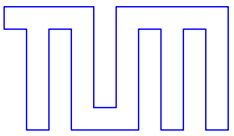
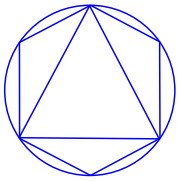


TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR MATHEMATIK



Forschungs- und Lehreinheit M3:
Wissenschaftliches Rechnen

Blood Flow Simulation for Angiographic Interventions

Blutfluß Simulation für angiographische Eingriffe

Diplomarbeit von Maximilian Hainz

Betreuer:	Dr. Martin Groher
Aufgabensteller:	Prof. Dr. Oliver Junge
Abgabadatum:	28.05.2009

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this diploma thesis only supported by declared resources.

München, den 28.05.2009

Maximilian Hainz

Abstract

For assessment of cerebrovascular diseases, it is beneficial to obtain three–dimensional (3D) information on vessel morphology and hemodynamics. Rotational angiography is routinely used to determine the 3D geometry interoperatively. In this thesis, a method to exploit the same acquisition to determine the blood flow waveform and the mean volumetric flow rate in the large cerebral arteries is proposed.

The method uses a model of contrast agent dispersion to determine the flow parameters from the spatial and temporal progression of the contrast agent concentration, represented by a flow map. Furthermore a method for visualization of the model is presented. The method was validated with several geometries.

Zusammenfassung

Um zerebrovaskulär Krankheiten zu beurteilen, ist es von Vorteil dreidimensionale (3D) Information über Gefäßmorphologien und Hämodynamik zu bekommen. Bei interoperativen Eingriffen wird Rotationsangiografie standardmäßig eingesetzt, um die 3D Information zu bestimmen. In dieser Diplomarbeit wird eine Methode vorgestellt um mit Hilfe von – aus Rotationsangiographie gewonnenen – Daten, die Wellenform des Blutes und den durchschnittlichen Blutfluß in den großen zerebralen Arterien zu bestimmen.

Die Methode verwendet ein Kontrastmittel–dispersions–Ausbreitungsmodel um die Fluß–Parameter aus dem räumlich–/ zeitlichen Fortschreiten des Kontrastmittels zu bestimmen, welches mit Hilfe einer "Flow Map" dargestellt wird. Zusätzlich wird eine Methode zur Visualisierung des Models vorgestellt. Die Methode wurde mit verschiedenen Gefäßgeometrien validiert.

Acknowledgments

First of all I would like to thank Dr. Martin Groher for bringing me into this very interesting field of research and for an extraordinary supervision. Directly following him, I'd like to thank Maximilian Baust. Any time I needed help, a new idea or just new motivation they were the ideal team for leading me through this thesis.
I would like to thank Prof. Dr. Oliver Junge for the mathematical supervision and for drawing attention to this interesting topic.
Last but not least I want to express my gratitude to Prof. Navab and the people of CAMP who gave me access to the student room and its facilities.

Contents

1 Introduction

For assesment of cerebrovascular diseases, it is beneficial to obtain 3D information on vessel morphology and hemodynamics. Rotational angiography is routenely used to determine the 3D geometry during an intervention. Additionally to 3D geomerty, rotational angiography provides 2D projections, which contain further information. In this diploma–thesis, a model–based method to quantify flow from rotational angiography is proposed. The volumetric mean flow rate and the blood flow waveform are estimated. For the case of a bifurcating vessel, the flow division at the bifurcation can also be estimated. The model is based on a advective dispersion model, which can explain the changing appearance of the contrast agent bolus and therefore can use the information from the inflow and outflow phase and observations at different injection sites without introducing a bias. The model uses a centerline and radius representation of the vessel tree which allows vessel tapering, stenoses and bifurcations. As it uses all observations at the same time, it is very robust to noise.
Furthermore a method for visualizing the estimated data is provided in this diploma thesis. Here is a brief outline of the structure of the thesis:
In chapter 2 the necessary basics for the method are provided. In particular we discuss view– and camera geometry, solving the diffusion–advection equatation as well as (un–)constrained optimization. Finally the medical background of the problemis proposed.
After presenting basics, we formulate the problem in chapter 3. We will have a close look on how to extract hemodynamic information out of data, given by rotational angiography. Afterwards we propose models for the injection of contrast agent and contrast agent propagation. We discuss a parameter estimation with these models. Last but not least a method for visualization of the estimated data is presented.
Chapter 4 discusses the validation of the problem of the previous chapter. Two different validation steps show the convergence and the robustness to noise of the method.
Chapter 5 discusses the achieved results and gives short outlook.

2 Basics

In this chapter we discuss the theoretical content of this diploma–thesis. We will have a short look on view geometry and camera transformations. This is important, because we deal with data obtained by rotational angiography.
Following, we discuss a numerical treatise of solving the diffusion–advection equation in 2D. This is a central point, needed to model the contrast agent propagation in a vessel. Afterwards we mention concepts for a parameter estimation.
The last section gives a background in interventional angiographic imaging. We talk about blood flow and the state of the art in its estimation.

2.1 Geometry Transformations

This section follows [Har03].
Due to the fact that we want to extract information of data given by rotational angiography, it is important to have a short look on view geometry and camera transformations first. For implementation, we need to know how to deal with different coordinate systems and how to project 3D points, also called voxel, on a 2D plane.

2.1.1 View Geometry

This subsection introduces the concept of Euclidean transformations. We need this concept everytime we have two or more different Euclidean coordinate frames. In this thesis we have two major coordinate frames, the camera– and the world coordinate frame. To point out the concept of Euclidean transformations, we give a definition:

Definition 1 *An Euclidean transformation is a function*

$$f : \mathbb{R}^n \to \mathbb{R}^n : \quad f(x) \mapsto R \cdot x + t, \tag{2.1}$$

with $R \in \mathbb{R}^{n\times n}$, $R^{-1} = R^T$, $\det R = 1$ *and* $t \in \mathbb{R}^n$.

The central property of Euclidean transformations is that they are length preserving. The matrix R is called rotation matrix. We can determine this rotation matrix analytically, if the rotation axis $\vec{r} \in \mathbb{R}^n$ ($\|\vec{r}\|_2 = 1$) and the angle δ is known:

$$Rx = \cos\delta \cdot x + \sin\delta \cdot \vec{r} \times x + (1 - \cos\delta)\left(\vec{r}^T x\right) \cdot \vec{r}. \tag{2.2}$$

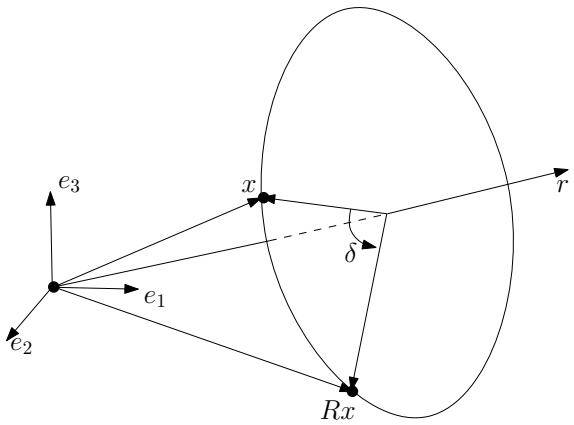


Figure 2.1: A point x and its image Rx

For example in $\mathbb{R}^3$ this rotation matrix is given by (see figure 2.1):

$$R = \cos\delta \cdot I + \sin\delta \cdot \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix} + (1 - \cos\delta) \cdot (\vec{r}\vec{r}^T) \tag{2.3}$$

In computer vision, a point $x \in \mathbb{R}^n$ is usually represented in homogeneous coordinates. homogeneous coordinates are defined as follows:

Definition 2 *A point* $\mathbf{x} \in \mathbb{R}^{n+1}$ *is called homogeneous point of* $x \in \mathbb{R}^n$*, if it satisfies the following relation:*

$$\mathbf{x} = \lambda \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \lambda \in \mathbb{R}\setminus\{0\} \tag{2.4}$$

In this diploma–thesis we use bold letters to indicate homogeneous coordinates. The big advantage of representing points using homogeneous coordinates is that all Euclidean transformations can be described by a matrix vector product:

$$y = R \cdot x + t \quad \Longleftrightarrow \quad \mathbf{y} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot \mathbf{x}. \tag{2.5}$$

Of course, we can calculate an inhomogeneous point x, if the corresponding homogeneous point $\mathbf{x}$ is known:

$$x_i = \frac{\mathbf{x}_i}{\mathbf{x}_{n+1}}, \qquad i \in \{1, \ldots, n\}\,. \tag{2.6}$$

2.1.2 Camera Transformations

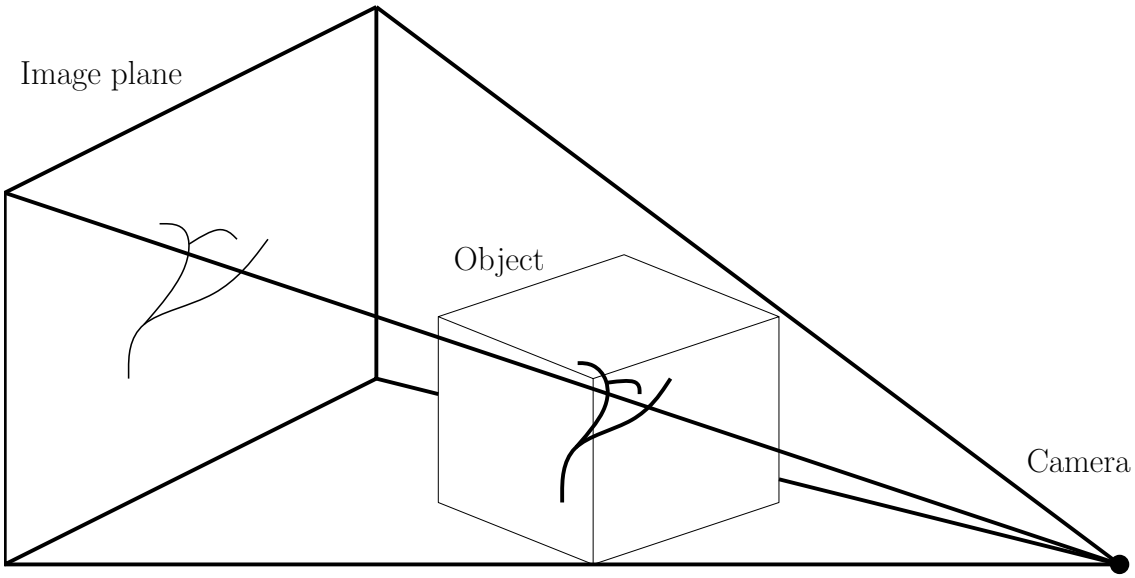


Figure 2.2: Pinhole camera model

In this section we have a look at the finite camera model, which is also called the *pinhole camera* model. The basic principle of this model can be seen in figure 2.2. We have a projection center (camera), a three dimensional volume with data (object) and a plane (image plane) on which the object is projected.
This model is principally designed for CCD[1] like sensors, but also applicable for X–ray images, scanned photographic negatives, etc.
In the pinhole camera model, a point with coordinates $x = [x_1, x_2, x_3]^T$ is mapped to the point $[fx_1/x_3, fx_2/x_3, f]^T$, where f is called the focal length or focal plane. The concept of the model is visualized in figure 2.3. c represents the camera center, x^{2D} is the mapped point of x. The mapped point is at the intersection of the image plane and the lengthened connecting line $\overline{cx}$. In particular, the mapping using a homogeneous representation is defined by

[1]Charged coupled device

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx_1 \\ fx_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \tag{2.7}$$

This expression assumes that the origin of coordinates in the image plane is in the principal point (see image 2.3). The origin of an image is usually in the lower left corner. So in general the mapping is defined by:

$$(x_1, x_2, x_3)^T \mapsto \left(f \cdot \frac{x_1}{x_3} + p_{x_1}, f \cdot \frac{x_2}{x_3} + p_{x_2} \right)^T \quad \text{or} \tag{2.8}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx_1 + x_3 p_{x_1} \\ fx_2 + x_3 p_{x_2} \\ x_3 \end{pmatrix} = \underbrace{\left[\begin{array}{ccc|c} f & & p_{x_1} & 0 \\ & f & p_{x_2} & 0 \\ & & 1 & 0 \end{array}\right]}_{K} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \tag{2.9}$$

If we insert x^{2D} as the mapped point of x, then the equation above has the concise form

$$\mathbf{x}^{2D} = K \cdot \left[\; I \; \middle| \; 0 \; \right] \cdot \mathbf{x} \tag{2.10}$$

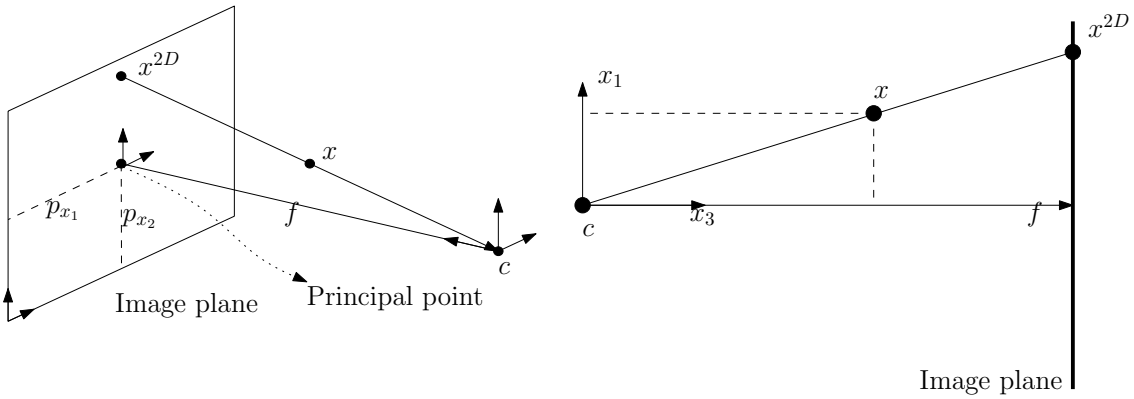


Figure 2.3: Camera model

In general, points in space will be expressed in terms of a different Euclidean coordinate frame. We have two Euclidean frames, the camera– and the world coordinate frame, which are related via rotation and translation. From now on we indicate

points in the camera coordinate frame with an index c. That is to say, a point x in the world coordinate frame corresponds to the point x^c in the camera coordinate frame. The mapping of the points is given by equation (2.5). We can write out the mapping as follows:

$$x^c = R \cdot (x - c)\,, \tag{2.11}$$

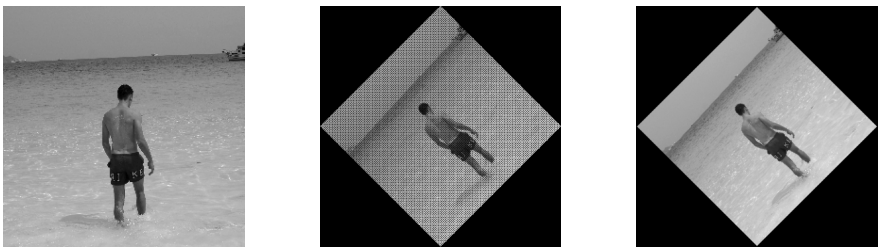
where c is the origin of the camera coordinate frame in world coordinates. R is the rotation matrix, which converts points in world coordinates to the corresponding points in the camera coordinate frame. Rewriting the equation above using a homogeneous representation leads to the formula:

$$\mathbf{x}^c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot \mathbf{x}, \qquad t = -R \cdot c. \tag{2.12}$$

We can combine this equation with (2.9). This leads to the general formulation:

$$\mathbf{x}^{2D} = K \cdot \left[\; R \;\middle|\; t \;\right] \cdot \mathbf{x} = \left[\; M \;\middle|\; p_4 \;\right] \cdot \mathbf{x}, \tag{2.13}$$

where $M \in \mathbb{R}^{3\times3}$ and $p_4 \in \mathbb{R}^3$.
Using this formula in algorithms generates effects that can be compared to *forward warping*. The effect of forward warping appears for example by rotating an image straight forward calculating $y = R \cdot x$. In this expression $R \in \mathbb{R}^{2\times2}$ is a rotation matrix and $x \in \mathbb{R}^2$ is a point in the image. The resulting points y are in general not on the grid of the new image. This is why the concept of *backward warping* is used in algorithms. On the mathematical point of view this is the inverse mapping $x = R^{-1}y$. That is to say, we have a certain point y in the new image and search its origin. The difference between forward– and backward warping can be seen in figure 2.4, where a image is rotated by the angle of 45 degrees using forward– and backward warping.
To avoid this effect in equation (2.13) we use a method called *pixel based volume*



(a) Original Image (b) Rotated Image using forward warping (c) Rotated Image using backward warping

Figure 2.4: The difference between forward– and backward warping

rendering. For a better understanding of *pixel based volume rendering* we first discuss the physical background [PEN99] on X–ray images:
The ionizing property of X–ray beams is used by intraoperative devices such as

C–arms to acquire images. Images can be produced by radiation, because material interacts with the energy of X–rays, part of the energy is lost during traversal. This "loss of energy" is called attenuation and can be physically described by the Lambert–Beer law for attenuation of radiation through a medium:
Given an initial energy E_0 of an X–ray source, the remaining energy E of a ray L traversing an object is given by

$$E = E_0 e^{-\int_L \mu(x)\mathrm{d}x}, \tag{2.14}$$

where $\mu(x)$ is the attenuation coefficient of the object at point x. X–ray intensities are measured in Hounsfield units, i.e. intensity values normalized by the attenuation of water:

$$H(\mu) = 1000\frac{\mu - \mu_W}{\mu_W}, \tag{2.15}$$

where μ is the attenuation coefficient of a certain material, and μ_W that of water. Combining this equation with equation (2.14) leads to:

$$E_I = E_0 e^{-\int_L \left(\frac{H(x)\mu_W}{1000} + \mu_W\right)\mathrm{d}x}, \tag{2.16}$$

where E_I is the energy in the image. For contrasted vessels, a logarithmic relationship can also be established between the remaining energy E_I and the resulting intensity, i.e. $I = -\log(E_I/E_0)$. Thus, the intensity I can be assumed to be proportional to the length of the path traversed through the 3D vessel and the contrast material's density ξ:

$$I \propto \mu \int_L \xi(x)\mathrm{d}x, \tag{2.17}$$

where μ is the attenuation coefficient of the contrast material.
For an implementation of (2.17), we have to determine the camera position c in world coordinates for the direction of the ray:

$$c = -M^{-1} \cdot p_4. \tag{2.18}$$

Then the ray is given by

$$\mathbf{x}(\lambda) = \lambda \cdot \begin{pmatrix} M^{-1}\mathbf{x}^{2D} \\ 0 \end{pmatrix} + \begin{pmatrix} c \\ 1 \end{pmatrix}. \tag{2.19}$$

For generating the 2D image I out of a 3D volume we use the formula above for each pixel x^{2D}. In particular we get a ray for each pixel. For a numerical approximation of (2.17) we sample the ray in steps λ_i by the size of the edge of a voxel:

$$\mathbf{x}^{2D} = \mu \cdot \sum_i \xi(\mathbf{x}(\lambda_i)). \tag{2.20}$$

For improving the results it is recommended to use trilinear interpolation for the discrete intensities of $\xi(\mathbf{x}(\lambda_i))$ of the 3D volume. Generating a 2D image out of a

3D volume is also called *digital reconstructed radiograph* (DRR).

So far, we considered the projection of points. Now we have a short look on projecting lengths, especially projecting the radius of a circle. If we have a length l_x corresponding to a point x, we can project it on the image plane by moving this point into the camera coordinate frame (equation (2.11)). Then we calculate a new point $x^c_{l_x}$ in the camera coordinate frame which has distance l_x to x^c by[2]:

$$x^c_{l_x} = x^c + \begin{pmatrix} 0 \\ l_x \\ 0 \end{pmatrix} \tag{2.21}$$

Therefore it is important not to use the x_3–axis, because this is, per definition of the *pinhole camera* model, the normal to the image plane. Finally we get the projected length by:

$$l^{2D}_x = \|K \cdot x^c - K \cdot x^c_{l_x}\| \tag{2.22}$$

2.2 Numerical Basics

The numerical aspects in this thesis are divided into two completely different subsections. On the one hand we will have a look on solving the diffusion–advection equation, because we use a advective dispersion model later on. On the other hand we address the problem of parameter identification. We would like to assume the mean volumetric flow rate and the shape of the waveform of blood flow in vessels. Therefore we develop a parameter dependend model of the blood flow later on.

2.2.1 Solving the diffusion–advection equation

In this section we talk about solving the diffusion–advection equation numerically. The diffusion–advection equation is defined as follows:

$$\frac{\partial C}{\partial t} = \underbrace{D \cdot \Delta_x C}_{\text{diffusion}} - \underbrace{v^T \cdot \nabla_x C}_{\text{advection}}, \tag{2.23}$$

where C is the concentration, D is a diffusion constant and v is the velocity.
This diffusion–advection equation is an extension of the non stationary heat conduction equation

$$\frac{\partial u}{\partial t} = \lambda \Delta_x u, \tag{2.24}$$

[2]We can do this, because Euclidean transformations are length preserving

with an advective term. This extension applies for all balances in non stationary fluids, where the advection can not be disregarded in comparison to the diffusion.

First we introduce the concept of finite differences and afterwards we will apply this concept to the diffusion advection–equation in two dimensional space.

Finite Differences With the help of the Taylor series we can calculate a function value $f(x)$ at a point $x+h$ ($h > 0$), if we know the function value and its derivatives. Inversely we can calculate the derivatives, if we know the function values at points x and $x + h$. Here is the general formulation of the Taylor series:

$$f(x+h) = f(x) + \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) + \cdots \tag{2.25}$$

$$f(x-h) = f(x) - \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) - \cdots \tag{2.26}$$

An approximation of $f'(x)$ can be done by:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \underbrace{\left(-\frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x) - \cdots\right)}_{=\mathcal{O}(h)}. \tag{2.27}$$

This approximation for $f'(x)$ is called *forward difference*. Analogously, we can approximate using *backward differences*:

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h). \tag{2.28}$$

We can reach a higher approximative order for the derivative $f'(x)$, if we subtract equations (2.25) and (2.26):

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}\left(h^2\right) \tag{2.29}$$

This approximation is called *central difference*. Inversely if we add equations (2.25) and (2.26) we get an approximation for f'':

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}\left(h^2\right). \tag{2.30}$$

For $x \in \mathbb{R}^2$, we can approximate analogously:

$$\nabla f(x) = \frac{1}{2h}\begin{pmatrix} f(x+he_1) - f(x-he_1) \\ f(x+he_2) - f(x-he_2) \end{pmatrix} + \mathcal{O}(h^2) \tag{2.31}$$

$$\begin{aligned} \Delta f(x) &= \frac{1}{h^2}\left(f(x+he_1) - 2f(x) + f(x-he_1) + \right. \\ &\quad \left. + f(x+he_2) - 2f(x) + f(x-he_2)\right) + \mathcal{O}(h^2) = \\ &\approx \frac{f(x+he_1) + f(x-he_1) - 4f(x) + f(x+he_2) + f(x-he_2)}{h^2}, \end{aligned} \tag{2.32}$$

where e_1 and e_2 are the normalized basis vectors in $\mathbb{R}^2$. In the formula above $\nabla f(x)$ was approximated using central differences. Of course, we can also estimate $\nabla f(x)$ using forward or backward differences instead of central differences.

Solving the diffusion–advection equation In chapter 3.2 we approximate the diffusion–advection equation in a tube with a radial–symmetric assumption. That is to say, we use a simplified approach for the concentration of iodine in a tube. In particular, we assume that the concentration of iodine, given by the diffusion–advection equation changes in length l and radius r in a tubular structure over time t.
To realize such a radial–symmetric approach we define $(r,l)^T := x \in \mathbb{R}^2$ in equation (2.23). From this it follows that $\nabla_x C\,(x,t)$ is given by $(\partial C/\partial r, \partial C/\partial l)^T$ and $\Delta_x C\,(x,t)$ is given by $(\partial^2 C/\partial r^2 + \partial^2 C/\partial l^2)$.
For simplification reasons we first have a look at the two dimensional stationary diffusion–advection equation, which is given by

$$0 = D\left(\frac{\partial^2 C}{\partial r^2} + \frac{\partial^2 C}{\partial l^2}\right) - v_r \frac{\partial C}{\partial r} - v_l \frac{\partial C}{\partial l}, \tag{2.33}$$

where v_r is the velocity in radial– and v_l is the velocity in longitudinal direction.
We define c_{ij} as the approximate concentration at $(r_i, l_j)^T$, where $(r_i, l_j)^T = (r_0 + ih, l_0 + jh)^T$. For simplification we assume the same grid size h in dimension r and l, which is not necessary. The stationary diffusion–advection equation (2.33) discretisized with finite differences (equations (2.31) and (2.32)) is given by:

$$\begin{aligned} 0 \;=\; & D \cdot \frac{c_{i-1,j} + c_{i,j-1} - 4c_{ij} + c_{i+1,j} + c_{i,j+1}}{h^2} - \\ & - \frac{v_r\,(c_{i+1,j} - c_{i-1,j}) + v_l\,(c_{i,j+1} - c_{i,j-1})}{2h} \quad \forall i,j. \end{aligned} \tag{2.34}$$

We can rewrite this equation and sort it by points:

$$\begin{aligned} 0 \;=\; & \frac{1}{h^2}\left[c_{i-1,j}\left(D + \frac{v_r h}{2}\right) + c_{i,j-1}\left(D + \frac{v_l h}{2}\right) + c_{ij}\,(-4D) + \right. \\ & \left. + c_{i+1,j}\left(D - \frac{v_r h}{2}\right) + c_{i,j+1}\left(D - \frac{v_l h}{2}\right)\right] \end{aligned} \tag{2.35}$$

For an easier spelling we introduce the *star*–operator (see figure 2.5). In this operator we combine all coefficients:

$$\begin{aligned} S(c_{ij}) \;:=\; & \frac{1}{h^2}\left[c_{i-1,j}\left(D + \frac{v_r h}{2}\right) + c_{i,j-1}\left(D + \frac{v_l h}{2}\right) + c_{ij}\,(-4D) + \right. \\ & \left. + c_{i+1,j}\left(D - \frac{v_r h}{2}\right) + c_{i,j+1}\left(D - \frac{v_l h}{2}\right)\right] \end{aligned} \tag{2.36}$$

We can rewrite equation (2.34) using the *star*–operator:

$$0 = S(c_{ij}) \qquad \forall i, j \in \mathcal{I}, \tag{2.37}$$

where $\mathcal{I}$ is the set of all points which are not next to the boundary.
We use this *star* operator and apply it to every point c_{ij} in the grid. As a result we get a equation for every point c_{ij}. We build up a system of equations by assembling the equations for all points c_{ij}.

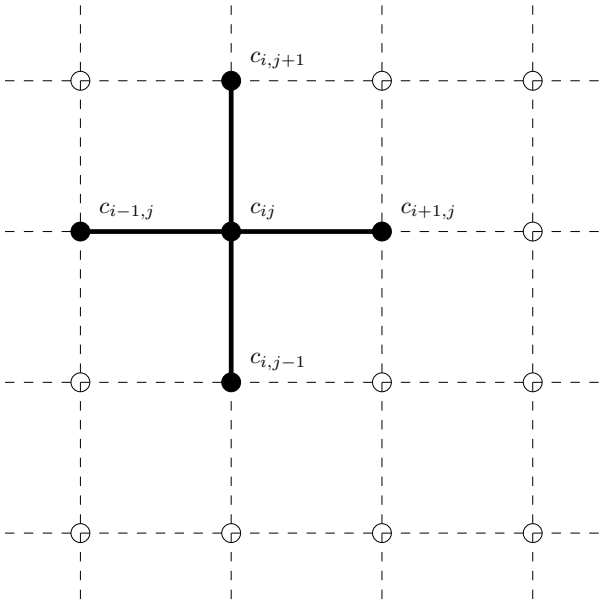


Figure 2.5: Star operator

Boundary conditions To obtain unique solutions, the system of equations has to be solved with boundary conditions. We add these conditions to the system of equations. Due to the fact that no more unknowns get into this system, it gets solvable. We introduce three different boundary conditions. For a simplified approach we just treat the left boundary of the first dimension of our problem. All other boundaries are treated analogously.

1. If we have a *Dirichlet* boundary condition

$$C\,(0, \cdot) = \gamma_D, \tag{2.38}$$

the value for $c_{0,\cdot}$ is directly given. We can use this equation and directly add it into the system of equations.

2. In case of a *Neumann* boundary condition

$$\left.\frac{\partial C}{\partial r}\right|_{r=0} = \gamma_N, \tag{2.39}$$

the derivative in direction to the normal of the boundary is given. If $\gamma_N = 0$ the molecular flow disappears through the boundary. $\gamma_N = 0$ is also used for symmetrical boundaries. We have to discretisize (2.39) with the right hand side differential quotient of first order. Using central differences or the left hand side differential quotient would arise problems, because points outside the model, which do not exist are required.

$$\frac{c_{1,\cdot} - c_{0,\cdot}}{h} = \gamma_N \tag{2.40}$$

For $\gamma_N = 0$ the method (2.40) at the boundary has order $\mathcal{O}(h^2)$. For $\gamma_N \neq 0$ we have order $\mathcal{O}(h)$. Finally we add this new equation to the system of equations.

3. If we have a *Cauchy* boundary condition, the molecular flow in the direction of the normal n^T to the boundary is given proportional to the potential difference $C(0,\cdot) - C(a,\cdot)$. This leads, related to (2.39), to the following formulation:

$$\left.\frac{\partial C}{\partial r}\right|_{r=0} + C(0,\cdot) = \gamma_C. \tag{2.41}$$

Analogously to (2.39) we replace this equation with:

$$\frac{c_{1,\cdot} - c_{0,\cdot}}{h} + c_{0,\cdot} = \gamma_C \tag{2.42}$$

This has to be added to the system of equations, too.

Discretisizing in time We discretisize the diffusion advection equation in time using forward differences (2.27). Therefore we get for the left–hand side of (2.23):

$$\frac{\partial C\,(r,l,t)}{\partial t} \approx \frac{C\,(r,l,t+\tau) - C\,(r,l,t)}{\tau}. \tag{2.43}$$

For an easier notation we introduce $C(r,l,t) = c_{ij}^{(m)}$ respectively $C(r,l,t+\tau) = c_{ij}^{(m+1)}$ for constant time steps τ. With this notation the discretisized diffusion advection equation is given by:

$$\frac{c_{ij}^{(m+1)} - c_{ij}^{(m)}}{\tau} = S\left(c_{ij}^{(m)}\right) \quad \text{resp.} \tag{2.44}$$

$$c_{ij}^{(m+1)} = c_{ij}^{(m)} + \tau \cdot S\left(c_{ij}^{(m)}\right). \tag{2.45}$$

Note that every point $c_{ij}^{(m+1)}$ is calculated by only using points of the former time step (see figure 2.6). We can imagine that this fact causes problems, because points of the same time step are not related.
For a numerical analysis we rewrite equation (2.45):

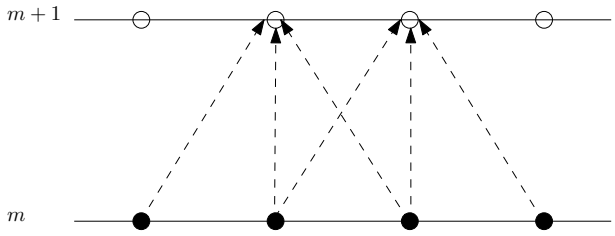


Figure 2.6: Explicit discretization of time

$$\begin{aligned} c_{ij}^{(m+1)} &= rc_{i-1,j}^{(m)}\left(D+\frac{v_r h}{2}\right) + rc_{i,j-1}^{(m)}\left(D+\frac{v_l h}{2}\right) + c_{ij}^{(m)}\left(1-4rD\right) + \\ &\quad + rc_{i+1,j}^{(m)}\left(D-\frac{v_r h}{2}\right) + rc_{i,j+1}^{(m)}\left(D-\frac{v_l h}{2}\right) \quad \text{resp.} \\ c_{ij}^{(m+1)} &= rWc_{i-1,j}^{(m)} + rSc_{i,j-1}^{(m)} + c_{ij}^{(m)}\left(1-4rD\right) + rEc_{i+1,j}^{(m)} + rNc_{i,j+1}^{(m)}, \\ r &= \frac{\tau}{h^2}. \end{aligned} \tag{2.46}$$

We have already discussed that this represents a system of equations. For the numerical analysis we rewrite the equation above as a system of equations[3]:

$$c^{(m+1)} = Ac^{(m)}, \tag{2.47}$$

with

$$A := \begin{pmatrix} 1-4rD & rE & \cdots & rS & & & & \\ rW & 1-4rD & rE & & rS & & & \\ & rW & 1-4rD & rE & & rS & & \\ \vdots & & \ddots & & \ddots & & \ddots & & \ddots & \\ rN & & & rW & & 1-4rD & & rE & & & rS \\ & \ddots & & & \ddots & & \ddots & & \ddots & & \ddots \end{pmatrix},$$

$$c^{(m)} := \left(c_{1,1}^{(m)}, c_{2,1}^{(m)}, \ldots, c_{1,2}^{(m)}, c_{2,2}^{(m)}, \ldots\right)^T.$$

Matrix A is a bandmatrix and dependend of parameters r, τ, h, D and v. For stability of the recursion (2.47), all eigenvalues have to satisfy $\|\lambda_v\| \leq 1$. According to [Sch09], this is equal to

$$\mathcal{C}_\Delta = \frac{\tau}{h}v \leq 1 \qquad \text{and} \qquad \mathcal{M} = \frac{\tau}{h^2}D \leq 1, \tag{2.48}$$

[3]with zero *Dirichlet* boundary

where $\mathcal{C}_\Delta$ is called the *Courant number* and $\mathcal{M}$ is called the module.
As a result we get restrictions concerning the time steps τ. We use an implicit formulation (see figure 2.7) to avoid this problem:

$$\frac{c_{ij}^{(m+1)} - c_{ij}^{(m)}}{\tau} = S\left(c_{ij}^{(m+1)}\right). \tag{2.49}$$

We have to solve a linear system of equations in every time step if we use this formula. This implies more computing time than solving (2.45), but as a result we have no restrictions concerning stability. We can see in figure 2.7 that all points of one time step are linked in this model. Of course, this represents the physics of the problem more accurate.
Both, explicit and implicit formulation have order $\mathcal{O}(\tau, h^2)$. If we want to reach

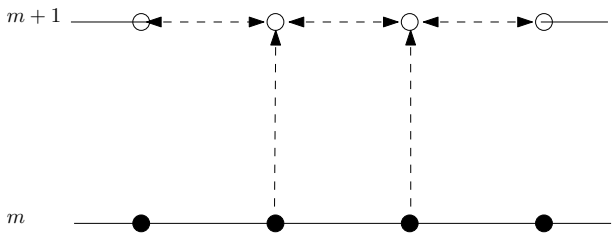


Figure 2.7: Implicit discretization of time

higher order in time, we have to apply central differences instead of forward/backward differences. Doing this without introducing new time steps can be done by evaluating $\partial C/\partial t$ at $t + \tau/2$:

$$\left.\frac{\partial C}{\partial t}\right|_{t+\tau/2} = \frac{c_{ij}^{(m+1)} - c_{ij}^{(m)}}{2 \cdot \frac{\tau}{2}} + \mathcal{O}\left(\tau^2\right). \tag{2.50}$$

If we discretisize the diffusion–advection equation with the formula above, the left

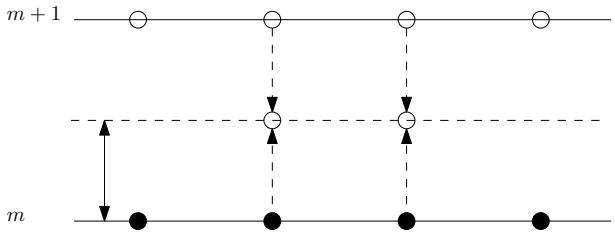


Figure 2.8: Method of Crank–Nicolson

hand side of the equation is the same as in equations (2.45) and (2.49). But the right hand side is the mean value of (2.45) and (2.49) (see figure 2.8):

$$\frac{c_{ij}^{(m+1)} - c_{ij}^{(m)}}{\tau} = \frac{1}{2}\left(S\left(c_{ij}^{(m+1)}\right) + S\left(c_{ij}^{(m)}\right)\right). \tag{2.51}$$

This equation is also known as the method of *Crank–Nicolson* [Per93].
Looking at the three methods (2.45), (2.49) and (2.51) leads to a general formulation:

$$\frac{c_{ij}^{(m+1)} - c_{ij}^{(m)}}{\tau} = \xi \cdot S\left(c_{ij}^{(m+1)}\right) + (1-\xi) \cdot S\left(c_{ij}^{(m)}\right), \qquad \xi \in [0,1]\,. \tag{2.52}$$

For $\xi = 0$ we get the explicit, for $\xi = 1$ the implicit and for $\xi = 1/2$ the Crank–Nicolson method. If we implement the general formulation, we can switch between the methods by varying ξ. For $\xi \geq 1/2$ the method is stable.

Upwind schema We shortly discuss a dominant advection in comparison to the diffusion. Therefore we have a look at a one dimensional pipe flow (see figure 2.9). It is obvious, that the function value at a certain point depends significantly

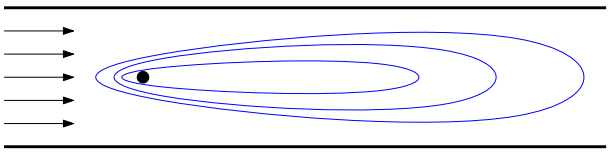


Figure 2.9: Advection–diffusion problem with big advectional ratio

on downstream values. Central differences consider upstream values as well as downstram values. Due to this fact, central differences generate an error respectively produce absurd physical results. We can avoid this fact using the so called *Upwind schema*. In this method only downstream values are considered for calculation of the derivative. The *Upwind schema* is defined as follows:

$$v_l \frac{\partial C}{\partial l} \approx \begin{cases} v_l \dfrac{c_{ij} - c_{i,j-1}}{h}, & v_l > 0 \\ v_l \dfrac{c_{i,j+1} - c_{ij}}{h}, & v_l < 0. \end{cases} \tag{2.53}$$

In contrast to central differences, the *Upwind schema* only has order $\mathcal{O}(h)$, but it represents the physics of the problem more precisely.

Different grid sizes For an easier introduction of the S–operator we assumed same grid size h in dimensions r and l. For different grid sizes we introduce a factor

β, which describes the ratio $\Delta r/\Delta l$. We can continue using the S–operator in which β appears:

$$\begin{aligned} S(c_{ij}) &= \frac{1}{h^2}\left[c_{i-1,j}\left(D+\frac{v_r h}{2}\right)+c_{i,j-1}\beta\left(D+\frac{v_l h}{2}\right)+\right. \\ &\quad \left.+c_{ij}\left(-2D(1+\beta)\right)+c_{i+1,j}\left(D-\frac{v_r h}{2}\right)+c_{i,j+1}\beta\left(D-\frac{v_l h}{2}\right),\right] \\ \beta &= \frac{\Delta r}{\Delta l}, \\ h &= \Delta r. \end{aligned} \tag{2.54}$$

2.2.2 Parameter Identification

In this subsection we discuss the problem:

$$x^* = \arg\min_{x\in\Omega} f(x). \tag{2.55}$$

A solution of (2.55) is defined as follows:

Definition 3 *The solution x^* of (2.55) is called global minimum of f. A point x^* is called a local minimum, if $\exists\delta > 0$ such that*

$$f(x^*) \le f(x), \qquad \forall x \in B_\delta(x^*), \tag{2.56}$$

where $B_\delta(x) := \{y \in \mathbb{R}^n : \|x-y\| \le \delta\}$.

We address this problem in two ways. On the one hand we will have a look at the *downhill simplex algorithm* [Nel65] which is suitable for continuous functions, on the other hand we discuss a standard algorithm for constrained nonlinear optimization – the *sequential quadratic programming* algorithm.

The Nelder–Mead Algorithm At first we discuss an unconstrained optimization algorithm, for which f has only to be continuously, This algorithm is known as the downhill simplex algorithm [Nel65]. Therefore we define $\Omega = \mathbb{R}^n$ in equation (2.55).
For dimension n we need $n+1$ points $x_0, \ldots, x_n$ in space, which define the initial simplex. In practice only one initial value $x^{(0)}$ is given. We can choose x_i as follows:

$$\begin{aligned} x_0 &= x^{(0)}, \\ x_i &= x_0 + \tau_i e_i \qquad i = 1, \ldots, n, \end{aligned} \tag{2.57}$$

with an initial length $\tau_i \in \mathbb{R}^+$. These points define the initial simplex. Furthermore we define:

$$\begin{aligned} x_h &= \max_{i\in\{0,\dots,n\}} f(x_i) \quad \text{[h for high]}, \\ x_l &= \min_{i\in\{0,\dots,n\}} f(x_i) \quad \text{[l for low]}, \\ x_\mu &= \max_{i\in\{0,\dots,n\}\setminus\{h\}} f(x_i), \\ x_c &= \tfrac{1}{n}\sum_{i\neq h} x_i, \quad \text{centroid of the points with } i \neq h. \end{aligned} \tag{2.58}$$

Assuming that we have $n+1$ initial points, the algorithm is given as follows:

1. Determine points x_h, x_l, x_μ.
2. Check whether all points satisfy the termination equation

$$\frac{1}{n+1}\sum_{i=0}^{n}\left(f(x_i)-\bar{f}\right)^2 < \text{TOL} \tag{2.59}$$

$$\text{with} \quad \bar{f} = \frac{1}{n+1}\sum_{i=0}^{n} f(x_i). \tag{2.60}$$

   If not, we do a reflection of the point x_h relative to the point x_c, which is given by

$$x_r = (1+\alpha)\,x_c - \alpha x_h, \tag{2.61}$$

   with a suitable factor $\alpha \geq 0$.
   In case $n = 2$ this can be easily interpreted, because the 3 points, defining the simplex, correspond to x_h, x_l and x_μ. Those are defining a plane with a gradient in direction $x_l - x_h$. In figure 2.10 we see that the step we do is a step in this direction.
3. If $f(x_l) \leq f(x_r) \leq f(x_\mu)$, then x_h is replaced by x_r and we start again from (2.) with the new simplex.
4. If $f(x_r) \leq f(x_l)$, the reflexion step generated a new minimum, which indicates that the local minimum of f might be outside the convex closure of the simplex. To verify this we calculate a new vertex

$$x_e = \beta x_r + (1-\beta)\,x_c, \tag{2.62}$$

   with a stretching factor $\beta > 1$, to expand the simplex (see figure 2.10). There are two possibilities before going back to step (2.):
   a) If $f(x_e) < f(x_l)$, we replace x_h with x_e
   b) Otherwise if $f(x_e) \geq f(x_l)$, we replace x_h with x_r, because $f(x_r) < f(x_h)$

5. If $f(x_r) > f(x_\mu)$, we predict the local minimum within the convex closure of the simplex. We can pursue three strategies to locate the minimum. If $f(x_r) < f(x_h)$ we use the contraction equation:

$$x_{co} = \gamma x_r + (1-\gamma)\, x_c, \qquad \gamma \in (0,1) \quad \text{(ouside contraction)}, \tag{2.63}$$

otherwise

$$x_{co} = \gamma x_h + (1-\gamma)\, x_c, \qquad \gamma \in (0,1) \quad \text{(inside contraction)}. \tag{2.64}$$

The effect of these two equations is visualized in figure 2.11. Finally we replace x_h with x_{co} if $f(x_{co}) < f(x_h)$, or we do a shrink if $f(x_{co}) \geq f(x_h) \vee f(x_{co}) > f(x_r)$ for n new points by dividing the distances of x_i to x_0 by 2 (see also figure 2.11). Then we go back to point (2.).

The nearly universal choices used in the standard Nelder–Mead algorithm are

$$\alpha = 1, \quad \beta = 2, \quad \gamma = \frac{1}{2}. \tag{2.65}$$

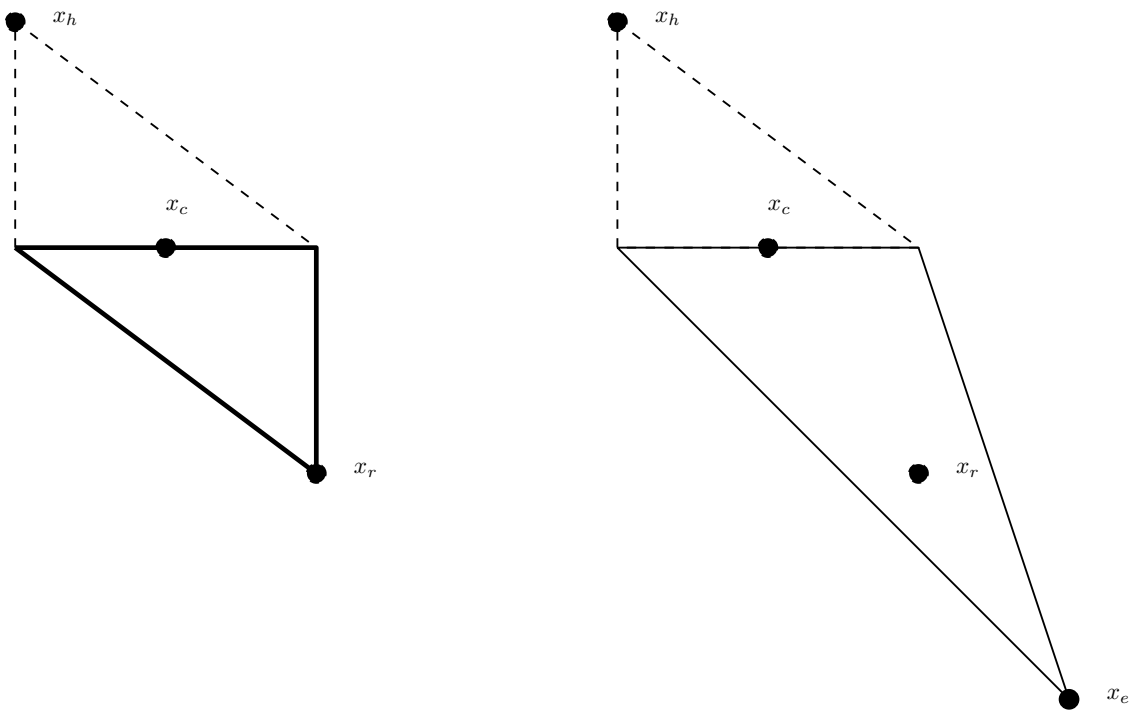


Figure 2.10: Nelder–Mead simplices after a reflection and an expansion step. The original simplex is shown with a dashed line.

Regarding convergence we have the following results according to [Lag98]:

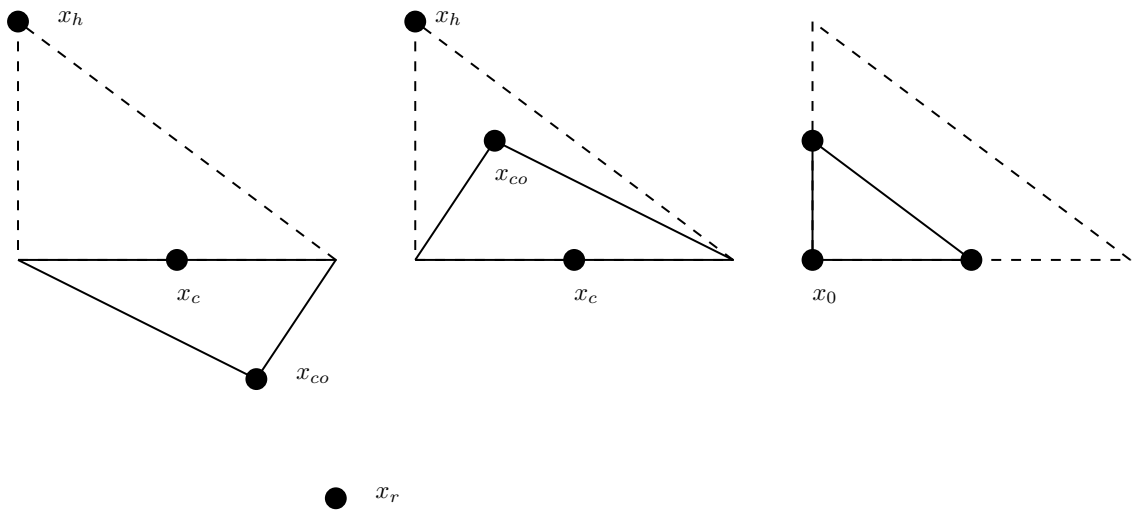


Figure 2.11: Nelder–Mead simplices after an outside contraction, an inside contraction, and a shrink. The original simplex is shown with a dashed line.

1. In dimension 1, the Nelder–Mead method converges to a minimizer, and convergence is eventually M–step linear[4].
2. When the reflection parameter $\alpha = 1$. In dimension 2, the function values at all simplex vertices in the standard Nelder–Mead algorithm converge to the same value.
3. In dimension 2, the simplices in the standard Nelder–Mead algorithm have diameters converging to zero.

Least Squares Optimization The *sequential quadratic programming* (SQP) algorithm is designed for constrained optimization. The SQP algorithm is one of the most efficient mathematical optimizing algorithms. It is the solid basis of several optimizing software packages (i.e. DONLP2, FilterSQP, KNITRO, SNOPT) [Rus06].

We will have a look at the general nonlinear optimizing–problem:

Definition 4 *The equation*

$$\min f(x) \quad \textit{subject to} \quad g(x) \leq 0, \quad h(x) = 0 \tag{2.66}$$

[4]By M-step linear convergence is meant that there is an integer M, independent of the function being minimized, such that the simplex diameter is reduced by a factor no less than 1/2 after M iterations.

is called the general nonlinear optimizing–problem.
The set

$$\Omega = \{x \in \mathbb{R}^n : g(x) \leq 0, h(x) = 0\} \tag{2.67}$$

is called the feasible field of (2.66).
A point $x \in \mathbb{R}^n$ is called feasible, if $x \in \Omega$.

In the following we assume $f : \mathbb{R}^n \to \mathbb{R}, g : \mathbb{R}^n \to \mathbb{R}^m, h : \mathbb{R}^n \to \mathbb{R}^p \in \mathcal{C}^2$. Furthermore we assume Ω as a compact, convex set.
With these assumptions the uniques solutions is guaranteed by the theorem of Weierstraß:

Theorem 1 *Let $(\mathcal{X}, d)$ be a metric space and Ω be a compact set of $\mathcal{X}$. Let $f : \Omega \to \mathbb{R}$ be a continuous function, then the image of $f(\Omega)$ is bounded and $x_\pm \in \Omega$ exists such that*

$$f(x_-) = \min_{x \in \Omega} f(x), \qquad f(x_+) = \max_{x \in \Omega} f(x). \tag{2.68}$$

Furthermore a first necessary condition for a local minimum is given by:

Theorem 2 *Let $\Omega \subset \mathbb{R}^n$ be a convex set, $x^* \in \Omega$ and $f \in \mathcal{C}^1(B_\delta(x^*))$ for an applicable $\delta > 0$. If*

$$\nabla f(x^*)^T (x - x^*) \geq 0 \qquad \forall x \in \Omega, \tag{2.69}$$

then x^* is a local minimum of f,

For the SQP algorithm we introduce the so called Lagrangian function:

Definition 5 *The function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$,*

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x) \tag{2.70}$$

is called lagrangian function of (2.66).

With this definition we can apply the results of [Kuh51]:

Theorem 3 *If f has a restricted local minimum in $x = x^*$, then $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^p$ exists such that*

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0, \tag{2.71}$$
$$h(x^*) = 0, \tag{2.72}$$
$$\lambda^* \geq 0, \quad g(x^*) \leq 0, \quad \lambda^{*T} g(x^*) = 0, \tag{2.73}$$
$$(x^* - x)^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*, \mu^*)(x^* - x) \geq 0, \qquad \forall x \in \Omega. \tag{2.74}$$

These conditions are called the Kuhn–Tucker conditions.

Remind that this theorem requires existing vectors λ^*, μ^*. A further geometrical restriction is needed to ensure this (see [Rus06] pp. 113–114). To complete this short theoretical section we notice that (2.71)–(2.74) are sufficient for existence of a local minimum, if the following precondition is satisfied:

Theorem 4 *Let (x^*, λ^*, μ^*) satisfy the necessary conditions (2.71)–(2.74) and let g be concave, then $f(x^*)$ is a local minimum of (2.66).*

For simplification reasons, we first discuss the problem:

$$\min f(x) \quad \text{subject to} \quad h(x) = 0. \tag{2.75}$$

Let x^* be a local solution of (2.75), then the Kuhn–Tucker conditions (2.71) and (2.72) have to be satisfied. We can define a system of equations by

$$F(x, \mu) := \begin{pmatrix} \nabla_x \mathcal{L}(x, \mu) \\ h(x) \end{pmatrix} = 0. \tag{2.76}$$

The nearly choice is to use the Newton–method to solve this system of equations, because we have $n + p$ unknowns and $n + p$ equations. We can use the Newton–method, because we assumed f and h in $\mathcal{C}^2$. As a result F is in $\mathcal{C}^1$ with

$$F'(x, \mu) = \begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x, \mu) & \nabla^2_{x\mu} \mathcal{L}(x, \mu) \\ \nabla h(x)^T & 0 \end{pmatrix} = \begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x, \mu) & \nabla h(x) \\ \nabla h(x)^T & 0 \end{pmatrix}. \tag{2.77}$$

Let $\left(x^{(k)}, \mu^{(k)}\right)$ be the current iteration value in the Newton–method, then the Newton–step $s^{(k)}$ is given by:

$$F'\left(x^{(k)}, \mu^{(k)}\right) s^{(k)} = -F\left(x^{(k)}, \mu^{(k)}\right). \tag{2.78}$$

The algorithm for solving (2.75) with a given initial value $x^{(0)}$ is described in algorithm 1.

For convergence analysis it is important that the matrix

$$F'(x^*, \mu^*) = \begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x^*, \mu^*) & \nabla h(x^*) \\ \nabla h(x^*)^T & 0 \end{pmatrix} \tag{2.79}$$

is invertable in (x^*, μ^*). Therefore we have this theorem:

Theorem 5 *Let $f, h \in \mathcal{C}^2$. If*

1. *rank $\nabla h(x) = p$,*
2. *$s^T \nabla^2_{xx} \mathcal{L}(x, \mu)\, s > 0 \quad \forall s \in \mathbb{R}^n \setminus \{0\} \quad$ with $\quad \nabla h(x)^T s = 0$,*

$\forall x \in \mathbb{R}^n, \mu \in \mathbb{R}^p$ then the matrix (2.79) is invertable.

Afterwards we use these results for convergence analysis of the Newton–method:

Theorem 6 *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differentiable function and $x^* \in \mathbb{R}^n$ a point with $F(x^*) = 0$ in which the Jacobian $F'(x^*)$ is invertable, then $\delta > 0$ and $C > 0$ exist such that*

1. *x^* is the only root of F in $B_\delta(x^*)$,*
2. *$\|F'^{-1}(x)\|_2 \leq C \quad \forall x \in B_\delta(x^*)$,*
3. *The Newton–method terminates for all $x^0 \in B_\delta(x^*)$ either with $x^k = x^*$ or a sequence $\left(x^k\right) \subset B_\delta(x^*)$ which converges superlinear to x^*.*
4. *If F' is Lipschitz–continuous function in $B_\delta(x^*)$ with a Lipschitz–constant L*

$$\|F'(x) - F'(y)\| \leq L\|x - y\| \quad \forall x, y \in B_\delta(x^*), \tag{2.80}$$

then the Newton–method converges quadraticly:

$$\|x^{k+1} - x^*\| \leq \frac{CL}{2}\|x^k - x^*\|^2 \quad \forall k > 0. \tag{2.81}$$

Algorithm 1: Algorithm for solving (2.75)

input : initial value $x^{(0)}$
output: local minimum x^*

for $k = 0, 1, 2, \ldots$ **do**
  if $h\left(x^{(k)}\right) = 0$ *and* $\nabla_x \mathcal{L}\left(x^{(k)}, \mu^{(k)}\right) = 0$ **then**
    STOP and return $\left(x^{(k)}, \mu^{(k)}\right)$;
  end
  Calculate $s^{(k)} = (s_x^{(k)}, s_\mu^{(k)})^T$ by solving (2.78);
  Set $x^{(k+1)} = x^{(k)} + s_x^{(k)}, \mu^{(k+1)} = \mu^{(k)} + s_\mu^{(k)}$;
end

We can reinterpret (2.78) as a quadratic optimizing problem:

$$\min_{d \in \mathbb{R}^n} \nabla f\left(x^{(k)}\right)^T d + \frac{1}{2} d^T H^{(k)} d \quad \text{s.t.} \quad h\left(x^{(k)}\right) + \nabla h\left(x^{(k)}\right)^T d = 0 \tag{2.82}$$

with $H^{(k)} = \nabla^2_{xx}\mathcal{L}\left(x^{(k)}, \mu^{(k)}\right)$.
This problem satisfies (2.71)–(2.74). As a result $\mu_{qp}^{(k)}$ exists with

$$\nabla f\left(x^{(k)}\right) + H^{(k)}d^{(k)} + \nabla h\left(x^{(k)}\right)\mu_{qp}^{(k)} = 0, \quad h\left(x^{(k)}\right) + \nabla h\left(x^{(k)}\right)^T d^{(k)} = 0. \tag{2.83}$$

If we set $s_x^{(k)} = d^{(k)}$ and $s_\mu^{(k)} = \mu_{qp}^{(k)} - \mu^{(k)}$, we get:

$$H^{(k)}s_x^{(k)} + \nabla h\left(x^{(k)}\right)s_\mu^{(k)} = -\nabla f\left(x^{(k)}\right) - \nabla h\left(x^{(k)}\right)\mu^{(k)}, \tag{2.84}$$

$$\nabla h\left(x^{(k)}\right)^T s_x^{(k)} = -h\left(x^{(k)}\right). \tag{2.85}$$

This is a solution of (2.78) if and only if $s^{(k)}$ is a solution of (2.82). We can write down an alternative algorithm for solving (2.75):

Algorithm 2: Alternative algorithm for solving (2.75)

input : initial value $x^{(0)}$
output: local minimum x^*

```
1 for k = 0, 1, 2, ... do
2     if (2.71) is satisfied then
3         STOP and return x* = x^(k);
4     end
5     Calculate a solution of (2.82) and μ_qp^(k);
6     Set x^(k+1) = x^(k) + d^(k), μ^(k+1) = μ_qp^(k);
7 end
```

Motivated by (2.82) we can reformulate the problem (2.66):

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \nabla f\left(x^{(k)}\right)^T d + \frac{1}{2}d^T H^{(k)} d \\ \text{subject to} \quad & g\left(x^{(k)}\right) + \nabla g\left(x^{(k)}\right)^T d \leq 0, \\ & h\left(x^{(k)}\right) + \nabla h\left(x^{(k)}\right)^T d = 0, \end{aligned} \tag{2.86}$$

with $H^{(k)} = \nabla^2_{xx}\mathcal{L}\left(x(k), \lambda^{(k)}, \mu^{(k)}\right)$. For solving (2.66) we modify algorithm 2 lines 5 and 6. As a result we get algorithm 3. This resulting algorithm is one of the most popular algorithms for nonlinear continuous optimization called *sequential quadratic programming*.
For convergence analysis of the SQP algorithm we have the following theorem:

Theorem 7 *Assuming the following preconditions:*

1. *Functions f, g and h are in $\mathcal{C}^2$*
2. $H^{(k)} = \nabla^2_{xx}\mathcal{L}\left(x^{(k)}, \lambda^{(k)}, \mu^{(k)}\right)$
3. (x^*, λ^*, μ^*) *satisfy (2.71)–(2.74)*
4. *Complementary condition:*

$$\forall i \in \{1, \ldots, m\}, \quad g_i(x^*) = 0 \Longrightarrow \lambda_i^* > 0. \tag{2.87}$$

5. $\forall i \in \{i : g_i(x^*) = 0\}$ *the matrix:*

$$(\nabla g_i(x^*), \nabla h(x^*)) \qquad \textit{has full rank in coloumns} \tag{2.88}$$

6. *Sufficient equation of second order:*

$$\begin{aligned} &d^T \nabla^2_{xx}\mathcal{L}(x^*, \lambda^*, \mu^*)\, d > 0 \quad \forall d \neq 0 \quad \textit{with} \\ &\nabla g_i(x^*)^T d = 0 \quad \forall i \in \{i : g_i(x^*) = 0\}, \\ &\nabla h(x^*)^T d = 0. \end{aligned} \tag{2.89}$$

7. *In step (2) of the sequential quadratic programming algorithm we choose the triple $\left(d^{(k)}, \lambda_{qp}^{(k)}, \mu_{qp}^{(k)}\right)$ of 2.86 with minimal distance:*

$$\| \left(x^{(k)} + d^{(k)}, \lambda_{qp}^{(k)}, \mu_{qp}^{(k)}\right) - \left(x^{(k)}, \lambda^{(k)}, \mu^{(k)}\right) \|. \tag{2.90}$$

Then $\delta > 0$ exists such that the SQP algorithm either terminates with $\left(x^{(k)}, \lambda^{(k)}, \mu^{(k)}\right) = (x^*, \lambda^*, \mu^*)$ or it generates a sequence $\left(x^{(k)}, \lambda^{(k)}, \mu^{(k)}\right)$ which converges superlinear to (x^*, λ^*, μ^*) for all $\left(x^{(0)}, \lambda^{(0)}, \mu^{(0)}\right) \in B_\delta(x^*, \lambda^*, \mu^*)$. Furthermore, if $\nabla^2 f$, $\nabla^2 g_i$ and $\nabla^2 h_i$ are Lipschitz continuous in $B_\delta(x^*, \lambda^*, \mu^*)$, then the convergence rate is quadratic.

Algorithm 3: Algorithm for solving (2.66)

input : initial value $x^{(0)}$
output: local minimum x^*

```
for k = 0, 1, 2, ... do
    if (2.71) is satisfied then
        STOP and return x* = x^(k);
    end
    Calculate a solution of (2.86) and multiplicators λ_qp^(k), μ_qp^(k);
    Set x^(k+1) = x^(k) + d^(k), λ^(k+1) = λ_q^(k) p and μ^(k+1) = μ_qp^(k) = μ^(k);
end
```

2.3 Medical Basics

This section gives a short overview of medical imaging devices, respectively interventional angiographic imaging. Furthermore we introduce a model for the blood flow and current methods for estimating hemodynamics.

2.3.1 Interventional Angiographic Imaging

This section follows [Kau96].
When we talk about angiography, also called arteriography, we mean the visualization of vessels, mainly arteries. If we want to display veins or lymphs we talk about venography or lymphography.
In conventional radiographs we can hardly differentiate between soft tissues. For a better view of anatomical structures contrast agent is injected into the region of interest. Most applied contrast agents for angiography consist of iodine or barium. Angiographic imaging is a medical imaging technique used to visualize the inside, or lumen, of blood vessels and organs of the body, with particular interest in the arteries, veins and the heart chambers. This is traditionally done by injecting a radio–opaque contrast agent into the blood vessel and imaging using X–ray based techniques. We can subdivide the imaging modalities according to [Gro08] into

Preoperative Imaging: CTA and MRA Computed Tomography Angiography (CTA) and Magnetic Resonance Angiography (MRA) are primaly used for diagnosis and planning angiographic interventions.

Intraoperative Imaging: C–Arms During an intervention it is not recommendable to move the patient on a new desk. That is the reason why mobile C–Arms are commonly used during an intervention. Compared to CTA and MRA, C–Arms do not take slices. A C–Arm consists primarily of an X–ray source and a flat panel X–ray detector which are on a "C"–shaped device. A C–Arm can take single images that look like normal X–ray images, or it can take an image series while rotating around the region of interest. With this method the 3D volume can be reconstructed (see [Hen08]).

Figure 2.12: Philips MultiDiagnost Eleva. Editing: User Glitzy queen00 in Wikipedia, GNU Free Documentation License

In this thesis we will use the information of projections as well as the 3D information. The major manufacturer of C–Arms are Philips Healthcare[5] (see figure 2.12), Siemens Medical[6] and General Electric.

Actual products have a framerate up to 30 fps/s. The 3D volume, which can be reconstructed is a 12 cm × 12 cm cube in the isocenter.

2.3.2 Blood Flow

One of the central points of this thesis is the model for blood flow. In the following are a few facts for a better understanding of the model [Win02]:
The blood consists of blood cells suspended in a liquid called blood plasma. Plasma, which comprises 55% of blood fluid, is mostly water (90% by volume). The blood cells present in blood are mainly red blood cells (also called erythrocytes) and white blood cells, including leukocytes and platelets. Red blood cells contain hemoglobin, an iron-containing protein, which facilitates transportation of oxygen. Blood is circulated around the body through blood vessels by the pumping action of the heart. In humans, blood is pumped from the strong left ventricle of the heart through arteries to peripheral tissues and returns to the right atrium of the heart through veins.
Blood accounts for 7% of the human body weight, with an average density of ap-

[5]Products can be found here: `http://www.healthcare.philips.com/us/products/interventional_xray/product/interventional_cardiology/index.wpd`

[6]Products can be found here: `http://www.medical.siemens.com/webapp/wcs/stores/servlet/CategoryDisplay~q_catalogId~e_-1~a_categoryId~e_12751~a_catTree~e_100010,1007660,12751~a_langId~e_-1~a_storeId~e_10001.htm`

proximately 1060 $^{kg}/_{m^3}$, very close to pure water's density of 1000 $^{kg}/_{m^3}$. The average adult has a blood volume of roughly 5 liters, composed of plasma and several kinds of cells. The formed elements of the blood are erythrocytes (red blood cells), leukocytes (white blood cells), and thrombocytes (platelets). By volume, the red blood cells constitute about 45% of whole blood, the plasma constitutes about 54.3%, white cells constitute 0.7%.
Whole blood (plasma and cells) exhibits non-Newtonian fluid dynamics. Its flow properties are adapted to flow effectively through tiny capillary blood vessels.

2.3.3 State-of-the-Art in Estimation of Hemodynamics

fMRI Functional magnetic resonance imaging (fMRI) is one actual method for estimation of hemodynamics [Sar07]. The basic principle is that changes in blood flow and blood oxygenation in the brain (collectively known as hemodynamics) are closely linked to neural activity. When nerve cells are active they consume oxygen carried by hemoglobin in red blood cells from local capillaries. The local response to this oxygen utilization is an increase in blood flow to regions of increased neural activity, occurring after a delay of approximately 1–5 seconds. This process, called hemodynamic response, leads to magnetic signal variation which can be detected using an MRI scanner. Given many repetitions of a thought, action or experience, statistical methods can be used to determine the areas of the brain which reliably have more of this difference as a result, and therefore which areas of the brain are active during that thought, action or experience. A sample fMRI scanner and a functional image of the brain can be seen in figure 2.13.

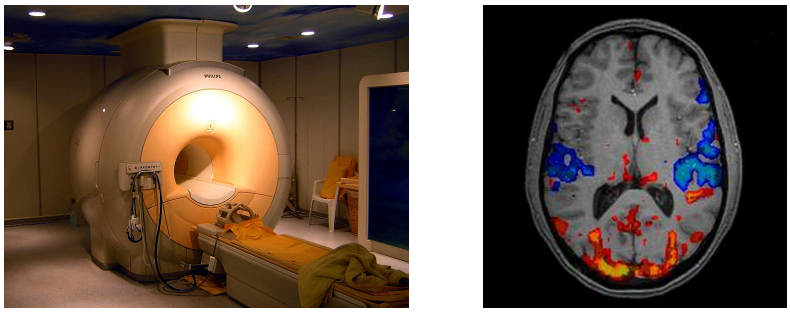


Figure 2.13: Functional magnetic resonance imaging, Photographed by Kasuga Huang, Editing: User Braegel in Wikipedia, GNU Free Documentation License

Advantages:

- It can noninvasively record brain signals (of humans and other animals) without risks of radiation inherent in other scanning methods, such as CT or PET scans.
- It can record on a spatial resolution of less than 1 millimeter, although resolution in the region of 3–6 millimeters is more typical, but with poor temporal resolution (on the order of seconds) compared with techniques such as electroencephalography (EEG). However, this is mainly because of the hemodynamic phenomena being measured, not because of the technique. EEG measures electrical/neural activity while fMRI measures blood activity, which has a slower response. The MRI equipment used for fMRI can be used for high temporal resolution, if one measures different phenomena.

Disadvantages:

- The signal is only an indirect measure of neural activity, and is therefore susceptible to influence by non-neural changes in the body.
- Signals are most strongly associated with the input to a given area rather than with the output. It is therefore possible (although unlikely) that a signal could be present in a given area even if there is no single unit activity.
- Different brain areas may have different hemodynamic responses, which would not be accurately reflected by the simplest version of the general linear model often used to filter fMRI time signals.
- The temporal response of the blood supply, which is the basis of fMRI, is slow relative to the electrical signals that define neuronal communication. To alleviate this problem, some research groups are attempting to combine fMRI signals that have relatively high spatial resolution with signals recorded with other techniques, EEG or magnetoencephalography (MEG), which have higher temporal resolution but worse spatial resolution.
- fMRI has often been used to show activation localized to specific regions, thus minimizing the distributed nature of processing in neural networks. Several recent multivariate statistical techniques work around this issue by characterizing interactions between "active" regions found via traditional univariate techniques.
- fMRI has poor signal–to–noise ratio, at least in comparison to many electrophysiological techniques. This necessitates extensive post-processing and published fMRI results are often heavily averaged over time and smoothed across space using one of several software packages.

For these reasons, Functional imaging provides insights into neural processing.

Doppler Ultrasound The Doppler effect is a change in the frequency of a wave, resulting from motion of the wave source or receiver, or in the case of a reflected wave, motion of the reflector [All06]. In medicine, Doppler ultrasound is used to detect and measure blood flow, and the major reflector is the red blood cell. The Doppler shift is dependent on the insonating frequency, the velocity of moving blood, and the angle between the sound beam and direction of moving blood, as expressed in the Doppler equation:

$$f_d = \frac{2 f_t v \cos\theta}{c}, \tag{2.91}$$

where f_d is the Doppler shift frequency (the difference between transmitted and received frequencies), f_t is the transmitted frequency, v is the blood velocity, c is the speed of sound, and θ is the angle between the sound beam and the direction of moving blood. The equation can be rearranged to solve for blood velocity, and this is the value calculated by the Doppler ultrasound machine:

$$v = \frac{f_d c}{2 f_t \cos\theta}. \tag{2.92}$$

The functionality of the doppler effect and a resulting image can be seen in figure

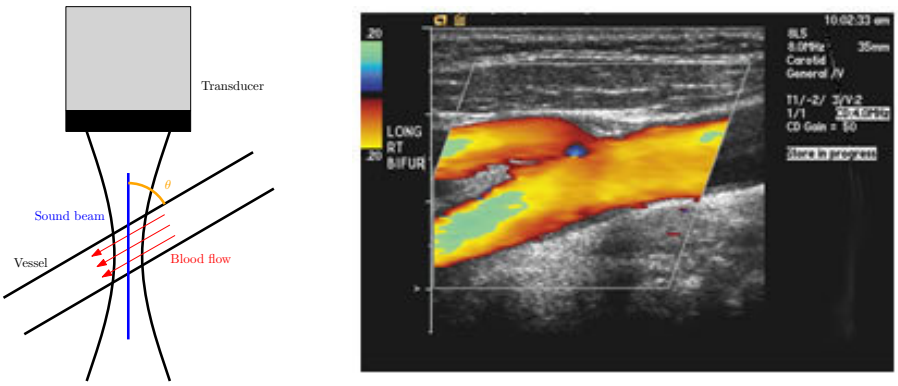


Figure 2.14: Doppler ultrasound

2.14.
Advantages:

- The obvious advantage of Doppler ultrasound is that it is much simpler, more patient-friendly and cheaper than the “gold standard” MRI.
- Doppler ultrasound has the capability to offer spatial information associated to velocity values
- Doppler ultrasound has the advantage of being a gentle, non invasive and easy to perform method with no contamination of radiation, which is an advantage especially on intensive care units with critical ill patients, who are not able to give their permission for the examination.

Disadvantages:

- Unfortunately, as the information is available only periodically, this technique suffers from the Nyquist theorem. This means that a maximum velocity exists for each pulse repetition frequency.
- There is a limitation in depth: The ultrasonic burst travels in the liquid at a velocity which depends on the physical properties of the liquid. The pulse repetition frequency gives the maximum time allowed to the burst to travel to the particle and back to the transducer.
- Ultrasound of the head, (i.e.: brain) can not be done, because the skull prevents transmission of the beam.

3 Blood Flow Estimation

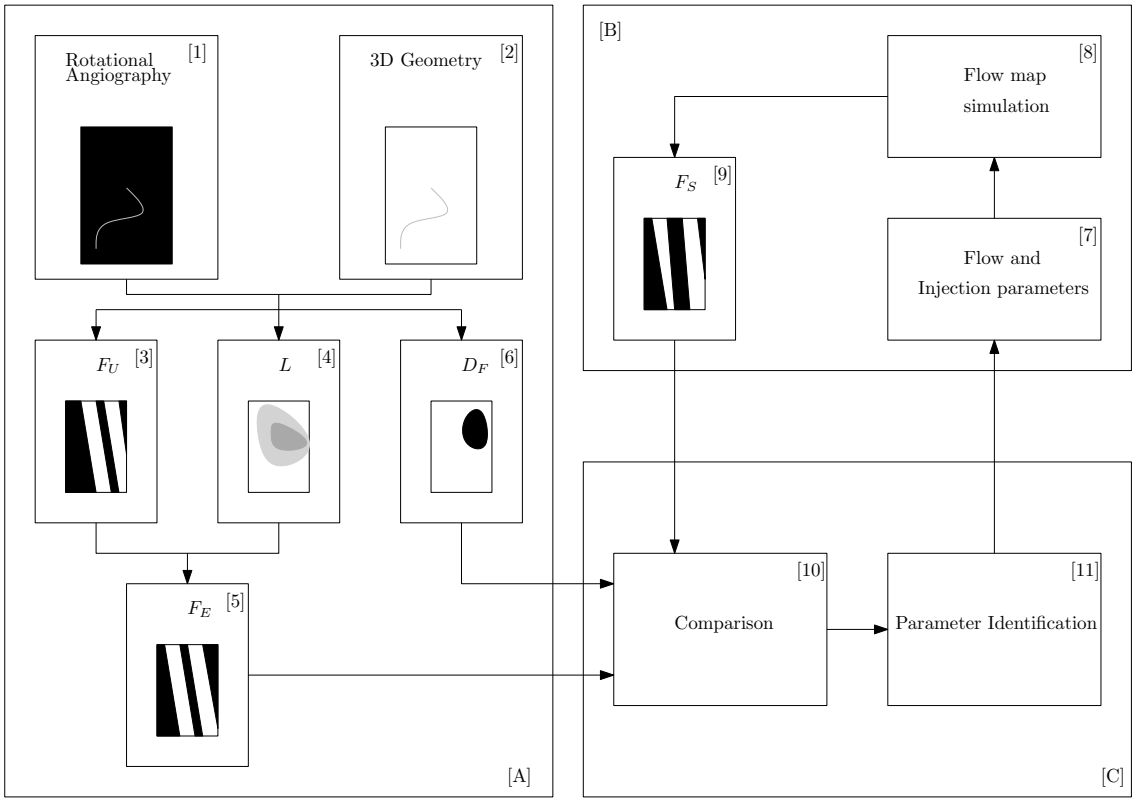


Figure 3.1: Workflow of the Method

In this thesis we address the problem of estimating hemodynamics from rotational angiography. Furthermore we will do a visualization of the estimated data. The workflow of the whole procedure is visualized in figure 3.1. This figure leads us through this chapter and will be explained step by step. First, we discuss how to get information about hemodynamics out of rotational angiography ([A]). Afterwards we have a focus on generating simulation data ([B]). Therefore we develop a simulation method. We will use this simulation method to do a parameter identification ([C]). The final step will be a visualization method of data given by the simulation method.
The first thing that attracts attention in figure 3.1 is the word “flow map”. That is why we give a short definition:

Definition 6 *A flow map is an 2D image. The x direction codes time and the y direction codes the length of the vessel. The intensities show the concentration of contrast agent. High intensities are white, and low intensities are black. In particular a flow map is a mapping:*

$$F := \begin{cases} F : \mathbb{R}^2 \to \mathbb{R}, \\ (l,t)^T \mapsto F(l,t). \end{cases} \tag{3.1}$$

In figure 3.2 two flow maps are visualized. One with a fast volumetric flow and one with a slow volumetric flow. That is to say the blood velocity inside vessel, illustrated by the left flow map, is bigger than in the vessel, illustrated by the right flow map. Of course the question arises why we need flow maps. Here is some

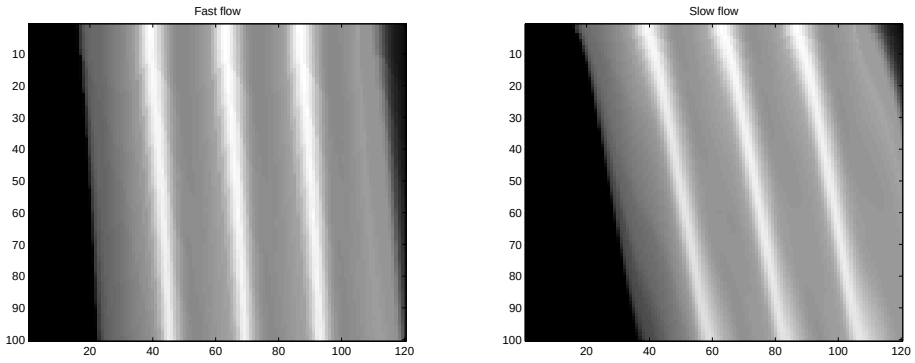


Figure 3.2: Sample flow maps

criteria for using flow maps:

- unlike medical data, flow maps illustrate only the region of interest.
- coherences to blood flow can be estimated easily
- a flow map is a representation that allows easy interpretation and optimization
- the dimension of the problem is reduced

3.1 Extracting Flow Maps

This section is visualized in figure 3.1 [A]. It aims to get a extracted flow map F_E [5] and a reliability map D_F [6]. Therefore the following information is available:

- an X–ray image series $I(x, y, t)$[1] (see fig. 3.1 [1]) taken while the C–arm is rotating around the patient
- a reconstructed 3D volume $V(x, y, z)$ (see fig. 3.1 [2]) respectively the segmented 3D geometry $\mathcal{V}$ of the vessel.
- a set of projection matrices $\mathbf{M}(t) = \left[\ M \ \middle|\ p_4\ \right](t)$ according to eq. (2.13) for each image in the image series $I(x, y, t)$

Most theoretical parts of this section are described in section 2.1.

The first thing we need for the extracted flow map is of course the anatomical structure of the vessel. In this thesis we will not discuss the algorithms which are needed for a centerline–radii representation of the vessel. But we describe the basic concept to get such a representation:
A way to segment blood vessels is described in [Gro08] pp. 73. From the segmented data sets we can extract a centerline image by applying topology–preserving thinning algorithms [Pal01]. A wave propagation algorithm [Zah94] can be used to get a centerline–radii representation of the blood vessel.
A centerline–radii representation of the vessel is a is given by a set of centerline points $p_C(l)$ and a set of corresponding radii $R_C(l)$ for discrete values $l \in \{l_0, l_1, \ldots, l_M\}$. Inversely, the geometry of the vessel can be estimated:

$$\mathcal{V}_e = \left\{ x \in V : x \in B_{R_C(l)}(p_C(l)), \quad l \in \{l_0, \ldots, l_M\} \right\}. \tag{3.2}$$

The index e shall indicate, that the 3D geometry is estimated.

Uncorrected flow map The first step towards the extracted flow map F_E is to extract the uncorrected flow map F_U (see fig. 3.1 [3]). Therefore we need the X–ray image series $I(x, y, t)$, the projection matrices $\mathbf{M}(t)$ and the centerline points $p_C(l)$. The projection matrices $\mathbf{M}(t)$ define the mapping between 3D and 2D data. In particular we can calculate the projected points $p_I(l)$ according to equation (2.13) by:

$$\mathbf{p}_I(l, t) = \mathbf{M}(t) \cdot \mathbf{p}_C(l) \qquad \forall l, t. \tag{3.3}$$

We take the intensity values $I(p_I(l, t), t)$ as entries for the flow map[2]. So the uncorrected flow map $F_U(l, t)$ can be determined as follows

$$F_U(l, t) = I(p_I(l, t), t) \qquad \forall l, t \tag{3.4}$$

[1] x and y are the coordinates of an image, $t \in \{0, \tau, 2\tau, \ldots, T\}$
[2] Note that $p_I(l, t)$ are 2D points

In general the projected centerline points $p_I(l,t)$ don't lie on the grid of the images $I(x,y,t)$. For improvements we use bilinear interpolation of the values $I(p_I(l,t),t)$. We shortly explain the expression "uncorrected" flow map above. The intensity values in the X–ray images show how much the X–ray energy gets attenuated while passing through tissue. This is a line integral according to equation (2.17). This implies for the extracted flow map, that we have to correct the intensities by the length of intersection through the vessel, to get reliable values.

Length map Due to the reason above we have to calculate a so called *length map* $L(l,t)$ (see fig. 3.1 [4]). Therefore we need the segmented vessel $\mathcal{V}$, the projection matrices $\mathbf{M}(t)$ and the centerline points $p_C(l)$.
For an simplified approach to explaining the algorithm we consider $\mathbf{M} \in \mathbf{M}(t)$ and $p_C \in p_C(l)$.
With the help of the projection matrix $\mathbf{M} = \left[\ M \mid p_4\ \right]$ we can calculate the camera origin c in world coordinates according to equation (2.18). Then the normalized direction of the ray d is given by

$$d = \frac{p_C - c}{\|p_C - c\|_2}. \tag{3.5}$$

We use this ray for computation of a rotation axis r. The aim is to rotate the coordinate frame, such that the x_1–axis is in ray direction (see figure 3.3). We

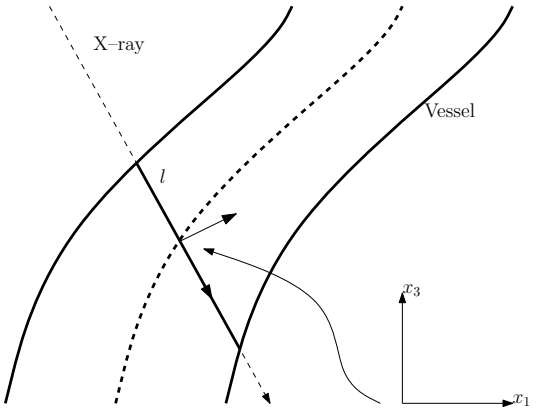


Figure 3.3: Calculating the length of ray intersection

calculate the rotation axis r as follows:

$$r = d \times e_1, \tag{3.6}$$

where e_1 is the normalized basis vector of the x_1–axis. Afterwards we determine the angle ϕ between e_1 and d:

$$\phi = \arccos(d^T e_1). \tag{3.7}$$

With this information we can build up a rotation matrix Q according to equation (2.3). We apply this rotation matrix to all points of the segmented volume $\mathcal{V}$ and define

$$\mathcal{V}_Q := \left\{ y \in \mathbb{R}^3 : y = Qx, x \in \mathcal{V} \right\} \tag{3.8}$$

as the set of rotated vessel points. Finally the length of intersection L is given by

$$L = \max_{p \in \mathcal{V}_l} p_x - \min_{p \in \mathcal{V}_l} p_x, \tag{3.9}$$

$$\mathcal{V}_l = \left\{ p \in \mathcal{V}_Q : |p_{y,z}| \leq \sqrt{3}/2 \right\}, \tag{3.10}$$

where $p = (p_x, p_y, p_z)^T$. We use $\sqrt{3}/2$ as threshold, because this is the maximum diameter of a voxel.
We get the whole *length map* $L(l,t)$, if we calculate the length of intersection L for every combination $\mathbf{M}(t)$, $p_C(l)$ ($t \in \{0, \tau, 2\tau, \ldots, T\}$, $l \in \{l_0, \ldots, l_M\}$).

Extracted flow map The extracted flow map (see fig. 3.1 [5]) is determined by information of the uncorrected extracted flow map $F_U(l,t)$ and the length map $L(l,t)$. We already discussed, that the values of the uncorrected extracted flow map have to be corrected by the length of intersection. Therefore the extracted flow map $F_E(l,t)$ is given by:

$$F_E(l,t) = \frac{F_U(l,t)}{L(l,t)} \qquad \forall l, t. \tag{3.11}$$

Reliability map Problems determing the extracted flow map F_E arise, if two or more centerline points are projected to the same point in the image plane. There are two effects entailing this: foreshortening and overlap. An example for artefacts in the extracted flow map due to vessel overlap can be seen in figure 3.4. Furthermore the corresponding reliability map can be seen in this figure.
To avoid this we introduce a so called *reliability map* D_F (see figure 3.1 [6]), which gives us the reliability of a point in the flow map. The reliability map is defined as follows:

$$D_F(l,t) = \begin{cases} 0, & \text{if foreshortening or overlap is detected} \\ 0, & \text{if indicated by the user} \\ 1, & \text{otherwise.} \end{cases} \tag{3.12}$$

For determing the expression "*if foreshortening or overlap is detected*" we need the projection martices $\mathbf{M}(t)$, the centerline points $p_C(l)$ and the corresponding radii $R_C(l)$. We can calculate the projected points $p_I(l,t)$ just as well as in equation (3.3). Furthermore we calculate the projected radii $R_I(l,t)$ by applying equation (2.22)

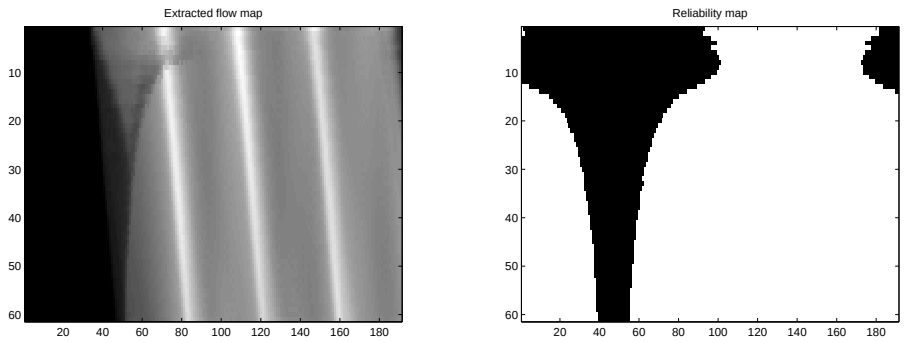


Figure 3.4: Artefacts in a flow map due to overlap of vessel segments and the corresponding reliability map

for every projection matrix $\mathbf{M}(t)$ and every radius $R_C(l)$. Then the expression is replaced by:

$$\begin{aligned} \exists i \quad : \quad & \|p_C(l) - p_C(l_i)\| > a \cdot (R_C(l) + R_C(l_i)) \wedge \\ & \|p_I(l,t) - p_I(l_i,t)\| < b \cdot (R_I(l,t) + R_I(l_i,t)) \qquad \forall l,t,. \end{aligned} \tag{3.13}$$

where a and b are scalar parameters. When $a = b = 1$, this means that overlap or foreshortening is detected when there is another point on the centerline which is further away than the sum of the radii but its projection point is nearer than the sum of the projected radii. In this case, most entries in the flow map would be unreliable. The lower the value of b is, the more entries are marked as reliable. Reasonable choices for a and b were found empirically to be $a \in [1.1, 2]$ and $b \in [0.6, 0.9]$.

3.2 Simulating Flow Maps

This section discusses the model for blood flow, the model for injection and the model for contrast agent propagation. It is visualized in figure 3.1 [B]. These models are needed for generating a simulated flow map F_S. Later on this simulated flow map is going to be compared with the extracted flow map F_E.

3.2.1 Simple Model

In this subsection we discuss a basic vessel segment without bifurcations or huge changes in radius (for example aneurysms).

Model for the blood flow We already discussed some basics about blood flow in section 2.3.2. We got to know, that the pumping action of the heart is the reason for a certain volumetric blood flow. The aim of this diploma–thesis is the estimation of the volumetric blood flow. Therefore we use a parameter dependend model according to [Wae08] for modeling the volumetric blood flow $Q_B(t)$ (see fig. 3.1 [7]) at a certain cross sectional area of a vessel:

$$Q_B(t) = \bar{Q}_B \cdot \omega(t), \tag{3.14}$$

where $\bar{Q}_B$ is the mean volumetric flow and $\omega(t)$ is the waveform.
The model assumes the heartbeat to be periodic. We can generate a periodic function using one of the trigonometric functions like $\cos(x)$. For a better approximation of the waveform $\omega(t)$ we modulate $\cos(x)$ with a piecewise linear function $f_{\beta,\gamma,\delta} : [0,1] \to [0,1]$ defined as follows:

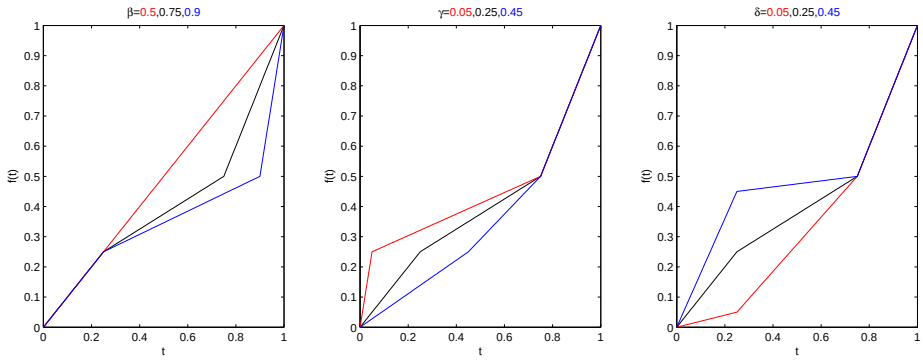


Figure 3.5: Influence of parameters β, γ, δ. The black line shows parameters $\beta = 0.75, \gamma = 0.25, \delta = 0.25$

$$f_{\beta,\gamma,\delta}(t) = \begin{cases} \dfrac{\delta}{\gamma} \cdot t & t \in [0, \gamma] \\ \dfrac{0.5 - \delta}{\beta - \gamma} \cdot t + 0.5 - \dfrac{(0.5 - \delta) \cdot \beta}{\beta - \gamma} & t \in [\gamma, \beta] \\ \dfrac{1}{2 \cdot (1 - \beta)} \cdot t + 1 - \dfrac{1}{2 \cdot (1 - \beta)} & t \in [\beta, 1]\,, \end{cases} \tag{3.15}$$

depending on parameters β, γ and δ.
The so defined piecewise linear function $f_{\beta,\gamma,\delta} : [0,1] \to [0,1]$ has the fixed values $f(0) = 0, f(\gamma) = \delta, f(\beta) = 0.5$ and $f(1) = 1$. The influence of these parameters can be seen in figure 3.5.
Modulating $\cos(x)$ with $f_{\beta,\gamma,\delta}$ and a baseline α generates the following function:

$$\tilde{\omega}(t) = \alpha + \cos(2\pi \cdot f_{\beta,\gamma,\delta}(t)) \tag{3.16}$$

For normalization reasons and achieving a specific periodity we define the waveform $\omega(t)$ as follows:

$$\omega(t) = \frac{\tilde{\omega}(t/T_H)}{\int_0^{T_H} \tilde{\omega}(t)\mathrm{d}t} \tag{3.17}$$

where T_H is the period of heart cycle. The influences of parameters α, β, γ and δ on the volumetric blood flow $Q_B(t)$ (see eq. (3.14)) can be seen in figure 3.6.

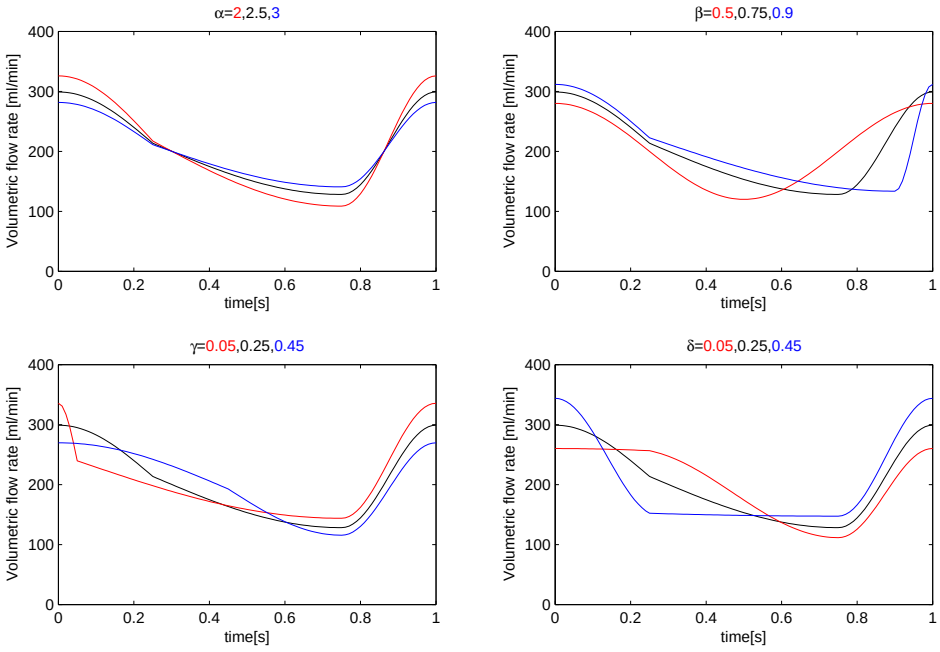


Figure 3.6: Influence of parameters $\alpha, \beta, \gamma, \delta$ on $Q_B(t)$. The black line shows parameters $\alpha = 2.5, \beta = 0.75, \gamma = 0.25, \delta = 0.25$. The mean volumetric blood flow $\bar{Q}_B$ in this example is $200\,{}^{\mathrm{ml}}/_{\mathrm{min}}$.

Model for injection Contrast agent illustrates the vessel in rotational angiography. That is the reason for creating a model (see fig 3.1 [7]) for the contrast agent injection. We do this again according to [Wae08]:
For the contrast agent injection, it is assumed that the injection flow at the outlet of the injector can be described by a rectangular function and that the flow circuit behaves in an analogous way to an electrical network. Then, the catheter is equivalent to a resistor plus a capacitor for the contrast agent. Therefore, the injection

curve corresponds to the charging curve of a capacitor, and the injection flow rate $Q_I(t)$ can be described in terms of its maximum flow rate $\bar{Q}_I$, given by

$$Q_I(t) = \begin{cases} 0, & t < T_S \\ \bar{Q}_I \cdot \left(1 - e^{-(t-T_S)/T_L}\right), & T_S \leq t \leq T_S + T_D \\ \bar{Q}_I \cdot \left(1 - e^{-T_S/T_L}\right) \cdot e^{-(t-(T_S+T_D))/T_L}, & t > T_S + T_D, \end{cases} \tag{3.18}$$

where T_L is the characteristic time of the lag, T_S is the start time of the injection and T_D is the duration of the injection.
As a result of mixing the blood flow rate and the contrast agent flow rate we get the total flow rate:

$$Q_T(t) = Q_B(t) + Q_I(t). \tag{3.19}$$

The total flow rate, the blood flow rate and the injection flow rate are visualized on the left diagram of figure 3.7. For mixing, we have to assume that the contrast agent mixes uniformly with the blood. From X–ray images of the catheter tip, this can be seen to be a reasonable assumption. The contrast agent concentration $C_0(t)$ at the site of injection is then given according to [Haw88]:

$$C_0(t) = \frac{Q_I(t)}{Q_T(t)}. \tag{3.20}$$

This is again visualized in figure 3.7. It is important to note that the concentration

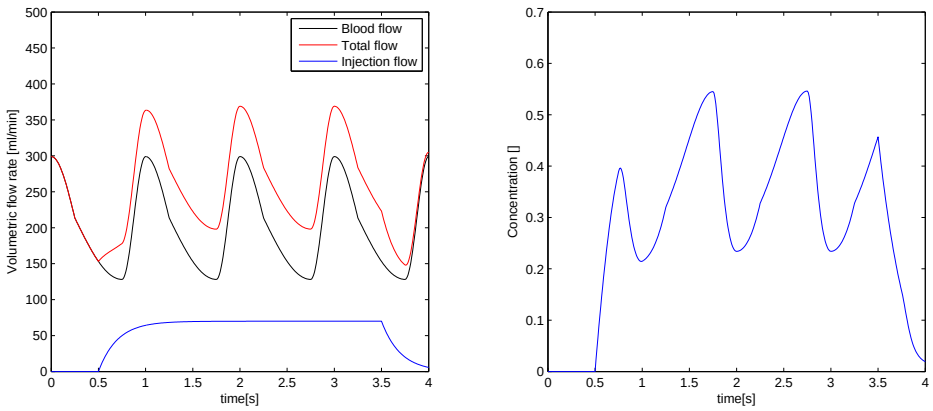


Figure 3.7: Prediction of contrast agent concentration at the injection site

changes during the cardiac cycle. During systole, when the blood flow is highest, the concentration is low. During diastole, when the blood flow is lowest, the concentration is high. This is particularly important because it explains the characteristic pattern that appears in the flow map.

Determing the velocity After discussing models for the volumetric blood flow, the volumetric injection flow and the mixing we have to find a model for contrast agent propagation (see fig. 3.1 [8]). By reason the movement of soluble substance in a moving medium is predetermined by diffusion and advection, we represent the contrast agent propagation with the diffusion–advection equation (2.23).
We reduce the model to be a radial–symmetric lamina flow through a tube. Such a model has already been discussed in [Tay53], where the diffusion–advection equation has been adapted as follows:

$$\begin{aligned}\frac{\partial C(r,l,t)}{\partial t} &= D \cdot \left(\frac{\partial^2 C(r,l,t)}{\partial r^2} + \frac{\partial^2 C(r,l,t)}{\partial l^2} \right) + \\ &\quad + \frac{D}{r} \cdot \frac{\partial C(r,l,t)}{\partial r} - v(r,l,t) \cdot \frac{\partial C(r,l,t)}{\partial l}, \end{aligned} \tag{3.21}$$

with boundary conditions

$$\frac{\partial C(0,l,t)}{\partial r} = \frac{\partial C(R,l,t)}{\partial r} = \frac{\partial C(r,L,t)}{\partial l} = 0, \tag{3.22}$$

$$C(r,0,t) = C_0(t). \tag{3.23}$$

$C_0(t)$ is the concentration, discussed above, at the inlet of the vessel. The boundary conditions in (3.22) are *Neumann* boundary conditions γ_N and equation (3.23) is a *Dirichlet* boundary condition γ_D. We apply this model to describe the contrast

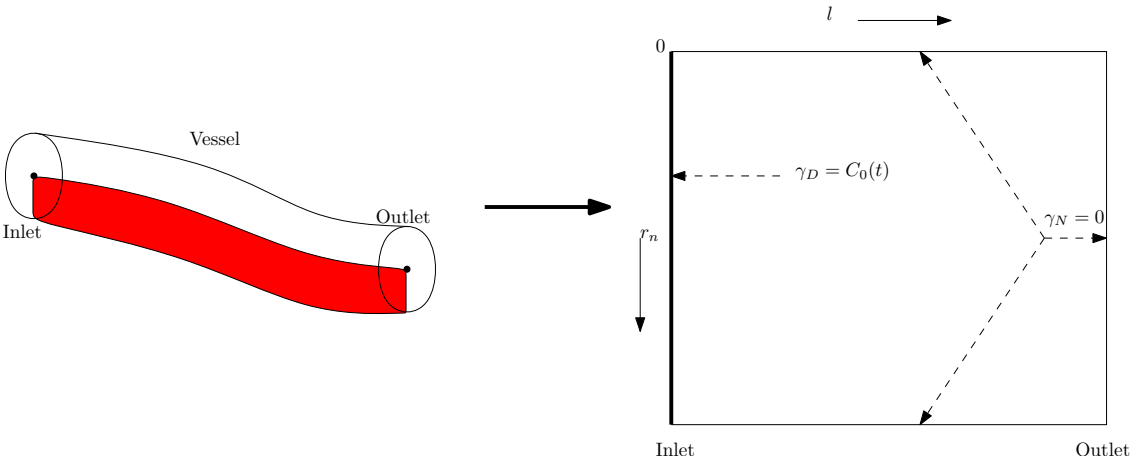


Figure 3.8: Boundary Conditions

agent propagation through the vessel (see figure 3.8).
For solving equation (3.21), using the concept of finite differences in section 2.2.1, we need the longitudinal velocity $v(r,l,t)$ of the blood inside the vessel. Therefore we compute the velocity v, with the help of the cross sectional area A and the total flow rate Q_T:

$$Q_T = v \cdot A. \tag{3.24}$$

But the formula above determines the mean velocity $\bar{v}$ across the cross sectional area A. Due to friction effects the velocity near the boundary is lower than in the isocenter of the vessel. For introducing a function for the flow profile we first divide the vessel into N laminae according to figure 3.9. The middle radius r_n of lamina $n \in \{1, 2, \ldots, N\}$ is given by

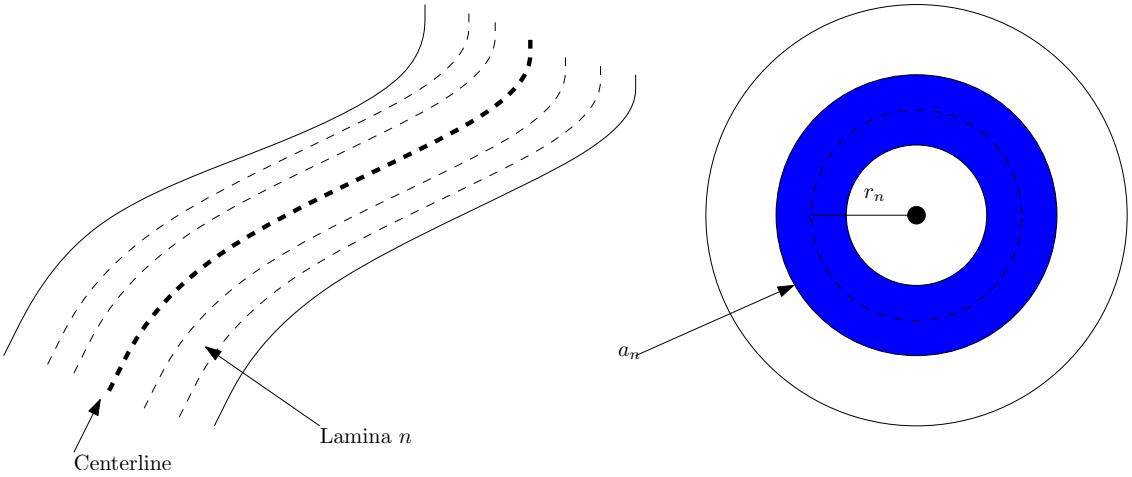


Figure 3.9: Dividing vessel into laminae

$$r_n(l) = \frac{n - 1/2}{N} \cdot R_C(l), \tag{3.25}$$

where $R_C(l)$ is the radius of the vessel at a certain length l. The cross–sectional area of lamina n is then given by

$$a_n(l) = \pi \cdot \left(\frac{n}{N} \cdot R_C(l)\right)^2 - \pi \cdot \left(\frac{n-1}{N} \cdot R_C(l)\right)^2. \tag{3.26}$$

With this information we introduce a function for the flow profile:

$$p(r, l) = 1 - \left(\frac{r}{R_C(l)}\right)^k, \quad 2 \leq k < \infty, \quad r \in \{r_1, r_2, \ldots, r_N\}. \tag{3.27}$$

This function can approximate a profile between parabolic flow ($k = 2$) and a plug flow ($k \to \infty$). Finally we can determine the velocity by

$$v(r, l, t) = \frac{Q_T(t)}{\sum_{n=1}^{N} p(r_n(l)) \cdot a_n(l)} \cdot p(r, l). \tag{3.28}$$

Contrast agent propagation Up to now we introduced several models/methods:

- the volumetric blood flow $Q_B(t)$,
- the volumetric injection flow $Q_I(t)$,
- the mixing of blood and contrast agent $Q_T(t)$ respectively the concentration of contrast agent at the inlet of a vessel segment $C_0(t)$,
- an algorithm to determine the velocity $v(r, l, t)$ inside a vessel
- and a equation for contrast agent propagation (3.21)

We now combine all these elements with the data given by rotational angiography. In fact, we only need the centerline–radii representation of the vessel, i.e. we need the radii $R_C(l), l \in \{l_0, \ldots, l_M\}$ and the set $\{0, \tau, 2\tau, \ldots, T\}$ of times where a X–ray image is taken.
For contrast agent propagation in a vessel we can write down an algorithm as follows:
In step 4 we use the concept of finite differences of section 2.2.1. Important for

Algorithm 4: Contrast agent propagation

input : mean volumetric blood flow $\bar{Q}_B$
input : parameters α, β, γ and δ
input : maximum volumetric injection flow $\bar{Q}_I$
input : chracteristic time of lag T_L
input : start time of injection T_S
input : duration of injection T_D
input : diffusion constant D
input : number of laminae N
input : flow profile k
input : set $\{0, \tau, 2\tau, \ldots, T\}$ of times where a X–ray image is taken
input : radii of the vessel $R_C(l)$ at certain lengths $l \in \{l_0, \ldots, l_M\}$
output: concentration map $C(r, l, t)$

1 determine $Q_B(t)$ for all $t \in \{0, \tau, 2\tau, \ldots, T\}$;
2 calculate $Q_T(t)$ and $C_0(t)$ for all $t \in \{0, \tau, 2\tau, \ldots, T\}$;
3 determine $v(r_n, l, t)$ for all $t \in \{0, \tau, 2\tau, \ldots, T\}, l \in \{l_0, \ldots, l_M\}, n \in \{1, 2, \ldots, N\}$ according to (3.28);
4 calculate the concentration map $C(r_n, l, t)$ for all $t \in \{0, \tau, 2\tau, \ldots, T\}, l \in \{l_0, \ldots, l_M\}, n \in \{1, 2, \ldots, N\}$ by solving (3.21);

implementation is to use the *Upwind schema* for advection along the vessel. Box [8] in figure 3.1 hides this algorithm.

Simulated flow map Finally, the concentration map C is used to determine the simulated flow map F_S (see fig. 3.1 [9]) using the relation:

$$F_S = \rho \cdot \frac{1}{N} \sum_{n=1}^{N} C(r_n, l, t), \tag{3.29}$$

where ρ is the iodine density in the contrast agent. Consequently, the simulated flow map contains the spatial and temporal progression of the mean iodine density in the vessel.

3.2.2 Extension: Vessel tree with variable Radius

Due to the fact that we assumed a vessel segment without bifurcations or huge changes in radius in the last subsection we extend this model, because vessels are highly branched in reality.

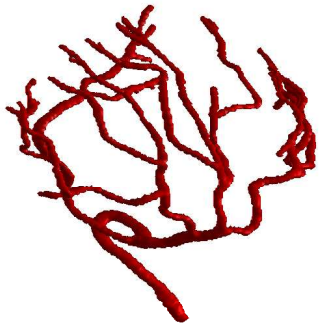


Figure 3.10: Segmented vessel tree in head

Changes in radius In section 3.2.1 we already modeled changes in radius. For small changes in radius we have no numerical problems using finite differences, but for huge changes in radius we get huge changes in blood velocity. This fact can cause unphysical results when using forward/backward differences (see figure 3.11). That is the reason for dividing the vessel into several segments J. For example in figure 3.11 the vessel segment would be divided into three segments (two with a huge radius and one with a tiny radius). As a result J depends on $|\partial R_C/\partial l|$.
For an easy implementation of this approach we solve the diffusion–advection equa-

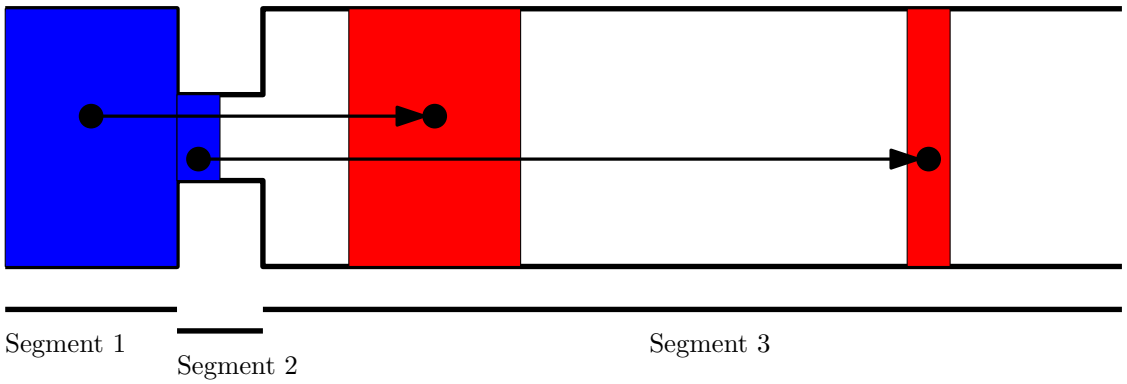


Figure 3.11: Changes in radius

tion for each segment $j \in \{0, \ldots, J\}$. Of course we have to adapt the boundary conditions $C_0^j(r,t)$ for each segment j by:

$$C_0^{j+1}(r,t) = C^j(r, L^j, t), \qquad j \in \{1, \ldots, J\}, \tag{3.30}$$

where L^j is the total length of segment j. After solving 3.21 for each segment, we can reconstruct the concentration map C by

$$C(r,l,t) = \left[C^0(r,l,t), \ldots, C^J(r,l,t)\right]. \tag{3.31}$$

Finally we generate the simulated flow map for the whole segment according to (3.29) .

Bifurcations For modeling a vessel tree, we need the anatomical structure and a applicable way to store it. This can be done by representing the vessel as a directed graph $G^d = \left(V^d, E^d\right)$ where V are the vertices and E the edges. Due to blood vessels have a directed 2–neighborhood, we have to preserve that the extracted graph has this 2–neighborhood, too. How to extract a vessel in such a way is described in [Pal01], too.
Due to the fact that we have a directed 2–neighborhood, each division is a bifurcation i.e. each vessel segment has either a mother or it is the root. In the following we introduce a division factor χ for each vertex V of the graph. An edge refers to a set of radii $R_C(l), l \in \{l_0, \ldots, l_M\}$. But for an simplified approach we discuss a single bifurcation. In particular we consider $Q_T^1(t)$ as the volumetric flow rate in the mother segment and $Q_T^2(t)$, $Q_T^3(t)$ as the volumetric flow rates in the daughter segments. To generate a relationship between these three flow rates we introduce the division factor $\chi \in (0,1)$ as follows:

$$Q_T^2(t) = \chi \cdot Q_T^1(t), \tag{3.32}$$
$$Q_T^3(t) = (1-\chi) \cdot Q_T^1(t). \tag{3.33}$$

Of course, as a result we get three different concentration maps C^i $(i = 1, 2, 3)$. For determination of the concentration maps we need the velocities in each segment. Therefore we need to apply the velocities in the daughter segments $(i = 2, 3)$. This can be done by calculating the velocity, according to equation (3.28) with $Q_T^i(t)$ $(i = 2, 3)$. Afterwards we adapt the boundary conditions according to equation (3.30) for each concentration map:

$$C_0^2(r,t) = C^1(r, L^1, t) \quad \text{and} \quad C_0^3(r,t) = C^1(r, L^1, t), \tag{3.34}$$

where L^1 is the total length of the mother segment.
The extended algorithm for a vessel tree including changes in radius is as follows:

Algorithm 5: Contrast agent propagation for a vessel tree

input : all values of algorithm 4
input : directed graph $G^d = \left(V^d, E^d\right)$
input : for each V a division factor χ
input : for each E the radii of the vessel $R_C(l)$ at certain lengths $l \in \{l_0, \ldots, l_M\}$
output: concentration maps $C^i(r, l, t)$, $i \in \{1, \ldots, J\}$

```
determine Q_B(t) for all t ∈ {0, τ, 2τ, ..., T};
determine Q_I(t) for all t ∈ {0, τ, 2τ, ..., T};
calculate Q_T(t) and C_0(t) for all t ∈ {0, τ, 2τ, ..., T};
for each E do
    determine Q_T^i(t), v^i(r_n, l, t) and C_0^i;
    divide the actual segment into several subsegments j, if |∂R_C/∂l| > ζ;
    for every subsegment j do
        determine C_0^{i,j};
        calculate the concentration map C^{i,j};
    end
    assemble the concentration map C^i;
end
```

Finally we generate a flow map for every segment according to equation (3.29):

$$F_S^i = \rho \cdot \frac{1}{N} \sum_{n=1}^{N} C^i(r_n, l, t) \qquad i = 1, 2, 3. \tag{3.35}$$

3.3 Parameter Identification

In the previous sections we got to know how we can get information out of rotational angiography and how we can model the blood flow respectively the contrast agent

propagation. The next step is of course to get these two things together and do a parameter identification (see figure 3.1, box [C]). Here is a short summary of the parameters we want to estimate:

t_S	start time of injection
t_D	duration of injection
t_l	characteristic time of lag
$\bar{Q}_B$	mean volumetric flow rate
$\alpha, \beta, \gamma, \delta$	parameters for generating the waveform function $\omega(t)$
k	flow profile
χ	division factor for branches

The main problem is of course that we want to estimate at least 9 parameters which take part in a partial differential equation in terms of a new boundary condition γ_D (see figure 3.8). We use the model of flow maps and compare them in a specific way. Therefore we need an error–function. Kinematic information is captured by the gradients of the pattern of the flow map, whereas densitometric information is captured by the magnitude of the intensities of the flow map. That is why the choice for the error–function is:

$$E\left(F_S\right) = \sum_{j=1}^{J} \sum_{l=l_j}^{L_j} \sum_{t=t_0}^{T} \left[\left(F_S^j\left(l,t\right) - F_E^j\left(l,t\right) \right)^2 + \left(\nabla F_S^j\left(l,t\right) - \nabla F_E^j\left(l,t\right) \right)^2 \right] \cdot D_F^j\left(l,t\right). \tag{3.36}$$

where T is the total time of acquisition, J is the number of segments, F_E is the extracted flow map (see equation (3.11)), F_S is the simulated flow map (see equation (3.29)) and D_F is the reliability map (see equation (3.12)). This error–function depends on all parameters $t_S, t_D, t_l, \bar{Q}_B, \alpha, \beta, \gamma, \delta, k, \chi$, which appear – as already told – as boundary condition in the diffusion–advection equation. Using this error function we formulate the optimizing problem:

$$\min_{x \in \Omega} E(x) \tag{3.37}$$

with $x = \left(t_S, t_D, t_l, \bar{Q}_B, \alpha, \beta, \gamma, \delta, k, \chi\right)^T$ and Ω is given by an empirical generated table:

Variable	lower bound	upper bound	unit
t_S	$t_{S,0}$	$t_{S,e}$	s
t_D	$t_{D,0}$	$t_{D,e}$	s
t_l	$t_{l,0}$	$t_{l,e}$	s
$\bar{Q}_B$	$\bar{Q}_{b,0}$	$\bar{Q}_{b,e}$	$^{\text{ml}}/_{\text{min}}$
α	1.2	5	
β	0.5	0.9	
γ	0.01	0.49	
δ	0.01	0.49	
k	2	10	
χ	0.01	0.99	

This problem can be solved using the SQP–algorithm.
Due to tiny gradients along variable χ we can not use the SQP–algorithm in this dimension. The best results in a complete vessel tree were achieved by optimizing all parameters, except χ, in the root segment and each χ with the downhill–simplex algorithm. Another justification for this approach is that commonly the root segment has biggest diameter. Consequently this segment contains most hemodynamic information.
Finally the algorithm for parameter estimation is given by:

Algorithm 6: Parameter Identification

input : Extracted Flow maps F_E^i
input : Initial values for the parameters (see table above)
input : all other values needed for algorithm 5
input : boundaries according to the table above
output: estimated parameters

1 $\left[t_S, t_D, t_l, \bar{Q}_B, \alpha, \beta, \gamma, \delta, k\right]$ =SQP($E^1(x)$,initial values,boundaries);
2 **for** *each branch i following* **do**
3 determine parameter χ_i with the downhill simplex algorithm;
4 **end**

3.4 Visualization

An interesting application is the visualization of the estimated data. Surgeons could do a blood flow simulation without injecting contrast agent. Due to the fact that we can estimate the blood flow we can try to visualize this data, too.
The basic idea of the visualization method is to generate a background image I_B

and a foreground image I_F. The goal is to calculate an image series using the formula:

$$I_V(t) = I_B + I_F(t), \qquad t \in \{0, \tau, 2\tau, \ldots, T\}, \tag{3.38}$$

where τ are the time steps where we have data in the concentration map C.
In particular, we visualize the blood flow with the concept of digital reconstructed radiograph (DRR) (see equation (2.20)).
Therefore we set all intensities of the vessel to zero and compute a DRR of this 3D volume. This is the image I_B.
We shortly explain the concept of digital radiograph reconstruction (DRR)? DRR is a possibility to simulate X–ray images. This means we use equation (2.19) and do a ray sampling of the volume with a given projection matrix $P = \left[\ M \mid p_4\ \right]$.
Here is the algorithm how to create DRRs:
Given: $M(t)$ Projection matrices, the 3D volume, centerline points $p_C(l)$ and radii $R_C(l)$ for each centerline point.

1. Do a simulation of contrast agent propagation according to section 3.2 with $\#M(t)$ time steps.
2. For each projection matrix, fill the concentration data from step one into the vessel, and create a X–ray image.

In a second step we compute the contribution of each voxel of the vessel to the resulting image. We do this again with the concept of DRR. The current ray is given by equation (2.19). We define a list of voxels as follows:

$$X(x^{2D}, \lambda) = \begin{cases} 1, & \text{if ray } \mathbf{x}(\lambda) \text{ is inside the vessel} \\ 0, & \text{otherwise} \end{cases} \tag{3.39}$$

x^{2D} is a arbitrary point in the resulting image I_F and $x(\lambda)$ is the ray corresponding to x^{2D}.
If a ray at a certain value λ is inside a voxel that corresponds to a vessel segment ($X(x^{2D}, \lambda) = 1$), we compute its contribution $Y(x^{2D}, \lambda)$ to the resulting image using trilinear interpolation of $V(\mathbf{x}(\lambda))$[3]. Therefore we set the value of the current voxel to one and all others to zero.
The third step is a lookup in the concentration map C. Additionally we fill in the data as accurate as possible. This step should not be underrated. Due to the fact that we solve the diffusion–advection equation with a radial symmetric assumption and without any concern of vessel geometry, we have to find the voxels corresponding to a certain length l and radius r. This can be done by

$$\begin{pmatrix} l(p_V) \\ r(p_V) \end{pmatrix} := \begin{pmatrix} \arg\min\limits_{p \in p_C(l)} \|p_V - p\| \\ \min\limits_{p \in p_C(l)} \|p_V - p\| \end{pmatrix} \tag{3.40}$$

[3] V is the 3D volume

where p_V is an arbitrary voxel of the vessel and $p_C(l)$, $l \in \{l_0, \ldots, l_M\}$ is the set of centerline points. We can estimate the value of the contribution from the concentration map using the following formula:

$$Z\left(p_V, t\right) = \frac{\int_{r(p_V-1/2)}^{r(p_V+1/2)} C(r, l(p_V), t)\mathrm{d}r}{r(p_V + 1/2) - r(p_V - 1/2)}. \tag{3.41}$$

Of course, this formula can only be applied if $r(p_V)$ is a value between $[1, N-1]$ (N is the number of laminae, see figure 3.12). For values of $p_V \in [0, 1) \cup (N-1, N]$ we have to reflect at the border. Finally the image $I_F(t)$ is given by

$$I_F(x^{2D}, t) = \sum_{\lambda} X(x^{2D}, \lambda) \cdot Y(x^{2D}, \lambda) \cdot Z(\mathbf{x}(\lambda), t). \tag{3.42}$$

The profit of visualizing this way is that we can precompute $\mathbf{x}(\lambda), X(x^{2D}, \lambda)$ and

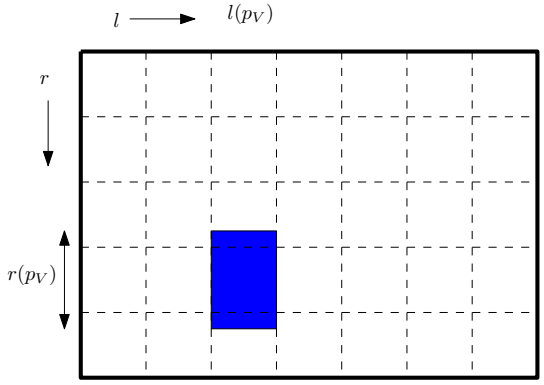


Figure 3.12: Contribution of the concentration map

$Y(x^{2D}, \lambda)$. For a numerical approximation of $Z(\mathbf{x}(\lambda), t)$ it is sufficient to store the coefficients c_i and nodes x_i given by a quadrature formula for an arbitrary t. Finally we replace the concentrations in (3.41) in the sum of the quadrature

$$\int_{r(p_V-1/2)}^{r(p_V+1/2)} C(r, l(p_V), t)\mathrm{d}r \approx \sum_i c_i C(x_i, l(p_V), t) \tag{3.43}$$

for every time step t.

Algorithm 7: Visualization

input : projection matrix $\mathbf{M} = \left[\ M \mid p_4\ \right]$
input : concentration map C
input : original volume V
input : set of points of the segmented vessel $\mathcal{V}$
input : set of centerline points $p_C(l)$
output: image series I_V

$V_{bg} = V$;
for *each* $p_V \in \mathcal{V}$ **do**
    $V_{bg}(p_V) = 0$;
    calculate $l(p_V)$ and $r(p_V)$ according to (3.40);
    calculate c_i, x_i for a numerical approximation of $Z\left(p_V, \cdot\right)$;
end
$I_B = \mathrm{drr}(V_{bg}, \mathbf{M})$;
for *each* $x^{2D} \in I_F$ **do**
    calculate $\mathbf{x}(\lambda)$ according to eq. (2.19);
    for *each* μ *inside* V **do**
        calculate $X(x^{2D}, \lambda)$;
        calculate $Y(x^{2D}, \lambda)$;
    end
end
for *each* $t \in \{0, \tau, 2\tau, \ldots, T\}$ **do**
    for *each* $p \in X(x^{2D}, \lambda) \neq 0$ **do**
        get corresponding $l(p)$, c_i, x_i;
        $I_F(p,t) = \sum_i c_i C(x_i, l(p), t)$;
    end
    $I_V(t) = I_B + I_F(t)$
end

4 Validation

In this chapter we discuss a validation of the model of the previous chapter. We do this theoretically in two steps. First we create a sample flow map on which we try to optimize using the SQP–algorithm. Secondly we do some evaluations with generated volumes. Practical results using real data with a solid ground truth of the method can be found in [Wae08].

4.1 Simulated Flow Maps

For a first step of validation we generate a single simulated flow map. We estimate the parameters using the SQP–algorithm. We use the following parameters for generating the flow map:

	Symbol	Value	Unit
Acquisition			
Start time	t_0	0	s
End time	T	4	s
Number of time steps	M	120	
Injection			
Volumetric flow rate of contrast agent	Q_I	70	$^{\text{ml}}/_{\text{min}}$
Start time of injection	t_S	0.5	s
Duration of injection	t_D	3	s
Characteristic time of lag	t_l	0.2	s
Iodine density	ρ	1	$^{\text{mg}}/_{\text{ml}}$
Molecular diffusion coefficient	D	10^{-5}	$^{m^2}/_{\text{s}}$
Blood flow			
Mean volumetric blood flow	$\bar{Q}_B$	200	$^{\text{ml}}/_{\text{min}}$
Shape parameter	α	2.5	
Shape parameter	β	0.6	
Shape parameter	γ	0.4	
Shape parameter	δ	0.2	
Duration of cardiac cycle	t_H	0.8	s
Flow profile	k	5	
Algorithm			
Number of laminae	N	10	

We consider a straight vessel with constant radius $R_D(l) = 2\,\text{mm}$, total length $L = 10\,\text{mm}$. Furthermore we assume 100 centerline points.
In figure 4.1 we can see the flow map, resulting from these parameters. For an analysis of the target function $E(x)$, figure 4.2 shows the course of the target function depending on the parameters. These plots were created by setting all parameters to the optimum $x^* = (t_S^*, t_D^*, t_l^*, \bar{Q}_b^*, \alpha^*, \beta^*, \gamma^*, \delta^*, k^*)^T = (0.5, 3, 0.2, 200, 2.5, 0.6, 0.4, 0.2, 5)^T$. Afterwards a single parameter is varied from the lower bound to the upper bound, which is given by the following table:

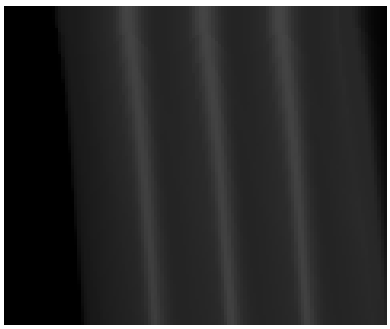


Figure 4.1: Sample flow map for optimization

Variable	lower bound	upper bound	unit
t_S	-1	1	s
t_D	1	4	s
t_l	0.1	0.7	s
$\bar{Q}_B$	50	400	$^{\text{ml}}/_{\text{min}}$
α	1.2	5	
β	0.5	0.9	
γ	0.01	0.49	
δ	0.01	0.49	
k	2	10	
χ	0.01	0.99	

We can see clearly that all functions have a single optimum, where the derivative is zero, except t_D. This fact can easily be explained: t_D is the duration of the injection. Due to the fact that the total acquisition time is 4 seconds and the start time of injection t_S is at time 0.5 s, the resulting total time which we can estimate is 3.5 s. All greater durations can not be estimated.
With this knowledge we try to estimate the parameters using the SQP–algorithm. We quantify the solution depending on initial values. Therefore we choose 100 random initial values within the given range. For every initial value we calculate the solution of the problem with a tolerance of 10^{-2}. The results can be seen in figure 4.3, where the error was calculated by the following formula:

$$\text{Error} = \frac{x^* - \text{optimized } x}{\text{upper bound} - \text{lower bound}}. \tag{4.1}$$

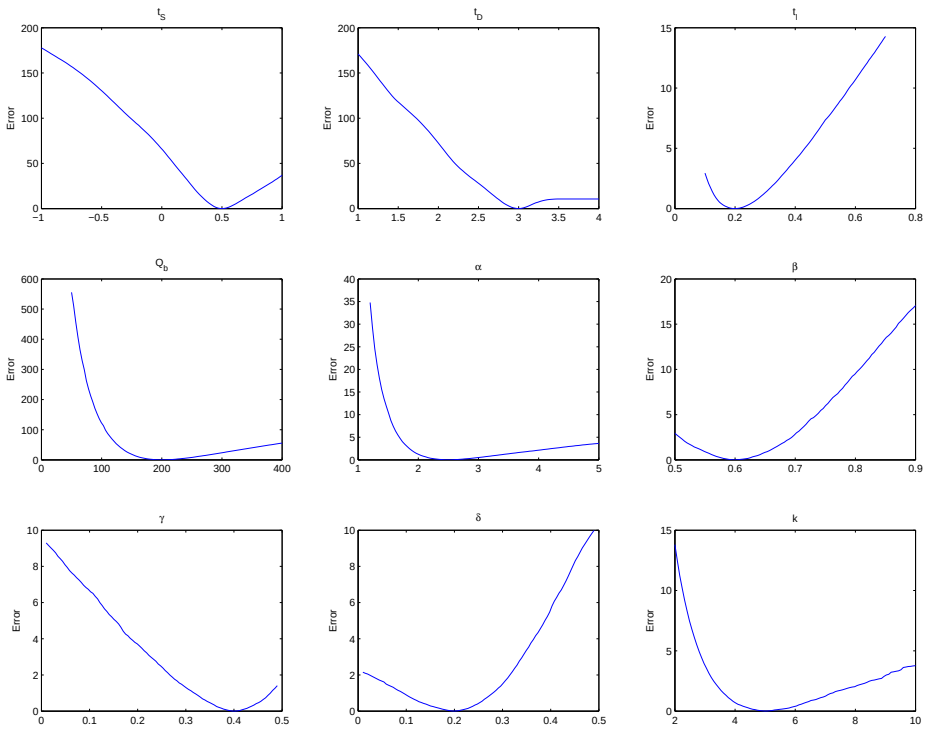


Figure 4.2: Course of the function $E(x)$ depending on parameters

The blue box illustrates the range containing 50% of the data. The horizontal black lines illustrate the range containing 90% or more of the data. The red line is the mean value and the red crosses are outliers. The left diagram in figure 4.3 demonstrates an optimization on all parameters, whereas the right diagram sets the parameter k to the optimum and optimizes all other parameters. The results

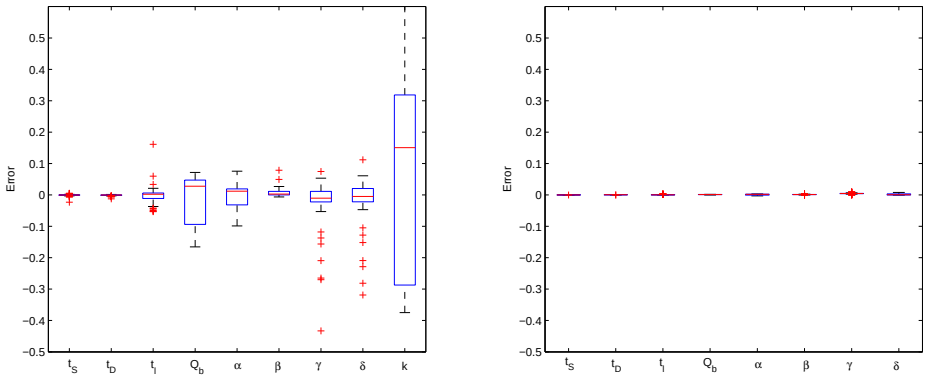


Figure 4.3: Relative error of the estimated parameters depending on initial values

indicate clearly that the parameters t_S and t_D can be estimated very accurate. All other parameters are not within the given tolerance of 10^{-2}, if we include parameter k in the optimization. We can give a reason for this fact. Parameter k describes the shape of the flow profile in the vessel. This flow parameter is not obvious visible in the flow map, because the flow map represents only the mean concentration of a certain cross–sectional area.
We can also see that an optimization without parameter k gives very accurate results.

Furthermore we want to quantify the robustness of the method against noise. For a first approach we apply Poisson noise with mean 100 to the original flow map (see figure 4.4).
In figure 4.5 the course of the error function $E(x)$ depending on parameters is visu-

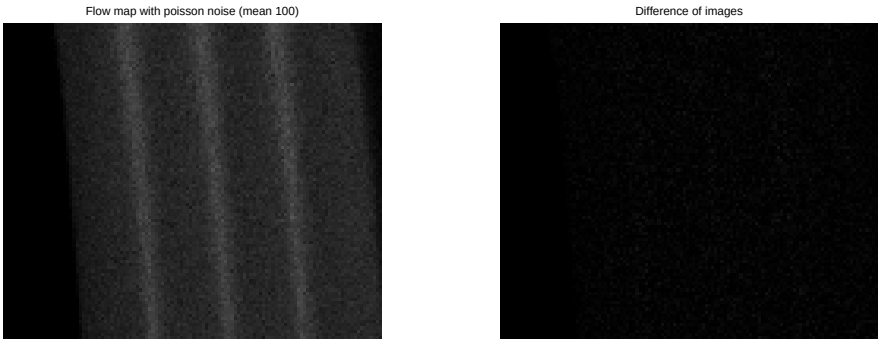


Figure 4.4: Simulated flow map with Poisson noise (mean 100) and difference to original flow map

alized. Again the parameters are fixed at $x^* = (0.5, 3, 0.2, 200, 2.5, 0.6, 0.4, 0.2, 5)^T$ and one parameter varies from the lower bound to the upper bound. The circles in figure 4.5 illustrate the minimum of the function and the red crosses illustrate the relative error at $E(x^*)$.
Additionally we use the SQP–algorithm for a parameter estimation. The dependence of initial values on the relative error is visualized in figure 4.6. 25 different noisy flow maps with mean 100 were optimized. Each with 25 different random initial values. Without k means k set to optimum. The "bad" influence of parameter k is again visible.

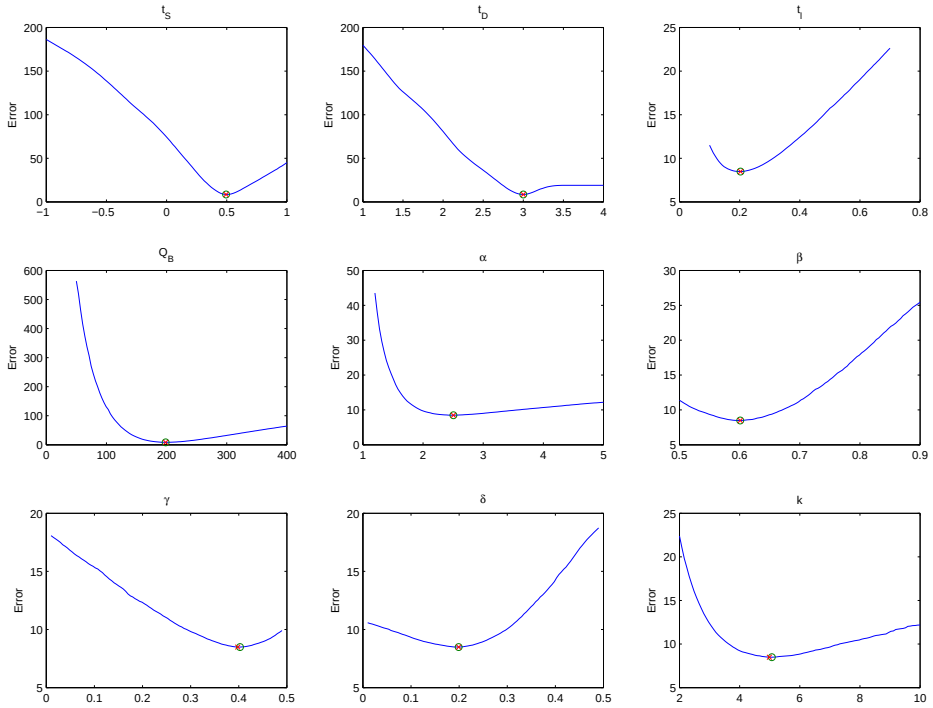


Figure 4.5: Course of the function $E(x)$ depending on parameters extracted form simulated flow map with Poisson noise (mean 100).

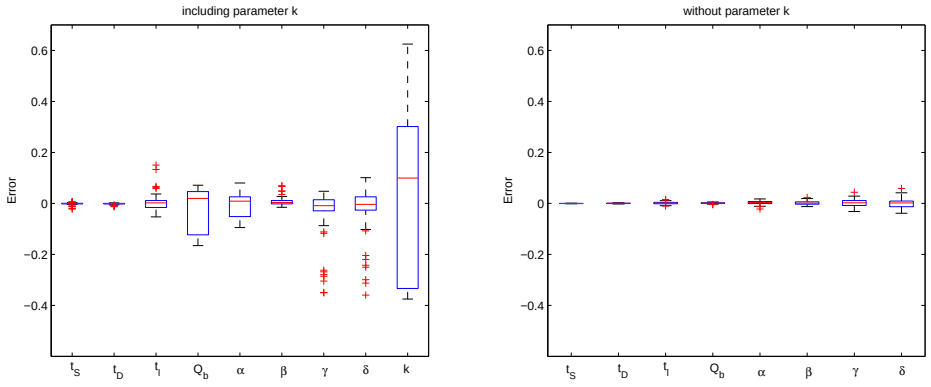


Figure 4.6: Relative error of the estimated parameters depending on initial values. To the simulated flow map Poisson noise with mean 100 was added.

Secondly we apply Poisson noise with mean 10 to the original flow map (see figure 4.4. In figure 4.8 we see the course of the error function $E(x)$ depending on para-

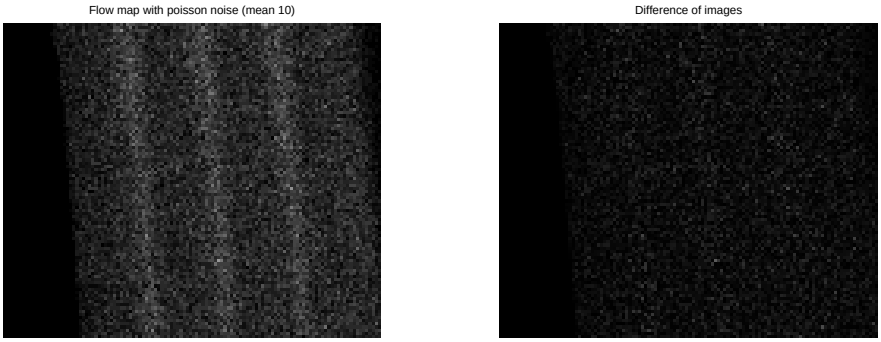


Figure 4.7: Simulated flow map with Poisson noise (mean 10) and difference to original flow map

meters. Again the parameters are fixed at $x^* = (0.5, 3, 0.2, 200, 2.5, 0.6, 0.4, 0.2, 5)^T$ and one parameter varies from the lower bound to the upper bound. The circles in figure 4.8 illustrate the minimum of the function and the red crosses illustrate the relative error error $E(x^*)$. Additionally we use the SQP–algorithm for a parameter estimation. The dependence of initial values on the relative error is visualized in figure 4.9. 100 different noisy flow maps with mean 10 were optimized with 100 different random initial values.

Finally we apply Poisson noise with mean 1 to the original flow map (see figure 4.10. In figure 4.11 we see the course of the error function $E(x)$ depending on parameters. Again the parameters are fixed at $x^* = (0.5, 3, 0.2, 200, 2.5, 0.6, 0.4, 0.2, 5)^T$ and one parameter varies from the lower bound to the upper bound. Additionally we use the SQP–algorithm for a parameter estimation. The dependence of initial values on the relative error is visualized in figure 4.12. 100 different noisy flow maps with mean 1 were optimized with 100 different random initial values. If we compare figures 4.6, 4.9 and 4.12 with figure 4.3, we can see the robustness of the method to noise.

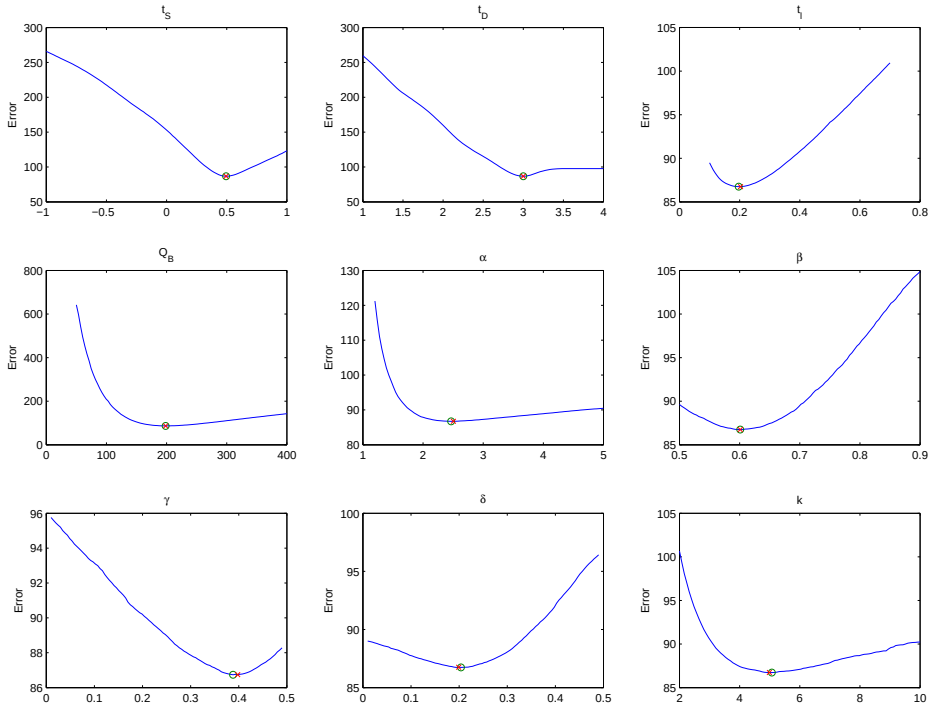


Figure 4.8: Course of the function $E(x)$ depending on parameters extracted form simulated flow map with Poisson noise (mean 10).

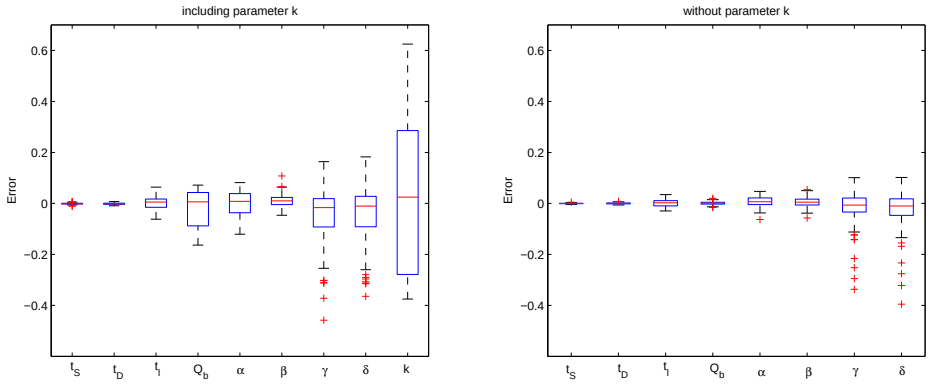


Figure 4.9: Relative error of the estimated parameters depending on initial values. To the simulated flow map Poisson noise with mean 10 was added.

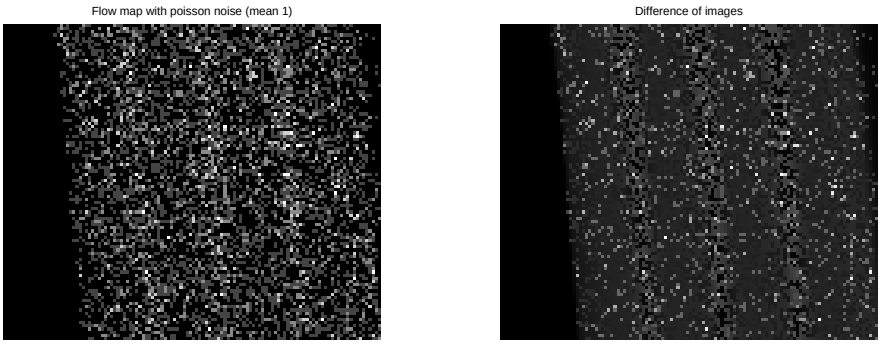


Figure 4.10: Simulated flow map with Poisson noise (mean 1) and difference to original flow map

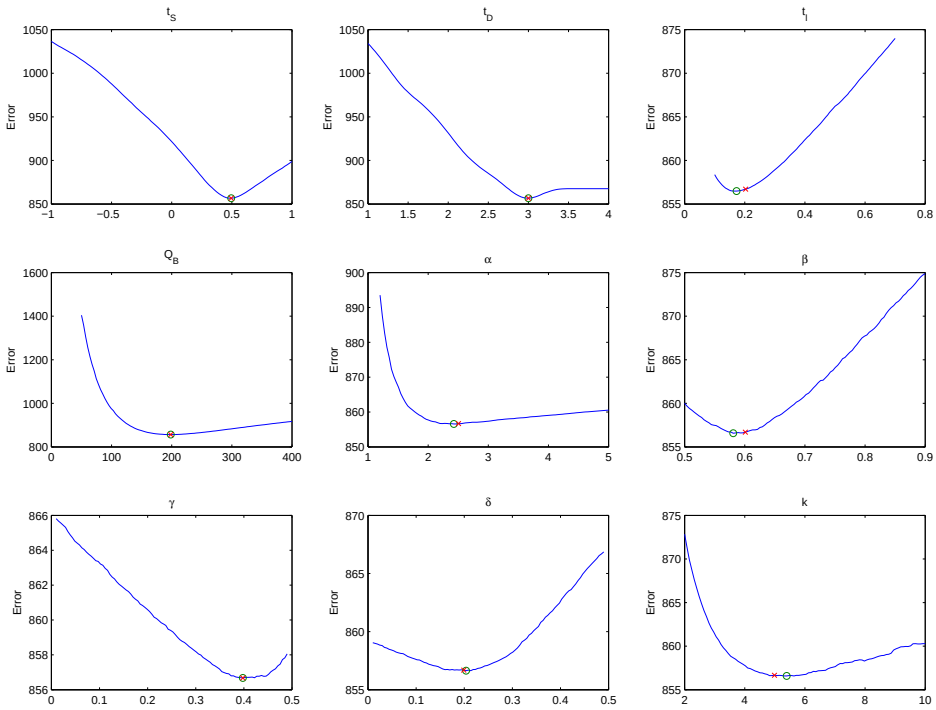


Figure 4.11: Course of the function $E(x)$ depending on parameters extracted form simulated flow map with Poisson noise (mean 1).

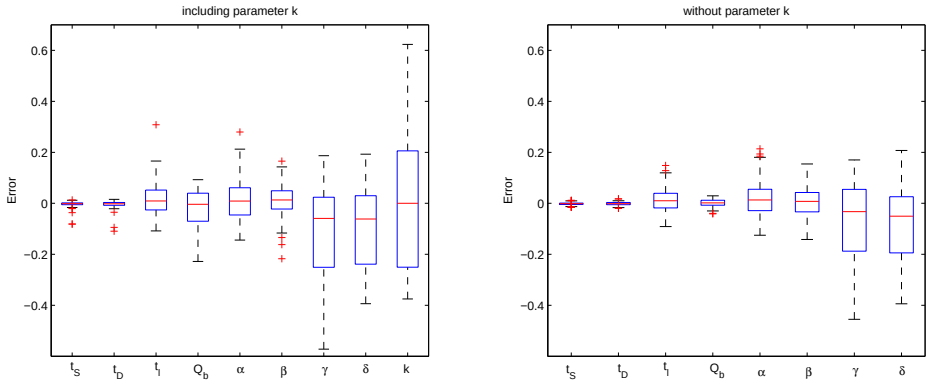


Figure 4.12: Relative error of the estimated parameters depending on initial values. To the simulated flow map Poisson noise with mean 1 was added.

4.2 Flow Maps extracted from DRRs

The simulation for all following examples was done with the parameters of the previous section.

4.2.1 Straight Tube

Figure 4.13: Straight tube

First we use a simple straight tube with a constant radius of 5 voxel and length of 100 voxel (see figure 4.13). The extracting algorithms returned 98 centerline points and corresponding radii around 5. The extracted flow map F_E and the length map L using this data can be seen in figure 4.14. The reliability map R is not illustrated, because we do not have the effects of foreshortening or overlap in a straight tube. Due to discretization effects the values of th length map L in figure 4.14 change.

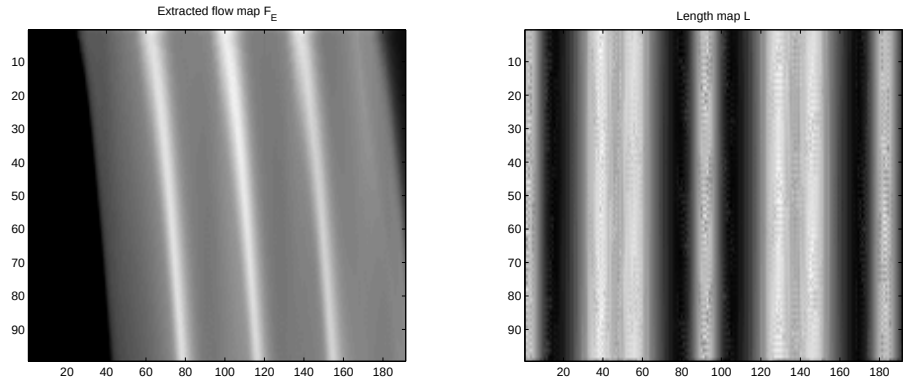


Figure 4.14: Extracted flow map and length map of the straight tube

The mean value is of course 10. To visualize the error depending on initial values, 625 random initial values were generated for the SQP–algorithm. The same was done by optimizing without parameter k. The results can be seen in figure 4.15. The error for t_S and t_D is in both cases (including / without parameter k) are with in the given tolerance of 10^{-2}. If we include parameter k, we can see that the mean relative error is around 7%. An optimization without parameter k gave accurate results. The mean relative error is around 5%. Figure 4.15 illustrates also that the variance of the mean error is much less if we do an optimization without parameter k.

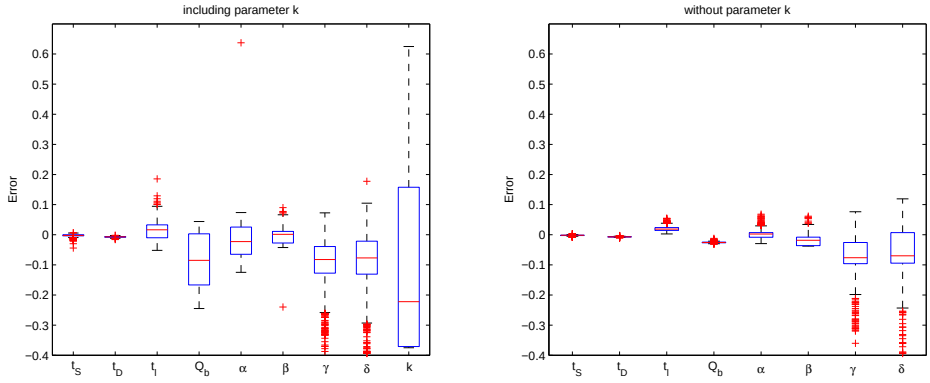


Figure 4.15: Relative error depending on initial values

Again we want to estimate the robustness of the method to noise. For that we put a Poisson noise with mean 10 on the image–series $I(x, y, t)$, that is to say the X–ray images. After that we extract the flow map F_E out of this image series. An example for an extracted flow map is visualized in figure 4.16. The length map

and the reliability map are of course the same as without noise. We generated 25

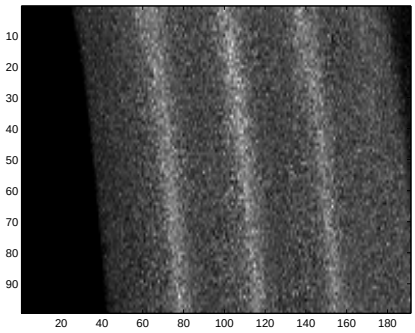


Figure 4.16: Extracted flow map from noisy image series

different image–series $I(x, y, t)$, and for each image series 25 random initial values to visualize the dependence on initial values. The results are shown in figure 4.17. The astonishing is, that this figure looks very similar to figure 4.15, which was

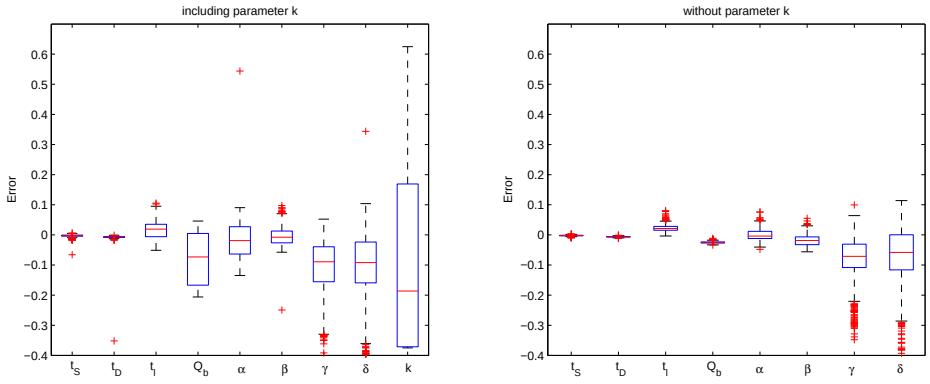


Figure 4.17: Relative error depending on initial values

created without noise. For this we can conclude, that the method is very robust to noise.

4.2.2 Curved Tube

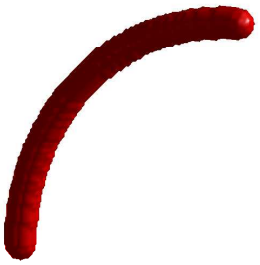


Figure 4.18: Curved tube

For a validation of curved tubes, a curved geometry was generated (see figure 4.18). The extraction algorithms returned 124 centerline points. The radius of this tube is again 5 voxels. In figure 4.19 the extracted flow map, the length map and the reliability map are illustrated. The white dots in the length map indicate an overlapping in the projection. These regions are marked as unreliable in the reliability map.
We optimize the extracted flow map with the SQP–algorithm. For validation 100

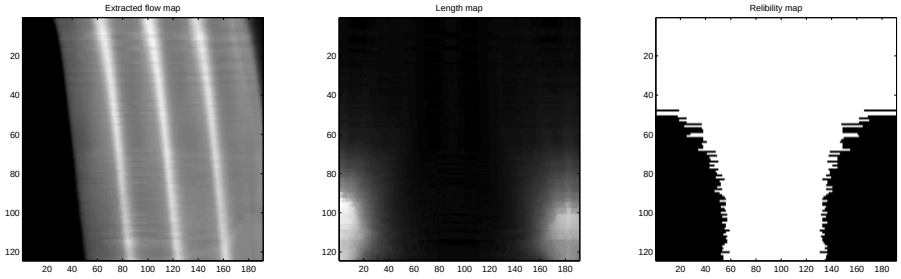


Figure 4.19: Extracted flow map, length map, reliability map of curved tube

random initial values were chosen the results can be seen in figure 4.20 and figure 4.21. In figure 4.21 Poisson noise with mean 10 was applied to the image–series $I(x, y, t)$.

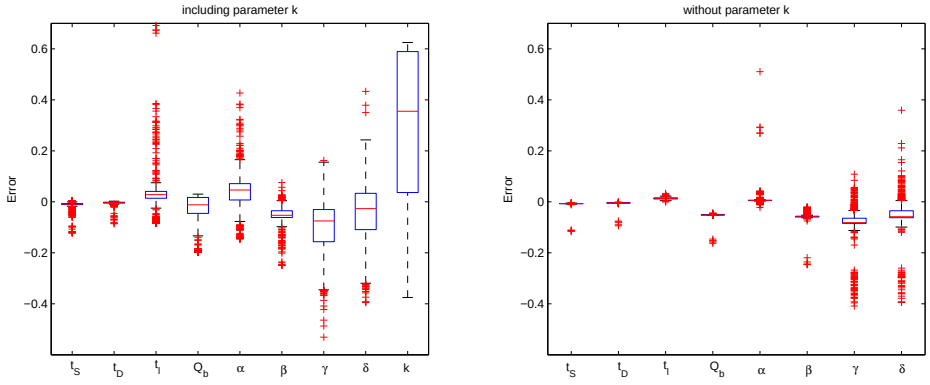


Figure 4.20: Relative error depending on initial values

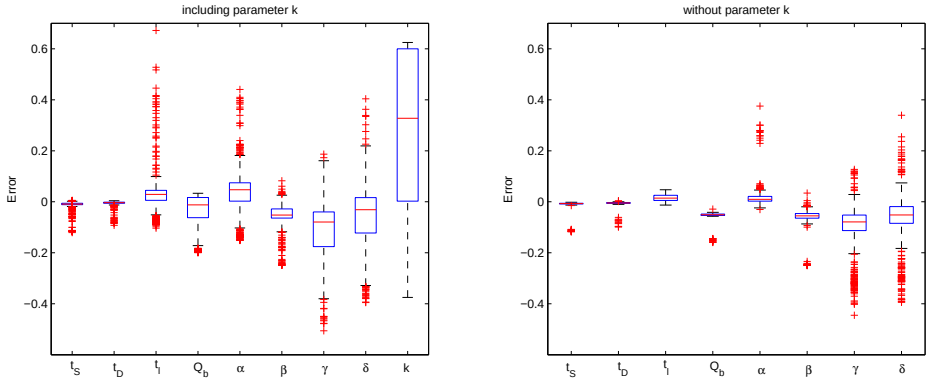


Figure 4.21: Relative error depending on initial values. Including Poisson noise with mean 10 in the image–series

4.2.3 Branch

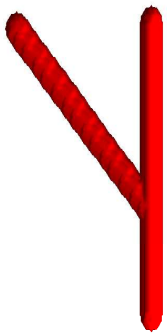


Figure 4.22: Branch

For a validation of bifurcations, a branched geometry was generated (see figure 4.22). This geometry has also a constant radius of 5 voxels. The extraction algorithms returned three centerline–radii representations of the branch. The first segment has 39, the second 61 and the third 68 centerline points. The extracted radii were about 5 voxels. The extracted flow maps, length maps and the reliability maps of this branch can be seen in figure 4.23.
Again we have a look on initial values. For validation 100 random initial values were chosen the results can be seen in figure 4.24 and figure 4.25. In figure 4.25 Poisson noise with mean 10 was applied to the image–series $I(x, y, t)$. We can see that the results for parameter χ are really bad. For this we do a further analysis. In figure 4.26 we present two times a course of the error function with respect to χ. For the first we set all remaining parameters to the optimum ($x = x^*$). For the other one we chose $x = (0.503, 2.987, 0.18, 198, 2.51, 0.601, 0.399, 0.204, 5.5)^T$, which stands for an arbitrary optimized value for x. The red cross indicates the minimum of the course of the function. We can see that the course of this function has several local minima and most regions have tiny gradients. This is why the estimation of parameter χ requires good initial values.

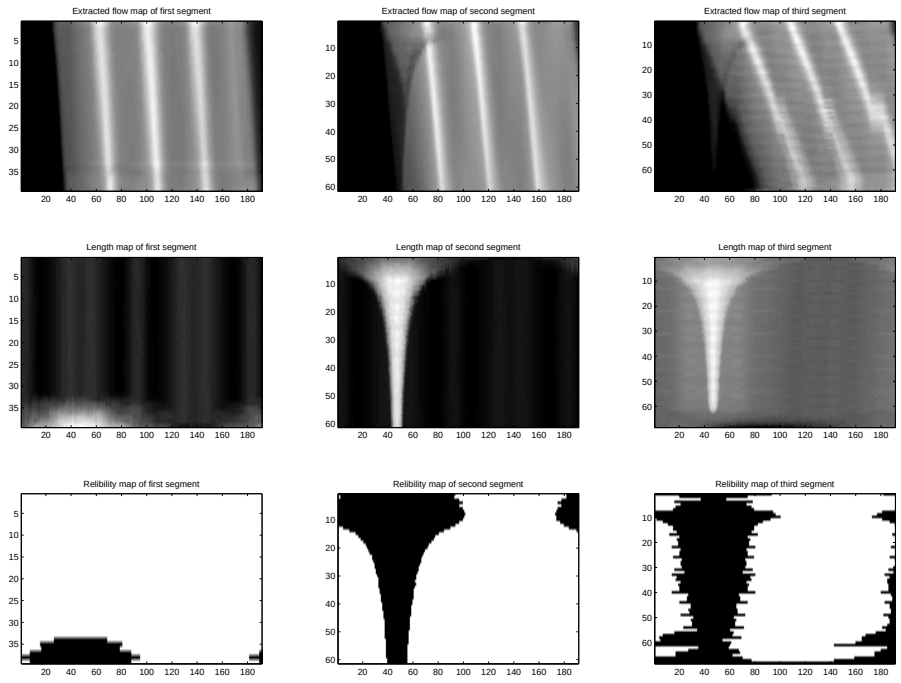


Figure 4.23: Extracted flow maps, length maps, reliability maps of branch

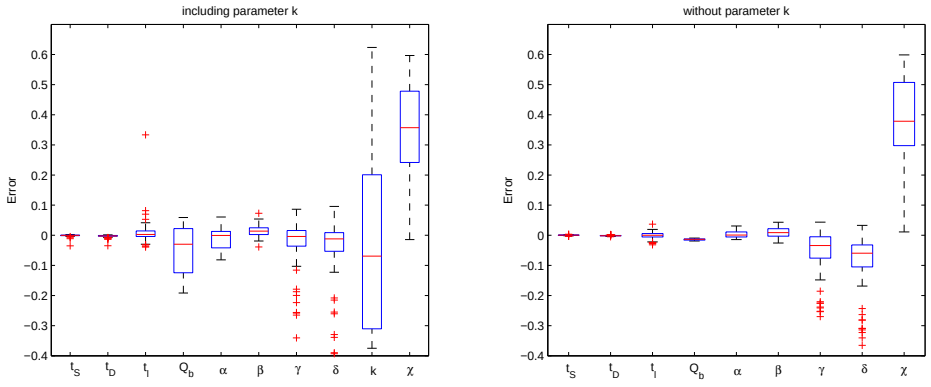


Figure 4.24: Relative error depending on initial values

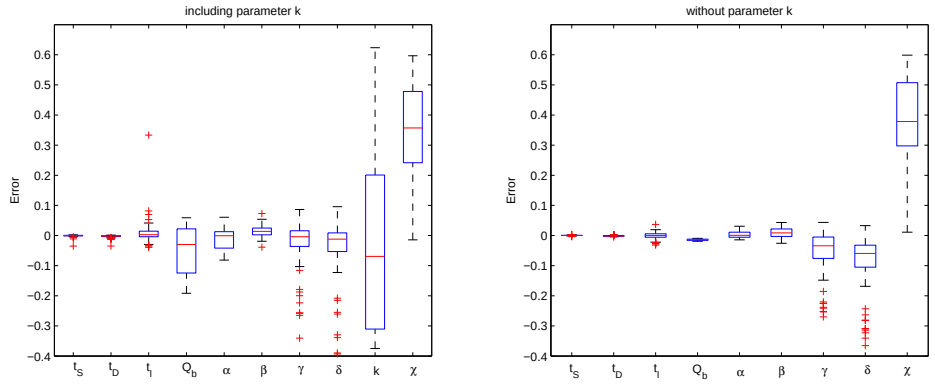


Figure 4.25: Relative error depending on initial values from noisy extracted flow map

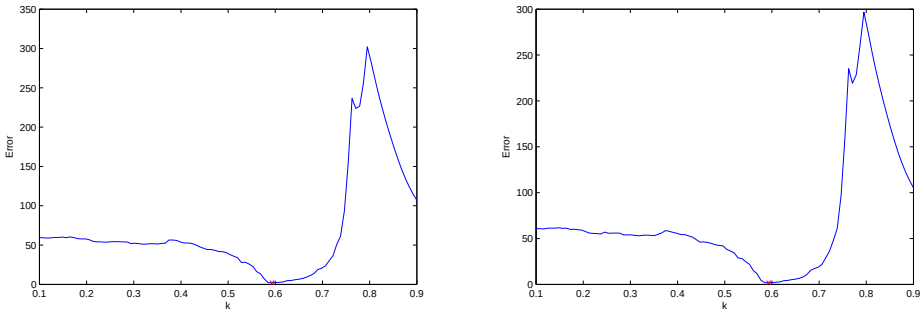


Figure 4.26: Error by varying k

5 Discussion and Conclusion

This chapter is a small forecast what could be done furthermore, to improve the achieved results. But first we will have a look at the results so far:
This is the first method to present quantitative results for blood flow estimation from rotational angiographic images, discovered by [Wae08]. Therefore the results are hard to directly compare to other methods. A major advantage of the method is that it uses an explicit model for the flow and contrast agent propagation. In this case, a simple convective dispersion model with laminar flow was used. The framework adopted for the flow fitting algorithm is such that the model can by readily replaced with an alternative, more sophisticated model if required.
A further advantage is that the model allows the use of additional information, for example information about the injection or information from the attenuation calibration. On the other hand, if this information is not available, the method can still give quantitative flow estimates. Unknown injection and calibration parameters can be estimated during the fitting process.
The algorithm for blood flow estimation is robust concerning noise. The estimated values for the parameters are relatively precise and adequate for practical purpuses, where commonly the first two digits are sufficient. It is possible that this is because the method uses two different kinds of information from the flow map: kinematic and densitometric information.
Due to the fact that vessels mostly cover only small parts of the volume (small arteries sometimes have a radius of 1 voxel) this method can not be applied for any data, because this method needs radial information. It does not make sence to estimate, for example the flow profile, if we have a vessel with radius of a voxel.
Section 4.1 showed that the flow profile is a very sensitive parameter which requires good initial values. One should weigh how far this parameter should be considered. Due to the small radius of vessels in the body it is questionable how the flow profile looks inside a vessel.
The modeling of branches is physically not correct (see figure 5.1). Furthermore the division factor χ is hard to optimize, because the function course has several local minima and huge regions with tiny gradients. This is why future works should have a closer look to the navier stokes equations - maybe for a first suggestion - just for incompressible fluids. On the one hand doing a simulation with navier stokes is rather time consuming. On the other hand, the velocity field according to the navier stokes equations is more realistic and, in a good implementation, is only necessary

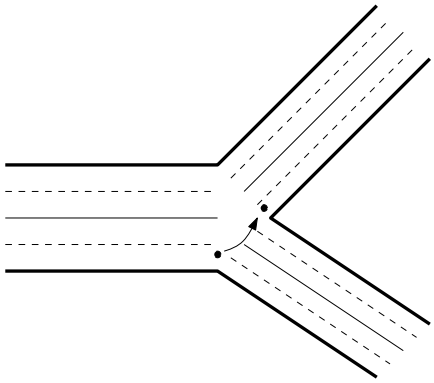


Figure 5.1: Incorrect setting of the new boundary conditions

as a precalculation step.
Calculating the diffusion–advection equation in 3D would really improve the validation with infilled volumes and the visualization, because we do not have discretization effects. But some disadvantages could result from solving the diffusion–advection equation in 3D: the flowmap would not be good mapping anymore, because the discretization problems would arise here instead of the visualization. A meaningful mapping could be, to infill the data, then do a raycast and finally extract the data again. Furthermore a consideration of solving the diffusion–advection equation in 3D does only make sence connected with a 3D velocity field. This is in general very time consuming. Another open question is the boundary condition at the inlet of the segment when solving in 3D.
If an intraoperative application developes out of this method, it is obligatory that the algorithm structure is as parallel as possible, because an optimization on a standard PC takes about 4 to 20 minutes, depending on how much data is given. Maybe this can be computed much faster using an internal hardware like the graphic card.
Another important implementation detail is the memory management. So far all algorithms were implemented in MatLab R2008b on a PC with 2 GB system memory. If there is a bad memory management, it is impossible to load data which occupies approximately 500 MB of system memory even if there is physically enough space, for example 1 GB. The reason, why this fact arises in MatLab, is that the memory gets very fragmented due to the fact that MatLab creates copies of a variable each time a command is executed.

Bibliography

[All06] Allan Paul, Dubbins Paul, McDicken Norman, Pozniak Myron. *Clinical Doppler ultrasound*. Elsevier Health Sciences, 2 edition, 2006. ISBN 0443101167.

[Gro08] Martin Groher. *2D-3D Registration of Vascular Images*. Dissertation, Technische Universität München, München, 2008.

[Har03] Hartley, Richard, Zisserman, Andrew. *Multiple View Geometry in computer vision*. Cambridge University Press, 2 edition, 2003. ISBN 0521–54051–8.

[Haw88] Hawkes, D., Colchester, A., Brunt, J., Wicks, D., du Bolay, G., Wallis, A. Development of a model to predict the potential accuracy of vessel blood flow measurements from dynamic angiographic recordings. *Mathematics and Computer Science in Medical Imaging*, 1988. Springer–Verlag Berlin.

[Hen08] Michel Henry. *Textbook of Peripheral Vascular Interventions*. Informa Health Care, 2 edition, 2008. ISBN 1841846430, 9781841846439.

[Kau96] Kauffmann, Günter, Moser, Ernst, Sauer, Rolf. *Radiologie*. Urban und Schwarzenberg, 1996. ISBN 3–541–18851–0.

[Kuh51] Kuhn H.W. and Tucker A.W. Nonlinear programming. *University of California Press, Berkeley and Los Angeles*, pages 481–492, 1951.

[Lag98] Lagarias Jefferey C., Reeds James A., Wright Margaret H. and Wright Paul E. Convergence properties of the nelder/mead simplex method in low dimensions. *Siam J. Optim.*, 9:112–147, 1998.

[Nel65] Nelder, J.A., Mead, R. A simplex method for function minimization. *The computer journal*, 7:308–313, 1965.

[Pal01] Palágyi, K., Sorantin, E., Balogh, E., Kuba, A., Halmai, C.,Erdőhelyi, B. and Hausegger, K. A sequential 3d thinning algorithm and its medical applications. In *Lecture Notes in Computer Science*, volume 2028, pages 409–415. In Proc. Int'l Conf. Information Processing in Medical Imaging (IPMI), Springer, 2001.

[PEN99] G. P. PENNEY. *Registration of Tomographic Images to X-ray Projections for Use in Image Guided Interventions*. Dissertation, University of London, 1999.

[Per93] Perrochet, P. and Bérod, D. Stability of the standard crank-nicolson-galerkin scheme applied to the diffusion-convection equation: Some new insights. *Water Resour. Res.*, 29(9):3291–3297, 1993.

[Rus06] Andrzej P. Ruszczyński. *Nonlinear optimization*. Princeton Univ. Press, 2006.

[Sar07] Gordon Sarty. *Computing brain activity maps from fMRI time-series images*. Cambridge University Press, 2007. ISBN 0521868262.

[Sch09] Schwarz, H.R., Köckler, N. *Numerische Mathematik*. Vieweg + Teubner, 7 edition, 2009. ISBN 978–3–8348–0683–3.

[Tay53] G. Taylor. Dispersion of soluble matter in solvent flowing slowly through a tube. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 219(1137):186–203, 1953.

[Wae08] Waechter, I., Bredno, J., Hermans, R., Weese, J. , Barratt, D.C., Hawkes, D.J. Model-based blood flow quantification from rotational angiography. *Medical Image Analysis*, 12:586–602, 2008. doi: 10.1016/j.media.2008.06.003.

[Win02] Wintermantel Erich, Ha Suk–Woo. *Medizintechnik mit biokompatiblen Werkstoffen und Verfahren*. Springer, 3 edition, 2002. ISBN 3540412611.

[Zah94] Zahlten, C., Jürgens, H., AND Peitgen, H.–O. Reconstruction of branching blood vessels from ct-data. *Eurographics Workshop of Visualization in Scientific Computing*, pages 161–168, 1994. Springer.