Viallon, P Croisille, J-F Fuzelier and S Avril. Ascending thoracic aorta aneurysm repair induces positive hemodynamic outcomes in a patient with unchanged bicuspid aortic valve. *Journal of Biomechanics*, 81:145–148, 2018.

[22] D Graafen, J Hamer, S Weber and LM Schreiber. Quantitative myocardial perfusion magnetic resonance imaging: the impact of pulsatile flow on contrast agent bolus dispersion. *Physics in Medicine and Biology*, 56:5167–5185, 2011.

[23] R Schmidt, D Graafen, S Weber and LM Schreiber. Computational Fluid Dynamics Simulations of Contrast Agent Bolus Dispersion in a Coronary Bifurcation: Impact on MRI-Based Quantification of Myocardial Perfusion. *Computational and Mathematical Methods in Medicine*, 2013, 2013.

[24] K Sommer, R Schmidt, D Graafen, H-C Breit and LM Schreiber. Contrast Agent Bolus Dispersion in a Realistic Coronary Artery Geometry: Influence of Outlet Boundary Conditions. *Annals of Biomedical Engineering*, 42(4):787–796, 2013.

[25] K Sommer, D Bernat, R Schmidt, H-C Breit and LM Schreiber. Resting myocardial blood flow quantification using contrast-enhanced magnetic resonance imaging in the presence of stenosis: A computational fluid dynamics study. *Medical Physics*, 42(7): 4375–4384, 2015.

[26] JPHM van den Wijngaard, MGJTB van Lier, JAE Spaan and M Siebes. 3D Imaging of vascular networks for biophysical modeling of perfusion distribution within the heart. *Journal of Biomechanics*, 46:229–239, 2012.

[27] MG van Lier, E Oost, JA Spaan, P van Horssen, AC van der Wal, E van Bavel, M Siebes and JP van den Wijngaard. Transmural distribution and connectivity of coronary collaterals within the human heart. *Cardiovasc Pathol*, 25(5):405–12, 2016.

[28] JAE Spaan, R ter Wee, JWGE van Teeffelen, G Streekstra, M Siebes, C Kolyva, H Vink, DS Fokkema and Ev Bavel. Visualisation of intramural coronary vasculature by an imaging cryomicrotome suggests compartmentalisation of myocardial perfusion areas. *Medical and Biological Engineering and Computing*, 43:431–435, 2005.

[29] B Liu, J Zheng, R Bach and D Tang. Influence of model boundary conditions on blood flow patterns in a patient specific stenotic right coronary artery. *Biomed Eng Online*, 14 Suppl 1:S6, 2015.

[30] A Alimi and O Wünsch. Analysis and Optimization of Inlet and Outlet Boundary Conditions for Flow Simulations in Circle of Willis. *Pamm*, 17(1):179–180, 2017.

[31] M Willemet, V Lacroix and E Marchandise. Inlet boundary conditions for blood flow simulations in truncated arterial networks. *Journal of Biomechanics*, 44(5):897–903, 2011.

[32] N Westerhof, F Bosman, CJ de Vries and A Noordergraaf. Analog studies of the human systemic arterial tree. *Journal of Biomechanics*, 2(2):121–143, 1969.

[33] R Burattini, P Sipkema, GA van Huis and N Westerhof. Identification of canine coronary resistance and intramyocardial compliance on the basis of the waterfall model. *Annals of Biomedical Engineering*, 13:385–404, 1985.

[34] HJ Kim, IE Vignon-Clementel, JS Coogan, CA Figueroa, KE Jansen and CA Taylor. Patient-specific modeling of blood flow and pressure in human coronary arteries. *Ann Biomed Eng*, 38(10):3195–209, 2010.

[35] Y Huo and GS Kassab. The scaling of blood flow resistance: from a single vessel to the entire distal tree. *Biophys J*, 96(2):339–46, 2009.

[36] Y Huo and GS Kassab. A scaling law of vascular volume. *Biophys J*, 96(2):347–53, 2009.

[37] J Spurk and N Aksel. *Strömungslehre, Einführung in die Theorie der Strömungen.* Springer, 2007. ISBN 978-3-540-38441-0.

[38] H Oertel jr. and E Laurien. *Numerische Strömungsmechanik.* 2003. ISBN 978-3-322-96851-7.

[39] JH Ferziger and M Peric. *Computational Methods for Fluid Dynamics.* Springer, 2002. ISBN 3-540-42074-6.

[40] WM Saltzmann. *Drug Delivery - Engineering Principles for Drug Therapy.* Oxford University Press, New York, USA, 2001. ISBN 0-19-508589-2.

[41] LS Caretto, AD Gosman, SV Tatankar and DB Spalding. Two calculation procedures for steady, three-dimensional flows with recurculation. In H. Cabannes and R. Temam, editors, *Third International Conference on Numerical Methods in Fluid Mechanics*, volume 19 of *Lecture Notes in Physics*, pages 60–68. Springer, 1973.

[42] RI Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65, 1986.

[43] T Holzmann. *Mathematics, Numerics, Derivations and OpenFOAM(R).* Holzmann CFD, 4 edition, 2016.

[44] M Gekle, F Markwardt, E Wischmeyer, N Klöcker, S Gründer, R Baumann, M Petersen, H Marti and A Schwab. *Taschenlehrbuch Physiologie.* Georg Thieme Verlag KG, 2010.

[45] A Faller and M Schünke. *Der Körper des Menschen: Einführung in Bau und Funktion.* Georg Thieme Verlag KG, 2008.

[46] RF Schmidt and F Lang. *Physiologie des Menschen.* Springer Medizin Verlag, 2007.

[47] http://ib.bioninja.com.au/standard-level/topic-6-human-physiology/62-the-blood-system/heart-structure.html, 2018-10-15.

[48] J Layland, D Carrick, M Lee, K Oldroyd and C Berry. Adenosine: physiology, pharmacology, and clinical applications. *JACC Cardiovasc Interv*, 7(6):581–91, 2014.

[49] H Lippert. *Lehrbuch der Anatomie.* Urban und Schwarzenberg, 1990.

[50] AC Burton. Relation of Structure to Function of the Tissues of the Wall of Blood Vessels. *Physiological Reviews*, 34(4):619–642, 1954.

[51] CB Wolff. Normal cardiac output, oxygen delivery and oxygen extraction. *Adv. Exp. Med. Biol.*, 599:169–182, 2007.

[52] T Ramanathan and H Skinner. Coronary blood flow. *Continuing Education in Anaesthesia Critical Care and Pain*, 5(2):61–64, 2005.

[53] S Sakamoto, S Takahashi, AU Coskun, MI Papafaklis, A Takahashi, S Saito, PH Stone and CL Feldman. Relation of distribution of coronary blood flow volume to coronary artery dominance. *Am J Cardiol*, 111(10):1420–4, 2013.

[54] H Wieneke, C von Birgelen, M Haude, H Eggebrecht, S Mohlenkamp, A Schmermund, D Bose, C Altmann, T Bartel and R Erbel. Determinants of coronary blood flow in humans: quantification by intracoronary Doppler and ultrasound. *J Appl Physiol*, 98 (3):1076–82, 2005.

[55] DK Molina and VJ DiMaio. Normal organ weights in men: part I - the heart. *Am J Forensic Med Pathol*, 33(4):362–7, 2012.

[56] P Chareonthaitawee, PA Kaufmann, O Rimoldi and PG Gamici. Heterogeneity of resting and hyperemic myocardial blood flow in healthy humans. *Cardiovascular Research*, 50:151–161, 2001.

[57] JIE Hoffmann. The history of the microsphere method for measuring blood flows with special reference to myocardial blood flow: a personal memoir. *Am J Physiol Heart Circ Physiol*, 312(4):H705–H710, 2017.

[58] PJ Lynch and CC Jaffe. Coronary.pdf: Patrick j. lynch, medical illustratorderivative work [1]: Fred the oyster (talk)adaption and further labeling: Mikael häggström [cc by-sa 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], via wikimedia commons, retrieved 2019-11-29.

[59] WF Boron and EL Boulpaep. *Medical Physiology.* Elsevier Ltd., 2012.

[60] S Parasuraman, S Walker, BL Loudon, ND Gollop, AM Wilson, C Lowery and MP Frenneaux. Assessment of pulmonary artery pressure by echocardiography-A comprehensive review. *Int J Cardiol Heart Vasc*, 12:45–51, 2016.

[61] JAE Spaan. *Coronary Blood Flow: Mechanics, Distribution, and Control.* Kluwer Academic Publishers, 1 edition, 1991.

[62] RE Austin, GS Aldea, DL Coggins, AE Flynn and JI Hoffman. Profound spatial heterogeneity of coronary reserve. Discordance between patterns of resting and maximal myocardial blood flow. *Circulation Research*, 67(2):319–331, 1990.

[63] R Fahraeus and T Lindqvist. The viscosity of blood in narrow capillary tubes. *The American Journal of Physiology*, 96:562–568, 1931.

[64] O Dössel. *Bildgebende Verfahren in der Medizin: Von der Technik zur medizinischen Anwendung.* Springer, 2000. ISBN 978-3-662-06046-9.

[65] E Haacke, R Brown, MR Thompson and R Venkaresen. *Magnetic Resonance Imaging: Physical Principles and Sequence Design.* Wiley-Liss, New York, 1999. ISBN 978-0471351283.

[66] P Sprawls. *Magnetic Resonance Imaging Principles, Methods, and Techniques.* Medical Physics Publishing Corporation, Madison, Wisconsin, USA, 2000. ISBN 0-944838-97-6.

[67] TJ Glover. *Pocket Reference.* Sequoia, 2 edition, 1997. ISBN 978-1885071002.

[68] R Chang. *Chemistry.* McGraw-Hill, 9 edition, 2007. ISBN 0-07-110595-6.

[69] A Carrington and AD MacLachlan. *Introduction to Magnetic Resonance.* A Harper International Edition, 1969. ISBN 978-0063561076.

[70] *PT 2025 NMR Teslameter Manual, Version 2.0 (Revision 1.0).* Metrolab Instruments, 2003.

[71] The NIST Reference on Constants Units and Uncertainty. https://physics.nist.gov/cgi-bin/cuu/value?gammap, 2018-12-06.

[72] F Bloch. Nuclear Induction. *Physical Review*, 70(7-8):460–474, 1946.

[73] MA Bernstein, KF King and XJ Zhou. *Handbook of MRI Pulse Sequences.* Elsevier, 2004. ISBN 978-0-12-092861-3.

[74] TP Roberts and D Mikulis. Neuro MR: principles. *J Magn Reson Imaging*, 26(4): 823–37, 2007.

[75] GJ Stanisz, EE Odrobina, J Pun, M Escaravage, SJ Graham, MJ Bronskill and RM Henkelman. T1, T2 relaxation and magnetization transfer in tissue at 3T. *Magn Reson Med*, 54(3):507–12, 2005.

[76] DD Traficante. Relaxation Can T2 be longer than T1? *Concepts in Magnetic Resonance*, 3:171–177, 1991.

[77] https://mri-q.com, 2018-10-22.

[78] RE Hendrick. Breast MRI: Using Physics to Maximize Its Sensitivity and Specificity to Breast Cancer. In *Proc. Amer. Assoc. of Phys. in Med.*, Pittsburgh, USA, 2004.

[79] J Ramalho, M Ramalho, M Jay, LM Burke and RC Semelka. Gadolinium toxicity and treatment. *Magn Reson Imaging*, 34(10):1394–1398, 2016.

[80] M Rogosnitzky and S Branch. Gadolinium-based contrast agent toxicity: a review of known and proposed mechanisms. *Biometals*, 29(3):365–76, 2016.

[81] D Graafen. *Untersuchung der Blutströmung in Herzkranzarterien mittels Computational Fluid Dynamics.* PhD thesis, University Mainz, 2008.

[82] N al Saadi, M Gross, A Bornstedt, B Schnackenburg, C Klein, E Fleck and E Nagel. Comparison of various parameters for determining an index of maocardial perfusion reserve in detecting coronary stenosis with cardiovascular magnetic resonance tomography. *Z Kardiol*, 90(11):824–34, 2001.

[83] M Jerosch-Herold, RT Seethamraju, CM Swingen, NM Wilke and AE Stillman. Analysis of myocardial perfusion MRI. *J Magn Reson Imaging*, 19(6):758–70, 2004.

[84] NM Wilke, M Jerosch-Herold, AE Stillman, K Kroll, N Tsekos, H Merkle, T Parrish, X Hu, Y Wang and JB Bassingthwaighte. Concepts of myocardial perfusion imaging in magnetic resonancen imaging. *Magn Res Q*, 10(4):249–86, 1994.

[85] KL Zierler. Theoretical Basis of Indicator-Dilution Methods For Measuring Flow and Volume. *Circ Res*, 10:393–407, 1962.

[86] JR Ewing, D Bonekamp and PB Barker. *Clinical Perfusion MRI: Techniques and Applications*, chapter Imaging of flow: basic principles. Cambridge University Press, 2013. ISBN 9781107013391.

[87] M Jerosch-Herold, N Wilke, Y Wang, G-R Gong, AM Mansoor, H Huang, S Gurchumelidze and AE Stillman. Direct comparison of an intravascular and an extracellular contrast agent for quantification of myocardial perfusion. *Int J Cardiac Imaging*, 15:453–464, 1999.

[88] *MMID4 Manual.* National Simulation Resource for Mass Transport and Exchange, Department of Bioengineering, University of Washington, 1998.

[89] M Schmitt, G Horstick, SE Petersen, A Karg, N Hoffmann, T Gumbrich, N Abegunewardene and WG Schreiber. Quantification of Resting Myocardial Blood Flow in a Pig Model of Acute Ischemia Based on First-Pass MRI. *Magnetic Resonance in Medicine*, 53:1223–1227, 2005.

[90] F Calamante, L Willats, DG Gadian and A Connelly. Bolus Delay and Dispersion in Perfusion MRI: Implications for Tissue Predictor Models in Stroke. *Magnetic Resonance in Medicine*, 55:1180–1185, 2006.

[91] SB Bender, MJ van Houwelingen, D Merkus, DJ Duncker and MH Laughlin. Quantitative analysis of exercise-induced enhancement of early- and late-systolic retrograde coronary blood flow. *J Appl Physiol*, 108:507–514, 2010.

[92] GA van Huis, P Sipkema and N Westerhof. Coronary input impedance during cardiac cycle as determined by impulse response method. *AJP Heart*, 253(2 Pt 2):H317–324, 1987.

[93] S Mantero, R Pietrabissa and R Fumero. The coronary bed and its role in the cardiovascular system: a review and an introductory single-branch model. *J Biomed Eng*, 14:109–116, 1992.

[94] S Goßner. *Grundlagen der Elektronik: Halbleiter, Bauelemente und Schaltungen; ein Lernbuch.* Shaker, 2001. ISBN 9783826588259.

[95] JO Attia. *Electronics and Circuit Analysis using Matlab.* CRC Press LLC, USA, 1999. ISBN 0-8493-1176-4.

[96] E Gerke, W Juchelka, U Mittmann and J Schmier. Der intramyokardiale Druck des Hundes in verschiedenen Tiefen bei Druckbelastung und Ischämie des Herzmuskels. *Basic Res Cardiol*, 70:537–546, 1975.

[97] GS Kassab, J Berkley and YCB Fung. Analysis of pig's coronary arterial blood flow with detailed anatomical data. *Annals of Biomedical Engineering*, 25:204–217, 1997.

[98] Y Zhou, GS Kassab and S Molloi. On the design of the coronary arterial tree: a generalization of Murray's law. *Phys Med Biol*, 44:2929–2944, 1999.

[99] S Kaul and AR Jayaweera. Determinants of microvascular flow. *Eur Heart J*, 27: 2272–2274, 2006.

[100] G Avanzolini, P Barbini, A Cappello and G Cevenini. CADCS simulation of the closed-loop cardiovascular system. *Int J Biomed Comput*, 22:39–49, 1988.

[101] AC Guyton and JE Hall. *Textbook of Medical Physiology 13th Edition.* Elsevier Saunders, 2005. ISBN 9781437700602.

[102] W Ganz, K Tamura, HS Marcus, R Donoso, S Yoshida and HJC Swan. Measurement of coronary sinus blood flow by continuous thermodilution in man. *Circulation*, 44: 181–195, 1971.

[103] JS Tanedo, RF Kelly, M Marquez, DE Burns, LW Klein, M Rosa Costanzo, JE Parrillo and SM Hollenberg. Assessing coronary blood flow dynamics with the TIMI frame count method: comparison with simultaneous intracoronary doppler and ultrasound. *Catheterization and Cardiovascular Interventions*, 53:459–463, 2001.

[104] AR Pries, D Neuhaus and P Gaehtgens. Blood viscosity in tube flow: dependence on diameter and hematocrit. *Am J Physiol*, 263(32):H1770–8, 1992.

[105] AR Pries, TW Secomb, T Geßner, MB Sperandio, JF Gross and P Gaehtgens. Resistance to blood flow in microsvessels in vivo. *Circ Res*, 75:904–915, 1994.

[106] G de Santis, P Mortier, M de Beule, P Segers, P Verdonck and B Verhegghe. Patient-specific computational fluid dynamics: structured mesh generation from coronary angiography. *Med Biol Eng Comput*, 48:371–380, 2010.

[107] *ANSYS FLUENT User's Guide, Release 14.0.* Inc. ANSYS, 2011.

[108] K Sommer. *Simulation of Bolus Transport in Magnetic Resonance Perfusion Imaging using Realistic Vascular Geometries.* PhD thesis, University Mainz, 2015.

[109] JT Dodge, BG Brown, EL Bolson and HT Dodge. Lumen diameter of normal human coronary arteries: influence of age, sex, anatomic variation, and left ventricular hypertrophy or dilation. *Circulation*, 56(16):5167–5185, 1992.

[110] E van Bavel and JAE Spaan. Branching patterns in the porcine coronary arterial tree - estimation of flow heterogeneity. *Circulation Research*, 71(5):1200–1212, 1992.

[111] T Pflederer, J Ludwig, D Ropers, WG Daniel and S Achenbach. Measurement of coronary artery bifurcation angles by multidetector computed tomography. *Investigative Radiology*, 41(11):793–798, 2006.

[112] Introductory OpenFOAM Course. Genoa, 2015. University of Genoa, DICCA, Dipartimento di Ingegneriy Civiel, Chimica e Ambientale.

[113] H Jasak. Finite Volume Discretisation in OpenFOAM - Best Practice Guidelines. In *CFD with OpenSource Software Course*, Chalmers University, Gothenburg, 2015.

[114] PD Ballyk, DA Steinman and CR Ethier. Simulation of non-Newtonian blood flow in an end-to-side anastomosis. *Biorheology*, 31(5):565–576, 1993.

[115] M Germano, U Piomelli, P Moi and WH Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A : Fluid Dynamics*, 3(7):1760, 1991.

[116] DK Lilly. A proposed modification of the Germano subgrid-scale closure method. *Physics of Fluids A : Fluid Dynamics*, 4(3):633–635, 1992.

[117] T Gatski, MY Hussaini and JL Lumley. *Simulation and Modeling of Turbulent Flows*. Oxford University Press, New York, 1 edition, 1996. ISBN 978-0195106435.

[118] M Lesieur, O Metais and P Conte. *Large-Eddy Simulations of Turbulence*. Cambridge University Press, New York, 2005. ISBN 978-0521781244.

[119] C Wieseotte, M Wagner and LM Schreiber. An estimate of Gd-DOTA diffusivitiy in blood by direct NMR diffusion measurement of its hydrodynamic analogue Ga-DOTA, Abstract No: 2618. In *22nd Annual Meeting of the International Society for Magnetic Resonance in Medicine*, Milan, Italy, 2014.

[120] FJH Gijsen, E Allanic, FN van de Vosse and JD Janssen. The influence of the non-Newtonian properties of blood on the flow in large arteries: unsteady flow in a 90 degree curved tube. *Journal of Biomechanics*, 32:705–713, 1999.

[121] M Mischi, JA den Boer and HHM Korsten. On the physical and stochastic representation of an indicator dilution curve as a gamma fit. *Physiological Measurement*, 29: 281–294, 2008.

[122] JD Cutnell and KW Johnson. *Physics (4th Edition, Volume 1)*. John Wiley and Sons Inc, 1997. ISBN 0471191124.

[123] RB King, JB Bassingthwaighte, JRS Hales and LB Bowell. Stability of Heterogeneity of Myocardial Blood Flow in Normal Awake Baboons. *Circ Res*, 57:285–295, 1985.

[124] K Kroll, N Wilke, M Jerosch-Herold, Y Wang, Y Zhang, RJ Bache and JB Bassingthwaighte. Modeling regional myocardial flows from residue functions of an intravascular indicator. *American Journal of Physiology*, 271(4 Pt 2):H1643–H1655, 1996.

[125] S Weber, A Kronfeld, RP Kunz, K Muennemann, G Horstick, KF Kreitner and WG Schreiber. Quantitative myocardial perfusion imaging using different autocalibrated parallel acquisition techniques. *J Magn Reson Imaging*, 28(1):51–9, 2008.

[126] IS Chan, AA Goldstein and JB Bassingthwaighte. SENSOP: A derivative-free solver for nonlinear least squares sensitivity scaling. *Ann Biomed Eng*, 21(6):621–631, 1993.

[127] K Levenberg. A method for the solution of certain problems in least squares. *Quart Appl Math*, 2:164–168, 1944.

[128] D Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J Soc Ind Appl Math*, 11:431–442, 1963.

[129] R Schmidt. *Numerische Strömungssimulation in verschiedenen Koronargefäßmodellen zur Untersuchung der Dispersion des Kontrastmittelbolus bei der myokardialen MR-Perfusionsmessung.* PhD thesis, Johannes Gutenberg-Universität Mainz, 2017.

[130] S Murty Bhallamudi, S Panday and PS Huyakorn. Sub-timing in fluid flow and transport simulations. *Advances in Water Resources*, 26(5):477–489, 2003.

[131] YJ Park, EA Sudicky, S Panday, JF Sykes and V Guvanasen. Application of implicit sub-time stepping to simulate flow and transport in fractured porous media. *Advances in Water Resources*, 31(7):995–1003, 2008.

[132] J Spaan, C Kolyva, J van den Wijngaard, R ter Wee, P van Horssen, J Piek and M Siebes. Coronary structure and perfusion in health and disease. *Philos Trans A Math Phys Eng Sci*, 366(1878):3137–53, 2008.

[133] JPHM van den Wijngaard, H Schulten, P van Horssen, RD ter Wee, M Siebes, MJ Post and JAE Spaan. Porcine Coronary Collateral Formation in the Absence of a pressure gradient remote of the ischemic border zone. *Am J Physiol Heart Circ Physiol*, 300: H1930–H1937, 2010.

[134] MS Olufsen. A structured tree outflow condition for blood flow in the larger systemic arteries, 1998.

[135] MS Olufsen, CS Peskin, WY Kim, EM Pedersen, A Nadim and J Larsen. Numerical Simulation and Experimental Validation of Blood Flow in Arteries with Structured-Tree Outflow Conditions. *Annals of Biomedical Engineering*, 28:1281–1299, 2000.

[136] W Cousins and PA Gremaud. Boundary conditions for hemodynamics: The structured tree revisited. *Journal of Computational Physics*, 231:6086–6096, 2012.

[137] A Duran, MS Celebi, S Piskin and M Tuncel. Scalability of openfoam for bio-medical flow simulations, 2014. URL `www.prace-ri.eu`.

[138] O Rivera, K Fürlinger and D Kranzlmüller. Investigating the scalability of OpenFOAM for the solution of transport equations and Large Eddy simulations. In *ICA 3PP 2011 Workshops*, volume 7017. Springer LNCS, 2011.

[139] M Culpo. Current bottlenecks in the scalability of openfoam on massively parallel clusters, 2016. URL `www.prace-ri.eu`.

[140] NHJ Pijls, B De Bruyne, K Peels, PH Van der Hoort, HJRM Bonnier, J Bartunek and JJ Koolen. Measurement of Franctional Flow Reserve to Assess the Functional Severity of Coronary-Artery Stenoses. *The New England Journal of Medicine*, 334 (26):1703–1708, 1996.

[141] NH Pijls, P van Schaardenburgh, G Manoharan, E Boersma, JW Bech, M van't Veer, F Bar, J Hoorntje, J Koolen, W Wijns and B de Bruyne. Percutaneous coronary intervention of functionally nonsignificant stenosis: 5-year follow-up of the DEFER Study. *J Am Coll Cardiol*, 49(21):2105–11, 2007.

[142] A Mehra and B Mohan. Value of FFR in clinical practice. *Indian Heart J*, 67(1): 77–80, 2015.

[143] PAL Tonino, B De Bruyne, NH Pijls, U Siebert, F Ikeno, M Van 't Veer, V Klauss, G Manoharan, T Engstrom, K Oldroyd, PN Ver Lee, P MacCarthy and WF Fearon. Fractional Flow Reserve versus Angiography for Guiding Percutatneous Coronary Intervention. *The New England Journal of Medicine*, 360(3):213–224, 2009.

[144] MJ Kern, A Lerman, JW Bech, B De Bruyne, E Eeckhout, WF Fearon, ST Higano, MJ Lim, M Meuwissen, JJ Piek, NH Pijls, M Siebes and JA Spaan. Physiological assessment of coronary artery disease in the cardiac catheterization laboratory: a scientific statement from the American Heart Association Committee on Diagnostic and Interventional Cardiac Catheterization, Council on Clinical Cardiology. *Circulation*, 114(12):1321–41, 2006.

[145] W Wijns, P Kolh, N Danchin, C Di Mario, V Falk, T Folliguet, S Garg, K Huber, S James, J Knuuti, J Lopez-Sendon, J Marco, L Menicanti, M Ostojic, MF Piepoli, C Pirlet, JL Pomar, N Reifart, FL Ribichini, MJ Schalij, P Sergeant, PW Serruys, S Silber, M Sousa Uva and D Taggart. Guidelines on myocardial revascularization. *Eur Heart J*, 31(20):2501–55, 2010.

[146] MR Patel, GJ Dehmer, JW Hirshfeld, PK Smith and JA Spertus. Appropriate use criteria for coronary revascularization focused update: a report of the American College of Cardiology Foundation Appropriate Use Criteria Task Force, Society for Cardiovascular Angiography and Interventions, Society of Thoracic Surgeons, American Association for Thoracic Surgery, American Heart Association, American Society of Nuclear Cardiology, and the Society of Cardiovascular Computed Tomography. *J Am Coll Cardiol*, 59(9):857–81, 2012.

[147] R Petraco, S Sen, S Nijjer, M Echavarria-Pinto, J Escaned, DP Francis and JE Davies. Fractional flow reserve-guided revascularization: practical implications of a diagnostic gray zone and measurement variability on clinical decisions. *JACC Cardiovasc Interv*, 6(3):222–5, 2013.

[148] M Gotberg, EH Christiansen, IJ Gudmundsdottir, L Sandhall, M Danielewicz, L Jakobsen, SE Olsson, P Ohagen, H Olsson, E Omerovic, F Calais, P Lindroos, M Maeng, T Todt, D Venetsanos, SK James, A Karegren, M Nilsson, J Carlsson, D Hauer, J Jensen, AC Karlsson, G Panayi, D Erlinge, O Frobert and FRSI i. Instantaneous Wave-free Ratio versus Fractional Flow Reserve to Guide PCI. *N Engl J Med*, 376(19):1813–1823, 2017.

[149] SS Nijjer, S Sen, R Petraco, J Mayet, DP Francis and JE Davies. The Instantaneous wave-Free Ratio (iFR) pullback: a novel innovation using baseline physiology to optimise coronary angioplasty in tandem lesions. *Cardiovasc Revasc Med*, 16(3):167–71, 2015.

[150] T Härle, W Bojara, S Meyer and A Elsässer. Comparison of instantaneous wave-free ratio (iFR) and fractional flow reserve (FFR) - first real world experience. *Int J Cardiol*, 199:1–7, 2015.

[151] BK Koo, A Erglis, JH Doh, DV Daniels, S Jegere, HS Kim, A Dunning, T DeFrance, A Lansky, J Leipsic and JK Min. Diagnosis of ischemia-causing coronary stenoses by noninvasive fractional flow reserve computed from coronary computed tomographic angiograms. Results from the prospective multicenter DISCOVER-FLOW (Diagnosis of Ischemia-Causing Stenoses Obtained Via Noninvasive Fractional Flow Reserve) study. *J Am Coll Cardiol*, 58(19):1989–97, 2011.

[152] SB Deng, XD Jing, J Wang, C Huang, S Xia, JL Du, YJ Liu and Q She. Diagnostic performance of noninvasive fractional flow reserve derived from coronary computed tomography angiography in coronary artery disease: A systematic review and meta-analysis. *Int J Cardiol*, 184:703–9, 2015.

[153] F Secchi, M Ali, E Faggiano, PM Cannao, M Fedele, S Tresoldi, G Di Leo, F Auricchio and F Sardanelli. Fractional flow reserve based on computed tomography: an overview. *Eur Heart J Suppl*, 18(Suppl E):E49–E56, 2016.

[154] M Tröbs, S Achenbach, J Rother, T Redel, M Scheuering, D Winneberger, K Klingenbeck, L Itu, T Passerini, A Kamen, P Sharma, D Comaniciu and C Schlundt. Comparison of Fractional Flow Reserve Based on Computational Fluid Dynamics Modeling Using Coronary Angiographic Vessel Morphology Versus Invasively Measured Fractional Flow Reserve. *Am J Cardiol*, 117(1):29–35, 2016.

[155] C Tesche, CN de Cecco, MH Albrecht, TM Duguay, RR Bayer, SE Litwin, DH Steinberg and UJ Schoepf. Coronary CT Angiography-derived Fractional Flow Reserve. *Radiology*, 285(1):17–33, 2017.

[156] TA Fairbairn, K Nieman, T Akasaka, BL Norgaard, DS Berman, G Raff, LM Hurwitz-Koweek, G Pontone, T Kawasaki, NP Sand, JM Jensen, T Amano, M Poon, K Ovrehus, J Sonck, M Rabbat, S Mullen, B De Bruyne, C Rogers, H Matsuo, JJ Bax, J Leipsic and MR Patel. Real-world clinical utility and impact on clinical decision-making of coronary computed tomography angiography-derived fractional flow reserve: lessons from the ADVANCE Registry. *Eur Heart J*, 0:1–11, 2018.

[157] NHJ Pijls. Coronary Physiology in the Cathlab: Theory and Practial Set-Up of FFR. In *Educational Training Program ESC*, European Heart House, Biot, France, 2014.

[158] L Itu, P Sharma, V Mihalef, A Kamen, C Suciu and D Comaniciu. A Patient-Specific Reduced-Order Model for Coronary Circulation. In *9th IEEE International Symposium on Biomedical Imaging*, volume 12, pages 832–835, Barcelona, Spain, 2012. IEEE.

[159] NG Uren, JA Melin, B De Bruyne, W Wijns, T Baudhuin and PG Gamici. Relation Between Myocardial Blood Flow and the Severity of Coronary Artery Stenosis. *The New England Journal of Medicine*, 330(25):1782–1790, 1994.

[160] T Hozumi, K Yoshida, Y Ogata, T Akasaka, Y Asami, T Takagi and S Morioka. Noninvasive Assessment of Significant Left Anterior Descending Coronary Artery Stenosis by Coronary Flow Velocity Reserve With Transthoracic Color Doppler Echocardiography. *Circulation*, 97(16):1557–1562, 1998.

[161] 2nd, DH Steinberg, KL Grant, C Canstein, C Schwemmer, M Schoebinger, LM Itu, S Rapaka, P Sharma C Tesche, CN de Cecco, S Baumann, M Renker, TW McLaurin, TM Duguay, RR Bayer and UJ Schoepf. Coronary CT Angiography-derived Fractional Flow Reserve: Machine Learning Algorithm versus Computational Fluid Dynamics Modeling. *Radiology*, 288(1):64–72, 2018.

[162] R Nakanishi, S Sankaran, L Grady, J Malpeso, R Yousfi, K Osawa, I Ceponiene, N Nazarat, S Rahmani, K Kissel, E Jayawardena, C Dailing, C Zarins, BK Koo, JK Min, CA Taylor and MJ Budoff. Automated estimation of image quality for coronary computed tomographic angiography using machine learning. *Eur Radiol*, 28 (9):4018–4026, 2018.

[163] J Martens, S Panzer, JPHM van den Wijngaard, M Siebes and LM Schreiber. Analysis of coronary contrast agent transport in bolus-based quantitative myocardial perfusion MRI measurements with computational fluid dynamics simulations. In M. Pop and G. A. Wright, editors, *FIMH 2017*, volume 10263, pages 369–380, Toronto, Canada, 2017. Springer International Publishing AG.

[164] F Calamante, P Yim and J Cebral. Estimation of bolus dispersion effects in perfusion MRI using image-based computational fluid dynamics. *NeuroImage*, 19:341–353, 2003.

[165] *Endspurt - Skripten fürs Physikum - Physiologie 1*. Georg Thieme Verlag KG, 2011. ISBN 9783131534415.

[166] JA Adam. Blood Vessel Branching: Beyond the Standard Calculus Problem. *Mathematics Magazine*, 84(3):196–207, 2011.

[167] RB King, A Deussen, GM Raymond and JB Bassingthwaighte. A vascular transport operator. *Am J Physiol*, 265(602):H2196–H2208, 1993.

[168] S Kamoi, C Pretty, P Docherty, D Squire, J Revie, YS Chiew, T Desaive, GM Shaw and JG Chase. Continuous stroke volume estimation from aortic pressure using zero dimensional cardiovascular model: proof of concept study from porcine experiments. *PLoS One*, 9(7):e102476, 2014.

[169] PP Lelovas, NG Kostomitsopoulos and TT Xanthos. A comparative anatomic and physiologic overview of the porcine heart. *Journal of the American Association for Laboratory Animal Science*, 53(5):432–438, 2014.

[170] MD Cerqueira, NJ Weissman, D V., AK Jacobs, S Kaul, WK Laskey, DJ Pennell, JA Rumberger, T Ryan and MS Verani. Standardized Myocardial Segmentation and Nomenclature for Tomographic Imaging of the Heart. *Ciculation*, 105:539–542, 2002.

[171] No Authors Listed. Standardization of Cardiac Tomographic Imaging. *Circulation*, 86(1):338–339, 1992.

[172] GS Kassab. Scaling laws of vascular trees: of form and function. *Am J Physiol Heart Circ Physiol*, 290(2):H894–903, 2006.

[173] JB Bassingthwaighte. Relative Dispersion: A Characterizing Feature of Specific Vascular Beds. *Anesth Analg*, 56(1):72–77, 1977.

[174] TJ Knopp, WA Dobbs, JF Greenleaf and JB Bassingthwaighte. Transcoronary Intravascular Transport Functions Obtained via a Stable Deconvolution Technique. *Ann Biomed Eng*, 4(1):44–59, 1976.

[175] N Kawel, EB Turkbey, JJ Carr, J Eng, AS Gomes, WG Hundley, C Johnson, SC Masri, MR Prince, RJ van der Geest, JA Lima and DA Bluemke. Normal left ventricular myocardial thickness for middle-aged and older subjects with steady-state free precession cardiac magnetic resonance: the multi-ethnic study of atherosclerosis. *Circ Cardiovasc Imaging*, 5(4):500–8, 2012.

[176] S Schalla, C Klein, I Paetsch, H Lehmkuhl, A Bornstedt, B Schnackenburg, E Fleck and E Nagel. Real-time MR image acquisition during high-dose dobutamine hydrochloride stress for detecting left ventricular wall-motion abnormalities in patients with coronary arterial disease. *Radiology*, 224(3):845–51, 2002.

[177] DE Longnecker, DL Brown, MF Newman and WM Zapol. *Anesthesiology*. McGraw-Hill Education, 2012. ISBN 978-0071785136.

[178] A Rossi, A Uitterdijk, M Dijkshoorn, E Klotz, A Dharampal, M van Straten, WJ van der Giessen, N Mollet, RJ van Geuns, GP Krestin, DJ Duncker, PJ de Feyter and D Merkus. Quantification of myocardial blood flow by adenosine-stress CT perfusion imaging in pigs during various degrees of stenosis correlates well with coronary artery blood flow and fractional flow reserve. *Eur Heart J Cardiovasc Imaging*, 14(4):331–8, 2013.

[179] R Fahmi, BL Eck, J Levi, A Fares, A Dhanantwari, M Vembar, HG Bezerra and DL Wilson. Quantitative myocardial perfusion imaging in a porcine ischemia model using a prototype spectral detector CT system. *Phys Med Biol*, 61(6):2407–31, 2016.

[180] T Kurita, H Sakuma, K Onishi, M Ishida, K Kitagawa, T Yamanaka, T Tanigawa, T Kitamura, K Takeda and M Ito. Regional myocardial perfusion reserve determined using myocardial perfusion magnetic resonance imaging showed a direct correlation with coronary flow velocity reserve by Doppler flow wire. *Eur Heart J*, 30(4):444–52, 2009.

[181] JHS Cullen, MA Horsfield, CR Reek, GR Cherryman, DB Barnett and NJ Samani. A myocardial perfusion reserve index in humans using first-pass contrast-enhanced magnetic resonance imaging. *Journal of the American College of Cardiology*, 33(5): 1386–1394, 1999.

[182] P Goykhman, PK Mehta, M Agarwal, C Shufelt, PJ Slomka, Y Yang, Y Xu, LJ Shaw, DS Berman, NB Merz and LE Thomson. Reproducibility of myocardial perfusion reserve - variations in measurements from post processing using commercially available software. *Cardiovasc Diagn Ther*, 2(4):268–77, 2012.

[183] O Muehling, M Jerosch-Herold, P Panse, A Zenovich, B Wilson, R Wilson and N Wilke. Regional Heterogeneity of Myocardial Perfusion in Healthy Human Myocardium: Assessment with Magnetic Resonance Perfusion Imaging. *Journal of Cardiovascular Magnetic Resonance*, 6(2):499–507, 2004.

[184] S Ben-Haim, VL Murthy, C Breault, R Allie, A Sitek, N Roth, J Fantony, SC Moore, MA Park, M Kijewski, A Haroon, P Slomka, K Erlandsson, R Baavour, Y Zilberstien, J Bomanji and MF Di Carli. Quantification of Myocardial Perfusion Reserve Using Dynamic SPECT Imaging in Humans: A Feasibility Study. *J Nucl Med*, 54(6):873–9, 2013.

[185] T Fritz-Hansen, JD Hove, KF Kofoed, H Kelbaek and HB Larsson. Quantification of MRI measured myocardial perfusion reserve in healthy humans: a comparison with positron emission tomography. *J Magn Reson Imaging*, 27(4):818–24, 2008.

[186] G Taylor. Dispersion of Soluble Matter in Solvent Flowing Slowly through a Tube. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 219(1137):186–203, 1953.

[187] K Tsuiki, Y Sato, K Sudo and J Nagasawa. Dependency of Transcoronay Circulatory Transport Function on Coronary Perfusion Pressure and Flow. *Tohoku J exp. Med.*, 126:377–383, 1978.

[188] J Pfitzner. Poiseuille and his law. *Anaesthesia*, 31:273–275, 1976.

[189] FW Prinzen and JB Bassingthwaighte. Blood flow distributions by microsphere deposition methods. *Cardiovasc Res*, 45(1):13–21, 2000.

[190] UKM Decking, VM Pai, E Bennett, JL Taylor, CD Fingas, K Zanger, H Wen and RS Balaban. High-resolution imaging reveals a limit in spatial resolution of blood flow measurements by microspheres. *Am J Physiol Heart Circ Physiol*, 287:H1132–H1140, 2004.

[191] JU Fluckiger, BC Benefield, KR Harris and DC Lee. Absolute quantification of myocardial blood flow with constrained estimation of the arterial input function. *J Magn Reson Imaging*, 38(3):603–9, 2013.

[192] TF Christian, DW Rettmann, AH Aletras, SL Liao, JL Taylor, RS Balaban and AE Arai. Absolute myocardial perfusion in canines measured by using dual-bolus first-pass MR imaging. *Radiology*, 232(3):677–84, 2004.

[193] TA Goldstein, M Jerosch-Herold, B Misselwitz, H Zhang, RJ Gropler and J Zheng. Fast mapping of myocardial blood flow with MR first-pass perfusion imaging. *Magn Reson Med*, 59(6):1394–1400, 2008.

[194] A Schuster, N Zarinabad, M Ishida, M Sinclair, JPHM van den Wijngaard, G Morton, GLTF Hautvast, B Bigalke, P van Horssen, N Smith, JAE Spaan, M Siebes, A Chiribiri and E Nagel. Quantitative assessment of magnetic resonance derived myocardial perfusion measurements using advanced techniques: microsphere validation in an explanted pig heart system. *Journal of Cardiovascular Magnetic Resonance*, 16:82, 2014.

[195] JU Fluckiger, BC Benefield, L Bakhos, KR Harris and DC Lee. A Comparison of Theory-Based and Experimentally Determined Myocardial Signal Intensity Correction Methods in First-Pass Perfusion Magnetic Resonance Imaging. *Comput Math Methods Med*, 2015:843741, 2015.

[196] A Schuster, M Sinclair, N Zarinabad, M Ishida, JP van den Wijngaard, M Paul, P van Horssen, ST Hussain, D Perera, T Schaeffter, JA Spaan, M Siebes, E Nagel and A Chiribiri. A quantitative high resolution voxel-wise assessment of myocardial blood flow from contrast-enhanced first-pass magnetic resonance perfusion imaging: microsphere validation in a magnetic resonance compatible free beating explanted pig heart model. *Eur Heart J Cardiovasc Imaging*, 16(10):1082–92, 2015.

[197] JB Bassingthwaighte, MA Malone, TC Moffett, RB King, IS Chan, JM Link and KA Krohn. Molecular and Particulate Depositions for Regional Myocardial Flows in Sheep. *Circ Res*, 66(5):1328–1344, 1990.

[198] G Morton, A Chiribiri, M Ishida, ST Hussain, A Schuster, A Indermuehle, D Perera, J Knuuti, S Baker, E Hedstrom, P Schleyer, M O'Doherty, S Barrington and E Nagel. Quantification of absolute myocardial perfusion in patients with coronary artery disease: comparison between cardiovascular magnetic resonance and positron emission tomography. *J Am Coll Cardiol*, 60(16):1546–55, 2012.

[199] PD Gatehouse, J Keegan, LA Crowe, S Masood, RH Mohiaddin, KF Kreitner and DN Firmin. Applications of phase-contrast flow and velocity imaging in cardiovascular MRI. *Eur Radiol*, 15(10):2172–84, 2005.

[200] J Jiang, C Strother, K Johnson, S Baker, D Consigny, O Wieben and J Zagzebski. Comparison of blood velocity measurements between ultrasound Doppler and accelerated phase-contrast MR angiography in small arteries with disturbed flow. *Phys Med Biol*, 56(6):1755–73, 2011.

[201] C Lally, AJ Reid and PJ Prendergast. Elastic behavior of porcine coronary artery tissue under uniaxial and equibiaxial tension. *Ann Biomed Eng*, 32(10):1355–64, 2004.

[202] A Pandit, XY Lu, C Wang and GS Kassab. Biaxial elastic material properties of porcine coronary media and adventitia. *Am J Physiol Heart Circ Physiol*, 288: H2581–H2587, 2005.

[203] M Zamir. *The Physics of Coronary Blood Flow.* Springer, 2005. ISBN 978-0387252971.

[204] A Hundertmark-Zaušková and M Lukáčová-Medvid'ová. Numerical study of shear-dependent non-Newtonian fluids in compliant vessels. *Computers & Mathematics with Applications*, 60(3):572–590, 2010.

[205] V Kanyanta, A Ivankovic and A Karac. Validation of a fluid-structure interation numerical model for predicting flow transients in arteries. *Journal of Biomechanics*, 42(11):1705–1712, 2009.

[206] X Ge, Z Yin, Y Fan, Y Vassilevski and F Liang. A multi-scale model of the coronary circulation applied to investigate transmural myocardial flow. *Int J Numer Method Biomed Eng*, 34(10):e3123, 2018.

[207] Z Duanmu, M Yin, X Fan, X Yang and X Luo. A patient-specific lumped-parameter model of coronary circulation. *Sci Rep*, 8(1):874, 2018.

[208] JM Idee, N Fretellier, C Robic and C Corot. The role of gadolinium chelates in the mechanism of neprhogenic systemic fibrosis: A critical update. *Crit Rev Toxicol*, 44 (10):895–913, 2014.

[209] JM Idee, M Port, D A., L E. and C Corot. Involvement of Gadolinium Chelates in the Mechanism of Nephrogenic Systemic Fibrosis: An Update. *Radiol Clin North Am*, 47 (5):855–869, 2009.

[210] L Lüdemann, B Nafz, F Elsner, C Große-Siestrup, M Meissler, N Kaufels, H Rehbein, PB Persson, HJ Michaely, P Lengsfeld, M Voth and M Gutberlet. Absolute Quantification of Regional Renal Blood Flow in Swine by Dynamic Contrast-Enhance Magnetic Resonance Imaging Using a Blood Pool Contrast Agent. *Investigative Radiology*, 44 (3):125–134, 2009.

[211] L Ludemann, D Prochnow, T Rohlfing, T Franiel, C Warmuth, M Taupitz, H Rehbein and D Beyersdorff. Simultaneous quantification of perfusion and permeability in the prostate using dynamic contrast-enhanced magnetic resonance imaging with an inversion-prepared dual-contrast sequence. *Ann Biomed Eng*, 37(4):749–62, 2009.

[212] HP Do, TR Jao and KS Nayak. Myocardial arterial spin labeling perfusion imaging with improved sensitivity. *Journal of Cardiovascular Magnetic Resonance*, 16(1):15, 2014.

[213] Z Zun, EC Wong and KS Nayak. Assessment of myocardial blood flow (MBF) in humans using arterial spin labeling (ASL): feasibility and noise analysis. *Magnetic Resonance in Medicine*, 62(4):975–983, 2009.

[214] Z Zun, P Varadarajan, RG Pai, EC Wong and KS Nayak. Arterial spin labeled CMF detects clinically relevant increase in myocardial blood flow with vasodilation. *JACC Cardiovasc Imaging*, 4(12):1253–61, 2011.

[215] M Ingrisch and S Sourbron. Tracer-kinetic modeling of dynamic contrast-enhanced mri and ct: a primer. *Journal of Pharmacokinetics and Pharmacodynamics*, 40(3): 281–300, 2013.

[216] FC Timothy, DW Rettmann, AH Aletras, SL LIao, TJ L., RS Balaban and AE Arai. Absolute myocardial perfusion in canines measured by using dual-bolus first-pass mr imaging. *Radiology*, 232(3):677–684, 2004.

[217] L Ostergaard, RM Weisskoff, DA Chesler, C Gyldensted and BR Rosen. High Resolution Measurement of Cerebral Blood Flow using Intravascular Tracer Bolus Passages. *Magnetic Resonance in Medicine*, 36(5):715–725, 1996.

[218] O Wu, L Ostergaard, RM Weisskoff, T Benner, BR Rosen and S G. Tracer arrival timing-insensitive technique for estimating flow in MR perfusion-weighted imaging using singular value decomposition with a block-circulant deconvolution matrix. *Magnetic Resonance in Medicine*, 50(1):164–174, 2003.

[219] F Calamante, DG Gadian and A Connelly. Quantification of bolus-tracking MRI: Improved charaterization of the tissue residue function using Tikhonov regularization. *Magnetic Resonance in Medicine*, 50(6):1237–1247, 2003.

[220] M Zreik, RW van Hamersvelt, JM Wolterink, T Leiner, MA Viergever and I Isgum. Automatic Detection and Characterization of Coronary Artery Plaque and Stenosis using Recurrent Convolutional Nerual Network in Coronary CT Angiography. In *1st Conference on Medical Imaging with Deep Learning (MIDL)*, Amsterdam, Netherlands, 2018.

[221] O Kjerland. *Segmentation of Coronary Arteries from CT-scans of the heart using Deep Learning.* Master thesis, Norwegian University of Science and Technology, Department of Computer Science, 2017.

[222] F Ring. *Deep Learning for Coronary Artery Segmentaion in CTA Images.* Master thesis, Chalmers University of Technology, Depeartment of Electrical Engineering, 2018.

[223] MK Kolandavel, ET Fruend, S Ringgaard and PG Walker. The effects of time varying curvature on species transport in coronary arteries. *Ann Biomed Eng*, 34(12):1820–32, 2006.

[224] A Theodorakakos, M Gavaises, A Andriotis, A Zifan, P Liatsis, I Pantos, EP Efstathopoulos and D Katritsis. Simulation of cardiac motion on non-Newtonian, pulsating flow development in the human left anterior descending coronary artery. *Phys Med Biol*, 53(18):4875–92, 2008.

[225] A Javadzadegan, ASC Yong, M Chang, MKC Ng, M Behnia and L Kritharides. Haemodynamic assessment of human coronary arteries is affected by degree of freedom of artery movement. *Computer Methods in Biomechanics and Biomedical Engineering*, 20(3):260–272, 2016.

[226] KI Aycock, RL Campbell, KB Manning and BA Craven. A resolved two-way coupled CFD/6-DOF approach for predicting embolus transport and the embolus-trapping efficiency of IVC filters. *Biomech Model Mechanobiol*, 16(3):851–869, 2017.

[227] C-H Huang and A Tsourkas. Gd-based macromolecules and nanoparticles as magnetic resonance contrast agents for molecular imaging. *Curr Top Med Chem*, 13(4):411–421, 2013.

[228] M Soltani and P Chen. Numerical Modeling of Interstitial Fluid Flow Coupled with Blood Flow through a Remodeled Solid Tumor Microvascular Network. *PLoS One*, 8 (6):e67025, 2013.

[229] S Qiang. *Numerical Simulation of Blood Flow through Permeable Vascular Network Embedded in Tumour Porous Interstitium.* Phd thesis, University Colloge London, Department of Mechanical Engineering, 2011.

[230] *OpenFOAM User Guide - Version 2.3.1.* OpenFOAM Foundation, 2014.

[231] *OpenFOAM Programmer's Guide - Version 2.3.1.* OpenFOAM Foundation, 2014.

List of Figures

List of Tables

Appendix A

Appendix

A.1 List of Abbreviations

AIF arterial input function

BC boundary condition

BV blood volume

CA contrast agent

CAD coronary artery disease

CE contrast-enhanced

CFD Computational Fluid Dynamics

CFL Courant-Friedrichs-Lewy

CFR coronary flow reserve

CT Computed Tomography

FFR Fractional Flow Reserve

FOV field of view

FSI fluid structure interactions

FVM Finite Volume Method

HPC High Performance Computing

iFR instantaneous wave-free ratio

LAD Left Anterior Descending

LCT left coronary tree

LCX Left Circumflex

LES Large Eddy Simulation

LHS left hand side

LMCA left main coronary artery

LUDS linear upwind differencing scheme

LV left ventricle

MBF myocardial blood flow

MPR myocardial perfusion reserve

MR magnetic resonance

MRI Magnetic Resonance Imaging

MTT mean transit time

MVTT mean vascular transit time

NMR nuclear magnetic resonance

PET Positron Emission Tomography

PIMPLE Combination of PI(SO) and (SI)MPLE algorithm

PISO Pressure-Implicit with Splitting of Operators

RCA right coronary artery

RCT right coronary tree

RD relative dispersion

RF radio frequency

RHS right hand side

RV right ventricle

SIMPLE Semi-Implicit Method for Pressure Linked Equations

SPECT Single-Photon Emission Computed Tomography

TBF tissue blood flow

UDS upwind differencing scheme

VBF volume blood flow

VTF vascular transport function

A.2 Used C++-programs

capillaryPressure.cpp

The C++-program *capillaryPressure.cpp* calculates how the electrical circuit described in Section 3.2 behaves.

First, required libraries and namespace are loaded:

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <stdint.h>
#include <string>

using namespace std;
```

The function `changingCapacitorVoltage` is defined, which depends on six variables:

- `timeIncrement`: Δt, incremental change of time variable,
- `previousVoltage`: $U(t-\Delta t)$, initial voltage across capacitor,
- `current`: $I(t)$, current flowing through capacitor at time t,
- `previousCurrent`: $I(t-\Delta t)$, current flowing through capacitor at time $t-\Delta t$,
- `capacity`: C, capacity of capacitor in circuit (i.e. compliance of the vessels within tissue),
- `density`: ρ, density of fluid.

Based on the initial voltage $U(t-\Delta t)$, the voltage across the capacitor at time t, $U(t)$ is returned according to eq. pC.

```
double changingCapacitorVoltage(double timeIncrement, double previousVoltage,
    double current, double previousCurrent, double capacity, double density)
{
  double result=0;
  result = previousVoltage + 1/capacity * (current - previousCurrent)/2 *
      timeIncrement/density;
  return result;
}
```

Next a function to count the number of lines in a file is defined:

```
int getNumberOfLines(string file) // function to obtain number of Lines from File
    with myocardial Pressure. Name of file is inserted in command line when programm
     is executed.
{
  int number_of_lines=0;
  string line;
  ifstream myfile(file.c_str());

  if(myfile.is_open())
  {
    while(!myfile.eof())
    {
      {
        getline(myfile,line);
        if (line.empty()) // make sure, only non-blank lines are counted!
        {
//          cout << "Empty line." << endl;
        }
        else
        {
```

```
//            cout << line << endl;
            number_of_lines++;
          }
        }
      }
      myfile.close();
    }

//  cout << "Number of lines in " << file << " is " << number_of_lines << endl;

  return number_of_lines;
}
```

The main program starts. A boolean `stressRestState` is defined by user intervention to distinguish between stress and rest state. Arterial, capillary and venous resistances as well as corresponding vessel compliance (cf. eq. 3.8) are defined with ratios according to their contribution to the total vascular flow resistance as described in Section 3.2. Since this program is devised to return pressures at mid-capillary level, the values are chosen to reflect the full vascular tree as in fig. 3.3.

```
int main()
{
  bool stressRestState;
  double R_arterial=345.6e8, R_capillary1=72e8, R_capillary2=72e8, R_venous=86.4e8;
  double capacitance=0;
  cout << "if rest state enter 0, if stress state enter 1: " ;
  cin >> stressRestState;
  if(stressRestState == 0)
  {
    cout << " REST STATE! " << endl;
    capacitance = 2.75e-11;
    cout << "total arterial resistance R_arterial: " << R_arterial << " kg/(m^4*s).
        " << endl
         << "capillary Resistance 1: R_capillary1: " << R_capillary1 << " kg/(m^4*s
             )." << endl
         << "capillary Resistance 2: R_capillary2: " << R_capillary2 << " kg/(m^4*s
             )." << endl
         << "venous Resistance: R_venous: " << R_venous << " kg/(m^4*s)." << endl
         << "compliance of intramyocardial vessels: capacitance: " << capacitance
             << " m^4*s^2/kg." << endl;
  }
```

If the calculations are performed for the stress state, resistance and compliance values are reduced according to [48].

```
  else if(stressRestState == 1)
  {
    cout << " STRESS STATE! " << endl;
    capacitance = 6.75e-11;
    R_arterial *= 0.25;//estimated total resistance of one artery in model and
        arterioles behind
//    R_capillary1 = 72e8;  //capillary resistance does not change under stress!
//    R_capillary2 = 72e8;
    R_venous *=0.02;
    cout << "total arterial resistance R_arterial: " << R_arterial << " kg/(m^4*s) 
        (reduced by 86%)." << endl
         << "capillary Resistance 1: R_capillary1: " << R_capillary1 << " kg/(m^4*s
             ) (unchanged)." << endl
         << "capillary Resistance 2: R_capillary2: " << R_capillary2 << " kg/(m^4*s
             ) (unchanged)." << endl
         << "venous Resistance: R_venous:" << R_venous << " kg/(m^4*s) (reduced by 
             98%)." << endl
         << "compliance of intramyocardial vessels: capacitance: " << capacitance
             << " m^4*s^2/kg." << endl;
  }
```

Previously defined files with timesptes of ventricular and aorta pressure are stored in according arrays. For simulations with incompressible fluids (like blood), in OpenFOAM, the pressure is scaled by the fluid's density, $p \to p/\rho$. Accordingly, the pressure values in the files must be adapted.

```
  int numberOfTimesteps;
  string filename;
  cout << "What is the name of the input-File with ventricular pressure timesteps 
      for one cycle (list WITHOUT times!!)?" << endl;
  cin >> filename;

  string filename1;
  cout << "What is the name of the input-File with inlet pressure timesteps for one
      cycle (list WITHOUT times!!)?" << endl;
  cin >> filename1;

  numberOfTimesteps = getNumberOfLines(filename);
  cout << "Number of timesteps in ventricular pressure file is: " <<
      numberOfTimesteps << endl;


  double myocardialPressureArray[numberOfTimesteps];
// readin of pressure timesteps from file into myocardialPressureArray
  ifstream infile(filename.c_str());
  int s=0;
  string line;
  stringstream Str;

  if(infile.is_open())
  {
    while(!infile.eof())
    {
      {
        getline(infile,line);
        if (line.empty()) //make sure empty lines are not read into array.
        {
//            cout << "Empty line." << endl;
        }
        else
        {
          Str << line;
          Str >> myocardialPressureArray[s];
          Str.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
          s++;
        }
      }
    }
    infile.close();
  }

  double inletPressureArray[numberOfTimesteps];

// readin of inlet pressure timesteps from file into inletPressureArray
  ifstream infile1(filename1.c_str());
  int t=0;
  string line1;
  stringstream Str1;

  if(infile1.is_open())
  {
    while(!infile1.eof())
    {
      {
        getline(infile1,line1);
        if (line1.empty()) //make sure empty lines are not read into array.
        {
//            cout << "Empty line." << endl;
        }
        else
        {
          Str1 << line1;
          Str1 >> inletPressureArray[t];
          Str1.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
//            cout << "inletPressureArray" << s << ": " << inletPressureArray[t] <<
    endl;
```

```
          t++;
        }
      }
    }
    infile1.close();
}
```

The variable `timeStepSize` (i.e. Δt) is defined based on cardiac cycle duration.

```
  double cycleDuration = 0;
  cout << "Insert cardiac cycle duration (in s): " << endl;
  cin >> cycleDuration;
  double timeStepSize=0;
  timeStepSize = cycleDuration/numberOfTimesteps;
  cout << "time step size is: " << timeStepSize << " s" << endl;
```

The density of blood is chosen as $1060^{kg}/_{m^3}$.

```
  double rho=1060;
  cout << " density rho: " << rho << " kg/m^3. " << endl;
```

A termination criterion is defined to check if periodicity of the system is reached. Usually periodicity is reached between three to five cycles.

```
    // choose tolerance for termination criterion
  double tolerance = 0.005;
  cout << "termination criterion for capacitor pressure is " << tolerance << " m^2/
      s^2, corresponding to ~" << tolerance*rho << " kg/(m*s^2) or ~" << tolerance*
      rho/133.322 << " mmHg . " << endl;

  int maxNumberOfCycles=50;
  cout << "maximum number of cycles to compute before automatic termination: " <<
      maxNumberOfCycles << "." << endl;
//  maxNumberOfIterations = maxNumberOfCycles*numberOfTimesteps;
```

The variable `capacitorVoltage[][]` is initialized to store the voltage across the capacitor. To start, an initial voltage is required as eq. 3.2 suggests.

```
  double capacitorVoltage[maxNumberOfCycles][numberOfTimesteps];

  cout << "Enter initial capacitor voltage in m^2/s^2: " << endl;
  cin >> capacitorVoltage[0][numberOfTimesteps-1];
```

Since the considered blood flow circuit disembogues into the atrium (cf. Section 3.2), mean atrial pressure is assigned as the lowest potential in the circuit.

```
  double meanRightAtrialPressure = 0;
  cout << "Enter mean right atrial pressure in m^2/s^2: (3.75 mmHg ~ 500 kg/(m*s^2)
      ~ 0.47 m^2/s^2" << endl;
  cin >> meanRightAtrialPressure;
  cout << "meanRightAtrialPressure: " << meanRightAtrialPressure << " m^2/s^2." <<
      endl;
```

Arrays, which contain flow values in the three branches of the circuit are defined. Initial flow values at time $t - \Delta t$ have to be calculated according to the chosen initial capacitor voltage.

```
  double modelCurrent[numberOfTimesteps], venousCurrent[numberOfTimesteps],
      capacitorCurrent[numberOfTimesteps]; //required for computation of new
      capacitorVoltage. all currents are positive in direction of the mutual node of
       the three branches. All three currents should add to zero!

  modelCurrent[numberOfTimesteps-1] = rho*(inletPressureArray[numberOfTimesteps-1]
      - (myocardialPressureArray[numberOfTimesteps-1] + capacitorVoltage[0][
      numberOfTimesteps-1]))/(R_arterial + R_capillary1);
  venousCurrent[numberOfTimesteps-1] = rho*(meanRightAtrialPressure - (
      myocardialPressureArray[numberOfTimesteps-1] + capacitorVoltage[0][
      numberOfTimesteps-1]))/(R_capillary2 + R_venous);
```

```
capacitorCurrent[numberOfTimesteps-1] = - modelCurrent[numberOfTimesteps-1] -
    venousCurrent[numberOfTimesteps -1];

cout << "initial modelCurrent (volume flow): " << modelCurrent[numberOfTimesteps
    -1] << " m^3/s." << endl;
cout << "initial venousCurrent (volume flow): " << venousCurrent[
    numberOfTimesteps-1] << " m^3/s." << endl;
cout << "initial capacitorCurrent (volume flow): " << capacitorCurrent[
    numberOfTimesteps-1] << " m^3/s." << endl;
```

Output streams for the results and a control file are defined.

```
ofstream output;
ofstream output2;
output.open ("control.txt");
```

The following `for`-loop creates scaled myocardial pressure value arrays for different locations between epi- to endocardium (scaling factor 0.65 to 1.25, according to [96].

```
// for loop for different capillaryPressureTimestepFiles:

  double tmpMyocardialPressureArray[numberOfTimesteps];
  double myocardialPressureScalingFactor=0;
  for(int K=0; K<7; K++)
  {
    cout << "myocardialPressureScalingFactor: " << 0.65+K*0.1 << ". " << endl;
    std::stringstream numberedFilenames2;
    std::stringstream numberedFilenames;
    myocardialPressureScalingFactor=0.65+K*0.1;
    numberedFilenames << "capillaryPressureTimesteps" <<
        myocardialPressureScalingFactor*100 << ".txt";
    numberedFilenames2 << "capacitorPressureTimesteps" <<
        myocardialPressureScalingFactor*100 << ".txt";
    ofstream finalOutput(numberedFilenames.str().c_str());
    ofstream finalOutput2(numberedFilenames2.str().c_str());
    for(int L=0; L<numberOfTimesteps; L++)
    {
      tmpMyocardialPressureArray[L]=myocardialPressureScalingFactor*
          myocardialPressureArray[L];
    }
```

A boolean is intialized for periodicity checking. Several `for`-loops for the actual calculations follow.

```
  bool isBreak = false;

  for(int cycleCounter=0; cycleCounter < maxNumberOfCycles; cycleCounter++)
  {

    for(int timeCounter=0;timeCounter < numberOfTimesteps; timeCounter++)
    {
```

With the help of the array `deviation[]`, periodicity is assessed between all time steps of two subsequent cycles.

```
  // check if periodicity to desired tolerance has occurred in subsequent cycles
    if(cycleCounter > 1)
    {
    if(timeCounter == numberOfTimesteps-1)
    {
      output << "timeCounter is " << timeCounter << ", the end of for-loop is 
          reached. checking for periodicity." << endl; //remainder << endl;
      double deviation[numberOfTimesteps];
      double maxDeviation=0;
      int timestepOfMaxDeviation=0;
      for(int r=0; r<numberOfTimesteps; r++)
      {
        deviation[r] = fabs(capacitorVoltage[cycleCounter-2][r] - capacitorVoltage[
            cycleCounter-1][r]);
        output << "␣capacitorVoltage␣cycle␣" << cycleCounter-2 << "␣timeCounter␣"
            << r << ":␣" << capacitorVoltage[cycleCounter -2][r] << endl;
        output << "␣capacitorVoltage␣cycle␣" << cycleCounter-1 << "␣timeCounter␣"
            << r << ":␣" << capacitorVoltage[cycleCounter -1][r] << endl;

          if(deviation[r] > maxDeviation)
          {
            maxDeviation = deviation[r];
            timestepOfMaxDeviation = r;
            output << "␣maxDeviation␣" << maxDeviation << "␣newly␣set␣at␣timestep␣"
                 << r << endl;
          }
        output << "␣difference␣at␣cycle␣" << cycleCounter-1 << "␣timeCounter␣" << r
             << ":␣" << deviation[r] << endl;
        output << "␣maxDeviation␣at␣cycle␣" << cycleCounter-1 << "␣timeCounter␣" <<
             timestepOfMaxDeviation << ":␣" << maxDeviation << endl;
      }
```

If periodicity is reached, the myocardial pressure values at pre-capillary level are written to disk for the full cycle and a boolean `isBreak` is set to `true`.

```
      if(maxDeviation < tolerance)
      {
        output << "maxDeviation␣is␣smaller␣than␣desired␣tolerance␣after␣" <<
            cycleCounter-1 << "␣cycles.␣Write␣resulting␣capacitor␣Pressures␣and␣
            capillaryPressure␣(BETWEEN␣ARTERIOLES␣AND␣CAPILLARIES!!)␣in␣file␣and␣
            leave!␣" << endl;
        for(int s=0; s<numberOfTimesteps; s++)
        {
        if(K==4)
        {
          output2 << s*timeStepSize << "␣" << capacitorVoltage[cycleCounter -1][s]
              << endl;
        }
        finalOutput << tmpMyocardialPressureArray[s] + capacitorVoltage[
            cycleCounter -1][s] + (R_capillary1/(R_arterial + R_capillary1) * (
            inletPressureArray[s] - (tmpMyocardialPressureArray[s] +
            capacitorVoltage[cycleCounter -1][s]))) << endl;
        finalOutput2 << capacitorVoltage[cycleCounter -1][s] << endl;
        }
        isBreak = true;
        cout << "termination␣criterion␣fulfilled␣after␣cycle:␣" << cycleCounter <<
            ".␣" << endl;
      }
      else
      {
        output << "␣maxDeviation␣=␣" << maxDeviation << "␣is␣larger␣than␣desired␣
            tolerance␣" << tolerance << "␣at␣timestep␣" << timestepOfMaxDeviation <<
             "␣within␣cycle,␣go␣on!␣" << endl;
      }

    }
    }
```

The intramyocardial pressure values used for the calculations are written to the control file.

```

      output << "intraMyocardialPressureArray␣value␣at␣time␣" << timeCounter*
          timeStepSize << "␣is:␣" << tmpMyocardialPressureArray[timeCounter] << endl
          ;
```

The calculation of the capacitor voltage $U(t)$ according to eq. 3.2 is performed by calling the initially defined function `changingCapacitorVoltage`.

```

      capacitorVoltage[cycleCounter][timeCounter] = changingCapacitorVoltage(
          timeStepSize,   capacitorVoltage[cycleCounter][timeCounter-1],
rho*(inletPressureArray[timeCounter] - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[cycleCounter][timeCounter-1]))/(R_arterial + R_capillary1) +
    rho*(meanRightAtrialPressure - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[cycleCounter][timeCounter-1]))/(R_capillary2 + R_venous),
capacitorCurrent[timeCounter-1],
capacitance,
rho);
```

The corresponding new flow values in the three circuit branches are calculated and written to the control file.

```
      modelCurrent[timeCounter] = rho*(inletPressureArray[timeCounter] - (
          tmpMyocardialPressureArray[timeCounter] + capacitorVoltage[cycleCounter][
          timeCounter-1]))/(R_arterial + R_capillary1);
      venousCurrent[timeCounter] = rho*(meanRightAtrialPressure - (
          tmpMyocardialPressureArray[timeCounter] + capacitorVoltage[cycleCounter][
          timeCounter-1]))/(R_capillary2 + R_venous);
      capacitorCurrent[timeCounter] = - modelCurrent[timeCounter] - venousCurrent[
          timeCounter];

      output << "modelCurrent: " << modelCurrent[timeCounter] << endl;
      output << "venousCurrent: " << venousCurrent[timeCounter] << endl;
      output << "capacitorCurrent: " << capacitorCurrent[timeCounter] << endl;

  output.precision(10);
  output << "capacitorVoltage at time " << timeCounter*timeStepSize << " in cycle "
       << cycleCounter << " is: " << capacitorVoltage[cycleCounter][timeCounter] <<
      endl;



    }
```

If periodicity is reached, the computation is stopped, otherwise, it goes on.

```
  if (isBreak == true)
  {
    break;
  }
  }
//  numberedFilenames.clear();
  }
  output2.close();
  output.close();
  return 0;
}
```

scalingLawResistance.cpp

Based on assumptions about the self-similarity of the coronary vascular tree (cf. Section 3.2) the C++-program *scalingLawResistance.cpp* calculates the flow resistances of vascular crowns associated to outlet radii of a 3D vascular geometry.

First required libraries and namespace are loaded similarly to Section A.2 and several general functions are defined.

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <stdint.h>
#include <string>

using namespace std;

int getNumberOfOutlets(string file) // function to obtain number of outlets from
    file with listed outlet radii. Name of file is inserted in command line when
    programm is executed.
{
  int number_of_lines=0;
  string line;
  ifstream myfile(file.c_str());

  if(myfile.is_open())
  {
    while(!myfile.eof())
    {
      {
        getline(myfile,line);
        if (line.empty()) // make sure, only non-blank lines are counted!
        {
//          cout << "Empty line." << endl;
        }
        else
        {
//          cout << line << endl;
          number_of_lines++;
        }
      }
    }
    myfile.close();
  }

//  cout << "Number of lines in " << file << " is " << number_of_lines << endl;

  return number_of_lines;
}

double readFile(string fileName, double array[])
{
  ifstream infile(fileName.c_str());
  int s=1;
  string line;
  stringstream Str;

  if(infile.is_open())
  {
    while(!infile.eof())
    {
      {
        getline(infile,line);
        if (line.empty()) //make sure empty lines are not read into array.
        {
//          cout << "Empty line." << endl;
        }
        else
        {
          Str << line;
          Str >> array[s];
          Str.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
//          cout << "outletRadius" << s << ":" << outletRadiusArray[s] << endl;
          s++;
        }
      }
    }
    infile.close();
  }

  return 0;
}
```

The function MU_app_rel is defined to account for changes of blood viscosity depending on vessel diameter according to [104].

```
double MU_app_rel(double diameter) // apparent relative blood viscosity after Pries
    , diameter in micrometers required
{
  double result = 0;
    result = 220 * exp(-1.3*diameter) + 3.2 - 2.44 * exp(-0.06*pow(diameter ,
        0.645));
//    cout << "mu_rel = " << result << endl;
  return result;
}
```

Several variables and arrays are initialized in the main function. Please see the in-code comments for explanations. File names containing required data are defined.

```
int main()
{
  int N_outlets = 0; //#outlets
  int N_orders=11; //#vessel orders
  double lengthArray[N_orders]; //lengths of vessels in orders
  double maxDiameterArray[N_orders];// maximum diameter in order
  double meanDiameterArray[N_orders];// mean diameters in order
  double volumeArray[N_outlets];//vessel volume between inlet and outlet
  double numberOfVesselsArray[N_orders];//#vessels in order
  double distanceArray[N_outlets];// distance between inlet and outlet

  //calculated variables
  double inletDiameter = 0;
  double diameterScaleRatio = 0;
  double sumLength[N_outlets];
  double equivalentResistance[N_outlets];

  string filename1;
  //values from Kassab et al, 1997
  string filename2 = "lengths.txt";
  string filename3 = "maxDiameters.txt";
  string filename4 = "meanDiameters.txt";
  string filename5 = "numbers.txt";
  //values computed by separate script
  string filename6 = "distances.txt";
  string filename7 = "volumes.txt";

// choose input file with list of outlet radii (in mm)
  cout << "What␣is␣the␣name␣of␣the␣input−File␣with␣outlet−radii␣in␣millimeters?" <<
       endl;
  cin >> filename1;

  N_outlets = getNumberOfOutlets(filename1);
  cout << "Number␣of␣outlets␣is:␣" << N_outlets << endl;
```

Arrays to store arterial, capillary and venous resistances are initialized. The values are calculated using the above variables and arrays.

```
// arrays for resistances and outletradii are created
  double resistanceFractionArray[N_outlets];
  double finalVesselResistanceArray[N_outlets];
  double capillaryResistanceArray[N_outlets];
  double venousResistanceArray[N_outlets];
  double outletRadiusArray[N_outlets];

// readin of outletradii from file into outletRadiusArray
  readFile(filename1, outletRadiusArray);

  for(int i=1; i<=N_outlets; ++i)
  {
    cout << "OutletRadius" << i << "␣is␣" << outletRadiusArray[i] << endl;
  }

  cout << "name␣of␣the␣input−File␣with␣lengths␣of␣vessels␣in␣millimeters:␣" <<
      filename2 << endl;
  cout << "name␣of␣the␣input−File␣with␣max−diameter␣in␣micrometers:␣" << filename3
      << endl;
  cout << "name␣of␣the␣input−File␣with␣mean−diameters␣in␣micrometers:␣" <<
      filename4 << endl;
  cout << "name␣of␣the␣input−File␣with␣number␣of␣vessels␣in␣orders:␣" << filename5
      << endl;
  cout << "name␣of␣input−File␣with␣distances␣between␣inlet␣and␣outlets␣in␣
      millimeters:␣" << filename6 << endl;
  cout << "name␣of␣the␣input−File␣with␣volumes␣between␣inlet␣and␣outlets␣in␣m^3:"
      << filename7 << endl;


  cout << "Please␣insert␣inlet␣diameter␣of␣the␣model␣in␣micrometers" << endl;
  cin >> inletDiameter;
```

The read-in of the required data from files is executed.

```
readFile(filename2, lengthArray);
readFile(filename3, maxDiameterArray);
readFile(filename4, meanDiameterArray);
readFile(filename5, numberOfVesselsArray);
readFile(filename6, distanceArray);
readFile(filename7, volumeArray);
```

Since the inlet diameter of the 3D geometry can deviate from the largest diameter in [97], the scaling factor `diameterScaleRatio` is set.

```
diameterScaleRatio = inletDiameter/maxDiameterArray[N_orders];
cout << "inletDiameter:␣" << inletDiameter/1000 << "␣mm." << endl;
cout << "diameterScaleRatio:␣" << diameterScaleRatio << endl;

for(int i=1; i<=N_orders; ++i)
{
  cout << "length␣in␣order" << i << "␣is␣" << lengthArray[i] << endl;
}

for(int i=1; i<=N_orders; ++i)
{
  cout << "maxDiameter␣in␣order" << i << "␣is␣" << maxDiameterArray[i] << endl;
}

for(int i=1; i<=N_orders; ++i)
{
  cout << "meanDiameter␣in␣order" << i << "␣is␣" << meanDiameterArray[i] << endl;
}

for(int i=1; i<=N_outlets; ++i)
{
  cout << "volume␣between␣inlet␣and␣outlet" << i << "␣is␣" << volumeArray[i] <<
      endl;
}

for(int i=1; i<=N_orders; ++i)
{
  cout << "number␣in␣order" << i << "␣is␣" << numberOfVesselsArray[i] << endl;
}

for(int i=1; i<=N_outlets; ++i)
{
  cout << "distance␣to␣outlet␣" << i << "␣is␣" << distanceArray[i] << endl;
}
```

The variable `stressRestScaleFactor` is set by user input.

```
double stressRestScaleFactor = 1;
cout << "Please␣insert␣vasodilation␣factor␣of␣vessels␣behind␣model␣outlet␣(1␣OR␣
    1.15):" << endl;
cin >> stressRestScaleFactor;

cout << "stressRestScaleFactor:␣" << stressRestScaleFactor << endl;
```

The arithmetic mean of the distances between inlet and outlets is calculated. The values for standard viscosity [120] and blood density [122] are set.

```
double sumDistances=0;
double meanDistance=0;
for(int n=1; n<=N_outlets; ++n)
{
  sumDistances=distanceArray[n];
}
meanDistance=sumDistances/N_outlets;
cout << "mean␣distance␣is:␣" << meanDistance << endl;


double mu_0=0.0046, rho=1060;
cout << "Standard␣viscosity␣is:␣" << mu_0 << "␣kg/(m*s)␣" << endl;
cout << "density␣is:␣" << rho << "␣kg/m^3␣" << endl;
```

The array `numberOfVesselsScaling` is defined to estimate the number of vessels downstream of all outlets. Since the numbers of vessels per order are given by discrete numbers [97], this variable is interpolated linearly depending on the outlet's radius.

```
double numberOfVesselsScaling[N_outlets];
for(int r=1; r<=N_outlets; r++)
{

for(int i=1; i<=N_orders; ++i)
{
  if(2*outletRadiusArray[r]*1000 > diameterScaleRatio*meanDiameterArray[i-1] &&
      2*outletRadiusArray[r]*1000 < diameterScaleRatio*meanDiameterArray[i])

  {
    cout << "outletDiameter" << r << " is smaller than maxDiameter at order " <<
        i << " but larger than maxDiameter at order " << i-1 << endl;
    numberOfVesselsScaling[r] = numberOfVesselsArray[i-1] + (numberOfVesselsArray
        [i] - numberOfVesselsArray[i-1])/(diameterScaleRatio*meanDiameterArray[i]
        - diameterScaleRatio*meanDiameterArray[i-1])*((2*outletRadiusArray[r
        ]*1000) - diameterScaleRatio*meanDiameterArray[i-1]);
  }

}
  cout << "numberOfVesselsScaling at outlet" << r << ": " <<
      numberOfVesselsScaling[r] << endl;
}
```

Approximations for the flow resistances between inlet and each outlet are made under the assumption of laminar flow through a tube with length and volume given by the program described in Section A.2.[1]

```
for(int i=1; i<=N_outlets; i++)
{
  double result=0;
  equivalentResistance[i]=0;
  result=8*M_PI*mu_0*MU_app_rel((inletDiameter+2*1000*outletRadiusArray[i])/2)
      /3.2
  * pow(0.001*distanceArray[i],3)
  / (pow(volumeArray[i],2));
  equivalentResistance[i]=result;
cout << "vessel Resistance @ outlet" << i << " is: " << equivalentResistance[i]
   << endl;
}
```

Output streams and further variables required to compute the crown resistances are defined.

```
ofstream output("finalVesselResistances.txt");
ofstream output4("venousResistances.txt");

double fullTreeCumulativeLength = 0;//cumulative length of all vessels in full
    coronary tree, from most proximal stem radius to smallest arteriolar (pre-
    capillary) vessels
double resistanceScalingConstant = 0;
double maxResistance = 0;
```

The cumulative length of the full coronary tree, which includes all vessels from the most proximal stem radius (inlet) to the smallest arteriolar (i.e. pre-capillary) vessels is computed.

```
for(int i=1; i<=N_orders; ++i)
{
  fullTreeCumulativeLength += numberOfVesselsArray[i]*0.001*lengthArray[i];
}
cout << "Cumulative length of total vascular tree (all orders 1-11) is: 
    fullTreeCumulativeLength " << fullTreeCumulativeLength << " m" << endl;
```

[1]This is indeed an anticipation of flow resistances within the 3D model itself, however, this quantity is required for the usage of the program *outletFlows.cpp*. This is described in Section A.2 and an explanation why it is used is given in Chapter 6.

The same is computed for all vascular trees with the outlet radii as their stem and the associated numbers of vessels scaled by `numberOfVesselsScaling`.

```
cumulativeCrownLengths[N_outlets];

for(int t=1; t<=N_outlets; ++t)
{
cumulativeCrownLengths[t] = 0;
for(int l=1; l<=N_orders; ++l)
{
  if(2*outletRadiusArray[t]*1000 > diameterScaleRatio*maxDiameterArray[l])
  {
    cumulativeCrownLengths[t] += numberOfVesselsArray[l]/numberOfVesselsScaling[t
        ] *0.001*lengthArray[l]*diameterScaleRatio;
  }
}
}

for(int t=1; t<=N_outlets; ++t)
{
  cout << "cumulativeCrownLength @ outlet" << t << ": " << cumulativeCrownLengths
      [t] << " m." << endl;
}
```

Several required variables are defined and some require to be set by user intervention.

```
double meanInletPressure = 0;
double meanRightAtrialPressure = 0.47, meanArterialPressureLoss = 0,
    meanModelVolumeFlow = 0, meanCapillaryPressure = 0;
cout << "Is this human or pig data? Enter 0 for pig, 1 for human. " << endl;
bool humanOrPig;
double stemFlowCrownLengthScaling = 0;  //proportionality of stem flow to crown
    length
cin >> humanOrPig;
if(humanOrPig == 0)
{
  stemFlowCrownLengthScaling = 6E-9; //Factor 6E-9 m^3/(s*m) is
      stemFlowCrownLengthScaling for pigs from Zhou et al, PhysMedBiol 1999
}
```

The variable `stemFlowCrownLengthScaling` is taken from [98], which is based on pig data. In order to approximate this for human data, the factor is further scaled with `diameterScaleRatio`.[2]

```
else //humanOrPig == 1
{
  stemFlowCrownLengthScaling = 6E-9 * pow(diameterScaleRatio,2.333); //Factor 6E
      -9 m^3/(s*m) is stemFlowCrownLengthScaling for pigs from Zhou et al,
      PhysMedBiol 1999
}

cout << "What is mean inlet Pressure @rest in m^2/s^2?" << endl;
cin >> meanInletPressure;
cout << "Enter mean capillary pressure in m^2/s^2:" << endl;
cin >> meanCapillaryPressure;//requires mean value from PRE-capillaryPressure

cout << "meanInletPressure: " << meanInletPressure << " m^2/s^2." << endl;
cout << "meanCapillaryPressure: " << meanCapillaryPressure << " m^2/s^2." << endl
    ;
cout << "stemFlowCrownLengthScaliing: " << stemFlowCrownLengthScaling << " m^2/s.
    " << endl;
```

The mean flow through the full 3D geometry (`meanModelVolumeFlow` is estimated according to eq. 3.7 and adjusted according to stress or rest state.

```
meanModelVolumeFlow = stemFlowCrownLengthScaling * fullTreeCumulativeLength;
```

[2]Estimate from relation $F_{\text{tot}} \propto D_{\text{max}}$[36].

```
if(stressRestScaleFactor==1)
{
meanArterialPressureLoss = meanInletPressure - meanCapillaryPressure; //fraction
    of pressure loss across arteries/arterioles in coronary vasculature. stress or
     rest state already accounted for in meanCapillaryPressure
}
else if(stressRestScaleFactor==1.15)
{
meanArterialPressureLoss = meanInletPressure - meanCapillaryPressure; //fraction
    of pressure loss across arteries/arterioles in coronary vasculature. stress or
     rest state already accounted for in meanCapillaryPressure
meanModelVolumeFlow *= 4; //increased blood flow at stress
}
cout << "meanArterialPressureLoss: " << meanArterialPressureLoss << endl;
cout << "meanModelVolumeFlow: " << meanModelVolumeFlow << " m^3/s." << endl;
```

Finally, an estimate for the resistance of the full vascular tree from the most proximal stem diameter to the smallest pre-capillary arterioles is made as required by eq. 3.6. Subsequently, the `resistanceScalingConstant` (= const_R from eq. 3.4 is computed.

```
maxResistance = rho * meanArterialPressureLoss / meanModelVolumeFlow;
cout << "maxResistance: " << maxResistance << " kg/(m^4*s)" << endl;
resistanceScalingConstant = maxResistance * pow(inletDiameter/1000000,4) /
    fullTreeCumulativeLength;
cout << "resistanceScalingConstant: R_max*D_max^4/L_max: " <<
    resistanceScalingConstant << " kg/(m*s)" << endl;
```

The array `crownResistanceArray[]` is defined to store the desired outlet resistances calculated by eq. 3.4. The ratios of arterial, capillary and venous resistances are estimated by their contributions to total vascular resistance as stated in [48, 99],

$$\begin{array}{llll} \text{Rest:} & R_{Art} \sim 0.6 \cdot R_{tot}, & R_{Cap} \sim 0.25 \cdot R_{tot}, & R_{Ven} \sim 0.15 \cdot R_{tot} \\ \text{Stress:} & \sim 0.14 \cdot R_{Art}, & \sim R_{Cap}, & \sim 0.02 \cdot R_{Ven} \end{array}$$

```
double crownResistanceArray[N_outlets];
for(int r=1; r<=N_outlets; ++r)
{

crownResistanceArray[r] = resistanceScalingConstant * cumulativeCrownLengths[r]/(
    pow(2*0.001*outletRadiusArray[r],4));

finalVesselResistanceArray[r] = crownResistanceArray[r];
cout << "crownResistance @ outlet" << r << ": " << finalVesselResistanceArray[r]
    << endl; ;
if(stressRestScaleFactor==1)
{
capillaryResistanceArray[r] = finalVesselResistanceArray[r]/5;
}
else if(stressRestScaleFactor==1.15)
{
capillaryResistanceArray[r] = finalVesselResistanceArray[r]*1.44;
}
cout << "capillaryResistance @ outlet" << r << ": " << capillaryResistanceArray[r
    ] << endl;
finalVesselResistanceArray[r] += capillaryResistanceArray[r] +
    equivalentResistance[r];
cout << "finalVesselResistance after addition of capillary contribution and
    equivalentResistance before outlet is: " << finalVesselResistanceArray[r] <<
    endl;
if(stressRestScaleFactor==1)
{
venousResistanceArray[r] = capillaryResistanceArray[r] + 1.2*
    capillaryResistanceArray[r];
}
else if(stressRestScaleFactor==1.15)
{
  venousResistanceArray[r] = capillaryResistanceArray[r] + 0.03*
      capillaryResistanceArray[r];
  }
  cout << "capillary + venous Resistance @ outlet" << r << ": " <<
      venousResistanceArray[r] << endl;

// store resistances in textfile:
  output.precision(10);
  output4.precision(10);
  output << finalVesselResistanceArray[r] << endl;
  output4 << venousResistanceArray[r] << endl;

  }

  return 0;
}
```

outletFlows.cpp

The following C++-program *outletFlows.cpp* gives an estimation of how volume blood flow through a 3D model behaves. It is based on the boundary condition described in Section 3.2 and possesses several similar elements as the program *capillaryPressure.cpp* (cf. Section A.2). In the following, selected steps are presented in detail.

First, the required libraries and namespace are loaded:

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <stdint.h>
#include <string>

using namespace std;
```

The same function `changingCapacitorVoltage` as in *capillaryPressure.cpp* is defined to reflect eq. 3.2, which describes how the voltage across a capacitor behaves.

```
double changingCapacitorVoltage(double timeIncrement, double previousVoltage,
    double current, double previousCurrent, double capacity, double density)
{
  double result=0;
  result = previousVoltage + 1/capacity * (current - previousCurrent)/2 *
      timeIncrement/density;
  return result;
}
```

Several functions are defined to retrieve required data from input files.

```
int getNumberOfLines(string file) // function to obtain number of Lines from File
    with myocardial Pressure. Name of file is inserted in command line when programm
     is executed.
{
  int number_of_lines=0;
  string line;
  ifstream myfile(file.c_str());

  if(myfile.is_open())
  {
    while(!myfile.eof())
    {
      {
        getline(myfile,line);
        if (line.empty()) // make sure, only non-blank lines are counted!
        {
//            cout << "Empty line." << endl;
        }
        else
        {
```

```
//          cout << line << endl;
          number_of_lines++;
        }
      }
    }
    myfile.close();
  }

//  cout << "Number of lines in " << file << " is " << number_of_lines << endl;

  return number_of_lines;
}

double readFile(string fileName, double array[], int s)
{
  ifstream infile(fileName.c_str());
//  int s=0;
  string line;
  stringstream Str;

  if(infile.is_open())
  {
    while(!infile.eof())
    {
      {
        getline(infile,line);
        if (line.empty()) //make sure empty lines are not read into array.
        {
//          cout << "Empty line." << endl;
        }
        else
        {
          Str << line;
          Str >> array[s];
          Str.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
//          cout << "outletRadius" << s << ":" << outletRadiusArray[s] << endl;
          s++;
        }
      }
    }
    infile.close();
  }

  return 0;
}
```

The main function starts and several arrays to store resistance and capacitance (i.e. compliance) values of vessels are initialized. The venous and arterial resistances are read from files, which must be created beforehand with the described program *scalingLawResistance.cpp* described in Section A.2.

```
int main()
{
  int N_outlets=0;

  cout << "Reading␣file:␣finalVesselResistances.txt" << endl;

  N_outlets = getNumberOfLines("finalVesselResistances.txt");
  cout << "Number␣of␣outlets␣is:␣" << N_outlets << endl;

  double vesselResistancesArray[N_outlets];
  double venousResistancesArray[N_outlets];
  double capacitanceArray[N_outlets];

  readFile("finalVesselResistances.txt",vesselResistancesArray,1);

  for(int i=1; i<=N_outlets; i++)
  {
    cout << "arterial␣Resistance␣@␣outlet" << i << ":␣" << vesselResistancesArray[i
        ] << "␣kg/(m^4*s)." << endl;
}
cout << "Reading file: venousResistances.txt" << endl;
readFile("venousResistances.txt",venousResistancesArray,1);

for(int i=1; i<=N_outlets; i++)
{
  cout << "venous Resistance @ outlet" << i << ": " << venousResistancesArray[i]
      << " kg/(m^4*s)." << endl;

}
```

The boolean `stressRestState` is defined to distinguish stress and rest state. It must be set by user input.

```
bool stressRestState;
double capacitance=0;
cout << "if rest state enter 0, if stress state enter 1: " ;
cin >> stressRestState;
```

According to eq. 3.8 the vessels' conjugated compliances are defined depending on the ratio of the time constant $\tau = 1.14, 1.3$s for rest and stress state, respectively.[3]

```
for(int i=1; i<=N_outlets; i++)
{
if(stressRestState == 0)
{
  capacitanceArray[i] = 1.14/vesselResistancesArray[i];//
  cout << "capacitance @ outlet" << i << ": " << capacitanceArray[i] << " m^4*s
      ^2/kg." << endl;
  cout << "COMPUTATION FOR REST STATE!!!" << endl;
}
if(stressRestState == 1)
{
  capacitanceArray[i] = 1.3/vesselResistancesArray[i];//
  cout << "capacitance @ outlet" << i << ": " << capacitanceArray[i] << " m^4*s
      ^2/kg." << endl;
  cout << "COMPUTATION FOR STRESS STATE!!!" << endl;
}
}
```

Next, external files with required data (outlet radii, arterial and venous resistances etc.) are read in.

```
int numberOfTimesteps;
string filename;
// choose input file with list of outlet radii (in m)
cout << "What is the name of the input-File with ventricular pressure timesteps 
    for one cycle (list WITHOUT times!!)?" << endl;
cin >> filename;

string filename1;
// choose input file with list of outlet radii (in m)
cout << "What is the name of the input-File with inlet pressure timesteps for one
    cycle (list WITHOUT times!!)?" << endl;
cin >> filename1;

numberOfTimesteps = getNumberOfLines(filename);
cout << "Number of timesteps in ventricular pressure file is: " <<
    numberOfTimesteps << endl;
```

The cardiac cycle duration is set manually.

```
double cycleDuration = 1;
cout << "Insert cardiac cycle duration (in s): " << endl;
cin >> cycleDuration;
double timeStepSize=0;
timeStepSize = cycleDuration/numberOfTimesteps;
cout << "time step size is: " << timeStepSize << " s" << endl;
```

[3]The value $\tau = 1.3$ for stress state is chosen as described in Section 3.2 to obtain a realistic ratio of diastolic to systolic flow of ~ 4. Please also compare [34].

An array to hold the outlet specific ventricular pressure scaling values is defined. These values are previously computed with the `C++`-program *computeBarycenter.cpp* presented in Section A.2.

```
  double ventricularPressureScalingArray[N_outlets];
  readFile("ventricularPressureScaling.txt",ventricularPressureScalingArray,1);
  for(int i=1; i<=N_outlets; i++)
  {
    cout << "ventricularPressureScaling at outlet" << i << ": " <<
        ventricularPressureScalingArray[i] << endl;

  }
```

The files with the aortic and ventricular pressure timesteps are read in.

```
  double myocardialPressureArray[numberOfTimesteps];
  double tmpMyocardialPressureArray[numberOfTimesteps];
// readin of pressure timesteps from file into myocardialPressureArray
  ifstream infile(filename.c_str());
  int s=0;
  string line;
  stringstream Str;

  if(infile.is_open())
  {
    while(!infile.eof())
    {
      {
        getline(infile,line);
        if (line.empty()) //make sure empty lines are not read into array.
        {
//          cout << "Empty line." << endl;
        }
        else
        {
          Str << line;
          Str >> myocardialPressureArray[s];
          Str.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
//          cout << "outletRadius" << s << ":" << outletRadiusArray[s] << endl;
          s++;
        }
      }
    }
    infile.close();
  }

  double inletPressureArray[numberOfTimesteps];

// readin of inlet pressure timesteps from file into inletPressureArray
  ifstream infile1(filename1.c_str());
  int t=0;
  string line1;
  stringstream Str1;

  if(infile1.is_open())
  {
    while(!infile1.eof())
    {
      {
        getline(infile1,line1);
        if (line1.empty()) //make sure empty lines are not read into array.
        {
//          cout << "Empty line." << endl;
        }
        else
        {
          Str1 << line1;
          Str1 >> inletPressureArray[t];
          Str1.clear(); // Stringstream needs to be cleared in order to be written
              correctly in next step.
//          cout << "inletPressureArray" << s << ": " << inletPressureArray[t] <<
    endl;
          t++;
      }
    }
  }
  infile1.close();
}
```

Blood density ρ and a termination criterion are defined to decide if periodicity within the RC-circuit is reached.

```
double rho=1060;
cout << " density rho: " << rho << " kg/m^3. " << endl;

// choose tolerance for termination criterion
double tolerance = 1e-12;
cout << "termination criterion for outletFlow is " << tolerance << " m^3/s, 
    corresponding. " << endl;


int maxNumberOfCycles=10;
cout << "maximum number of cycles to compute before automatic termination: " <<
    maxNumberOfCycles << "." << endl;
double capacitorVoltage[N_outlets][numberOfTimesteps];
double initialCapacitorVoltage = 0;
```

An initial capacitor voltage for the computation of the first time step (t = numberOfTimesteps$-$1) needs to be set.

```
cout << "Enter initial capacitor voltage in m^2/s^2: (usually ~0.5)" << endl;
cin >> initialCapacitorVoltage;
for(int i=1; i<=N_outlets; i++)
{
  capacitorVoltage[i][numberOfTimesteps-1]=initialCapacitorVoltage;
  cout << "initial capacitor Voltage @ outlet" << i << ": " << capacitorVoltage[i
      ][numberOfTimesteps-1] << endl;
}
double meanRightAtrialPressure = 0;
cout << "Enter mean right atrial pressure in m^2/s^2: (3.75 mmHg ~ 500 kg/(m*s^2)
    ~ 0.47 m^2/s^2" << endl;
cin >> meanRightAtrialPressure;
cout << "meanRightAtrialPressure: " << meanRightAtrialPressure << " m^2/s^2." <<
    endl;
```

Arrays to hold the flow values in the three branches of the RC-circuit (c.f. fig. 3.1 are defined. The initial flow values at t = numberOfTimesteps $-$ 1 are calculated.

```
double modelCurrent[N_outlets][numberOfTimesteps][2], venousCurrent[N_outlets][
    numberOfTimesteps], capacitorCurrent[N_outlets][numberOfTimesteps]; //required
     for computation of new capacitorVoltage. all currents are positive in
     direction of the mutual node of the three branches. Like this all three
     currents always add to zero!

for(int i=1; i<=N_outlets; i++)
{
modelCurrent[i][numberOfTimesteps -1][0] = rho*(inletPressureArray[
    numberOfTimesteps-1] - (myocardialPressureArray[numberOfTimesteps-1] +
    capacitorVoltage[i][numberOfTimesteps -1]))/vesselResistancesArray[i];
venousCurrent[i][numberOfTimesteps-1] = rho*(meanRightAtrialPressure - (
    myocardialPressureArray[numberOfTimesteps-1] + capacitorVoltage[i][
    numberOfTimesteps-1]))/venousResistancesArray[i];
capacitorCurrent[i][numberOfTimesteps-1] = - modelCurrent[i][numberOfTimesteps
    -1][0] - venousCurrent[i][numberOfTimesteps-1];

cout << "initial modelCurrent (volume flow) @ outlet" << i << ": " <<
    modelCurrent[i][numberOfTimesteps -1][0] << " m^3/s." << endl;
cout << "initial venousCurrent (volume flow) @ outlet" << i << ": " <<
    venousCurrent[i][numberOfTimesteps-1] << " m^3/s." << endl;
cout << "initial capacitorCurrent (volume flow) @ outlet" << i << ": " <<
    capacitorCurrent[i][numberOfTimesteps-1] << " m^3/s." << endl;
}

for(int z=0;z < numberOfTimesteps; z++)
```

```
{
  cout << "inletPressure␣timesteps␣" << z << ":␣" << inletPressureArray[z] <<
      endl;
  cout << "MyocardialPressure␣timesteps␣" << z << ":␣" << myocardialPressureArray
      [z] << endl;
}
```

Output streams to store the results are opened. The array `totalFlow` is defined, which adds the flow values at all outlets for each timestep. This quantity can later be used as inlet boundary condition for CFD-simulations.

```
ofstream output;
ofstream output2;
output.open ("control.txt");
double testMeanTotalFlux = 0;
double totalFlow[numberOfTimesteps];
for(int k=0; k<numberOfTimesteps; k++)
{
  totalFlow[k] = 0;
}
for(int L=1; L<=N_outlets; L++)
{
```

For each model outlet an array is defined, which is filled with the values from the previously defined myocardial pressure array, scaled depending on the outlet's position within the myocardium (using the assigned value stored in the `ventricularPressureScalingArray`, previously computed with *computeBarycenter.cpp*).

```
  if((ventricularPressureScalingArray[L]-0) <= 0.143 /*equals ~1/7*/)
  {
  cout << "ventricularPressureScaling␣for␣outlet" << L << ":␣is␣between␣" << 0 <<
      "␣&␣" << 0.143 << ",␣scale␣ventricular␣Pressure␣by␣1.25" << endl;
    for(size_t y=0; y<numberOfTimesteps; y++)
    {
      tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 1.25;

    }
  }
  else if((ventricularPressureScalingArray[L]-1*0.143) <= (0.143))
  {
    cout << "ventricularPressureScaling␣for␣outlet" << L << ":␣is␣between␣" <<
        1*0.143 << "␣&␣" << 2*0.143 << ",␣scale␣ventricular␣Pressure␣by␣1.15" <<
        endl;
    for(size_t y=0; y<numberOfTimesteps; y++)
    {
      tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 1.15;

    }
  }
  else if((ventricularPressureScalingArray[L]-2*0.143) <=  (0.143))
  {
    cout << "ventricularPressureScaling␣for␣outlet" << L << ":␣is␣between␣" <<
        2*0.143 << "␣&␣" << 3*0.143 << ",␣scale␣ventricular␣Pressure␣by␣1.05" <<
        endl;
    for(size_t y=0; y<numberOfTimesteps; y++)
    {
      tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 1.05;

    }
  }
  else if((ventricularPressureScalingArray[L]-3*0.143) <= (0.143))
  {
    cout << "ventricularPressureScaling␣for␣outlet" << L << ":␣is␣between␣" <<
        3*0.143 << "␣&␣" << 4*0.143 << ",␣scale␣ventricular␣Pressure␣by␣0.95" <<
        endl;
    for(size_t y=0; y<numberOfTimesteps; y++)
    {
      tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 0.95;

    }
  }
```

```
    else if((ventricularPressureScalingArray[L]-4*0.143) <= (0.143))
    {
      cout << "ventricularPressureScaling for outlet" << L << ": is between " <<
          4*0.143 << " & " << 5*0.143 << ", scale ventricular Pressure by 0.85" <<
          endl;
      for(size_t y=0; y<numberOfTimesteps; y++)
      {
        tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 0.85;

      }
    }
    else if((ventricularPressureScalingArray[L]-5*0.143) <= (0.143))
    {
      cout << "ventricularPressureScaling for outlet" << L << ": is between " <<
          5*0.143 << " & " << 6*0.143 << ", scale ventricular Pressure by 0.75" <<
          endl;
      for(size_t y=0; y<numberOfTimesteps; y++)
      {
        tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 0.75;

      }
    }
    else if((ventricularPressureScalingArray[L]-6*0.143) <= (0.143))
    {
      cout << "ventricularPressureScaling for outlet" << L << ": is between " <<
          6*0.143 << " & " << 7*0.143 << "~=" << 1 << ", scale ventricular Pressure 
          by 0.65" << endl;
      for(size_t y=0; y<numberOfTimesteps; y++)
      {
        tmpMyocardialPressureArray[y] = myocardialPressureArray[y] * 0.65;

      }
    }
```

As long as the boolean `isBreak` is set to `false`, the following loops contain calculations of flow values and capacitor voltages for all timesteps until `cycleCounter` reaches a maximum number. Starting at cycle 1, at the end of each cycle, the results are checked for periodicity with the tolarance set at the beginning.

```
  bool isBreak = false;
  for(int cycleCounter=0; cycleCounter < maxNumberOfCycles; cycleCounter++)
  {
    double meanOutletFlow = 0;
    for(int timeCounter=0;timeCounter < numberOfTimesteps; timeCounter++)
    {
    if(cycleCounter > 1 && timeCounter == numberOfTimesteps-1)
    {
            output << " outlet" << L << "timeCounter is " << timeCounter << ", the 
                end of for-loop is reached. checking for periodicity." << endl; //
                remainder << endl;
      double deviation[numberOfTimesteps];
      double maxDeviation=0;
      int timestepOfMaxDeviation=0;
      for(int r=0; r<numberOfTimesteps; r++)
      {
```

Control values for all calculation results are stored to `output`, which redirects to the file *control.txt*.

```
        deviation[r] = fabs(modelCurrent[L][r][cycleCounter%2] - modelCurrent[L][r
            ][(cycleCounter-1)%2]);
        output << " modelCurrent cycle " << cycleCounter << " timeCounter " << r <<
             ": " << modelCurrent[L][r][cycleCounter%2] << endl;
        output << " modelCurrent cycle " << cycleCounter-1 << " timeCounter " << r
            << ": " << modelCurrent[L][r][(cycleCounter-1)%2] << endl;
          if(deviation[r] > maxDeviation)
          {
            maxDeviation = deviation[r];
            timestepOfMaxDeviation = r;
            output << " maxDeviation " << maxDeviation << " newly set at timestep "
                 << r << endl;
          }
        output << " difference at cycle " << cycleCounter-1 << " timeCounter " << r
             << ": " << deviation[r] << endl;
        output << " maxDeviation at cycle " << cycleCounter-1 << " timeCounter " <<
              timestepOfMaxDeviation << ": " << maxDeviation << endl;
    }
```

The maximum deviation between subsequent cycles is assessed.

```
    if(maxDeviation < tolerance)
    {
        output << "maxDeviation is smaller than desired tolerance after " <<
            cycleCounter-1 << " cycles. Write resulting modelCurrents in file and 
            leave! " << endl;
        cout << "writing outletFlow" << L << endl;
```

Output streams to store the flow values for the outlets in different formats (direct usage in simulations as well as later data analysis and further computation programs) are opened.

```
    double finalMeanOutletFlow = 0;
    std::stringstream numberedFilenames;
    std::stringstream numberedFilenames2;
    numberedFilenames2 << "outletFlowTimesteps" << L << ".txt";
    numberedFilenames << "outletFlow" << L;
    ofstream output3(numberedFilenames.str().c_str());
    output3 << "(" << endl;
    ofstream output2(numberedFilenames2.str().c_str());

    for(int G=0; G<10; G++)
    {
    for(int k=0; k<numberOfTimesteps; k++)
    {
      if(G==0)
      {
```

The flow values at all timesteps are written to file line per line for all outlets.

```
      output2 <<  modelCurrent[L][k][cycleCounter%2] << endl;
      finalMeanOutletFlow += modelCurrent[L][k][cycleCounter%2];
      totalFlow[k] += modelCurrent[L][k][cycleCounter%2];
      output << "totalFlow into outlets 1 to " << L << "@ timestep: " << k << ": "
          << totalFlow[k] << " m^3/s." << endl;
      output << "modelCurrent outlet" << L << "@ timestep: " << k << ": " <<
          modelCurrent[L][k][cycleCounter%2] << " m^3/s." << endl;
      }
      output3 << "(" << (static_cast<double> (G)*numberOfTimesteps + k)*
          timeStepSize << " " << -modelCurrent[L][k][cycleCounter%2] << ")" << endl;
    }
    }
```

The mean outlet flow value at each outlet is printed to screen.

```
    finalMeanOutletFlow = finalMeanOutletFlow/numberOfTimesteps;
    testMeanTotalFlux += finalMeanOutletFlow;
    cout << "finalMeanOutletFlow at outlet" << L << " at cycle " << cycleCounter-1
        << ": " << finalMeanOutletFlow << " m^3/s" << endl;

    output3 << ")" << endl;
    output2.close();
    output3.close();
    meanOutletFlow += modelCurrent[L][timeCounter][cycleCounter%2];
```

If periodicity is reached, the boolean `isBreak` is set to `true` and the computation loop stops. Otherwise, the calculations continue for another full cycle and the periodicity criterion is checked again.

```
        isBreak = true;
        cout << "termination criterion fulfilled after cycle: " << cycleCounter <<
            ". " << endl;
    }
    else
    {
        output << " maxDeviation = " << maxDeviation << " is larger than desired 
            tolerance " << tolerance << " at timestep " << timestepOfMaxDeviation <<
             " within cycle, go on! " << endl;
        }
    }
```

The control values are written to the stream `output` (i.e. *control.txt*). Afterwards, the calculations of the model, venous and capacitor currents as well as the capacitor voltage (by calling the initially defined function `changingCapacitorVoltage`) are performed for the first timestep (`timeCounter == 0`).

```
        output << "intraMyocardialPressureArray value at time " << timeCounter*
            timeStepSize << " is: " << tmpMyocardialPressureArray[timeCounter] << endl
            ;
        if(timeCounter==0)
        {
        modelCurrent[L][timeCounter][cycleCounter%2] = rho*(inletPressureArray[
            timeCounter] - (tmpMyocardialPressureArray[timeCounter] + capacitorVoltage
            [L][numberOfTimesteps-1]))/vesselResistancesArray[L];
        venousCurrent[L][timeCounter] = rho*(meanRightAtrialPressure - (
            tmpMyocardialPressureArray[timeCounter] + capacitorVoltage[L][
            numberOfTimesteps-1]))/venousResistancesArray[L];
        capacitorCurrent[L][timeCounter] = - modelCurrent[L][timeCounter][
            cycleCounter%2] - venousCurrent[L][timeCounter];
        meanOutletFlow += modelCurrent[L][timeCounter][cycleCounter%2];

        output << "modelCurrent @ outlet" << L << ": " << modelCurrent[L][timeCounter
            ][cycleCounter%2]  << " m^3/s" << endl;
        output << "venousCurrent @ outlet" << L << ": " << venousCurrent[L][
            timeCounter] << " m^3/s" << endl;
        output << "capacitorCurrent @ outlet" << L << ": " << capacitorCurrent[L][
            timeCounter] << " m^3/s" << endl;
        capacitorVoltage[L][timeCounter] = changingCapacitorVoltage(timeStepSize,
            capacitorVoltage[L][numberOfTimesteps-1],
rho*(inletPressureArray[timeCounter] - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[L][timeCounter-1]))/vesselResistancesArray[L]
+ rho*(meanRightAtrialPressure - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[L][timeCounter-1]))/venousResistancesArray[L],
capacitorCurrent[L][numberOfTimesteps-1],
capacitanceArray[L],
rho);
        }
```

Subsequently, the same computational steps are done to calculate these variables iteratively at each following timestep starting from the previous result.

```
        else
        {
        modelCurrent[L][timeCounter][cycleCounter%2] = rho*(inletPressureArray[
            timeCounter] - (tmpMyocardialPressureArray[timeCounter] + capacitorVoltage
            [L][timeCounter-1]))/vesselResistancesArray[L];
        venousCurrent[L][timeCounter] = rho*(meanRightAtrialPressure - (
            tmpMyocardialPressureArray[timeCounter] + capacitorVoltage[L][timeCounter
            -1]))/venousResistancesArray[L];
        capacitorCurrent[L][timeCounter] = - modelCurrent[L][timeCounter][
            cycleCounter%2] - venousCurrent[L][timeCounter];
        meanOutletFlow += modelCurrent[L][timeCounter][cycleCounter%2];

        output << "modelCurrent @ outlet" << L << ": " << modelCurrent[L][timeCounter
            ][cycleCounter%2]  << " m^3/s" << endl;
        output << "venousCurrent @ outlet" << L << ": " << venousCurrent[L][
            timeCounter] << " m^3/s" << endl;
        output << "capacitorCurrent @ outlet" << L << ": " << capacitorCurrent[L][
            timeCounter] << " m^3/s" << endl;
        capacitorVoltage[L][timeCounter] = changingCapacitorVoltage(timeStepSize,
            capacitorVoltage[L][timeCounter-1],
rho*(inletPressureArray[timeCounter] - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[L][timeCounter-1]))/vesselResistancesArray[L]
+ rho*(meanRightAtrialPressure - (tmpMyocardialPressureArray[timeCounter] +
    capacitorVoltage[L][timeCounter-1]))/venousResistancesArray[L],
capacitorCurrent[L][timeCounter-1],
capacitanceArray[L],
rho);
        }
```

The output precision is set and the control and mean values are stored.

```
  output.precision(10);
  output << "capacitorVoltage @ outlet" << L << "at time " << timeCounter*
      timeStepSize << " is: " << capacitorVoltage[L][timeCounter] << endl;

    meanOutletFlow = meanOutletFlow/numberOfTimesteps;
    output << "meanOutletFlow at outlet" << L << " at cycle " << cycleCounter << ":
        " << meanOutletFlow << " m^3/s" << endl;

    output << "modelCurrent outlet" << L << " timestep " << timeCounter << ": " <<
        modelCurrent[L][timeCounter][cycleCounter%2] << endl;
    }
```

The `for`-loop is left.

```
  if (isBreak == true)
  {
    break;
    cout << "isBreak == " << isBreak << ". Breaking for-loop!" << endl;
  }
  }
  }
```

An additional output stream is defined, in which the total flow across all outlets is stored. The mean total flow is calculated and printed to the screen.

```
  ofstream totalFlowOutput;
  totalFlowOutput.open ("totalFlow.txt");
  for(int k=0; k<numberOfTimesteps; k++)
  {
    totalFlowOutput << totalFlow[k] << endl;
    cout << "totalFlow @ timestep " << k << ": " << totalFlow[k] << " m^3/s." <<
        endl;
  }

  cout << "meanTotalFlux: " << testMeanTotalFlux << "m^3/s." << endl;

  totalFlowOutput.close();
  output.close();
  return 0;
}
```

computeVolume.cpp

The `C++`-program presented in this section is devised to compute the volumes of the vessels leading from the inlet to each outlet. For this purpose, the centerlines are computed beforehand with the `vmtkcenterlines`-program from the software package *VMTK*[4]. From the created *vtp*-files the first 4 entries, which correspond to the *x,y,z*-coordinates as well as the radius of largest sphere still fitting in the vessel at the respective coordinate need to be stored in separate files of the name *centerlinei*.txt, where *i* denotes the outlet number.

Analogous to all other programs presented in this chapter, several libraries and namespace are loaded.

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <stdint.h>
#include <string>
#include <vector>

using namespace std;
```

The function `getNumberOfLines` is defined.

[4]The Vascular Modeling Toolkit, www.vmtk.org.

```
int getNumberOfLines(string file) // function to obtain number of outlets from file
     with listed outlet radii. Name of file is inserted in command line when
    programm is executed.
{
  int number_of_lines=0;
  string line;
  ifstream myfile(file.c_str());

  if(myfile.is_open())
  {
    while(!myfile.eof())
    {
      {
        getline(myfile,line);
        if (line.empty()) // make sure, only non-blank lines are counted!
        {
//            cout << "Empty line." << endl;
        }
        else
        {
//            cout << line << endl;
          number_of_lines++;
        }
      }
    }
    myfile.close();
  }

  cout << "Number of lines in " << file << " is " << number_of_lines << endl;

  return number_of_lines;
}
```

A function to retrieve the three coordinates of the points along the centerline is created.

```
double getCoordinateEntries(string file, int numberOfEntries, double arrayX[],
    double arrayY[], double arrayZ[], double arrayRadius[])

{
  string line;
  ifstream infile(file.c_str());
  if(infile.fail())
  {
    cout << "Error: File " << file.c_str() << " could not be opened." << endl;
    return 1;
  }
  stringstream fileName;
  for(int i=0; i<numberOfEntries; i++)
  {
//    getline(infile, line);
    infile >> arrayX[i];
    infile >> arrayY[i];
    infile >> arrayZ[i];
    infile >> arrayRadius[i];


    cout << file.c_str() << " arrayX[" << i << "]: " << arrayX[i] << endl;
    cout << file.c_str() << " arrayY[" << i << "]: " << arrayY[i] << endl;
    cout << file.c_str() << " arrayZ[" << i << "]: " << arrayZ[i] << endl;
    cout << file.c_str() << " arrayRadius[" << i << "]: " << arrayRadius[i] << endl
        ;

  }
  infile.close();

  return 0;
}
```

The main function starts. The scaling factor and the number of model outlets need to be set by user input.

```
int main()
{
  int N_outlets = 0;
```

```
double scalingFactor = 0;

cout << "What␣is␣the␣number␣of␣outletPatches?" << endl;
cin >> N_outlets;

cout << "What␣is␣the␣scalingFactor␣of␣the␣model?" << endl;
cin >> scalingFactor;

stringstream tmpFileName;
string fileName;
```

Two arrays `distances` and `volumes` are generated. A string stream is defined, to hold the coordinates and the maximum sphere radii of the centerline points to be read in.

```
int numberOfEntries[N_outlets];
double distances[N_outlets];
double volumes[N_outlets];

for(int j=0; j<N_outlets; j++)
{
  double tmpOutletCenterpoint[3];
  tmpFileName << "centerline" << j+1 << ".txt";
  fileName = tmpFileName.str();
```

The previously defined function `getNumberOfLines` is called to count the number of entries in each *centerline*file. Arrays to store *x,y,z*-coordinates and the radius are defined.

```
  numberOfEntries[j+1] = getNumberOfLines(fileName);
  double xCoordinate[numberOfEntries[j+1]];
  double yCoordinate[numberOfEntries[j+1]];
  double zCoordinate[numberOfEntries[j+1]];
  double radiusCoordinate[numberOfEntries[j+1]];
```

Similarly the function `getCoordinateEntries` is called, which fills the generated arrays with the 4 values.

```
  getCoordinateEntries(fileName, numberOfEntries[j+1], xCoordinate, yCoordinate,
      zCoordinate, radiusCoordinate);
  tmpFileName.str("");
  double squareRootSum=0;
  volumes[j]=0;
```

A `for`-loop to compute the sum of the Euclidian distances between subsequent points on the centerline is started. Additionally, the volume of the cylinder associated to this distance with radius as stored in `radiusCoordinate` is calculated. This serves as an approximation of the volume of this vessel segment. At the end of the calculations, all values in the arrays `distances` and `volumes` are scaled with the initially set scaling factor.

```
  for(int i=1; i<numberOfEntries[j+1]; i++)
  {
    double partialCylinderLength=0;
    squareRootSum += sqrt(pow(xCoordinate[i]-xCoordinate[i-1],2)+pow(yCoordinate[
        i]-yCoordinate[i-1],2)+pow(zCoordinate[i]-zCoordinate[i-1],2));
    partialCylinderLength = sqrt(pow(xCoordinate[i]-xCoordinate[i-1],2)+pow(
        yCoordinate[i]-yCoordinate[i-1],2)+pow(zCoordinate[i]-zCoordinate[i-1],2))
        ;
    cout << "xCoordinate␣is:␣" << xCoordinate[i] << endl;
    cout << "yCoordinate␣is:␣" << yCoordinate[i] << endl;
    cout << "zCoordinate␣is:␣" << zCoordinate[i] << endl;
    cout << "radiusCoordinate␣is:␣" << radiusCoordinate[i] << endl;
    cout << "partialCylinderLengths␣is:␣" << partialCylinderLength << endl;
    volumes[j]+=pow(scalingFactor,3)*partialCylinderLength*M_PI*pow(
        radiusCoordinate[i-1],2);
    cout << "volume␣at␣entry␣" << i << "␣is:␣" << volumes[j] << endl;
  }
  distances[j]=scalingFactor*squareRootSum;
  cout << "distance␣to␣outlet" << j+1 << "␣is:␣" << distances[j] << "." << endl;
  cout << "volume␣between␣inlet␣and␣outlet" << j+1 << "␣is:␣" << volumes[j] << ".
      " << endl;
}
```

Two output streams to store the obtained results are opened. In the resulting files, volumes are stored in m^3 and distances in mm.

```
  ofstream output("volumes.txt");
  ofstream output2("distances.txt");
  for(int n=0; n<N_outlets; n++)
  {
    cout << "Distance Inlet to outlet" << n+1 << " is: " << 1000*distances[n] << " 
        mm. " << endl;
    cout << "Volume between Inlet and outlet" << n+1 << " is: " << volumes[n] << " 
        m^3 or " << 1000000*volumes[n] << " ml respectively " <<endl;
    output.precision(10);
    output << volumes[n] << endl;
    output2.precision(10);
    output2 << 1000*distances[n] << endl;

  }

  return 0;
}
```

computeBarycenter.cpp

The C++-program *computeBarycenter.cpp* serves to calculate the barycenter of all outlets of a 3D vascular geometry[5]. It gives an approximation of the center of the ventricle, which is supplied by the considered model outlets. The distance of the outlets to the barycenter is then used in the following computation of both capillary pressure (cf Section A.2) and outlet flow (cf Section A.2). In the following, the main steps of the program are described.

At first required libraries and namespace are loaded:

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <stdint.h>
#include <string>
#include <vector>

using namespace std;
```

Next a function to count the number of lines in a file is defined:

```

int getNumberOfLines(string file) // function to obtain number of outlets from file
     with listed outlet radii. Name of file is inserted in command line when
    programm is executed.
{
  int number_of_lines=0;
  string line;
  ifstream myfile(file.c_str());

  if(myfile.is_open())
  {
    while(!myfile.eof())
    {
      {
        getline(myfile,line);
        if (line.empty()) // make sure, only non-blank lines are counted!
        {
//          cout << "Empty line." << endl;
        }
        else
        {
          number_of_lines++;
        }
      }
    }
    myfile.close();
  }
```

[5]Usage of this program only makes sense if the outlets of the model encompass the whole ventricle.

```
  cout << "Number_of_lines_in_" << file << "_is_" << number_of_lines << endl;

  return number_of_lines;
}
```

Then the function `getCenterOfInletPatch` is defined, which calculates the center of the inlet patch from coordinates read in from the file `INLET.TXT`

```
double getCenterOfInletPatch(int numberOfEntries, double centerpoint[])
{
  double result[3];
  double sumX=0, sumY=0, sumZ=0;
  string fileName;
  string line;
  fileName = "INLET.txt";
  ifstream infile(fileName.c_str());
  if(infile.fail())
  {
    cout << "Error:_File_INLET.txt_could_not_be_opened." << endl;
    return 1;
  }
  double arrayX[numberOfEntries];
  double arrayY[numberOfEntries];
  double arrayZ[numberOfEntries];
```

The file must be a table, in which the columns represent the x, y, z-coordinates of the respective points on the inlet patch. The coordinates are summed and the file is closed.

```
  for(int i=0; i<numberOfEntries; i++)
  {
    infile >> arrayX[i];
    infile >> arrayY[i];
    infile >> arrayZ[i];

    sumX+= arrayX[i];
    sumY+= arrayY[i];
    sumZ+= arrayZ[i];
  }
  infile.close();
```

Subsequently, the arithmetic means of the x, y, z-coordinates are computed and defined as the centerpoint of the inlet patch.

```

  result[0]=sumX/numberOfEntries;
  result[1]=sumY/numberOfEntries;
  result[2]=sumZ/numberOfEntries;

  for(int t=0; t<3; t++)
  {
    centerpoint[t]=result[t];
  }

  return 0;
}
```

The following function `getCenterOfOutletPatch` works analogously, but loops over all outlets.

```

double getCenterOfOutletPatch(string file, int numberOfEntries, double centerpoint
    [])
{
  double result[3];
  double sumX=0, sumY=0, sumZ=0;
  string line;
  ifstream infile(file.c_str());
  if(infile.fail())
  {
    cout << "Error:_File_" << file.c_str() << "_could_not_be_opened." << endl;
    return 1;
  }
```

```
    double arrayX[numberOfEntries];
    double arrayY[numberOfEntries];
    double arrayZ[numberOfEntries];
    stringstream fileName;
    for(int i=0; i<numberOfEntries; i++)
    {
        infile >> arrayX[i];
        infile >> arrayY[i];
        infile >> arrayZ[i];

       sumX+= arrayX[i];
       sumY+= arrayY[i];
       sumZ+= arrayZ[i];

    }
    infile.close();

    result[0]=sumX/numberOfEntries;
    result[1]=sumY/numberOfEntries;
    result[2]=sumZ/numberOfEntries;

    for(int t=0; t<3; t++)
    {
       centerpoint[t]=result[t];
    }

    return 0;
}
```

The main function is called to calculate the barycenter of all outlets. At the beginning the variables `N_outlets` and `scalingFactor` are initialized and then set by user intervention.

```
int main()
{
    int N_outlets = 0;
    double scalingFactor;

    cout << "What is the number of outletPatches?" << endl;
    cin >> N_outlets;

    cout << "What is the scalingFactor of the model?" << endl;
    cin >> scalingFactor;
```

After that `inletCenterpoint[3]` and `outletCenterpoint[N_outlets][3]` are initialized and set by calling the previously defined functions `getCenterOfInletPatch` and `getCenterOfOutletPatch`.

```
    double inletCenterpoint[3];
    double outletCenterpoint[N_outlets][3];

    int numberOfVertices[N_outlets];

    stringstream tmpFileName;
    string fileName;

    for(int j=0; j<N_outlets; j++)
    {
       double tmpOutletCenterpoint[3];
       tmpFileName << "OUTLET" << j+1 << ".txt";
       fileName = tmpFileName.str();
       numberOfVertices[j+1] = getNumberOfLines(fileName);
       getCenterOfOutletPatch(fileName, numberOfVertices[j+1], tmpOutletCenterpoint);
       tmpFileName.str("");
       for(int z=0; z<3; z++)
       {
          outletCenterpoint[j][z]=tmpOutletCenterpoint[z];
       }
    }
```

In the next step, the arithmetic mean of all x, y, z-coordinates is computed and the resulting value is defined as the model's barycenter.

```
  double xSum=0, ySum=0, zSum=0;
  double baryCenterCoordinate[3];
  for(int n=0; n<N_outlets; n++)
  {
    xSum += outletCenterpoint[n][0];
    ySum += outletCenterpoint[n][1];
    zSum += outletCenterpoint[n][2];
  }
  baryCenterCoordinate[0]=xSum/N_outlets;
  baryCenterCoordinate[1]=ySum/N_outlets;
  baryCenterCoordinate[2]=zSum/N_outlets;

  cout << "baryCenter␣is␣at:␣" << baryCenterCoordinate[0] << ",␣" <<
      baryCenterCoordinate[1] << ",␣" << baryCenterCoordinate[2] << ",␣or␣scaled:␣"
      << baryCenterCoordinate[0]*scalingFactor << ",␣" << baryCenterCoordinate[1]*
      scalingFactor << ",␣" << baryCenterCoordinate[2]*scalingFactor << endl;
```

The distances between the outlet centerpoints and the barycenter are then calculated and the variables `maxDistance` and `minDistance` are defined.

```
  double distances[N_outlets];

  for(int n=0; n<N_outlets; n++)
  {
    double squareSum=0;
    double result=0;
    for(int m=0; m<3; m++)
    {
      squareSum += pow(baryCenterCoordinate[m]-outletCenterpoint[n][m],2);
    }
    result = sqrt(squareSum)*scalingFactor;
    distances[n] = result;
    cout << "Distance␣baryCenter␣to␣outlet" << n+1 << "␣is:␣" << 1000*distances[n]
        << "␣mm.␣" << endl;
  }

  double maxDistance=0, minDistance=1000;
  for(int n=0; n<N_outlets; n++)
  {
    if(maxDistance < distances[n])
    {
      maxDistance = distances[n];
    }
    if(minDistance > distances[n])
    {
      minDistance = distances[n];
    }
    cout << "maxDistance:␣" << maxDistance << "␣m␣,␣minDistance:␣" << minDistance
        << "␣m." << endl;
  }

  double difference=0;
  difference = maxDistance - minDistance;
  cout << "difference:␣" << difference << "␣m." << endl;
```

Scaling factors for all outlets are defined and written to disk.

```
  ofstream output("ventricularPressureScaling.txt");

  double scalingFactors[N_outlets];
  for(int n=0; n<N_outlets; n++)
  {
    scalingFactors[n] = (distances[n]-minDistance)/difference;
    cout << "scalingFactor␣@outlet" << n+1 << ":␣" << scalingFactors[n] << endl;
    output.precision(10);
    output << scalingFactors[n] << endl;
  }

  return 0;
}
```

A.3 Customized OpenFOAM-solvers

`resistanceDataPimpleFoam`

The following script `resistanceDataPimpleFoam` integrates the condition on the outlet pressure as derived in Section 3.2, Eq. 3.3 into a dedicated OpenFOAM-solver. It is based on OpenFOAM's native solver `pimpleFoam`. Starting point of this solver is the solver `pisoResistanceFoam`, which was developed for OpenFOAM-2.2.2 and kindly provided by Karsten Sommer. It is described in detail in his PhD-thesis [108]. In the following the main steps are described and substantial changes in comparison to the original solvers are pointed out.Since in this work, the more recent version OpenFOAM-2.3.1 is used, slight changes of the wording and used classes were made, which are not pointed out explicitely.

In the beginning of the file, the additional header file `DataEntry.H` is loaded, which is later used to import supplementary input parameters.

```
#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "turbulenceModel.H"
#include "pimpleControl.H"
#include "fvIOoptionList.H"
#include "IOporosityModelList.H"
#include "IOMRFZoneList.H"
#include "fixedFluxPressureFvPatchScalarField.H"
#include "DataEntry.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

The header file `sstream` is required for transformation between strings and numbers.

```


int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"
    #include "createFvOptions.H"
    #include "initContinuityErrs.H"
    #include "readTimeControls.H"
    #include <sstream>

    pimpleControl pimple(mesh);
```

Several input variables stored in the input file *controlDict* are read with the help of `IOdictionary` class type objects. One of which are the outlet resistances which must be computed beforehand with the `C++`-script described in Section A.2.

```
    // * * * * Read resistance, initial flux data & pulse properties * * * * * * *
        * * * * * * //

    Info<< "Reading pulse properties ..." << nl << endl;

    IOdictionary controlDictCopy
    (   IOobject   ("controlDict",runTime.system(),mesh,IOobject::MUST_READ,IOobject
        ::NO_WRITE) );

    autoPtr<DataEntry<scalar> > nOutletsValue
    (   DataEntry<scalar>::New ("nOutlets", controlDictCopy) );
    int nOutlets = nOutletsValue().value(0);

    autoPtr<DataEntry<scalar> > startWriteTime
    (   DataEntry<scalar>::New   ("startWriteTime", controlDictCopy)   );
    scalar startWritingAt = startWriteTime().value(0);

    autoPtr<DataEntry<scalar> > pulseDuration
```

```
  ( DataEntry<scalar>::New ("pulseDuration", controlDictCopy) );

      autoPtr<DataEntry<scalar> > timeStep
  ( DataEntry<scalar>::New ("deltaT", controlDictCopy) );

scalar pulseT = pulseDuration().value(0);
scalar dt = timeStep().value(0);
Info << "pulse duration: " << pulseT << nl;
Info << "time step: " << dt << nl;
```

The number of timesteps per cardiac cycle is computed.

```
//calculate nTimeStepsInPulse:
double ratio = pulseT / dt;
double ratioRounded = ratio * 10000.0;
int tmpInt = ratioRounded + 0.5;
float tmpFloat = tmpInt;
ratioRounded = tmpFloat / 10000.0;
int ratioInt = ratioRounded;
int nTimeStepsInPulse = ratioInt;

Info << "nTimeStepsInPulse: " << nTimeStepsInPulse << nl;

  std::ostringstream convertNum2Str;
  string resistanceString = "resistance";

  scalar resistanceArray[nOutlets];
  scalar initialFluxArray[nOutlets];
  scalar pressureScalingArray[nOutlets];
  int leftOrRightVentricleArray[nOutlets];
```

Dedicated arrays are initialized to store the assigned outlet resistances and initial fluxes (required for computations in the very first time step) and scaling of the capillary pressure (cf. A.2).

```
  for (int i=0; i<nOutlets; i++)
  {
    std::ostringstream convertNum2Str;
    // read resistance values:
    convertNum2Str << i+1;
    string tmpString = "resistance"+convertNum2Str.str();
    autoPtr<DataEntry<scalar> > tmpResistanceValue
    ( DataEntry<scalar>::New (tmpString, controlDictCopy) );
    resistanceArray[i] = tmpResistanceValue().value(0);

    Info << "resistance @ outlet" + convertNum2Str.str() + ": " <<
        resistanceArray[i] << nl;

    // read initial flux values:
    tmpString = "initialFlux"+convertNum2Str.str();
    autoPtr<DataEntry<scalar> > tmpInitialFluxValue
    ( DataEntry<scalar>::New (tmpString, controlDictCopy) );
    initialFluxArray[i] = tmpInitialFluxValue().value(0);
```

In comparison to Karsten Sommer's solver `pisoResistanceFoam` [108], the following array that stores the scaling of the capillary pressure (cf. A.2) represents the beginning of a series of fundamental changes.

```
    // read capillaryPressureScaling:
    tmpString = "capillaryPressureScaling"+convertNum2Str.str();
    autoPtr<DataEntry<scalar> > tmpPressureScalingValue
    ( DataEntry<scalar>::New (tmpString, controlDictCopy) );
    pressureScalingArray[i] = tmpPressureScalingValue().value(0);

    Info << "capillary Pressure Saling @ outlet" + convertNum2Str.str() + ": " <<
          pressureScalingArray[i] << nl;
```

An array is constructed, which stores boolean values to indicate if the considered outlet supplies the left or the right ventricle. These values (and all others required for the previously defined arrays) must be stored in the case's *controlDict*-file.

```
      // check if vessel supplies left or right ventricle: 1 -> left , 0 -> right
      tmpString = "leftOrRightVentricle"+convertNum2Str.str();
      autoPtr<DataEntry<scalar> > tmpLeftOrRightVentricleValue
      (   DataEntry<scalar>::New (tmpString, controlDictCopy) );
      leftOrRightVentricleArray[i] = tmpLeftOrRightVentricleValue().value(0);

      if(leftOrRightVentricleArray[i] == 1)
      {
       Info << "outlet" + convertNum2Str.str() + "␣supplies␣left␣ventricle␣(boolean
           :␣" << leftOrRightVentricleArray[i] << ").␣" << nl;
      }
      else if(leftOrRightVentricleArray[i] == 0)
      {
       Info << "outlet" + convertNum2Str.str() + "␣supplies␣right␣ventricle␣(
           boolean:␣" << leftOrRightVentricleArray[i] << ").␣" << nl;
      }
    }
```

For each model outlet an array is defined, which is filled with the pressure values read from the capillary pressure files, which are previously computed and created with the help of the script `capillaryPressure.cpp` (cf. Section A.2). Depending on the value in the boolean array `leftOrRightVentricleArray` and the array with the scaling values `capillaryPressureScaling`, the pressure values for one full cardiac cycle are assigned to each `capillaryPressureArray`. This procedure effectuates consideration of the position of model outlets in endo- or epicardium.

```
    // capillary Pressure data vectors:
  Info << "Reading␣capillaryPressures" << endl;
  scalar capillaryPressureArray[nOutlets][nTimeStepsInPulse];
 for (int L=0; L<nOutlets; L++)
 {
if(leftOrRightVentricleArray[L] == 1)
{
  scalar tCounter = 0;
   if((pressureScalingArray[L]-0) <= 0.143 /*equals ~1/7*/)
   {
    Info << "capillaryPressureScaling␣for␣outlet" << L+1 << ":␣is␣between␣" << 0 <<
        "␣&␣" << 0.143 << ",␣use␣LVCapillaryPressure125Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject    ( "LVCapillaryPressure125Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure125",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
    }
   }
   else if((pressureScalingArray[L]-1*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling␣for␣outlet" << L+1 << ":␣is␣between␣" <<
        1*0.143 << "␣&␣" << 2*0.143 << ",␣use␣LVCapillaryPressure115Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject    ( "LVCapillaryPressure115Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure115",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//     Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
    }
    else if((pressureScalingArray[L]-2*0.143) <=  (0.143))
    {
     Info << "capillaryPressureScaling for outlet" << L+1 << ": is between " <<
         2*0.143 << " & " << 3*0.143 << ", use LVCapillaryPressure105Dict" << endl;

     IOdictionary capillaryPressuresDict
     ( IOobject    ( "LVCapillaryPressure105Dict",runTime.system(),mesh,IOobject::
         MUST_READ,IOobject::NO_WRITE )   );
     autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar >::New("
         capillaryPressure105",capillaryPressuresDict) );

     for (int i=0; i<nTimeStepsInPulse; i++)
     {
      tCounter = i*dt;
      capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
     }
    }
    else if((pressureScalingArray[L]-3*0.143) <= (0.143))
    {
     Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
         3*0.143 << " & " << 4*0.143 << ", use LVCapillaryPressure95Dict" << endl;

     IOdictionary capillaryPressuresDict
     ( IOobject    ( "LVCapillaryPressure95Dict",runTime.system(),mesh,IOobject::
         MUST_READ,IOobject::NO_WRITE )   );
     autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar >::New("
         capillaryPressure95",capillaryPressuresDict) );

     for (int i=0; i<nTimeStepsInPulse; i++)
     {
      tCounter = i*dt;
      capillaryPressureArray[L][i] = dataEntry().value(tCounter);
 //     Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
     << " is: " << capillaryPressureArray[L][i] << endl;
     }
    }
    else if((pressureScalingArray[L]-4*0.143) <= (0.143))
    {
     Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
         4*0.143 << " & " << 5*0.143 << ", use LVCapillaryPressure85Dict" << endl;

     IOdictionary capillaryPressuresDict
     ( IOobject    ( "LVCapillaryPressure85Dict",runTime.system(),mesh,IOobject::
         MUST_READ,IOobject::NO_WRITE )   );
     autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar >::New("
         capillaryPressure85",capillaryPressuresDict) );

     for (int i=0; i<nTimeStepsInPulse; i++)
     {
      tCounter = i*dt;
      capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
     }
    }
    else if((pressureScalingArray[L]-5*0.143) <= (0.143))
    {
     Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
         5*0.143 << " & " << 6*0.143 << ", use LVCapillaryPressure75Dict" << endl;

     IOdictionary capillaryPressuresDict
     ( IOobject    ( "LVCapillaryPressure75Dict",runTime.system(),mesh,IOobject::
         MUST_READ,IOobject::NO_WRITE )   );
     autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar >::New("
         capillaryPressure75",capillaryPressuresDict) );

     for (int i=0; i<nTimeStepsInPulse; i++)
     {
      tCounter = i*dt;
      capillaryPressureArray[L][i] = dataEntry().value(tCounter);
 //     Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
     << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-6*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
        6*0.143 << " & " << 7*0.143 << "~=" << 1 << ", use LVCapillaryPressure65Dict
        " << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject   ( "LVCapillaryPressure65Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure65",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
}

else if(leftOrRightVentricleArray[L] == 0)
{
//Info << "Using RVCapillaryPressureArray for outlet" << L+1 << endl;
  scalar tCounter = 0;
   if((pressureScalingArray[L]-0) <= 0.143 /*equals ~1/7*/)
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between " << 0 <<
         " & " << 0.143 << ", use RVCapillaryPressure125Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject   ( "RVCapillaryPressure125Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure125",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-1*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between " <<
        1*0.143 << " & " << 2*0.143 << ", use RVCapillaryPressure115Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject   ( "RVCapillaryPressure115Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure115",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-2*0.143) <=  (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between " <<
        2*0.143 << " & " << 3*0.143 << ", use RVCapillaryPressure105Dict" << endl;
    IOdictionary capillaryPressuresDict
    ( IOobject  ( "RVCapillaryPressure105Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )  );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure105",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//     Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-3*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
        3*0.143 << " & " << 4*0.143 << ", use RVCapillaryPressure95Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject  ( "RVCapillaryPressure95Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )  );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure95",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
 //    Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-4*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
        4*0.143 << " & " << 5*0.143 << ", use RVCapillaryPressure85Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject  ( "RVCapillaryPressure85Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )  );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure85",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//     Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-5*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling for outlet" << L+1 << ": is between "<<
        5*0.143 << " & " << 6*0.143 << ", use RVCapillaryPressure75Dict" << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject  ( "RVCapillaryPressure75Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )  );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure75",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
 //    Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
   else if((pressureScalingArray[L]-6*0.143) <= (0.143))
   {
    Info << "capillaryPressureScaling␣for␣outlet" << L+1 << ":␣is␣between␣"<<
        6*0.143 << "␣&␣" << 7*0.143 << "~=" << 1 << ",␣use␣RVCapillaryPressure65Dict
        " << endl;

    IOdictionary capillaryPressuresDict
    ( IOobject   ( "RVCapillaryPressure65Dict",runTime.system(),mesh,IOobject::
        MUST_READ,IOobject::NO_WRITE )   );
    autoPtr<DataEntry<scalar> > dataEntry (DataEntry<scalar>::New("
        capillaryPressure65",capillaryPressuresDict) );

    for (int i=0; i<nTimeStepsInPulse; i++)
    {
     tCounter = i*dt;
     capillaryPressureArray[L][i] = dataEntry().value(tCounter);
//      Info << "capillaryPressure @outlet" << L+1 << " for timestep " << tCounter
    << " is: " << capillaryPressureArray[L][i] << endl;
    }
   }
}
 }
```

After these definitions and calculations, the following code is again left unaltered from [108] except for partial modification due to the transfer between the software versions. An array to store the current flux values at all outlets is initialized. Further variables required for the computation of pressures and fluxes at the outlets and the inlet are defined.

```
scalar outFluxArray[nOutlets];

  scalar inletPressureVector =0;
  scalar outletPressureVector[nOutlets];

  scalar localInflux;
  scalar totalInfluxSum;
  scalar totalOutflux;

  scalar totalInletPressureSum;
  scalar totalInletAreaSum;

  // * * * * Define and set variables * * * * * * * * * * * * * //

  word inletPatchName = "INLET";
  label inletPatch = mesh.boundaryMesh().findPatchID(inletPatchName);

  word outletPatchNameArray[nOutlets];
  label outletPatchArray[nOutlets];
  for (int i=0; i<nOutlets; i++)
  {
    std::ostringstream convertNum2Str;
    convertNum2Str << i+1;
    outletPatchNameArray[i] = "OUTLET"+convertNum2Str.str();
    outletPatchArray[i] = mesh.boundaryMesh().findPatchID(outletPatchNameArray[i]);
  }

  label thisProcNb = Pstream::myProcNo();
  int nProcessors = Pstream::nProcs();

  int timeCounter = 0;
  scalar deltaTimeCounter = 0;

  scalar rho = 1060;

  scalar newPressureValueArray[nOutlets];
```

A scalar field `phiTemp` is initialized, which will later be used to calculate temporary flow values at the outlets to enable computation and assignment of pressures at the outlets.

```
    surfaceScalarField phiTemp
  ( IOobject ("phiTemp",runTime.timeName(),mesh,IOobject::READ_IF_PRESENT,IOobject
      ::NO_WRITE),
    linearInterpolate(U) & mesh.Sf() );
```

```
scalarField totalInletPressureIntegral(nProcessors,0);
scalarField totalInletArea(nProcessors,0);
scalar localInletPressureIntegral;
scalar localInletArea;

int count = -1;
```

The following lines are left unchanged from the implementation of the PIMPLE algorithm in OpenFOAM's native solver `pimpleFoam`. The PIMPLE-loop to start computing the Navier-Stokes-equations in the geometry as described in Chapter 2 starts.

```
  Info<< "\nStarting time loop\n" << endl;

  while (runTime.run())
  {
      #include "readTimeControls.H"
      #include "CourantNo.H"
#include "setDeltaT.H"

      runTime++;

      Info<< "Time = " << runTime.timeName() << nl << endl;
```

The following lines correspond to the implementation of `pisoResistanceFoam` in [108].

```
  // * * * * check if sub-domain contains part of outlet  * * * * * * * * * * *
      //

  scalar isOutletArray[nOutlets];
  for(int j=0; j<nOutlets;j++)
  {
    isOutletArray[j] = phi.boundaryField()[outletPatchArray[j]].size();    // size
         of boundary field
  }

timeCounter = (static_cast<int> (runTime.value()/dt));//%nTimeStepsInPulse;
```

For the implementation of the outlet BC as defined in Eq. 3.3 the capillary pressure is required – possibly in higher detail than available from the input data file. Therefore, the variable `deltaTimeCounter` is defined and will later be used to interpolate between values of the `capillaryPressureArrays`.

```
if (runTime.value() - static_cast<scalar> (timeCounter)*dt < 1e-8)
{
  deltaTimeCounter = dt;
}
else
{
  deltaTimeCounter = runTime.value() - static_cast<scalar> (timeCounter)*dt;
}
Info << "deltaTimeCounter: " << deltaTimeCounter << nl;

      count++;

  //*************** END OF INSERTION FOR OUTLET PRESSURE CALCULATION AND
      ASSIGNMENT **************
```

The file `UEqn.H` is left unchanged from the original implementation in OpenFOAM's `pimpleFoam`. It is thus not presented in written form here; however, it can be found with other files on the CD that is handed in together with this thesis.

```

      // Pressure-velocity PIMPLE corrector loop
while (pimple.loop())
      {
  #include "UEqn.H"
```

In the next step, `pEqn.H` is called, where the calculation of the outlet pressures based on eq. 3.3 is performed. Please see the following Section A.3. In the remaining part of the pimple-loop the original `pimpleFoam`-solver code is left unchanged.

```
        // --- Pressure Corrector Loop
  while (pimple.correct())
  {
    #include "pEqn.H"
  }


  if (pimple.turbCorr())
  {
        turbulence->correct();
  }
}
```

As in `pisoResistanceFoam` from [108], to account for turbulence, the diffusion coefficient field is calculated based on the chosen turbulence model.

```
// calculate Diffusion coefficient field:

D = Dfactor / turbulence->nu();


  scalar t = runTime.value();
```

The solver code distinguishes between parallel and serial runs. Depending on how the computational grid is decomposed, it must be identified, which parts of the outlets are associated to which processor. Obtained values from the different processors are thus collected, the mean values for flux and pressure on the patches are calculated and the values are redistributed to all processors.

```
if ( Pstream::parRun() )
{

  // get outfluxes and outlet pressures:
  scalar totalOutletArea[nOutlets];
  scalar totalOutletPressureIntegral[nOutlets];
  for(int outletCount=0; outletCount<nOutlets; outletCount++)
  {
    scalarField tmpOutletArea(nProcessors,0);
    tmpOutletArea[thisProcNb] = sum(mesh.magSf().boundaryField()[outletPatchArray
        [outletCount]]);
    reduce( tmpOutletArea, sumOp<scalarField>() );

    scalarField tmpOutletPressureIntegral(nProcessors,0);
    tmpOutletPressureIntegral[thisProcNb] = sum(mesh.magSf().boundaryField()[
        outletPatchArray[outletCount]]
  *p.boundaryField()[outletPatchArray[outletCount]]);
    reduce( tmpOutletPressureIntegral, sumOp<scalarField>() );

    scalarField tmpOutflux(nProcessors,0);
    tmpOutflux[thisProcNb] = sum(phi.boundaryField()[outletPatchArray[outletCount
        ]]);
    reduce( tmpOutflux, sumOp<scalarField>() );

    // sum outfluxes, outlet areas and outlet pressure integrals of all
        processors:
    totalOutletArea[outletCount] = 0;
    totalOutletPressureIntegral[outletCount] = 0;
    outFluxArray[outletCount] = 0;
    for (int co=0; co<nProcessors; co++)
    {
      outFluxArray[outletCount] += tmpOutflux[co];

      totalOutletArea[outletCount] += tmpOutletArea[co];
      totalOutletPressureIntegral[outletCount] += tmpOutletPressureIntegral[co];
    }
```

```
      outletPressureVector[outletCount] = totalOutletPressureIntegral[outletCount]
          / totalOutletArea[outletCount];
  }
}
```

The same calculation is done for simulations on only one processor (serial).

```
else  // serial run
{
  for(int outletCount=0; outletCount<nOutlets; outletCount++)
      {
  outFluxArray[outletCount] = sum(phi.boundaryField()[outletPatchArray[
      outletCount]]);
      }
}
```

A variable is introduced to reduce informative output of the solver. Pressure and flow data are only written, when `runTime` is an integer multiple of 0.01 s.

```

  int T = t * 10000000.0 + 0.5;
  if ( T % 100000 == 0 ) // write outfluxes and inlet pressure only if runtime is
       multiple of 0.01!
  {
```

Pressure and flow at the inlet are calculated from the simulated data. Again, it needs to be distinguished between parallel and serial runs.

```

// Determine influx, outfluxes using phi field (not phiTemp!):
if ( Pstream::parRun() )
{
      // get inlet pressure:
      localInletPressureIntegral = sum(mesh.magSf().boundaryField()[inletPatch]
  *p.boundaryField()[inletPatch]);
      localInletArea = sum(mesh.magSf().boundaryField()[inletPatch]);
      scalarField totalInletPressureIntegral(nProcessors,0);
      scalarField totalInletArea(nProcessors,0);
      totalInletPressureIntegral[thisProcNb] = localInletPressureIntegral;
      totalInletArea[thisProcNb] = localInletArea;
      reduce( totalInletPressureIntegral, sumOp<scalarField>() );
      reduce( totalInletArea, sumOp<scalarField>() );
      // sum inlet pressures of all processors:
      totalInletPressureSum = 0;
      totalInletAreaSum = 0;
      for (int co=0; co<nProcessors; co++)
      {
  totalInletPressureSum += totalInletPressureIntegral[co];
  totalInletAreaSum += totalInletArea[co];
      }

inletPressureVector = totalInletPressureSum / totalInletAreaSum;
      Info << "inlet␣pressure:␣" << inletPressureVector << nl;

      // get Influx:
      localInflux = sum(phi.boundaryField()[inletPatch]);
      scalarField totalInflux(nProcessors,0);
      totalInflux[thisProcNb] = localInflux;
      reduce( totalInflux, sumOp<scalarField>() );
      // sum inlet pressures of all processors:
      totalInfluxSum = 0;
      for (int co=0; co<nProcessors; co++)
      {
  totalInfluxSum += totalInflux[co];
      }

    Info << "Influx:␣" << (-1)*totalInfluxSum << nl;
```

Analogously, pressure and flow at the outlets are computed and all variables are written into the log-file.

```

  totalOutflux = 0;
  for(int outletCount=0; outletCount<nOutlets; outletCount++)
```

```
    {
      Info << "outlet" << outletCount+1 << "␣flux:␣" << outFluxArray[outletCount]
          << nl;
      Info << "outlet" << outletCount+1 << "␣pressure:␣" << outletPressureVector[
          outletCount] << nl;// from boundary field
      Info << "outlet" << outletCount+1 << "␣pressure␣difference:␣" <<
          inletPressureVector - outletPressureVector[outletCount] << nl;
      Info << "outlet" << outletCount+1 << "␣capillaryPressure:␣" <<
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse] +(
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse+1] -
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse]) / dt *
           deltaTimeCounter << nl;
      totalOutflux += outFluxArray[outletCount];
    }

  Info << "total␣Outflux:␣" << totalOutflux << nl;



  }
```

The same is done for serial computation.

```
  else  // serial run
  {
        localInletPressureIntegral = sum(mesh.magSf().boundaryField()[inletPatch]
    *p.boundaryField()[inletPatch]);
        localInletArea = sum(mesh.magSf().boundaryField()[inletPatch]);

inletPressureVector = localInletPressureIntegral / localInletArea;
        Info << "inlet␣pressure:␣" << inletPressureVector << nl;
      Info << "Influx:␣" << (-1)*sum(phi.boundaryField()[inletPatch]) << nl;

  totalOutflux = 0;
    for(int outletCount=0; outletCount<nOutlets; outletCount++)
    {
      outFluxArray[outletCount] = sum(phi.boundaryField()[outletPatchArray[
          outletCount]]);
      Info << "outlet" << outletCount+1 << "␣flux:␣" << outFluxArray[outletCount]
          << nl;
outletPressureVector[outletCount] = (sum(mesh.magSf().boundaryField()[
    outletPatchArray[outletCount]]
    *p.boundaryField()[outletPatchArray[outletCount]])  ) / sum(mesh.magSf().
        boundaryField()[outletPatchArray[outletCount]]);
      Info << "outlet" << outletCount+1 << "␣pressure:␣" << outletPressureVector[
          outletCount] << nl; // from boundary field
      Info << "outlet" << outletCount+1 << "␣pressure␣difference:␣" <<
          inletPressureVector - outletPressureVector[outletCount] << nl;
      Info << "outlet" << outletCount+1 << "␣capillaryPressure:␣" <<
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse]+(
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse+1] -
          capillaryPressureArray[outletCount][timeCounter%nTimeStepsInPulse]) / dt *
           deltaTimeCounter << nl;
      totalOutflux += outFluxArray[outletCount];
    }

  Info << "total␣Outflux:␣" << totalOutflux << nl;

  }
      Info << "timeCounter:␣" << timeCounter << nl;
    }// write outfluxes and inlet pressure only if runtime is multiple of 0.01!
```

If `runTime` is larger than `startWritingAt` initially defined in *controlDict*, the solver starts writing computed physical fields to disk. Additionally, computation times are written to the log-file every computation time step. If the `endTime` defined in *controlDict* is reached, the `while`-pimple-loop is stopped and the simulation ends.

```
    if (runTime.value() >= startWritingAt )
    {
      runTime.write();
    }
```

```
        Info<< "ExecutionTime␣=␣" << runTime.elapsedCpuTime() << "␣s"
            << "␣␣ClockTime␣=␣" << runTime.elapsedClockTime() << "␣s"
            << nl << endl;

    }

    Info<< "End\n" << endl;

    return 0;
}
```

pEqn.H

The following file is called by the main script *resistanceDataPimpleFoam*, which is presented in the Section A.3. Several changes are made to the original *pEqn.H*-file from OpenFOAM's native **pimpleFoam**-solver, which are pointed out in the following.

In the beginning, the solver code remains unchanged. Different fields required to solve the Navier-Stokes equations on the grid are computed. For details about functioning and nomenclature please refer to the original solver codes [6] as well as OpenFOAM's user and programmer's guide [230, 231].

```
surfaceScalarField rAUf("rAUf", fvc::interpolate(rAU));

volVectorField HbyA("HbyA", U);
HbyA = rAU*UEqn().H();

if (pimple.nCorrPISO() <= 1)
{
    UEqn.clear();
}

            surfaceScalarField phiHbyA
            (
                "phiHbyA",
                (fvc::interpolate(HbyA) & mesh.Sf())
              + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
            );

fvOptions.makeRelative(phiHbyA);

            adjustPhi(phiHbyA, U, p);

// Update the fixedFluxPressure BCs to ensure flux consistency
setSnGrad<fixedFluxPressureFvPatchScalarField>
(
    p.boundaryField(),
    (
        phiHbyA.boundaryField()
      - fvOptions.relative(mesh.Sf().boundaryField() & U.boundaryField())
    )/(mesh.magSf().boundaryField()*rAUf.boundaryField())
);

                // Non-orthogonal pressure corrector loop
while (pimple.correctNonOrthogonal())
                {
                    // Pressure corrector
                    fvScalarMatrix pEqn
                    (
                    fvm::laplacian(rAUf, p) == fvc::div(phiHbyA)
                    );

                    pEqn.setReference(pRefCell, pRefValue);

    pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));
```

[6] www.openfoam.org

As in [108], the previously defined field `phiTemp` is calculated identically to the later computation of the proper flux field `phi`.

```
        phiTemp = phiHbyA - pEqn.flux();
```

Analogous to the calculations presented in Section A.3, where fluxes are computed based on the variable `phi`, this needs to be done for the separately defined variable `phiTemp`, as well. Except for version differences, the following piece of code is identical to `pisoResistanceFoam` from [108].

```
        if ( Pstream::parRun() )  // parallel run
        {
          word outletPatchNameArray[nOutlets];
          label outletPatchArray[nOutlets];
          for (int i=0; i<nOutlets; i++)
          {
      std::ostringstream convertNum2StrBla;
      convertNum2StrBla << i+1;
      outletPatchNameArray[i] = "OUTLET"+convertNum2StrBla.str();
      outletPatchArray[i] = mesh.boundaryMesh().findPatchID(outletPatchNameArray[i
          ]);
          }

          // get outfluxes:
          for(int outletCount=0; outletCount<nOutlets; outletCount++)
          {
      scalarField tmpOutflux(nProcessors,0);
      tmpOutflux[thisProcNb] = sum(phiTemp.boundaryField()[outletPatchArray[
          outletCount]]);
      reduce( tmpOutflux, sumOp<scalarField>() );

      // sum outfluxes of all processors:
      outFluxArray[outletCount] = 0;
      for (int co=0; co<nProcessors; co++)
      {
        outFluxArray[outletCount] += tmpOutflux[co];
      }
          }
```

The calculation of the outlet pressures is done according to eq. 3.3. Please note, how the value of `capillaryPressureArray` is interpolated linearly.

```
          // calculate new p values @ outlets and scatter
          if (count > 0)
          {
      for(int outletCount=0;outletCount<nOutlets;outletCount++)
      {
        newPressureValueArray[outletCount] = (capillaryPressureArray[outletCount][
            timeCounter%nTimeStepsInPulse] + (capillaryPressureArray[outletCount][
            timeCounter%nTimeStepsInPulse+1] - capillaryPressureArray[outletCount][
            timeCounter%nTimeStepsInPulse]) / dt * deltaTimeCounter //
            capillaryPressure at computed Time
                + (1/rho) * resistanceArray[outletCount] * outFluxArray[outletCount
                    ]);
```

The new pressure values are collected and redistributed to all processors.

```
        scalar tmpNewPressureValue;
        for (int outletCount=0; outletCount<nOutlets; outletCount++)
        {
          tmpNewPressureValue = newPressureValueArray[outletCount];
          Pstream::scatter(tmpNewPressureValue);  // broadcast new p value to all
              slave processors
          newPressureValueArray[outletCount] = tmpNewPressureValue;
        }
      }
          }
```

The computation of outlet pressures for the first time step is done with the value stored in `initialFluxArray`.

```
        else
        {
    for(int outletCount=0;outletCount<nOutlets;outletCount++)
    {
      newPressureValueArray[outletCount] = (capillaryPressureArray[outletCount][
          timeCounter%nTimeStepsInPulse] + (capillaryPressureArray[outletCount][
          timeCounter%nTimeStepsInPulse+1] - capillaryPressureArray[outletCount][
          timeCounter%nTimeStepsInPulse]) / dt * deltaTimeCounter //
          capillaryPressure at computed Time
              + (1/rho) *resistanceArray[outletCount] * initialFluxArray[
                  outletCount]);

      scalar tmpNewPressureValue;
      for (int outletCount=0; outletCount<nOutlets; outletCount++)
      {
        tmpNewPressureValue = newPressureValueArray[outletCount];
        Pstream::scatter(tmpNewPressureValue);  // broadcast new p value to all
            slave processors
        newPressureValueArray[outletCount] = tmpNewPressureValue;
      }
    }
        }
```

As in `pisoResistanceFoam` from [108], the new pressure values are set at the model outlets.

```
        // set new p values @ outlets
        for(int outletCount=0;outletCount<nOutlets;outletCount++)
        {
    if ( isOutletArray[outletCount] > 0 )
    {
      p.boundaryField()[outletPatchArray[outletCount]] == newPressureValueArray[
          outletCount];
    }
        }

      }
```

The same is done for serial computation.

```
      else  // serial run (one processor)
      {
        word outletPatchNameArray[nOutlets];
        label outletPatchArray[nOutlets];
        for (int i=0; i<nOutlets; i++)
        {
    std::ostringstream convertNum2StrBla;
    convertNum2StrBla << i+1;
    outletPatchNameArray[i] = "OUTLET"+convertNum2StrBla.str();
    outletPatchArray[i] = mesh.boundaryMesh().findPatchID(outletPatchNameArray[i
        ]);
        }

        // sum outfluxes:
        for(int outletCount=0; outletCount<nOutlets; outletCount++)
        {
    outFluxArray[outletCount] = sum(phiTemp.boundaryField()[outletPatchArray[
        outletCount]]);
        }

        // calculate new p values @ outlets:
        if (count > 0)
        {
    for(int outletCount=0;outletCount<nOutlets;outletCount++)
    {
      newPressureValueArray[outletCount] = (capillaryPressureArray[outletCount][
          timeCounter%nTimeStepsInPulse] + (capillaryPressureArray[outletCount][
          timeCounter%nTimeStepsInPulse+1] - capillaryPressureArray[outletCount][
            timeCounter%nTimeStepsInPulse]) / dt * deltaTimeCounter //
            capillaryPressure at computed Time
                + (1/rho) * resistanceArray[outletCount] * outFluxArray[outletCount
                    ]);
        }
            }
            else
            {
        for(int outletCount=0;outletCount<nOutlets;outletCount++)
        {
          newPressureValueArray[outletCount] = (capillaryPressureArray[outletCount][
              timeCounter%nTimeStepsInPulse] + (capillaryPressureArray[outletCount][
              timeCounter%nTimeStepsInPulse+1] - capillaryPressureArray[outletCount][
              timeCounter%nTimeStepsInPulse]) / dt * deltaTimeCounter //
              capillaryPressure at computed Time
                  + (1/rho) *resistanceArray[outletCount] * initialFluxArray[
                      outletCount]);

          scalar tmpNewPressureValue;
          for (int outletCount=0; outletCount<nOutlets; outletCount++)
          {
            tmpNewPressureValue = newPressureValueArray[outletCount];
            Pstream::scatter(tmpNewPressureValue);  // broadcast new p value to all
                slave processors
            newPressureValueArray[outletCount] = tmpNewPressureValue;
          }
        }
            }

            // set new p values @ outlets
            for(int outletCount=0;outletCount<nOutlets;outletCount++)
            {
        if ( isOutletArray[outletCount] > 0 )
        {
          p.boundaryField()[outletPatchArray[outletCount]] == newPressureValueArray[
              outletCount];
        }
            }
          }
```

From here the original file `pEqn.H` remains unchanged. The inherent flux value `phi` is computed and velocity values are corrected according to the momentum equation.

```
                    if (pimple.finalNonOrthogonalIter())
                    {
                          phi = phiHbyA - pEqn.flux();
                    }
                }


                #include "continuityErrs.H"

// Explicitly relax pressure for momentum corrector
p.relax();

                U = HbyA - rAU*fvc::grad(p);
                U.correctBoundaryConditions();
    fvOptions.correct(U);
```

transportFoam

In this section the implementation of the second step of the CFD-simulations is described. After solution of the Navier-Stokes equations the stored fields of flux `phi` and diffusion coefficient D are loaded by the solver *transportFoam*, which solves the advection-diffusion equation. Similar to the Navier-Stokes solver *resistanceDataPimpleFoam* this solver is a corrected and improved version of Karsten Sommer's solver *nonNewtonianTransportFoam*, which is described in detail in [108],as well. In the following, the solver code is described where substantial differences are pointed out, omitting alterations, which were required due to the different employed OpenFOAM version (2.3.1 instead of 2.2.2 as in [108].

The sub-timestepping technique that is presented in Section 3.5 is integrated in the solver using OpenFOAM's `subcycle` class, which is called along with other headers, such as `DataEntry.H` to enable read-in of stored physical fields.

```
#include "fvCFD.H"
#include "DataEntry.H"
#include "subCycle.H"
```

Similar header files as for the Navier-Stokes solver *resistanceDataPimpleFoam*, are included; however, the required input parameters for this solver are stored in an additional file *controlDict2*.

```
int main(int argc, char *argv[])
{
  #include "setRootCase.H"
  Foam::Time runTime("controlDict2", args);
  #include "createMesh.H"
  #include "createFields.H"
  #include "initContinuityErrs.H"
  #include <sstream>
```

As above `IOdictionary` class type objects are used to read required input variables from the files *controlDict* as well as *controlDict2*, such as the pulse duration or the number of sub-cycles deployed.

```
  Info<< "Reading pulse properties ..." << nl << endl;

  IOdictionary controlDictCopy
  ( IOobject("controlDict",runTime.system(),mesh,IOobject::MUST_READ,IOobject::
      NO_WRITE ) );

  autoPtr<DataEntry<scalar> > nOutletsValue
  (   DataEntry<scalar>::New ("nOutlets", controlDictCopy) );
  int nOutlets = nOutletsValue().value(0);

  IOdictionary controlDictCopy2
  ( IOobject("controlDict2",runTime.system(),mesh,IOobject::MUST_READ,IOobject::
      NO_WRITE) );

  autoPtr<DataEntry<scalar> > pulseDuration
  ( DataEntry<scalar>::New("pulseDuration", controlDictCopy2) );  //obtain
      pulseDuration from controlDict2!!

  autoPtr<DataEntry<scalar> > startTime
  ( DataEntry<scalar>::New("startTime",controlDictCopy2) );

  autoPtr<DataEntry<scalar> > nSubCyclesValue
  ( DataEntry<scalar>::New("Subcycles",controlDictCopy2) );
  int nSubCycles = nSubCyclesValue().value(0);
  Info << "No. of subcycles: " << nSubCycles << nl;

  // organize time objects:
  scalar pulseT = pulseDuration().value(0);
  scalar dt1 = runTime.deltaT().value();
```

As in the solver *resistanceDataPimpleFoam*, the number of timesteps per cycle is calculated.

```
  double ratio = pulseT / dt1;
  // round ratio:
  double ratioRounded = ratio * 10000.0;
  int tmp = ratioRounded + 0.5;
  float tmpFloat = tmp;
  ratioRounded = tmpFloat / 10000.0;
  int nTimeStepsInPulse1 = ratioRounded;
```

The value of the object `runTime` is set to the start time, which is set in the file *controlDict2*.

```
scalar breakTime = startTime().value(0);  // time where periodicFoam was
    interupted = new start time of runTime
runTime.value() = breakTime;

Info << "nTimeStepsInPulse1: " << nTimeStepsInPulse1 << nl;
Info << "breakTime: " << breakTime << nl;
```

Subsequently, an additional object of class type `Time` is generated. This object `runTime2` governs the time passed within one cardiac cycle. It thus represents the time that has passed since the beginning of each new cycle.

```
// create inner-cycle time:
Foam::Time runTime2("controlDict2", args);

scalar cycleStartTime = breakTime - (nTimeStepsInPulse1)*dt1;
runTime2.value() = cycleStartTime;

Info << "cycleStartTime: " << cycleStartTime << nl << endl;
```

As before, the names of inlet and outlets are defined and the processors containing parts of these patches are determined.

```
word inletPatchName = "INLET";
label inletPatch = mesh.boundaryMesh().findPatchID(inletPatchName);
word outletPatchNameArray[nOutlets];
label outletPatchArray[nOutlets];
for (int i=0; i<nOutlets; i++)
{
  std::ostringstream convertNum2Str;
  convertNum2Str << i+1;
  outletPatchNameArray[i] = "OUTLET"+convertNum2Str.str();
  outletPatchArray[i] = mesh.boundaryMesh().findPatchID(outletPatchNameArray[i]);
  Info << "outletPatchArray:" << outletPatchArray[i] << nl;
}

label thisProcNb = Pstream::myProcNo();
int nProcessors = Pstream::nProcs();
```

Based on these variables the areas of the inlet and outlet patches are retrieved from the mesh. This is done since average values of concentrations are required at each time step of the simulation. This is indeed an important difference to the previous version of the transport-solver. In `nonNewtonianTransportFoam` from [108], average values on the patches are only computed by division with the number of cells on the patch, which strictly speaking is only correct for cell-faces all being of equal size on the patch. However, in a curved and complex tube-like geometry with mesh refinement towards vessel walls, this is barely the case. The obtained values for patch areas are then given as output to the log-file.

```
// get inlet patch size and area:
scalarField totalInletArea(nProcessors,0);
scalar localInletArea = sum(mesh.magSf().boundaryField()[inletPatch]);
totalInletArea[thisProcNb] = localInletArea;
reduce( totalInletArea, sumOp<scalarField>() );
scalar totalInletAreaSum = 0;

for (int co=0; co<nProcessors; co++)
{
totalInletAreaSum += totalInletArea[co];
}
Info << nl << "inlet patch area: " << totalInletAreaSum << nl;

//get outlet patch sizes and areas:
scalar totalOutletAreas[nOutlets];
scalar tmpArea = 0;

for(int outletCount=0; outletCount<nOutlets; outletCount++)
{
  scalarField tmpOutletArea(nProcessors,0);
  tmpArea = sum(mesh.magSf().boundaryField()[outletPatchArray[outletCount]]);
  tmpOutletArea[thisProcNb] = tmpArea;
  reduce( tmpOutletArea, sumOp<scalarField >() );
  scalar totalOutletAreaSum = 0;

  for (int co=0; co<nProcessors; co++)
  {
    totalOutletAreaSum += tmpOutletArea[co];
  }
  totalOutletAreas[outletCount] = totalOutletAreaSum;
  Info << "outlet" << outletCount+1 << " patch area: " << totalOutletAreas[
      outletCount] << nl;
}
```

In the following, the code is left unchanged from `nonNewtonianResistanceFoam` from [108]. A class type `fvMesh`-object is constructed, based on the previously created `runTime`-object `runTime2`.

```
// create new mesh:
Foam::fvMesh mesh2
( Foam::IOobject(Foam::fvMesh::defaultRegion,runTime2.timeName(),runTime2,Foam::
    IOobject::MUST_READ) );
```

After starting the run time loop, the `runTime2` is incremented automatically each timestep. At the beginning of each new cycle this variable is then reset to its initial value.

```
while (runTime.loop())
{

  Info<< "runTime = " << runTime.timeName() << nl;

  runTime2.operator++();

  // * * * * if beginning of new cycle reached: reset cycle-time * * * * * * * *
      * * * //

  scalar t = runTime.value();
  ratio = t / pulseT;

  // round ratio:
  ratioRounded = ratio * 10000000.0;
  tmpFloat = ratioRounded + 0.5;
  int ratioInt = tmpFloat / 10000000.0;
  float modulo = t - static_cast<float> (ratioInt) * pulseT;
  double moduloRounded = modulo * 10000000.0;
  tmpFloat = moduloRounded + 0.5;
  moduloRounded = tmpFloat / 10000000.0;


  Info << "moduloRounded: " << moduloRounded << nl;

  Info << "t: " << t  << nl;
  Info << "breakTime: " << breakTime  << nl;

  if ( t > breakTime )
  {
    if ( moduloRounded < 0.0000001 )  // new cycle!
    {
Info << nl << "New cycle started!" << nl << endl;

// reset cycle-time to startTime:
runTime2.value() = cycleStartTime - dt1;
runTime2.operator++();
    }
  }
```

In order to solve the advection-diffusion equation the stored `phi`-fields are stored in the variable `phiTemp`.

```
  Info<< "Reading field phi" << nl;
  surfaceScalarField phiTemp
  ( IOobject("phi",runTime2.timeName(),mesh2,IOobject::READ_IF_PRESENT,IOobject::
      NO_WRITE ), mesh2 );
```

Embedded in an `if`-statement to include subtime-stepping if required from within *controlDict2*, the actual solution of the advection-diffusion equation is performed.

```
  if (nSubCycles > 1)
  {
for
(
    subCycle<volScalarField> cSubCycle(c, nSubCycles);
    !(++cSubCycle).end();
)
{
  fvScalarMatrix cEqn
  (
    fvm::ddt(c)
    + fvm::div(phiTemp, c)
    -fvc::div(D * fvc::grad(c))
  );
  cEqn.solve();
}
  }
  else
  {
  fvScalarMatrix cEqn
  (
    fvm::ddt(c)
    + fvm::div(phiTemp, c)
    -fvc::div(D * fvc::grad(c))
  );
  cEqn.solve();
  }
```

In the next steps, the averages of the mass concentration (variable `c`) across the inlet and outlet patches is computed. Again the implementation here differs from *nonNewtonianTransportFoam* from [108] such that the concentration values are weighted by the fraction of each cell-face within the whole patch. With the help of the values for the inlet and the outlet patch areas computed at the beginning, the average concentration is written in the log-file.

```
// get inlet concentration average:
scalarField totalInletConcentrationAverage(nProcessors,0);
scalar localInletConcentrationAverage = sum(
mesh.magSf().boundaryField()[inletPatch]
*c.boundaryField()[inletPatch]
) / totalInletAreaSum;
totalInletConcentrationAverage[thisProcNb] = localInletConcentrationAverage;
reduce( totalInletConcentrationAverage, sumOp<scalarField>() );
scalar totalInletConcentrationAverageSum = 0;
for (int co=0; co<nProcessors; co++)
{ totalInletConcentrationAverageSum += totalInletConcentrationAverage[co]; }
Info << "inlet concentration average:" << totalInletConcentrationAverageSum << nl
    ;

//get outlet concentration averages:

for(int outletCount=0; outletCount<nOutlets; outletCount++)
{
  scalarField tmpOutletConcentrationAverage(nProcessors,0);
  tmpOutletConcentrationAverage[thisProcNb] = sum(
mesh.magSf().boundaryField()[outletPatchArray[outletCount]]
*c.boundaryField()[outletPatchArray[outletCount]]
) / totalOutletAreas[outletCount];
  reduce( tmpOutletConcentrationAverage, sumOp<scalarField>() );
  scalar totalOutletConcentrationAverageSum = 0;
  for (int co=0; co<nProcessors; co++)
  {
    totalOutletConcentrationAverageSum += tmpOutletConcentrationAverage[co];
    }
      Info << "outlet" << outletCount+1 << " concentration average:" <<
           totalOutletConcentrationAverageSum << nl;
  }
```

If the `endTime` defined in *controlDict* is reached, the run time loop is stopped and the simulation ends (cf. [108]).

```

    runTime.write();

    Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
    << "  ClockTime = " << runTime.elapsedClockTime() << " s"
    << nl << endl;
  }
  return 0;
}
```

A.4 Affidavit - Eidesstattliche Erklärung

Affidavit

I hereby confirm that my thesis entitled *Development of an in-silico Model of the Arterial Epicardial Vasculature* is the result of my own work. I did not receive any help or support from commercial consultants. All sources and / or materials applied are listed and specified in the thesis.

Furthermore, I confirm that this thesis has not yet been submitted as part of another examination process neither in identical nor in similar form.

Place, Date Signature

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, die Dissertation *Entwicklung eines in-silico Modells der Arteriellen Vaskulatur* eigenständig, d.h. insbesondere selbständig und ohne Hilfe eines kommerziellen Promotionsberaters, angefertigt und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet zu haben.

Ich erkläre außerdem, dass die Dissertation weder in gleicher noch in ähnlicher Form bereits in einem anderen Prüfungsverfahren vorgelegen hat.

Ort, Datum Unterschrift

A.5 Own Share Statement

Statement of individual author contributions and of legal second publication rights

Publication: Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM (2017) Analysis of Coronary Contrast Agent Transport in Bolus-Based Quantitative Myocardial Perfusion MRI Measurements with Computational Fluid Dynamics Simulations. Pop M., Wright G. (eds), FIMH 2017. LNCS 10263. Springer. 2017, p. 369-380. DOI: 10.1007/978-3-319-59448-4_35

Participated in	**Author Initials**, Responsibility decreasing from left to right				
Study Design	JM	LMS	SP	MS	JPHMvdW
Methods Development	JM	SP	JPHMvdW	LMS	MS
Data Collection	JM	SP	JPHMvdW	LMS	MS
Data Analysis and	JM	LMS	SP	MS	JPHMvdW
Interpretation	JM	LMS	SP	MS	JPHMvdW
Manuscript Writing	JM	LMS	SP	MS	JPHMvdW
Writing of Introduction	JM	LMS	SP	MS	JPHMvdW
Writing of Material &	JM	LMS	SP	MS	JPHMvdW
Methods	JM	LMS	SP	MS	JPHMvdW
Writing of Discussion	JM	LMS	SP	MS	JPHMvdW
Writing of First Draft	JM	LMS	SP	MS	JPHMvdW

The doctoral researcher confirms that he has obtained permission from both the publishers and the co-authors for legal second publication.
The doctoral researcher and the primary supervisor confirm the correctness of the above mentioned assessment.

Name: J. Martens	____________________	______________________________
Name: L. Schreiber	____________________	______________________________
	Place, Date	Signature

Statement of individual author contributions to figures/tables/chapters included in the manuscipt

Publication: Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM (2017) Analysis of Coronary Contrast Agent Transport in Bolus-Based Quantitative Myocardial Perfusion MRI Measurements with Computational Fluid Dynamics Simulations. Pop M., Wright G. (eds), FIMH 2017. LNCS 10263. Springer. 2017, p. 369-380. DOI: 10.1007/978-3-319-59448-4_35					
Figure	**Author Initials**, Responsibility decreasing from left to right				
1	JM	LMS	SP	MS	JPHMvdW
2	JM	LMS	SP	MS	JPHMvdW
3	JM	LMS	SP	MS	JPHMvdW
4	JM	LMS	SP	MS	JPHMvdW
5	JM	LMS	SP	MS	JPHMvdW
6	JM	LMS	SP	MS	JPHMvdW

I also confirm my primary supervisor's acceptance.

Name: J. Martens ____________________ ______________________________

Place, Date Signature

A.6 Curriculum Vitae

A.7 List of Publications

Scientific Publications

2019 Dewey M, Siebes M, Kachelrieß M, Kofoed KF, Maurovich P,Nikolaou K, Bai W, Kofler A, Manka R, Kozerke S, Plein S, Chiribiri A, Schaeffter T, Michaellek F, Bengel F, Nekolla S, Knaapen P, Lubberink M, Senior R, Tang M-X, Piek JJ, van de Hoef T, Martens J, Schreiber LM: *EXPERT CONSENSUS DOCUMENT – Quantitative cardiac imaging for the assessment of myocardial ischemia.* Nature Reviews Cardiology. Submitted.

2019 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM: *Development of a Computational Fluid Dynamics (CFD)-Model of the Arterial Epicardial Vasculature.* Zemzemi N., Ozenne V., Vigmond E., Coudière Y. (eds), FIMH 2019. LNCS. Springer. 2019. Submitted.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM: *Analysis of Coronary Contrast Agent Transport in Bolus-Based Quantitative Myocardial Perfusion MRI Measurements with Computational Fluid Dynamics Simulations.* Pop M., Wright G. (eds), FIMH 2017. LNCS 10263. Springer. 2017, p. 369-380. DOI: 10.1007/978-3-319-59448-4_35.

Conference Contributions

2018 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Analyse des Kontrastmitteltransports in Herzkranzgefäßen mittels Computational Fluid Dynamics (CFD) Simulationen unter Berücksichtigung des intramyokardialen Gewebedrucks* (Talk), Proc Germ Soc Med Phys 49.

2018 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Computational Fluid Dynamics Analysis of Contrast Agent Bolus Dispersion in the Coronary Vasculature Including Arterioles* (Poster), Proc Biophys Soc (Thematic Meeting: Heart by Numbers), Berlin, Germany.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Analysis of Coronary Contrast Agent Transport in Bolus-Based Myocardial Perfusion MRI Measurements with Computational Fluid Dynamics Simulations* (Poster), GSLS EUREKA.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *CFD-Analysis of Contrast Agent Transport in Coronary Arteries and its Impact on Quantification of Myocardial Blood Flow with Bolus-Based Perfusion MRI Measurements* (Talk), Proc Germ Soc Med Phys 48.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Analysis of Coronary Contrast Agent Transport in Bolus-Based Quantitative Myocardial Perfusion MRI Measurements with Computational Fluid Dynamics Simulations* (Talk), Proc Functional Imaging and Modelling of the Heart 9.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Analyse systematischer Messfehler bei der quantitativen Perfusions-MRT von Computational Fluid Dynamics (CFD) Simulationen des Kontrastmitteltransports in Koronargefäßen* (Poster), Proc Germ Soc Cardiol.

2017 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Computational Fluid Dynamics (CFD) Simulationen des Kontrastmitteltransports in Koronargefäßen und deren Umsetzung auf Hochleistungscomputern* (Poster), Proc Germ Cardiol Diag Days.

2016 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *Computational Fluid Dynamics (CFD) Simulations of Mass Transport in Coronary Arteries on High Performance (HPC) Clusters* (Poster), GSLS EUREKA.

2016 Martens J, Panzer S, van den Wijngaard JPHM, Siebes M, Schreiber LM, *CFD-Simulationen des Massentransports in Koronararterien und die Umsetzung auf Hochleistungscomputern* (Talk), Proc Germ Soc Med Phys 47.

2016 Martens J, Schreiber LM, *Simulating Coronary Mass Transport with OpenFOAM* (Talk), CINECA: HPC enabling of OpenFOAM for CFD applications, Bologna, Italy.

A.8 Acknowledgements

An erster Stelle möchte ich mich vielmals bei Frau Prof. Schreiber bedanken – zum Einen für die Möglichkeit, wieder in Wissenschaft und Forschung zu arbeiten, zum Anderen für ihre Unterstützung bei meiner Promotion. Insbesondere gebührt ihr Dank für die anspruchsvollen Diskussionen, wichtige Hinweise und immer neuen Antrieb.

Herzlichen Dank an Herrn Prof. Bauer und Herrn Prof. Köstler für die Begutachung und Ko-Betreuung meiner Dr.-Arbeit. Herzlichen Dank an Herrn Prof. Jakob für die Übernahme des Prüfungsvorsitzes.

Vielen Dank an Sabine Panzer, Jeroen van den Wijngaard und Maria Siebes für die gute Zusammenarbeit.

Lieben Dank an alle Kollegen und Ex-Kollegen (insbesondere auch Henning) des Lehrstuhls für Molekulare und Zelluläre Bildgebung, in der Geschäftsstelle und am DZHI im Allgemeinen, für die Zerstreuung und Ablenkung zum Beispiel in den Mittagspausen oder kurze Gespräche auf dem Flur. Besonderer Dank gilt Maria für geduldige Erklärungen und Korrekturen rund um die MRT.

Bei der Graduate School of Life Sciences bedanke ich mich von Herzen für die organisatorische, finanzielle und menschliche Unterstützung.

Bei Adrien bedanke ich mich sehr für keine Zweck-WG! ;-)

Meinem guten Freund Eymen danke ich für inhaltliche Anmerkungen und Korrekturen zur besseren Lesbarkeit.

Größter Dank gebührt meiner Familie im weitesten Sinn aber insbesondere meinen Eltern Alwine und Reinhard sowie meinen Geschwistern Moritz, Wolfram und Carola für Unterstützung, Ablenkung und offene Ohren sowie die richtigen Worte zur richtigen Zeit. Ausdrücklich bedanken möchte ich mich bei meinen Geschwistern für teilweise sehr kurzfristiges und trotzdem immer detailliertes und aufheiterndes Korrektur-Lesen!

Zu guter Letzt möchte ich mich besonders bei Debo bedanken für ihre vielfältige unschätzbare Unterstützung während der bewegten Zeit meiner Promotion – an guten wie an schlechten Tagen... ;-)