Parameter Identification for a Two-Compartment Contrast Flow Field Model

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation (Monografie)

von
Sophie Externbrink

aus
Bergisch Gladbach

2026

1. Gutachter: Prof. Dr. Daniel Ruprecht
2. Gutachter: Prof. Dr.-Ing. Tobias Knopp

Tag der mündlichen Prüfung: 26. März 2026

DOI: https://doi.org/10.15480/882.16937
ORCiD: https://orcid.org/0000-0001-7874-0496

I'm gonna live outside, live outside of all of this now!
- Enter Shikari

Summary

This thesis discusses the application of a two-compartment model for reconstructing perfusion parameters from dynamic contrast-enhanced ultrasound data, with the goal of providing information about tumour perfusion and thus treatment response.

The analysed two-compartment model can reconstruct a variety of concentration flows, including those with diffusive flow, and reconstructs a conversion function that is directly related to medical perfusion. This makes it more suitable for medical applications compared to the classical advection-diffusion model, which relies on pseudo-diffusion and thus does not directly relate to medical perfusion.

In this thesis, the parameter identification problem is solved using an optimisation setting and a *first-optimise-then-discretise* approach. The adjoint and gradient equations are derived analytically for both the two-compartment and advection-diffusion model. The optimisation problem is then solved using either a conjugate gradient or BFGS descent direction and the partial differential equations are discretised using a fifth-order weighted essentially non-oscillatory solver in space and a third-order IMEX or Runge-Kutta solver in time.

The reconstruction is further applied to 3D medical ultrasound data, which requires projecting the data onto a 2D-plane before feeding it into the parameter identification algorithm. The first results show velocities corresponding to average velocities of blood flow in the liver and a space-dependent conversion function, providing proof of feasibility for medical applications.

Current limitations include the computational time and the inability of handling inflow boundary conditions during the reconstruction. This restricts the measurement data to time steps where there is no or only little inflow of contrast agent. Meaning that only part of the measurement data can be used and therefore the reconstructed parameters are less reliable.

Acknowledgements

First, I would like to thank Daniel Ruprecht and Sebastian Götschel for their academic guidance, open doors and ears for my questions, and their unwavering support throughout this process.

I am also deeply grateful to all my colleagues for creating such a positive and productive work environment. Special thanks go to Abdul Qadir for being the best office mate imaginable — listening to all my problems and joining in on power naps instead of judging. I would also like to thank Lars for analysing the St.Pauli games after the weekend, Lina for bringing much-needed women's power to our male-dominated field, and Philip and Fynn for the extended coffee/tea and math breaks.

A big thank you to Simon for making my daily commute enjoyable and for the countless Fritz-Kola breaks that helped me stomach the mensa food.

To my large and loving family: thank you for your unwavering belief in me.

Finally, I am endlessly grateful to Paul for listening to my ideas, providing food and love, and offering fast tech support whenever needed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Liver cancer is one of the leading causes of cancer-related death due to its poor prognosis, despite being relatively uncommon. In Germany alone, there are currently around 9,800 new cases of cancer diagnosed each year, resulting in almost 8,200 deaths (in 2021) [82]. The average life expectancy for patients without the correct treatment based on tumour type is 8.7 months [1]. The tumour vascular system is also referred to as the "fingerprints" of cancer, because vascular abnormalities can vary significantly depending on the type of cancer and patient [59]. These vascular properties regulate the response to most cancer therapies, including not only the classical radiation and chemotherapy but also the rather new immunotherapies [38]. Thus, an optimal therapy on a patient-by-patient basis is crucial.

Being able to judge cancer's response to therapy is an important aspect of patient care [61]. Cancer treatment is often tough on the patient and accompanied by many side effects. Thus, changing the treatment early if it is unsuccessful can reduce the unwanted treatment effects and offer individual tumour treatment. But judging effectiveness of a treatment is not straightforward. One possibility are computed tomography (CT), magnetic resonance imaging (MRI) or positron emission tomography (PET) scans which offer direct visualisation of the cancerous region and thus the ability to judge treatment response. On the downside they are expensive, which limits the access, not usable at the bedside and limited due to radiation exposure (CT/PET) [75]. Therefore, they are unsuitable for regularly applications during treatment. Another possibility is to measure the treatment effect indirectly using tumour perfusion and vascular properties as indicators. To live and grow, the

tumour needs to be perfused and fed by blood vessels. If the treatment is working the perfusion parameters are changing and the blood flow towards the tumour is reduced. To visualise the blood flow and perfusion a three-dimensional dynamic contrast-enhanced ultrasound (3D DCE-US) can be used [37]. When using a contrast agent, the measured data can mathematically be described as a tracer model [2]. Using a mathematical model we want to reconstruct the tumor perfusion and vascular properties for the patient, allowing a more individual tumour treatment.

To reconstruct those parameters mathematically, we need to solve the inverse problem. This means that, given some concentration data, we choose a mathematical model and using some initial guesses we reconstruct the model parameters. These so called parameter identification problems, are well studied and even though the problems are almost always ill-posed they are used in various fields to determine parameters using collected data and a chosen model. For example the advection-diffusion model has been successfully used in order to diagnose prostate cancer [79], in ocean engineering [33] and environmental monitoring [81]. Furthermore, there are numerous applications where the advection in models has been reconstructed computationally starting already in the early 2000s [12] and with a variety of applications, for example in tracer dispersion [52] or formation and evolution of orogenic topography [18].

Finding a good mathematical model as forward problem to solve the inverse problem is crucial, if we are to obtain results that relate to physiological parameters and thus can be applied to cancer treatment. Tracer models based on advection-diffusion equations are most commonly found, but they lack the ability to achieve physically accurate solutions for the transportation of tracer through an organ. They use the concept of regions-of-interests which are isolated systems receiving indicator through one source (the arterial input function) [41] [72]. Therefore, Sourbron proposed different models including an advection-diffusion and a two-compartment model in [71]. The advection-diffusion approach was used by Hristov et al. [37] to reconstruct physically interpretable parameters that are sensitive to whole vascular network changes, and correlate to histology. However, the reconstructed mean-absolute diffusion has no physiological interpretation in terms of perfusion, and correlations are rather weak. This motivates the application of a two-compartment model, which tackles the issues of correctly modelling bolus dispersion and thus reconstructing parame-

ters relating directly to physiological perfusion.

The two-compartment model of Sourbron is based on advection equations. They separate the concentration of tracer into two variables. The tracer concentration found in arterial blood and the one in venous blood. This separation enables the model to describe the diffusion of tracer seen in organs through transformation from arterial to venous blood, rather than diffusion. The transformation from arterial to venous blood can be described by a space-dependent conversion function. Sourbron introduced this model to improve the existing models used for medical applications. Following his suggestion, this thesis applies this model as a basis for a parameter identification, shifting the problem from a modelling problem to an optimisation problem. The goal of the parameter identification is to reconstruct the velocities of the tracer and the conversion function, given the data of the concentration of tracer over time.

This thesis describes and analyses the reconstruction of perfusion related quantities for the two-compartment model, including a Leray projection to achieve divergence free velocity fields, which is indeed able to recover the velocities and conversion function. Furthermore, we compare the results of the two-compartment model to the parameter identification used on the classical advection-diffusion model. The comparison shows that the two-compartment model can reconstruct a multitude of concentration flows, including diffusive concentration flows. Furthermore, it reconstructs a conversion function, which directly relates to the medical perfusion and potentially yields information about the success of the cancer treatment. This shows that the two-compartment model is much more suitable for the medical application even though both produce reasonable results. Lastly, the reconstruction is applied to 3D medical data from ultrasound measurements. In order to run these reconstructions the data has to be adapted and transformed into 2D, before it is fed into the parameter identification. The results obtained in this thesis are reconstructed velocities with values that correspond to the average velocities in the liver and a space-dependent conversion function. Thus, providing the first proof of feasibility for medical applications.

Chapter 2

Parameter Identification for Tumour Perfusion

Regarding the problem of tracer transportation in blood flow of an organ the most common model is an advection-diffusion model [41], [16], [37]. In this thesis we compare the advection-diffusion model to a two-compartment model [71]. We start by giving an introduction to inverse problems and parameter identification in general. Before presenting the theory needed for solving the parameter identification using the advection-diffusion model in Section 2.3 and the two-compartment model in Section 2.4. The parameter identification in those sections will be solved via an optimisation setting using the formal Lagrange principle, for in depth explanations and mathematical background the reader is referred to Tröltzsch (2009) [76].

2.1 Inverse Problems

Mathematical simulations are widely known and we encounter them regularly in everyday life, for example in weather forecasts or climate predictions. We start by selecting a partial differential equation (PDE) e that models the real process, set parameters c, and compute the state u. This can be written as $e(c, u) = 0$, which is in an optimisation setting known as the *forward problem*. A parameter identification has a similar setting. Here we also choose a model but instead of setting the model parameters, we want to discover the parameters with which a given "result" has been derived. This means given e and u we want to derive c. This problem is almost always ill-posed [6][74], which means that we cannot guarantee any or all of

the following three: existence, uniqueness and stability. A parameter identification can also be interpreted as an optimal control problem and thus be written as an optimisation problem with PDE constraints. To do so, we define a function we want to minimise. This function is called *cost functional* and is defined by the mismatch of the calculated u and measured value u^δ, and some regularisation term. Putting everything together we get the following optimisation problem, for parameters c and state u chosen from real Banach spaces C and U, respectively

$$\begin{aligned} \min_{c\in C, u\in U} \frac{1}{2}||u-u^\delta||^2 + \frac{\lambda}{2}||c||^2 \\ \text{s.t. } e(c,u) = 0. \end{aligned} \tag{2.1.1}$$

Here $\lambda > 0$ can be physically interpreted as some sort of weighting cost, for example energy costs of the parameter c, or mathematically as a regularisation, which ensures smoothness. Furthermore, we call u the state of the optimisation and c the control. The control can be applied at different points of the PDE, for example at the initial condition, boundary or as *distributed control* on the whole domain. Thus, the model, the cost functional and the control define what kind of optimisation problem we derive, for example a *linear-quadratic parabolic boundary control problem* would be an optimisation problem with a linear parabolic PDE constraint, a quadratic cost functional and a control which acts on the boundary. In general, if we have a mathematical equation or problem we want to be able to say if the problem is solvable or even uniquely solvable. Because the problem is most likely ill-posed we do not expect to find existence, uniqueness or stability in general. But examining the problem for existence of a solution yields a better understanding of the possibilities and restrictions.

If we have a look at the optimisation problem we can see that the PDE constraint defines the state u regarding control c. Thus, to show the existence of a solution for the minimisation problem we need the PDE to be uniquely solvable. Here the definition of solvable matters, in general weak solutions are enough to show the existence of a solution. Most often strong solutions, which for example need to be twice differentiable if the PDE contains the Laplace operator, are not guaranteed [76]. If the problem is uniquely solvable we can write the cost functional J as a reduced cost functional j only dependent on control c, with C being again a real

Banach space

$$j(c) = \min_{c \in C} \frac{1}{2}||u(c) - u^{\delta}||^2 + \frac{\lambda}{2}||c||^2. \tag{2.1.2}$$

Next we define the *solution operator* $S : c \mapsto u(c)$, which describes the connection between control and state. If the solution operator is continuous we know that a small change in the control only results in a small change in the state variable. This results in the continuity of the reduced cost functional with respect to the control c. In order to apply Weierstrass theorem on *Existence of a Global Minimum*, which can be found in Anton (1998) [3] or any other standard book on analysis, we further need a compact set C from which we choose the control. If we have continuous function on a compact set we get the existence of a solution for the inverse problem. Regarding elliptic and parabolic PDEs the existing theory is quite extensive and analogies to linear-quadratic hyperbolic problems and examples can be found in Tröltzsch (2009) [76].

In Section 2.3 and 2.4 we will formally introduce two different mathematical models as forward problems. The resulting optimisations will be solved using the Lagrangian approach. The theory discussed in those sections is based on Tröltzsch book [76]. This approach is a *first optimise then discretise* method, which means that we analytically derive a gradient in which direction we minimise the cost functional. Opposed to this is the *first discretise then optimise* approach, where the gradient is calculated numerically after the PDE is discretised. For further information about the existence of such solutions and the behaviour of PDE constrained optimisation systems and inverse problems in general, the reader is referred to Tarantola (1987) [74], Tröltzsch (2009), [76], Isakov (2017) [42] and Aster et al. (2013) [6].

2.2 Medical Background

Being able to derive information about tumour vascular networks, which are physiologically abnormal and vary not only by tumour type but also from patient to patient, is important because they influence treatment effects. Therefore, characterising vascular networks non-invasively at the patient bedside could provide rapid clinical decision support, and thus improve patient care. The goal is to derive vascular information from the tumour using ultrasound as imaging method. For this, we

use that the application of a contrast agent results in data that can be interpreted as a tracer model [71]. This data should help us derive information about the vascular changes which in return imply if the tumour is responding to chemotherapy. In the following we will give a short explanation about the liver blood flow. The information is taken from Jarnagin (2017) [43] and for more information the reader is referred to Chapter 4 in this book.

The liver receives a dual blood supply through the hepatic artery and the portal vein. This is unique for an organ and means that $75 - 80\%$ of the blood entering the organ is venous blood from the stomach, intestine and spleen, only the remaining $25 - 20\%$ are well-oxygenated blood from the hepatic artery, for an illustration see Figure 2.1. In the liver 40% of blood are held in large vessels like the hepatic

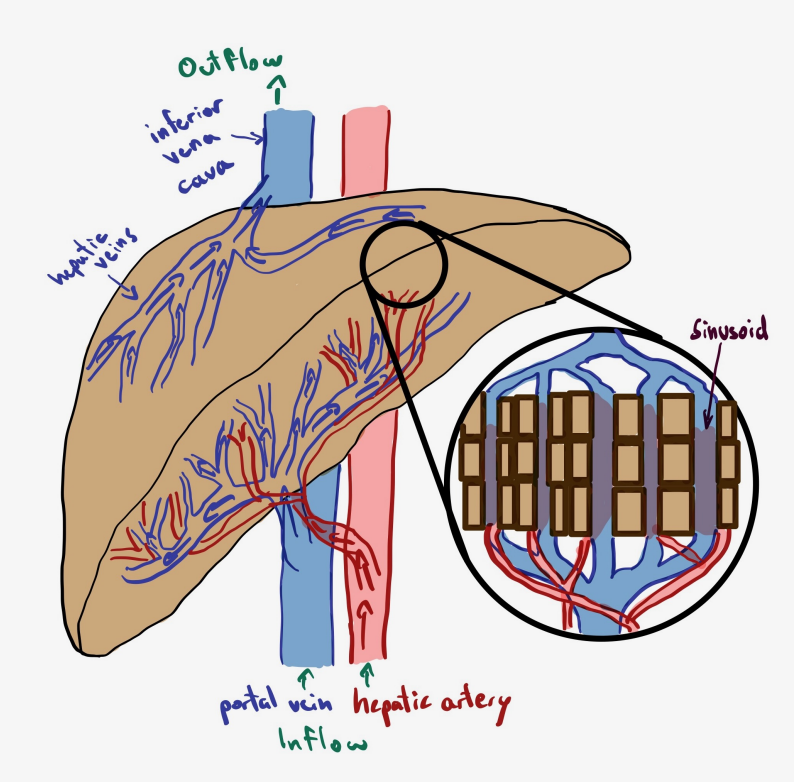
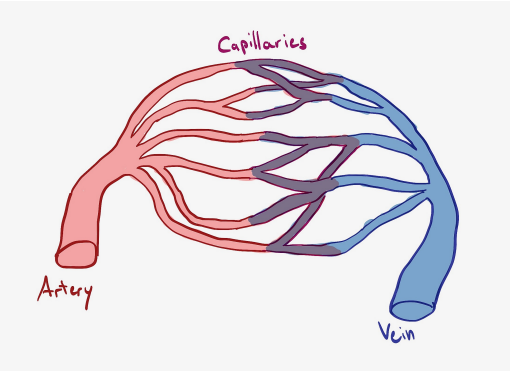


Figure 2.1: Sketch of the vascular system of the liver.

arteries, portal vein and hepatic veins. The other 60% are held in sinusoids. Liver sinusoids are a type of capillaries which are also known as sinusoidal capillaries, they are lined by specialised endothelial cells known as the liver sinusoidal endothelial cells. Those cells are porous and have a scavenging function, which means that they

cleanse the cells. These properties result in pseudo-diffusive behaviour of the contrast agent. The hepatic sinusoids mix the high pressure, well-oxygenated arterial blood with the low pressure, less well-oxygenated, but nutrient rich, portal venous blood. Afterwards the hepatic sinusoids are drained by the hepatic venous system into the inferior vena cava.

This complex model can be simplified to a model, where we have an inflow through an artery, followed by a branching out via capillaries, before the blood is drained by a vein. This simplified version is illustrated in Figure 2.2. The simplification is reasonable because we cannot distinguish between arterial and venous blood on the ultrasound and the visualisation quality is not high enough to distinguish capillaries [37]. Furthermore, the simplified model is applicable to more modelling problems, because it is not as unique as the liver vascular system.



(a) Schematic representation of a vascular system, vessel color indicating presence of oxygenated (red) and deoxygenated (blue) blood.

(b) The finite resolution of the imaging system (ultrasound) eliminates small vessel details.

Figure 2.2: A simplified vascular model of the liver.

2.3 Advection-Diffusion Model

When modeling tracer transport in organ blood flow we need to be able to describe an in- and outflow via the main artery, respectively vein. Furthermore, upon entering the organ the tracer spreads out through a lot of smaller arteries. This can be modelled via a diffusive part. On the other hand we also need to model the tracer

exiting the organ. Here, the tracer flows together via a large network of small veins. This can only be modelled by negative diffusion, which would be physically not correct and results in unstable solutions of the PDE. The general behaviour of the advection-diffusion model can be seen in Figure 2.3. As simplification a 2D model is

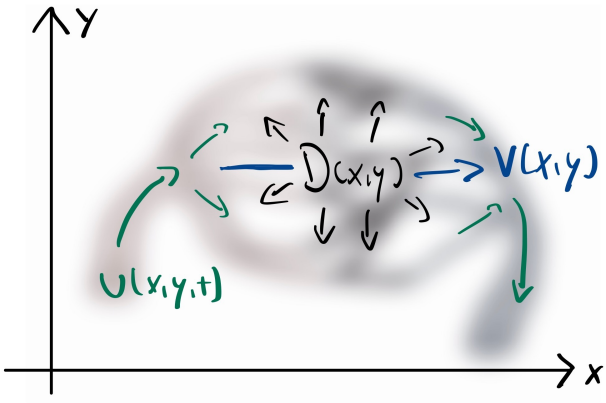


Figure 2.3: Sketch of the transportation of tracer in blood flow through an organ using the advection-diffusion equation.

sufficient to capture the main character of tracer transport in blood flow. Thus, the development and evaluation of the optimisation scheme will be done in 2D, with the option to add a third spatial dimension later. This also yields shorter calculation times because smaller mathematical systems need to be solved. The advection-diffusion model is given by the following PDE which is defined on $Q = \Omega \times [0, T]$, over space and time. Here Ω is the space domain with boundary Γ. The PDE is defined with no-flow zero von Neumann boundary conditions (BC) and initial condition u_0

$$\begin{aligned} u_t + \nabla \cdot (V\, u) - \nabla \cdot (D\, \nabla u) &= 0 \\ u(\cdot, t = 0) &= u_0 \\ \partial_\nu u|_\Gamma &= 0. \end{aligned} \tag{2.3.1}$$

As mentioned before, we use the Lagrangian approach for the parameter identification and thus an adjoint optimisation. Writing the problem in an optimisation

manner results in

$$\begin{aligned}
&\text{State:} && u : Q \to \mathbb{R}, \text{ with } Q = \Omega \times [0,T] \\
&\text{Control:} && c = (V, D) \\
&\text{PDE:} && u_t + \nabla \cdot (V\, u) - \nabla \cdot (D\, \nabla u) = 0 \text{ on } Q \\
&\text{BC:} && \partial_\nu u|_\Gamma = 0 \\
&\text{initial cond.:} && u(x, y, 0) = u_0(x, y).
\end{aligned} \tag{2.3.2}$$

Here, V is the velocity field with $V = \begin{pmatrix} v_x & v_y \end{pmatrix}^T$, $v_x, v_y : \Omega \to \mathbb{R}$ and D the diffusion, $D : \Omega \to \mathbb{R}$. Thus, all parameters are space-dependent and $(\cdot)$ denotes the scalar product. After setting $\Sigma = \Gamma \times [0,T]$, we can define the cost functional, which consists of the mismatch of the calculated concentration u, the known data u^δ and some regularisation parameters, which ensure that the parameters are reasonably bound, by punishing high values, and leading to improved smoothness properties of the optimal control variables, typically we choose $\lambda \in [10^{-5}, 10^{-3}]$ [36].

$$J(u,c) = J(u,(V,D)) = \frac{1}{2}||u - u^\delta||^2_{L_2(Q)} + \frac{\lambda_1}{2}||V||^2_{L_2(Q)^2} + \frac{\lambda_2}{2}||D||^2_{L_2(Q)} \tag{2.3.3}$$

This yields the following optimisation problem

$$\begin{aligned}
\min_{u \in U, V \in \overline{V}, D \in \overline{D}} & \frac{1}{2}||u - u^\delta||^2_{L_2(Q)} + \frac{\lambda_1}{2}||V||^2_{L_2(Q)^2} + \frac{\lambda_2}{2}||D||^2_{L_2(Q)} \\
\text{s.t. } & e(u,c) = 0,
\end{aligned} \tag{2.3.4}$$

where $e(u,c)$ is the given PDE and the L_2-norm is given by $||u||_{L_2(Q)} = (u,u)_{L_2(Q)}$ with the L_2 inner product $(u,v)_{L_2(Q)} = \int_Q u(x)\, v(x)\, dx$.

2.3.1 Existence of Solution

Now regarding the existence of the solution we have a look at the three steps explained in Section 2.1. First of all we inspect the forward problem, which is an advection-diffusion equation. In general, we expect to get a unique solution, but not for all control parameters. For example if we choose $D = 0$ we get an advection equation which needs different boundary conditions to be analytically solvable, for example none at the outflow boundaries. This restriction can become an issue in the implementation because the diffusion is a control parameter and thus will be adapted during the optimisation process. It is not always possible to ensure that the

diffusion is non-zero at the boundary, which would ensure solvability for the chosen zero von Neumann boundary conditions. Furthermore, there are certain domains and velocity fields for which the advection-diffusion equation is not uniquely solvable [9][8]. But if we could ensure that the problem is uniquely solvable the next step would be to show that the solution operator $S : c \mapsto u(c)$ is continuous [21]. This is fulfilled for the advection-diffusion equation a proof is omitted in this thesis [76]. The last step, showing compactness of the control parameters is not straightforward. Checking if we can find a compact set for the control parameters requires in-depth knowledge of functional analysis and would exceed the scope of this thesis [36]. We are aware that the existence of a solution for the inverse problem is analytically not given and cannot be expected. Nevertheless, we are able to calculate solutions numerically and thus find controls c, which yield reasonable results by stabilising the ill-posed problem using constraints and regularisation terms.

2.3.2 Derivation of Adjoint System and Gradient

After defining the cost functional we can define the Lagrangian, which yields a function that describes the relationship between the gradient of the cost functional and the constraints. Thus, it is a reformulation of the original problem.

$$L(u, c, p) = J(u, c) - (p, u_t + \nabla \cdot (V\, u) - \nabla \cdot (D\, \nabla u))_{L_2(Q)}, \tag{2.3.5}$$

where p is the adjoint state. The next step is to derive the adjoint equation, for this we compute the derivative of $L(u, c, p)$ with respect to the state u. Assume that V and D are sufficiently regular such that $u \in H^1([0,T],\, H^1(\Omega))$, $\Delta u(\cdot, t) \in H^1(Q)^*$, $\partial_\nu u \in H^1([0,T], H^{-\frac{1}{2}}(\Gamma))$ and direction $d \in H^1([0,T], H^1(\Omega))$, with ν the outer normal direction of the boundary. Here, H stands for Hilbert space, $H^1(Q)$ is also known as Sobolev space $W^{1,2}(Q)$, which is a Banach space and thus complete and normed.

$$\begin{aligned} L_u(u, c, p)\, d &= J_u(u, c)\, d - (p, d_t + \nabla \cdot (V\, d) - \nabla \cdot (D\, \nabla d))_{L_2(Q)} \\ &= J_u(u, c)\, d - \int_0^T \int_\Omega p\, (d_t + \nabla \cdot (V\, d) - \nabla \cdot (D\, \nabla d))\, dx\, dt, \end{aligned} \tag{2.3.6}$$

here $L_u(u, c, p)\, d$ stands for the action of a linear operator L_u on a vector in H^1. We start by using integration by parts to calculate the adjoint formulation for the time

derivative

$$\begin{aligned}\int_0^T \int_\Omega p\, d_t \, dx\, dt &= \int_\Omega [d(x,t)\, p(x,t)]_0^T \, dx - \int_0^T \int_\Omega d\, p_t \, dx\, dt \\ &= \int_\Omega (d(x,T)\, p(x,T) - d(x,0)\, p(x,0))\, dx \\ &\quad - \int_0^T \int_\Omega d\, p_t \, dx\, dt.\end{aligned} \tag{2.3.7}$$

Next, we use Green's identity [76] and partial integration to derive the adjoint formulation for the integral with the space gradient and Laplace operator using the knowledge that $\nabla \cdot V = \mathrm{div}(V)$ stands for the divergence of velocity field V.

$$\begin{aligned}\int_0^T \int_\Omega p\, \nabla \cdot (V\, d)\, dx\, dt &= -\int_0^T \int_\Omega \nabla p \cdot V\, d\, dx\, dt \\ &\quad + \int_0^T \int_\Sigma p\, V \cdot \nu\, d\, dS\, dt \\ \int_0^T \int_\Omega p\, \nabla \cdot (D\, \nabla d)\, dx\, dt &= -\int_0^T \int_\Omega \nabla p \cdot (\nabla d\, D)\, dx\, dt \\ &\quad + \int_0^T \int_\Sigma p\, D\, \partial_\nu d\, dS\, dt \\ &= +\int_0^T \int_\Omega \nabla \cdot (D\, \nabla p)\, d\, dx\, dt \\ &\quad - \int_0^T \int_\Sigma \partial_\nu p\, D\, d\, dS\, dt \\ &\quad + \int_0^T \int_\Sigma p\, D\, \partial_\nu d\, dS\, dt\end{aligned} \tag{2.3.8}$$

And for completeness, we also write down the derivative of the cost functional with respect to u in direction d

$$J(u,c)\, d = (u - u^\delta, d)_{L_2(Q)}. \tag{2.3.9}$$

This yields in total

$$\begin{aligned}L_u(u,c,p)\, d = &(u - u^\delta, d)_{L_2(Q)} + (p_t, d)_{L_2(Q)} + (\nabla p \cdot V, d)_{L_2(Q)} \\ &+ (\nabla \cdot (D\, \nabla p), d)_{L_2(Q)} \\ &- \int_\Sigma d\, (p\, V \cdot \nu + \partial_\nu p\, D) - p\, D\, \partial_\nu d\, dS\, dt \\ &- \int_\Omega (d(x,T)\, p(x,T) - d(x,0)\, p(x,0))\, dx\end{aligned} \tag{2.3.10}$$

Next, we have a closer look at the choice of the direction in which we calculated the derivative.

1. First we choose $d \in C_0^\infty(\Omega)$ this yields $d = \partial_\nu d = 0$ on Γ and $d(T) = d(0) = 0$ on Ω.

$$\begin{aligned} &\int_0^T \int_\Omega d\left[(u - u^\delta) + (p_t + \nabla p \cdot V + \nabla \cdot (D\,\nabla p))\right] dx\, dt = 0 \\ &\Rightarrow -p_t - \nabla p \cdot V - \nabla \cdot (D\,\nabla p) = u - u^\delta \text{ on } Q \\ &\text{This is true because } C_0^\infty(\Omega) \text{ lies dense in } L_2(\Omega). \end{aligned} \tag{2.3.11}$$

2. Next, we have a look at the boundary and because $\partial_\nu u|_\Gamma = 0$ we have to choose $\partial_\nu d|_\Gamma = 0$ and d variable on Γ, which results in

$$p\,V \cdot \nu + \partial_\nu p\, D = 0 \text{ on } \Gamma. \tag{2.3.12}$$

3. If we decide to choose different boundary conditions for the PDE for example Dirichlet zero boundary conditions, $u|_\Gamma = 0$, and therefore get $d|_\Gamma = 0$ and $\partial_\nu d|_\Gamma$ variable we get the following restriction

$$p\,D = 0 \Rightarrow p|_\Gamma = 0. \tag{2.3.13}$$

Furthermore, we get the terminal condition $p(\cdot, T) = 0$ from the boundary of the time derivative. In total this yields the so called adjoint equation

$$\begin{aligned} -p_t - \nabla p \cdot V - \nabla \cdot (D\,\nabla p) &= u - u^\delta \text{ on } Q \\ p(\cdot, T) &= 0 \\ p\,V \cdot \nu + \partial_\nu p\, D &= 0 \text{ on } \Gamma \end{aligned} \tag{2.3.14}$$

We can see that the adjoint equation is quite similar to the forward problem. The main difference is that we are solving the equation backward in time now. This enables us to use substitution and, therefore, the same solver for the forward and backward problem. We set $\tau = T - t$, where T is the final time and $t \in [0, T]$. Then it follows that

$$p_\tau - \nabla p \cdot V - \nabla \cdot (D\,\nabla p) = (u(\tau) - u^\delta(\tau)), \tag{2.3.15}$$

with initial condition $p(\cdot, 0) = 0$. Furthermore, we rewrite the equation in a way, that we can reuse the solver for $\nabla f(u) = \nabla \cdot (V\,u)$.

$$p_\tau + \nabla \cdot (-V\,p) - \nabla \cdot (D\,\nabla p) = (u(\tau) - u^\delta(\tau)) - \nabla \cdot V\,p \tag{2.3.16}$$

again with starting condition $p(\cdot, 0) = 0$. This equation can be solved with the same solver as the forward problem, we only have to adapt the right-hand side and change the boundary conditions accordingly. Now that we have derived the adjoint problem the next step is to derive the descent direction for the optimisation. For this we need to calculate the derivative of the Lagrangian with respect to c. Because c contains three variables (v_x, v_y, D) we calculate the partial derivatives individually. You can see the results in the following

$$\begin{aligned}
L_{v_x}(u,p,c)\,d_{v_x} &= J_{v_x}(u,c)\,d_{v_x} - \left(p, \nabla\cdot\left(\begin{pmatrix} d_{v_x} \\ 0 \end{pmatrix} u\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_x}(u,c)\,d_{v_x} + \int_0^T \int_\Omega \frac{\partial p}{\partial x}\, d_{v_x}\, u\, dx\, dt - \int_\Sigma p\, d_{v_x}\, \nu_x\, u\, dS\, dt \\
L_{v_y}(u,p,c)\,d_{v_y} &= J_{v_y}(u,c)\,d_{v_y} - \left(p, \nabla\cdot\left(\begin{pmatrix} 0 \\ d_{v_y} \end{pmatrix} u\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_y}(u,c)\,d_{v_y} + \int_0^T \int_\Omega \frac{\partial p}{\partial y}\, d_{v_y}\, u\, dx\, dt - \int_\Sigma p\, d_{v_y}\, \nu_y\, u\, dS\, dt \\
L_D(u,p,c)\,d_D &= J_D(u,c)\,d_D - (p, -\nabla\cdot(d_D\,\nabla u))_{L_2(Q)} \in \mathbb{R} \\
&= J_D(u,c)\,d_D - \int_0^T \int_\Omega \nabla p\cdot\nabla u\, d_D\, dx\, dt + \int_\Sigma p\,\partial_\nu u\, d_D\, dS\, dt.
\end{aligned} \tag{2.3.17}$$

Here ν_x is the first component of the normal direction, while ν_y is the second component. Using further that the derivatives of the cost functional are given by

$$\begin{aligned}
J_{v_x}(u,c)\,d_{v_x} &= (\lambda_1\, v_x, d_{v_x})_{L_2(Q)} \\
J_{v_y}(u,c)\,d_{v_y} &= (\lambda_1\, v_y, d_{v_y})_{L_2(Q)} \\
J_D(u,c)\,d_D &= (\lambda_2\, D, d_D)_{L_2(Q)}.
\end{aligned} \tag{2.3.18}$$

We identify the gradient of the reduced cost functional by Riesz representation theorem for constant velocities v_x, v_y and diffusion D. We write $\nabla p = \begin{pmatrix} \frac{\partial p}{\partial x} & \frac{\partial p}{\partial y} \end{pmatrix}^T$ and derive the gradient

$$\begin{aligned}
\nabla j(V) &= \int_0^T \int_\Omega \lambda_1\, V + \nabla p\, u\, dx\, dt \\
\nabla j(D) &= \int_0^T \int_\Omega \lambda_2\, D - \nabla p\cdot\nabla u\, dx\, dt,
\end{aligned} \tag{2.3.19}$$

where the integral is applied on each vector entry individually for velocity V. If we have space-dependent velocities and diffusion we have to adapt the descent direction

by losing the space integral to derive a space-dependent gradient

$$\begin{aligned} \nabla j(V) &= \int_0^T \lambda_1 V + \nabla p \, u \, dt \\ \nabla j(D) &= \int_0^T \lambda_2 D - \nabla p \cdot \nabla u \, dt. \end{aligned} \tag{2.3.20}$$

The boundary conditions derived for the gradient are also called complementary slackness conditions and are always fulfilled if p and u are calculated correctly [10].

2.3.3 Implementation

Putting everything together we get the following optimisation steps that are illustrated in Figure 2.4 and Algorithm 1.

Algorithm 1 Parameter Identification - finds velocity V and diffusion D for given concentration over time y^δ

Require: V_0, D_0, y^δ
Ensure: V, D

while V or D are still adapted **do** ▷ k-loop
  while $||\nabla j_V|| > \text{tol}_1$ and $|J_{i+1} - J_i| > \text{tol}_2$ and $\alpha_i^V > \text{tol}_\alpha$ **do** ▷ i-loop
    Calculate u_i with given V_i and D_{k-1}, via the forward problem
    Calculate p_i with given V_i, D_{k-1} and u_i, via the backward problem
    Calculate ∇j for V_i and find suitable step size α_i^V via line search
    Optimise: $V_{i+1} = V_i - \alpha_i^V \nabla j_V$
  end while
  Optimal velocity field V_k for round k found
  while $||\nabla j_D|| > \text{tol}_1$ and $|J_{i+1} - J_i| > \text{tol}_2$ and $\alpha_i^D > \text{tol}_\alpha$ **do** ▷ i-loop
    Calculate u_i with given V_k and D_i, via the forward problem
    Calculate p_i with given V_k, D_i and u_i, via the backward problem
    Calculate ∇j for D_i and find suitable step size α_i^D via line search
    Optimise: $D_{i+1} = D_i - \alpha_i^D \nabla j_D$
  end while
  Optimal diffusion D_k for round k found
end while
Optimal diffusion $D = D_k$ and velocity $V = V_k$ found

Here α is the step length which can be set to one or by your favourite step size algorithm, for example the Armijo algorithm. When choosing the step size algorithm convergence should be ensured thus for the conjugate gradient method (strong) Wolfe conditions are needed while Armijo conditions are sufficient for the steepest descent method [57]. Note that we calculate the parameters sequentially, after each other. We do so because the parameters have a different amount of influence on the cost functional and different magnitude in values. Adapting them together results in less adaptation of the 'weaker' parameter, which is the diffusion D. Thus, it is sensible to split the calculation of the optimal controls. Another possibility would be to update the velocity and the diffusion parameter alternating, but this results in longer calculation times. This is the case because in the given variant the diffusion parameter converges within few iterations. With an alternating search we have to do as many steps in the diffusion as in the velocity.

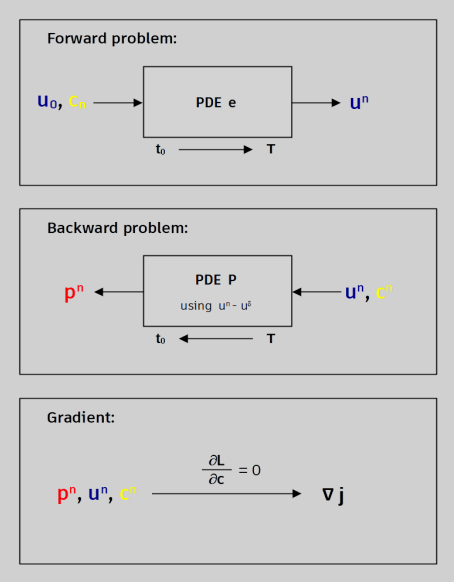


Figure 2.4: Illustration of the optimisation.

In line three, four, ten and eleven the advection-diffusion problem has to be solved. We use a cell-centred and uniform grid in space as well as in time. In our implementation the flux $\nabla \cdot f(u)$ is calculated via a weighted essentially non-oscillatory (WENO) solver which is implemented as a fifth-order solver, see Section 3.1.2. Because of the diffusive part we further need an implicit solver for the time step, here a third-order IMEX solver was implemented, see Section 3.2.2. The second derivative in space, for the Laplace operator, and the derivatives needed for

the gradient are calculated via a second-order central finite differences scheme, see Section 3.1.1. For the calculations of the gradient we further need to integrate, which is done in the simplest manner using the quadrature rule.

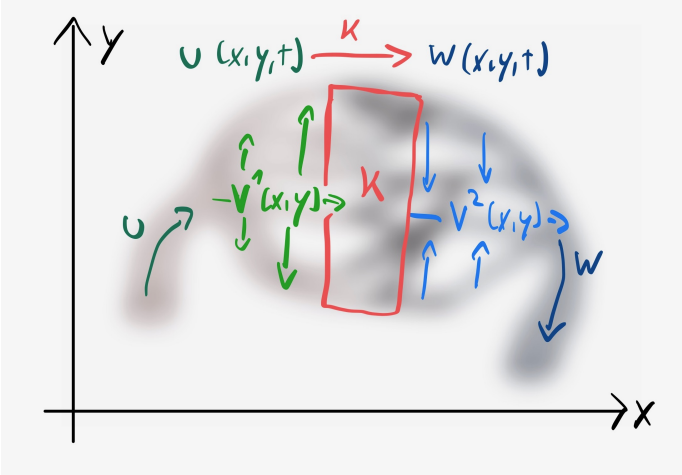
2.4 Two-Compartment Model

An alternative model for representing the movement of tracer in the blood flow through the liver is a simplified two-compartment model, which can be found in Figure 2.5. This model is introduced in Sourbron (2014) [71]. The basis of the model is an advection equation without diffusion. We assume space-dependent velocities $v : \Omega \to \mathbb{R}^2$. Furthermore, we have two components u and w which represent tracer in the oxygenated and deoxygenated blood, respectively. As introduced in Section 2.2 our model assumes that there is only oxygenated blood at the inflow boundary and only deoxygenated blood at the outflow boundary. The oxygenated blood is converted into deoxygenated blood using a space-dependent transformation rate $\kappa(x, y)$. In addition, both components should be able to move with different velocities v^1 and v^2. Where v^1 is the velocity of the oxygenated blood which is fast at the inflow boundary and is reduced to zero at the outflow boundary. And v^2 is the velocity of the deoxygenated blood which is slow at the inflow boundary and fast at the outflow boundary, compare Figure 2.6.

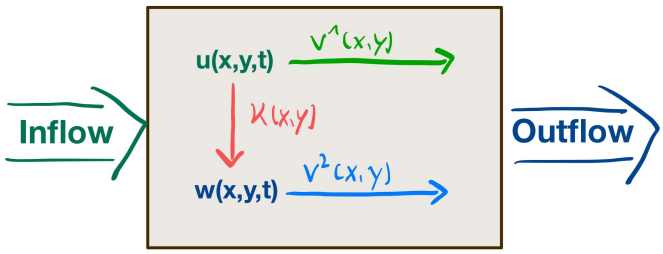
Altogether, the following system of equations applies

$$\begin{aligned} u_t(x, y, t) + \nabla \cdot f(u) &= -\kappa(x, y)\, u(x, y, t) \\ w_t(x, y, t) + \nabla \cdot g(w) &= \kappa(x, y)\, u(x, y, t) \\ f(u) &= v^1(x, y)\, u(x, y, t) \\ g(w) &= v^2(x, y)\, w(x, y, t) \end{aligned} \tag{2.4.1}$$

with $(x, y) \in \Omega$, for given space domain Ω and $t \geq 0$. The two-compartment model is chosen because during the travel through an organ the bolus is spreading out when entering and slimming down while exiting. This means that in the advection-diffusion model there has to be negative diffusion, which is physically not possible. We expect that the two-compartment model is able to achieve this behaviour, through the flexibility in velocities and the transformation parameter, while still being physically correct. This means that we need to introduce a new optimisation



(a) Sketch of transportation of tracer in blood flow through an organ using the two-compartment model.



(b) Schematic representation of the two-compartment model we simulate.

Figure 2.5: Schematic explanation of the two-compartment model and its functionality.

Vessel Type	Internal Diameter	Total Area [cm^2]	⟨Pressure⟩ [mmHg]	Mean Flow Velocity	Vessel Class
Aorta	2.5 cm	5	100	30 cm/s	Conduit (Feeding)
Arteries	3-10 mm	20	90	20 mm/s	
Arterioles	30 um	40	60		
Capillaries	5 um	2500	17	0.3 mm/s	Exchange
Venules	70 um	250	10	3 mm/s	Conduit (Draining)
Veins	0.5 cm	80	5	10 mm/s	
Vena Cava	1.2 cm	8	0		

Figure 2.6: Average haemodynamic parameters of the human circulation, from Hudson (2011) [39] with data from Guyton (1996) [30].

system.

$$\begin{aligned}
&\text{State:} && u, w : Q \to \mathbb{R}, \text{ with } Q = \Omega \times [0, T] \\
&\text{Control:} && c = (v^1, v^2) \\
&\text{PDE System eq. 1:} && u_t + \nabla \cdot (v^1\, u) + \kappa\, u = 0 \text{ on } Q \\
&\text{PDE System eq. 2:} && w_t + \nabla \cdot (v^2\, w) - \kappa\, u = 0 \text{ on } Q \\
&\text{Boundary condition:} && \partial_\nu u|_{\Gamma^1} = 0 \quad \wedge \quad \partial_\nu w|_{\Gamma^1} = 0 \\
&\text{initial condition:} && u(x, y, 0) = h(x, y) \quad \wedge \quad w(x, y, 0) = 0,
\end{aligned} \tag{2.4.2}$$

where v^1 and v^2 are the space-dependent velocity fields with $v^1, v^2 : \Omega \to \mathbb{R}^2$ and $\kappa : \Omega \to \mathbb{R}$ is also a space-dependent function. Again we set $\Sigma = \Gamma \times [0, T]$, where Γ is the boundary of Ω and Γ^1 contains inflow boundaries only. We start by defining the cost functional, which consists of the mismatch of the calculated concentrations of arterial and venous blood, u and w, with the known concentrations, u^δ and w^δ and some regularisation terms.

$$\begin{aligned}
J(u, w, c) = J(u, w, (v^1, v^2)) = \frac{1}{2}||u - u^\delta||^2_{L_2(Q)} + \frac{1}{2}||w - w^\delta||^2_{L_2(Q)} \\
+ \frac{\lambda_1}{2}||v^1||^2_{L_2(Q)^2} + \frac{\lambda_2}{2}||v^2||^2_{L_2(Q)^2}
\end{aligned} \tag{2.4.3}$$

This yields the following optimisation problem

$$\begin{aligned}
&\min J(u, w, c) \\
&\text{s.t. } e(u, w, c) = 0.
\end{aligned} \tag{2.4.4}$$

Omitting that we minimise over variables $u \in U$, $w \in W$ and $c \in C$, with real Banach spaces U, W and C. Furthermore, $e(u, w, c)$ is the given System of PDEs.

2.4.1 Existence of Solution

Again we have a look at the existence of the solution of the inverse problem. We start by looking at the forward problem. Here we have a coupled system of advection equations. In general, we expect to get a unique solution, but we have to be careful with the chosen boundary conditions. In order to get the existence of a solution we cannot have boundary conditions at the outflow boundaries [21]. Furthermore, for the advection equation there are as well domains and velocities for which we lose the uniqueness of the solution [56]. But in Modena and Székelyhidi (2018) the velocities are defined on a d-dimensional flat torus, which is not the case in our application

and thus not relevant. If we could ensure that the problem is uniquely solvable the next step would be to show that the solution operator $S : c \mapsto u(c)$ is continuous. This is fulfilled for the two-compartment system of equations, because it merely is a composition of advection equations [21], a detailed proof is omitted in this thesis. The last step, showing compactness of the control parameters is not straightforward. As for the advection-diffusion model, checking if we can find a compact set for the control parameters needs in-depth knowledge of functional analysis and would exceed the scope of this thesis [36]. We again are aware that the existence of a solution for the inverse problem is not given and cannot be expected. Nevertheless we can solve the problem numerically by using regularisation to stabilise the ill-posed inverse problem.

2.4.2 Derivation of Adjoint System and Gradient

Using the cost functional we now define the Lagrangian

$$\begin{aligned} L((u,w),(p,q),c) = J(u,w,c) &- (p, u_t + \nabla \cdot (v^1\, u) + \kappa\, u)_{L_2(Q)} \\ &- (q, w_t + \nabla \cdot (v^2\, w) - \kappa\, u)_{L_2(Q)}, \end{aligned} \tag{2.4.5}$$

with p and q the adjoint states. The next step is to derive the adjoint equation, therefore we have to find the derivative of $L(u,w,c,p)$ with respect to the states u and w, respectively. We calculate these derivatives in the following. Assume that $u \in H^1([0,T], H^1(\Omega)), \partial_\nu u \in H^1([0,T], H^{-\frac{1}{2}}(\Gamma))$ and for the descent direction $d \in H^1([0,T], H^1(\Omega))$, the same counts for w.

$$\begin{aligned} L_u((u,w),(p,q))\, d = J_u(u,w,c)\, d &- (p, d_t + \nabla \cdot (v^1\, d) + \kappa\, d)_{L_2(Q)} \\ &- (q, -\kappa\, d)_{L_2(Q)} \end{aligned} \tag{2.4.6}$$

Now we use partial integration to get the adjoint formulation for the time derivative.

$$\begin{aligned} \Rightarrow \int_0^T \int_\Omega p\, d_t\, dx\, dt &= \int_\Omega [d(x,t)\, p(x,t)]_0^T dx - \int_0^T \int_\Omega d\, p_t\, dx\, dt \\ &= \int_\Omega (d(x,T)\, p(x,T) - d(x,0)\, p(x,0))\, dx \\ &\quad - \int_0^T \int_\Omega d\, p_t\, dx\, dt \end{aligned} \tag{2.4.7}$$

Next we use Green's identity to derive the adjoint formulation for the integral with the space gradient and Laplace operator.

$$\Rightarrow \int_0^T \int_\Omega p\,\nabla\cdot(v^1\,d)\,dx\,dt = -\int_0^T \int_\Omega \nabla p\cdot v^1\,d\,dx\,dt + \int_0^T \int_\Sigma p\,v^1\cdot\nu\,d\,dS\,dt \quad (2.4.8)$$

And for completeness we also write down the derivative of the cost functional with respect to u and in direction d

$$J_u(u,w,c)\,d = (u-u^\delta,d)_{L_2(Q)}. \quad (2.4.9)$$

This yields in total

$$\begin{aligned} L_u((u,w),(p,q),c)\,d = {} & (u-u^\delta,d)_{L_2(Q)} + (p_t,d)_{L_2(Q)} \\ & + (\nabla p\cdot v^1,d)_{L_2(Q)} + (\kappa\,(q-p),d)_{L_2(Q)} \\ & - \int_\Sigma d\,p\,v^1\cdot\nu\,dS\,dt \\ & - \int_\Omega (d(x,T)\,p(x,T) - d(x,0)\,p(x,0))\,dx. \end{aligned} \quad (2.4.10)$$

Then we do the same for the state w and derive

$$\begin{aligned} L_w((u,w),(p,q),c)\,d = {} & (w-w^\delta,d)_{L_2(Q)} + (q_t,d)_{L_2(Q)} + (\nabla q\cdot v^2,d)_{L_2(Q)} \\ & - \int_\Sigma d\,q\,v^2\cdot\nu\,dS\,dt \\ & - \int_\Omega (d(x,T)\,q(x,T) - d(x,0)\,q(x,0))\,dx. \end{aligned} \quad (2.4.11)$$

Now we have a closer look at the choice of the direction in which we calculated the derivative.

1. First we choose $d\in C_0^\infty(\Omega)$ this yields $d=\partial_\nu d=0$ on Γ and $d(T)=d(0)=0$ on Ω.

$$\begin{aligned} & \int_0^T \int_\Omega d\,[(u-u^\delta) + (p_t+\nabla p\cdot v^1 + \kappa\,(q-p))]\,dx\,dt = 0 \\ & \Rightarrow -p_t - \nabla p\cdot v^1 - \kappa\,(q-p) = u-u^\delta \text{ on } Q \\ & \int_0^T \int_\Omega d\,[(w-w^\delta) + (q_t+\nabla q\cdot v^2)]\,dx\,dt = 0 \\ & \Rightarrow -q_t - \nabla q\cdot v^2 = w-w^\delta \text{ on } Q \end{aligned} \quad (2.4.12)$$

   This is true because $C_0^\infty(\Omega)$ lies dense in $L_2(\Omega)$.

2. Next we have a look at the boundaries and because $\partial_\nu u|_\Gamma = 0$, we know that $\partial_\nu d|_\Gamma = 0$ and therefore $d|_\Gamma$ variable we get the following restriction

$$\begin{aligned} p\, v^1 \cdot \nu = 0 \Rightarrow p|_\Gamma = 0 \\ q\, v^2 \cdot \nu = 0 \Rightarrow q|_\Gamma = 0. \end{aligned} \tag{2.4.13}$$

Now we can write down the adjoint system defined on Q, including the terminal condition given by the boundary condition of the time derivative

$$\begin{aligned} -p_t - \nabla p \cdot v^1 + \kappa\, p = u - u^\delta + \kappa q \\ -q_t - \nabla q \cdot v^2 = w - w^\delta \\ p(x,T) = 0 \\ q(x,T) = 0, \end{aligned} \tag{2.4.14}$$

which has to fulfill zero Dirichlet boundary conditions. Again the adjoint equation behaves quite similar to the forward problem, but we are solving the equation backward in time now. This yields the possibility to substitute in a way that enables us to reuse the solver of the forward for the backward problem. We set $\tau = T - t$, where T is the final time and $t \in [0, T]$. This yields

$$\begin{aligned} p_\tau - \nabla p \cdot v^1 + \kappa\, p = (u(\tau) - u^\delta(\tau)) + \kappa q \\ q_\tau - \nabla q \cdot v^2 = (w(\tau) - w^\delta(\tau)). \end{aligned} \tag{2.4.15}$$

To ensure that we can use the same solver for the flux as well we have to reorder the equation the following way

$$\begin{aligned} p_\tau - \nabla \cdot (v^1\, p) + (\kappa + \nabla \cdot v^1)\, p = (u(\tau) - u^\delta(\tau)) + \kappa\, q \\ q_\tau - \nabla \cdot (v^2\, q) + \nabla \cdot v^2\, q = (w(\tau) - w^\delta(\tau)). \end{aligned} \tag{2.4.16}$$

Now that we have derived the adjoint problem we can deduce the descent direction needed for the optimisation. For this we calculate the derivative of the Lagrangian with respect to c. Because c contains two variables (v^1, v^2) we calculate the partial

derivatives individually. The resulting expressions are

$$\begin{aligned}
L_{v_x^1}(u,p,c)\,d_{v_x^1} &= J_{v_x^1}(u,c)\,d_{v_x^1} - \left(p, \nabla\cdot\left(\begin{pmatrix} d_{v_x^1} \\ 0 \end{pmatrix} u\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_x^1}(u,c)\,d_{v_x^1} + \int_0^T \int_\Omega \frac{\partial p}{\partial x}\, d_{v_x^1}\, u\, dx\, dt - \int_\Sigma p\, d_{v_x^1}\, \nu_x\, u\, dS\, dt \\
L_{v_y^1}(u,p,c)\,d_{v_y^1} &= J_{v_y^1}(u,c)\,d_{v_y^1} - \left(p, \nabla\cdot\left(\begin{pmatrix} 0 \\ d_{v_y^1} \end{pmatrix} u\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_y^1}(u,c)\,d_{v_y^1} + \int_0^T \int_\Omega \frac{\partial p}{\partial y}\, d_{v_y^1}\, u\, dx\, dt - \int_\Sigma p\, d_{v_y^1}\, \nu_y\, u\, dS\, dt \\
L_{v_x^2}(w,q,c)\,d_{v_x^2} &= J_{v_x^2}(u,c)\,d_{v_x^2} - \left(q, \nabla\cdot\left(\begin{pmatrix} d_{v_x^2} \\ 0 \end{pmatrix} w\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_x^2}(u,c)\,d_{v_x^2} + \int_0^T \int_\Omega \frac{\partial q}{\partial x}\, d_{v_x^2}\, w\, dx\, dt - \int_\Sigma q\, d_{v_x^2}\, \nu_x\, w\, dS\, dt \\
L_{v_y^2}(w,q,c)\,d_{v_y^2} &= J_{v_y^2}(u,c)\,d_{v_y^2} - \left(q, \nabla\cdot\left(\begin{pmatrix} 0 \\ d_{v_y^2} \end{pmatrix} w\right)\right)_{L_2(Q)} \in \mathbb{R} \\
&= J_{v_y^2}(u,c)\,d_{v_y^2} + \int_0^T \int_\Omega \frac{\partial q}{\partial y}\, d_{v_y^2}\, w\, dx\, dt - \int_\Sigma q\, d_{v_y^2}\, \nu_y\, w\, dS\, dt
\end{aligned} \tag{2.4.17}$$

From this we can identify the gradient of the reduced cost functional by Riesz representation theorem for constant velocities v^1 and v^2, similar to the gradient in the advection-diffusion model, with $\nabla u = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix}$, $v^1 = \begin{pmatrix} v_x^1 \\ v_y^1 \end{pmatrix}$ and $v^2 = \begin{pmatrix} v_x^2 \\ v_y^2 \end{pmatrix}$.

$$\begin{aligned}
\nabla j(v^1) &= \int_0^T \int_\Omega \lambda_1\, v^1 + \nabla p\, u\, dx\, dt \\
\nabla j(v^2) &= \int_0^T \int_\Omega \lambda_2\, v^2 + \nabla q\, w\, dx\, dt,
\end{aligned} \tag{2.4.18}$$

where the integral is applied on every entry of the vector separately. If we have space-dependent velocities we have to adapt the descent direction by losing the space integral in order to derive a space-dependent gradient

$$\begin{aligned}
\nabla j(v^1) &= \int_0^T \lambda_1\, v^1 + \nabla p\, u\, dt \\
\nabla j(v^2) &= \int_0^T \lambda_2\, v^2 + \nabla q\, w\, dt.
\end{aligned} \tag{2.4.19}$$

2.4.3 Including the Transformation Parameter

If we want to include the transformation parameter κ, the cost functional J and the control c gain another parameter

$$\begin{aligned} J(u,w,c) &= J(u,w,(v^1,v^2,\kappa)) \\ &= \frac{1}{2}||u-u^\delta||^2_{L_2(Q)} + \frac{1}{2}||w-w^\delta||^2_{L_2(Q)} \\ &\quad + \frac{\lambda_1}{2}||v^1||^2_{L_2(Q)^2} + \frac{\lambda_2}{2}||v^2||^2_{L_2(Q)^2} + \frac{\lambda_3}{2}||\kappa||^2_{L_2(Q)}. \end{aligned} \tag{2.4.20}$$

Furthermore we also get a new Lagrangian

$$\begin{aligned} L((u,w),(p,q),c) = J(u,w,c) &- (p, u_t + \nabla\cdot(v^1\,u) + \kappa\,u)_{L_2(Q)} \\ &- (q, w_t + \nabla\cdot(v^2\,w) - \kappa\,u)_{L_2(Q)}. \end{aligned} \tag{2.4.21}$$

With help from the Lagrangian we can derive the descent direction for κ

$$L_\kappa\, d_\kappa = (\lambda_3\,\kappa, d_\kappa)_{L_2(Q)} - (p, d_\kappa\,u)_{L_2(Q)} - (q, -d_\kappa\,u)_{L_2(Q)} \tag{2.4.22}$$

which gives us via the Riesz representation theorem

$$\nabla j(\kappa) = \int_0^T \int_\Omega \lambda_3\,\kappa + (q-p)\,u\,dx\,dt. \tag{2.4.23}$$

Again, if we have space-dependent parameter we have to adapt the descent direction by losing the space integral

$$\nabla j(\kappa) = \int_0^T \lambda_3\,\kappa + (q-p)\,u\,dt. \tag{2.4.24}$$

2.4.4 Adapting to Real Measuring Data

In reality we are not able to measure the concentrations of arterial and venous blood separately. Therefore, we can only input the added concentrations which we call y^δ. This changes the cost functional of the problem to

$$\begin{aligned} J(u,w,c) &= J(u,w,(v^1,v^2,\kappa)) \\ &= \frac{1}{2}||u+w-y^\delta||^2_{L_2(Q)} + \frac{\lambda_1}{2}||v^1||^2_{L_2(Q)^2} + \frac{\lambda_2}{2}||v^2||^2_{L_2(Q)^2} + \frac{\lambda_3}{2}||\kappa||^2_{L_2(Q)}. \end{aligned} \tag{2.4.25}$$

This adaption also influences the right-hand side of the adjoint problem but the gradient stays unchanged. We derive the following new adjoint system on Q

$$\begin{aligned} p_\tau - \nabla \cdot (v^1\, p) + (\kappa + \nabla \cdot v^1)\, p &= (u(\tau) + w(\tau) - y^\delta(\tau)) + \kappa\, q \\ q_\tau - \nabla \cdot (v^2\, q) + \nabla \cdot v^2\, q &= (u(\tau) + w(\tau) - y^\delta(\tau)) \\ p(x, T) &= 0 \\ q(x, T) &= 0 \end{aligned} \tag{2.4.26}$$

It is visible that the changes are small and therefore, the changes in implementation are minor as well.

2.4.5 Implementation

Putting everything together we get the following optimisation steps written down in Algorithm 2. Here α is the step length satisfying strong Wolfe conditions, see Section 3.4.2, if the conjugate gradient or BFGS method is applied. And Armijo conditions, see Section 3.4.1, if the steepest descent method is used. Again we start by calculating the optimal velocity fields and only then adapt the transformation parameter until an optimal control is found. Using the same argument as for the advection-diffusion model, the parameters have different influence on the cost functional and are of different magnitude. Thus, it is reasonable to fix one first before moving on to the next. We start with the velocity fields because they have a bigger influence on the concentration flow, before adapting the transformation parameter. The implementation is quite similar to the optimisation using the advection-diffusion model. We still discretise using a cell-centred and uniform grid, in space and time, with ghost cells for the space-grid. Again we use a WENO solver for the flux $\nabla \cdot f(u)$, see Section 3.1.2, calculate the derivatives using a second-order central finite difference scheme, see Section 3.1.1, and the integrals using the quadrature rule. The only difference in implementation is the time stepping algorithm. Because we no longer have a diffusive part we can use an explicit solver. For higher accuracy a total variation diminishing third-order Runge-Kutta solver is chosen, see Section 3.2.1. Regarding the boundary conditions we use the standard outflow-flow boundary condition which sets the ghost cells behind the boundary to the value of the cell inside the domain [20]. This is the easiest application of an outflow boundary for a more sophisticated approach which includes setting the outflow boundary by a standard extrapolation of suitable order and accuracy to derive a stable scheme

the reader is referred to Goldberg and Tadmore (1978) [28] or Tan (2012) [73].

Algorithm 2 Parameter Identification - finds velocities v^1, v^2 and transformation parameter κ for given added concentration over time y^δ

Require: v_0^1,v_0^2, κ_0, y^δ

Ensure: v^1, v^2, κ

while v^1 and v^2 or κ are still adapted **do** ▷ k-loop
  while $||\nabla j_v|| >$ tol$_1$ and $|J_{i+1} - J_i| >$ tol$_2$ and $\alpha_i^v >$ tol$_\alpha$ **do** ▷ i-loop
    Calculate u_i, w_i with given v_i^1, v_i^2 and κ_{k-1}, via the forward problem
    Calculate p_i, q_i with given v_i^1, v_i^2, κ_{k-1}, u_i and w_i via the backward problem
    Calculate ∇j_v for velocities v_i^1 and v_i^2
    Find suitable step size via line search: α_i^v
    $$\begin{pmatrix} v_{i+1}^1 \\ v_{i+1}^2 \end{pmatrix} = \begin{pmatrix} v_i^1 \\ v_i^2 \end{pmatrix} - \alpha_i^v \nabla j_v.$$
  end while
  Optimal velocity fields v_k^1 and v_k^2 for round k found
  while $||\nabla j_\kappa|| >$ tol$_1$ and $|J_{i+1} - J_i| >$ tol$_2$ and $\alpha_i^\kappa >$ tol$_\alpha$ **do** ▷ i-loop
    Calculate u_i, w_i with given v_k^1, v_k^2 and κ_i, via the forward problem
    Calculate p_i, q_i with given v_k^1, v_k^2, κ_i, u_i and w_i via the backward problem
    Calculate ∇j_κ for transformation parameter κ_i
    Find suitable step size via line search: α_i^κ
    Optimise: $\kappa_{i+1} = \kappa_i - \alpha_i^\kappa \nabla j_\kappa$
  end while
  Optimal transformation parameter κ_k for round k found
end while
Optimal velocity fields $v^1 = v_k^1$, $v^2 = v_k^2$ and $\kappa = \kappa_k$ found

Chapter 3

Numerical Methods

In the following, we present the numerical solvers used in the implementation of the parameter identification. We start with the space-discretisations in Section 3.1, move on to the time-discretisations in Section 3.2, the line search methods in Section 3.4, a variety of descent directions in Section 3.3 and conclude with the Leray-Projection in Section 3.5. The Leray-Projection projects a velocity field onto a divergence-free velocity field. By applying this projection during the optimisation we can ensure that the found velocity field is divergence-free, for more information about the integration into the optimisation the reader is referred to Section 5.4.

3.1 Space-Discretisation

To solve both PDEs, the advection-diffusion equation and the two-compartment system of equations in space we need to be able to discretise the flux, $\nabla \cdot f(u)$, this is done via a fifth-order WENO solver, see Section 3.1.2. Furthermore, a few calculations, for example the gradient, require calculation of the first and second-order derivatives in space. These are solved using finite differences, which are explained in Section 3.1.1. To discretise the domain we use a cell-centred uniform mesh. This yields for a one dimensional domain $[a, b]$ the following discretisation

$$a = x_{\frac{1}{2}} < x_{1+\frac{1}{2}} < x_{2+\frac{1}{2}} < \cdots < x_{n+\frac{1}{2}} = b. \tag{3.1.1}$$

With cells $I = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$, corresponding cell centres x_i, and spacing $\Delta x = x_{i+1} - x_i$. Furthermore, for the WENO solver we need to introduce ghost cells. These are cells added to the grid at the boundaries to approximate values outside the computational domain, for a visualisation see Figure 3.1.

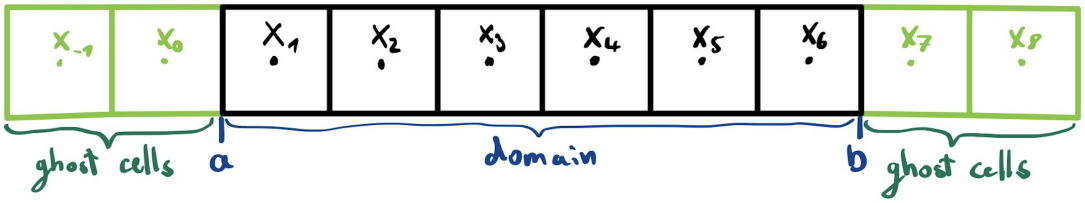


Figure 3.1: Ghost cell visualisation for a one-dimensional domain $[a, b]$ with six inner cells and two ghost cells (added to each end).

3.1.1 Finite-Differences

Finite differences are a fundamental tool in numerical analysis which enable us to approximate derivatives of functions, while only using a discrete set of data points. The basic idea is to measure the rate of change near a point using the surrounding points, which surrounding points are used, can differ. We separate the method into three larger groups: forward, backward and central finite differences. Forward differences only make use of the function values at the given and the following points, backward differences only make use of the function values at the given and the previous points and central finite differences make use of the function values at the given point, the previous and the following points. These methods are very well studied, therefore we can find high order and adaptive schemes as well as intensive stability and error analysis for different methods and problems. The interested reader is referred to LeVeque (2007) [50] for an extensive overview. In the following we use central finite differences at the centre of the domain and forward and backward finite differences at the boundary of the domain. For the implementation, we derived first- and second-order derivatives with second- and fourth-order convergence. Using the tables from [26] we approximate the first derivative via

$$f'(x_i) = \frac{-\frac{1}{2}f(x_{i-1}) + \frac{1}{2}f(x_{i+1})}{\Delta x} + \mathcal{O}(\Delta x^2)$$

or (3.1.2)

$$f'(x_i) = \frac{\frac{1}{12}f(x_{i-2}) - \frac{2}{3}f(x_{i-1}) + \frac{2}{3}f(x_{i+1}) - \frac{1}{12}f(x_{i+2})}{\Delta x} + \mathcal{O}(\Delta x^4).$$

As mentioned before, we need to introduce ghost cells. If we are able to fill those with exact function values, we can apply central differences on the whole domain, but in case we do not know the exact values we have to use forward and backward

finite differences at the boundaries to avoid those ghost cell values. This results in the following equations at the boundaries for the second-order scheme

$$\begin{aligned} f'(x_1) &= \frac{-f(x_1) + f(x_2)}{\Delta x} + \mathcal{O}(\Delta x) \\ &\text{and} \\ f'(x_n) &= \frac{f(x_{n-1}) - f(x_n)}{\Delta x} + \mathcal{O}(\Delta x). \end{aligned} \tag{3.1.3}$$

This adaptation reduces the order at the boundary by one to order one. For the fourth-order approximation we derive

$$\begin{aligned} f'(x_1) &= \frac{-\frac{3}{2}f(x_1) + 2f(x_2) - \frac{1}{2}f(x_3)}{\Delta x} + \mathcal{O}(\Delta x^2) \\ &\text{and} \\ f'(x_n) &= \frac{\frac{3}{2}f(x_{n-2}) - 2f(x_{n-1}) + \frac{1}{2}f(x_n)}{\Delta x} + \mathcal{O}(\Delta x^2), \end{aligned} \tag{3.1.4}$$

which reduces the fourth-order scheme to a second-order scheme. These boundary approximations are also needed for $f'(x_2)$ and $f'(x_{n-1})$. We could improve the approximation at the boundaries in convergence order by using more data points, but for this a larger stencil would be necessary, which would result in major adaptions in the whole implementation. Because the results for the second-order scheme are still reasonable we accept the trade-off of losing two orders of convergence but being able to keep the general implementation structure. For the second-order derivatives we get the following equations

$$\begin{aligned} f''(x_i) &= \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) \\ &\text{or} \\ f''(x_i) &= \frac{-\frac{1}{12}f(x_{i-2}) + \frac{4}{3}f(x_{i-1}) - \frac{5}{2}f(x_i) + \frac{4}{3}f(x_{i+1}) - \frac{1}{12}f(x_{i+2})}{\Delta x^2} + \mathcal{O}(\Delta x^4). \end{aligned} \tag{3.1.5}$$

Because of the stencil restriction, it is not advisable to use the fourth-order approximation if we do not have exact ghost cell values, because the reduction in order would yield a first-order scheme. Therefore, we only adapt the second-order approximation at the boundaries, which results in a first-order scheme and the following

equations

$$f''(x_1) = \frac{f(x_1) - 2f(x_2) + f(x_3)}{\Delta x^2} + \mathcal{O}(\Delta x)$$
$$\text{and} \tag{3.1.6}$$
$$f''(x_n) = \frac{f(x_{n-2}) - 2f(x_{n-1}) + f(x_n)}{\Delta x^2} + \mathcal{O}(\Delta x).$$

If we have a two-dimensional grid the equations can be similarly applied in y-direction.

3.1.2 Weighted Essentially Non-Oscillatory Solver

A weighted essentially non-oscillatory (WENO) solver is a high-order numerical scheme used for solving hyperbolic PDEs, containing fluxes. It is particularly suitable for problems involving shocks, discontinuities and complex wave structures. The method is called non-oscillatory because it avoids Gibbs phenomenon, which describes the oscillatory behaviour of the Fourier series of a piecewise continuous differentiable periodic function around a discontinuity [34].

The concept behind the WENO solver can be described in the following way. First consider a finite differences framework. To approximate a flux at a given point, the WENO algorithm selects several candidate stencils and computes interpolation polynomials on these stencils. Afterwards, we calculate smoothness indicators for each stencil and assign non-linear weights inversely proportional to the smoothness indicators. This increases the influence of the smooth parts of the solution and down-weights the stencils containing the discontinuities. The final reconstructed flux is then a weighted sum of the candidate interpolations at each stencil. This results in a balance between accuracy and non-oscillatory behaviour in the calculated solution. For an extensive explanation on how the WENO algorithm works and was implemented for the two-compartment model the reader is referred to Externbrink (2022) [22] and for in-depth explanations about the solver in general we refer to Lu et al. (1994) [51], Shu (1997) [67], Shu (2020) [68] and Jiang and Shu (1996) [44].

For completeness a short summary on how to calculate the fifth-order WENO method is given in the following. The goal is to calculate the following approximation

$$f(u)_x\Big|_{x=x_i} = \frac{1}{\Delta x}\left(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\right) + \mathcal{O}(\Delta x^5), \tag{3.1.7}$$

with numerical flux $\hat{f}_{i+\frac{1}{2}} = f(u_{i-s}, \dots, u_{i+t})$, for u smooth on the stencil $(u_{i-s}, \dots, u_{i+t})$. To avoid upwinding we introduce the Lax-Friedrich flux splitting

$$\begin{aligned} f^+(u) &= \frac{1}{2}(f(u) + \alpha u) \\ f^-(u) &= \frac{1}{2}(f(u) - \alpha u), \end{aligned} \tag{3.1.8}$$

with $\alpha = \max_u |f'(u)|$, for u in the relevant range. The next step is to derive the polynomial approximations on each stencil. They are given by

$$\begin{aligned} h^1_{i+\frac{1}{2}} &= \frac{1}{3}\overline{h}_{i-2} - \frac{7}{6}\overline{h}_{i-1} + \frac{11}{6}\overline{h}_i, \text{ for } S_1 = \{I_{i-2}, I_{i-1}, I_i\} \\ h^2_{i+\frac{1}{2}} &= -\frac{1}{6}\overline{h}_{i-1} + \frac{5}{6}\overline{h}_i + \frac{1}{3}\overline{h}_{i+1}, \text{ for } S_2 = \{I_{i-1}, I_i, I_{i+1}\} \\ h^3_{i+\frac{1}{2}} &= \frac{1}{3}\overline{h}_i + \frac{5}{6}\overline{h}_{i+1} - \frac{1}{6}\overline{h}_{i+2}, \text{ for } S_3 = \{I_i, I_{i+1}, I_{i+2}\}, \end{aligned} \tag{3.1.9}$$

where $\overline{h}_i$ are the cell averages given via flux splitting. For calculating $h^-_{i+\frac{1}{2}}$ we use $\overline{h}_i = f^+(u_i)$ for all i in S^+. And for $h^+_{i+\frac{1}{2}}$ we define $\overline{h}_i = f^-(u_i)$ for all i in S^-. To ensure stability via upwinding S^- should be biased one stencil to the right, which results in the following overall stencil $S = \{I_{i-2}, I_{i-1}, I_i, I_{i+1}, I_{i+2}, I_{i+3}\}$ and therefore, in the need of three ghost cells for the discretisation in space. After defining these third-order approximations on each stencil, we now calculate the smoothness parameters for the non-linear weights. They are given via

$$\begin{aligned} \beta_0 &= \frac{13}{12}(\overline{h}_i - 2\overline{h}_{i+1} + \overline{h}_{i+2})^2 + \frac{1}{4}(3\overline{h}_i - 4\overline{h}_{i+1} + \overline{h}_{i+2})^2 \\ \beta_1 &= \frac{13}{12}(\overline{h}_{i-1} - 2\overline{h}_i + \overline{h}_{i+1})^2 + \frac{1}{4}(\overline{h}_{i-1} - \overline{h}_{i+1})^2 \\ \beta_2 &= \frac{13}{12}(\overline{h}_{i-2} - 2\overline{h}_{i-1} + \overline{h}_i)^2 + \frac{1}{4}(\overline{h}_{i-2} - 4\overline{h}_{i-1} + 3\overline{h}_i)^2. \end{aligned} \tag{3.1.10}$$

This yields the following non-linear weights

$$\begin{aligned} \alpha_0 &= \frac{d_0}{(\epsilon + \beta_0)^2}, \qquad \tilde{\alpha_0} = \frac{d_2}{(\epsilon + \beta_0)^2} \\ \alpha_1 &= \frac{d_1}{(\epsilon + \beta_1)^2}, \qquad \tilde{\alpha_1} = \frac{d_1}{(\epsilon + \beta_1)^2} \\ \alpha_2 &= \frac{d_2}{(\epsilon + \beta_2)^2}, \qquad \tilde{\alpha_2} = \frac{d_0}{(\epsilon + \beta_2)^2}, \end{aligned} \tag{3.1.11}$$

with $d_0 = \frac{3}{10}$, $d_1 = \frac{3}{5}$ and $d_2 = \frac{1}{10}$. After normalising the weights

$$\begin{aligned} w_0 &= \frac{\alpha_0}{\alpha_0 + \alpha_1 + \alpha_2}, & \tilde{w}_0 &= \frac{\tilde{\alpha_0}}{\tilde{\alpha_0} + \tilde{\alpha_1} + \tilde{\alpha_2}} \\ w_1 &= \frac{\alpha_1}{\alpha_0 + \alpha_1 + \alpha_2}, & \tilde{w}_1 &= \frac{\tilde{\alpha_1}}{\tilde{\alpha_0} + \tilde{\alpha_1} + \tilde{\alpha_2}} \\ w_2 &= \frac{\alpha_2}{\alpha_0 + \alpha_1 + \alpha_2}, & \tilde{w}_2 &= \frac{\tilde{\alpha_2}}{\tilde{\alpha_0} + \tilde{\alpha_1} + \tilde{\alpha_2}}, \end{aligned} \tag{3.1.12}$$

we finally get the wanted fifth-order approximations

$$\begin{aligned} h^-_{i+\frac{1}{2}} &= w_0\, h^3_{i+\frac{1}{2}} + w_1\, h^2_{i+\frac{1}{2}} + w_2\, h^1_{i+\frac{1}{2}} \\ h^+_{i+\frac{1}{2}} = h^+_{j-\frac{1}{2}} &= \tilde{w}_0\, h^3_{j-\frac{1}{2}} + \tilde{w}_1\, h^2_{j-\frac{1}{2}} + \tilde{w}_2\, h^1_{j-\frac{1}{2}}, \text{ for } j = i+1. \end{aligned} \tag{3.1.13}$$

And therefore, the numerical fluxes

$$\begin{aligned} \hat{f}_{i+\frac{1}{2}} &= \hat{f}^-_{i+\frac{1}{2}} + \hat{f}^+_{i+\frac{1}{2}} = h^-_{i+\frac{1}{2}} + h^+_{i+\frac{1}{2}} \\ \hat{f}_{i-\frac{1}{2}} &= h^-_{i-\frac{1}{2}} + h^+_{i-\frac{1}{2}}. \end{aligned} \tag{3.1.14}$$

For a two-dimensional problem we have to derive the fluxes in both dimensions. We can do so by fixing the x- or y-direction and then solving the resulting one-dimensional problem with the known solver. Using this kind of dimensional-splitting results in a reduction of the approximation order, because the reconstruction methods cannot ensure the previous approximation orders anymore [67]. All the previous explained methods that work in the one-dimensional case, e. g. flux splitting, can be applied in the two-dimensional case as well. They just have to be applied to each dimension separately. The same applies for even higher dimensions.

3.2 Time-Discretisation

After discretising the PDE in space we derive a semi-discrete problem that can be solved by an ordinary differential equation (ODE) time solver. For this we also use a cell-centred uniform grid, which discretises the time domain $[0, T]$ in evenly spaced time steps with distance Δt.

3.2.1 Runge-Kutta Solver

Suitable to the WENO space solver a total variation diminishing (TVD) third-order Runge-Kutta (RK) method was chosen. The method is suitable because it also

prevents spurious oscillations near discontinuities such as sharp gradients or shocks. It is an explicit scheme constructed to preserve or even reduce the total variation of the numerical solution thus avoiding the introduction of new extrema. The method was introduced by Shu and Osher in [44].

For a general ODE

$$u_t = L(u), \tag{3.2.1}$$

with spatial operator $L(u)$ we write down the following third-order Runge-Kutta scheme with three stages [44]

$$\begin{aligned} u^1 &= u^i + \Delta t\, L(u^i) \\ u^2 &= \frac{3}{4}u^i + \frac{1}{4}u^1 + \frac{1}{4}\Delta t\, L(u^1) \\ u^{i+1} = u^3 &= \frac{1}{3}u^i + \frac{2}{3}u^2 + \frac{2}{3}\Delta t\, L(u^2). \end{aligned} \tag{3.2.2}$$

If we add a time-dependent right-hand side term to the ODE we get the following problem

$$u_t = L(u) + g(t). \tag{3.2.3}$$

To solve this equation we adapt the scheme and by evaluating the right-hand side term at the correct intermediate points we derive

$$\begin{aligned} u^1 &= u^i + \Delta t\, (L(u^i) + g(t)) \\ u^2 &= \frac{3}{4}u^i + \frac{1}{4}u^1 + \frac{1}{4}\Delta t\, (L(u^1) + g(t + \Delta t)) \\ u^{i+1} = u^3 &= \frac{1}{3}u^i + \frac{2}{3}u^2 + \frac{2}{3}\Delta t\, (L(u^2) + g(t + \frac{1}{2}\Delta t)). \end{aligned} \tag{3.2.4}$$

We can also add a right-hand side term depending on the concentration u, which results in the following ODE

$$u_t = L(u) + g(u) \tag{3.2.5}$$

and the corresponding Runge-Kutta scheme

$$\begin{aligned} u^1 &= u^i + \Delta t\, (L(u^i) + g(u^1)) \\ u^2 &= \frac{3}{4}u^i + \frac{1}{4}u^1 + \frac{1}{4}\Delta t\, (L(u^1) + g(u^2)) \\ u^{i+1} = u^3 &= \frac{1}{3}u^i + \frac{2}{3}u^2 + \frac{2}{3}\Delta t\, (L(u^2) + g(u^3)). \end{aligned} \tag{3.2.6}$$

3.2.2 IMEX Solver

If we want to solve the advection-diffusion equation, for stability reasons, we need an implicit time solver and therefore cannot use the Runge-Kutta solver which is applied for the two-compartment model. Furthermore, we still want to be able to use the WENO solver in space, this excludes the use of fully implicit solvers because the WENO solver is not linear and thus cannot be written as a matrix, which would make the implementation of a fully implicit solver rather costly, because we would need to use a Newton solver with approximated Jacobian. To avoid this, we use a so-called IMEX (implicit-explicit) solver. This allows us to handle the advection part explicitly, as before, and only the diffusive part is handled implicitly. This idea is an established one and first papers on stability and convergence are already found in the late 1970s [15][78]. Since then IMEX schemes have been widely used, especially in conjunction with spectral methods, for the time integration of spatially discretised PDEs of diffusion-convection type. An introduction can be found in Hundsdorfer and Verwer (2003) [40] and an intensive analysis of the performance of such schemes, with particular attention to the relative performance on the context of fast multigrid algorithms can be found in Asher et al. (1995) [5].
To apply the IMEX solver we first need to split the advection-diffusion equation into two parts. An explicit part f_E and an implicit part f_I

$$u_t = f_E(t,u) + f_I(t,u) \tag{3.2.7}$$

with $f_E(t,u) = \nabla \cdot (V(x,y)\,u(x,y))$ representing the convection term and $f_I(u,t) = \nabla \cdot (D(x,y)\,\nabla u(x,y))$ representing the diffusion. This is an intuitive choice, because an equation with $f_E \equiv 0$ is generally stiff and linear, whereas the system with $f_I \equiv 0$ is not too stiff and is often nonlinear [5].

Furthermore, the diffusion operator is discretised using finite differences and therefore, the discretised operator can be written in matrix form resulting in a matrix-vector multiplication, $f_I(u,t) = A\,u(t)$. When using a Runge-Kutta or any linear multistep time-discretisation scheme it can be represented using the Butcher notation

$$\begin{aligned} z_i &= u_n + \Delta t \sum_{j=1}^{i-1} a_{ij}^E f_E(t_n + c_j\Delta t, z_j) + \Delta t \sum_{j=1}^{i} a_{ij}^I f_I(t_n + c_j\Delta t, z_j) \\ u_{n+1} &= u_n + \Delta t \sum_{j=1}^{s} b_j (f_E(t_n + c_j\Delta t, z_j) + f_I(t_n + c_j\Delta t, z_j)). \end{aligned} \tag{3.2.8}$$

The parameters can be found in the corresponding Butcher tableaus

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{3.2.9}$$

We combine an explicit Runge-Kutta (ERK) method with a diagonally implicit Runge-Kutta (DIRK) method. DIRK methods are defined by the form of matrix A in the Butcher notation. As the name indicates the matrix A is a lower triangular matrix. To be comparable to the explicit Runge-Kutta solver we need a third-order solver. Furthermore, we choose a strong stability preserving (SSP) and A-stable method that uses the same explicit Runge-Kutta solver and an implicit solver that has the same c and b values. An extensive analysis of this method can be found in Gardner et al. (2018) [27].

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{6} & -\frac{1}{3} & \frac{2}{3} \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}. \end{array} \tag{3.2.10}$$

Filling in those parameters into equation 3.2.8 we derive the following system that has to be solved at every time step.

$$\begin{aligned} z_1 &= u_n \\ [I - \Delta t A]\, z_2 &= u_n + \Delta t f_E(t_n, u_n) \\ \left[I - \frac{2}{3}\Delta t A\right] z_3 &= u_n + \Delta t \left[\frac{1}{4} f_E(t_n, u_n) + \frac{1}{4} f_E(t_n + \Delta t, z_2) \right. \\ &\qquad \left. + \frac{1}{6} f_I(t_n, u_n) - \frac{1}{3} f_I(t_n + \Delta t, z_2)\right] \\ u_{n+1} &= u_n + \Delta t \left[\frac{1}{6}\left[f_E(t_n, u_n) + f_I(t_n, u_n)\right] \right. \\ &\qquad + \frac{1}{6}\left[f_E(t_n + \Delta t, z_2) + f_I(t_n + \Delta t, z_2)\right] \\ &\qquad \left. + \frac{2}{3}\left[f_E(t_n + \frac{1}{2}\Delta t, z_3) + f_I(t_n + \frac{1}{2}\Delta t, z_3)\right]\right]. \end{aligned} \tag{3.2.11}$$

If we want to include a source term g, dependent on space and time or on u itself, we can adapt the linear system accordingly

$$\begin{aligned}
z_1 &= u_n \\
[I - \Delta t A]\, z_2 &= u_n + \Delta t \left[f_E(t_n, u_n) + g(t_n, u_n) \right] \\
\left[I - \frac{2}{3}\Delta t A \right] z_3 &= u_n + \Delta t \left[\frac{1}{4} \left[f_E(t_n, u_n) + g(t_n, u_n) \right] \right. \\
&\quad + \frac{1}{4} \left[f_E(t_n + \Delta t, z_2) + g(t_n + \Delta t, z_2) \right] \\
&\quad \left. + \frac{1}{6} f_I(t_n, u_n) - \frac{1}{3} f_I(t_n + \Delta t, z_2) \right] \\
u_{n+1} &= u_n + \Delta t \left[\frac{1}{6} \left[f_E(t_n, u_n) + f_I(t_n, u_n) + g(t_n, u_n) \right] \right. \\
&\quad + \frac{1}{6} \left[f_E(t_n + \Delta t, z_2) + f_I(t_n + \Delta t, z_2) + g(t_n + \Delta t, z_2) \right] \\
&\quad \left. + \frac{2}{3} \left[f_E(t_n + \frac{1}{2}\Delta t, z_3) + f_I(t_n + \frac{1}{2}\Delta t, z_3) \right. \right. \\
&\quad \left. \left. + g(t_n + \frac{1}{2}\Delta t, z_3) \right] \right].
\end{aligned} \tag{3.2.12}$$

This method has been chosen because it can be evaluated at the same intermediate points as the chosen explicit Runge-Kutta time solver, which simplifies the implementation. Furthermore, we also get a third-order solver which makes both optimisations, the parameter identification using the advection-diffusion model and the two-compartment model, more comparable. The solver has been tested thoroughly, see Section 4.3.

3.3 Descent Directions

In Chapter 2 the parameter identification problem was introduced and written as an optimisation problem. In order to solve the optimisation problem we are in need of a descent direction in which we minimise the cost functional. The most intuitive method is the steepest descent method.

3.3.1 Steepest Descent

Given a function j at point c_k. We want to find a descent direction d such that $j(c_k) > j(c_k - \alpha d_k)$ for some step size α. How to derive the step size α is discussed later on in Chapter 3.4. Using Taylor's theorem [57] we write, assuming that j is continuously differentiable,

$$j(c_k - \alpha d_k) = j(c_k) - \alpha\, d_k^T\, \nabla j(c_k) \tag{3.3.1}$$

and derive the necessary condition

$$-d_k^T\, \nabla j(c_k) < 0 \tag{3.3.2}$$

for our descent direction. This condition is fulfilled for $d_k = \nabla j(c_k)$ and thus, our first descent direction is derived. As the name, steepest descent direction, indicates this direction ensures the most rapid decrease. But the steepest descent direction is in general, not the fastest way to reach a stationary point or minimum. Especially for complex problems it tends to reach the stationary point in a 'zig-zag'-line. An example for a function depending on two parameters can be seen in Figure 3.2.

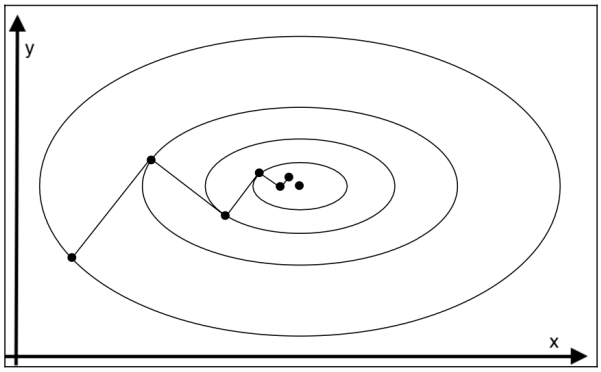


Figure 3.2: Illustration of steepest descent method [57] for a function depending on two parameters x and y.

To summarise, we can say that this method is quite simple to get an optimisation started, because we only need to calculate the gradient and the numerical implementation is quite forward. On the downside the behaviour of always choosing the most rapid descent leads to a large number of steps and therefore long computation times.

3.3.2 Nonlinear Conjugate Gradient Method

The nonlinear conjugate gradient method is used to find a local minimum of a nonlinear function j using its gradient ∇j thus we expect the function to be continuously differentiable and bounded from below [32]. Given a function $j : \mathbb{R}^n \to \mathbb{R}$ of n variables to minimise, the gradient ∇j indicates the direction of maximum increase, one simply steps in the opposite direction, the steepest descent direction with $d_0 = \nabla j(c_0)$.
After finding a descent direction we can perform a line search in this direction until a minimum is found, $\alpha_0 = \arg\min_\alpha j(c_0 - \alpha\, d_0)$. And update the parameters $c_1 = c_0 - \alpha\, d_0$. After the first step we use a descent direction composed of the previous descent directions combined with the new gradient. This yields the following algorithm.

1. Calculate the steepest descent direction: $s_n = \nabla j(c_n)$
2. Compute β_n according to one of the formulas below
3. Update the conjugate gradient direction: $d_n = s_n + \beta_n\, d_{n-1}$
4. Perform a line search: optimise $\alpha_n = \arg\min_\alpha j(c_n - \alpha\, d_n)$
5. Update the position: $c_{n+1} = c_n - \alpha_n\, d_n$
6. Repeat until change in j or $||\nabla j||$ too small

A pure quadratic function reaches a minimum within n iterations, a non quadratic function needs more steps [32]. The search direction d_n has to be set to the steepest descent direction at least every n steps. There are a lot of possibilities to calculate β_n, two of them are the Fletcher and Reeves parameter with $\beta_n^{FR} = \frac{\langle s_n, s_n\rangle}{\langle s_{n-1}, s_{n-1}\rangle}$ and the Dai and Yuan parameter which is defined as $\beta_n^{DY} = \frac{\langle s_n, s_n\rangle}{\langle -d_{n-1}, s_n - s_{n-1}\rangle}$ [32]. Both the FR and the DY parameter are prone to jamming. Which means that the algorithm takes a lot of small steps without making progress towards the minimum. We try to avoid this by changing the descent direction to the steepest descent if $||s_k - s_{k-1}||^2_{L_2(Q)} \leq \text{tol}$, with $\text{tol} = 5e^{-4}$. This is checked for each parameter individually. To ensure that we actually find a descent direction we need to implement our line search algorithm with strong Wolfe conditions or an exact line search [57].

3.3.3 Newton's method

Newton's method is a different approach to find the minimum of a function. Instead of only using the gradient, we also take the second-order derivative of the objective function into account for the descent direction.

Consider a twice continuous differentiable function f in one variable. Using the Taylor theorem we write $j(c_k + \epsilon) = j(c_k) + j'(c_k)\,\epsilon + \frac{1}{2} j''(c_k)\,\epsilon^2$ which is minimised, when $\frac{d}{d\epsilon} j(c_k + \epsilon) = 0$. By plugging in we derive

$$\frac{d}{d\epsilon}[j(c_k) + j'(c_k)\,\epsilon + \frac{1}{2} j''(c_k)\,\epsilon^2] = j'(c_k) + j''(c_k)\,\epsilon$$
$$\epsilon = -\frac{j'(c_k)}{j''(c_k)}$$

which gives us a descent direction ϵ. Therefore, we can define

$$c_{k+1} = c_k - \frac{j'(c_k)}{j''(c_k)} \tag{3.3.3}$$

or generalised to n dimensions

$$c_{k+1} = c_k - H(c_k)^{-1}\,\nabla j(c_k). \tag{3.3.4}$$

Newton's method is more efficient than the steepest descent method because it approximates the cost functional at c_k with a paraboloid and then steps towards the minimum of this paraboloid. This results in far fewer steps needed before reaching a minimum. But Newton's method also has some downsides. Compared to the steepest descent method it is sensitive to initial conditions, especially when the objective function is non-convex [57]. If the algorithm is started far away from the solution it steps may not even form a descent direction [57]. Thus, in general we only get local convergence which can be extended to global convergence if the matrices H_k have a bounded condition number and are positive definite and the step lengths satisfy the Wolfe conditions [57]. Furthermore, for non-convex functions it is not ensured that we actually find a minimum, it could also lead to a saddle point or maximum. Here an adaption of the algorithm, as presented in Truong et al. (2023) [77], can ensure that saddle points are avoided. Another aspect is that the Newton method is computationally expensive, because the Hessian matrix has to be calculated and inverted. This increases the computationally costs of calculating the gradient from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$ ($\mathcal{O}(n^2)$ for the calculation of the Hessian and $\mathcal{O}(n^3)$

for the inverse Hessian). That is where the Quasi-Newton methods come into play. Instead of calculating the Hessian matrix we try to approximate its behaviour, while only needing the gradient of the cost functional.

3.3.4 Quasi-Newton Method

We need to find a B with

$$c_{k+1} = c_k - B_k^{-1} \nabla j(c_k), \tag{3.3.5}$$

which needs to fulfill the *quasi-Newton* or *secant equation*

$$B_{k+1}(c_{k+1} - c_k) = \nabla j(c_{k+1}) - \nabla j(c_k). \tag{3.3.6}$$

This can be obtained from the first-order Taylor expansion of $\nabla j(c_{k+1})$

$$\nabla j(c_{k+1}) = \nabla j(c_k) - \alpha B_{k+1} p_k = \nabla j(c_k) + B_{k+1}(c_{k+1} - c_k). \tag{3.3.7}$$

Remember that $c_{k+1} = c_k - \alpha\, p_k$ with descent direction $p_k = B_k^{-1} \nabla j(c_k)$. For a better readability we define $\Delta c_k = c_{k+1} - c_k$ and $y_k = \nabla j(c_{k+1}) - \nabla j(c_k)$ and therefore derive

$$B_{k+1} \Delta c_k = y_k, \tag{3.3.8}$$

which is also known as the *secant equation*. By construction this matrix B_{k+1} is positive definite, the reader is encouraged to check by multiplying with Δc_k^T. This also yields the so called *curvature condition* which states the following

$$\Delta c_k^T\, y_k > 0. \tag{3.3.9}$$

To ensure that the curvature condition holds we need to apply Wolfe or strong Wolfe conditions [57].

BFGS

If we have more than one dimension the quasi-Newton condition does not uniquely specify the Hessian estimate B. Therefore, we need to define further constraints on B to derive a formula for it. The most known and used method for constraining B is the BFGS method named after the creators Broyden [11], Fletcher [25], Goldfarb [29] and Shanno [63], who independently came up with the formula and restriction in 1970.

$$\begin{aligned} \min_{B_{k+1}} \quad & ||B_{k+1} - B_k|| \\ \text{subject to} \quad & B_{k+1}^T = B_{k+1} && \text{(symmetry)} \\ & B_{k+1}\Delta c_k = y_k && \text{(quasi-Newton condition)} \end{aligned}$$

Here the Frobenius norm is used. Note that this can be formulated for B_{k+1}^{-1} in the same manner. This system can be solved by updating B_k with two rank one matrices U and V, for further explanations the reader is referred to Nocedal and Wright (1999) [57] and Johnson (2019) [45].

$$B_{k+1} = B_k + a\ uu^T + b\ vv^T \tag{3.3.10}$$

A natural choice are $u = y_k$ and $v = B_k\Delta c_k$. This yields

$$\begin{aligned} & B_{k+1}\Delta c_k + a\ y_k y_k^T + b\ B_k\Delta c_k\Delta x_k^T B_k^T\Delta c_k = y_k \\ \iff\ & y_k(1 - a\ y_k^T\Delta x_k) = B_k\,\Delta c_k(1 + b\ \Delta c_k^T B_k^T\Delta c_k) \\ & \Rightarrow a = \frac{1}{y_k^T\,\Delta c_k} \quad \wedge \quad b = -\frac{1}{\Delta c_k^T B_k^T\Delta c_k}. \end{aligned}$$

And finally the BFGS formula

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T\Delta c_k} - \frac{B_k\Delta c_k\Delta c_k^T B_k^T}{\Delta c_k^T B_k\Delta c_k}. \tag{3.3.11}$$

In order to include this into the optimisation we define the inverse of this matrix $B_k^{-1} = H_k$. With this definition we derive the descent direction $d_k = H_k\nabla j(c_k)$. The formulation of the inverse matrix H_k can be derived using the Sherman–Morrison–Woodbury formula [64]

$$H_{k+1} = (I - \rho_k\Delta c_k y_k^T)H_k(I - \rho_k y_k\Delta c_k^T) + \rho_k\Delta c_k\Delta c_k^T, \tag{3.3.12}$$

with $\rho_k = \frac{1}{y_k^T\Delta c_k}$ and I the identity matrix. For the first step in the optimisation we have to define a H_0. This can simply be the identity matrix or a multiple of the identity matrix reflecting the scaling of the variables. Other choices for example calculating an approximation of the Hessian matrix at c_0 using finite difference is possible as well but numerically more costly. A formula that is often quite effective is $H_0 = \frac{y_k^T\Delta c_k}{y_k^T y_k} I$ [57]. Putting everything together we derive the following algorithm for the BFGS method, see Algorithm 3.

Algorithm 3 BFGS Method

Require: starting point c_0, convergence tolerance $\epsilon > 0$, inverse Hessian approximation H_0

1: $k = 0$
2: **while** $||\nabla j(c_k)|| > \epsilon$ **do**:
3: Compute search direction $d_k = H_k \nabla j(c_k)$
4: Set $c_{k+1} = c_k - \alpha\, d_k$ where α_k is computed from a line search procedure satisfying the Wolfe conditions (3.4.2)
5: Define $\Delta c_{k+1} = c_{k+1} - c_k$ and $y_k = \nabla j(c_{k+1}) - \nabla j(c_k)$
6: Compute H_{k+1} using (3.3.12)
7: $k \to k + 1$
8: **end while**

L-BFGS

If we take a closer look at the BFGS Algorithm 3 it becomes apparent that the matrix H_k is huge for fine space discretisations ($n_x\, n_y \times n_x\, n_y$). To avoid having to store the whole matrix there is another algorithm, called the L-BFGS, where the L stands for limited-memory. As the name suggests the memory usage is optimised. The savings in memory are achieved by storing only individual vectors rather than the entire matrix. That in turn means that we only use the curvature information from the most recent iterations to construct the Hessian approximation. Assuming, that the curvature information from earlier iterations is less likely to be relevant to the actual behaviour of the Hessian at the current iteration. We start by defining H_{k+1} in a slightly different way

$$H_{k+1} = V_k^T H_k V_k + \rho_k \Delta c_k \Delta c_k^T \quad \text{with} \quad V_k = I - \rho_k y_k \Delta c_k^T. \tag{3.3.13}$$

As mentioned before we store a modified version of H_k implicitly, by storing m vector pairs $\{\Delta c_i, y_i\}$. Implicitly means that we actually calculate the product $H_k \nabla j(c_k)$, instead of only H_k. This product can be obtained by performing a sequence of inner products and vector summations involving the pairs $\{\Delta c_i, y_i\}$ and $\nabla j(c_k)$. To update the vector pairs, the oldest vector pair is replaced by the new vector pair $\{\Delta c_k, y_k\}$. Practical experience has shown that satisfactory results are often produced for m between 3 and 20 [57]. This means that at the current iteration the set of vector pairs is given by $\{\Delta c_i, y_i\}$ for $i = \{k - m, \cdots, k - 1\}$. Because we do not simply update the matrix H_k to derive H_{k+1} we do not only need

a H_0 to start the algorithm but instead a H_k^0 for every iteration step. A typical choice proven to be effective is

$$H_k^0 = \frac{\Delta c_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} I. \tag{3.3.14}$$

This gives us the opportunity to write H_k recursively as

$$\begin{aligned} H_k =& (V_{k-1}^T \cdots V_{k-m}^T) H_k^0 (V_{k-m} \cdots V_{k-1}) \\ &+ \rho_{k-m} (V_{k-1}^T \cdots V_{k-m+1}^T) \Delta c_{k-m} \Delta c_{k-m}^T (V_{k-m+1} \cdots V_{k-1}) \\ &+ \rho_{k-m+1} (V_{k-1}^T \cdots V_{k-m+2}^T) \Delta c_{k-m+1} \Delta c_{k-m+1}^T (V_{k-m+2} \cdots V_{k-1}) \\ &+ \cdots \\ &+ \rho_{k-1} \Delta c_{k-1} \Delta c_{k-1}^T. \end{aligned}$$

Now we have all the information needed to write down the Algorithm 4 needed to calculate $H_k \nabla f(x_k)$.

Algorithm 4 L-BFGS two loop recursion [57]

Require: $\{\Delta c_i, y_i\}$ for $i = \{k-m, \cdots k-1\}$, $\nabla j(c_k)$ and H_k^0
Ensure: $r = H_k \nabla j(c_k)$
1: $q = \nabla j(c_k)$
2: **for** $i = \{k-1, k-2, \cdots, k-m\}$ **do**:
3: $\quad \alpha_i = \rho_i \Delta c_i^T q$
4: $\quad q = q - \alpha_i y_i$
5: **end for**
6: $r = H_k^0 q$
7: **for** $i = \{k-m, \cdots, k-2, k-1\}$ **do**:
8: $\quad \beta = \rho_i y_i^T r$
9: $\quad r = r + \Delta c_i (\alpha_i - \beta)$
10: **end for**

Putting everything together the limited-memory BFGS algorithm can be stated formally as follows, see Algorithm 5.

Algorithm 5 L-BFGS [57]

Require: starting point c_0, integer $m > 0$

$k = 0$
repeat
  Choose H_k^0; ▷ for example by using (3.3.14)
  Compute $p_k = H_k \nabla j(c_k)$ from Algorithm 4
  Set $c_{k+1} = c_k - \alpha_k d_k$, where α_k is computed from a line search procedure satisfying the Wolfe conditions (3.4.2)
  if $k > m$ **then**
    Discard the vector pair $\{\Delta c_{k-m}, y_{k-m}\}$ from storage
  end if
  Compute and save Δc_{k+1} and y_{k+1}
  $k \to k + 1$
until convergence

3.4 Line Search Methods

Line search methods are used to find an appropriate step size that improves the objective function along a given search direction. The primary challenge is balancing between taking sufficiently large steps to ensure fast convergence and avoiding steps that are too large, which may lead to overshooting the optimal solution or numerical instability. To achieve this, various criteria and strategies have been developed, including exact and inexact line search methods. Exact line search aims to find the step size that minimises the function along the search direction precisely, but it can be computationally expensive or infeasible in practice. In contrast, inexact line search methods, such as the Wolfe conditions [80], and Armijo rule [4], provide practical and efficient criteria to accept a step size that guarantees sufficient decrease and curvature conditions thus ensuring convergence of iterative optimisation algorithms. The Armijo rule and Wolfe conditions are explained in more detail in the following sections.

3.4.1 Armijo Rule

During the optimisation we want to find the step size α such that we get a sufficient reduction in the following sense. Let $j(c)$ be our objective function, c_k the point

we are at, d_k the descent direction at step k and α the step size. Then, the Armijo condition is given by

$$j(c_k - \alpha d_k) \leq j(c_k) - \rho\,\alpha\, d_k^T\, \nabla j(c_k) \tag{3.4.1}$$

for some $\rho \in (0, 1)$. We have to be careful when using this condition because it is always satisfied for sufficiently small values of α. Using these small steps to reduce the objective function is not efficient and would result in massive computation time and costs.

The meaning of equation 3.4.1 is demonstrated in Figure 3.3. The solid line represents the graph of $j(c_k - \alpha d_k)$ for $\alpha > 0$ and the dashed line represents the right-hand side of equation 3.4.1 ($j(c_k) - \rho\,\alpha\, d_k^T\, \nabla j(c_k)$). In Figure 3.3 we can see that the condition is fulfilled for $\alpha \in [0, a] \cup [b, c]$. Due to the general requirement $\nabla j(c_k)^T\, d_k < 0$ and the (Lipschitz-) continuous differentiability of j the existence of $\alpha > 0$ is ensured [35]. When calculating α we start with $\alpha^0 = \beta^0 = 1$. Then, we set $\beta \in (0, 1)$ and $\alpha = \beta^l$ for $l = 0, 1, 2, \ldots$, we stop when equation 3.4.1 holds true.

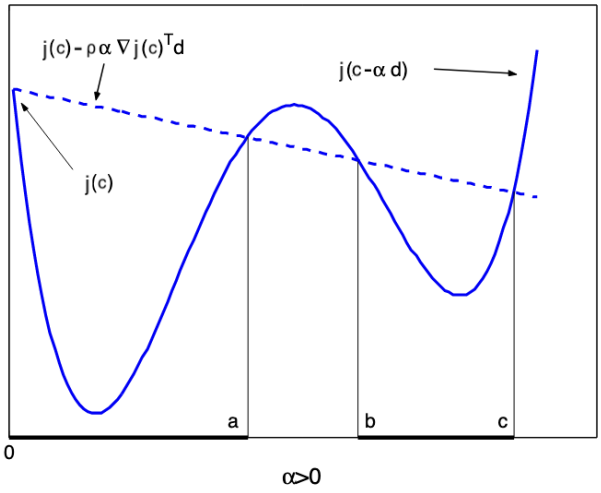


Figure 3.3: Illustration of the Armijo rule [35] with objective function $j(c)$ and the dotted line representing the Armijo condition.

3.4.2 Strong Wolfe Conditions

The goal of a line search algorithm is to find the step size α, which solves the minimisation $\arg\min_\alpha j(c_k - \alpha\, d_k)$. First we define $\Phi(\alpha) = j(c_k - \alpha\, d_k)$ with $\alpha > 0$.

Ideally we would like to find the global minimum of Φ, but because we do not have information about $\Phi(\alpha)$ and each evaluation of $\Phi(\alpha)$ and $\Phi'(\alpha)$ is expensive, we cannot find the global minimum without substantial investments in computing. Therefore, instead of using an exact line search, an inexact line search, which gives an acceptable decrease in f, while keeping the cost of the line search moderate, is implemented. The strong Wolfe condition contains of two equations

$$\begin{aligned} j(c_k - \alpha\, d_k) &\leq j(c_k) - \tilde{c}_1\, \alpha\, \nabla j(c_k)^T\, d_k, \quad \tilde{c}_1 \in (0,1) \\ |\nabla j(c_k - \alpha\, d_k)^T\, d_k| &\leq \tilde{c}_2\, |\nabla j(c_k)^T\, d_k|, \quad \tilde{c}_2 \in (\tilde{c}_1, 1). \end{aligned} \tag{3.4.2}$$

The first equation guarantees a sufficient decrease and is also known as Armijo condition, which is explained in 3.4.1. The second equation is also known as curvature condition and prevents the algorithm to take tiny steps, which would always fulfill the Armijo condition. The curvature condition demands that the magnitude of the derivative of Φ, which is defined by $\frac{d\Phi(\alpha)}{d\alpha} = \nabla j(c_k - \alpha\, d_k)^T\, (-d_k)$, at the desired α be at least less than the slope of the linear approximation at x_k. Furthermore, we use absolute values, which yields the strong Wolfe conditions, to avoid large step sizes that skip far past a stationary point. Some overshooting from a stationary point can still happen, but the extent is minimised. It can be shown that as long as $0 < \tilde{c}_1 < \tilde{c}_2 < 1$, there exists a step length satisfying the strong Wolfe conditions, for a continuously differentiable function f that is bounded from below, see Lemma 3.1 in [57]. In general, a good choice for the constant $\tilde{c}_1$ and $\tilde{c}_2$ are $\tilde{c}_1 = 10^{-4}$ and a much larger $\tilde{c}_2 \in (0.1, 0.9)$. With $\tilde{c}_2 = 0.1$ for the nonlinear conjugate gradient and $\tilde{c}_2 = 0.9$ for Newton or Quasi-Newton methods [57]. Furthermore, global convergence can be shown for the FR-parameter if $\tilde{c}_2 \leq 0.5$ using strong Wolfe conditions. This is a significant restriction on the choice of the line search parameter. This restriction is not needed for the DY-parameter for which global convergence can be shown if the Lipschitz assumptions hold. For further information the reader is referred to Hager and Zang (2006) [32].

The pseudocode for an implementation can be found below in Algorithm 6 and 7. The algorithm is split into two parts. The first part, Algorithm 6, selects an interval which includes the optimal step size and the second part, Algorithm 7, then finds the optimal step size contained in this interval. For the implementation we need to define a starting point α_0, a maximal step size α_{max} and a minimal step size α_{min} for which we assume the step size to be zero and which implies that we reached an optimal point.

Algorithm 6 Line Search Algorithm - finds a step length α fulfilling the strong Wolfe conditions

Require: α_0, α_{max}, α_{min}

Ensure: α_{yay}

Choose $\alpha_1 \in (\alpha_0, \alpha_{max})$
repeat ▷ i-loop
  Evaluate $j(c_k - \alpha_i d_k)$
  if Armijo condition (1) does not hold **or** $j(c_k - \alpha_i d_k) \geq j(c_k - \alpha_{i-1} d_k)$ **then**
    $\alpha_{yay} = zoom(\alpha_{i-1}, \alpha_i) \rightarrow$ break
  end if
  Evaluate $\nabla j(c_k - \alpha_i d_k)^T d_k$
  if Sufficient decrease condition (2) holds **then**
    $\alpha_{yay} = \alpha_i \rightarrow$ break
  end if
  if $\nabla j(c_k - \alpha_i d_k)^T d_k \leq 0$ **then**
    $\alpha_{yay} = zoom(\alpha_i, \alpha_{i-1}) \rightarrow$ break
  end if
  Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{max})$
until step size α_{yay} found or smaller than α_{min} (then set $\alpha_{yay} = 0$)

The line search Algorithm 6 uses the knowledge that the interval (α_{i-1}, α_i) contains step lengths satisfying the strong Wolfe conditions if one of the following conditions is satisfied

i α_i violates the Armijo or sufficient decrease conditions

ii $\Phi(\alpha_i) \geq \Phi(\alpha_{i-1})$

iii $\Phi'(\alpha_i) \geq 0$.

Because d_k is a descent direction we know that $\Phi'(0) < 0$. This information yields that the graph of the cost functional is decreasing from zero onwards. If one of the three conditions hold we know that the graph must have increased in between or is increasing at the moment. Therefore, there must be a minimum in (α_{i-1}, α_i).

Algorithm 7 Zoom-Algorithm - Interpolate (using quadratic, cubic, or bisection) to find a trial step length α between α_{lo} and α_{hi}

Require: α_{lo}, α_{hi}
Ensure: α_{yay}
Evaluate $j(c_k - \alpha_{lo} d_k)$
repeat ▷ m-loop
  Evaluate $j(c_k - \alpha_m d_k)$
  if Armijo condition (1) does not hold **or** $j(c_k - \alpha_m d_k) \geq j(c_k - \alpha_{lo} d_k)$ **then**
    $\alpha_{hi} = \alpha_m$
  else
    Evaluate $\nabla j(c_k - \alpha_m d_k)^T d_k$
    if Sufficient decrease condition (2) holds **then**
      $\alpha_{yay} = \alpha_m \rightarrow$ break
    end if
    if $\nabla j(c_k - \alpha_m d_k)^T d_k(\alpha_{hi} - \alpha_{lo}) \leq 0$ **then**
      $\alpha_{hi} = \alpha_{lo}$
    end if
    $\alpha_{lo} = \alpha_m$
    Evaluate $j(c_k - \alpha_{lo} d_k)$
  end if
until step size α_{yay} found or too small ($\alpha_{yay} = 0$)

3.5 Leray-Projection

The Leray-Projection can be informally seen as the projection of a velocity field onto a divergence-free vector field. The Leray-Projection $\mathbb{P}$ is defined by

$$\mathbb{P}(u) = u - \nabla \Delta^{-1}(\nabla \cdot u), \tag{3.5.1}$$

for vector field $u = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$, with flow in x- (u_x) and y-direction (u_y). Here $\nabla \cdot u$ is the divergence, defined by $\nabla \cdot u = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}$. One can show that a given vector field u can be decomposed via the so called Helmholtz-Hodge decomposition

$$u = \nabla q + v \text{ with } \nabla \cdot v = 0. \tag{3.5.2}$$

For more details of existence and uniqueness of this decomposition the reader is referred to Mishra and Weber (2016) [54]. For the Leray-Projector we can see that

$$\mathbb{P}(u) = v. \tag{3.5.3}$$

To calculate the Leray-Projection we first need to solve the Poisson problem

$$\Delta\xi = \nabla \cdot u. \tag{3.5.4}$$

Afterwards, we can derive the divergence-free velocity field via

$$\mathbb{P}(u) = u - \nabla\xi. \tag{3.5.5}$$

The numerical derivative can be calculated via a finite difference scheme [26]. When programming the Leray-Projection several problems can occur. First of all the Poisson problem has to be solved. We implemented second-order central finite differences. Here one must be careful with the boundary conditions. In general, for $\Delta\xi = \nabla \cdot u$ zero Dirichlet boundary conditions, $\xi = 0$ on $\partial\Omega$, work quite well if the area at the boundaries is constant and does not vary. If we have a flow at the boundary, we need to use von-Neumann boundary conditions. They need to fulfill $\partial_n\xi = n \cdot u$. Here one needs to be especially careful with the signs of the normal derivative because mistakes in sign result in instable solutions. The next step is to calculate the derivative of the solution of the Poisson problem ($\nabla\xi$). Because we do not calculate the values of the ghost cells for ξ calculating the derivative with central finite differences does not work. At the boundaries we need to apply forward and backward finite differences for the left and right boundary, respectively. The same holds true for the lower and upper boundary. When using the Leray-Projection we do not solely get velocity fields with less divergence but also a smoothened velocity field, where high velocity areas are reduced and spread out.

3.6 Poisson Solver

To solve the Poisson problem required in the Leray-Projection, we discretise the domain space $\Omega = [a, b] \times [c, d]$. As before we use a uniform and cell-centred mesh. We discretise with n cells in x-direction and m cells in y-direction. Furthermore, the distance between cell centres in x- and y-direction are Δx and Δy respectively.

The general Poisson equation is given by

$$\Delta u = \frac{\partial^2 u}{(\partial x)^2} + \frac{\partial^2 u}{(\partial y)^2} = g(x,y), \quad (x,y) \in [a,b] \times [c,d]. \tag{3.6.1}$$

For the discretised (second-order) system we derive

$$(\Delta u)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = g(x_i, y_j). \tag{3.6.2}$$

Expanding the Poisson Difference Equation 3.6.2 yields

$$\Delta y^2(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \Delta x^2(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) = \Delta x^2\, \Delta y^2\, g(x_i, y_j). \tag{3.6.3}$$

Now we want to use this equation to derive a matrix A such that

$$A\, u = \Delta x^2\, \Delta y^2\, g, \tag{3.6.4}$$

where u and g are discretised and flattened matrices. In the following we write $g_{i,j}$ instead of $g(x_i, y_j)$. We define A in the following way

$$\begin{aligned} A &= \begin{pmatrix} T & \Delta y^2 I & & & \\ \Delta y^2 I & T & \Delta y^2 I & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta y^2 I & T & \Delta y^2 I \\ & & & \Delta y^2 I & T \end{pmatrix} \\ T &= \begin{pmatrix} -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & & \\ \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 \\ & & & \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 \end{pmatrix}. \end{aligned} \tag{3.6.5}$$

Here I is the identity matrix of size $m \times m$. Matrix T is also of size $m \times m$ and matrix A has size $n\,m \times n\,m$. For different boundary conditions we have to adapt the linear equation system.

If we have Dirichlet boundary conditions we have to adapt the right-hand side g with some vector b which includes the values at the boundary conditions. This means that, using *Matlab-notation*, $b(0:m)$ contains the values at the left boundary, $b((n-1)m : nm)$ contains the values from the right boundary, $b(i\,m)$ contains the values at the lower boundary and $b((i+1)m - 1)$ contains the values at the upper boundary, for i from 0 to n. When implementing these equations the programmer must pay particular attention to the corner cells to really subtract both sides. The resulting vector b is subtracted from the right-hand side of the equation before solving the linear system of equations, $A\, u = \Delta x^2\, \Delta y^2\, g - b$.

For periodic boundary conditions the changes are not applied to the right-hand side of the equation but rather to the matrix A itself. This yields the following new matrix

$$
\begin{aligned}
A_{per} &= \begin{pmatrix} T_p & \Delta y^2\, I & & & -\Delta y^2\, I \\ \Delta y^2\, I & T_p & \Delta y^2\, I & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta y^2\, I & T_p & \Delta y^2\, I \\ -\Delta y^2\, I & & & \Delta y^2\, I & T_p \end{pmatrix} \\
T_p &= \begin{pmatrix} -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & & -\Delta x^2 \\ \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 \\ -\Delta x^2 & & & \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 \end{pmatrix}.
\end{aligned}
\tag{3.6.6}
$$

Finally, the linear system $A_{per}\, u = \Delta x^2 \Delta y^2\, g$ is solved.

Lastly, we consider von-Neumann boundary conditions. Here, we have to adapt the matrix and the right-hand side. Furthermore, we introduce ghost cells $(u_{-1,i}, u_{n+1,i}, u_{i,-1}, u_{i,n+1})$ at the boundaries (left, right, down, up), respectively. Therefore, at the left border we derive

$$
\begin{aligned}
\Delta y^2(u_{-1,j} - 2u_{0,j} + u_{1,j}) + \Delta x^2(u_{0,j-1} - 2u_{0,j} + u_{0,j+1}) &= \Delta x^2\, \Delta y^2\, g_{0,j} \\
-\frac{u_{0,j} - u_{-1,j}}{\Delta x} &= h_{-\frac{1}{2},j},
\end{aligned}
\tag{3.6.7}
$$

where $h(x, y)$ describes the value of the derivative at the boundary. The second equation is given by the von-Neumann condition and the negative sign is needed because the normal derivative is pointing outwards. Putting everything together we get the following equation for the left boundary

$$
\Delta y^2(-u_{0,j} + u_{1,j}) + \Delta x^2(u_{0,j-1} - 2u_{0,j} + u_{0,j+1}) = \Delta x^2\, \Delta y^2\, g_{0,j} - \Delta x\, \Delta y^2\, h_{-\frac{1}{2},j}. \tag{3.6.8}
$$

Now we do the same for the right, lower and upper side and derive the following

matrices A_N,T_1 and T

$$
\begin{aligned}
A_N &= \begin{pmatrix} T_1 & \Delta y^2\, I & & & \\ \Delta y^2\, I & T & \Delta y^2\, I & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta y^2\, I & T & \Delta y^2\, I \\ & & & \Delta y^2\, I & T_1 \end{pmatrix} \\
T_1 &= \begin{pmatrix} -\Delta x^2 - \Delta y^2 & \Delta x^2 & & & \\ \Delta x^2 & -2\Delta x^2 - \Delta y^2 & \Delta x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x^2 & -2\Delta x^2 - \Delta y^2 & \Delta x^2 \\ & & & \Delta x^2 & -\Delta x^2 - \Delta y^2 \end{pmatrix} \\
T &= \begin{pmatrix} -\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & & \\ \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x^2 & -2\Delta x^2 - 2\Delta y^2 & \Delta x^2 \\ & & & \Delta x^2 & -\Delta x^2 - 2\Delta y^2 \end{pmatrix}.
\end{aligned}
\tag{3.6.9}
$$

The values of the function g, as in the Dirichlet case, are stored in a flattened matrix b which we subtract from the right-hand side g. Leaving us with solving the linear system $A_N\, u = \Delta x^2\, \Delta y^2\, g - b$.

Chapter 4

Code Verification

Testing code is an essential part of computational mathematics. The programmer has to be able to trust the numerical values in order to evaluate and interpret the results. We start by giving a simplified overview of the code including the five major steps needed and the corresponding solvers, for this see Figure 4.1. The code was implemented in *Python* and therefore to verify the code *Python* specific testing frameworks were used.

Figure 4.1 visualises the different stages of solving a parameter identification problem. We can see that the step zero and five are setting the starting parameters or updating the parameters for the following cycle, because of its simplicity, this does not require additional testing. The second and third step of the parameter identification code consist in solving PDEs. The detailed test scenarios and results for the explicit third-order Runge-Kutta and fifth-order WENO solver can be found in Section 4.2 and an analysis of the implicit third-order IMEX solver can be found in Section 4.3. Other solvers used include derivatives calculated via finite differences, integrals derived using the quadrature rule and the Leray-Projection which makes use of a Poisson solver. How those solvers were tested and a general introduction to software testing can be found in the following Section 4.1. We complete this chapter by verifying the implemented gradient using the Taylor-Test in Section 4.4.

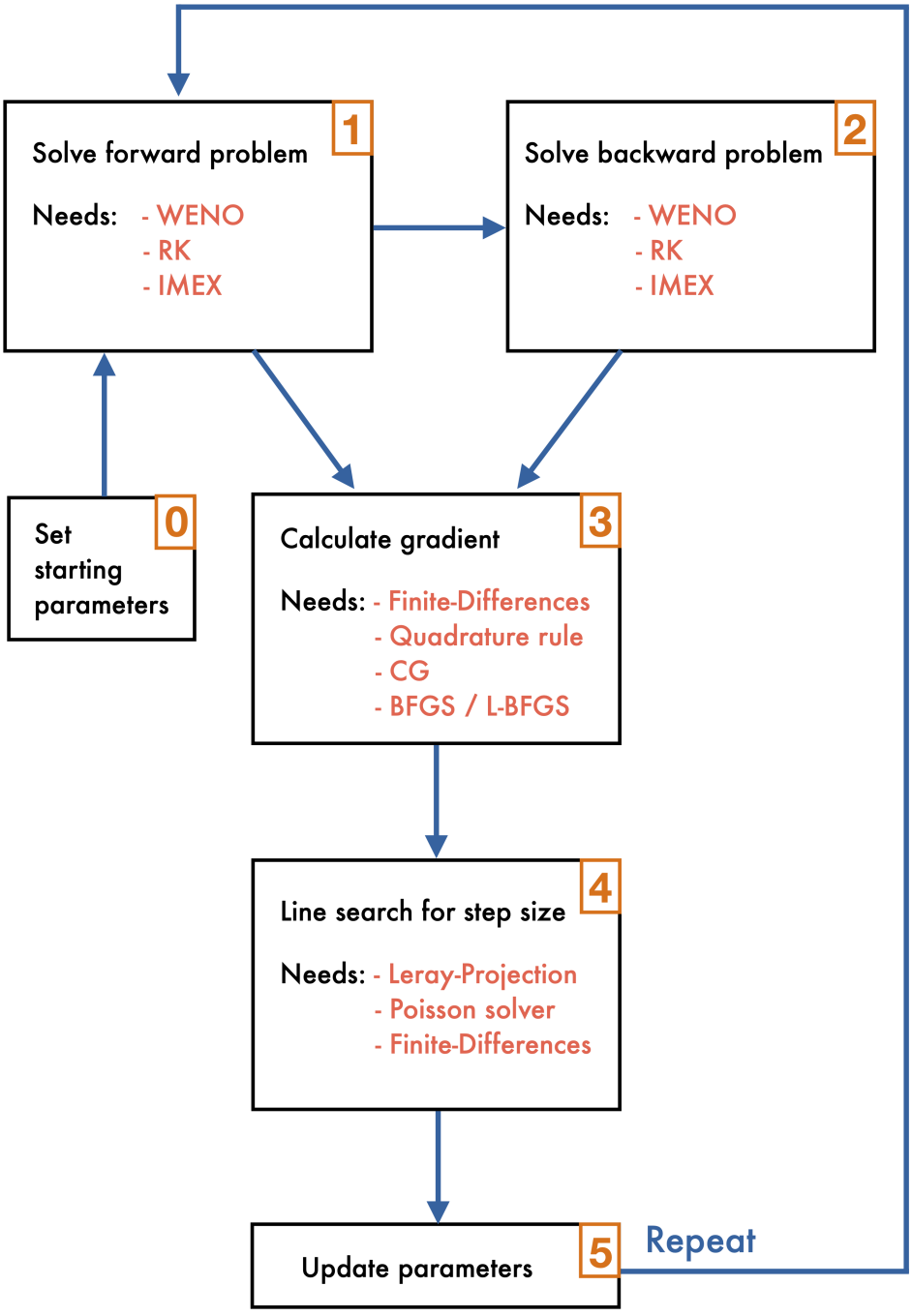


Figure 4.1: A simplified illustration of the parameter identification code, including the five major steps and the used solvers.

4.1 Verification Methods

When evaluating code, one must verify not only its mathematical correctness but also that its functions and classes behave as intended. The programmer should confirm that each function or class returns a variable of the correct type and shape. For instance, if an array is returned, she should verify that it has the correct size. The type checking can be done using *mypy*, for more information the reader is referred to Lehtosalo (2025) [49]. The other tests were implemented using *pytest*, which is a framework that makes it easy to write small and readable tests. It can be scaled to support complex functional testing and can be included in a git-pipeline [48] [60]. Tests should always be written with an expected solution in mind. Otherwise the tester is prone to adapting the test to give the desired result instead of debugging the code correctly. More detailed explanations on how to properly write tests for software can be found in numerous books, including Jorgensen and DeVries (2022) [46] and Feathers et al. (2015) [24].

After checking the software in general, the mathematical output was tested using problems with analytic solutions. For testing derivatives and integrals, those are easy to find. In our case the solvers were first tested on constant functions, then on polynomials of higher order and lastly on a composition of trigonometric functions or trigonometric functions and polynomials. Regarding the two-dimensional case, a particular focus was placed on testing both directions individually. Using the analytic solution it was checked that the errors are sufficiently low and numerically converge in the expected order. Furthermore, the different possibilities for setting ghost cells were tested and the numerically calculated order of convergence compared to the expected order of convergence, see Section 3.1.1. For the Leray-Projection, we started by testing the Poisson solver. As before we chose an analytic solution for testing, which can be easily done by setting the right-hand side of the equation to the divergence of the gradient of the wanted solution. The numerical convergence was checked individually for all implemented boundary conditions, including von-Neumann boundary conditions, Dirichlet boundary conditions and combinations of both. After successfully testing the Poisson solver the Leray-Projection was tested by first assuring that the divergence stays zero if applied on a divergence free velocity field, before assuring that the divergence of a velocity field is indeed significantly reduced if not eliminated by the application of the projection and lastly by testing

the numerical convergence order which should be two - and indeed is.

4.2 Verification of Runge-Kutta and WENO Solver

Because of their importance to the code, the mathematical testing of the space and time solvers will be explained in more detail in the following sections. The fifth-order WENO and the third-order RK solver have been implemented and tested in Externbrink (2022) [22], an excerpt of the test cases will be presented in the following. To evaluate the solvers a problem with a given analytic solution was chosen to derive their numerical convergence order. The numerical convergence order was calculated for the l_2-norm and l_∞-norm which are defined as $(\sum_{i,j:(x_i,y_j)\in\Omega} |e_{i,j}|^2)^{\frac{1}{2}}$ and $\max_{i,j:(x_i,y_j)\in\Omega} |e_{i,j}|$, respectively. Here, $e_{i,j}$ is the relative error at point (x_i, y_j). For the one-dimensional case the error is summed or maximised only over one variable. To be able to see the fifth-order of convergence from the WENO solver we set $\Delta t = (\Delta x)^{\frac{5}{3}}$ for the third-order Runge-Kutta time integration.

Example 4.1. The first test case is given by the following advection equation with $x \in \Omega = [1, 3]$, $t \in [0, 2]$, a solution dependent source term and a sine function as initial condition

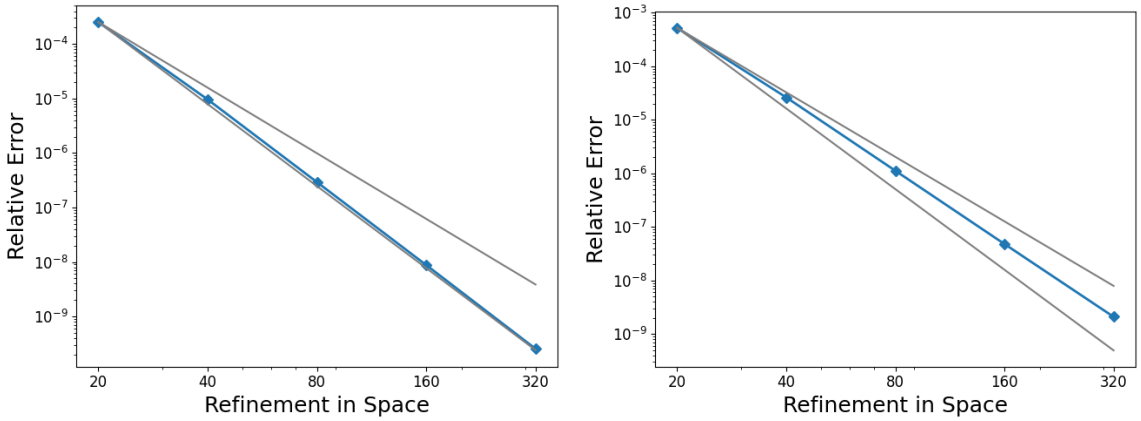
$$\begin{aligned} u_t + \nabla \cdot f(u) &= -c\, u \\ f(u) &= x\, u \\ u(x, 0) &= \Phi(x) = \sin(\pi\, x). \end{aligned} \tag{4.2.1}$$

The analytic solution is then given by

$$u(x, t) = \Phi(x\, e^{-t})\, e^{-(c+1)\, t}. \tag{4.2.2}$$

The errors are calculated at the final time $t = 2$. In the convergence plots in Figure 4.2 we can clearly see the desired fifth-order convergence of the space solver and small final errors. This shows that in the one-dimensional case the solver behaves as expected.

Example 4.2. The second test case is given by the following advection equation



(a) Convergence plot using the l_∞-norm (b) Convergence plot using the l_2-norm

Figure 4.2: Numerical convergence of WENO and RK solver for a one-dimensional problem with solution dependent source term and reference lines for convergence order 4 and 5 in grey [22].

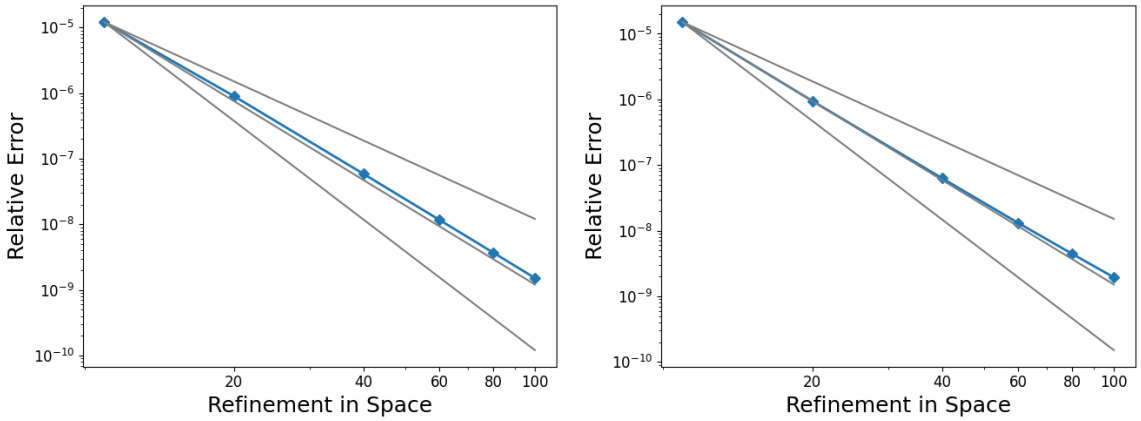
with $x \in \Omega = [1,3] \times [1,3]$, $t \in [0,2]$ and a solution dependent source term

$$\begin{aligned} u_t + \nabla \cdot f(u) &= -c\,u \\ f(u) &= \begin{pmatrix} x \\ 0 \end{pmatrix} u \\ u(x,y,0) &= \Phi(x,y) = e^{-((x-0.5)^2+(y-2)^2)}. \end{aligned} \tag{4.2.3}$$

The analytic solution is given by

$$u(x,y,t) = \Phi(x\,e^{-t}, y)\,e^{-(c+1)\,t}. \tag{4.2.4}$$

If we take a look at the numerical convergence plots in Figure 4.3 we see again small errors but only a fourth-order convergence. This loss in one order can be explained by the dimensional splitting which is needed to apply the WENO solver in two-dimensions and can reduce the order. This test has also been applied in y-direction to ensure that both directions work the same way. Furthermore, the space domain has been adapted to include negative values and ensure that there are no bugs regarding the change in sign. Both changes did not result in a different convergence order or higher errors thus we are sure that the solver works and the results can be trusted.



(a) Convergence plot using the l_∞-norm (b) Convergence plot using the l_2-norm

Figure 4.3: Numerical convergence of WENO and RK solver for a two-dimensional problem with flow in x-direction, solution dependent source term and reference lines for convergence order 3,4 and 5 in grey [22].

4.3 Verification of IMEX Solver

To test the third-order IMEX solver and determine its numerical convergence order, the following model problem was used

$$\begin{aligned} u_t(t) &= i\alpha u(t) - \beta u(t) \\ u(0) &= 1, \end{aligned} \tag{4.3.1}$$

with α and β real and positive constants and $i = \sqrt{-1}$. It is an adapted Dahlquist test equation, where the first part, containing the imaginary unit, mimics the advection and the second part mimics the diffusion. Therefore, the first part will be handled explicitly and the second implicitly. This test equation has been used to benchmark IMEX solvers in numerous publications [5] [53]. The exact solution of the equation is given by

$$u(t) = e^{(i\alpha-\beta)t}. \tag{4.3.2}$$

The resulting relative errors of the IMEX solver, given in the l_2 norm are shown in Figure 4.4. As expected we can clearly see the third-order convergence of the time solver.

Next, the solver was tested on a space-dependent problem with only an advection

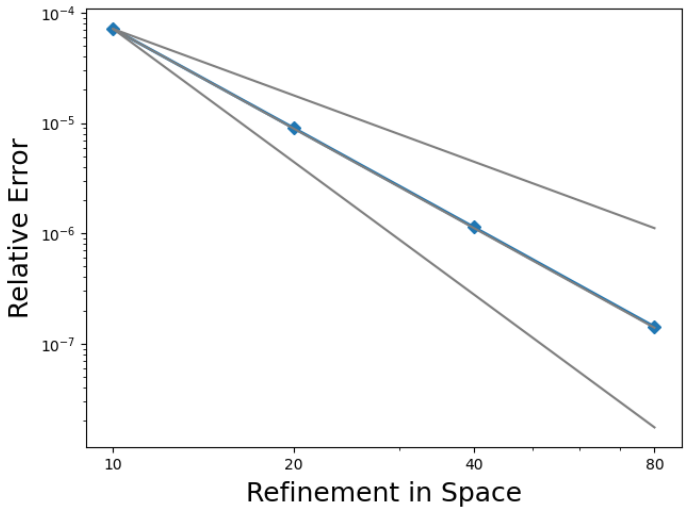


Figure 4.4: Numerical convergence of the IMEX solver for the Dahlquist test case with $\alpha = 2$ and $\beta = 1$, constant in space and with reference lines for convergence order 2, 3 and 4 in grey.

part. The test problem is given by

$$\begin{aligned} u_t &= v \cdot \nabla u \\ u(t=0) &= \sin(\sqrt{\lambda} x), \end{aligned} \tag{4.3.3}$$

with $u = u(x, y, t)$, $\lambda = 1$ and $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ on the time interval $[0, 1]$ and space interval $[-1, 1] \times [-1, 1]$. The exact solution is given by

$$u(x, y, t) = \sin(\sqrt{\lambda}(x - v_x t)). \tag{4.3.4}$$

The numerical convergence results of the advection problem show a fifth-order convergence (see Figure 4.5a) which is expected, because the space solver (WENO) is of order five and the problem is relatively simple. Because the problem is constant in y-direction the code has also been tested in y-direction (constant in x-direction), the numerical convergence remained unchanged.

Additionally, a test problem containing only the diffusive part has been chosen from Güttel and Pearson (2018) [31] and is given by

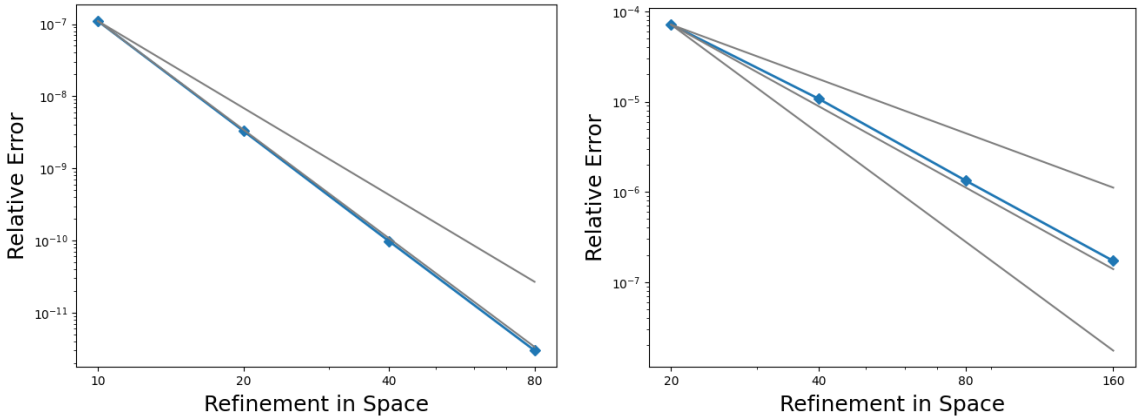
$$\begin{aligned} u_t &= D\, \Delta u \\ u(t=0) &= \frac{2}{\pi^2 \beta} e^T \cos(\frac{\pi x}{2}) \cos(\frac{\pi y}{2}), \end{aligned} \tag{4.3.5}$$

with $u = u(x, y, t)$, $\beta = 1$ and $D = 1$ on the time interval $[0, 1]$, with final time $T = 1$ and space domain $[-1, 1] \times [-1, 1]$.

The exact solution is given by

$$u(x, y, t) = \left[\frac{2}{\pi^2 \beta} e^T - \frac{4}{(4 + 2\pi^2)\beta} e^t \right] \cos\left(\frac{\pi x}{2}\right) \cos\left(\frac{\pi y}{2}\right). \tag{4.3.6}$$

The numerical convergence can be seen in Figure 4.5b . Here we can see the third-order convergence of the IMEX time solver.



(a) only advection - explicit - with reference lines for convergence order 4 and 5 in grey

(b) only diffusion - implicit - with reference lines for convergence order 2, 3 and 4 in grey

Figure 4.5: Numerical convergence of IMEX solver applied on a space-dependent problem where only the explicit or implicit part is activated.

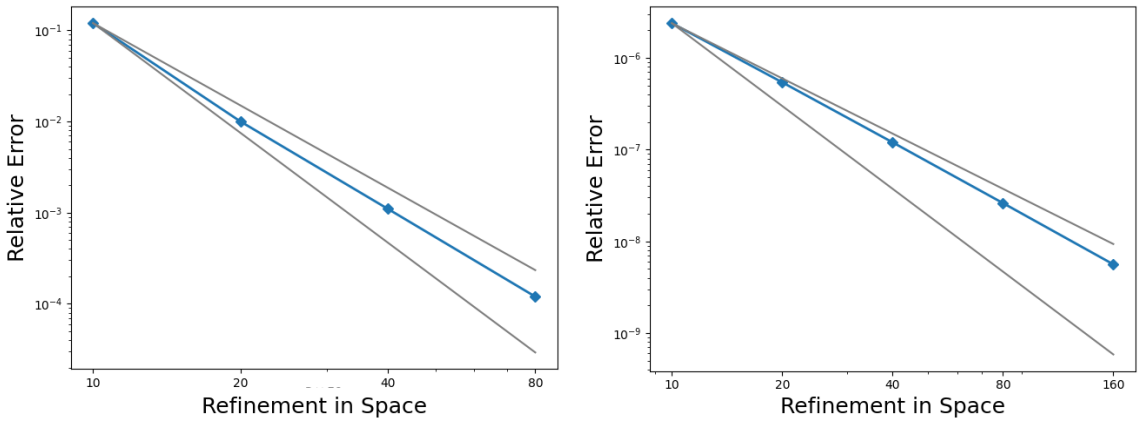
The final two test problems are combined advection-diffusion problems

$$u_t + v \cdot \nabla u - D\, \Delta u = f \tag{4.3.7}$$

with $u = u(x, y, t)$, a right-hand side source term f, $D = 1$ and $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.2 \end{pmatrix}$. Again on the time interval $[0, 1]$ and space domain $[-1, 1] \times [-1, 1]$. The first problem has the following starting value, source term and exact solution

$$\begin{aligned} u(t = 0) &= \sin(\sqrt{\lambda} x) \\ f &= 0 \\ u &= \sin(\sqrt{\lambda}(x - v_x t))\, e^{-\lambda t}, \end{aligned} \tag{4.3.8}$$

setting $\lambda = 1$. In Figure 4.6b you can see that the relative error is low, but the numerical convergence is only a bit over two. An explanation for this low order of convergence can be that the WENO space solver may lose in accuracy once the error becomes small.



(a) advection-diffusion with high oscillation and reference lines for convergence order 3 and 4 in grey

(b) advection-diffusion with low oscillation and reference lines for convergence order 2 and 3 in grey

Figure 4.6: Numerical convergence of the IMEX solver applied on a space-dependent advection-diffusion equation.

The second combined problem has the following starting value, source term and exact solution

$$\begin{aligned} u(t=0) &= \sin(\sqrt{\lambda}x) \\ f &= \sin(\sqrt{\lambda}x)\lambda \\ u &= \sin(\sqrt{\lambda}(x - v_x t)). \end{aligned} \tag{4.3.9}$$

We set $\lambda = 25$ quite high in order to get a numerically difficult problem. This is the case because a higher λ introduces oscillations of a higher frequency. For this problem we observe the desired third-order of convergence in the time solver, but because the problem is considerably more difficult, the relative errors increase significantly compared to the previous problem, see Figure 4.6a.

4.4 Taylor-Test

To verify that our implemented gradient is indeed a gradient we use the Taylor-Test [55]. Again, we denote the reduced cost functional with $j(c)$. It is only dependent on control c, where c is either defined by a velocity field V and diffusion D, for the advection-diffusion model ($c = (V, D)$), or by two velocity fields and the transformation parameter κ, for the two-compartment model ($c = (V^1, V^2, \kappa)$). Furthermore, we need some perturbation of these parameters which we denote by δc. For the case, that the parameters in c are constant we know that $j : \mathbb{R}^2/\mathbb{R}^3 \to \mathbb{R}$ and therefore $\nabla j : \mathbb{R}^2/\mathbb{R}^3 \to \mathcal{L}(\mathbb{R}^2/\mathbb{R}^3, \mathbb{R})$. The test makes use of the fact that

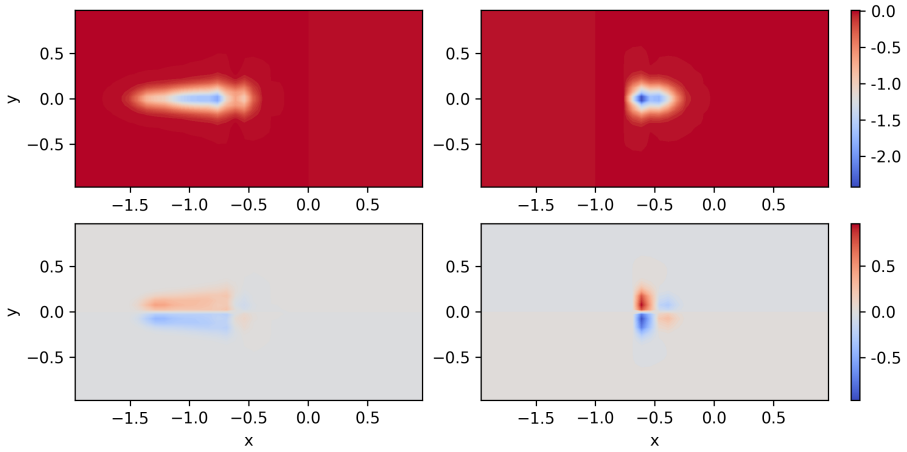
$$\begin{aligned} &|j(c + h\,\delta c) - j(c)| \to 0 \text{ at } \mathcal{O}(h), \text{ but} \\ &|j(c + h\,\delta c) - j(c) - h\,\nabla j(c)^T\,\delta c| \to 0 \text{ at } \mathcal{O}(h^2). \end{aligned} \tag{4.4.1}$$

This can be verified via Taylor's theorem. Here, we call $|j(c + h\,\delta c) - j(c)|$ the first-order Taylor remainder and the second-order Taylor remainder is given by $|j(c + h\,\delta c) - j(c) - h\,\nabla j(c)^T\,\delta c|$. Thus, for given j and ∇j we can check if ∇j is the gradient of j by repeatedly halving h, for some choice of h and δc, and checking if the result decreases by a factor of four. If we have a look at the non-constant system case, we can see that

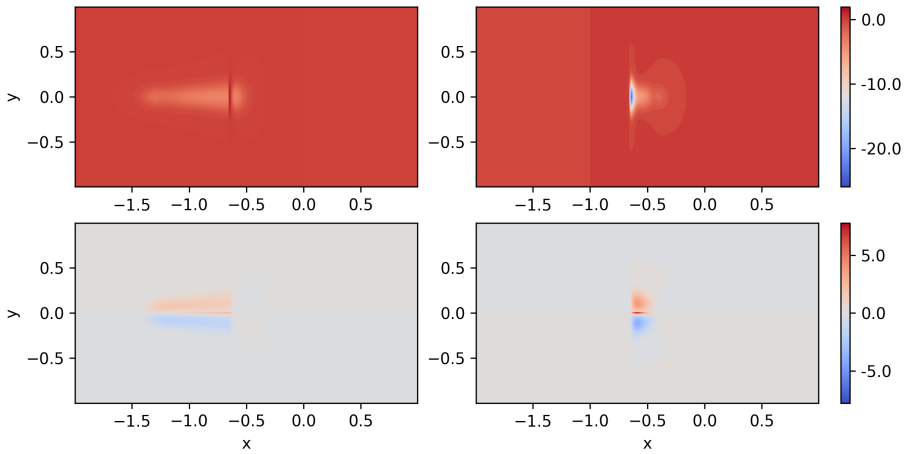
$$\begin{aligned} j &: L_2(\Omega) \to \mathbb{R} \\ \nabla j &: L_2(\Omega) \to \mathcal{L}(L_2(\Omega), \mathbb{R}). \end{aligned} \tag{4.4.2}$$

Therefore, in this case $\nabla j(c)^T\,\delta c$ denotes the L_2 product given by $\int_\Omega \nabla j(c)^T\,\delta c\,dx$. It is necessary to choose δc in a slightly smaller order of magnitude than c, so that the perturbations are not too large [55]. If the perturbation is too small the test can fail even if the gradient is correct. The same holds if h is chosen too large. The Taylor Test has been used to check the gradient for the advection-diffusion and the two-compartment model. Both gradients passed the Taylor test, for constant and space-dependent variables.

When applying the Taylor-Test we noted that for finer grids the optimisation detects parameter discontinuities that are masked by numerical diffusion on the coarser grids. This can be seen in Figure 4.7. Here it becomes apparent that the finer the spatial grid, the sharper the edge in the gradient. The sharp edge fits the



(a) space discretisation $n_x = n_y = 40$



(b) space discretisation $n_x = n_y = 320$

Figure 4.7: Gradients for velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) calculated on different grids for test problem using the two-compartment model with a discontinuous transformation parameter.

discontinuity of the transformation parameter

$$\kappa(x, y) = \begin{cases} 14, -0.65 < x < -0.35 \end{cases} \tag{4.4.3}$$

which was used in this test case.

Because of the sharp edge in the gradient we introduced a smooth transformation function, which is implemented via a sigmoid function instead of a step function

$$\kappa = \begin{cases} \frac{1}{1+e^{-20\,(x+0.65)}}, & \text{for } x < -0.5 \\ \frac{1}{1+e^{20\,(x+0.35)}}, & \text{else} \end{cases} . \tag{4.4.4}$$

Introducing smoothed parameters to simplify the optimisation is a common approach in computational mathematics [57]. We expect that with the smooth transformation parameter the difference between the coarse and fine grid is not as dominant and thus not relying on numerical diffusion for stable results. As shown in Figure 4.8 the results are noticeably smoother and do not change in behaviour when the gradient is computed on a coarser grid. Instead we can now see a rather sharp edge in the gradient of the y velocities. This can be explained by the change in sign in the velocity in y-direction. In the test case the velocities in y-direction were set to $v_y^1 = y$ and $v_y^2 = \frac{6}{10}\, y$.

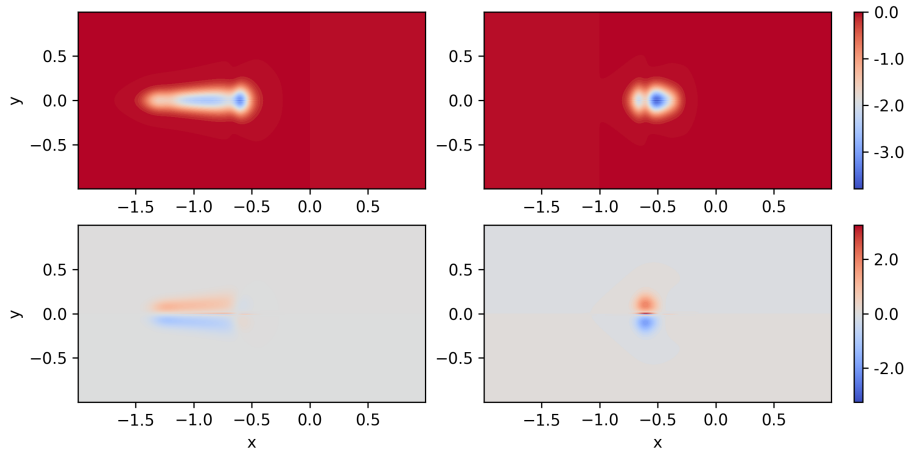


Figure 4.8: Gradients for velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) calculated with smoothened transformation parameter and $n_x = n_y = 320$.

For a problem which does not contain this change in sign for example if the y-velocities are set to $v_y^1 = v_y^2 = 0$, the behaviour vanishes, compare Figure 4.9. Thus, the smoother and less complex the parameters, the more stable the optimisation, because a finer gradient can be used without risking unstable solutions because of evolving discontinuities. If using smoother parameters is not possible, the programmer should use a coarser grid which introduces numerical diffusion and thus stabilises the optimisation.

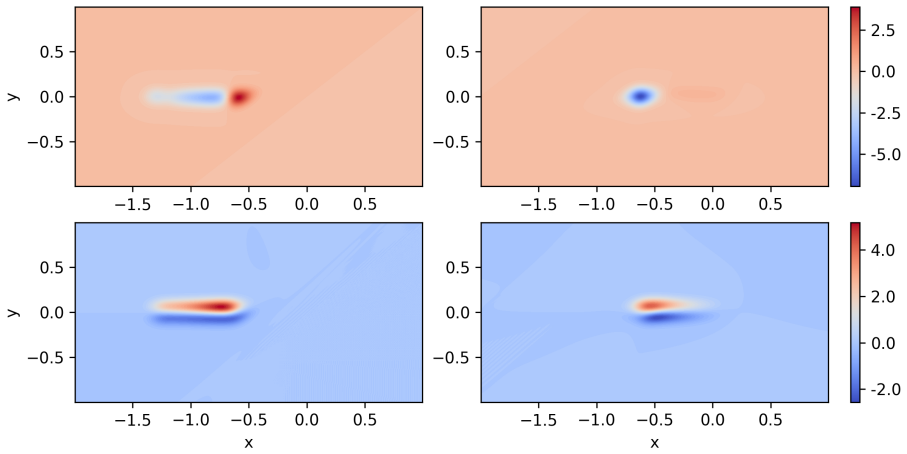


Figure 4.9: Gradients for velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) calculated with smoothened transformation parameter, no flow in y-direction and $n_x = n_y = 320$.

Chapter 5

Evaluation of the Parameter Identification Solver

5.1 General Evaluation of Reconstruction Results

As described in the preceding chapter, the constituent components of the parameter identification solver have undergone rigorous testing and evaluation. However, evaluating the entire parameter identification solver is not a straightforward process. This is due to the fact that the behaviour of the cost functional J is unknown. Thus, even though we can guarantee that we find a stationary point, given continuous differentiability [57]. We cannot know if the stationary point is indeed a minimum or even the global minimum. In general, a large number of minima is possible and because each evaluation of the cost functional is expensive we cannot deduce the behaviour of J via multiple evaluations. It is therefore only possible to establish whether the solver has converged, rather than assess the quality of the resulting solution. However, the ability to judge the quality of a solution is crucial, particularly in the context of problem reconstruction where the analytical solution is not known. In order to identify appropriate criteria for evaluating the computed results, a series of reconstructions have been carried out. These reconstructions have been carried out with only one round of adjusting the velocities and the diffusion or transformation parameter, to reduce the calculation time. The subsequent four general aspects have been identified.

1. A reasonable reduction in the gradient-norm
2. A visual inspection

3. A reasonable change has been observed in the cost functional

4. A reasonable change has been observed in all parameters

The first indicator is a mathematical parameter while the others are deduced through a combination of experience and expectation. To derive these indicators the inverse solvers have been applied multiple times to reconstruction problems, which were derived using artificial data. The artificial data has been created using the two-compartment model and the advection-diffusion model for ten different parameter values each. A table containing the names and parameters of these problems can be found in Table 5.1 and Table 5.2. The table is divided into two sections. The first containing random problems, with the objective of achieving authentic medical imitations and the second with problems especially modelled for a comparison of the two models. In order to be able to compare both models we tried to model similar behaviour with the two models. The idea is that for both models the problems 1 to 3 have a lower velocity in x-direction and 4 to 6 have a higher velocity in x-direction. Furthermore, in problems 1 and 4 we model only a slight bolus spread out, in problems 2 and 5 a slightly larger spread out and in problems 3 and 6 the bolus spreads out significantly. The resulting comparison of both models is discussed in Section 5.5. For now we focus on the general evaluation of both optimisation models.

To construct, biologically speaking, realistic artificial data via the two-compartment model, it is essential to ensure the absence of arterial blood at the exit of the organ, towards the end of the space domain, see Section 2.2. We achieve this by setting the velocity of the arterial blood flow, v_x^1, to zero in the middle of the organ. This guarantees that all arterial blood gets transformed into venous blood and is ensured via the v_x^1-constraint, which is applied to all v_x^1 velocities. The constraint is also enforced, when doing the reconstruction. Furthermore, we introduce a κ-constraint which ensures that the transformation from arterial into venous blood exclusively occurs within the organ, i.e. near the middle of the domain. This constraint is also enforced in the reconstruction. The last constraint, v_y^2, is added to achieve a more realistic result in some of the problems, which for these specific velocities enforces that the slimming of the bolus only starts towards the end of the organ. This constraint is not added to all simulations, see Table 5.1, and is not enforced in the reconstruction. A visualisation of most of the constraint functions from Tables 5.1 and 5.2 can be found in Figure 5.1. For the missing v_y^D-constraint function the reader only has

to change the switching point of the v_y^2-constraint function from $x = -0.5$ to $x = -1$.

Two-Compartment Model						
name	velocity v^1		velocity v^2		κ	appl. v_y^2 -constr.
	v_x^1·constr.	v_y^1	v_x^2	v_y^2		
simple	1.5	0.1	2	0.2	14	
simple2	1.5	0	1.8	0	14	
v_y	1.5	0.3	2	0.4	14	
realistic	$-2x$	$2y$	$x+2$	$-y$	14	
1 (low)	$x+2.1$	y	$x+2.1$	$-y$	11	yes
2 (middle)	$x+2.1$	$2y$	$x+2.1$	$-1.5y$	11	yes
3 (high)	$x+2.1$	$2.5y$	$x+2.1$	$-2y$	11	yes
4 (lowFast)	$0.5x+2.1$	y	$1.5x+3$	$-y$	14	yes
5 (middleFast)	$0.5x+2.1$	$2y$	$1.5x+3$	$-1.5y$	14	yes
6 (highFast)	$0.5x+2.1$	$2.5y$	$1.5x+3$	$-2y$	14	yes
v_x^1-constraint	$\begin{cases} 1, \text{ for } x < -0.5 \\ 0, \text{ else} \end{cases}$					
v_y^2-constraint	$\begin{cases} 0, \text{ for } x < -0.5 \\ 1, \text{ else} \end{cases}$					
κ-constraint	$\begin{cases} \frac{1}{1+e^{-20\,(x+1.1)}}, \text{ for } x < -0.8 \\ \frac{1}{1+e^{20\,(x+0.5)}}, \text{ else} \end{cases}$					
initial condition	$u(x,y,0) = 3\,e^{-\frac{(x+1.3)^2+y^2}{0.02}}$			$w(x,y,0) = 0$		

Table 5.1: Parameters used to derive ten different problems of artificial data using the two-compartment model.

In order to make both models comparable the diffusion parameter is also constrained to the area of the organ. This constraint was used in all simulations and is also enforced in the reconstruction. Similar to the two-compartment data, for a more realistic result, a constraint for the velocity in y-direction was used in one diffusion simulation, where the velocities made the application necessary.

In the following we refer to the problems by name or number and add a "_T" if the problem was solved using the parameter identification with the two-compartment

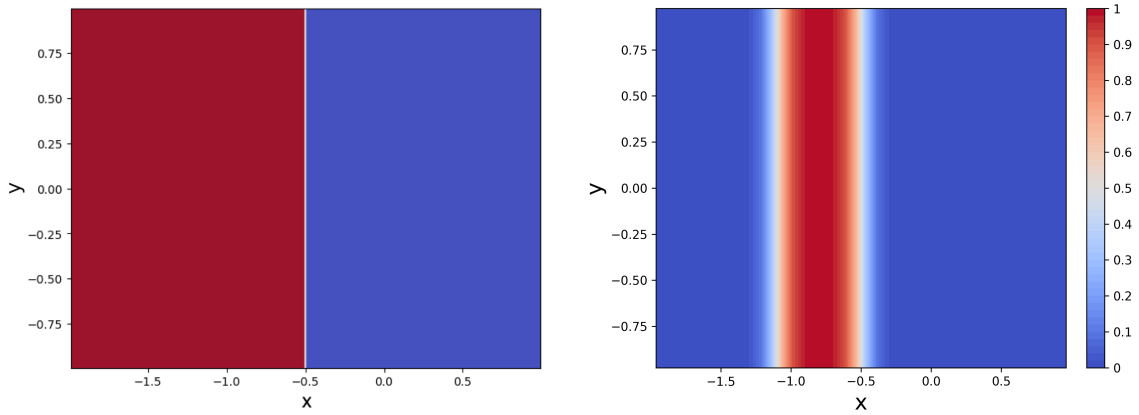
model and a "_D" if the problem was solved using the parameter identification with the advection-diffusion model.

Advection-Diffusion Model				
name	velocity v		diffusion	appl. v_y
	v_x^1	v_y^1	D	-constr.
simple	1.2	0	0.02	
v_y	1.2	0.5	0.02	
realistic	1.5	$-4y$	0.03	yes
complex	$-x+0.5$	$-5y(x+1)$	0.03	
7 (low)	$x+2.1$	0	0.01	
8 (middle)	$x+2.1$	0	0.03	
9 (high)	$x+2.1$	0	0.05	
10 (lowFast)	$1.5x+3$	0	0.01	
11 (middleFast)	$1.5x+3$	0	0.03	
12 (highFast)	$1.5x+3$	0	0.05	
v_y-constraint	$\begin{cases} 1, \text{ for } x < -1 \\ 0, \text{ else} \end{cases}$			
D-constraint	$\begin{cases} \frac{1}{1+e^{-20\,(x+1.1)}}, \text{ for } x < -0.8 \\ \frac{1}{1+e^{20\,(x+0.5)}}, \text{ else} \end{cases}$			
initial condition	$u(x,y,0) = 3\,e^{-\frac{(x+1.3)^2+y^2}{0.02}}$			

Table 5.2: Parameters used to derive ten different problems of artificial data using the advection-diffusion model.

Reduction in the Gradient-Norm

The gradient norm is a mathematically reasonable indicator, because we search for parameters which minimise the cost functional J. Thus, as a necessary condition the gradient at this point should be zero, $\nabla J = 0$ [57]. This strong condition will not be satisfied in the actual computation because the gradient will never be exactly zero. Nevertheless, if the gradient norm is reduced we get closer to a potential extrema, with respect to the previous step. The verification that it is indeed a minimum and not a maximum can be done via the second derivative or in our case via the chosen line-search and gradient descent method. That the gradient actually is smaller for



(a) Constraint used on v_x^1 - switching the colours results in the constraint used on v_y^2.

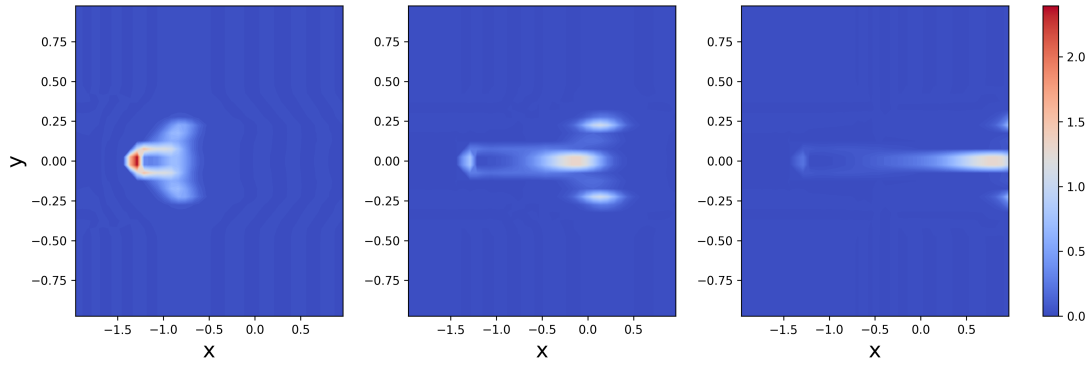
(b) Constraint for the transformation and diffusion parameter.

Figure 5.1: Visualisation of different constraint functions applied to the PDE parameters, here *red* and *blue* represent 1 and 0 respectively and the white represents values in between.

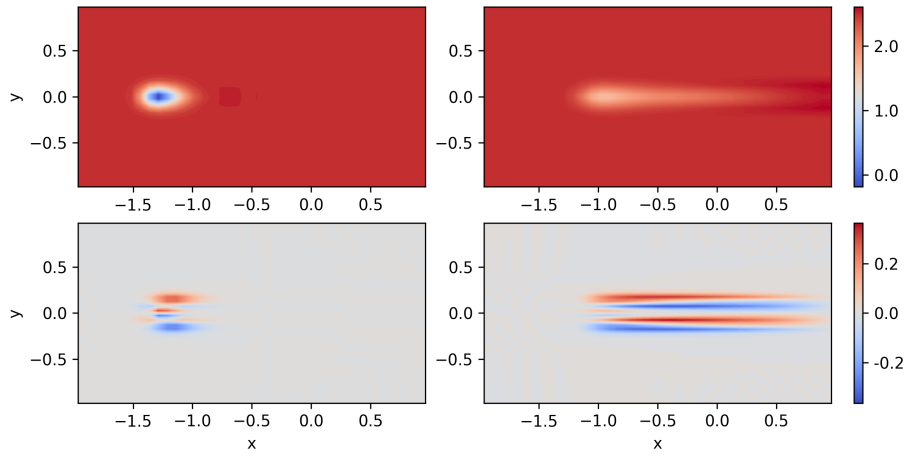
good results and larger for failed optimisations can be verified in Table 5.3. The gradient norm of the velocities is derived using the 2-norm on the discrete arrays of the corresponding gradient. For the transformation or diffusion parameter the gradient norm is derived by taking the absolute value for a scalar or the 2-norm for a space-dependent gradient. Comparing various results we derived, that the gradient in the velocity norm should be $||\nabla J_v||_2^{\text{T-opt}} \sim 0.005$ or less for reconstructions using the two-compartment model and $||\nabla J_v||_2^{\text{D-opt}} \sim 0.001$ or less for reconstructions using the advection-diffusion model. The values for the advection-diffusion model are lower because we only have one velocity field instead of two. Furthermore, the final gradient norm for the transformation parameter should be around $||\nabla J_\kappa||_2^{\text{opt}} \sim 0.001$ or less and the gradient norm of the diffusion parameter around $||\nabla J_D|_2^{\text{opt}} \sim 0.01$. Regarding $||\nabla J_D||_2^{\text{opt}}$ we can see that the values are lower for problems where we reconstruct artificial data derived with the advection-diffusion model compared to the reconstructions of the artificial data derived via the two-compartment model, see Table 5.3. This can be explained by the fact that the advection-diffusion model fits its own data better than data derived using a different model. These threshold estimates for the gradient norm only apply to comparable problems, if the values of the parameters or concentrations differ significantly they should be recalibrated.

Visual Inspection

The second indicator to evaluate the calculated solution is the visual appearance of the concentrations. Here we are interested in two things. First of all: is the bolus form preserved? This is an important property of the artificial data, which we reconstruct and thus should be preserved in the reconstructed concentrations. An example where the optimisation failed to retain the bolus form can be found in Figure 5.2a.



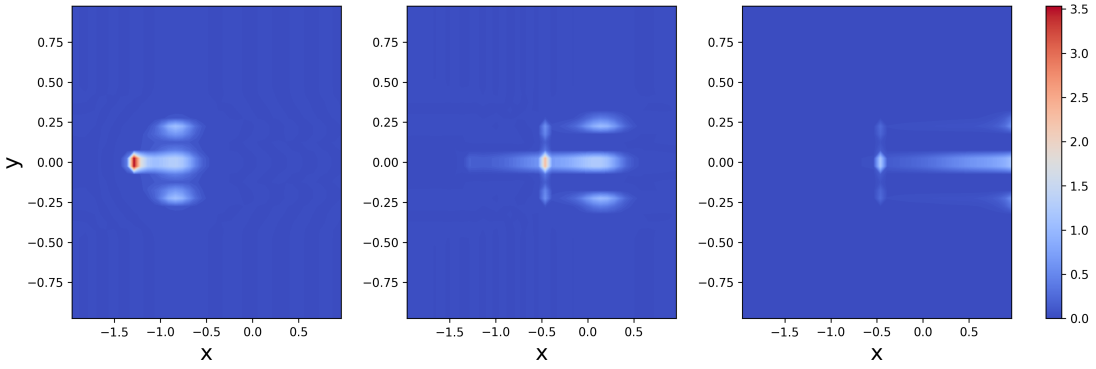
(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.



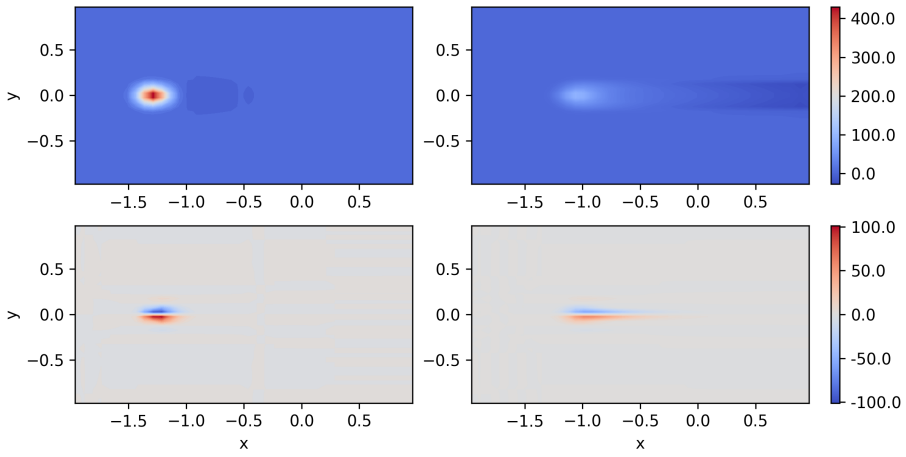
(b) Reconstructed velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.

Figure 5.2: Optimisation failed to retain bolus form instead a splitting into multiple parts can be observed (run 6.1_T).

And secondly: can we see points of high concentrations? These points of high concentrations indicate unstable solutions and are therefore also an indicator for a failed optimisation. An example for this case can be found in Figure 5.3a.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.



(b) Gradient for velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.

Figure 5.3: Optimisation failed visually - High concentration areas appear and we can see high values in the gradient for the velocities, resulting from instabilities in the forward and backward problem (run 10.2_T).

Change in Cost Functional and Parameters

The third indicator is the change in the cost functional. Based on experimental results, we want to see at least a reduction of 50% to ensure a solution that is a reasonable fit to the data. If we get a reduction by 80% or more we expect a result that is a good fit to the measured data. This correlates with another smaller parameter, where we want to see a minimum amount of steps. If only three to five steps are taken, then the decrease in the cost functional is small. The stopping point could then be a stationary point, likely a saddle point or local minimum or we could have started the optimisation in a flat area of the cost functional, see Figure 5.17, resulting in the algorithm not finding a sufficient descent direction. The results of a few selected optimisation runs can be found in Table 5.3. We can see, that the final value of the cost functional and the visual evaluation is worse if we do not see a reasonable decrease in cost functional J.

Run	$\|\|\nabla \mathbf{J}_{\mathbf{v}}\|\|_2$	$\|\|\nabla \mathbf{J}_{\kappa}\|\|_2 / \|\|\nabla \mathbf{J}_{\mathbf{D}}\|\|_2$	**Decr.**of **J** in %	$\mathbf{J}_{\text{end}}$	**Steps** $\mathbf{v}$	**Steps** κ/D	**Start** $\mathbf{v^1}$ $\mathbf{v^2}$	**Start** κ/D	**Solver**	**Visual**
10.2_T	407.86	0.07885	34.66	0.06367	3	1	$\begin{pmatrix}2.5\\0\end{pmatrix}$ $\begin{pmatrix}2.5\\0\end{pmatrix}$	30	CG	Bad
10.3_T	0.00299	0.00074	84.07	0.00918	28	2	$\begin{pmatrix}2.0\\0\end{pmatrix}$ $\begin{pmatrix}2.0\\0\end{pmatrix}$	25	CG	Good
6.1_T	73.267	0.02827	34.99	0.05412	4	0	$\begin{pmatrix}2.5\\0\end{pmatrix}$ $\begin{pmatrix}2.5\\0\end{pmatrix}$	20	CG	Bad
6.4_T	0.00264	0.00136	88.23	0.00993	38	1	$\begin{pmatrix}2.5\\0\end{pmatrix}$ $\begin{pmatrix}2.5\\0\end{pmatrix}$	25	CG	Good
4.1_D	0.03506	0.39504	37.91	0.01753	4	0	$v_x = 2.0$ $v_y = 0$	0.07	BFGS	Okay
4.5_D	0.00227	0.15591	88.99	0.00692	43	1	$v_x = 1.5$ $v_y = 0$	0.07	CG	Good
12.1_D	0.00015	0.0002	82.51	0.00288	12	2	$v_x = 2.0$ $v_y = 0$	0.07	BFGS	Good

Table 5.3: Results of exemplary optimisation runs showing indicators to evaluate the success of the resulting reconstruction, the gradient norm is given by the 2-norm applied to the discrete calculated gradient.

Which leads us to the last indicator, change in all parameters, not only the velocity. We use this parameter, because if we see no change in the transformation or diffusion parameter, then we either made a lucky starting guess, which is unlikely.

Or we reached a local minimum that is not a good representation of the true solution, because we did not adapt the second perfusion parameter. This relation can also be seen in Table 5.3.

In general, if the optimisation failed regarding one of these indicators it is useful to restart the solver using changed starting parameters. How sensitive the inverse solver is to the starting parameters can be seen in Table 5.3. If we change the v_x velocity by 0.5 we converge to a completely different solution. The same holds true for the others parameters, this can for example be seen for the transformation parameter κ in Table 5.3 for the runs "6.1_T" and "6.4_T". If changing the starting parameters does not yield the desired quality of reconstruction we can try to change the descent solver. We implemented the conjugate gradient, BFGS and steepest descent method. In general, the conjugate gradient method works better for the two-compartment model and the BFGS method works better for the advection-diffusion model, therefore it is logical to start with those by default and change if the resulting reconstructions are dissatisfactory. If both are unsuccessful or result in only a few optimisation steps the steepest descent method should be applied to derive a sufficient starting condition for the more sophisticated descent directions.

5.2 Evaluation of the Advection-Diffusion Model

In the following section we have a closer look at the reconstruction using the advection-diffusion model. A positive aspect of the advection-diffusion model is that it is widely known and therefore there are many solvers available and the theory is well-known.

5.2.1 Comparing Reconstruction Results

What we expected is that the advection part of the equation models the transport velocity and the diffusion parameter models the diffusive behaviour. But when calculating different reconstructions it became obvious, that the model already corrects some of the mismatched diffusive behaviour via the velocities, before updating the diffusion parameter, as seen in Figure 5.4c, Figure 5.5c and Figure 5.2b. Even though this is unexpected it is a reasonable behaviour. This behaviour can be seen especially well, if the artificial data has been calculated with velocity in x-direction

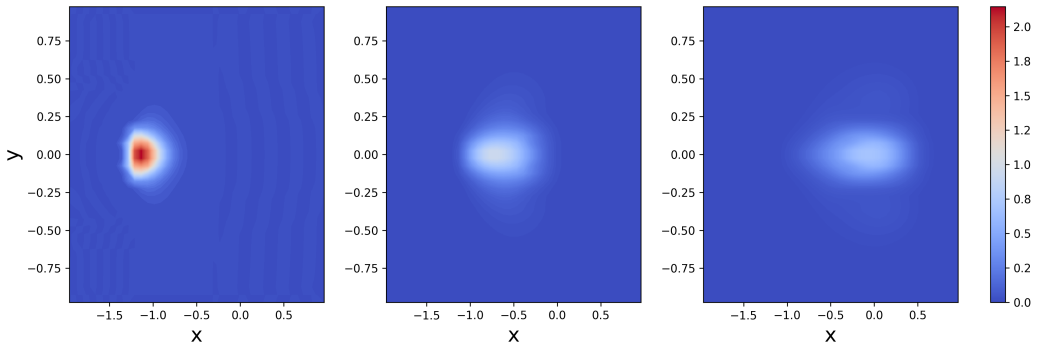
and zero velocity in y-direction, $V = (v_x \quad 0)^T$.

As mentioned we expected the reconstruction to first adjust the velocity in x-direction, without doing any corrections in the y-direction because we started with the true velocity, before correcting the diffusion parameter. Instead the reconstruction corrected the mismatch in the diffusion parameter by adjusting the velocity in y-direction, which results in "inflowing" velocities, see Figure 5.4c, towards the centre of the bolus, if D_{start} has a higher value than the desired diffusion parameter or "outflowing" velocities, see Figure 5.7b, if D_{start} has a lower value than the used diffusion parameter. This could not be remedied by adjusting the velocities and diffusion parameter simultaneously as seen in Figures 5.4 and 5.5, where the similarity between the reconstructed velocities and resulting concentrations is clearly visible.

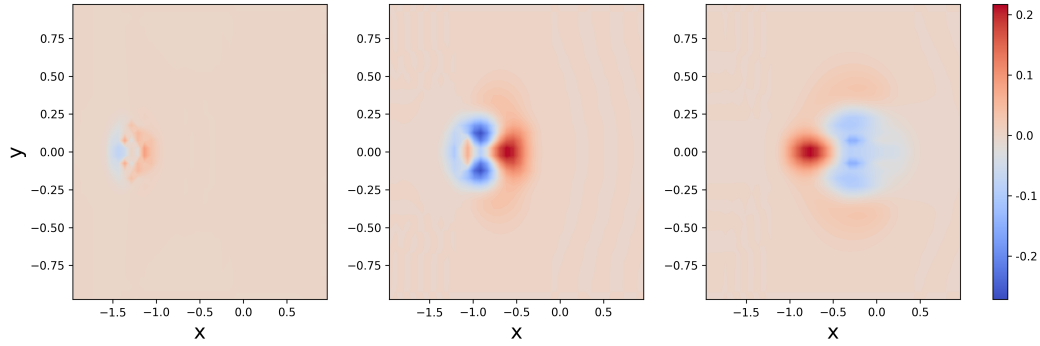
In either cases we can see an "inflow", meaning a flow towards the centre of the bolus, in the y-velocity. This results in a higher reconstructed diffusion parameter than expected or wanted. In this example the artificial data has been calculated with $D = 0.01$ and the reconstructed values are $D = 0.061$ for run 7.1_D and $D = 0.057$ for run 7.5_D. These values are nowhere near the true solution. Nevertheless, the reconstructed concentrations are satisfactory and the cost functional is low as well. This suggests that the reconstruction is not uniquely solvable.

More examples that reinforce this assumption are run 6.2_D and 6.3_D. Here, two completely different sets of parameters were found for the same artificial data. In both cases all indicators judging the quality of the result of the reconstruction were sufficiently fulfilled and the cost functional values were both low. This multiplicity of good solutions is problematic, because we have to decide which one is a better approximation and the indicators given are ambivalent. For example is the decrease in the cost functional and the final value of the cost functional in run 6.3_D slightly better, but visually run 6.2_D looks more realistic, see Figures 5.6 and 5.7.

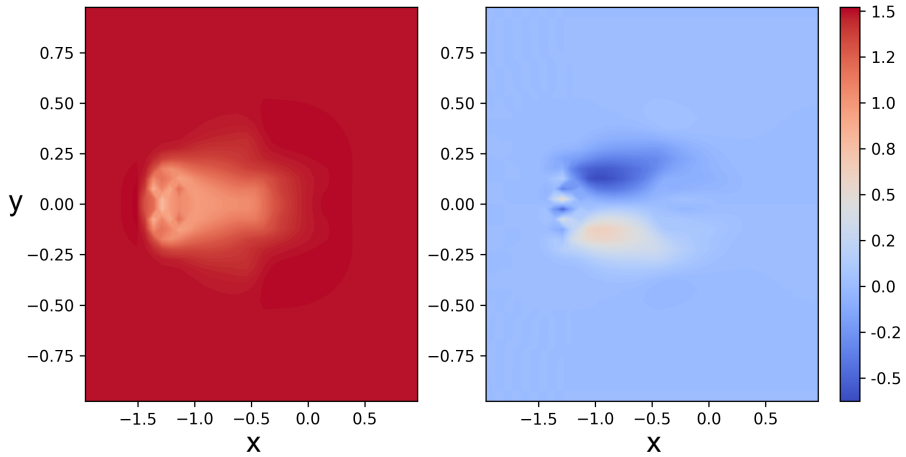
This ability, to correct errors in one parameter using another, makes the usage of the inverse solver complicated, because even if we find a reasonable result we cannot say how close we are to the true values.



(a) Reconstructed concentrations 7.1_D at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

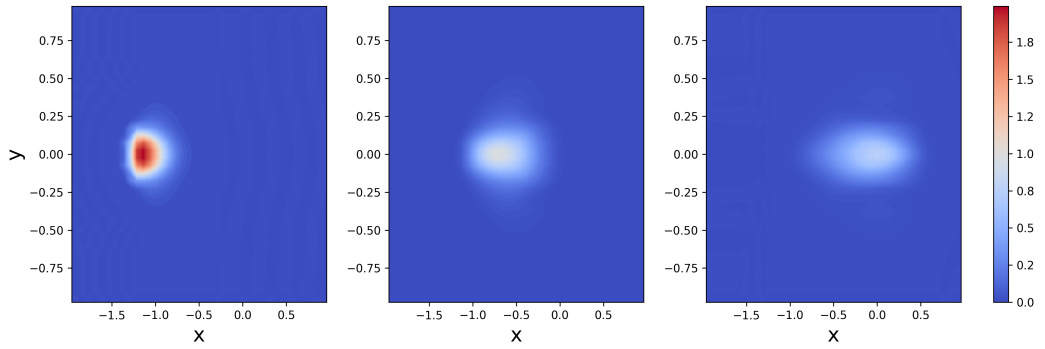


(b) Mismatch of reconstructed concentrations 7.1_D and artificial data at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

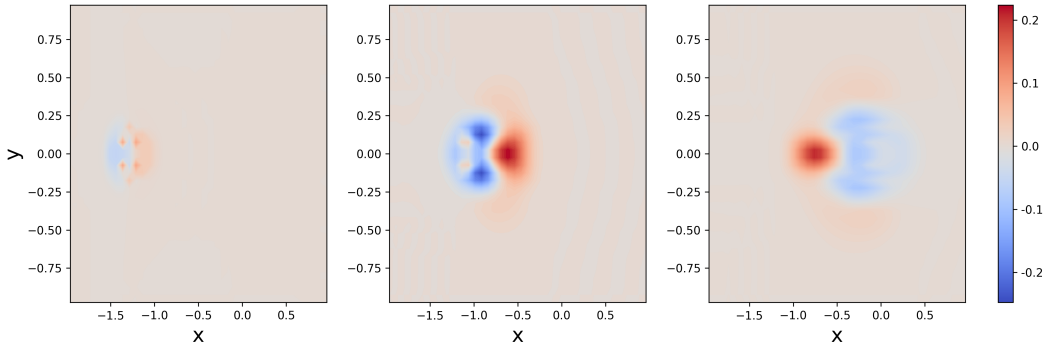


(c) Reconstructed velocities v_x and v_y from *left* to *right* - Run 7.1_D.

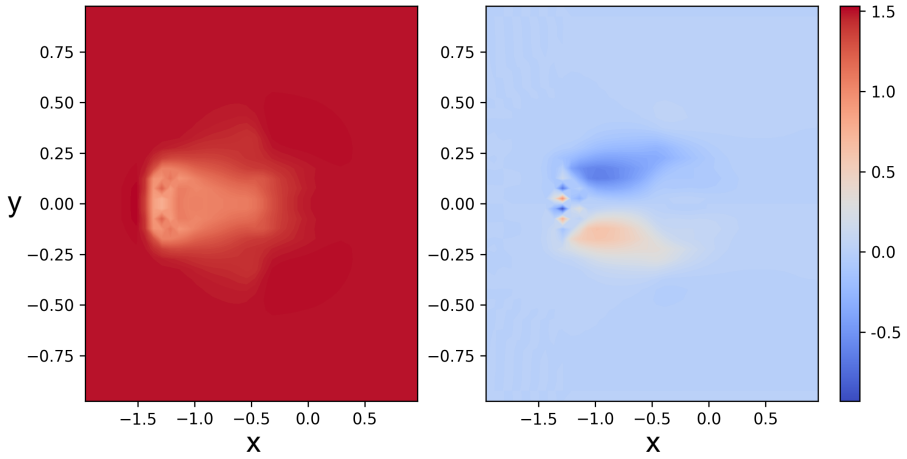
Figure 5.4: Comparison of reconstructing velocity and diffusion successively (5.4a, 5.4b and 5.4c) and simultaneously (5.5) for problem 7.



(a) Reconstructed concentrations 7.5_D at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

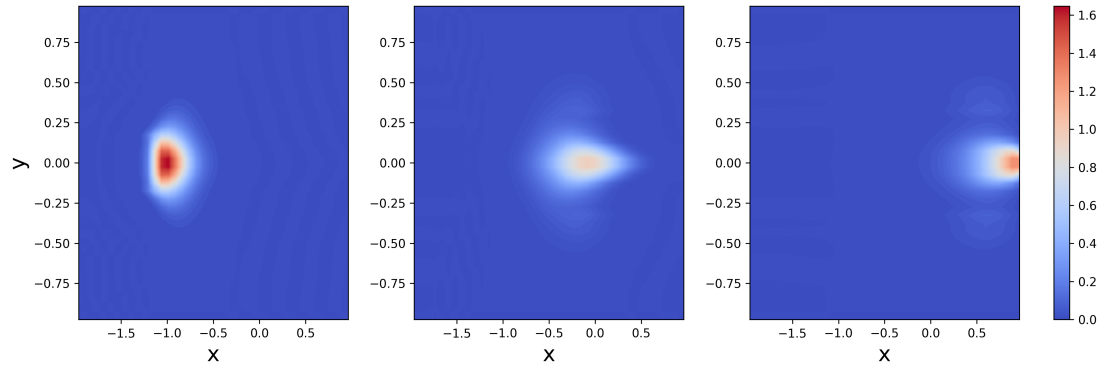


(b) Mismatch of reconstructed concentrations 7.5_D and artificial data at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

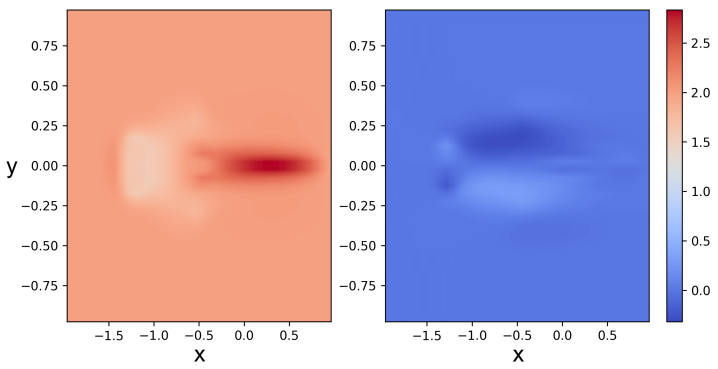


(c) Reconstructed velocities v_x and v_y from *left* to *right* - Run 7.5_D.

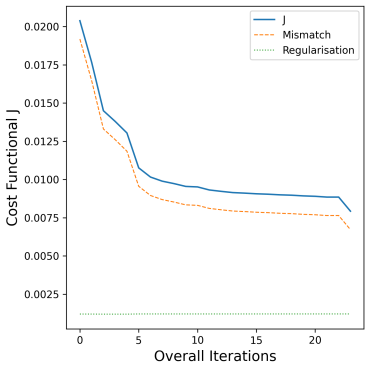
Figure 5.5: Comparison of reconstructing velocity and diffusion successively (5.4) and simultaneously (5.5a, 5.5b and 5.5c) for problem 7.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

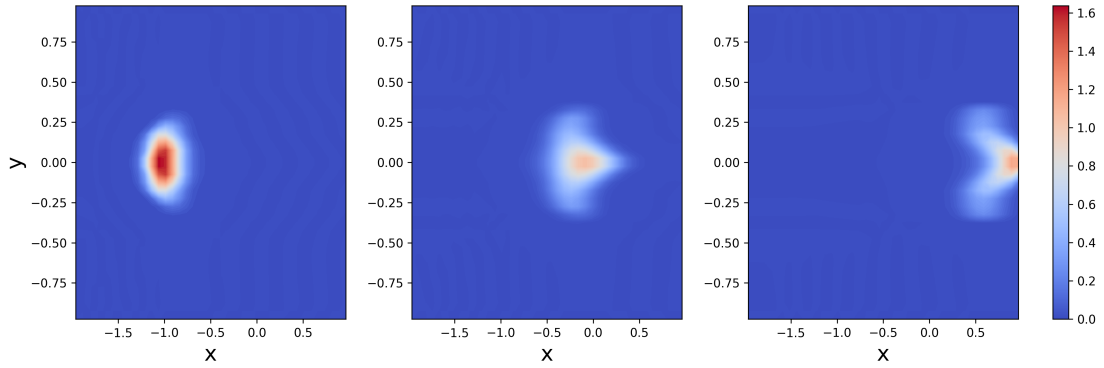


(b) Reconstructed velocities v_x and v_y from *left* to *right*.

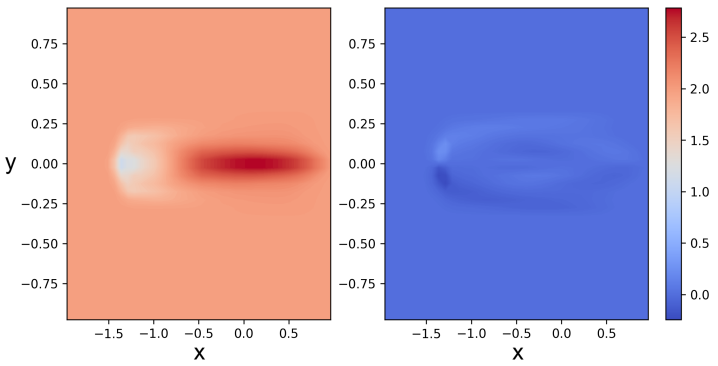


(c) Cost functional

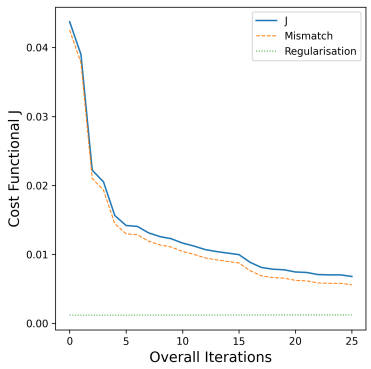
Figure 5.6: Reconstruction for problem 6 run 6.2_D - Resulting in high diffusion $D = 0.052$.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.



(b) Reconstructed velocities v_x and v_y from *left* to *right*.



(c) Cost functional

Figure 5.7: Reconstruction for problem 6 run 6.3_D - Resulting in low diffusion $D = 0.004$.

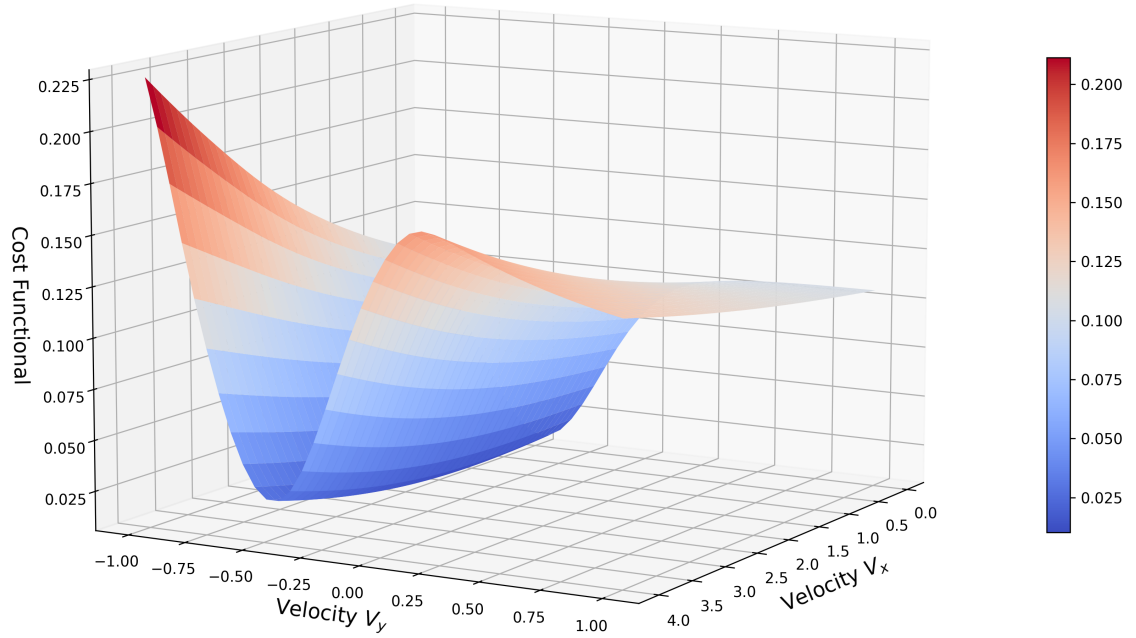
5.2.2 Loss Landscape

This is further visualised in a loss landscape in Figure 5.8, which illustrates the cost functional values for two varying scalar parameters. Plotting the true cost functional is not possible because it depends on three space-dependent parameters and thus depending on the discretisation, multiple hundreds or even thousands of variables. The cost functional was calculated for problem 7, where we only varied two scalar parameters over 40 equidistant points each and the others were set to the true values.

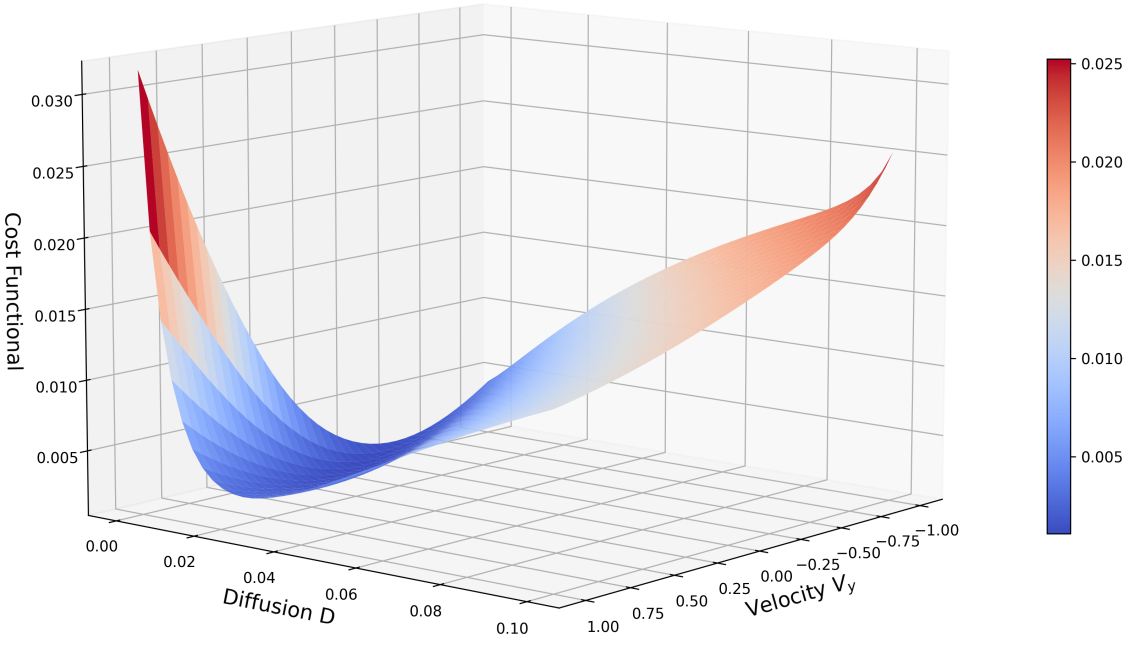
For the Figure 5.8a, we set the velocity in x-direction to be constant and chosen from $v_x \in [0, 4]$. The velocity in y-direction is set to $v_y = a\,y$ with $a \in [-1, 1]$. This gives us the possibility to still have a space-dependent v_y and thus a complexity that compares better to the actual reconstruction problem. For Figure 5.8b, the first parameter is again the velocity in y-direction, $v_y = a\,y$ with $a \in [-1, 1]$, and the second parameter is the diffusion D chosen from the interval $[0, 0.1]$.

In both figures we discover narrow valleys with small slopes, which are hard to navigate for minimisation problems. This is the case because the slope near the minimum is so flat that numerical errors make finding a good descent direction impossible. Most likely the algorithm terminates early inside the valley, because the optimisation steps are small and thus do not exceed the chosen stopping criteria. In Figure 5.8a it looks like the optimal value of v_y is clear, while multiple values for v_x result in similarly low cost functional values. In Figure 5.8b we can see an area of low influence in the v_y values and the diffusion D, which shows that the optimal value for one parameter is depending on the choice of the other parameters. Furthermore, we can see that the cost functional values in both problems vary significantly, which make them hard to compare.

To summarise, we can say that the plot is indeed only a simplified sketch which cannot capture the whole complex behaviour of the cost functional. Nevertheless, the problem gets only more complicated if we include more degrees of freedom, this may add even more problems to the loss landscape, for example sharp spikes.



(a) Varying velocities v_x and a for $v_y = a\,y$



(b) Varying a for velocity $v_y = a\,y$ and diffusion D

Figure 5.8: Cost functional for problem 7, with two varying scalar parameters. The other parameters are fixed and set to be the true value.

5.2.3 Relation Between the Velocity and Diffusion Parameter

Finally we want to mention a few relations between the modelling parameters that became apparent when analysing the model. In order to be able to compare the parameters we need to convert the space-dependent velocity to a scalar value. A simple way to derive an average value of the velocity is the L_1-norm, which is defined by $||V||_{L_1} = \sum_{i,j\in\Omega} |V[i,j]|$, for a space domain Ω and a discrete velocity field V. In the following we further make use of an integral-norm, which calculates the average value of the velocity by integrating over the relevant area M and then divide by the surface area

$$V_{\text{scalar}} = \frac{\int_M |V|\, dx}{\int_M 1\, dx}. \tag{5.2.1}$$

In the following, we will denote this derivation of scalar values with a subscript "I" for integral-norm. The relevant area is the area where the velocity was adjusted by the reconstruction algorithm. This does not include the whole domain, because the reconstruction algorithm can only adjust the velocities where there is a signal, which is in our application the tracer concentration. To define this areas of change we use two different methods. The first defines the mask by every point where the velocity changed sufficiently and is denoted by "M"

$$M = \begin{cases} 1, \text{ if } |V_{\text{start}}[i,j] - V_{\text{end}}[i,j]| > 0.05 \\ 0, \text{ else} \end{cases}, \tag{5.2.2}$$

for discrete velocity fields at point (x_i, y_j). The second defines a quadratic mask also including the velocity areas with a sufficient change and is denoted by "MQ"

$$MQ = \text{Rectangle with corners at } \begin{cases} \min \text{index of "M" in } x \text{ and } y \\ \max \text{index of "M" in } x \text{ and } y \end{cases}. \tag{5.2.3}$$

A visualisation of both masks for an exemplary velocity field can be found in Figure 5.9.

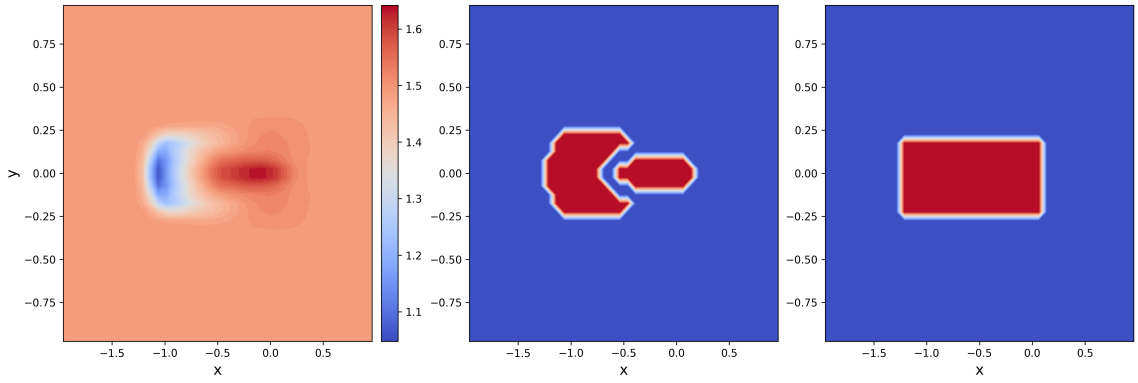


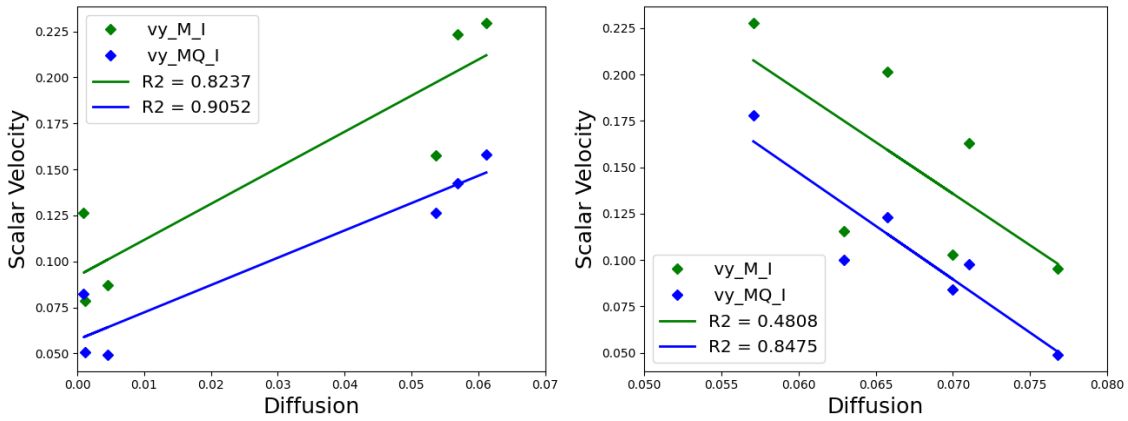
Figure 5.9: Visualisation of masks "M" (*centre*) and "MQ" (*right*) applied to a velocity field (*left*).

Furthermore, we need to be able to define the strength of a relation. This can be done via the coefficient of determination also known as R^2. It is often used in statistics and defined the following way

$$\begin{aligned} R^2 &= 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \\ SS_{\text{res}} &= \sum_{i=1}^{n}(y_i - u_i)^2, \quad SS_{\text{tot}} = \sum_{i=1}^{n}(y_i - \overline{y})^2 \\ \overline{y} &= \frac{1}{n}\sum_{i=1}^{n} y_i, \end{aligned} \tag{5.2.4}$$

with y_i the i'th data point out of a dataset with n data points, $\overline{y}$ the average of all y_i values and u_i the calculated i'th dependent variable based on the fitting equation, in our case a linear function [19].

When reconstructing data which was derived using the two-compartment model, we can see a linear relation between the model parameters v_y and D, see Figure 5.10. This is reasonable because the velocity in y-direction can correct a mismatch in the diffusion parameter by "pushing" the concentrations inwards or outwards, as seen for example in Figure 5.5c. Nevertheless, the strength of the correlation is misleading, because the reconstructions started with different diffusion parameters. The lower three points, in Figure 5.10a, are derived with $D_{\text{start}} = 0$ and the upper three with $D_{\text{start}} = 0.07$. As to expect, the reconstruction found parameters that are close to the starting values thus the correlation could be based on the choice of starting values, rather than a linear relation between the v_y velocity and the diffusion.



(a) Problem $(1-6)$: derived via the two-compartment model.

(b) Problem $(7-12)$: derived via the advection-diffusion model.

Figure 5.10: Relation between scalar velocity v_y and D for reconstructions of problem 1 to 12, including a linear fitting function with R^2 value.

When we reconstruct artificial data derived by the advection-diffusion model, the relation looks weaker but because all results are gained with the same starting values the values are more comparable. What Figure 5.10b shows is that we started in all cases with a diffusion parameter that was set higher than the true parameter. To correct the mismatch in the diffusion parameter the reconstruction uses higher activity in the y-velocity, and thus higher values for the scalar value of v_y, to reconstruct a lower valued diffusion parameter. This explains the decreasing linear fit. But as explained this relation is highly problem dependent and therefore not visible in all reconstructions. There are no indications for further relations between the parameters of the advection-diffusion model thus we expect the reconstructed parameters to be otherwise independent from each other.

5.3 Evaluation of the Two-Compartment Model

5.3.1 Comparing Reconstruction Results

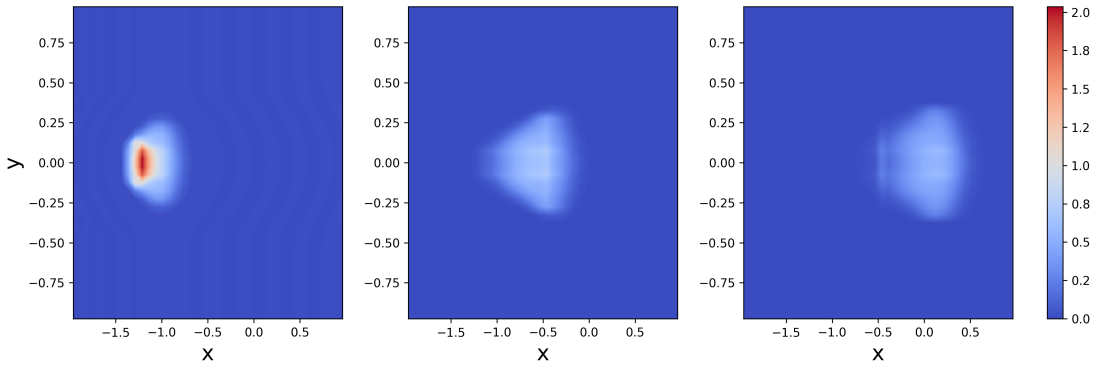
After analysing the results derived by the advection-diffusion model we now do the same for the two-compartment model. This model was chosen because it describes the medical reality more accurately and without needing a pseudo-diffusion term.

A positive aspect of managing without a diffusion parameter is that we do not need to use an implicit time solver, which simplifies the numerical handling of the equations. Here simplified means, that we avoid solving linear systems in each time step, which not only makes the algorithm easier to understand and implement but also results in cheaper computational steps. Using the two-compartment model provides us with many free variables, because we have two adjustable velocity fields and the transformation parameter κ. This enables us to reconstruct a variety of different shapes, one of them being the ability to model diffusive behaviour while only using velocities, see Figure 5.11.

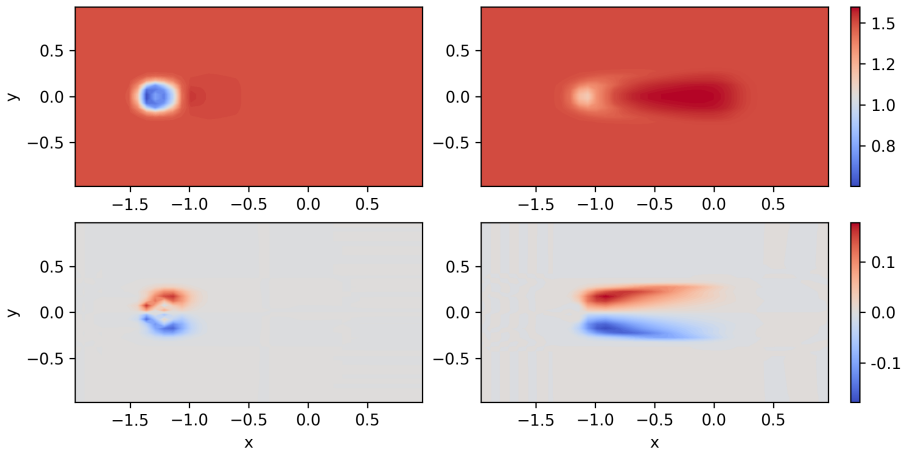
When analysing the reconstructed results it became apparent, that the transformation parameter does not significantly affect the shape of the bolus. It can only be too low, which results in a "damming" of the arterial concentrations. This can be seen in Figure 5.12.

In Section 5.1 we have already shown that the reconstructed solutions are quite sensitive with regard to the starting conditions. Another aspect to which the reconstruction is sensitive is the regularisation parameter. Because the influence of the transformation parameter is minor, setting the regularisation parameter too high, results in the reconstruction massively reducing the transformation parameter in value in order to reduce the overall cost functional value, while ignoring the increasing error between the artificial and reconstructed data. This is visualised in Figure 5.13.

In Run 1.2_T, seen in Figure 5.13, we used $\lambda = 1e^{-4}$ as regularisation parameter for all three parameters. We can clearly see that the adaption of the cost functional in the last step is only decreasing because we decrease the regularisation terms. It is also visible that the mismatch between reconstructed and artificial data increases. This behaviour results in bad reconstructions not only visually but also with regard to the accuracy of the reconstructed parameters. To avoid this behaviour we decrease the regularisation parameter for the transformation parameter.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

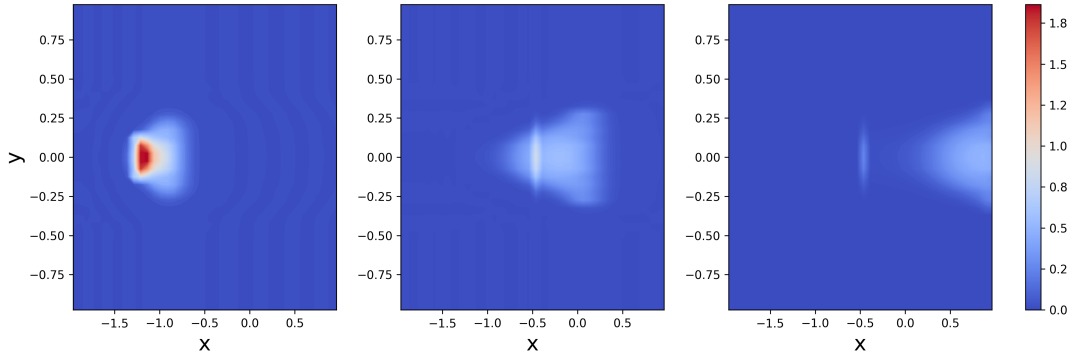


(b) Reconstructed velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.

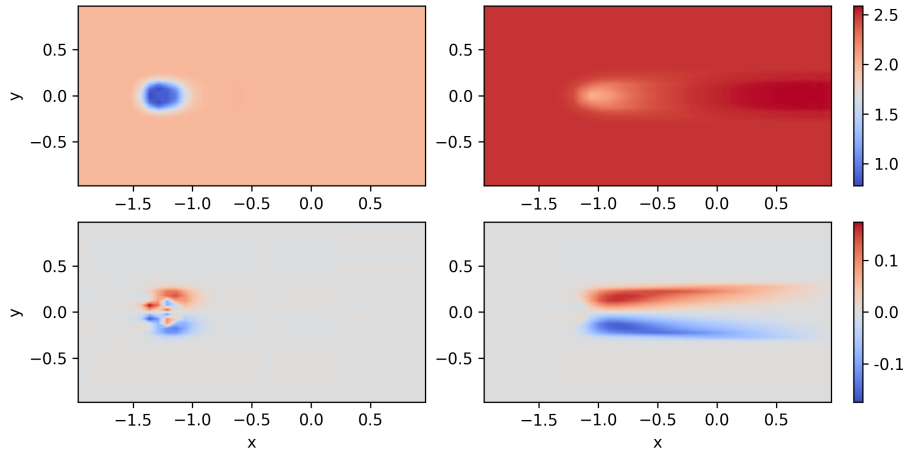
Figure 5.11: Reconstruction of problem 9 using the two-compartment model - Modelling of diffusive behaviour clearly visible (run 9.1_T).

An adjustment to $\lambda_\kappa = 1e^{-5}$ yields a different reconstruction and a more sensible handling of the transformation parameter, compare with Figure 5.14. We can still see a decrease in the regularisation parameter and a minor increase in the mismatch but the overall behaviour is satisfactory.

Similar to the advection-diffusion model there are no strong linear relations between the model parameters of the two-compartment model. But we can see a minor linear relation between the velocity v_x^1 and κ for reconstructions applied on artificial data derived using the two-compartment model, see Figure 5.15. This can be



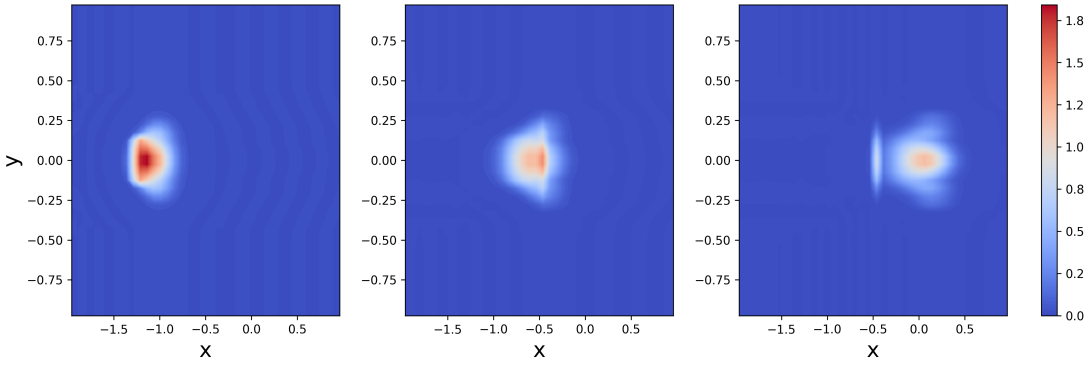
(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.



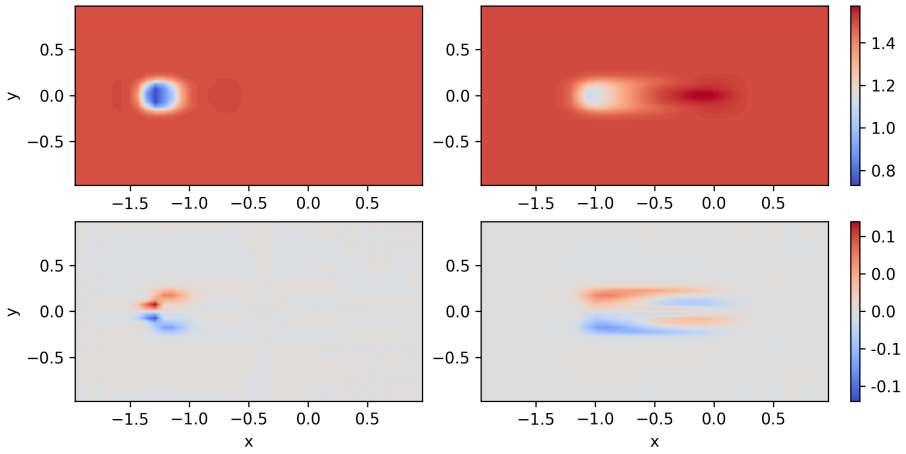
(b) Reconstructed velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.

Figure 5.12: Reconstruction of problem 11 resulting in a low transformation parameter and thus a "damming" at $x = -0.5$ (run 11.3_T).

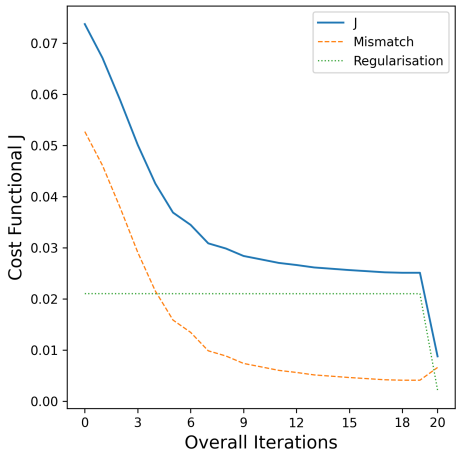
expected because the faster the velocity flows the faster the transformation between arterial and venous blood must occur in order to prevent a build up of arterial blood.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

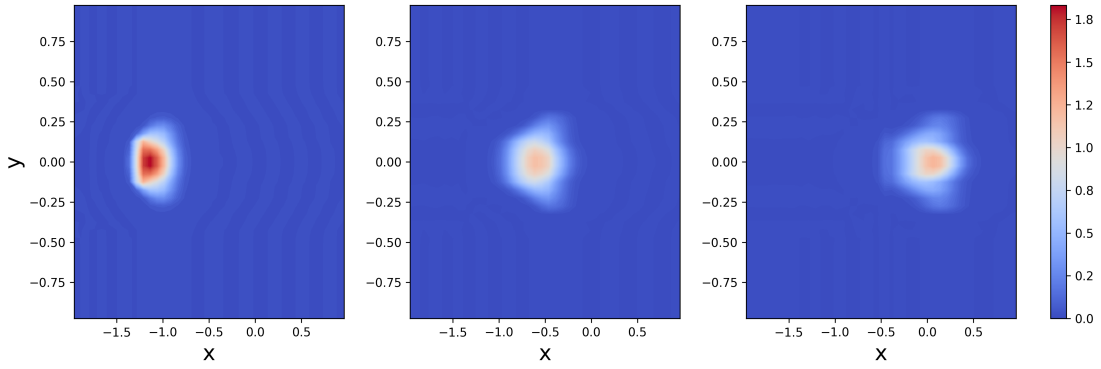


(b) Reconstructed velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.

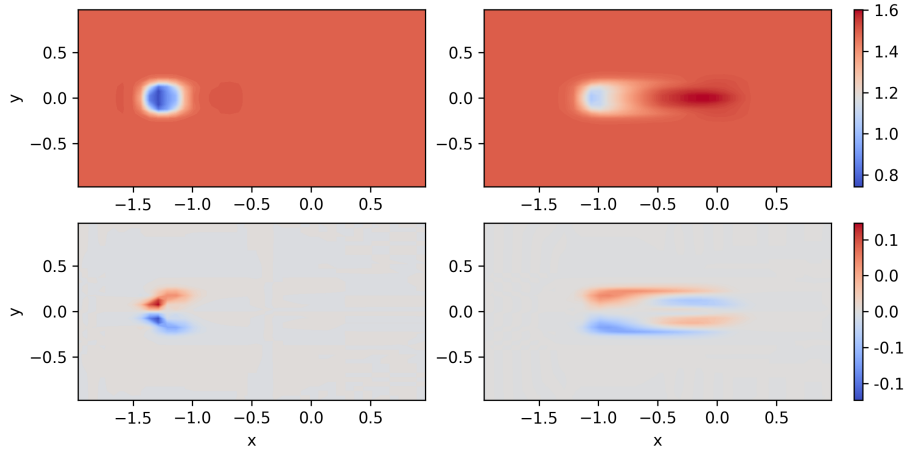


(c) Cost functional

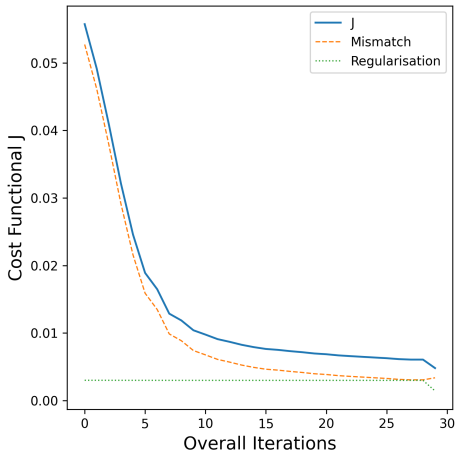
Figure 5.13: Reconstruction of problem 1 with default regularisation parameter $\lambda_\kappa = 1e^{-4}$ - Run 1.2_T.



(a) Reconstructed concentrations at time steps $t_1 = 0.2$, $t_2 = 0.6$ and $t_3 = 1$.

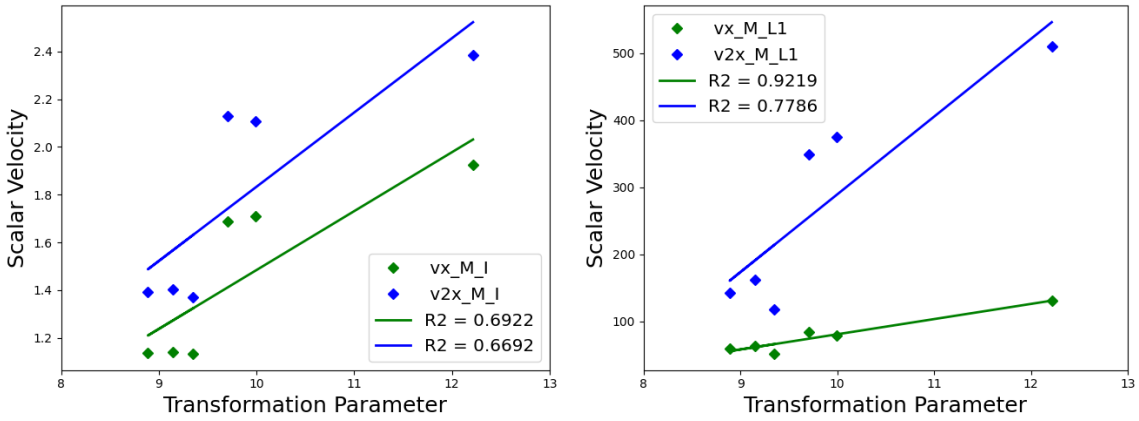


(b) Reconstructed velocities v_x^1, v_x^2, v_y^1 and v_y^2 from *left* to *right* and *up* to *down*.



(c) Cost functional

Figure 5.14: Reconstruction of problem 1 with adjusted regularisation parameter $\lambda_\kappa = 1e^{-5}$ - Run 1_T.



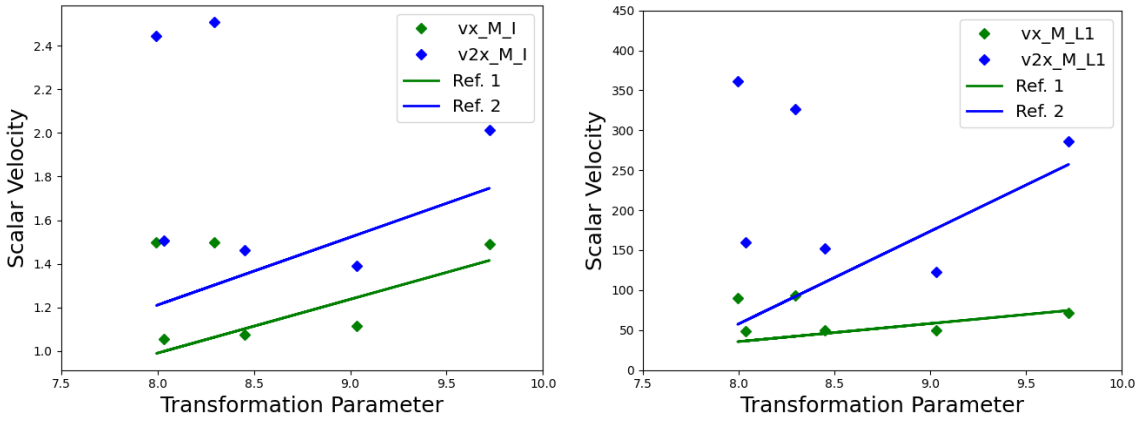
(a) Derived with mask (M) and scalar velocities derived using integral "I".

(b) Derived with mask (M) and scalar velocities derived using "L1".

Figure 5.15: Relation between κ and v_x for integral-norm and L1-norm for derivation of scalar velocities - Runs $1-6$ - including a linear fit with R^2 value.

This relation cannot be seen for data derived by the advection-diffusion model. This is the case because some reconstructions indeed show a build up of arterial blood and therefore cannot reproduce the expected relation by design. This can be checked in Figure 5.16, the linear lines represent the expected relation between the transformation parameter κ and the velocities v_x^1 and v_x^2 derived from the linear fit of the relation for problem 1 to 6. If our reconstruction is under or near this linear fit, see blue and green trend-line in Figure 5.16, we expect to see no build up of arterial blood. But if the reconstructed parameters lie above the line we see a build up of arterial concentrations. This is indeed the case for the two runs producing the scalar v_x values, which are far away from the line. Even though the reconstruction is quite fitting, we would expect the results to be even better if the "damming" is not observed. Thus, checking if the velocities in x-direction, especially the arterial velocity v_x^1, fulfill this relation can help us judge the reconstruction results.

Otherwise, there are no linear relations between model parameters of the two-compartment model thus we expect the remaining reconstructed parameters to be independent from each other.



(a) Derived with mask (M) and scalar velocities derived using integral "I".

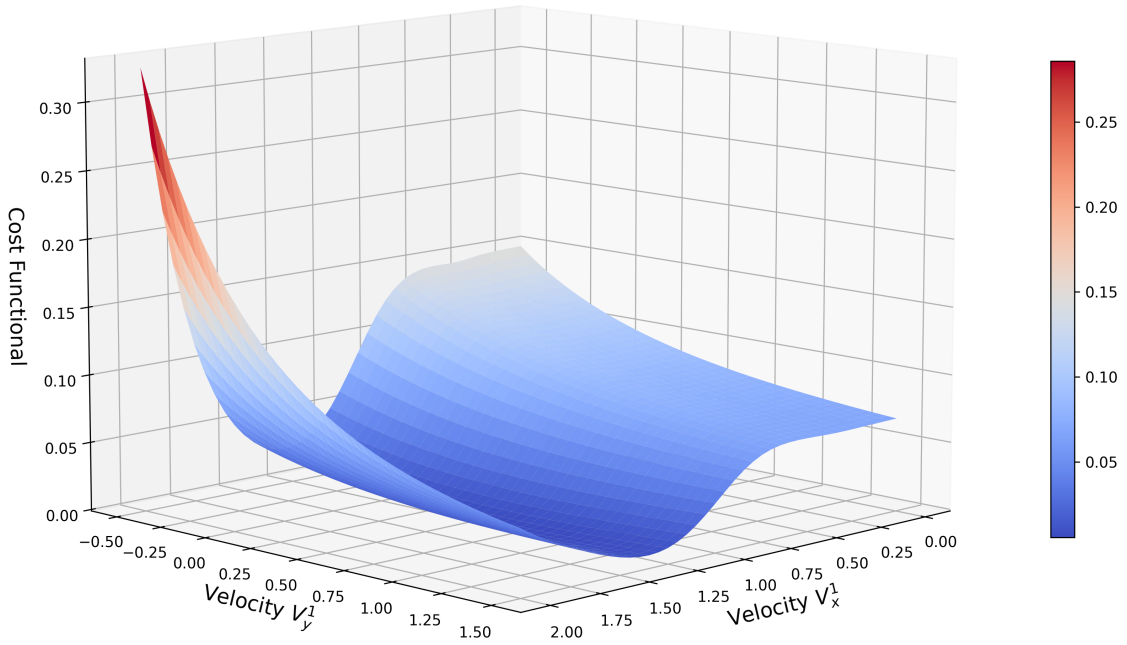
(b) Derived with mask (M) and scalar velocities derived using "L1".

Figure 5.16: Relation between κ and v_x for integral-norm and L1-norm for derivation of scalar velocities - Runs $7 - 12$.

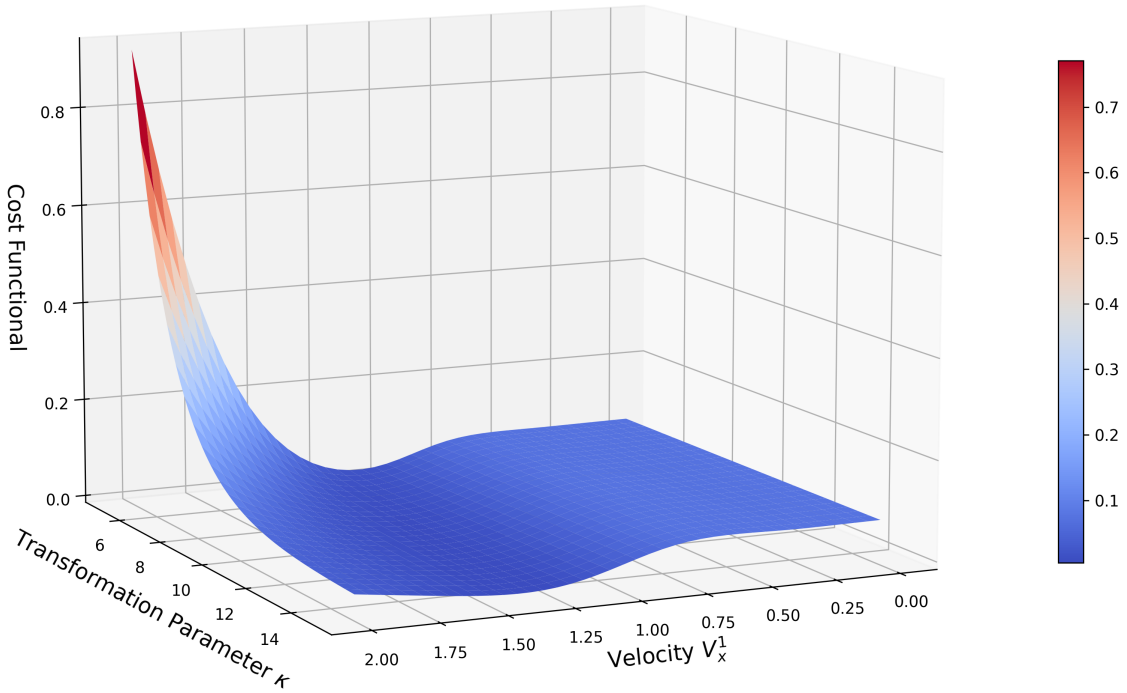
5.3.2 Loss Landscape

As for the advection-diffusion model we want to have a look at the loss landscape for the two-compartment model, this time for problem 1. Again we present a simplified cost functional plot, where we fix all except two parameters and calculate the cost functional values for two varying scalar variables. In Figure 5.17a we varied the velocity v^1 in x-direction over 41 equidistant values from $[0, 2]$ and the velocity in y-direction, which is defined by $v_y^1 = a\, y$, over 41 equidistant points $a \in [-0.5, 1.5]$. Here again, we can see a valley which is hard to navigate for the optimisation algorithm. Furthermore, we can see flat areas which are even worse, because we are not even near a local minimum in one parameter and the algorithm will most likely terminate inside the plane, resulting in a failed optimisation.

In Figure 5.17b we varied again the velocity v^1 in x-direction choosing 41 equidistant points from $[0, 2]$ and the transformation parameter κ with 51 equidistant values from $[5, 15]$. The negative effect of the transformation parameter being set too low is clearly visible. Beside this, the area is mostly flat but close to the optimal value, which shows that for example setting the transformation parameter too high has almost no effect on the cost functional. Furthermore, the loss landscape explains why a good starting value is necessary and why the optimisations fail to achieve good results if the starting parameters are too far off.



(a) Varying velocities v_x^1 and a in $v_y^1 = a\,y$



(b) Varying velocity v_x^1 and transformation parameter κ

Figure 5.17: Cost functional values for only two variable scalar parameters for problem 1. The other parameters are fixed and set to the true values.

5.4 Leray-Projection in Reconstruction

When modelling blood flow through an organ it can be helpful to model or reconstruct velocities without divergence to obtain realistic results. This can be achieved by using a Leray-Projection, which we introduced in Section 3.5. We tried two variants of applying the Leray-Projection during the reconstruction. The first is to apply the projection after each velocity update and therefore multiple times in every step of the optimisation algorithm, depending on how fast we find a fitting step size. The second variant is to apply the Leray-Projection only once after the velocities are found by the optimisation algorithm before the transformation or diffusion parameter is reconstructed. A third possibility is to start with a divergence-free velocity field and use the Leray-Projection only on the gradient of the velocity. In this case we would reduce the applications of the Leray-Projection needed in the first version to one application per optimisation step. But applying the Leray-Projection only on the gradient did not result in a satisfactorily reduced divergence of the velocity field, probably due to cumulative errors, thus this idea has been omitted. A schematic representation of all three application points can be found in Figure 5.18.

An overview of results showing the usefulness of the Leray-Projection can be found in Table 5.4. We can see that applying the Leray-Projection reduces the divergence in all cases. Furthermore, applying the Leray-Projection after every velocity update results in most of the cases in the lowest divergence. Also if we compare the final cost functional values of the cases with Leray-Projection we get mostly lower values for the second case. This shows that if we know that the velocity field is divergence-free we should apply the Leray-Projection after every velocity update even if it increases the computation time. It becomes also apparent that the divergence can still be quite high, meaning far from zero, even after applying the Leray-Projection. This could be due to the chosen boundary conditions, which are set to zero Dirichlet boundary conditions for easier implementation. In the test scenarios using zero Dirichlet conditions was sufficient but during the reconstruction the velocity fields are adjusted and it is possible that different boundary conditions are more adequate. Furthermore, with every adaptation the velocity fields get more complex and thus the Leray-Projection struggles to obtain divergence-free velocity fields. This could be improved with a finer spatial grid, but changing the spatial grid during the reconstruction is not possible.

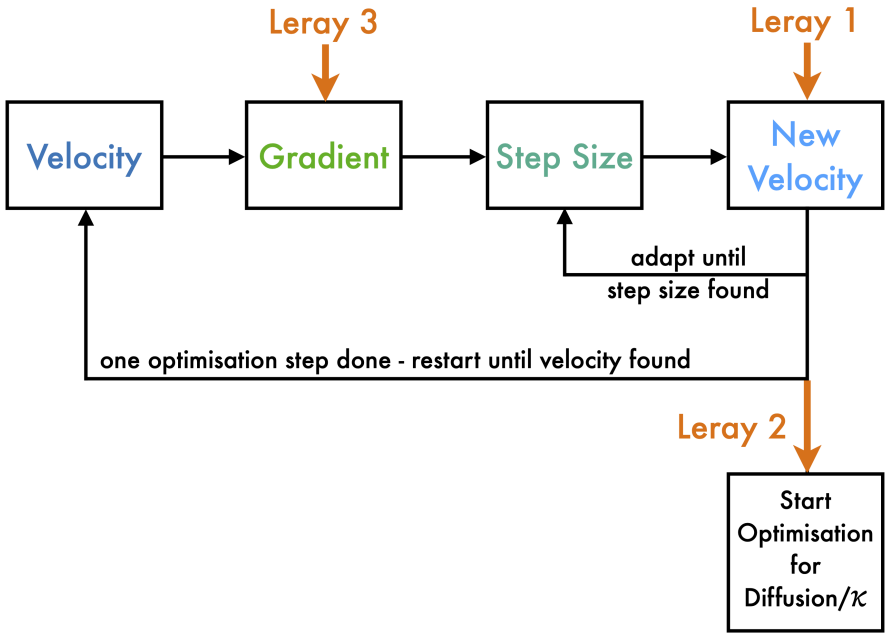


Figure 5.18: Schematic illustration of three possible application points for the Leray-Projection. Leray 1 is applied directly after every velocity update, Leray 2 is only applied on the found final velocity and Leray 3 is applied on the gradient and thus in need of a divergence-free starting velocity.

Run	TwoComp_v_y_T	TwoComp_realistic_T	Diff_simple_D	Diff_v_y_D
No Leray	$J = 0.02183329$ div = 0.6943 Visual: Okay Steps: 10	$J = 0.019376$ div = 0.25794 Visual: Okay Steps: 11	$J = 0.00192518$ div = 0.32966 Visual: Okay Steps: 20	$J = 0.00781705$ div = 0.317683 Visual: Okay Steps: 4
Leray 2: once after vel. found	$J = 0.051415$ div = 0.25794 Visual: Okay Steps: 10	$J = 0.04826503$ div = 0.1103 Visual: Good Steps: 11	$J = 0.00412429$ div = 0.0148308 Visual: Good Steps: 7	$J = 0.02813636$ div = 0.0347783 Visual: Okay Steps: 4
Leray 1: applied after every vel. update	$J = 0.01460223$ div = 0.20676 Visual: Good Steps: 14	$J = 0.02533685$ div = 0.1578 Visual: Very Good Steps: 5	$J = 0.00602691$ div = 0.0018344 Visual: Good Steps: 3	$J = 0.00205877$ div = 0.0288764 Visual: Very Good Steps: 15

Table 5.4: Efficiency comparison for application of Leray-Projection.

When we apply the Leray-Projection we restrict the inverse solver to divergence-free velocity fields. This means that we do not find the same minimum as when using the same starting conditions without the Leray-Projection. This can result in two effects, the first being that we have fewer options and therefore do not find a good descent direction, which results in fewer steps and a worse overall result. The other option is that we indeed find a good descent direction, which the algorithm does not find without the Leray-Projection, and therefore get an overall better result.

If we use the Leray-Projection only once after a fitting velocity field was found, we adapt the velocities and therefore the cost functional, in general, by increasing it. This, again results in a different solution and how efficient it is depends on how much the velocity field changed compared to the previous velocity field and cost functional value. If the adaptation in the velocity field is minor, the reconstruction is improved by the Leray-Projection, but if there are considerable changes to the velocity field after applying the Leray-Projection the reconstruction is only able to change the transformation or diffusion parameter, which cannot correct errors made in the velocity field. Therefore, the finished results are then unsatisfactory and often have a higher cost functional value than before the application of the Leray-Projection.

A positive aspect of applying the projection in general, is that the visual results are always better. This is the case because the Leray-Projection smoothens the velocity fields a bit, thus spreading the concentrations more evenly. This is also one of the reasons why the advection-diffusion model reconstructs velocity fields with lower divergence. Meaning, because it also includes the diffusion parameter the concentrations are overall smoother and the velocity fields less complex, because they are less needed to adapt the shape of the bolus. Another aspect is that we only have one velocity field compared to the two of the two-compartment model.

In conclusion, we can say that the Leray-Projection can improve the reconstruction process. But if the algorithm only does a handful of steps the user should either retry with other starting conditions or try the reconstruction without the Leray-Projection for comparison. Using the second variant and applying the Leray-Projection only after velocities mostly results in more problems than advantages, therefore, we do not encourage this usage.

5.5 Relation between Two-Compartment and Advection-Diffusion Model

To compare both approaches we analyse and explain a few parameters and aspects, including the cost functional values, runtimes and the linear relation between parameters of both models. First of all we can see that the reconstructions using the advection-diffusion model result in general, in lower cost functional values, compared to the two-compartment model. This can be explained via the overall smoother behaviour of the advection-diffusion model, which results in lower cost functional values, because the artificial data is derived with a smooth bolus and the smoother the reconstructed concentrations, the lower the mismatch. Another, rather minor, aspect is that the advection-diffusion model has fewer regularisation parameters than the two-compartment model, which also results in a generally lower cost functional value.

As a second aspect we can compare the run time of both algorithms. Here the advection-diffusion model (Figure 5.20) is also superior to the two-compartment model (Figure 5.19). In general, it takes more than twice as long to reconstruct the parameters using the two-compartment model as it does to reconstruct the parameters using the advection-diffusion model. This is the case because the reconstruction with the two-compartment model uses an average of 26 steps and the reconstruction using the advection-diffusion model only 17. The average computation times per step are roughly 11 minutes for the advection-diffusion model and 16 minutes for the two-compartment model, which is also in favour of the advection-diffusion model. Again, this is to be suspected because in the advection-diffusion model we only have to adjust half the velocities and therefore it should result in lower computation times and fewer steps.

After analysing the models individually we now have a closer look at the relations of these two models, because they both model the same medical problem the results should be comparable, even if it is only visually. First, we have a look at the reconstructed velocities for problems 1 to 12. The results for the velocities in x-direction can be found in Figure 5.21.

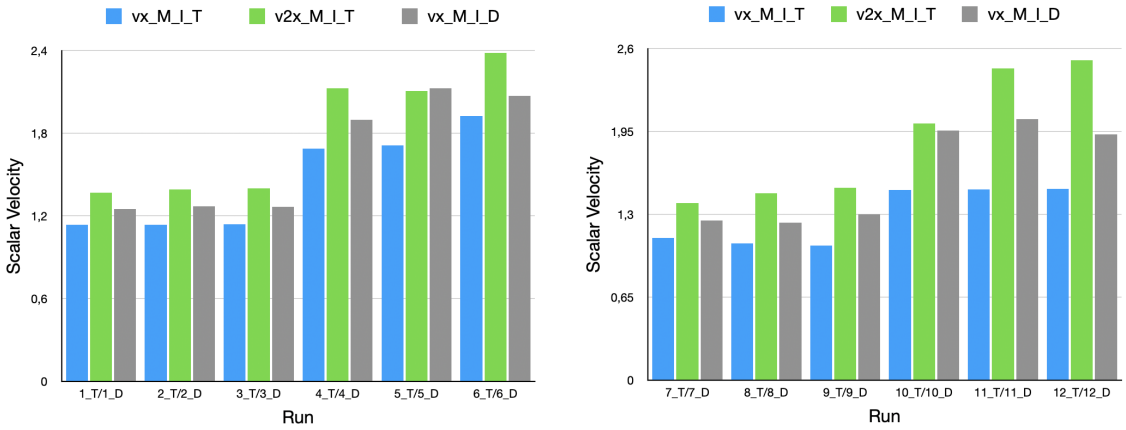
If we have a look at the parameters used for the creation of the artificial data

Nr.	Name	Time	Steps in Vel	Steps in Kappa	Total Steps	Minutes per Step
1.1	1 (Low) Solver: CG	8:30	23	1	25	20m 24s
2.1	2 (Middle) Solver: CG	7:39	28	1	30	15m 18s
3.1	3 (High) Solver: CG	8:28	34	1	36	14m 7s
4.2	4 (LowFast) Solver: CG	7:48	27	1	29	16m 8s
5.2	5 (MiddleFast) Solver: CG	10:47	37	2	40	16m 11s
6.4	6 (HighFast) Solver: CG	10:52	38	1	40	16m 18s
7.1	7 (Low) Solver: CG	6:31	23	1	25	15m 38s
8.1	8 (Middle) TwoComp CG	7:21	25	1	27	16m 20s
9.1	9 (High) TwoComp CG	6:27	23	1	25	15m 29s
10.3	10 (LowFast) Solver: CG	10:05	28	2	31	19m 31s
11.3	11 (MiddleFast) Solver: CG	4:57	22	1	24	12m 23s
12.2	12 (HighFast) Solver: CG	5:21	21	1	23	13m 57s
	AVERAGE:	463m 30s	26	1	28	15m 53s
	Standard Deviation:	117m 28s	5.9001	0.3892	6.0672	2m 13s

Figure 5.19: Runtimes for $1 - 12\_T$.

Nr.	Name	Time	Steps in Vel	Steps in Kappa	Total Steps	Minutes per Step
7.5	7 (Low) Solver: BFGS - Both	3:22	20	20	41	4m 56s
8.2	8 (Middle) Solver: CG	6:07	20	1	22	16m 41s
9.2	9 (High) Solver: BFGS	2:04	11	0	12	10m 20s
10.1	10 (LowFast) Solver: BFGS	2:17	14	1	16	8m 34s
11.1	11 (MiddleFast) Solver: BFGS	3:08	16	1	18	10m 27s
12.1	12 (HighFast) Solver: BFGS	2:28	12	2	15	9m 52s
1.1	1 (Low) Solver: BFGS	1:58	7	2	10	11m 48s
2.1	2 (Middle) Solver: BFGS	2:29	15	2	18	8m 17s
3.2	3 (High) Solver: CG	4:57	18	1	20	14m 51s
4.5	4 (LowFast) Solver: CG	12:41	43	1	45	16m 55s
5.1	5 (MiddleFast) Solver: CG	13:22	45	1	47	17m 4s
6.3	6 (HighFast) Solver: CG	6:52	23	1	25	16m 29s
	AVERAGE:	195m 0s	17	1	19	11m 7s
	Standard Deviation:	241m 5s	11.8960	5.4627	12.9226	4m 6s

Figure 5.20: Runtimes for $1 - 12\_D$.



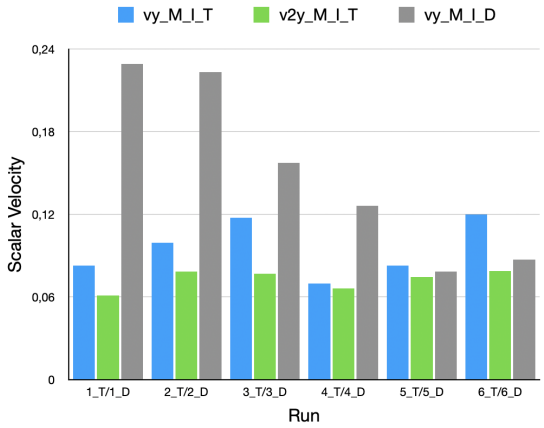
(a) Reconstructed x-velocities for problems 1 - 6.

(b) Reconstructed x-velocities for problems 7 - 12.

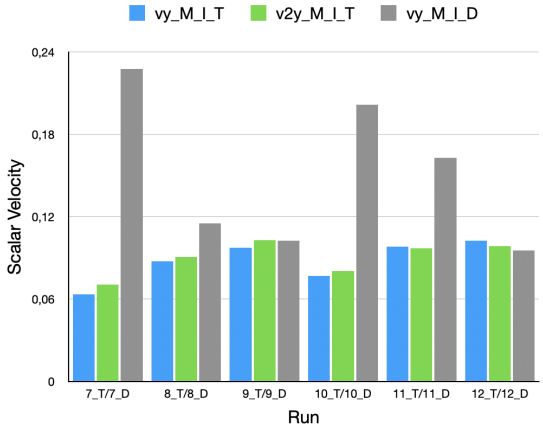
Figure 5.21: Reconstructed velocities in x-direction, scalar value of velocities derived with the integral-norm and application of mask (M_I).

we can see that the v_x values for 1 to 3, 4 to 6, 7 to 9 and 10 to 12 are the same in each case. Furthermore, 1 to 3 and 7 to 9 are constructed with low velocities in x-direction and 4 to 6 and 9 to 12 are created with higher v_x values, for a more structured overview, see Table 5.1 and 5.2. We expect to see these differences and similarities in the reconstructed velocities. This is indeed the case and we can also see that the velocity v_x reconstructed using the advection-diffusion model is often the average of the two velocities v_x^1 and v_x^2 reconstructed with the two-compartment model, see Figure 5.21. This similarity shows that the two models perceive the same behaviour in the artificial data. A last aspect visible in Figure 5.21 is that the faster velocities are more complex to reconstruct and therefore result in a higher variance in the reconstructed velocities.

If we now look at the reconstructed velocities in y-direction it is clearly visible that the two models are not comparable and differ quite a lot, see Figure 5.22. This can be explained by the interaction between the velocity v_y and the diffusion D in the advection-diffusion model. If we only look at the v_y's reconstructed with the two-compartment model we can see an increase in the velocities from 1 to 3, 4 to 6, 7 to 9 and 10 to 12, respectively. For 1 to 6 this is expected because the artificial data is chosen that way, see Table 5.1. But because we can see the same behaviour for runs 7_D to 9_D, where only the diffusion was increased in the artificial data



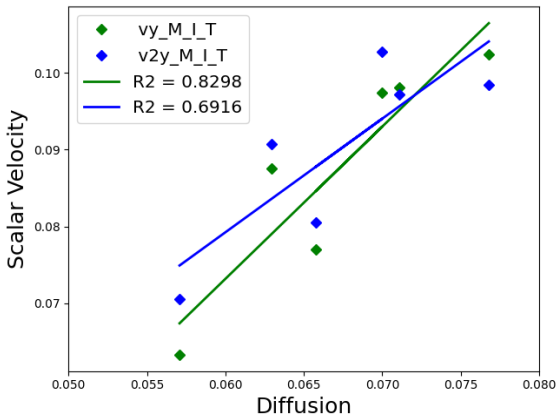
(a) Reconstructed y-velocities for problems 1 - 6.



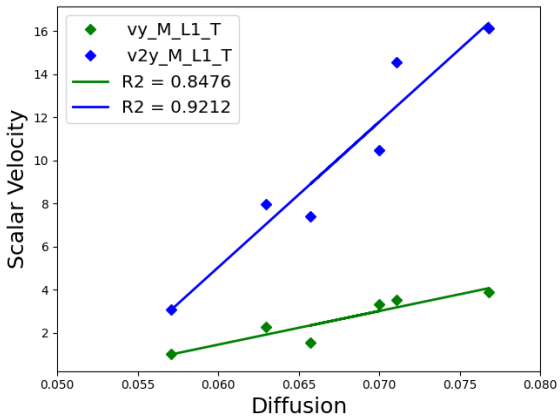
(b) Reconstructed y-velocities for problems 7 - 12.

Figure 5.22: Reconstructed velocities in y-direction, scalar value of velocities derived with the integral-norm (I) and application of mask(M).

and not the velocity in y-direction we expect to see a relation if we compare the velocities v_y^1 and v_y^2 with the diffusion D. This relation can be found in Figure 5.23.



(a) Scalar velocities derived using (I).



(b) Scalar velocities derived using (L1).

Figure 5.23: Plotted relation between model parameters v_y^1, v_y^2 and diffusion D for problems 7 - 12.

Using the L_1-norm the relation becomes even clearer than in the integral-norm. This shows that the two-compartment model is indeed able to pick up on the diffusion in the data and thus model the diffusive behaviour. Therefore, the two-

compartment model can reconstruct a velocity which yields information about the (pseudo-)diffusion in the artificial or measured data.

On the other hand if we look at the parameters of the advection-diffusion model we cannot find a linear relation to those of the two-compartment model. But to be able to reconstruct the transformation parameter using the advection-diffusion model we should find such a relation. Because we cannot find such a relationship, neither between the transformation parameter κ and the velocities nor between κ and the diffusion, we do not expect the advection-diffusion model to be able to generate information about the transformation parameter, and thus perfusion, in any way.

To conclude we can say, that even though the results derived by the advection-diffusion model are quite good and it is reasonably faster in computing time, there is no reason why the model should be able to deliver the medical information, being the transformation parameter κ, needed, which makes it unusable in the present field of application. The two-compartment model on the other hand is not only able to reconstruct the transformation parameter κ, which should relate directly to the perfusion of the area. But can also reconstruct pseudo-diffusion using the two different velocity fields and thus would be a good fit to reconstruct the medical data even if the modeling assumption of having no diffusion is wrong. Looking at the different reconstructions it is visible that the two-compartment model is more versatile, due to the range of free parameters and thus worth the longer computation times.

Chapter 6

Reconstructions on the Way to Ultrasound Measurements

The previous problems were designed with the reconstruction of the parameters already in mind. This means that the parameters used are smoothened, so that no discontinuities appear and the transformation from arterial to venous concentration happens in the middle of the domain. This is important because to derive a good reconstruction the algorithm needs to be able to get information about the behaviour of the parameters. If we, for example, want to reconstruct the velocity v^2 but the concentration in the given data is only modelling the first half, before the transformation, where v^2 has no influence on the concentrations, then we cannot expect the parameter identification to reconstruct v^2. In the following chapter, we try to reconstruct artificial data that was derived to simulate concentrations with medical properties thus modeling the transportation of tracer in the liver more accurately, regardless of particular smooth parameters. Reconstructing the parameters for these concentrations is an intermediate step for reconstructing data derived via actual ultrasound measurements. Furthermore, we reconstruct noisy data in Section 6.3 and 6.4.

The two problems are posed on a time-grid with 120 time steps, on domain $T = [0, 1]$. The space domain is defined by $\Omega = [1, 3] \times [1, 3]$, with 40 cells in x- and y-direction and three ghost cells on either side. Furthermore, the initial conditions for both problems are given by $u_0 = 3\, e^{-\frac{(x-1.1)^2+y^2}{0.02}}$ and $w_0 = 0$.

6.1 Artificial Problem with Wide Transformation Domain (WTD)

The artificial data was derived with parameters that imitate medical behaviour as explained in Section 2.4 Table 2.6. We have a high velocity after the inflow, which reduces over space, and a velocity before the outflow boundary which starts slow and grows over space. Furthermore, the maximum value is lower than the maximum of the inflow/arterial blood flow. The velocities in y-direction are chosen to model a spreading out of the concentration after the inflow and a slimming before the outflow, all represented in Figure 6.1.

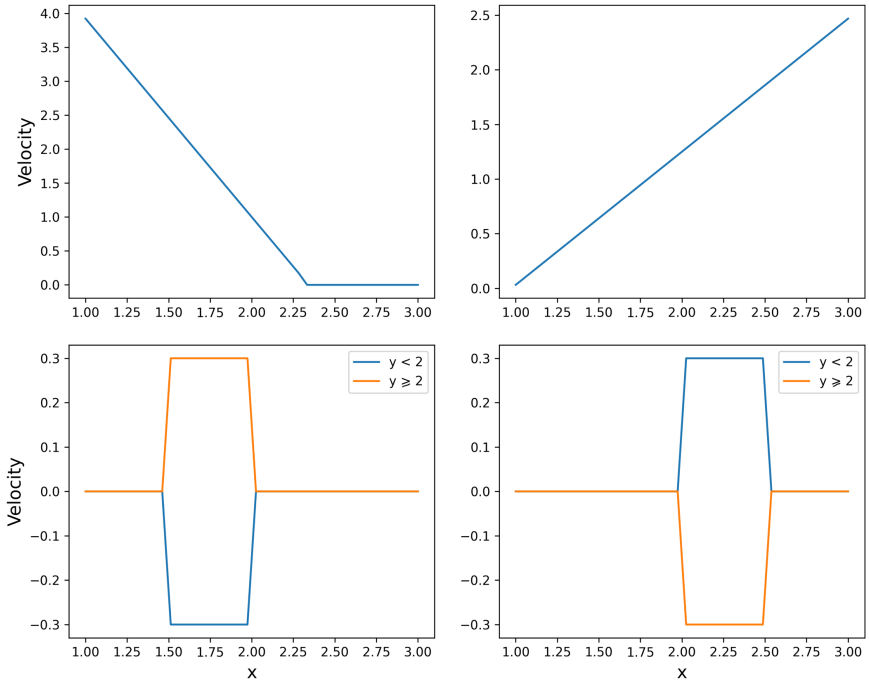


Figure 6.1: True velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) for problem WTD plotted over x if not indicated otherwise constant in y.

In this problem the transformation parameter is defined on a wide domain and set to $\kappa = 7$, see Figure 6.2. These parameters result in concentrations that are visualised in Figure 6.3. The simulation starts with an empty domain before a bolus of concentration enters, expands in x- and y-direction until $x = 2$, before contracting down and exiting the domain.

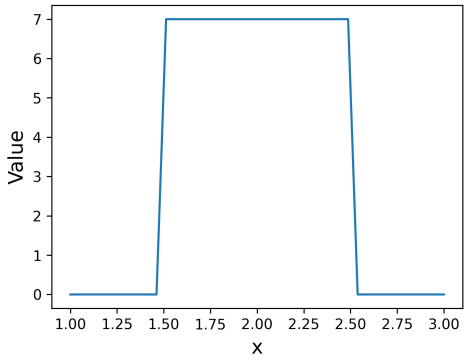


Figure 6.2: Transformation parameter κ for problem WTD plotted over x, constant in y.

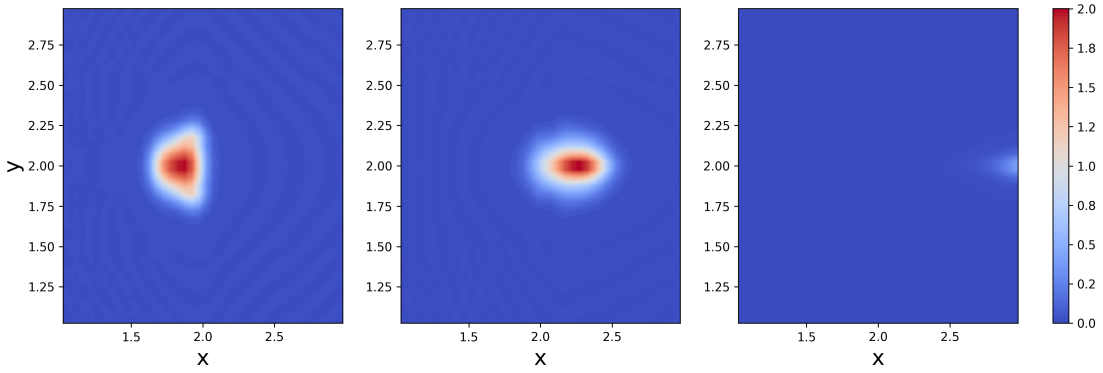


Figure 6.3: Concentrations of artificial data computed using parameters of problem WTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ (from *left* to *right*).

6.1.1 One Round of Velocity and Transformation Parameter Reconstruction

As in the previous chapter, we start the reconstruction with only one round of adapting the velocities and the transformation parameter. This provides reasonable results with significantly less computation time and shortens the computational time when trying out different parameters, as for example, starting values or regularisation factors. Before feeding the parameter identification algorithm with the concentrations, the data has to be adapted. The major problem is the inflow of concentration,

because in order to simulate this, the true values need to be known at the ghost cells. To avoid this difficulty we cut off the time steps where the concentration has not fully entered the domain yet and then choose the new first time step as initial condition u_0. Furthermore, we need to set the domain in which the transformation parameter κ is active and the constraint for the v_x^1 velocity, which is, as before, set towards the end of the κ domain to ensure that the arterial concentration gets fully transformed into venous concentration. This is done using the true constraints or for the transformation parameter a slightly smoothened constrained using the same sigmoid function as in Section 4.4.

To run the parameter identification we further need to set the starting parameters, which are set to $\kappa = 18$, $v_x^1 = 2$, $v_x^2 = 2.2$, and $v_y^1 = v_y^2 = 0$. The tolerances for the stopping criteria, which ensure that the gradient norm is large enough and that the decrease in the cost functional is sufficient, are set to $\text{tol}_1 = 1e^{-5}$ and $\text{tol}_2 = 1e^{-7}$, respectively. And lastly, the regularisation parameters are set to $\lambda = [1e^{-4}, 1e^{-4}, 1e^{-5}]$ and the threshold for the step size is set to $\text{tol}_\alpha = 5e^{-5}$. Using these parameters and the conjugate gradient descent with DY parameters the transformation parameter was reconstructed with $\kappa = 10.6$. The reconstructed concentrations are shown in Figure 6.6 and the corresponding velocities in Figure 6.4. The flow of concentrations is similar to the artificial concentrations seen in Figure 6.3. The flow velocity in x-direction is reasonably reconstructed and we see the wanted expanding and contracting of the concentration over time.

The success of the reconstruction is judged by analysing the cost functional and gradient norm of the velocities, which are plotted in Figure 6.5. We see the wanted reduction by over 80% in the cost functional and a decrease in the gradient norm of the velocities. The final gradient norm is $||\nabla J_v|| = 0.0027$, which is low enough to indicate a stationary point. Furthermore, the final gradient norm value of κ is $||\nabla J_\kappa|| = 1e^{-4}$ and the final cost functional value is $J = 0.0129$. These all indicate a successful reconstruction. To obtain this successful optimisation, we had to adapt the tolerance parameters for the gradient norm and change in cost functional. Thus, adapting the tolerance parameters is an important step in calculating a satisfactory reconstruction and should be checked during the reconstruction.

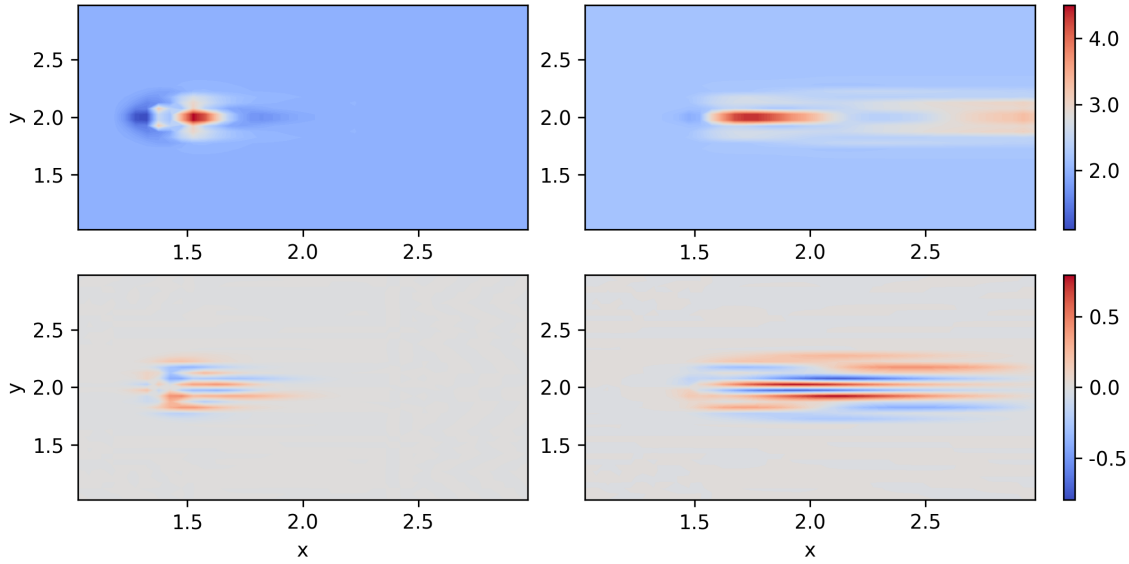
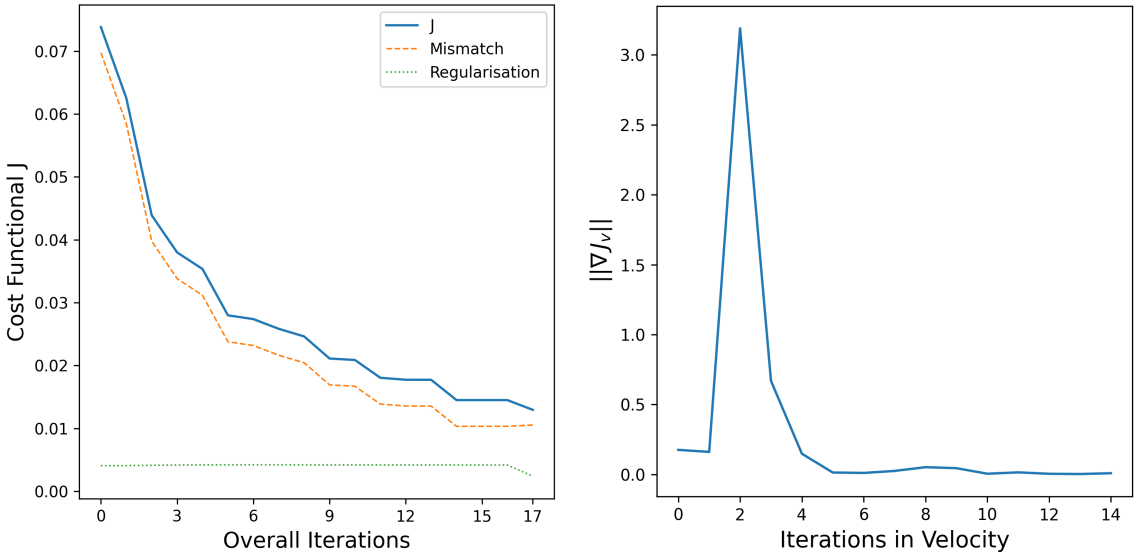


Figure 6.4: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) of problem WTD.



(a) Cost functional

(b) Gradient norm of velocities

Figure 6.5: Indicators of success for the reconstruction of problem WTD.

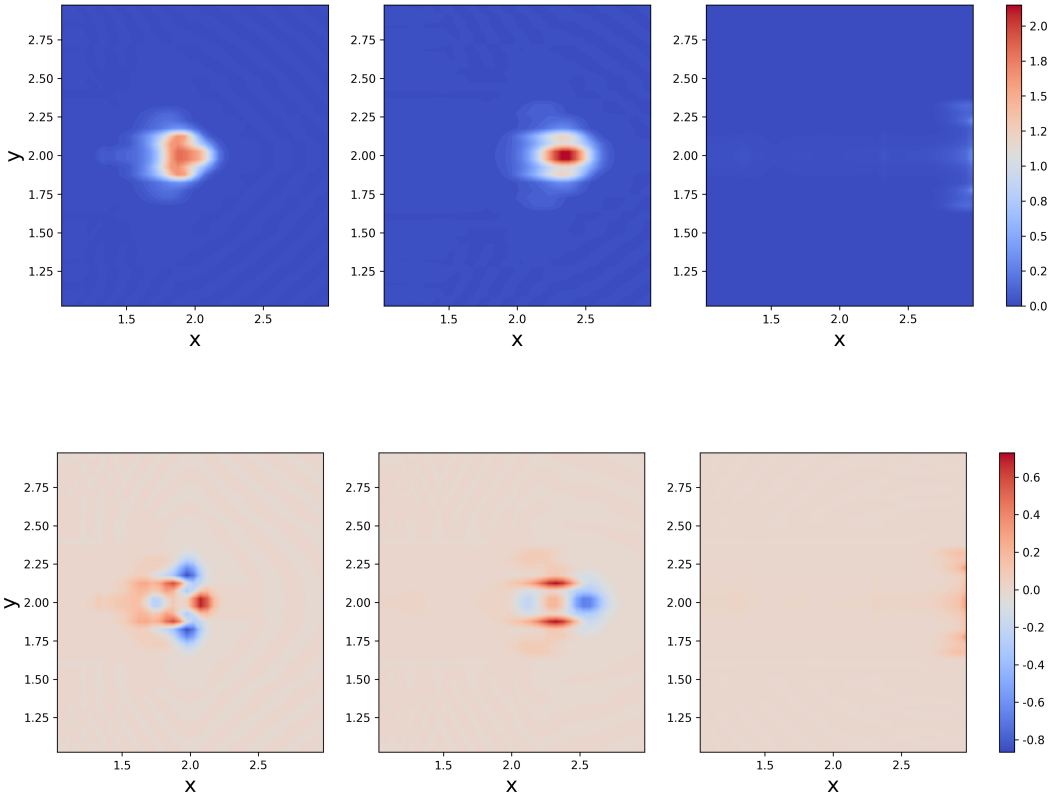


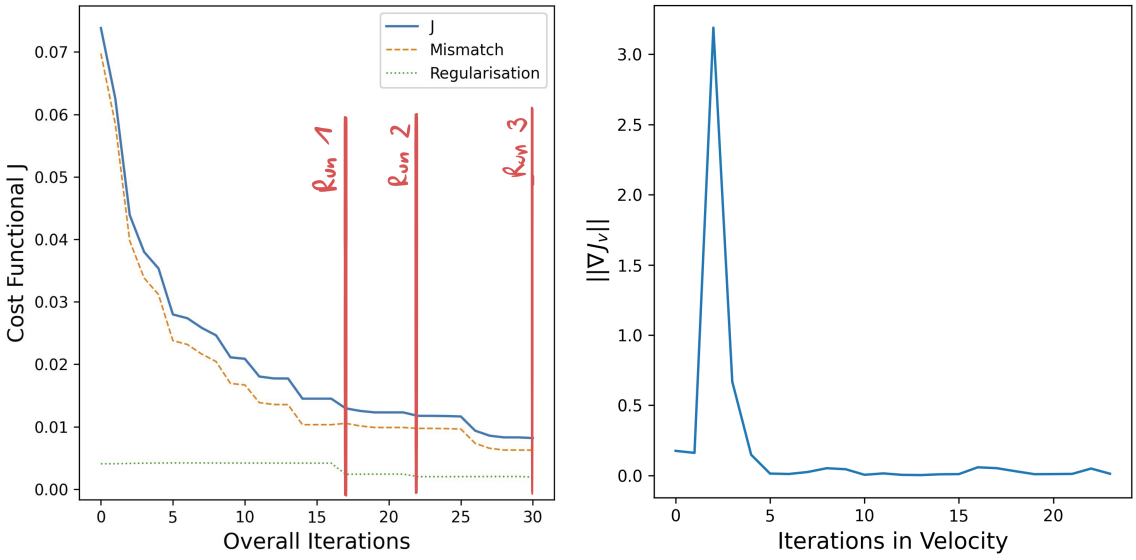
Figure 6.6: Reconstructed concentrations (*top*) and mismatch of reconstructed solution and artificial data (*bottom*) for problem WTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ (from *left* to *right*).

6.1.2 Multiple Rounds in Reconstructing Velocities and Transformation Parameter

So far, we reconstructed the parameters in a way, where we first calculated the velocities before reconstructing the transformation parameter. This approach is reasonable, because the velocities have significantly more influence over the concentrations compared to the transformation parameter. Nevertheless, adapting the transformation parameter also changes the concentrations. Thus, it is reasonable that the gradient in the velocities changes after optimising the transformation parameter, which could result in improved parameters that fit the data even better.

This means that we have to apply the optimisation multiple times until no further steps, in the velocities and the transformation parameter, are taken. This optimisation with multiple rounds was applied to problem WTD and the results are presented in the following.

After deriving the results seen in Section 6.1, the optimisation process was restarted with the calculated parameters. Two restarts were needed for the optimisation to converge. In the second run the changes in velocity and the transformation parameter were minor, but in the third run the reconstruction could find parameters that decrease the cost functional further, see Figure 6.7a. In Figure 6.7 we can see



(a) Cost functional

(b) Gradient norm of velocities

Figure 6.7: Indicators of success for the reconstruction of problem WTD with 3 optimisation rounds.

that the gradient norm of the velocities is neither increasing nor decreasing after the first run, which took 16 steps in the velocities. This means that reapplying the optimisation changed the results and because the cost functional is significantly reduced, to $J = 0.0082$, we can assume that we are now closer to a good reconstruction.

Figure 6.8 shows that the reconstructed concentrations are spread slightly more evenly and thus behave smoothened, with less points of high concentration occurring. This is an improvement, because the first round of optimisation often reconstructs

concentrations with areas of higher flow and thus the tendency to unevenly spread concentrations. Furthermore, we see that the error, or mismatch to the artificial data, is further reduced.

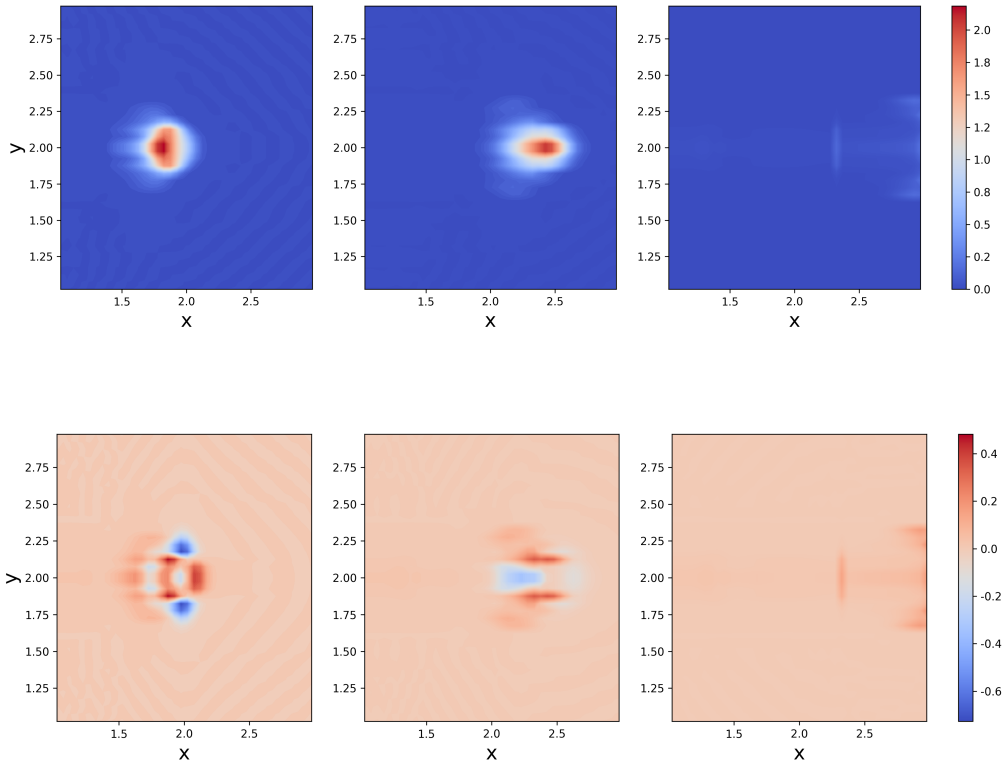


Figure 6.8: Reconstructed concentrations (*top*) and the mismatch of reconstructed concentration and artificial data (*bottom*) for problem WTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ (from *left* to *right*) after 3 optimisation rounds.

If we take a look at the reconstructed velocities, see Figure 6.9, we see that the general behaviour stays the same, but they are some adaptations that result in the smoothened behaviour of the concentrations, see Figure 6.8. Finally the adaptations in the velocities resulted in a better approximation of the transformation parameter κ. The final reconstructed value is $\kappa = 7.24$ which equals almost the true $\kappa_{\text{ex}} = 7$ and is significantly more accurately than the previous calculated $\kappa = 10.6$.

To conclude we can say, that doing only one optimisation run yields a reasonable result, and reconstructs the main behaviour. But applying the optimisation multiple times results in parameters that can reconstruct the finer details and thus

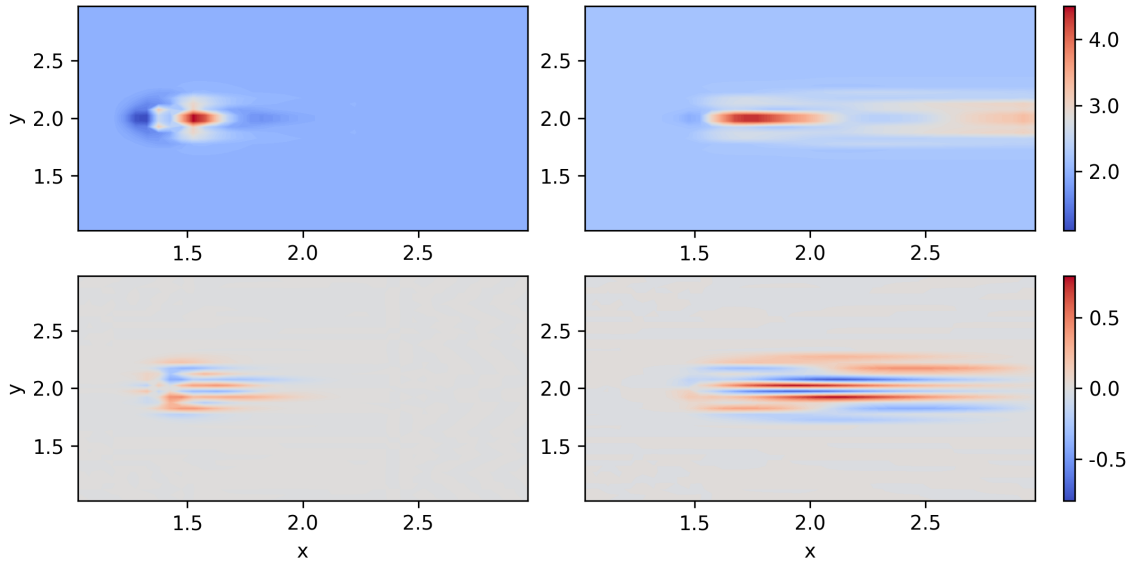


Figure 6.9: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) of Problem WTD for 3 optimisation rounds.

a transformation parameter that fits the data more accurately. Therefore, with regard to medical applications, it is advisable to apply multiple optimisation rounds in order to achieve the best result possible for the transformation parameter even if this results in longer computation times.

6.2 Artificial Problem with Narrow Transformation Domain (NTD)

Here as well, the artificial data was derived with parameters that imitate medical behaviour, see Figure 6.10. The only difference to the velocities in problem WTD is that v_x^1 is set to zero earlier, at $x = 2.1$. This avoids that the arterial concentration exits the domain of transformation without being transformed into venous concentration.

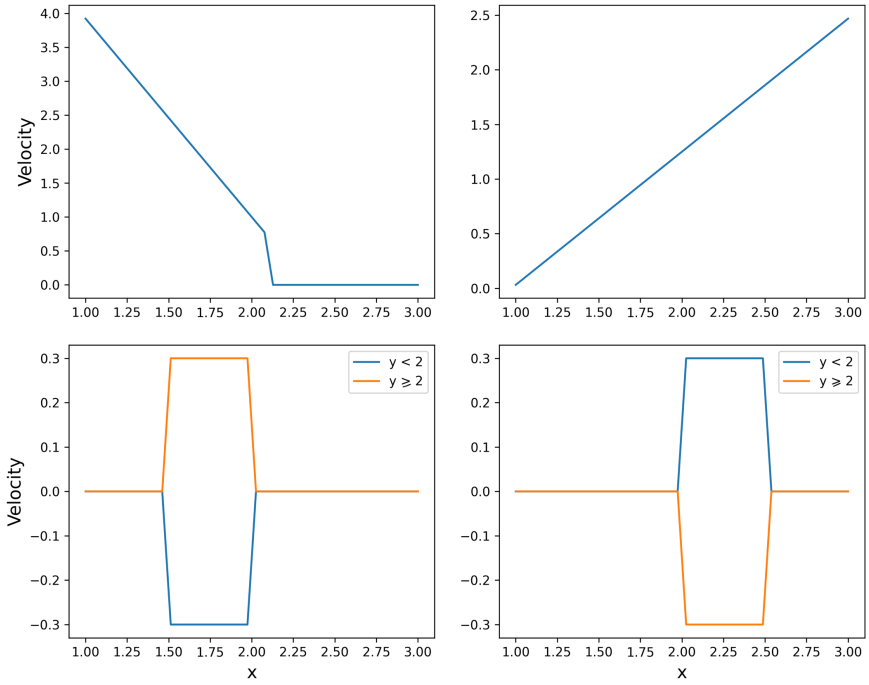


Figure 6.10: True velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) for problem NTD plotted over x if not indicated otherwise constant in y.

In the second problem κ acts on a narrower domain thus modelling a different situation, and was set to $\kappa = 9$, visualised in Figure 6.11. This domain models a smaller organ compared to the previous example and yields the opportunity to study how the reconstruction algorithm handles smaller domains for κ. The presented parameters result in concentrations that are visualised in Figure 6.12. A clear difference is visible to problem WTD. The bolus is more concentrated, because it does not spread out as much in x-direction. This is a result of the narrower domain

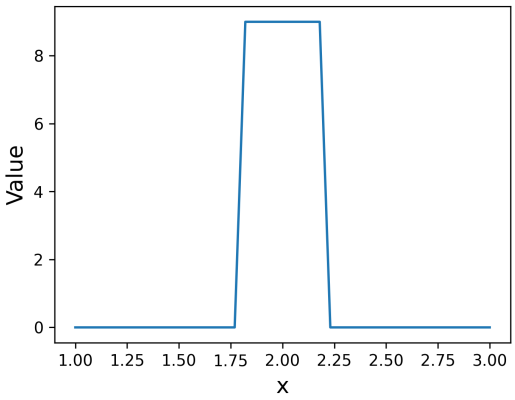


Figure 6.11: Transformation parameter κ for problem NTD plotted over x, constant in y.

for κ which results in less overlap of the areas in which both velocities are active and therefore a more compact form. Here as well, we have to adapt the data by cutting out the first time steps until the whole concentration has entered the domain and then set the resulting concentration to be our starting condition u_0. Furthermore, we need to set the domain in which the transformation parameter κ is active, here again we use the true domain, which is smoothed by applying a sigmoid instead of a step function. The constraint for the v_x^1 velocity is then chosen to be zero from $x = 2.1$ onwards, set towards the end of the κ domain.

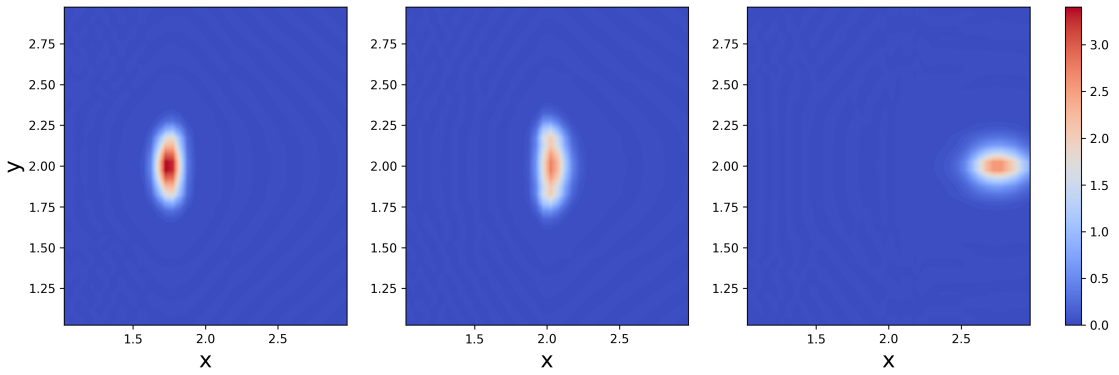


Figure 6.12: Concentrations of artificial data derived using parameters of problem NTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.75$ (from *left* to *right*).

To start the parameter identification we set the starting parameters to $\kappa = 16$, $v_x^1 = 1.8$, $v_x^2 = 2$, and $v_y^1 = v_y^2 = 0$. The tolerances for the stopping criteria, which

ensure that the gradient norm is large enough and that the decrease in the cost functional is sufficient, are here set to $\text{tol}_1 = 1e^{-4}$ and $\text{tol}_2 = 1e^{-5}$, respectively. The regularisation parameters are again set to $\lambda = [1e^{-4}, 1e^{-4}, 1e^{-5}]$ and the threshold for the step size is set to $\text{tol}_\alpha = 5e^{-5}$. The first try of the reconstruction did not result in a sufficient adaptation of the transformation parameter. We adapted the next run by changing the regularisation parameter from $\lambda_k = 1e^{-5}$ to $\lambda_k = 5e^{-5}$. This resulted in a sufficient decrease in κ, which was reconstructed with $\kappa = 11.3$ after the first and $\kappa = 8.86$ after the final, 24'th optimisation round. Thus, showing again how accurate the transformation parameter is reconstructed after multiple rounds of optimisation. The reconstructed concentrations and mismatch can be seen in Figure 6.13. The flow of concentration is similar to the artificial data. It successfully reconstructs the expanding on a narrow domain and reconstructs the velocities in x-direction accurately. On the downside, a small amount of damming is visible, which implies that the velocity in x-direction is too high. Knowing that the transformation parameter is reconstructed almost exactly and not too low.

The velocities reconstructed to achieve these concentrations can be seen in Figure 6.14. As indicator of the success of the reconstruction we have a look at the cost functional and the gradient norm of the velocities, which can be both seen in Figure 6.15. We can see the wanted decrease in both the cost functional and the gradient norm of the velocities. The final gradient norm of the velocity is $||\nabla J_v||_2 = 0.0003$, which is low enough to indicate a stationary point.

Furthermore, the final value of the gradient norm for κ is $||\nabla J_\kappa||_2 = 0.0017$ and the final cost functional value is $J = 0.004$. These all indicate a successful reconstruction. We conclude that adapting the regularisation parameter to achieve a reasonable adaptation in κ is an important step to derive a successful reconstruction.

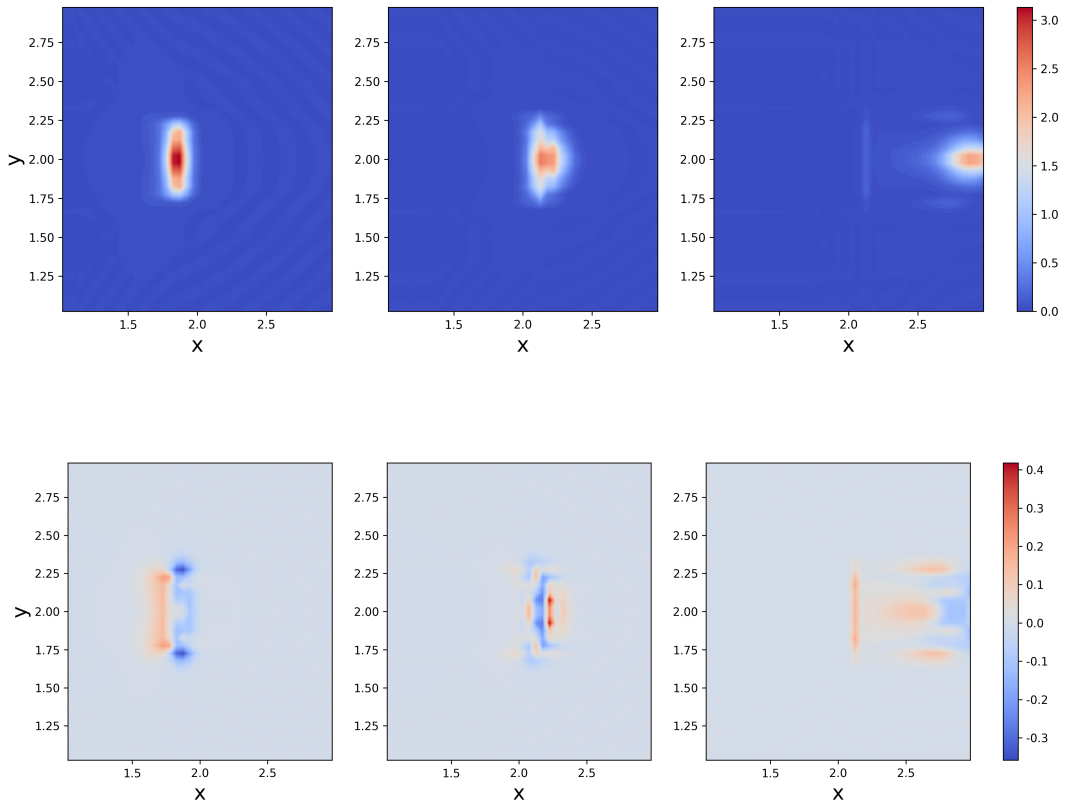


Figure 6.13: Reconstructed concentrations (*top*) and mismatch between reconstructed concentrations and artificial data (*bottom*) of problem NTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.75$ (from *left* to *right*) after 24 optimisation rounds.

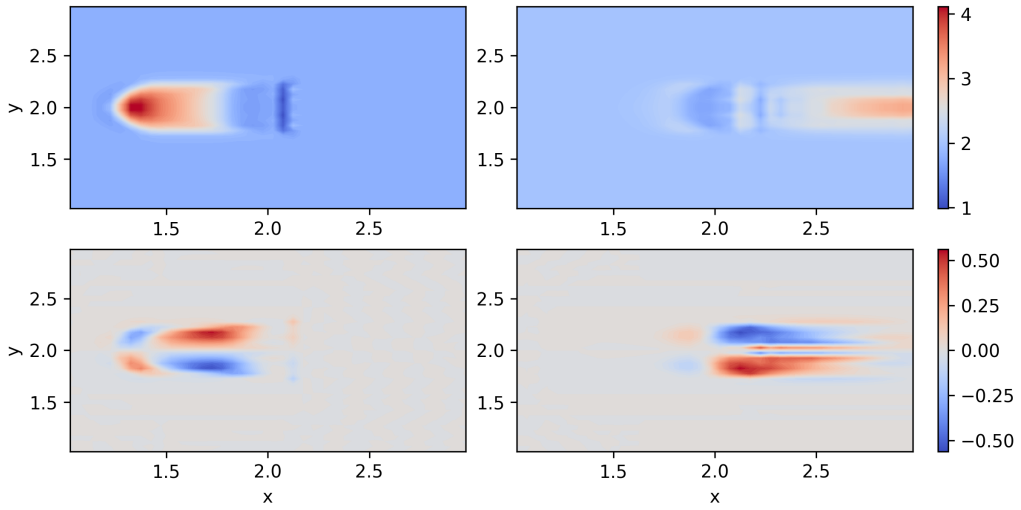
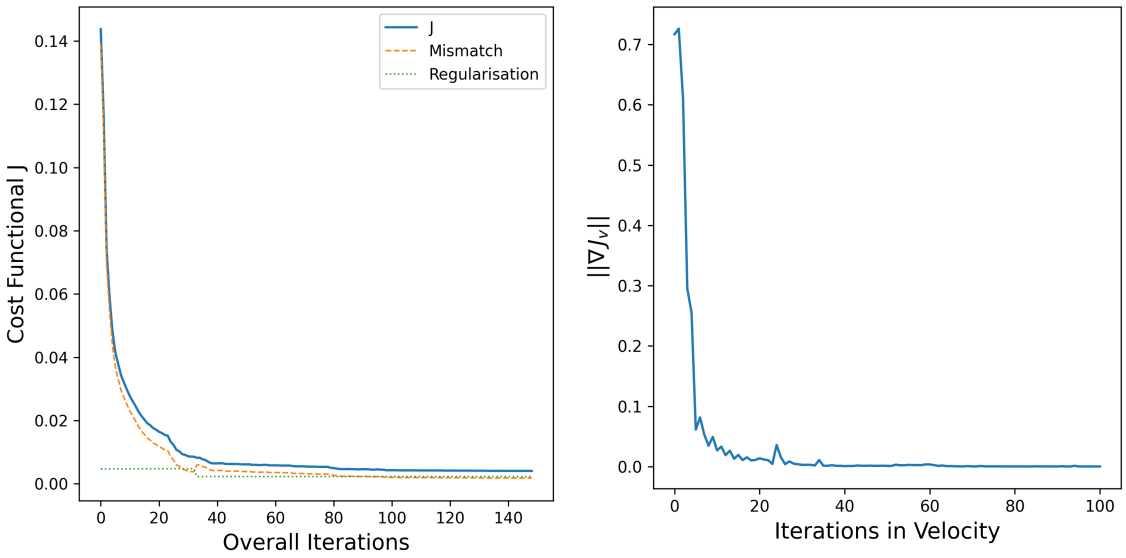


Figure 6.14: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) of problem NTD after 24 optimisation rounds.



(a) Cost functional

(b) Gradient norm of velocities

Figure 6.15: Indicators of success for the reconstruction of problem NTD after 24 optimisation rounds.

6.3 Artificial Problem with Wide Transformation Domain and 5 % Noise

On the way to using the reconstruction solver on ultrasound data, we further examined how the algorithm reacts to noisy data. This was done by adding around five percent Gaussian noise [47]

$$n(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{6.3.1}$$

with $\mu = 0$ and $\sigma = 0.15$, to problem WTD. The five percent are given with respect to the maximal concentration value. The resulting concentrations are visualised in Figure 6.16, the noise is clearly visible. To avoid instabilities at the boundaries the noise was removed there in order to have zero concentrations at the boundaries. This could also be achieved by adding a layer of zero concentration around the data, when its not possible to remove the noise.

Using the same set-up as in Section 6.1, including the same tolerances, regularisation parameters and descent directions, we again reconstructed the velocities and transformation parameter. Four rounds of optimisation were needed before no more

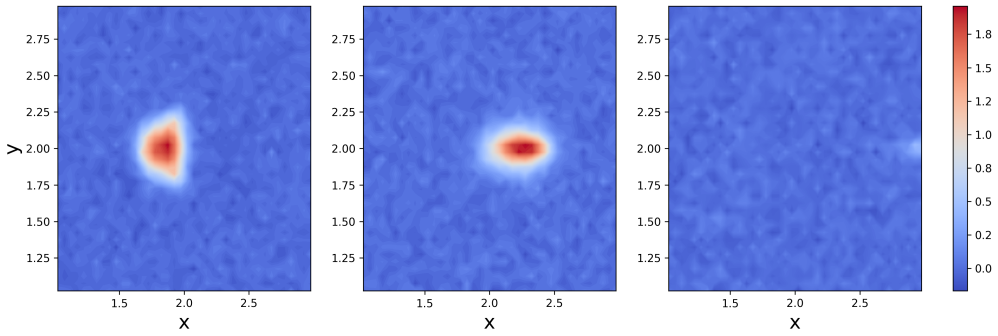


Figure 6.16: Concentrations of problem WTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ with 5% noise.

improvement was found. The resulting concentrations differ quite a lot compared to the reconstruction without noise, see Figure 6.17. Even though we can still see a spreading out and slimming down the concentration bolus is less symmetric and the error to the data increases. Nevertheless, the results look still reasonable and capture the main movement of the concentrations.

If we look at the reconstructed velocities, Figure 6.18, some noisy artifacts become visible for the velocities of the arterial blood. They result from some arterial concentrations introduced through the noisy starting condition which cannot be transformed into venous concentrations, because they are past the transformation domain. In general, the velocity fields are adapted on a larger domain compared to the reconstructions without the noise. This is reasonable because the starting condition is always set as the first measurement and thus is now noisy as well. This results in artifacts of the concentration flow on the whole domain. Nevertheless, we do not expect that these artifacts introduce anymore information about the reconstructed parameters. Furthermore, introducing noisy data resulted into smoothened velocity fields, with less sharp changes in direction. This is most likely also a result of the noisy starting condition.

For medical applications the perfusion or transformation parameter is the most important one. Even with the added noise the transformation parameter has been reconstructed accurately with $\kappa = 6.96$ almost equal the true $\kappa_{\text{ex}} = 7$, see Figure 6.19.

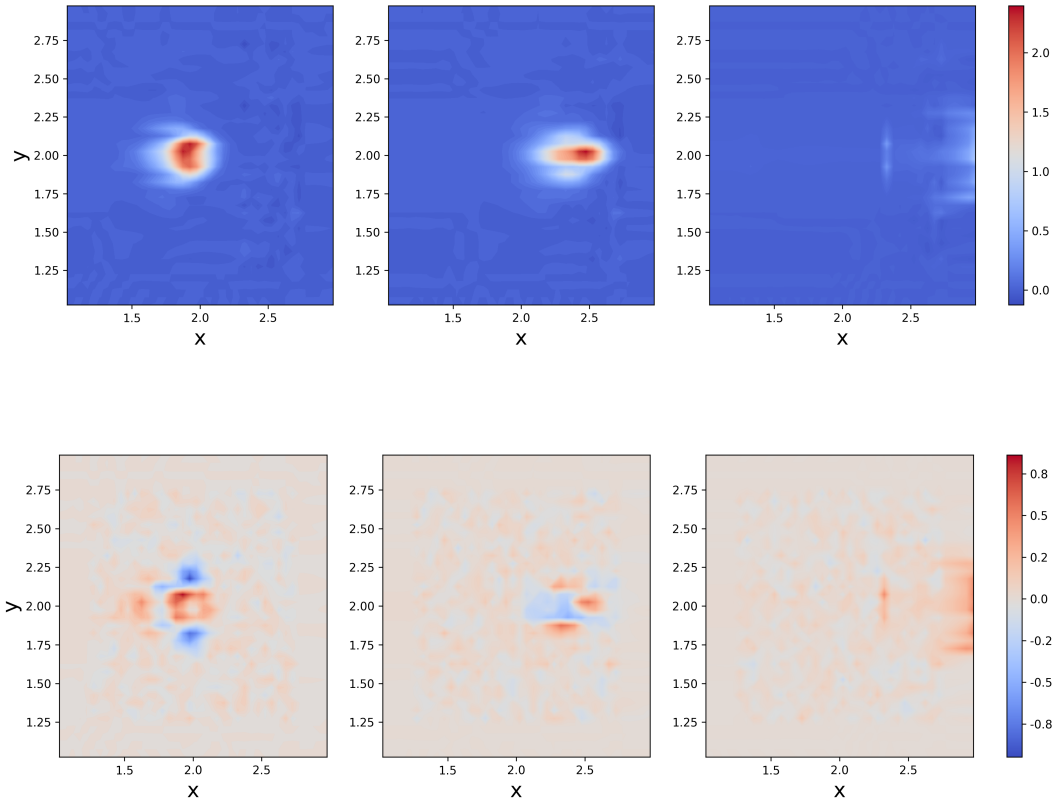


Figure 6.17: Reconstructed concentrations (*top*) and the mismatch of reconstructed concentrations and artificial data (*bottom*) of problem WTD with 5% noise at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ (from *left* to *right*) after 4 optimisation rounds.

Lastly, we have a look at the indicators for a successful reconstruction, see Figure 6.20. The values in both gradient norms indicate a stationary point with $||\nabla J_v||_2 = 0.0032$ and $||\nabla J_\kappa||_2 = 0.0004$. They are of the same order of magnitude as the values without noise. Where we can see a slight difference, compared to the reconstruction without noise, is in the cost functional. Here the value is a bit higher $J = 0.0126$ compared to $J_{\text{no noise}} = 0.0082$. This can be explained by the higher mismatch, which is due to the added noise throughout the whole domain.

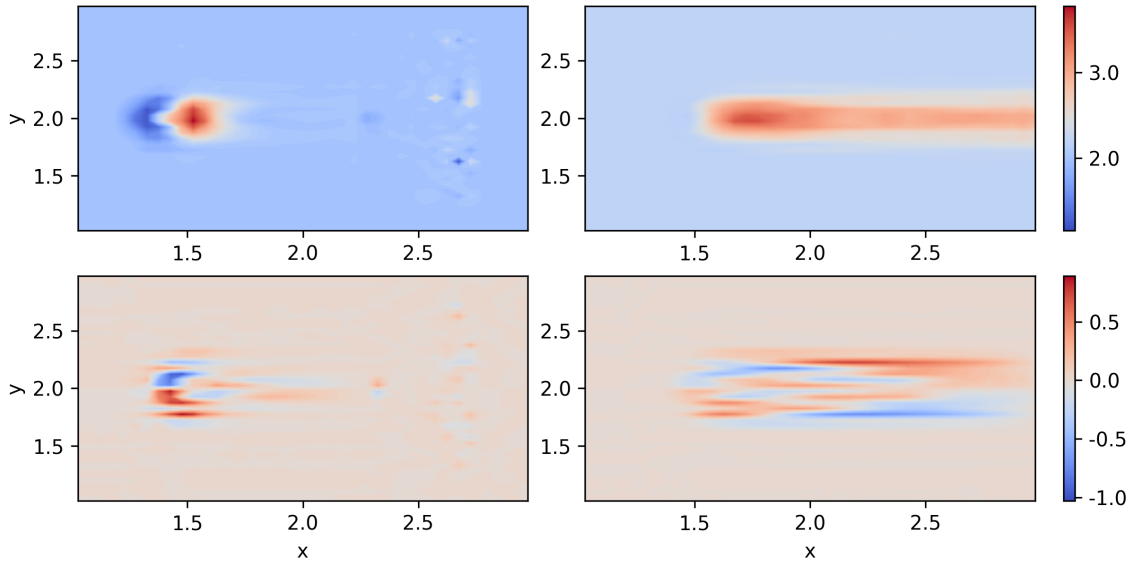


Figure 6.18: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) of problem WTD with 5% noise and 4 optimisation round.

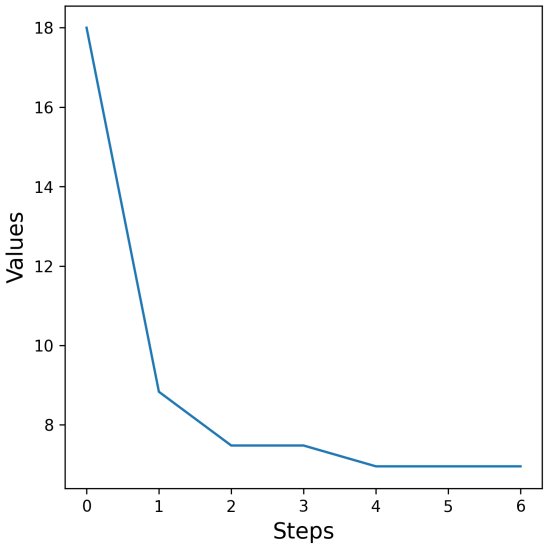
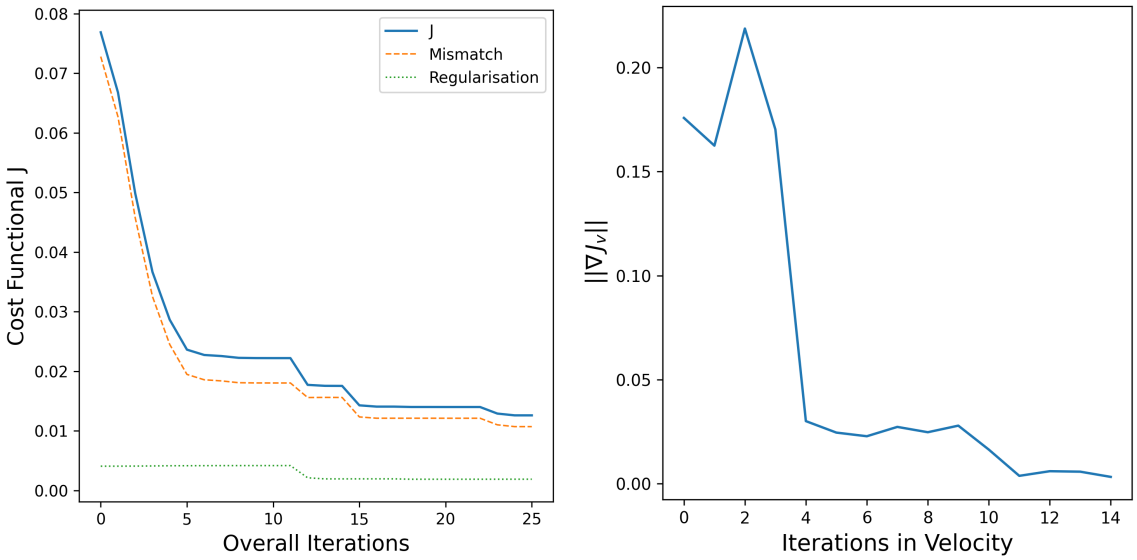


Figure 6.19: Steps in transformation parameter κ for problem WTD with 5% noise and 4 optimisation rounds.



(a) Cost functional

(b) Gradient norm of velocities

Figure 6.20: Indicators of success for the reconstruction of problem WTD with 5% noise and 4 optimisation rounds.

6.4 Artificial Problem with Wide Transformation Domain and 10 % Noise

As a next step, we added not only five percent but around ten percent Gaussian noise, with $\mu = 0$ and $\sigma = 0.3$. This results in concentrations with significant disturbances, as shown in Figure 6.21. Here, the bolus shape of the concentrations is fading. We can still see the movement, including the spreading out and slimming down, of the tracer but the edges are blurred and not visible anymore, which increases the complexity of the reconstruction problem.

The reconstruction is started with the same starting parameters and regularisation parameters as before. Only the tolerance for the decrease in the cost functional is adapted from $\text{tol}_2 = 1e^{-7}$ to $\text{tol}_2 = 1e^{-6}$. That the reconstruction struggles to match the data is visible in the reconstructed concentrations, see Figure 6.22. At best, there is still a spreading out and slimming down visible, but the errors are quite high and the resemblance to the data is rather faint.

That the reconstruction is not as good as the previous becomes also visible in

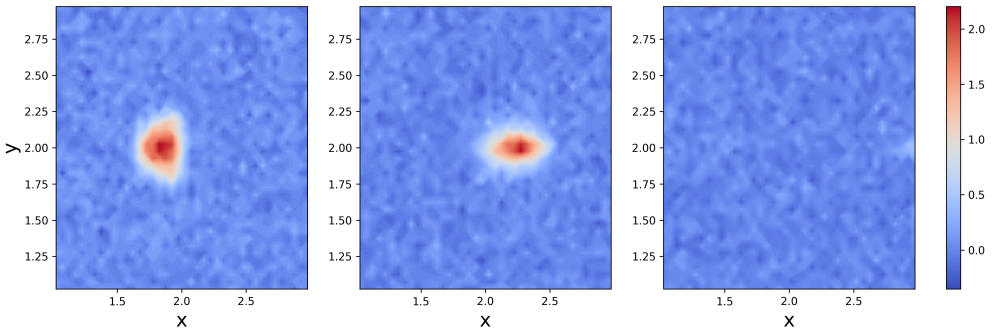


Figure 6.21: Concentrations of problem WTD at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ with 10% noise.

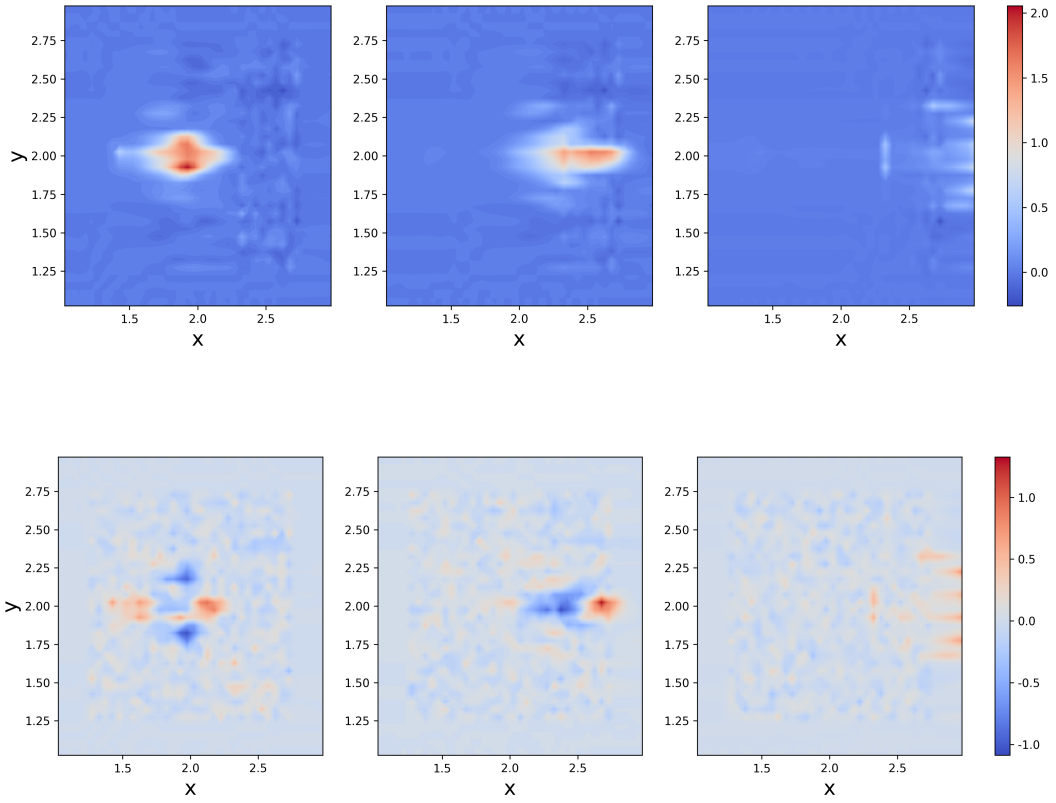
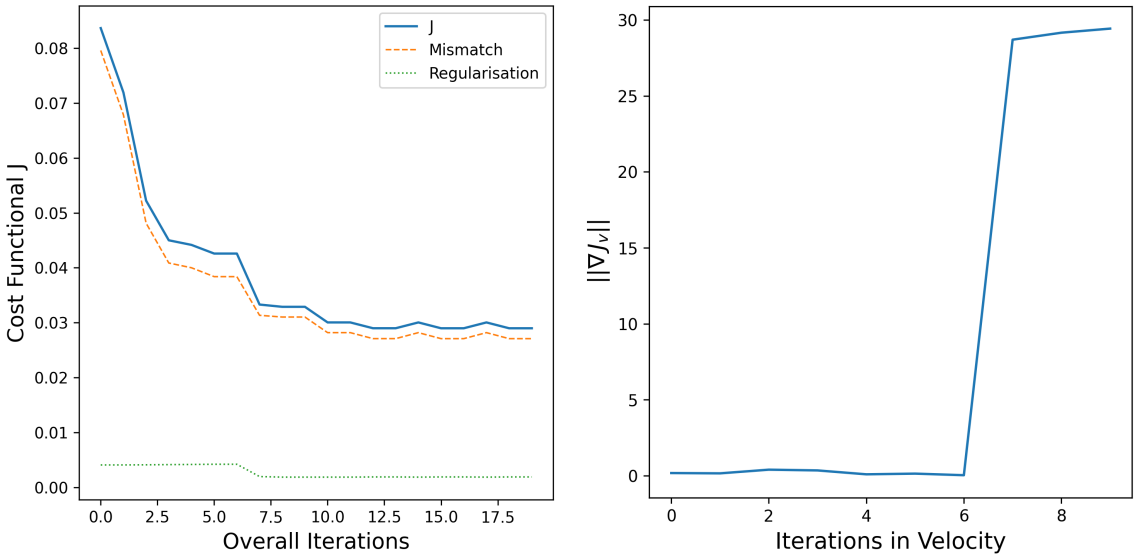


Figure 6.22: Reconstructed concentrations (*top*) and mismatch between reconstructed concentrations and artificial data (*bottom*) of problem WTD with 10% noise at time steps $t_1 = 0.3$, $t_2 = 0.45$ and $t_3 = 0.8$ (from *left* to *right*) for 4 optimisation rounds.

the cost functional values which decrease rather wobbly towards the final value $J = 0.0289$, see Figure 6.23a, this is more than double compared to the problem with only five percent noise. Furthermore, there are overall less steps taken in this reconstruction, with #steps = 19, compared to #steps = 25 for the reconstruction with five percent noise and #steps = 30 for the reconstruction without noise. Lastly, we can see that the gradient norm in the velocity, Figure 6.23b, is not near zero but instead increased significantly before terminating. This is a good indicator for a failed optimisation, showing that we are not at a stationary point.



(a) Cost functional

(b) Gradient norm of velocities

Figure 6.23: Indicators of success for the reconstruction of problem WTD with 10% noise and 4 optimisation rounds.

Nevertheless, the reconstructed velocities (Figure 6.24) look still quite similar to the velocities calculated for five percent and no noise. This is surprising considering the resulting concentrations and that the gradient norm of the velocities indicate a failed reconstruction. Furthermore, the transformation parameter has been reconstructed almost exact with $\kappa = 6.51$.

To conclude we can say, that the reconstruction is able to deal with some added noise, but has its limits, as seen in the ten percent example. Even though the added noise influences the behaviour of the reconstructed concentration bolus quite fast, the

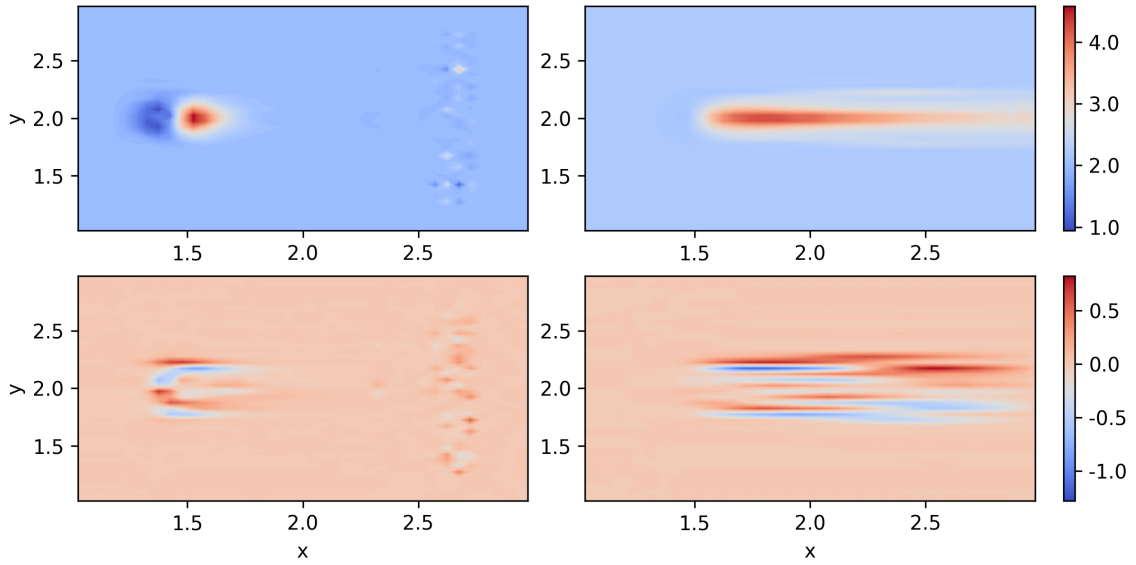


Figure 6.24: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) of problem WTD with 10% noise and 4 optimisation rounds.

reconstructed velocities and especially the reconstructed transformation parameter are still accurate, see Table 6.1. For only moderate noise, the five percent case, we can even see improvements in all indicators except the final cost functional value, which is likely due to mismatches outside the bolus region. Here, the concentrations are zero for the artificial data without noise and thus also for the reconstruction, resulting in a zero mismatch for most of the domain. This is not the case if we add noise. Now the reconstruction would need to correctly reconstruct the noise in the background in order to achieve a low mismatch value. What would make the reconstruction on noisy data easier would be the usage of a noise free starting concentration, because it avoids the transportation of noise from the concentrations at $t = 0$ throughout the simulation. Putting everything together, we should derive reasonable results from ultrasound data as well.

name	$\|\|\nabla J_v\|\|_2$	$\|\|\nabla J_\kappa\|\|_2$	cost functional	error in κ
WTD	0.012	0.00093	0.0082	0.24
WTD 5 %	0.0032	0.0004	0.0126	0.04
WTD 10 %	29.43	0.00017	0.0289	0.49
NTD	0.0003	0.0017	0.004	0.14

Table 6.1: Main indicator of success for reconstructions of problems WTD and NTD and for reconstructions with noisy data for problem WTD.

Chapter 7

Reconstruction of Ultrasound Data from a Mouse Tumour

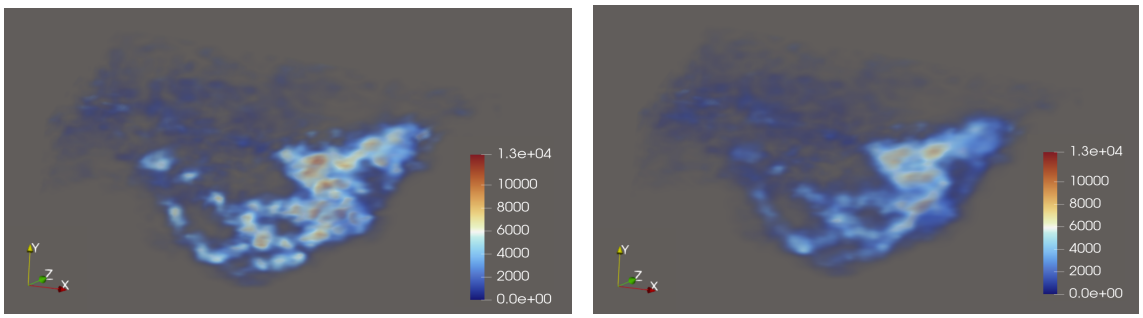
As a final step, the parameter identification was applied to actual ultrasound data, showing a mouse tumour using contrast agent. In order to use the programmed optimisation the data had to be preprocessed, which is explained in the following Section 7.1. In Section 7.2.1 we present the reconstructed results using a restricted transformation parameter, as before, and in Section 7.2.2 results derived using an unrestricted transformation parameter.

7.1 Data Processing

In Figure 7.1 the 3D ultrasound data of a mouse tumour at $t = 12$ is shown. The data was derived from a medical team at the Translational Ultrasound Lab (University of California, San Diego) and Stanford Medicine (Stanford University). It consists of a grid with 86, 44 and 77 cells in x-, y- and z-direction, respectively. The cells are equidistant with a length of 0.3 millimeters (mm). This results in a space domain Ω of $25.8 \times 13.2 \times 23.1$ mm^3. The time was measured in seconds (sec), and contains 50 measurements in 26.789 sec, which results in a frame rate of roughly one picture every 0.54 sec.

The whole dataset was smoothed using a Savitzky-Golay finite impulse response (FIR) smoothing filter [62]. The purpose of this filter is to reduce the noise on a dataset. This is achieved through a process known as convolution, whereby succes-

sive subsets of adjacent data points are fitted with a low-degree polynomial using the method of linear least squares. In general, it smoothes the data and increases its precision without distorting the signal tendency [62]. The difference before and after smoothing of a time step is shown in Figure 7.1.



(a) Original 3D ultrasound picture (b) Smoothened 3D ultrasound picture

Figure 7.1: Visualisation of 3D ultrasound data derived at time $t = 12$.

The next step during the data processing is to project the 3D data onto a 2D domain so that the reconstruction algorithm can be applied. This was done by summing the discrete concentrations C over the z-axis and thus results in a 2D domain with x- and y-axis

$$C_{2D}(x, y) = \sum_{i \in [0,77]} C(x, y, i). \tag{7.1.1}$$

We chose to project onto the x-y-plane because it contains more information about the concentration flow, compared to the other two planes. Here we can see the inflow, the transportation through the organ and the outflow of the tracer. In the other two planes the inflow happens in the middle of the domain and thus we only get little information about the transportation of contrast. Therefore, picking the plane where we get the most information about the tracer movement is reasonable and should result in the best possible reconstruction for the perfusion parameters, which include the velocities and the transformation parameter. The resulting 2D concentrations are shown in Figure 7.2.

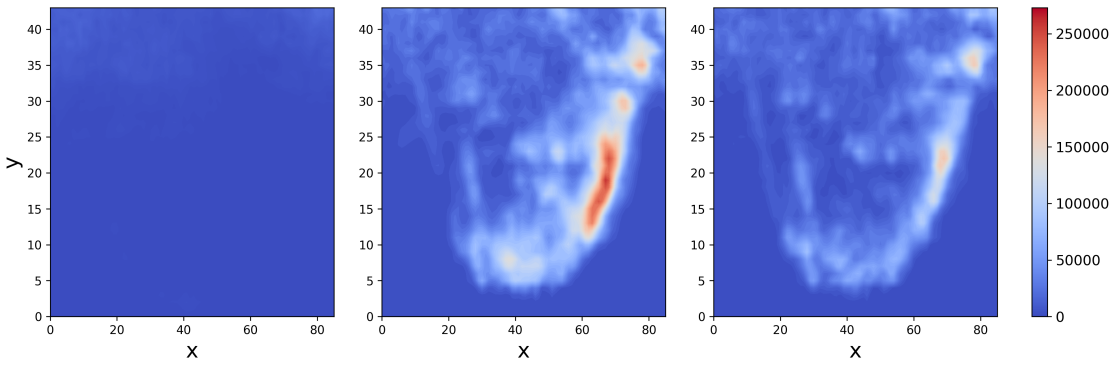


Figure 7.2: 2D projection of the 3D ultrasound data derived by summing up all concentrations over the z-axis, at time steps $t_1 = 0$, $t_2 = 10.7$ and $t_3 = 21.4$.

Because the measured concentrations are quite large, around $1.5e^5$. We rescaled the concentrations to a maximum of 5, which is comparable to the values we tested the reconstruction on

$$\overline{C}(x, y) = \frac{5\, C_{2D}}{\max_{x,y\in\Omega} C_{2D}}. \tag{7.1.2}$$

This does not change the behaviour and only rescales the concentrations, see Figure 7.3.

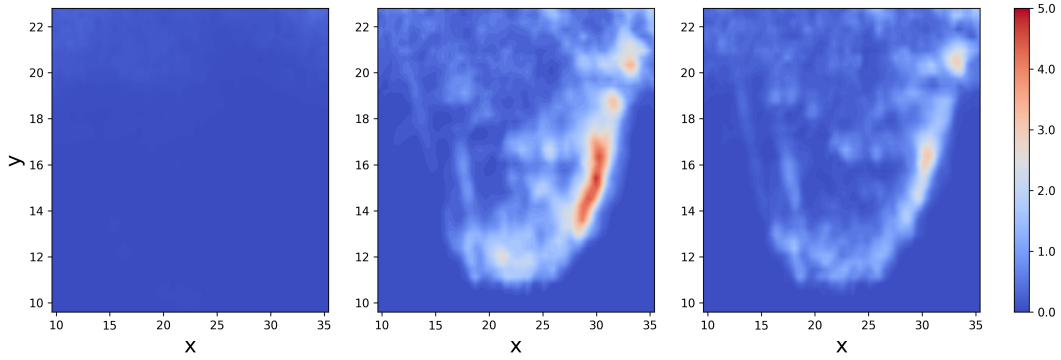


Figure 7.3: 2D projection scaled to concentrations $\overline{C}$ that are easier to handle by the reconstruction algorithm, with $\max_{\overline{C}} = 5$ and again at time steps $t_1 = 0$, $t_2 = 10.7$ and $t_3 = 21.4$.

Another adaption made for the reconstruction algorithm is a zero padding of 16 cells that was fitted around the domain. This ensures that we have no concentrations

near the boundaries in the beginning and thus stabilises the forward and backward solver. Furthermore, it ensures that we can use outflow boundaries, as before, even though there is an inflow on some boundaries. Adding this padding increases the domain to $\Omega = 35.4 \times 22.8\ mm^2$ and the number of cells to 118 and 76 in x- and y-direction, respectively. The resulting plot can be seen in Figure 7.4.

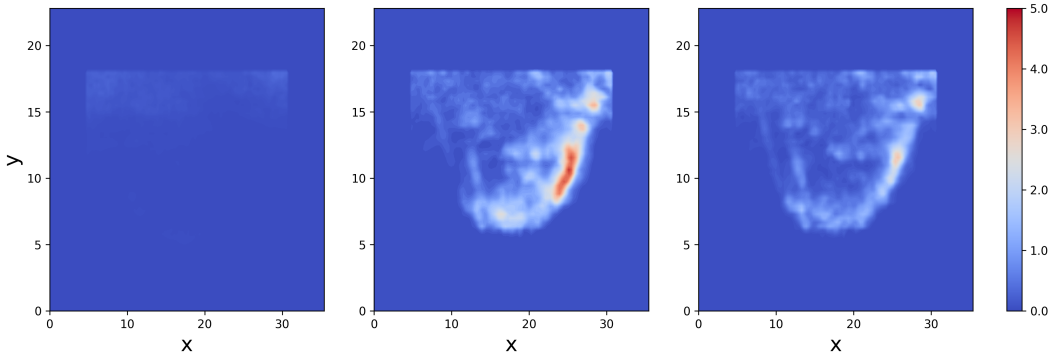


Figure 7.4: A zero padding was added to the data to simplify the conditions at the boundaries for the reconstruction and enable us to use outflow boundary conditions on the whole domain.

As explained in the previous chapter, we cannot model an inflow using the parameter reconstruction algorithm. Thus, we must restrict the time steps to an interval where the inflow of concentration is minimal or zero. In Figure 7.5 we can see the overall concentration plotted over the time steps. It is visible that there is an inflow in the first half and an outflow in the second half. One choice of time interval would be to choose the concentration maximum and then only the time steps afterwards, but this results in only half the time steps and therefore, little information about the concentration movement.

That is why we chose to start at time step $t_s = 13$, which equals $t = 6.96\,sec$, and end at time step $t_s = 46$, which equals $t = 24.65\,sec$. In this interval most of the concentration is already in the domain and we can still see a large part of the measured concentration movement. This implies that the reconstruction, which cannot model inflow and thus increase in concentrations, will not be able to reconstruct the same concentrations as the measured data. But because the difference in overall concentration is at most 22%, we expect the reconstruction to still give an accurate

representation. Furthermore, for the algorithm to be stable we need more than the resulting 33 time steps. We generate more by linearly interpolating between the existing time steps and thus derive 120 time steps. Putting everything together, we get a new time domain $T = [0, 17.69]$, where one time step equals 0.147 *sec*.

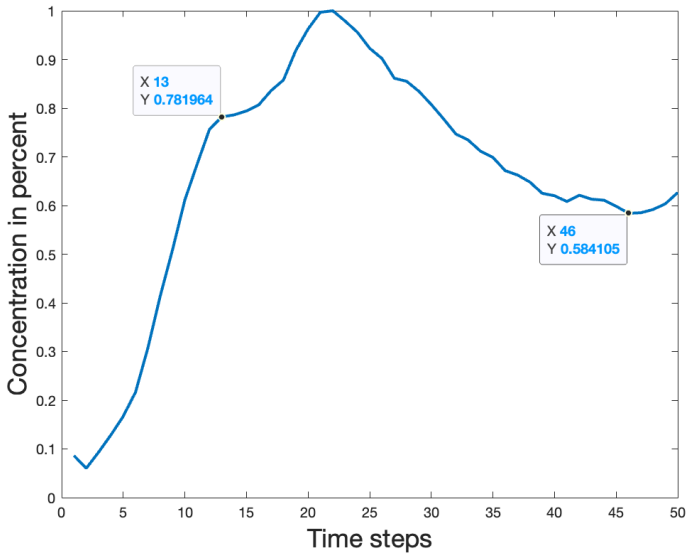


Figure 7.5: Percentage of concentration over time for the whole ultrasound measurement. The plot suggests to try the reconstruction from time step $t_s = 13$ until $t_s = 46$, indicated in the plot.

In a last step the data was mirrored in order to gain a flow from left to right rather than right to left. A flow from right to left would work as well, but the implementation was so far only used on flows from left to right and thus choosing the starting parameters is more intuitive that way. The resulting final concentrations can be found in Figure 7.6.

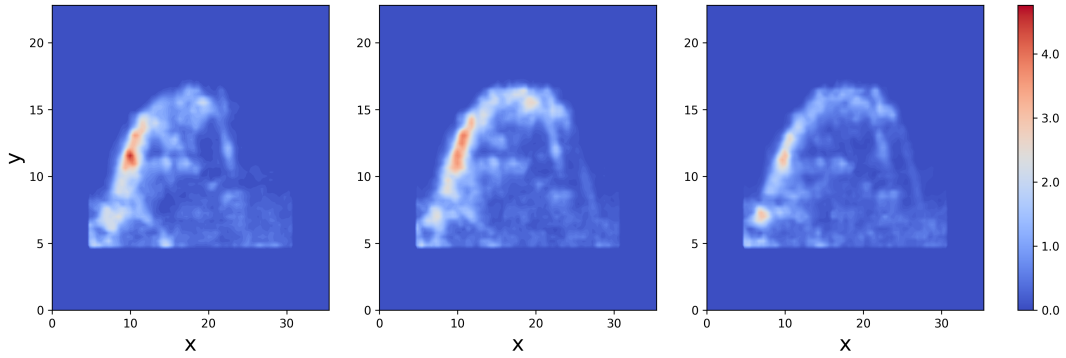


Figure 7.6: Chosen time interval from the original data, which avoids too much inflow of concentration. Furthermore, the data was flipped to allow the concentrations to flow from left to right. The concentrations are given at the new time steps $t_1 = 0$, $t_2 = 8.82$ and $t_3 = 17.69$.

7.2 Reconstructed Results

7.2.1 Using a Restricted Transformation Parameter

After preprocessing the ultrasound data, we can now reconstruct the perfusion parameters, including the velocities and the transformation parameter. For this we have to set some starting parameters and constraints, as before. This is complicated because we previously knew on which domain κ acted and now have to define it ourselves. The transformation domain describes the area of the organ, in which the transformation from arterial to venous concentration happens. Furthermore, we expect to only have arterial blood inside or before the transformation domain, not afterwards. These restrictions help us setting a reasonable domain reaching from $x = 12$ to $x = 24$, see Figure 7.7. It is visible that the largest amount of arterial concentrations has not yet entered the domain and only little is beyond thus we expect reasonable reconstruction results. The constraint for the velocity in x-direction of the arterial concentration is set to zero according to the chosen transformation domain, onwards starting from $x = 23$. This choice ensures that the arterial concentration does not leave the domain of transformation until it has been transformed.

We started the reconstruction with the following starting parameters, $v_x^1 = 0.1$, $v_y^1 = -0.1$, $v_x^2 = 0.1$, $v_y^2 = 0.1$ and $\kappa = 0.68$. Furthermore, we used the stan-

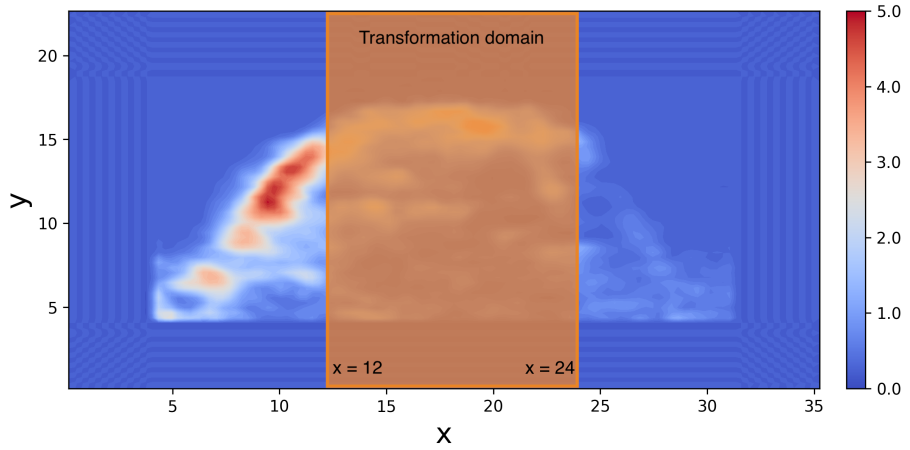


Figure 7.7: Chosen domain restriction for transformation parameter κ for the reconstruction of ultrasound data.

dard regularisation parameters $\lambda = [1e^{-4}, 1e^{-4}, 1e^{-5}]$ and tolerance for the stopping criterion, which means that the gradient norm and the improvement in the cost functional must be both larger than $\text{tol}_{1,2} = 1e^{-5}$. In comparison to the previous reconstructions we had to change the descent direction method. Both, the BFGS and the CG method only found minor improvements in the reconstruction. Therefore, we used the steepest descent method, which resulted in a decent reconstruction.

To evaluate the results, we start by doing a first visual check of the reconstructed concentration flow, see Figure 7.8. It is visible, that even though the mismatch is quite high the main flow of the tracer agent is reconstructed. The high errors can be explained by the mismatch in overall concentration, because we have up to 22% less concentration in the reconstructed problem. Thus, resulting in an inevitable error between reconstructed concentration and ultrasound data.

Next, we will have a look at the indicators of success for the optimisation. In Figure 7.9, we can see that the cost functional is sufficiently reduced, from $J_{\text{start}} = 13.32$ to $J_{\text{final}} = 9.39$, indicating a successful optimisation. Furthermore, the gradient norm is reduced as well, from $||\nabla J_v||_2^{\text{start}} = 19.06$ to $||\nabla J_v||_2^{\text{final}} = 0.87$. Even though, the final value is compared to the previous reconstructions 100 times higher, the reduction is massive and we can expect to be at or near a stationary point.

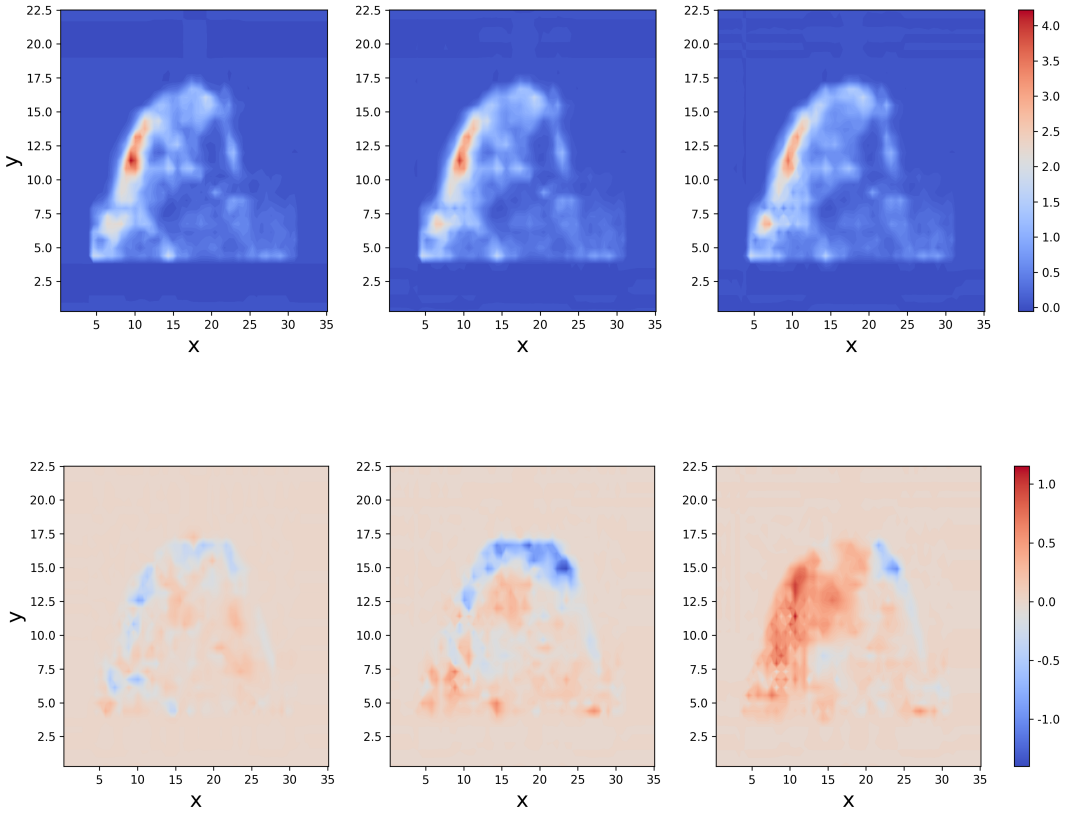
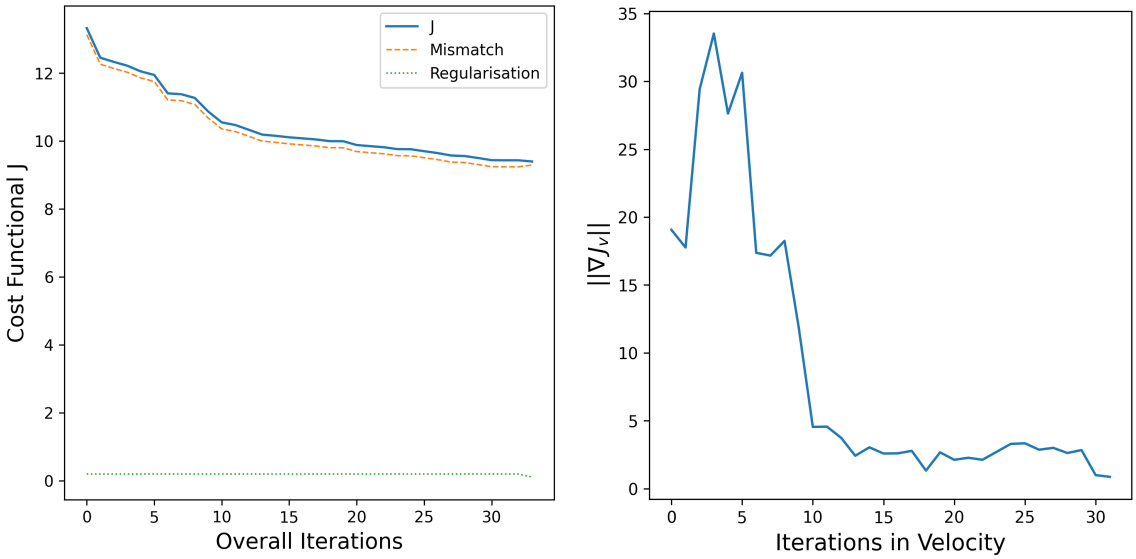


Figure 7.8: Reconstructed concentrations (*top*) and mismatch between the reconstructed concentrations and ultrasound data (*bottom*) at time steps $t_1 = 1.47$, $t_2 = 7.06$ and $t_3 = 14.11$ (from *left* to *right*) using the steepest descent method.

This brings us to the reconstructed parameters, the transformation parameter is adapted from $\kappa = 0.67$ to $\kappa = 0.49$ and the reconstructed velocities are shown in Figure 7.10. As expected, we can see a higher activity of the velocity in the arterial component at the beginning and a lower activity towards the end of the domain. The opposite holds for the velocity in the venous component. Here, we can only observe activity in the transformation domain. This is due to the fact that we have no concentration of the venous component at the beginning and slow velocities in x-direction. These result in a stationary behaviour of the venous component. Furthermore, the average velocity values are quite low, which is not unreasonable for capillaries, where the velocities are low.



(a) Cost functional

(b) Gradient norm of velocities

Figure 7.9: Indicators of success for the reconstruction of the ultrasound data.

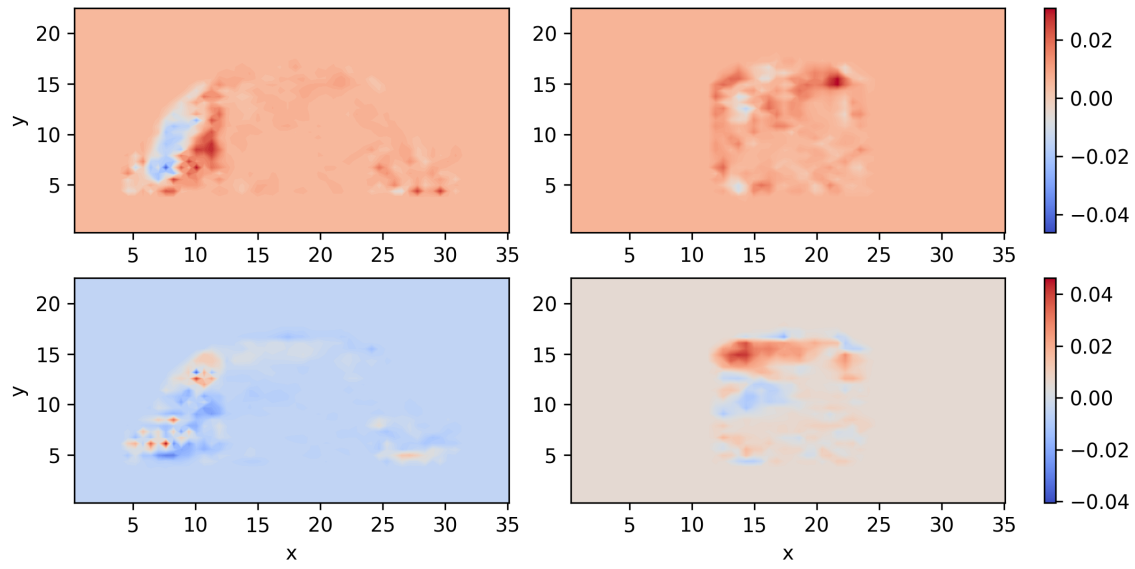


Figure 7.10: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) for the ultrasound data.

Overall we can say, that the reconstruction seems to generate reasonable results for the ultrasound data. However, one uncertainty is given by setting the transformation domain. It would be reasonable to assume that changing the domain would result in different reconstructed perfusion parameters. Thus, in the following we will allow the transformation parameter to be unconstrained and space-dependent.

7.2.2 Using an Unrestricted Transformation Parameter

Letting the transformation parameter be unrestricted and space-dependent should result in a parameter that acts on the domain needed while not being restricted to a constant value. We start the reconstruction with the same parameters as before, leaving out the restriction for the transformation domain. However, we still restrict velocity v_x^1 to zero from $x = 23$ onwards, but this is a soft constraint because there is no significant movement afterwards.

Compared to the previous reconstruction, the concentrations are slightly more fitting, see Figure 7.11. Furthermore, the indicators of success for the optimisation improved. We not only got a lower final cost functional value $J_{\mathrm{f},\,\kappa} = 8.29$, compared to $J_{\mathrm{final}} = 9.39$, but also a lower final gradient norm in the velocity $||\nabla J_v||_2^{\mathrm{f},\,\kappa} = 0.204$, see Figure 7.12. We can further see, that the reconstruction did more steps, 187 compared to 34. This is due to the fact that the reconstruction with the unconstrained and space-dependent transformation parameter needed four rounds of optimisation, while the other only needed one. So far we observed, that more steps relate to a better result, adding the improved cost functional and gradient norm value we expect the optimisation to have found a stationary point with good reconstruction results.

The reconstructed velocities shown in Figure 7.13 present with higher average values compared to the previous velocities, which now match the average velocity in capillaries, $0.3mm/sec$ [30]. Furthermore, we can see that the velocities for the venous component are not restricted to the centre domain anymore. This is the case, because the transformation starts earlier, see Figure 7.14. The transformation parameter is adapted reasonably, with higher values towards the centre of the organ and lower values at the beginning.

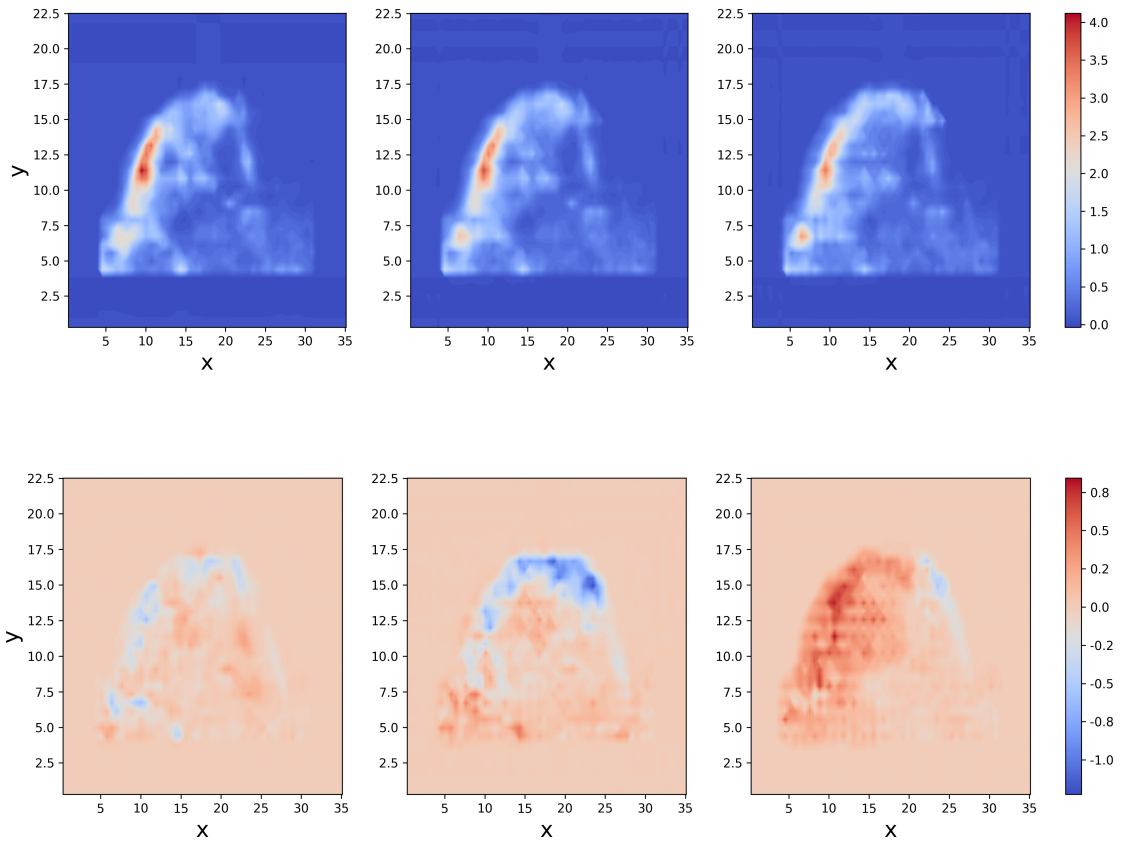
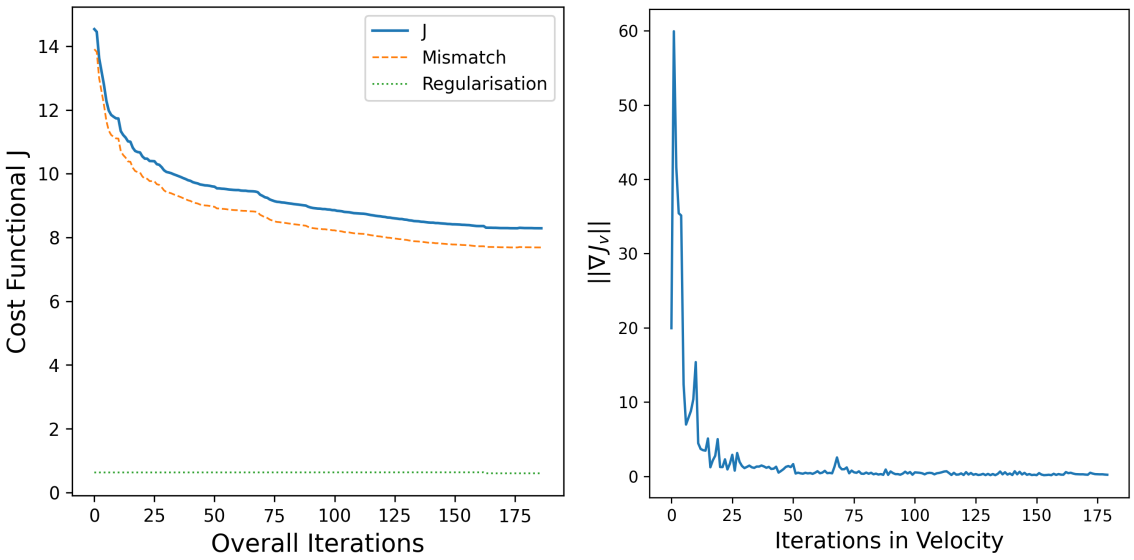


Figure 7.11: Reconstructed concentrations (*top*) and mismatch of reconstructed concentrations and ultrasound data (*bottom*) at time steps $t_1 = 1.47$, $t_2 = 7.06$ and $t_3 = 14.11$ (from *left* to *right*) for an unrestricted and space-dependent transformation parameter.

To conclude, we can say that the reconstructions derived from ultrasound data look promising and deliver reasonable velocity fields and transformation parameter, if we allow a unrestricted transformation parameters. In order, to further benchmark the reconstructed values, we would need more ultrasound data with knowledge about the corresponding perfusions, to check if the reconstructed parameters fit the medical parameters.



(a) Cost functional

(b) Gradient norm of velocities

Figure 7.12: Indicators of success for the reconstruction of the ultrasound data with space-dependent transformation parameter after 4 optimisation rounds.

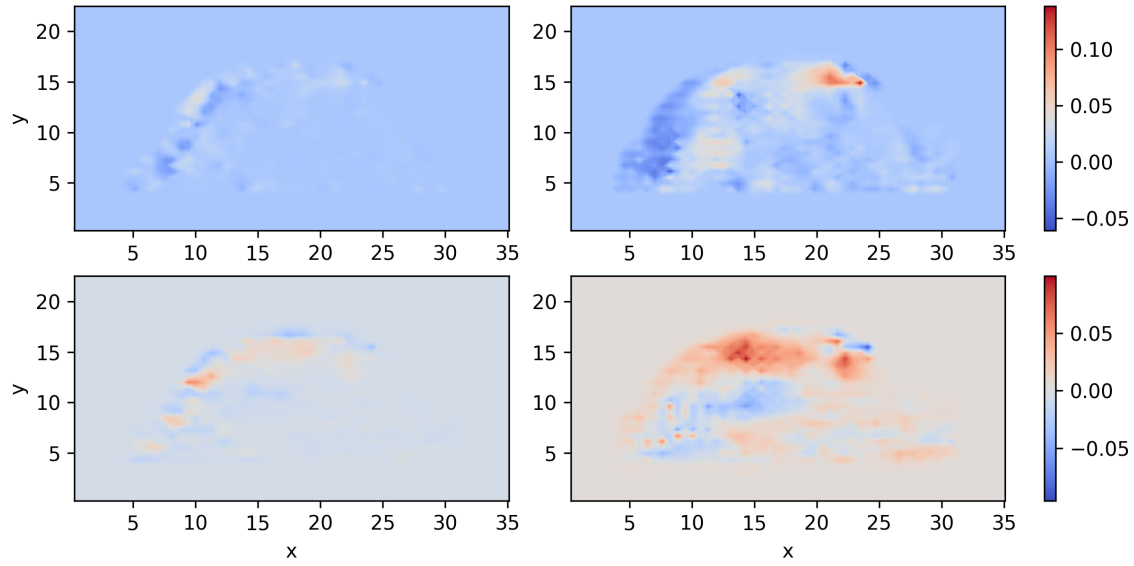


Figure 7.13: Reconstructed velocities (*left* to *right* and *up* to *down*: $v_x^1, v_x^2, v_y^1, v_y^2$) for the ultrasound data with space-dependent transformation parameter.

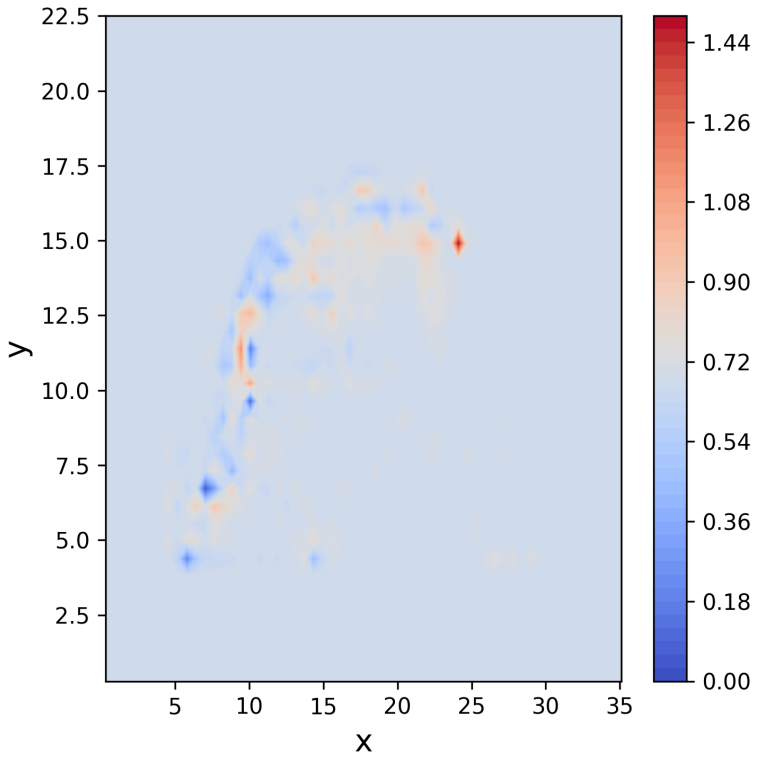


Figure 7.14: Reconstructed space-dependent transformation parameter κ for the ultrasound data.

Chapter 8

Summary and Outlook

Results. This thesis investigates parameter identification for a two-compartment model for tracer transport in blood, and compares it to parameter identification for an advection-diffusion model, with regard to medical usage. A *first-optimise-then-discretise* approach is used for both resulting inverse problems, and thus the adjoint and gradient equations are derived analytically in Section 2.3.2 and 2.4.2. These are then used to implement an inverse solver in *Python* and after thoroughly testing the code (Section 4), we successfully reconstructed the velocities and transformation parameter for the two-compartment model as well as the velocities and diffusion parameter for the advection-diffusion model, from given concentrations over time. The software is able to reconstruct space-dependent parameters.

The focus of this thesis is the two-compartment model, including its possibilities and limitations. The model was chosen because it offers a parameter which directly relates to the medical perfusion, and thus yields important information about tumour perfusion. However, reconstructing the transformation parameter is not straightforward, because its influence on the concentration flow is lower than the influence of the velocities. Thus, we cannot reconstruct all parameter at once but have to optimise the velocities and the conversion function alternating, until convergence is reached, see Algorithm 2.

When the two models were compared, it became apparent that the two-compartment model benefits from having two velocity fields, one velocity field for the arterial concentration and one for the venous concentration. These are, by the choice of

the model, independent of each other and therefore introduce additional degrees of freedom, compared to the advection-diffusion model. These additional degrees of freedom enable reconstructions of a multitude of different concentration flows. This includes a good reconstruction of a diffusive concentration flow (see Figure 5.11). Therefore, while the two-compartment model can never mathematically imitate diffusion, it is possible to reconstruct parameters that result in similar concentration flows. Furthermore, the velocities appear to depend linearly on the diffusion parameter, offering the possibility of indirectly reconstructing the diffusion parameter (Section 5.5). This is not the case for the advection-diffusion model: even though it can also reconstruct parameters resulting in concentration flows similar to the artificial data calculated using the two-compartment model, it cannot reconstruct the conversion function. Together with the fact that the diffusion parameter does not relate to the medical perfusion parameter [37], we can conclude that the two-compartment model is the better choice for reconstructing perfusion parameters.

As mentioned, the two-compartment model splits concentrations into arterial and venous parts. This makes it possible to define a conversion function that transforms arterial concentration into venous concentration and relates it to the perfusion of the area. Conversely, we cannot measure arterial and venous concentrations individually, only the overall concentration throughout the domain. This means that the inverse solver only has information about the added concentrations. This can result in a reconstruction containing only arterial concentrations and no transformation into venous concentrations. Thus, the parameter identification solver requires the following constraints: the domain in which the transformation parameter is active and the setting of the arterial concentration velocities to zero at the end of this domain. These constraints eliminate the multitude of possible solutions and ensure that the conversion from arterial to venous blood occurs in the right place. This information is known for artificial data and can simply be given to the algorithm. However, this is not the case for actual measurement data. Thus, two options were tested. Firstly, a domain was guessed based on the evolution of the concentrations over time, see Section 7.2. Secondly, no domain was set thus assuming that the conversion is happening across the entire domain, see Section 7.2.2. While both approaches are reasonable, the latter is recommended for further research because an educated guess still introduces uncertainty, whereas assuming that the perfusion is happening over the whole domain is reasonable if we have data from inside the organ.

Limitations and Outlook. This thesis is an extensive feasibility study of applying the two-compartment model to ultrasound data to estimate tumour perfusion. However, additional research is required before this approach can be applied to patient treatment. The first adaptation required is to enable inflow over the boundaries during the reconstruction. This would allow us to use the entire measurement for the reconstruction thus avoiding the restriction to a relatively small number of time steps during which the contrast agent only leaves the domain and does not enter it. This can be achieved by using the ghost cells, that have already been implemented, to ensure Dirichlet boundary conditions at the inflow boundaries. Furthermore, in order to derive Dirichlet boundary conditions at the intermediate steps of the time solver, the data would need to be interpolated accordingly. One limitation of this approach is that it is not possible to allow both inflow and outflow over the same boundary. Therefore, it only works if those boundaries are clearly separated, which is not always the case.

Another difficulty is to distinguish between arterial and venous concentrations. So far we made the assumption that there is only arterial concentration at the beginning and venous at the end. In order to mathematically solve the reconstruction problem we need to set the boundary conditions for the arterial and venous concentrations. However, this separation is not possible with regard to the medical application. We will not be able to separate the observed concentration inflow into arterial or venous concentrations. An easy fix would be to assume that all inflow is arterial and exclude possible venous inflow. This could, combined combined with the space-dependent conversion function, deliver good results, but the assumption should be verified.

Another aspect on the way to medical application is to ensure that the reconstructed conversion function indeed relates to the perfusion parameter. To prove this, significantly more data is needed. The given ultrasound data consists of 3D data over time, therefore, instead of projecting onto a 2D plane, the solver could be extended to a third dimension. However, adding another dimension would increase computational time again and it is unclear whether reconstructions would be better, given that the concentration movement in our example mostly happens in one plane. To evaluate how important the third dimension is for the reconstruction it would be

interesting to compare the results in 3D with those in 2D. Another aspect is that 2D ultrasounds are more common thus a software that works reliable on 2D data would result in a more accessible treatment option. However, the concentrations derived from 2D ultrasounds contain not only an inflow over the boundaries but also an inflow inside the domain from the not measured third dimension. This problem does not occur when projecting 3D data onto the 2D plane thus it needs to be accounted for by a different approach. For example, by introducing a source and sink term as added control parameters into the parameter identification.

Furthermore, a faster code is needed for medical applications. Currently, reconstruction takes between ten and twenty hours. This should be reduced to a maximum of one or two hours in order to be useful in tumour treatment. Improving the code to reduce calculation time should be feasible. So far, the focus has not been on efficient programming, even simple approaches, such as using built-in solvers, should reduce computational time. Additionally, sophisticated approaches. such as parallel computing, should speed up the computations massively.

Finally, it would be mathematically interesting to prove that the problem is analytic solvable and thus a solution exists. A rough sketch of the needed steps are shown in Section 2.4.1.

Bibliography

[1] Ananthakrishnan, A., Gogineni, V., and Saeian, K. Epidemiology of primary and secondary liver cancers. *Seminars in Interventional Radiology 23* (2006), 47–63.

[2] Anderson, D. H. *Compartmental Modeling and Tracer Kinetics.* Springer, 1983.

[3] Anton, H. *Calculus with Analytic Geometry.* Wiley, 1998.

[4] Armijo, L. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics 16*, 1 (1966), 1–3.

[5] Ascher, U. M., Ruuth, S. J., and Wetton, B. T. R. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis 32*, 3 (1995), 797–823.

[6] Aster, R. C., Borchers, B., and Thurber, C. H. *Parameter Estimation and Inverse Problems.* Elsevier, 2013.

[7] Bagirov, A. M., Gaudioso, M., Karmitsa, N., Mäkelä, M. M., and Taheri, S. *Numerical Nonsmooth Optimization.* Springer, 2020.

[8] Bonicatto, P., Ciampa, G., and Crippa, G. Advection-diffusion equation with rough coefficients: weak solutions and vanishing viscosity. *Journal de Mathématiques Pures et Appliquées 167* (2022), 204–224.

[9] Bonicatto, P., Ciampa, G., and Crippa, G. Weak and parabolic solutions of advection–diffusion equations with rough velocity field. *Journal of Evolution Equations 24*, 1 (2023).

[10] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2009.

[11] Broyden, C. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics 6*, 1 (1970), 76–90.

[12] Coles, C. Parameter estimation in the 1-d transport equation with advection. *PERGAMON Computers and Mathematics with Applications*, 44 (2002), 1493–1502.

[13] Colombi, R., Rohde, N., Schlüter, M., and von Kameke, A. Coexistence of inverse and direct energy cascades in faraday waves. *Fluids 7* (2022), 148.

[14] Colombi, R., Schlüter, M., and von Kameke, A. Three dimensional flows beneath a thin layer of 2d turbulence induced by faraday waves. *Experiments in Fluids 8* (2021), 62.

[15] Crouzeix, M. Une méthode multipas implicite-explicite pour l'approximation des équations d'évolution paraboliques. *Numerische Mathematik 35* (1980), 257–276.

[16] Cui, Y., Zhang, C., Li, X., Liu, H., Yin, B., Xu, T., Zhang, Y., and Wang, D. Intravoxel incoherent motion diffusion-weighted magnetic resonance imaging for monitoring the early response to zd6474 from nasopharyngeal carcinoma in nude mouse. *Scientific Report 5*, 16389 (2015).

[17] Dolezal, A., and Wong, S. Relativistic hydrodynamics and essentially non-oscillatory shock capturing schemes. *Journal of Computational Physics v120* (1995), 266–277.

[18] Dou, Y.-X., and Han, B. Reconstruction of a velocity field for a 3d advection diffusion equation. *International Journal of Thermal Sciences 50* (2011), 2340–2354.

[19] Draper, N. R., and Smith, H. *Applied Regression Analysis*. John Wiley and Sons, Inc., 1998.

[20] Dullemond, C. P., and Springel, V. Lecture notes in numerical fluid dynamics, 2011.

[21] Evans, L. C. *Partial Differential Equations.* American Mathematical Society, 2010.

[22] Externbrink, S. Two-component model for tracer simulation. Master's thesis, Hamburg University, 2022.

[23] Fatemi, E., Jerome, J., and Osher, S. Solution of the hydrodynamic device model using high order non-oscillatory shock capturing algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems v10* (1991), 232–244.

[24] Feathers, M., Martin, R. C., and Osherove, R. *The Art of Unit Testing.* Mitp Verlags GmbH and Co. KG, 2015.

[25] Fletcher, R. A new approach to variable metric algorithms. *The Computer Journal 13*, 3 (1970), 317–322.

[26] Fornberg, B. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation 51*, 184 (1988), 699–706.

[27] Gardner, D. J., Guerra, J. E., Hamon, F. P., Reynolds, D. R., Ullrich, P. A., and Woodward, C. S. Implicit–explicit (imex) runge–kutta methods for non-hydrostatic atmospheric models. *Geoscientific Model Development 11*, 4 (2018), 1497–1515.

[28] Goldberg, M., and Tadmore, E. Scheme-independent stability criteria for difference approximations of hyperbolic initial-boundary value problems, i. *Mathematics of Computation 32* (1978), 1097–1107.

[29] Goldfarb, D. A family of variable-metric methods derived by variational means. *Mathematics of Computation 24* (1970), 23–23.

[30] Guyton, A., and Hall, J. *Textbook of Medical Physiology.* W.B. Saunders, 1996.

[31] Güttel, S., and Pearson, J. W. A rational deferred correction approach to parabolic optimal control problems. *IMA Journal of Numerical Analysis 38*, 4 (2018), 1861–1892.

[32] Hager, W. W., and Zhang, H. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization 2* (2006), 35–58.

[33] Hernández-Fontes, J., Torres, L., Mendoza, E., and Escudero, M. Identification of the advection-diffusion equation for predicting green water propagation. *Elsevier Ocean Engineering 214* (2020).

[34] Hewitt, E., and Hewitt, R. E. The gibbs-wilbraham phenomenon: An episode in fourier analysis. *Archive for History of Exact Sciences 21* (1979), 129–160.

[35] Hintermüller, P. D. M. Lecture notes in nonlinear optimization.

[36] Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S. *Optimization with PDE Constraints*. Springer, 2009.

[37] Hristov, D., Mustonen, L., von Eyben, R., Götschel, S., Minion, M., and Kaffas, A. E. Dynamic contrast-enhanced ultrasound modeling of an analog to pseudo-diffusivity in intravoxel incoherent motion magnetic resonance imaging. *IEEE Transactions On Medical Imaging 10* (2022).

[38] Huang, Y., Snuderl, M., and Jain, R. K. Polarization of tumor-associated macrophages: A novel strategy for vascular normalization and antitumor immunity. *Cancer Cell 19*, 1 (2011), 1–2.

[39] Hudson, J. M. *Quantification of Blood Flow Using Ultrasound Contrast Agent*. PhD thesis, University of Toronto, 2011.

[40] Hundsdorfer, W., and Verwer, J. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer, 2003.

[41] Iima, M., and Bihan, D. L. Clinical intravoxel incoherent motion and diffusion mr imaging: Past, present, and future. *Radiology 278*, 1 (2016), 13–32.

[42] Isakov, V. *Inverse Problem for Partial Differential Equations*. Springer, 2017.

[43] Jarnagin, W. R. *Blumgart's Surgery of the Liver, Biliary Tract, and Pancreas*. Elsevier, 2017.

[44] Jiang, G.-S., and Shu, C.-W. Efficient implementation of weighted eno schemes. *Journal of Computational Physics 126* (1996), 202–228.

[45] Johnson, S. G. Lecture notes in introduction to numerical methods, 4 2019.

[46] Jorgensen, P. C., and DeVries, B. *Software Testing: A Craftsman's Approach*. Auerbach Publications, 2022.

[47] Jr., P. R. P. *Probability, Random Variables and Random Signal Principles*. McGraw-Hill, Inc., 2001.

[48] Krekel, H., and Pytest-Dev Team. pytest, 2015.

[49] Lehtosalo, L., and mypy contributors. mypy documentation, 2025.

[50] LeVeque, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 2007.

[51] Lu, X.-D., Osher, S., and Chan, T. Weighted esentially non-oscillatory schemes. *Journal of Computational Physics 115* (1994), 200–212.

[52] Maryshev, B., Cartalade, A., Latrille, C., and Néel, M.-C. Accuracy and efficiency of adjoint state based parameter identification for fractional advection diffusion equation with space-dependent coefficients. *IEEE Xplore* (2016).

[53] Minion, M. L. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences 1*, 3 (2003), 471–500.

[54] Mishra, P. D. S., and Weber, D. F. Lecture notes in topics in mathematical and computational fluid dynamics, 2016.

[55] Mitusch, S. dolfin-adjoint, 2017.

[56] Modena, S., and Jr., L. S. Non-uniqueness for the transport equation with sobolev vector fields. *Annals of PDE 4*, 18 (2018).

[57] Nocedal, J., and Wright, S. J. *Numerical Optimization*. Springer, 1999.

[58] Nurkanovic, A. *Numerical Methods for Optimal Control of Nonsmooth Dynamical Systems*. PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2023.

[59] Raoa, S. R., Shelton, S. E., and Dayton, P. A. The "fingerprint" of cancer extends beyond solid tumor boundaries: Assessment with a novel ultrasound imaging approach. *IEEE Trans Biomedical Engineering 63*, 5 (2015), 1082–1086.

[60] Reitz, K., and Real Python. The hitchhiker's guide to python, 2024.

[61] Saridaki, Z., Asimakopoulou, N., Boukovinas, I., and Souglakos, J. How to identify the right patients for the right treatment in metastatic colorectal cancer (mcrc). *Curr Colorectal Cancer Rep. 11*, 4 (2015), 151–159.

[62] Savitzky, A., and Golay, M. J. E. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry 36*, 8 (1964), 1627–1639.

[63] Shanno, D. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation 24* (1970), 647–656.

[64] Sherman, J., and Morrison, W. J. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics 20* (1949), 620–624.

[65] Shu, C., and Cai, W. Uniform high-order spectral methods for one- and two-dimensional euler equations. *Journal of Computational Physics v104* (1993), 427–443.

[66] Shu, C., Zang, T., Erlebacher, G., Whitaker, D., and Osher, S. High order eno schemes applied to two- and three- dimensional compressible flow. *Applied Numerical Mathematics v9* (1992), 45–71.

[67] Shu, C.-W. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. *ICASE 1*, 97–65 (1997).

[68] Shu, C.-W. Essentially non-oscillatory and weighted essentially non-oscillatory schemes. *Cambridge University Press* (2020), 1–63.

[69] Shu, C.-W., and Osher, S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics 77* (1988), 439–471.

[70] Shu, C.-W., and Osher, S. Efficient implementation of essentially non-oscillatory shock-capturing schemes ii. *Journal of Computational Physics 83* (1989), 32–78.

[71] Sourbron, S. A tracer-kinetic field theory for medical imaging. *IEEE Transactions On Medical Imaging 33*, 4 (2014).

[72] Sourbron, S., and Buckley, D. Tracer kinetic modelling in mri: estimating perfusion and capillary permeability. *Physics in Medicine and Biology 57*, 2 (2011), 1–33.

[73] Tan, S. *Boundary conditions and applications of WENO finite difference schemes for hyperbolic problems.* PhD thesis, Applied Mathematics at Brown University, 2012.

[74] Tarantola, A. *Inverse Problem Theory.* Elsevier, 1987.

[75] Tirumani, S. H., Kim, K. W., Nishino, M., Howard, S., Krajewski, K., Jagannathan, J., Cleary, J., Ramaiya, N., and Shinagare, A. Update on the role of imaging in management of metastatic colorectal cancer 1. *Radio Graphics 34*, 1 (2014), 1908–1928.

[76] Tröltzsch, F. *Optimale Steuerung partieller Differentialgleichungen.* Vieweg+Teubner, 2009.

[77] Truong, T. T., To, T. D., Nguyen, H.-T., Nguyen, T. H., Nguyen, H. P., and Helmy, M. A fast and simple modification of newton's method avoiding saddle points. *Journal of Optimization Theory and Applications 199* (2023), 805–830.

[78] Varah, J. Stability restrictions on second order, three level finite difference schemes for parabolic equations. *SIAM Journal on Numerical Analysis 17*, 2 (1980), 300–309.

[79] Wildeboer, R. R., van Sloun, R. J. G., Schalk, S. G., Mannaerts, C. K., van der Linden, J. C., Huang, P., Wijkstra, H., and Mischi, M. Convective-dispersion modeling in 3d contrast-ultrasound imaging for the localization of prostate cancer. *IEEE Transactions on Medical Imaging 37*, 12 (2018), 2593–2602.

[80] Wolfe, P. Convergence conditions for ascent methods. *SIAM Review 11*, 2 (1969), 226–235.

[81] You, J., Zhang, Z., Zhang, F., and Wu, W. Cooperative filtering and parameter identification for advection–diffusion processes using a mobile sensor network. *IEEE Transactions on Control Systems Technology 31*, 2 (2023), 527–542.

[82] Zentrum für Krebsregisterdaten and Gesellschaft der epidemologischen Krebsregister in Deutschland e.V. Krebs in Deutschland für 2019/2020, 2023.

[83] Zhang, P., Wong, S., and Shu, C.-W. A weighted essentially non-oscillatory numerical scheme for a multi-class traffic flow model on an inhomogeneous highway. *Journal of Computational Physics 2*, 212 (2006), 739–756.