DEGREE PROJECT IN THE FIELD OF TECHNOLOGY
ENGINEERING PHYSICS
AND THE MAIN FIELD OF STUDY
COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Grey-box modelling of distributed parameter systems

PATRIK BARKMAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Grey-box modelling of distributed parameter systems

PATRIK BARKMAN

Master in Machine Learning
Date: November 24, 2018
Supervisor: Johan Hoffman
Examiner: Tino Weinkauf
School of Electrical Engineering and Computer Science
Swedish title: Hybridmodellering av distribuerade parametersystem

Abstract

Grey-box models are constructed by combining model components that are derived from first principles with components that are identified empirically from data. In this thesis a grey-box modelling method for describing distributed parameter systems is presented. The method combines partial differential equations with a multi-layer perceptron network in order to incorporate prior knowledge about the system while identifying unknown dynamics data. A gradient-based optimization scheme which relies on the reverse mode of automatic differentiation is used to train the network. The method is presented in the context of modelling the dynamics of a chemical reaction in a fluid. Lastly, the grey-box modelling method is evaluated on a one-dimensional and two-dimensional instance of the reaction system. The results indicate that the grey-box model was able to accurately capture the dynamics of the reaction system and identify the underlying reaction.

Sammanfattning

Hybridmodeller konstrueras genom att kombinera modellkomponenter som härleds från grundläggande principer med modelkomponenter som bestäms empiriskt från data. I den här uppsatsen presenteras en metod för att beskriva distribuerade parametersystem genom en hybridmodellering. Metoden kombinerar partiella differentialekvationer med ett neuronnätverk för att inkorporera tidigare känd kunskap om systemet samt identifiera okänd dynamik från data. Neuronnätverket tränas genom en gradientbaserad optimeringsmetod som använder sig av bakåt-läget av automatisk differentiering. För att demonstrera metoden används den för att modellera kemiska reaktioner i en fluid. Metoden appliceras slutligen på ett en-dimensionellt och ett två-dimensionellt exempel av reaktions-systemet. Resultaten indikerar att hybridmodellen lyckades återskapa beteendet hos systemet med god precision samt identifiera den underliggande reaktionen.

Acknowledgement

Firstly, I would like to thank my supervisor Johan Hoffman for his support, enthusiasm and readiness for adhering to my suggestions and ideas. Secondly, I also thank Tino Weinkauf for his guidance in realizing this thesis work. I would also like to thank Simon Funke and the dolfin-adjoint team for their initiative of combining neural networks with partial differential equations, which was the starting point of this thesis. Lastly, I would like to express sincere gratitude to my family for their devoted support, encouragement and comfort in times of struggle.

Contents

Chapter 1

Introduction

Modelling physical systems is central to many scientific disciplines and industrial applications. Models can be used to predict the behaviour of a system, which in turn is crucial for controlling, monitoring and optimizing the system. There are different means of constructing such a model depending on the prior knowledge about the system. If the underlying mechanics of the physical process is well known, the model may be constructed solely from first-principles such as fundamental physical laws. These models are referred to as *white-box* models due to the transparent nature of their construction. If little prior knowledge is available about the system, a data-driven approach may be more viable. By measuring how the system responds to certain input signals, a model can be constructed by mimicking this behaviour. The resulting models are however often difficult to interpret and are consequently referred to as *black-box* models. These approaches can be combined such that the model is partially constructed from first-principles and partially empirically constructed from data. Such models are referred to as *grey-box* models. Grey-box models are of interest in many practical applications such as weather forecasting or modelling flow through porous media. In these applications and almost any modelling task there is to some extent lack of prior knowledge about the system. This makes grey-box modelling a viable approach.

This thesis considers grey-box models that can be seen as a combination of white-box and black-box components. For these models, differential equations and artificial neural networks are commonly used as the white-box and black-box component respectively. Grey-box models of this kind have mainly been used to model biochemical and chemical systems. Such systems are often spatially distributed in their nature due to physical processes such as heat transfer, diffusion or advection. Nonetheless, grey-box models have mainly been ap-

plied to systems under the assumption that these processes can be neglected. In this thesis, a grey-box method for modelling such *distributed parameter systems* is presented.

1.1 Objective, contribution and scope

The objective of this thesis is to present and evaluate a method to model distributed parameter systems with a grey-box model. Specifically this method combines partial differential equations with multi-layer perceptron networks to incorporate prior knowledge of the system but also to recover unknown dynamics from data. The main features of the method are the following:

- Seamlessly and efficiently identifies the parameters in the grey-box model using the reverse mode of automatic differentiation.
- Handles distributed systems with complex geometries by using the finite element method.

In order to demonstrate and evaluate the method a distributed reaction-advection-diffusion system is studied. The method is applied to a one-dimensional and a two-dimensional instance of the reaction system. In addition, the robustness of the method with respect to observability and measurement noise is investigated for the one-dimensional case.

1.2 Thesis overview

The thesis is organized as follows:

- Chapter 2 introduces the theory and previous work in the literature that is of relevance to this thesis.
- Chapter 3 describes the reaction-advection-diffusion system, the grey-box modelling method and the tests that were carried out to evaluate the method.
- Chapter 4 presents the results from the evaluation of the grey-box method.
- Chapter 5 discusses the results and the proposed grey-box method. Lastly, some comments on possible future work is given and the overall outcome of the thesis is summarized.

Chapter 2

Background

This chapter begins by introducing the theory relevant for this thesis. Firstly, the problem of modelling a physical system is defined and the concepts of white-box, black-box and grey-box modelling are introduced. Then artificial neural networks and in particular multi-layer perceptron networks are discussed. This is followed by a section on how partial differential equations are used to model distributed parameter systems and the numerical methods used to solve them. Then the theory of automatic differentiation is presented and the adjoint equation is derived. Lastly, previous work related to the problem and method used in this thesis are presented.

2.1 Modelling Physical Systems

Fundamental to many scientific disciplines is the concept of modelling. It can often be thought of as the task of translating knowledge about some system, phenomena or feature of the world into an abstract representation referred to as a model. This thesis concerns the problem of constructing a model of a physical system. We will consider a system to be an entity that one can interact with by passing it some inputs and observe a corresponding response through its outputs. Most systems are dynamic objects and as such we will be interested in their behaviour in time. The task is then to build a mathematical model which accurately reproduces this dynamic behaviour. Such a model can be used for a wide range of purposes including control, optimization and forecasting where predicting the system behaviour is crucial.

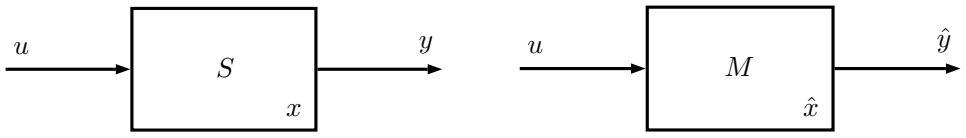
In this section we will first introduce some notation in order to formulate the modelling task as an optimization problem. Secondly, different modelling approaches will be discussed.

2.1.1 The modelling problem

Let S be the system of interest which we would like to model. The system has some internal state $x(t)$ which completely describes the system at a point in time $t \in \mathcal{T} = [0, T]$ where $T \in \mathbb{R}$ is some time horizon. In general, the state cannot be manipulated or observed directly. Instead, one can indirectly influence the state through some input $u(t) \in \mathcal{U}$ and observe its response through some output $y(t) \in \mathcal{Y}$. The system can consequently be seen as a mapping from the space of input signals $\mathbb{U} = \{u \mid u : \mathcal{T} \to \mathcal{U}\}$ to the space of output signals $\mathbb{Y} = \{y \mid y : \mathcal{T} \to \mathcal{Y}\}$

$$S[u] = y, \quad S : \mathbb{U} \to \mathbb{Y}. \tag{2.1}$$

A model M can also be seen as a transformation $M : \mathbb{U} \to \mathbb{Y}$. By giving the model some input u it yields a prediction $M[u] = \hat{y}$ of the system output. Moreover, the model has an internal state $\hat{x}$ which serves as an estimate of the system state x. In many applications, being able to estimate the internal state of the system is crucial. A schematic representation of the system and the model can be seen in figure 2.1.



(a) System. (b) Model.

Figure 2.1: Schematic drawing of a system and a model.

In order to measure the performance of the model one can introduce a loss functional $L : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$ on the form

$$L[y, \hat{y}] = \int_{\mathcal{T}} l(y(t), \hat{y}(t)) \, \mathrm{d}t \tag{2.2}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is some distance measure in the output space. If the model M accurately approximates the system then the loss is small. Note that this does not necessarily mean that the model state $\hat{x}$ accurately approximates the system state x. For a given input signal u, we will consider the modelling problem to correspond to the following optimization problem

$$M_\star = \operatorname*{argmin}_M L[M[u], S[u]] \tag{2.3}$$

That is, we want to find a model that minimizes the loss for a given input signal.

2.1.2 White-, black- and grey-box models

There are numerous ways to classify different types of mathematical models used for describing physical systems. For example, on can classify the models according to the nature of the knowledge that was used to construct them. There are at least two fundamentally distinct approaches for building models that result in so called *white-box* and *black-box* models respectively. These two approaches can be combined and the resulting models are referred to as *grey-box* models. The model used in this thesis will be of this latter type.

White-box models

Sometimes one has sufficient prior information about the system such that it is possible to construct a model from first-principles. These models are referred to as first-principle, mechanistic or phenomenological models but are also termed *white-box* models due to the transparent nature of their construction [1, 2]. While such models are often very accurate due to the inherent validity of the first principles used to build them, they can be very cumbersome and difficult to construct.

In the case of physical systems, these models often take the form of differential equations that can be derived from physical laws or conservation relations. A component of the model used in this thesis will be a white-box model in the form of a partial differential equation which will be discussed in section 2.3.

Black-box models

When there is little knowledge about the system or when modelling from first principles is too difficult, a data-driven (empirical) approach may be more viable. In order to obtain information about the system, one measures how it responds to different input signals. That is, given an input signal u the corresponding output $S[u] = y$ is measured. It is important to pick the input signal so as to retrieve as much information about the system as possible. The overall procedure of identifying a model from data is known as *system identification* which includes the task of generating as informative data as possible [3].

Now assume that the model M is on the form of some parametrised function M_θ with parameters θ. The optimization problem in equation (2.3) can then be written as

$$\theta_\star = \underset{\theta}{\operatorname{argmin}}\, J(\theta) \tag{2.4}$$

where

$$J(\theta) = L[M_\theta[u], S[u]]. \tag{2.5}$$

This class of optimization problems where the task is to fit some function to input-output data are referred to as a *supervised learning* problems in machine learning contexts.

Due to the lack of prior information about the system, the model M_θ needs to be very flexible in the sense that it in principle is able to realize any transformation. This property often entail that the number of parameters in such models is very large and that they are difficult to interpret. Consequently, they are referred to as *black-box* models [1, 2]. In addition, these models seldom give any guarantees on how well they predict the system behaviour outside the domain in which they have been identified.

Artificial neural networks are archetypical black-box models and are a popular choice when it comes to data-driven modelling. Artificial neural networks will be covered in more depth in section 2.2 and will be a core component in the methodology of this thesis.

Grey-box models

Grey-box or *hybrid-models* attempt to combine the advantages and redeem the disadvantages of white-box and black-box models. They do so by partially deriving the model from first principles while still estimating unknown components of the system from data [1, 2]. These models can often be separated into black-box and white-box components. In this way, as much prior information as possible about the system can be incorporated in the model, leaving only what is necessary to be estimated directly from data.

Given that these models include black-box components, the optimization problem in equation (2.4) still holds for the grey-box modelling case.

2.2 Artificial neural networks

Artificial neural networks are computational models inspired by biological neural networks and are used within a wide range of machine learning tasks [4, 5, 6]. Just like a biological neural network, an artificial neural network consists of a system of interconnected neurons. The connections and neurons in the network are associated with numbers (weights) that determine how the neurons interpret the information received from other neurons. The networks can *learn* to perform some task by adjusting these weights in order to alter the way information is propagated through the network. This process is referred to

as *training*. Artificial neural networks have relatively recently received much attention after having achieved state-of-the-art results in areas such as computer vision and speech recognition [7, 8, 9].

2.2.1 Multi-layer perceptron

One of the simplest and most commonly used artificial neural networks is the *multi-layer perceptron* [4, 5]. It is a feed-forward network which implies that its connections can be seen as an acyclic directed graph. As such, information can only propagate in one direction through the network. In the multi-layer perceptron the neurons are arranged in layers such that each neuron is connected to the neurons in the next layer (see figure 2.2). Information can be fed to the network through the first layer and then propagated through the network, layer by layer, until the last layer is reached. The resulting values in the neurons of the last layer serve as output of the network. The in-between layers of the network which are not used for input and output are commonly referred to as *hidden* layers.

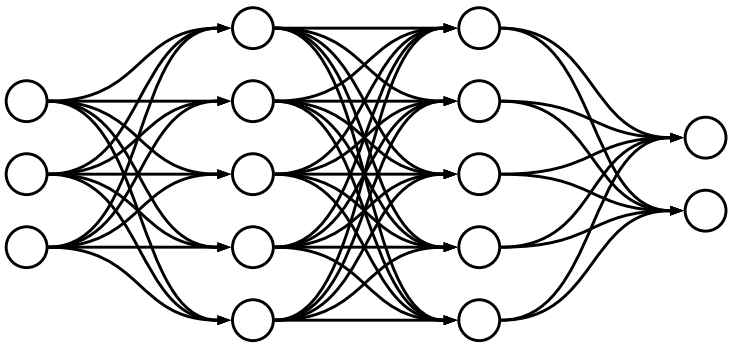


Figure 2.2: Multi-layer perceptron with two hidden layers.

To propagate the input through the network, the neurons in each layer perform a local operation that combines the *activation* of the neurons in the previous layer. Let $\boldsymbol{a}_i \in \mathbb{R}^{n_i}$ denote the activations of the neurons in layer i containing n_i neurons. Then the activations of the next layer is computed as follows

$$\boldsymbol{a}_{i+1} = \varphi_{i+1}\left(W_i \boldsymbol{a}_i + \boldsymbol{b}_{i+1}\right) \tag{2.6}$$

where the matrix $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$ and the bias $\boldsymbol{b}_{i+1} \in \mathbb{R}^{n_{i+1}}$ are adjustable weights and φ_{i+1} is some non-linear function referred to as an *activation function*. A common choice for the activation function is the sigmoid function

$$\varphi(x) = \frac{1}{1 + e^{-x}}. \tag{2.7}$$

Some input $\boldsymbol{x} \in \mathbb{R}^{n_1}$ can be given to the network by setting $\boldsymbol{a}_1 = \boldsymbol{x}$. The corresponding output $\boldsymbol{y}$ is then simply the $\boldsymbol{a}_\ell$ for a network with ℓ layers. This process of giving the network some input and reading the resulting output can be regarded as a mathematical transformation. Consequently a multi-layer perceptron with n input neurons and m output neurons can be seen as a function $N : \mathbb{R}^n \to \mathbb{R}^m$.

Multi-layer perceptrons are able to realize a very wide range of functions. It has been shown that multi-layer perceptrons with a single hidden layer are *universal function approximators* with only mild assumptions on the activation function [10]. Note that this does not give any information on how many neurons that should be used in the hidden layer or how to find the weights in order to approximate a given function.

In order to train the multi-layer perceptron, the learning task is formulated as an optimization problem similar to equation (2.4). The resulting optimization problem is often high dimensional and in general one has to resort to gradient based optimization algorithms to solve it [4, 5, 6]. Even though such algorithms only finds local minima, the final performance of the network is often sufficient.

2.3 Distributed parameter systems

Distributed parameter systems are systems whose state space cannot be fully described by a finite number of parameters (i.e. their state space is infinite dimensional) [1]. Instead, the state is described by the *distributions* of a number of parameters. On the other hand, a *lumped parameter system* is a system that can be described by a finite number of parameters [1]. For example, the state of a rigid body is specified by six parameters (position and orientation) while the state of a fluid is given by the distribution of several variables such as the velocity and pressure at different points in space.

A distinguishing feature of models that describe distributed parameter systems is that they can incorporate spatial variation. These models are often derived from conservation relations for the corresponding physical system and take the form of partial differential equations. In this thesis we will consider a distributed parameter system and model it with a system of partial differential equations as a white-box component in a grey-box model.

This section introduces the concept of a partial differential equation and its corresponding weak formulation. Finally, the finite difference and the finite element methods are described, which are numerical methods for solving partial differential equations.

2.3.1 Partial differential equations

A partial differential equation (PDE) is a differential equation for some multivariate function that includes the partial derivatives of that function [11]. In this thesis, we will consider the independent variables in the PDE to be time t and spatial position $\boldsymbol{r} = (r_1, \ldots, r_d)$. That is, we will consider PDEs on the form

$$f\left(t, \boldsymbol{r}, \frac{\partial \psi}{\partial t}, \frac{\partial^2 \psi}{\partial t^2}, \ldots, \frac{\partial \psi}{\partial r_1}, \ldots, \frac{\partial \psi}{\partial r_n}, \frac{\partial^2 \psi}{\partial r_1 r_2}, \ldots\right) = 0, \quad t \in \mathcal{T}, \boldsymbol{r} \in \Omega \tag{2.8}$$

where $\mathcal{T} = [0, T]$ is some time interval, $\Omega \subset \mathbb{R}^d$ is some domain in space and $\psi : \mathcal{T} \times \Omega \to \mathbb{R}$ is the unknown function. Just as for any differential equation, boundary conditions must be specified for the PDE to have a unique solution. In this thesis we will study PDEs where it is sufficient to specify an initial condition

$$\psi(0, \boldsymbol{r}) = \psi_0(\boldsymbol{r}), \quad \boldsymbol{r} \in \Omega \tag{2.9}$$

and a boundary condition

$$g\left(t, \boldsymbol{r}, \frac{\partial \psi}{\partial t}, \frac{\partial^2 \psi}{\partial t^2}, \ldots, \frac{\partial \psi}{\partial r_1}, \ldots, \frac{\partial \psi}{\partial r_n}, \frac{\partial^2 \psi}{\partial r_1 r_2}, \ldots\right) = 0, \quad t \in \mathcal{T}, \boldsymbol{r} \in \partial\Omega \tag{2.10}$$

where $\partial\Omega$ is the boundary of Ω.

PDEs can describe phenomena such as wave propagation, diffusion and advection. Consequently PDEs naturally arise in fields such as acoustics, electrodynamics, heat transfer and fluid dynamics. Some of the most famous PDEs are Maxwell's equations that describe electrodynamics [12] and the Navier-Stokes equation that describe fluid motion [13].

2.3.2 Weak formulation

A partial differential equation along with its boundary conditions has a corresponding *weak formulation* [14, 15, 16]. The weak formulation is more permitting in the sense that it allows for (weak) solutions that only need to be weakly differentiable. These solutions are defined with respect to certain *test*

functions. In this sense, the weak formulation is equivalent to formulating the problem so as to allow for a solution in the form of a *distribution*. While the reverse is true, a solution to the weak formulation of a PDE is not necessarily a solution to the PDE itself. In addition, the solution to the weak formulation is in general not unique.

To illustrate the concept of a weak formulation, consider the Poisson equation

$$-\nabla^2\psi(\boldsymbol{r}) = f(\boldsymbol{r}), \quad \boldsymbol{r} \in \Omega \tag{2.11}$$

$$\psi(\boldsymbol{r}) = g(\boldsymbol{r}), \quad \boldsymbol{r} \in \partial\Omega. \tag{2.12}$$

where $f : \Omega \to \mathbb{R}$ and $g : \partial\Omega \to \mathbb{R}$ are given functions. We will seek a weak solution $\psi : \Omega \to \mathbb{R}$ in some function space V whose characteristics will become clear later. A weak formulation of the Poisson equation can then be written

$$-\int_\Omega (\nabla^2\psi)\cdot\varphi \,\mathrm{d}r = \int_\Omega f\cdot\varphi \,\mathrm{d}r \tag{2.13}$$

which should hold for any test function φ in some function space $\hat{V}$. Note that the boundary conditions in equation 2.12 are not included explicitly in the weak formulation. Instead, they can be satisfied by incorporating them in the function space V which effectively imposes restrictions on the solution ψ. To reduce the constraints on V, the weak formulation is integrated by parts so as to remove higher order derivatives.

$$-\int_\Omega (\nabla^2\psi)\cdot\varphi \,\mathrm{d}r = -\int_{\partial\Omega} \nabla\psi\cdot\varphi \,\mathrm{d}\boldsymbol{S} + \int_\Omega \nabla\psi\cdot\nabla\varphi \,\mathrm{d}r \tag{2.14}$$

Choosing $\hat{V}$ such that $\varphi(\boldsymbol{r}) = 0$ on the boundary $\partial\Omega$ we get the alternative weak formulation

$$\int_\Omega \nabla\psi\cdot\nabla\varphi \,\mathrm{d}r = \int_\Omega f\cdot\varphi \,\mathrm{d}r, \quad \forall\varphi \in \hat{V}. \tag{2.15}$$

It suffices to require that the first order derivatives of functions in V are square integrable for the weak formulation in equation (2.15) to be valid. To be precise, the function spaces V and $\hat{V}$ can be chosen as

$$V = \{\psi \in H^1(\Omega) \mid \psi(\boldsymbol{r}) = g(\boldsymbol{r}), \boldsymbol{r} \in \partial\Omega\} \tag{2.16}$$

$$\hat{V} = \{\varphi \in H^1(\Omega) \mid \varphi(\boldsymbol{r}) = 0, \boldsymbol{r} \in \partial\Omega\} \tag{2.17}$$

where $H^1(\Omega)$ is a Sobolev space defined as

$$H^1(\Omega) = \left\{\psi \in L^2(\Omega) \mid \frac{\partial\psi}{\partial r_i} \in L^2(\Omega), 1 \le i \le d\right\} \tag{2.18}$$

and where $L^2(\Omega)$ is the set of square integrable functions on Ω .

2.3.3 Numerical methods

In order to represent distributed parameter systems numerically they are usually approximated by lumped parameter systems. For the partial differential equation studied in this thesis this corresponds to *discretizing* the time and space dimensions such that an approximate finite dimensional model is created. There are several approaches for performing this discretization out of which the finite difference, finite volume, finite element and spectral methods are the most common [14, 17, 15]. In this thesis the finite difference method will be used to discretize the time dimension and the finite element method will be used to discretize the spatial domain. We select the finite difference method in time due to its simplicity and the finite element method in space since it handles complex geometries with ease.

Finite difference method

In the finite difference method differential equations are approximated by difference equations. This corresponds to approximating derivatives by finite differences. For example, the time derivative $\dot{\psi} = \partial\psi/\partial t$ may be approximated using backward differences

$$\dot{\psi}(t) = \frac{\psi(t) - \psi(t - \Delta t)}{\Delta t} + \mathcal{O}(\Delta t) \tag{2.19}$$

There are several finite difference schemes, which can be classified as explicit or implicit. An explicit method can be written as a difference equation on the form

$$\psi(t + \Delta t) = F(\psi(t)) \tag{2.20}$$

which in general is straightforward to solve for $\psi(t + \Delta t)$. For an implicit method, computing $\psi(t + \Delta t)$ involves solving an equation on the form

$$G(\psi(t + \Delta t), \psi(t)) = 0. \tag{2.21}$$

While this often implies that implicit methods are more computationally demanding, they are more numerically stable which allows for larger time steps.

Certain differential operators result in particularly computationally demanding implicit methods. In these cases it may be feasible to split the operator into two operators where one is treated explicitly and the other implicitly

$$\psi(t + \Delta t) = H_{\text{imp}}(\psi(t + \Delta t)) + H_{\text{exp}}(\psi(t)). \tag{2.22}$$

In these cases, linear terms in the operator are often treated implicitly while non-linear terms are treated explicitly. These mixed methods are sometimes

referred to as implicit-explicit methods [18]. In this thesis an implicit-explicit method was used.

Finite element method

The finite element method is a numerical method for finding approximate solutions to the weak formulation of a partial differential equation. To discretize the problem defined by the weak formulation, the infinite dimensional function spaces V and $\hat{V}$ are approximated by discrete subspaces $V_h \subset V$ and $\hat{V}_h \subset \hat{V}$. These discrete function spaces are constructed by partitioning the spatial domain Ω into a finite number of small sub-domains referred to as elements. This partitioning can be carried out using a triangulation algorithm that constructs a polygonal mesh which approximates the domain. At every vertex ν in the mesh a local basis function $\phi_\nu : \Omega \to \mathbb{R}$ is defined. In this thesis we will use piecewise linear basis functions (see figure 2.3).

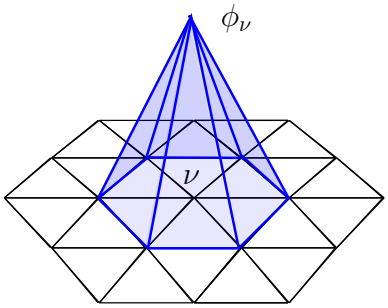


Figure 2.3: Piecewise linear basis function located at vertex ν.

These basis functions can then be combined to form a function on the form

$$\psi(\boldsymbol{r}) = \sum_{\nu \in \mathcal{V}} c_\nu \phi_\nu(\boldsymbol{r}), \quad c_\nu \in \mathbb{R}. \tag{2.23}$$

where $\mathcal{V}$ is the set of vertices in the mesh. For this particular choice of basis functions we have that $\psi(\boldsymbol{r}_\nu) = c_\nu$ where $\boldsymbol{r}_\nu$ is the position of vertex ν. Varying the coefficients c_ν allows for a range of functions to be realized. By choosing the basis functions to comply with some specified boundary condition they may serve as basis functions for a discrete function space.

To exemplify how to proceed with solving a discrete version of the weak formulation, consider the discretized weak formulation of the Poisson equation

$$\int_\Omega \nabla\psi \cdot \nabla\varphi \, \mathrm{d}r = \int_\Omega f \cdot \varphi \, \mathrm{d}r, \quad \forall \varphi \in \hat{V}_h. \tag{2.24}$$

Let $\{\phi_\nu\}_{\nu\in\mathcal{V}}$ and $\{\hat{\phi}_\nu\}_{\nu\in\mathcal{V}}$ be the basis functions for V_h and $\hat{V}_h$ respectively, assuming that the dimensions of the two spaces are equation. Writing ψ in terms of the basis functions of V_h as in equation (2.23) and setting φ to be one of the basis functions $\hat{\phi}_\mu$ gives

$$\sum_{\nu\in\mathcal{V}} c_\nu \int_\Omega \nabla\phi_\nu \cdot \nabla\hat{\phi}_\mu \,\mathrm{d}r = \int_\Omega f \cdot \hat{\phi}_\mu \,\mathrm{d}r. \tag{2.25}$$

This can be interpreted as a matrix equation

$$\sum_{\nu\in\mathcal{V}} A_{\mu\nu} c_\nu = b_\mu \tag{2.26}$$

where

$$A_{\mu\nu} = \int_\Omega \nabla\phi_\nu \cdot \nabla\hat{\phi}_\mu \,\mathrm{d}r, \qquad b_\mu = \int_\Omega f \cdot \hat{\phi}_\mu \,\mathrm{d}r. \tag{2.27}$$

The overlap between basis functions is small due to their local nature, causing the matrix equation to become very sparse. Consequently, the solution to the weak formulation in equation (2.24) is given by the solution to the sparse matrix equation in equation (2.26).

The procedure of writing ψ in terms of the basis functions in V_h while setting ϕ to each of the basis functions of $\hat{V}_h$, can be carried out for any weak formulation. This results in a sparse system of *algebraic equations* for the coefficients in equation (2.23) which are not necessarily linear. Solving this system of equations allows for finding an approximate solution to the weak formulation of a partial differential equation.

2.4 Automatic differentiation

Since a grey-box model will be used in this thesis, an optimization problem equivalent to that in equation equation (2.4) will have to be solved. In particular, the grey-box model will consist of a black-box component in the form of a multi-layer perceptron and a white-box component in the form of a partial differential equation. As mentioned in section 2.2.1, multi-layer perceptrons are usually trained using gradient based optimization methods. In its simplest form this corresponds to iteratively updating the parameters θ of the network accordingly

$$\theta_{i+1} = \theta_i - \eta \frac{\mathrm{d}J}{\mathrm{d}\theta} \tag{2.28}$$

where J is given by equation 2.5 and the step size $\eta \in \mathbb{R}$ is known as the *learning rate*. Due to the intricate structure of the grey-box, it is very difficult to

derive a closed analytical expression for the derivative $\mathrm{d}J/\,\mathrm{d}\theta$ and it may also be infeasible to estimate through finite differences. To compute the derivative we will use the method of *automatic differentiation* [19].

Automatic differentiation is a method for computing the derivative of a function which can be seen as a composition of many elementary operations. By assuming that the derivative of each elementary operation can be computed analytically, the chain rule is utilized to compute the derivative of the function. This is implemented by first constructing a computational graph of the function and then computing the derivative using the forward or reverse mode of automatic differentiation. In this thesis, the reverse mode will be used.

2.4.1 Computational graphs

Consider a set of functions $f_1, \ldots, f_n$ that depend on a number of independent variables $x_1, \ldots, x_m$. Now assume that the functions can be seen as a compositions of several elementary operations. A *computational graph* is an acyclic directed graph which indicates how the elementary functions are combined to give a more complicated function. A node in the graph either corresponds to an independent variable x_i or the output of an elementary operation applied to the values at its parent nodes.

An example of a computational graph representing the functions

$$\begin{aligned} f_1(x_1, x_2) &= h_1(g_1(x_1, x_2), g_2(x_1, x_2)) \\ f_2(x_1, x_2) &= h_2(g_1(x_1, x_2), g_2(x_1, x_2)) \end{aligned} \tag{2.29}$$

can be seen in figure 2.4.

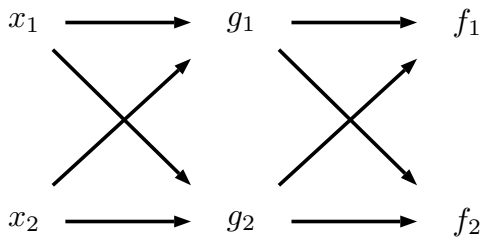


Figure 2.4: Computational graph.

2.4.2 Forward and reverse mode

Since the computational graph stores the order in which the elementary operations are applied it can be used to evaluate the derivative $\partial f_i/\partial x_j$ using the

chain rule. This is done by systematically assembling the components of the chain rule corresponding to the derivatives at each node in the graph. This process is referred to as the forward or backward mode of automatic differentiation depending on the order in which this assembly is carried out.

Forward mode

In the forward mode of automatic differentiation one first *fixes the independent variable* x_j, with respect to which one wants to perform the differentiation. Starting from this independent variable one computes the derivative of each sub-expression in the graph recursively. To illustrate this process, consider computing the derivative of the function in equation 2.29.

$$\begin{aligned}\frac{\partial f_i}{\partial x_j} &= \frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_j} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_j} = \\ &= \frac{\partial h_i}{\partial g_1}\left(\frac{\partial g_1}{\partial x_1}\frac{\partial x_1}{\partial x_j} + \frac{\partial g_1}{\partial x_2}\frac{\partial x_2}{\partial x_j}\right) + \frac{\partial h_i}{\partial g_2}\left(\frac{\partial g_2}{\partial x_1}\frac{\partial x_1}{\partial x_j} + \frac{\partial g_2}{\partial x_2}\frac{\partial x_2}{\partial x_j}\right)\end{aligned} \tag{2.30}$$

In the forward mode, the expression above is evaluated in the order indicated by the parentheses. Note that this means that the expressions inside the parentheses corresponding to $\partial g_1/\partial x_j$ and $\partial g_2/\partial x_j$ need to be recomputed for $j = 1$ and $j = 2$. In contrast they are independent of the choice $i = 1$ or $i = 2$. Consequently, the forward mode is efficient when computing the derivative of many dependent variables with respect to a few independent variables (i.e. $m \ll n$).

Reverse mode

In the reverse mode[1] of automatic differentiation one first *fixes the dependent variable* f_i, with respect to which one wants to perform the differentiation. Starting from this dependent variable one computes the derivative with respect to each sub-expression in the graph recursively. To illustrate how this contrasts to the forward mode, consider computing the derivative of the function in equation (2.29) using the reverse mode.

$$\begin{aligned}\frac{\partial f_i}{\partial x_j} &= \frac{\partial f_i}{\partial x_1}\frac{\partial x_1}{\partial x_j} + \frac{\partial f_i}{\partial x_2}\frac{\partial x_2}{\partial x_j} = \\ &= \left(\frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_1} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_1}\right)\frac{\partial x_1}{\partial x_j} + \left(\frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_2} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_2}\right)\frac{\partial x_2}{\partial x_j}\end{aligned} \tag{2.31}$$

[1] In machine learning context, the reverse mode of automatic differentiation is referred to as the *backpropagation algorithm* [20].

The parentheses indicate the evaluation order carried out in the reverse mode. In this case, the expressions inside the parentheses corresponding to $\partial f_i/\partial x_1$ and $\partial f_i/\partial x_2$ need to be recomputed for $i = 1$ and $i = 2$, while they are independent of the choice $j = 1$ or $j = 2$. The reverse mode is therefore efficient when computing the derivative of few number of dependent variables with respect to many independent variables (i.e. $m \gg n$). As will become clearer later, this makes the reverse mode preferable to use in this thesis where the derivative of the loss function J needs to computed with respect to the parameters in the multi-layer perceptron.

2.4.3 Adjoint equation

The white-box component in this thesis can be seen as some function $f(x_1, \ldots, x_m)$ which involves solving a system of partial differential equations. It will be considered as an elementary operation and consequently we need to be able to compute the derivative $\partial f/\partial x_i$ for each i. This can be done efficiently by solving the *adjoint equation*.

To derive the adjoint equation, consider the following alternative formulation of the white-box component

$$f(x_1, \ldots, x_m) = \tilde{f}(\psi(x_1, \ldots, x_m), x_1, \ldots, x_m) \tag{2.32}$$

where the function $\psi(x_1, \ldots, x_m)$ is implicitly defined as the solution to a partial differential equation on the form

$$F(\psi, x_1, \ldots, x_m) = 0 \tag{2.33}$$

which is defined by the parameters $x_1, \ldots, x_m$. The chain rule now gives

$$\frac{\partial f}{\partial x_i} = \frac{\partial \tilde{f}}{\partial \psi}\frac{\partial \psi}{\partial x_i} + \frac{\partial \tilde{f}}{\partial x_i}. \tag{2.34}$$

In this equation $\partial \tilde{f}/\partial \psi$ and $\partial \tilde{f}/\partial x_i$ are in general straightforward to compute, while computing $\partial \psi/\partial x_i$ requires some more work. Implicitly differentiating equation (2.33) with respect to x_i gives

$$\frac{\partial F}{\partial \psi}\frac{\partial \psi}{\partial x_i} = -\frac{\partial F}{\partial x_i}. \tag{2.35}$$

This equation is known as the tangent linear equation[2]. Solving it for $\partial \psi/\partial x_i$ corresponds to the forward mode of automatic differentiation since it does

[2]The tangent linear equation is also referred to as the sensitivity equation.

not depend on the function f. As noted above, this is not preferable for the scenario in this thesis. Instead we will continue to derive the adjoint equation, corresponding to the reverse mode of automatic differentiation.

Suppose for the moment that the tangent linear equation is invertible such that we can write

$$\frac{\partial \psi}{\partial x_i} = -\left(\frac{\partial F}{\partial \psi}\right)^{-1} \frac{\partial F}{\partial x_i}. \tag{2.36}$$

Inserting this expression into equation (2.34) gives

$$\frac{\partial f}{\partial x_i} = -\frac{\partial \tilde{f}}{\partial \psi}\left(\frac{\partial F}{\partial \psi}\right)^{-1} \frac{\partial F}{\partial x_i} + \frac{\partial \tilde{f}}{\partial x_i}. \tag{2.37}$$

Taking the adjoint (Hermitian transpose) of the above equation gives

$$\frac{\partial f}{\partial x_i}^* = -\frac{\partial F}{\partial x_i}^*\left(\frac{\partial F^*}{\partial \psi}\right)^{-1} \frac{\partial \tilde{f}}{\partial \psi}^* + \frac{\partial \tilde{f}}{\partial x_i}^*. \tag{2.38}$$

Now introduce the *adjoint variable* λ as

$$\lambda = \left(\frac{\partial F^*}{\partial \psi}\right)^{-1} \frac{\partial \tilde{f}}{\partial \psi}^*. \tag{2.39}$$

Rearranging gives the adjoint equation

$$\frac{\partial F^*}{\partial \psi}\lambda = \frac{\partial \tilde{f}}{\partial \psi}^*. \tag{2.40}$$

By solving this equation for λ, the desired derivative can be computed as follows

$$\frac{\partial f}{\partial x_i} = -\lambda^* \frac{\partial F}{\partial x_i} + \frac{\partial \tilde{f}}{\partial x_i} \tag{2.41}$$

Note that equation (2.40) does not include x_i. Consequently, solving the adjoint equation allows for computing $\partial f / \partial x_i$ for any i through the equation (2.41).

2.5 Related work

The concept of grey-box modelling is well established in the literature and it has proven useful in many domains. In the context of modelling physical systems, grey-box models have in particular been used to model chemical and biochemical processes. Stosch et al. [2] gives a very detailed review of the state of grey-box modelling in the literature.

Many grey-box model structures have been proposed. A serial grey-box model was proposed by Psichogios and Ungar [21] in which the black-box model is intended to capture the unknown dynamics corresponding to some parameter in the white-box model. A grey-box model in which the black- and white-box components were placed in a parallel manner was proposed by Kramer, Thompson, and Bhagat [22] and Su et al. [23]. The purpose of such a model was for the black-box component to compensate for modelling flaws in the white box component. Common choices for the black-box components in these models are artificial neural networks such as the multi-layer perceptron [24, 25, 21, 26] and the radial basis function network [4, 5, 27, 22].

Depending on the structure of the grey-box and the choice of black-box model, different methods for identifying the parameters in the black-box component can be used. In the *direct* approach, the expected output of the black-box model is computed from the experimentally measured system output [26, 2]. In this way, input-output pairs of the black-box models can be generated and standard optimization methods for the given black-box model can be used to identify its parameters. This procedure is however often prohibitive, since calculating the expected output of the black-box model can involve solving complex inverse problems. This is the case for grey-box models such as the serial model described above, where one has to invert the transformation corresponding to the white-box component. Instead, the *indirect* approach is more viable. In this method, the sensitivity of the grey-box model output with respect to the black-box model output is computed [2]. This corresponds to computing the derivative of the grey-box output with respect to the black-box output. For the serial structure this has mainly been done by solving the tangent linear equations [21, 24]. In this thesis, the indirect method will be used, but we will solve the adjoint equation to compute the desired derivative.

A comparative study of the extrapolation properties of a black-box and a serial grey-box model was carried out by Can et al. [28]. It was shown that for a particular task, the grey-box model was able to perform better than the black-box model outside the respective domains in which they were identified.

As mentioned above, grey-box models have mainly been applied to chemical and biochemical processes [2]. Modelling applications for chemical systems include for instance chemical reactors, polymerization processes, crystallization, metallurgic processes, distillation columns and drying processes. For biochemical processes, grey-box models have for example been used to describe yeast fermentation as well as cultivations of fungi and bacteria. In terms of domain independent applications, grey-box models have been used for monitoring, controlling and optimization of different process systems. Due

to their good extrapolation properties, grey-box models have also been used for predicting how the dynamics of a system changes as the system size is increased. Lastly, grey-box models have also been studied as means of reducing model complexity while still maintaining high accuracy of the original system.

Romijn et al. [24] proposed a serial grey-box model for approximating nonlinear terms in PDEs with the purpose of reducing the computational complexity. They used a finite-element scheme for discretizing the PDE in combination with principal component analysis in order to reduce the order of the system further. To test their model, they studied a one-dimensional heat convection-conduction model representing a glass melting process. A multi-layer perceptron was used to capture the unknown radiative heating contribution to the model. Deng, Li, and Chen [29] developed a grey-box model for modelling a snap curing oven used in the semiconductor packaging industry which is described by a distributed thermal process. To discretize the distributed system a spectral method was used. They used radial basis function networks for both estimating the state of the system as well as identifying unknown dynamics in the system. Input and output to the system was restricted to specific heaters and sensors. Qi et al. [27] used the same grey-box approach to model a flexible manipulator partially described by the one-dimensional Euler-Bernoulli beam equation. Similarly they also restricted observations to the end point of the beam. Gupta et al. [25] applied a grey-box model to the process of column floatation used for separating hydrophobic and hydrophilic species. They modelled the concentration of these substances through a one-dimensional PDE which was discretized using the finite difference method. Unknown parameters such as floatation rates were modelled using a multi-layer perceptron which was trained using the direct approach.

Grey-box modelling for distributed parameter systems has been discussed in a more general setting by Liu and Jacobsen [30] and Liu [31]. They discuss some potential pitfalls associated with the discretization of distributed parameter systems in the context of grey-box modelling.

While the use of the adjoint equation is an established methodology in PDE-constrained optimization [32], its usage is less common in training grey-box models. Of particular relevance to this thesis is the work by Berg and Nyström [33]. They proposed a method for solving inverse PDE problems by representing the unknown quantity by a multi-layer perceptron network. This can be seen as a PDE-constrained optimization problem which they solved by using the adjoint equation together with the BFGS optimization algorithm [34]. They used the finite element method to solve the PDE and the corresponding adjoint equation.

Chapter 3

Method

In this chapter, a method for modelling unknown dynamics in distributed systems will be presented. Even though the method in principle can be applied to a wide range of distributed systems, it will be explained in the context of modelling chemical reactions. The method is of a grey-box type meaning that is is constructed by combining prior knowledge about the system with a data-driven approach for identifying unknown dynamics.

The outline of the chapter is the following. First, an introduction to chemical reactions and a description of the distributed system in which the reaction takes place is given. Then a grey-box method for identifying the unknown parts of the reaction equations is presented. Finally, details on the different experiments that were carried out to evaluate the method are provided.

3.1 A distributed reaction system

The system considered is that of a distributed reaction system corresponding to a fluid in which one can dissolve chemical substances. These substances can be transported in the fluid through diffusion as well as advection. This is what makes the system intrinsically distributed. In addition, the chemicals can undergo chemical reactions that result in new chemical substances. The system can be influenced by adding chemicals to the fluid and its state can be partially observed by measuring the concentrations of the chemicals at specific sensor locations.

In this section, the model that was used to simulate the above mentioned reaction system is described. From such simulations, one can artificially construct measurement data that could have been collected from a real physical system. From this data the task was to estimate the interaction between the

chemicals in the fluid.

3.1.1 Chemical reactions

Consider a chemical reaction where the reactants A and B react to give the product C. Such a reaction can be represented by the following chemical equation

$$\alpha A + \beta B \rightarrow \gamma C \tag{3.1}$$

where $\alpha, \beta, \gamma \in \mathbb{N}^+$ are referred to as *stoichiometric coefficients*. The rate at which a reaction takes place is given by the *reaction rate*. For a closed system of constant volume in which the chemicals are uniformly distributed, the reaction rate is defined as

$$R = -\frac{1}{\alpha}\frac{\mathrm{d}[A]}{\mathrm{d}t} = -\frac{1}{\beta}\frac{\mathrm{d}[B]}{\mathrm{d}t} = \frac{1}{\gamma}\frac{\mathrm{d}[C]}{\mathrm{d}t} \tag{3.2}$$

where $[A]$, $[B]$ and $[C]$ denote the concentration of each respective chemical [35]. In general, the reaction rate cannot be determined theoretically, but has to be estimated empirically. A commonly used model for the reaction rate in chemical reactions can be written

$$R = k[A]^a[B]^b \tag{3.3}$$

where k is the reaction rate constant which does not depend on the concentration of the reactants (see figure 3.1) [35]. The exponents a and b are referred to as partial reaction orders and depend on the particular mechanism of the reaction. For a one-step reaction these equal the stoichiometric coefficient for the corresponding reactant. This is not the case for multi-step reactions which may involve intermediate reactions that have different reaction rates [35]. The total reaction order of the chemical reaction is given by the sum of the partial reaction orders.

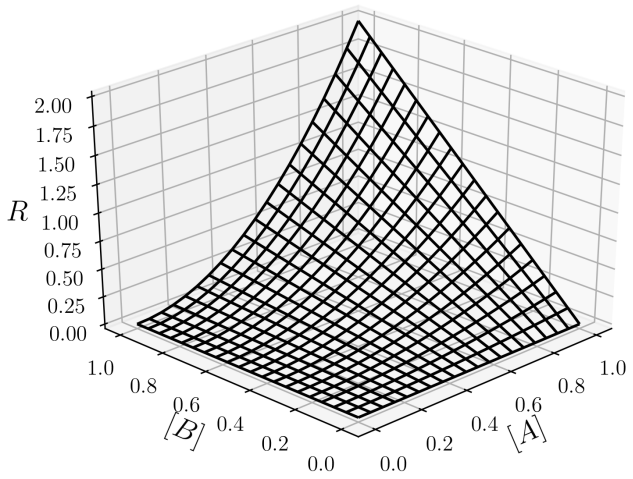


Figure 3.1: Reaction rate model $R = 2[A]^2[B]$.

3.1.2 Reaction-advection-diffusion equations

Let A, B and C be three chemicals that react according to equation (3.1). The reaction takes place in a fluid in which the chemicals are dissolved. Due to advection and diffusion the concentration of each chemical varies in time and space. Let $c_1 = [A]$, $c_2 = [B]$ and $c_3 = [C]$ denote the concentration of each respective chemical. Then these concentrations are functions $c_i : \mathcal{T} \times \Omega \to \mathbb{R}$ where $\mathcal{T} = [0, T]$ is a time interval and $\Omega \subset \mathbb{R}^d$ the spatial domain in which we study the dissolved chemicals. The desired dynamics of the reaction system can then be described by the following system of partial differential equations

$$\begin{aligned} \dot{c}_1 + \boldsymbol{w} \cdot \nabla c_1 - D\nabla^2 c_1 &= -\alpha R(c_1, c_2) + g_1 \\ \dot{c}_2 + \boldsymbol{w} \cdot \nabla c_2 - D\nabla^2 c_2 &= -\beta R(c_1, c_2) + g_2 \\ \dot{c}_3 + \boldsymbol{w} \cdot \nabla c_3 - D\nabla^2 c_3 &= \gamma R(c_1, c_2) + g_3 \end{aligned} \tag{3.4}$$

where $D \in \mathbb{R}^+$ is a diffusion constant and $\boldsymbol{w} : \mathcal{T} \times \Omega \to \mathbb{R}^d$ is the velocity field that transports the chemicals in the fluid through advection. The reaction rate $R : \mathbb{R}^2 \to \mathbb{R}$ determines the reaction kinetics. Lastly, $g_i : \mathcal{T} \times \Omega \to \mathbb{R}^+$ for $i = 1, 2, 3$ are source terms corresponding to adding each corresponding chemical to the system.

By writing $\boldsymbol{c} = (c_1, c_2, c_3)$, equation (3.4) can be written more compactly as

$$\dot{\boldsymbol{c}} + (\boldsymbol{w} \cdot \nabla)\boldsymbol{c} - D\nabla^2 \boldsymbol{c} = \boldsymbol{f}(c_1, c_2) + \boldsymbol{g}, \tag{3.5}$$

where $\boldsymbol{g} = (g_1, g_2, g_3)$ and

$$\boldsymbol{f} = \begin{pmatrix} -\alpha \\ -\beta \\ \gamma \end{pmatrix} R(c_1, c_2). \tag{3.6}$$

Initially the system does not contain any chemicals which corresponds to the initial condition

$$\boldsymbol{c}(t = 0, \boldsymbol{r}) = \boldsymbol{0}, \quad \boldsymbol{r} \in \Omega. \tag{3.7}$$

In addition, assuming that the chemicals do not diffuse across the borders of the domain gives the boundary condition

$$\nabla c_i(t, \boldsymbol{r}) \cdot \boldsymbol{n} = 0, \quad t \in \mathcal{T}, \boldsymbol{r} \in \partial\Omega \tag{3.8}$$

for $i = 1, 2, 3$ and where $\boldsymbol{n} \in \mathbb{R}^d$ is an unit outward normal vector to the boundary $\partial\Omega$. This boundary condition is appropriate when modelling walls and boundaries where the flow of chemicals is dominated by advection.

3.1.3 Input and output

Input

The system can be influenced by dissolving more chemicals into the fluid. This is modelled by the source terms g_1, g_2 and g_3 in equation (3.4). For simplicity and with practical applications in mind, these source terms will be chosen such that one is restricted to only adding each chemical at a specific region of the fluid. In particular we will assume the following separable form

$$g_i(t, \boldsymbol{r}) = u_i(t)\chi_i(\boldsymbol{r}), \quad i = 1, 2, 3 \tag{3.9}$$

where $u_i : \mathcal{T} \to \mathbb{R}^+$ is some signal and $\chi_i : \Omega \to \{0, 1\}$ is a characteristic function on the form

$$\chi_i(\boldsymbol{r}) = \begin{cases} 1, & \boldsymbol{r} \in \Omega_i \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

where Ω_i is a small subset of Ω. Consequently the signals $u_i(t)$ can be seen as input to the system. Equation (3.9) can be written in vector form as follows

$$\boldsymbol{g}(t, \boldsymbol{r}) = \boldsymbol{u}(t) \odot \boldsymbol{\chi}(\boldsymbol{r}) \tag{3.11}$$

where $\boldsymbol{u} = (u_1, u_2, u_3)$, $\boldsymbol{\chi} = (\chi_1, \chi_2, \chi_3)$ and $\odot$ is the Hadamard product.

Output

At any point in time, the state of the system can be partially observed through a number of sensors. This corresponds to measuring the concentrations $\boldsymbol{c}$ at specific locations $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_M$. In addition, each measurement is associated with some noise. Consequently the measurement $\boldsymbol{y}_i : \mathcal{T} \to \mathbb{R}^3$ from a sensor at location $\boldsymbol{r}_i \in \Omega$ can be written

$$\boldsymbol{y}_i(t) = \boldsymbol{c}(t, \boldsymbol{r}_i) + \boldsymbol{\varepsilon}_i(t) \tag{3.12}$$

where $\boldsymbol{\varepsilon}_i$ is some random signal corresponding to the noise in the observation. We will denote the output from all sensors at a given time by $\boldsymbol{y} : \mathcal{T} \to \mathbb{R}^{3M}$.

By collecting measurements from the sensors, information about the dynamics of the system can be obtained. In particular, this data can be used to recover the stoichiometric coefficients and the reaction rate of the chemical reaction in the system.

A schematic overview of the domain Ω along with input domains Ω_i and sensor placement can be seen in figure 3.2.

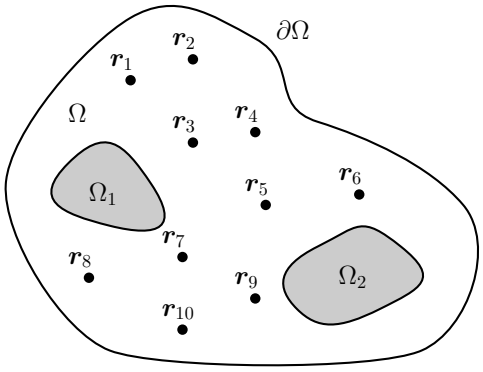


Figure 3.2: Domain Ω of the reaction system along with two input domains Ω_1, Ω_2 and 10 sensor placements $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{10}$.

3.1.4 Simulating the system

In order to generate artificial data from the reaction system, the reaction-advection-diffusion equations need to be solved. This will be done numerically using the finite difference and finite element methods for the time and space dimensions respectively.

Temporal discretization

The time interval $\mathcal{T} = [0, T]$ is uniformly discretized with a sampling period Δt such that $T = N\Delta t$ where $N \in \mathbb{N}^+$. Now introduce the following notation for the different quantities in the system at these sampled times

$$\begin{aligned} \boldsymbol{u}^n &= \boldsymbol{u}(n\Delta t), \\ \boldsymbol{c}^n(\boldsymbol{r}) &= \boldsymbol{c}(n\Delta t, \boldsymbol{r}), \\ \boldsymbol{w}^n(\boldsymbol{r}) &= \boldsymbol{w}(n\Delta t, \boldsymbol{r}), \\ \boldsymbol{f}^n(\boldsymbol{r}) &= \boldsymbol{f}(c_1^n(\boldsymbol{r}), c_2^n(\boldsymbol{r})). \\ \boldsymbol{y}_i^n &= \boldsymbol{c}^n(\boldsymbol{r}_i) + \boldsymbol{\varepsilon}_i^n, \\ \boldsymbol{y}^n &= \left(\boldsymbol{y}_1^n, \ldots, \boldsymbol{y}_M^n\right)^T, \end{aligned} \tag{3.13}$$

where $n \in [1, N]$. Given this discretization, equation (3.5) can be discretized using an implicit-explicit procedure into the following difference equation[1]

$$\frac{\boldsymbol{c}^n - \boldsymbol{c}^{n-1}}{\Delta t} + (\boldsymbol{w}^n \cdot \nabla)\boldsymbol{c}^n - D\nabla^2\boldsymbol{c}^n = \boldsymbol{f}^{n-1} + \boldsymbol{g}^n \tag{3.14}$$

where $\boldsymbol{g}^n = \boldsymbol{u}^n \odot \boldsymbol{\chi}$. By solving this equation for $\boldsymbol{c}^n$ given $\boldsymbol{c}^{n-1}$ we can calculate the sensor outputs $\boldsymbol{y}^n$.

Starting with the initial condition $\boldsymbol{c}^0 = \boldsymbol{0}$, equation (3.14) can be solved iteratively. This results in a time series of outputs $\{\boldsymbol{y}^n\}_{n=1}^N$ that can be seen as sampled measurements from a real physical system. The time series $\{\boldsymbol{u}^n\}_{n=1}^N$ corresponds to the input to the system. The procedure of computing $\{\boldsymbol{y}^n\}_{n=1}^N$ for a given input $\{\boldsymbol{u}^n\}_{n=1}^N$ is shown in algorithm 1.

Algorithm 1: Simulating the reaction system.

input: input signal $\{\boldsymbol{u}^n\}_{n=1}^N$
output: output signal $\{\boldsymbol{y}^n\}_{n=1}^N$

$\boldsymbol{c}^0 \leftarrow \boldsymbol{0}$
for $n \leftarrow 1$ **to** N **do**
    $\boldsymbol{f}^{n-1} \leftarrow \boldsymbol{f}(c_1^{n-1}, c_2^{n-1})$
    $\boldsymbol{c}^n \leftarrow$ ***TimeStep(*** $\boldsymbol{c}^{n-1}, \boldsymbol{f}^{n-1}, \boldsymbol{u}^n$***)*** `// Solve equation (3.14)`
    $\boldsymbol{y}^n \leftarrow$ ***ObserveWithNoise(*** $\boldsymbol{c}^n$***)*** `// Equation (3.13)`

[1] Note that this is still a partial differential equation with respect to the spatial dimensions.

Spatial discretization

The finite element method will be used solve equation (3.14). Since the equation is linear in $\boldsymbol{c}^n$ its weak formulation can be written

$$a(\boldsymbol{c}^n, \boldsymbol{v}) = L(\boldsymbol{v}) \tag{3.15}$$

where

$$a(\boldsymbol{c}^n, \boldsymbol{v}) = \int_\Omega \frac{1}{\Delta t}\boldsymbol{c}^n \cdot \boldsymbol{v} + (\boldsymbol{w}^n \cdot \nabla)\boldsymbol{c}^n \cdot \boldsymbol{v} + D\nabla \boldsymbol{c}^n \cdot \nabla \boldsymbol{v} \,\mathrm{d}r \tag{3.16}$$

$$L(\boldsymbol{v}) = \int_\Omega \left(\frac{1}{\Delta t}\boldsymbol{c}^{n-1} + \boldsymbol{f}^{n-1} + \boldsymbol{g}^n\right) \cdot \boldsymbol{v} \,\mathrm{d}r. \tag{3.17}$$

The finite element method will be used to solve this weak formulation of the partial differential equation numerically using piecewise linear basis functions.

FEniCS [36, 37] will be used to solve the above weak formulation of the partial differential equation. It is a computing platform that provides several tools for solving partial differential equations using the finite element method. Specifically, we will use the Python package provided by DOLFIN [38, 39], which is the main programming interface to the FEniCS problem solving framework.

3.2 Grey-box modelling using neural networks and partial differential equations

Now we will turn to the problem of modelling the above described reaction system under the assumption that the dynamics of the chemical reaction are unknown. Specifically, it will be assumed that neither the stoichiometric coefficients α, β, γ nor the reaction rate R in equation (3.4) are known. The remaining properties of the system such as advection, diffusion, sources and boundary conditions are still assumed to be modelled as described above.

Given these assumptions on the prior knowledge about the system, it is natural to consider a grey-box approach to model the reaction system. The white-box component will correspond to the reaction-advection-diffusion equations. A neural network will be used as a black-box component to model the unknown dynamics of the chemical reaction. In order to train the neural network, input and output data is collected by dissolving chemicals into the fluid and measuring its response through the sensors.

3.2.1 Neural network modelling of the chemical reaction

A neural network will be used to model the chemical reaction in the system, corresponding to $\boldsymbol{f}$ in equation (3.6). When constructing the network, we will incorporate the fact that the chemical reaction is partly described by the stoichiometric coefficients and partly by the reaction rate.

The reaction rate R will be modelled by a multi-layer perceptron $N : \mathbb{R}^2 \to \mathbb{R}$ with one hidden layer containing 8 neurons. This function can be written explicitly as

$$N(c_1, c_2) = \boldsymbol{v}^T \varphi \left(W \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \boldsymbol{b} \right) + a. \tag{3.18}$$

where $\boldsymbol{v} \in \mathbb{R}^8$, $\boldsymbol{b} \in \mathbb{R}^8$, $W \in \mathbb{R}^{8\times 2}$, $a \in \mathbb{R}$ and φ is the sigmoid function applied to each element of the vector. Multi-layer perceptrons with single hidden layers are common black-box models in grey-box modelling contexts [24, 25, 21, 26]. The number of neurons in the hidden layer was set to 8 after some initial experiments that indicated that this gave the network sufficient expressibility so to model the chemical reaction.

The stoichiometric coefficients are simply represented by a vector $\boldsymbol{q} \in \mathbb{R}^3$. To approximate $\boldsymbol{f}$ we construct the following the mapping $\hat{\boldsymbol{f}} : \mathbb{R}^2 \to \mathbb{R}^3$ as follows

$$\hat{\boldsymbol{f}}(c_1, c_2) = \boldsymbol{q} N(c_1, c_2). \tag{3.19}$$

This function can be interpreted as a multi-layer perceptron with two hidden layers as depicted in figure 3.3. In addition, $\hat{\boldsymbol{f}}$ can be represented by the computational graph in figure 3.4.

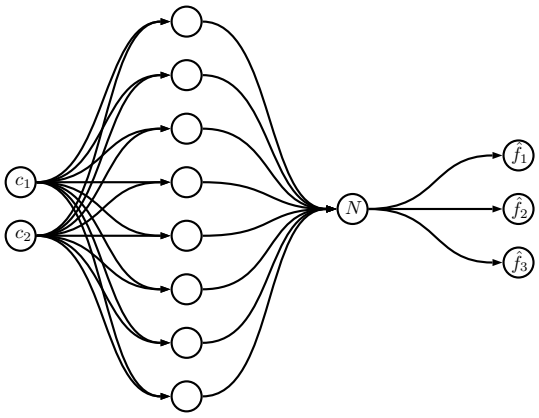


Figure 3.3: Schematic overview of the neural network.

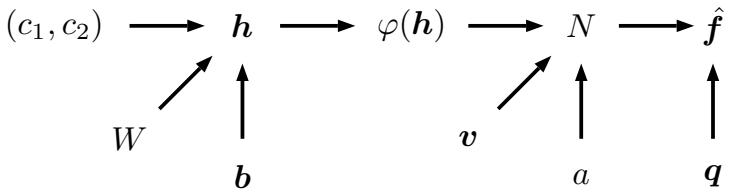


Figure 3.4: Computational graph of the neural network.

In order to approximate the chemical reaction the parameters $\boldsymbol{q}$, $\boldsymbol{v}$, W, $\boldsymbol{b}$, a of the network need to be identified. For brevity, these parameters will jointly be referred to as $\boldsymbol{\theta} \in \mathbb{R}^p$ and to indicate that they parametrize the network we will occasionally denote its corresponding function as $\hat{\boldsymbol{f}}(c_1, c_2; \boldsymbol{\theta})$ such that $\hat{\boldsymbol{f}} : \mathbb{R}^2 \times \mathbb{R}^p \to \mathbb{R}^3$.

In this thesis, the neural network library PyTorch [40] was used to implement the neural network.

3.2.2 Grey-box model structure

The reaction-advection-diffusion equations in combination with the neural network will be used to estimate the evolution of the reaction system. Analogously to the procedure in section 3.1.4 this is done by solving the reaction-advection-diffusion equations numerically using the finite difference and finite element method. The only difference in principle is that $\boldsymbol{f}$ is exchanged for $\hat{\boldsymbol{f}}$ corresponding to the neural network approximation of the chemical reaction. Similarly to equation (3.13), the following notation for the estimated quantities in the system is used

$$\begin{aligned} \hat{\boldsymbol{c}}^n(\boldsymbol{r}) &= \hat{\boldsymbol{c}}(n\Delta t, \boldsymbol{r}), \\ \hat{\boldsymbol{y}}_i^n &= \hat{\boldsymbol{c}}^n(\boldsymbol{r}_i), \\ \hat{\boldsymbol{y}}^n &= \left(\hat{\boldsymbol{y}}_1^n, \ldots, \hat{\boldsymbol{y}}_M^n\right)^T. \end{aligned} \tag{3.20}$$

To compute $\hat{\boldsymbol{f}}^n$ as an estimate for $\boldsymbol{f}^n$ we assume that it can be written in terms of the finite element basis functions as in equation (2.23). Given this assumption it suffices to consider the coefficients in the basis expansion. To be precise, $\hat{\boldsymbol{f}}$ is computed accordingly

$$\hat{\boldsymbol{f}}^n(\boldsymbol{r}; \boldsymbol{\theta}) = \sum_{\nu \in \mathcal{V}} \hat{\boldsymbol{f}}(\hat{c}_1^n(\boldsymbol{r}_\nu), \hat{c}_2^n(\boldsymbol{r}_\nu); \boldsymbol{\theta}) \phi_\nu(\boldsymbol{r}). \tag{3.21}$$

Note that this limits $\hat{\boldsymbol{f}}^n$ to be piecewise linear. This means that it might not be able to estimate $\boldsymbol{f}^n$ perfectly, but may still serve as an accurate approximation.

For the estimated quantities in equation (3.20) and equation (3.21) we get the following analogous equation

$$\frac{\hat{\boldsymbol{c}}^n - \hat{\boldsymbol{c}}^{n-1}}{\Delta t} + (\boldsymbol{w}^n \cdot \nabla)\hat{\boldsymbol{c}}^n - D\nabla^2\hat{\boldsymbol{c}}^n = \hat{\boldsymbol{f}}^{n-1} + \boldsymbol{g}^n \tag{3.22}$$

where $\boldsymbol{g}^n = \boldsymbol{u}^n \odot \boldsymbol{\chi}$. Given $\hat{\boldsymbol{c}}^{n-1}$, $\boldsymbol{w}^n$ and $\boldsymbol{g}^n$ the equation can be solved for $\hat{\boldsymbol{c}}^n$. The estimated measurements $\hat{\boldsymbol{y}}_i^n$ can then be computed according to equation (3.20). Since the initial condition $\hat{\boldsymbol{c}}^0 = \boldsymbol{c}^0 = \boldsymbol{0}$ is known, equation (3.22) can be solved repeatedly starting from $n = 1$ such that a time series of sensor outputs can be estimated. That is, given the initial condition $\hat{\boldsymbol{c}}^0 = \boldsymbol{0}$, the inputs $\{\boldsymbol{g}^n\}_{n=1}^N = \{\boldsymbol{u}^n\boldsymbol{\chi}\}_{n=1}^N$, the flow field $\{\boldsymbol{w}^n\}_{n=1}^N$ and the network parameters $\boldsymbol{\theta}$ one can through this procedure generate the estimated concentrations $\{\hat{\boldsymbol{c}}^n\}_{n=1}^N$ and the estimated output $\{\hat{\boldsymbol{y}}^n\}_{n=1}^N$ for each time step.

The procedure described above can be seen as a discrete version of the model described in section 2.1. We can analogously consider this procedure to correspond to a model $M_{\boldsymbol{\theta}}$ which maps the input series $\{\boldsymbol{u}^n\}_{n=1}^N$ to the output series $\{\hat{\boldsymbol{y}}\}_{n=1}^N$. That is, it can be seen as the following transformation

$$M_{\boldsymbol{\theta}}[\{\boldsymbol{u}^n\}_{n=1}^N] = \{\hat{\boldsymbol{y}}^n\}_{n=1}^N \tag{3.23}$$

where $\boldsymbol{\theta}$ indicates its dependency on the network parameters. The procedure which corresponds to carrying out this transformation is shown in algorithm 2.

Algorithm 2: Simulating the reaction system using a neural network.

input: input signal $\{\boldsymbol{u}^n\}_{n=1}^N$, network parameters $\boldsymbol{\theta}$
output: estimated output signal $\{\hat{\boldsymbol{y}}^n\}_{n=1}^N$

$\hat{\boldsymbol{c}}^0 \leftarrow \boldsymbol{0}$
for $n \leftarrow 1$ **to** N **do**
    $\hat{\boldsymbol{f}}^{n-1} \leftarrow$ ***Reaction***$(\hat{c}_1^{n-1}, \hat{c}_2^{n-1}; \boldsymbol{\theta})$ `// Equation (3.21)`
    $\hat{\boldsymbol{c}}^n \leftarrow$ ***TimeStep***$(\hat{\boldsymbol{c}}^{n-1}, \hat{\boldsymbol{f}}^{n-1}, \boldsymbol{u}^n)$ `// Solve equation (3.22)`
    $\hat{\boldsymbol{y}}^n \leftarrow$ ***Observe***$(\hat{\boldsymbol{c}}^n)$ `// Equation (3.20)`

3.2.3 Training the neural network

The ability of the grey-box model to accurately estimate the state of the reaction system depends on how well the neural network is able to capture the dynamics of the chemical reaction. That is, the parameters $\boldsymbol{\theta}$ of the neural

network need to be chosen such that $\hat{\boldsymbol{f}}(\cdot;\boldsymbol{\theta})$ approximates $\boldsymbol{f}(\cdot)$ as close as possible. This will be done through a data-driven approach where one measures how the system responds to a given input signal. For the reaction system, this corresponds to dissolving chemicals into the fluid and measuring the resulting concentration of the chemicals at the sensor locations.

The learning problem

The task of learning the parameters in the network can then be formulated as the optimization problem

$$\boldsymbol{\theta}_\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) \tag{3.24}$$

where

$$J(\boldsymbol{\theta}) = L[M_{\boldsymbol{\theta}}[\{\boldsymbol{u}^n\}_{n=1}^N], \{\boldsymbol{y}^n\}_{n=1}^N]. \tag{3.25}$$

The loss function L measures the difference between the estimated output $\{\hat{\boldsymbol{y}}^n\}_{n=1}^N$ and the measured output

$$L[\{\hat{\boldsymbol{y}}^n\}_{n=1}^N, \{\boldsymbol{y}^n\}_{n=1}^N] = \frac{1}{3MN}\sum_{n=1}^{N} \|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|^2. \tag{3.26}$$

where $\|\cdot\|$ is the L^2-norm. The process of computing $J(\boldsymbol{\theta})$ can be represented with the computational graph in figure 3.5 in which $l^n = \|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|^2$.

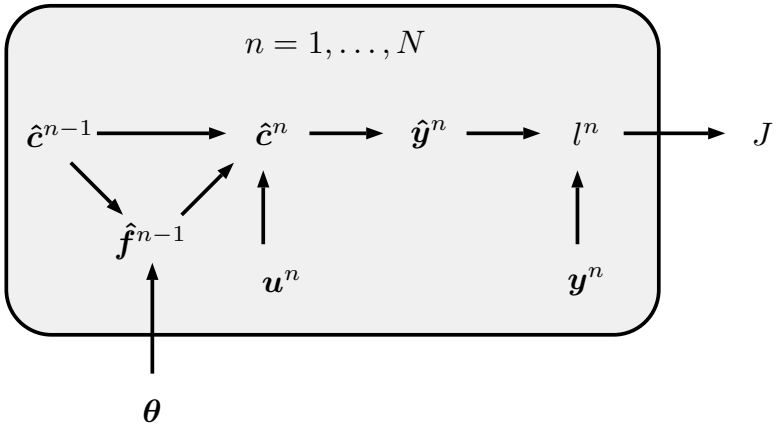


Figure 3.5: Computational graph corresponding to computing the loss J.

This formulation of the learning problem is analogous to the framework introduced in section 2.1 where signals have been replaced by time series and integrals replaced by sums.

The training algorithm

A gradient-based algorithm was used to find solutions corresponding to local minima to the optimization problem in equation (3.24). Such algorithms iteratively update the parameter $\boldsymbol{\theta}$ using the gradient $\mathrm{d}J/\,\mathrm{d}\boldsymbol{\theta}$. To efficiently compute this derivative, the reverse mode of automatic differentiation was used. As described in section 2.4 this corresponds to constructing a computational graph and then carrying out the reverse mode automatic differentiation procedure.

In this thesis, we relied on PyTorch and dolfin-adjoint [41] to construct the computational graph in figure 3.5 and carry out the reverse mode of automatic differentiation. dolfin-adjoint was responsible for constructing the computational graph of the white-box model which was implemented in the FEniCS framework as well as deriving and solving the adjoint-equations. The models created in the PyTorch framework natively support automatic differentiation. To be precise, the white-box component constructed in FEniCS was implemented as a PyTorch module such that the gradients computed by dolfin-adjoint were made available to the PyTorch optimization framework.

Given that the gradient $\mathrm{d}J/\,\mathrm{d}\boldsymbol{\theta}$ can be computed, many gradient-based optimizations algorithms become available. In this thesis, the limited-memory BFGS (L-BFGS) algorithm was used [34]. The L-BFGS algorithm belongs to the family of quasi-Newton methods meaning that it approximates the second-order derivative $\mathrm{d}^2J/\,\mathrm{d}\boldsymbol{\theta}^2$, i.e. the Hessian, in order to make better updates to $\boldsymbol{\theta}$. In contrast to the BFGS algorithm, L-BFGS avoids high memory consumption by not storing the Hessian explicitly, but rather maintains a history of the past iterations in order to approximate the Hessian [34]. In this thesis, a history of 100 iterations was maintained to approximate the Hessian. Just like any gradient-based algorithm one has to chose the step size which is also referred to as the learning rate (see equation (2.28)). Even though it is common to adaptively search for the optimal learning rate at each iteration when using L-BFGS, we used a more simplistic approach with a static learning rate.

The reason for using the L-BFGS algorithm over more commonly used gradient-based optimization algorithms such as Adam [42] is the fact that it approximates the Hessian in order to reach a solution in fewer iterations. For the model used in this thesis, each iteration involves solving the reaction-advection-diffusion equation as well as its corresponding adjoint equation. Due to the fact that partial differential equations are computationally demanding to solve, carrying out as few iterations as possible is of high priority. Even though quasi-Newton algorithms may be more unstable than simpler algo-

rithms, this makes L-BFGS viable to use in this thesis.

In conclusion, automatic differentiation in combination with the L-BFGS algorithm will be used to solve the optimization problem in equation (3.24). This procedure iteratively updates the network parameters $\boldsymbol{\theta}$ until some stopping criteria is fulfilled such as having carried out a specified number of iterations. The network parameters were initialised as proposed by Glorot and Bengio [43]. An overview of the training procedure is given in algorithm 3.

Algorithm 3: Training the neural network.

input: input signal $\{\boldsymbol{u}^n\}_{n=1}^N$, output signal $\{\boldsymbol{y}^n\}_{n=1}^N$
output: parameters of trained network $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow$ ***InitialiseNetwork()*** `// Glorot and Bengio [43]`
for $i \leftarrow 0$ **to** E **do**
    `// Construct computational graph by computing the loss`
    $\hat{\boldsymbol{y}} \leftarrow$ ***Simulate(***$\{\boldsymbol{u}^n\}_{n=1}^N, \boldsymbol{\theta}$***)*** `// Algorithm 2`
    $J \leftarrow L(\{\hat{\boldsymbol{y}}^n\}_{n=1}^N, \{\boldsymbol{y}^n\}_{n=1}^N)$
    `// Compute derivative using the reverse mode of automatic differentiation`
    $\mathrm{d}J/\,\mathrm{d}\boldsymbol{\theta} \leftarrow$ ***ReverseModeAD(***$J, \boldsymbol{\theta}$***)***
    `// Update network parameters`
    $\boldsymbol{\theta} \leftarrow$ ***L-BFGS(***$\boldsymbol{\theta}, \mathrm{d}J/\,\mathrm{d}\boldsymbol{\theta}$***)***

3.3 Evaluation

To test the grey-box model and the method for identifying the parameters in its black-box component, two reaction systems were studied. In this section, we will first describe the general approach to testing the proposed model and the common components between the systems. Then we will present the details of each reaction system and the different tests that were carried out.

3.3.1 Testing the model

Training, validation and test data

In order to test the model on a given reaction system, the first step was to generate artificial data from such a system using algorithm 1. To properly train and evaluate the performance of the grey-box model, the system was simulated

three times using different input signals for each simulation. The resulting input-output pairs $(\{\boldsymbol{u}^n\}_{n=1}^N, \{\boldsymbol{y}^n\}_{n=1}^N)$ from each respective simulation were labelled training, validation and test data. The training data was used to train the neural network in the grey-box model according to algorithm 3. Continuously during the training, the performance of the grey-box model was evaluated on the validation data. In this way, one can monitor how well the grey-box model generalises to data which has not been used for training. When the training was completed, the grey-box model which yielded best performance on the validation set was saved for further testing. The performance of this model was then evaluated on the test data. By using a dataset which has not been used during training, one can get an unbiased estimate of the performance of the model.

Reaction rate

To simulate a reaction system described by the equations in section 3.1 the stoichiometric coefficients α, β, γ and a model for the reaction rate $R(c_1, c_2)$ need to be specified. In this thesis, the following reaction rate was used for both systems

$$R(c_1, c_2) = 2{c_1}^2 c_2. \tag{3.27}$$

Input

The input signal $\{\boldsymbol{u}^n\}_{n=1}^N$ to the system was chosen according to a randomized scheme. This makes sure that a varying range of combinations of chemical concentrations are present in the system and ensures variety between the training, validation and test data.

To be precise, an input signal u_i was chosen according to the following procedure. First, the time interval $\mathcal{T} = [0, N\Delta t]$ is divided into subintervals with varying lengths on the form $m\Delta t$ where $m \in \mathbb{N}$ is chosen uniformly from the interval $[m_{\text{min}}, m_{\text{max}}]$. During each time interval u_i remains constant with a value uniformly sampled from some interval $[u_{\text{min}}, u_{\text{max}}]$.

For the systems studied in this thesis, it was assumed that one could only dissolve chemical A and chemical B into the system. As such, $u_3^n = 0$ for all n while u_1^n and u_2^n are decided from the above described procedure. An example of generated random signals u_1 and u_2 can be seen in figure 3.6.

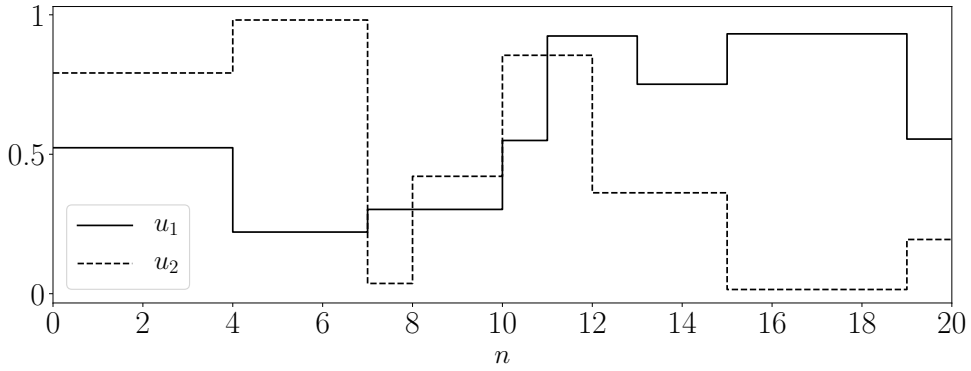


Figure 3.6: Example of random input signals u_1 and u_2 for $m_{\text{min}} = 1$, $m_{\text{max}} = 5$, $u_{\text{min}} = 0$, $u_{\text{max}} = 1$ and $N = 20$.

Apart from choosing the input signal, the input domain Ω_i that defines χ_i needs to be chosen. For the studied reaction systems, these domains were defined according to the following equation

$$\Omega_i = \{\boldsymbol{r} \in \Omega \mid \|\boldsymbol{r} - \boldsymbol{r}_i\| < \rho_i\} \tag{3.28}$$

where $\|\cdot\|$ denotes the L_2 norm. This corresponds to being able to add the chemical corresponding to concentration c_i to the system in a circle of radius ρ_i around point $\boldsymbol{r}_i$.

Output

The noise ε_i^n associated with the each sensor measurement $\boldsymbol{y}_i^n$ was generated by drawing a sample from a normal distribution with zero mean and standard deviation σ. However, it was made sure that the resulting measurement $\boldsymbol{y}_i^n$ always was positive by truncating the sampled noise if necessary.

Performance measures

When it comes to measuring the performance of the trained grey-box model, there are several options. A natural performance measure is simply the loss function in equation (3.26). It measures how well the model predicts the sensor measurements. This loss serves as a proxy for estimating the difference between the estimated and true chemical concentrations in the whole domain. This in turn gives some information about how close the neural network models the chemical reaction.

Note that it is not necessarily the case that a small loss implies that the neural network accurately captures the dynamics of the chemical reaction. It may be that an alternative model of the chemical reaction could lead to the same distribution of chemicals. Such ambiguities are inherent to inverse problems and are not easily dealt with.

Since we are in this thesis studying artificial systems we have access to the ground truth chemical dynamics and concentration distributions. In order to evaluate the method more thoroughly we can thus use two additional performance measures:

$$L_{\boldsymbol{c}}[\{\hat{\boldsymbol{c}}^n\}_{n=1}^N, \{\boldsymbol{c}^n\}_{n=1}^N] = \frac{1}{3N|\Omega|}\sum_{n=1}^{N}\int_{\Omega}\|\hat{\boldsymbol{c}}^n - \boldsymbol{c}^n\|^2\,\mathrm{d}r, \tag{3.29}$$

$$L_{\boldsymbol{f}}[\{\hat{\boldsymbol{f}}^n\}_{n=1}^N, \{\boldsymbol{f}^n\}_{n=1}^N] = \frac{1}{3N|\mathcal{V}|}\sum_{n=1}^{N}\sum_{\nu\in\mathcal{V}}\|\hat{\boldsymbol{f}}^n(\boldsymbol{r}_\nu) - \boldsymbol{f}^n(\boldsymbol{r}_\nu)\|^2. \tag{3.30}$$

The loss $L_{\boldsymbol{c}}$ measures the difference between the estimated concentration distribution and the actual distribution of chemicals in the domain. The loss $L_{\boldsymbol{f}}$ measures the error in the neural network estimation of the chemical reaction dynamics. It is worth emphasizing that these measures would not be accessible in a situation where the ground truth dynamics are unknown.

Identifying the reaction rate and the stoichiometric coefficients

Given that the neural network in the grey-box model has been trained to approximate the chemical reaction, it is desirable to identify the stoichiometric coefficients α, β and γ as well as the reaction rate $R(c_1, c_2)$. Their corresponding estimates may be computed from $\boldsymbol{q} = (q_1, q_2, q_3)^T$ and $N(c_1, c_2)$ in equation (3.19) as follows

$$\hat{\alpha} = -\frac{|q_1|}{\delta}, \quad \hat{\beta} = -\frac{|q_2|}{\delta}, \quad \hat{\gamma} = \frac{|q_3|}{\delta}, \tag{3.31}$$

$$\hat{R}(c_1, c_2) = \delta \max(N(c_1, c_2), 0), \tag{3.32}$$

where $\delta = \min(|q_1|, |q_2|, |q_3|)$. This choice of δ makes sure that the estimated stoichiometric coefficients have no common divisor. To retrieve the reaction rate corresponding to these estimated stoichiometric coefficients the neural

network estimation is multiplied by δ. In addition, the neural network estimation of the reaction rate is truncated since the reaction rate is known to be a non-negative quantity[2].

3.3.2 One-dimensional reaction-diffusion system

The first reaction system that was studied is a one-dimensional reaction-diffusion system. That is, we consider the case when there is no advection in the system which implies that $\boldsymbol{w} = \mathbf{0}$. The domain is taken to be the unit interval $\Omega = [0, 1]$.

Firstly, it was assumed that the system could be observed through $M = 8$ sensors each associated with a noise level of $\sigma = 0.1$. The remaining constants that define the system and the algorithm for simulating the system were chosen according to table 3.1. The grey-box modelling method presented in section 3.2 was then used to train a neural network to model the unknown chemical dynamics in the system. As described in section 3.3.1, the model was trained using a training and validation set while its performance was measured on a test dataset. In addition, the stoichiometric coefficients as well as the reaction rate were identified from the neural network structure. Apart from measuring the final performance of the grey-box model, its performance during the training phase was studied.

Secondly, to investigate the robustness of the grey-box model, the number of sensors M and noise level σ was varied. Data was generated from systems using 1, 2, 4, 8, 16 and 32 sensors for each of the noise levels 0, 0.05, 0.1, 0.15 and 0.2. As before, the remaining constants in the system were chosen according to table 3.1. The grey-box modelling method was applied to each of these 30 datasets in order to estimate its robustness to sparse observations and noise.

In order to construct the finite element spaces that were necessary for the data generation as well as the grey-box modelling method, the domain was discretized uniformly into 100 intervals. For simplicity the sensors in the system were assumed to be located on the vertices of this mesh, i.e. in between the intervals. For a given number of sensors M, the domain was split uniformly into $M + 1$ intervals and the sensor were placed at the closest mesh vertex corresponding to the position in between these intervals (see figure 3.7).

[2]This truncation operation could also be included directly in the grey-box model. Some initial tests indicated that this results in difficulties learning the network parameters.

description	constant	value
diffusion constant	D	0.01
stoichiometric coefficient	α	1
stoichiometric coefficient	β	2
stoichiometric coefficient	γ	1
Ω_1 centre	$\boldsymbol{r}_1$	0.45
Ω_1 radius	ρ_1	0.1
Ω_2 centre	$\boldsymbol{r}_2$	0.55
Ω_2 radius	ρ_2	0.1
u_1 input interval length	$[m_{\text{min}}, m_{\text{max}}]$	$[1, 4]$
u_1 input value interval	$[u_{\text{min}}, u_{\text{max}}]$	$[0, 2]$
u_2 input interval length	$[m_{\text{min}}, m_{\text{max}}]$	$[1, 4]$
u_2 input value interval	$[u_{\text{min}}, u_{\text{max}}]$	$[0, 3]$
time step	Δt	0.15
number of time steps	N	50

Table 3.1: Values of constants used for simulating the system.

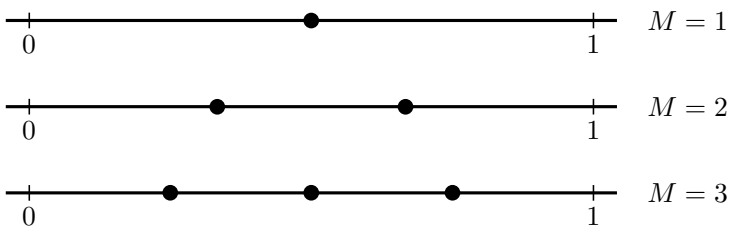


Figure 3.7: Sensor placements in $\Omega = [0, 1]$ for $M = 1, 2, 3$.

The grey-box modelling method requires that a learning rate η is specified for the L-BFGS algorithm. Some initial tests indicated that a rather wide range of learning rates yielded similar performance. One of the lower learning rates $\eta = 0.1$ in this range was chosen to compensate for the unstable nature of the L-BFGS algorithm. The learning algorithm (see algorithm 3) was carried out for 1000 iterations using this learning rate in all the tests described above. The training algorithm was run three times for each test using different network initialisations. This results in three trained grey-box models for each test out of which the model which performed best on the validation set was used for further testing on the test dataset.

3.3.3 Two-dimensional reaction-advection-diffusion system

Lastly, a two-dimensional reaction system was studied. The domain Ω is a channel through which a fluid flows from left to right. In the channel a cylinder is placed, resulting in a non-stationary fluid flow pattern known as a von Kármán vortex street (see figure 3.8). Chemical A and chemical B can be dissolved into the fluid near the left boundary (see figure 3.9). These chemicals react to give the product C while they are transported through the channel via advection and diffusion.

To be precise, the channel corresponds to the rectangle $[0, 5] \times [0, 1.5]$ in which a disk with centre $(0.7, 0.75)$ and radius 0.2 has been cut out. Fluid enters the channel through the border at $r_1 = 0$ with a given velocity profile and freely leaves the system at $r_1 = 5$. The fluid cannot flow through the top ($r_2 = 1.5$), bottom ($r_2 = 0$) or the cylinder. The velocity field $\boldsymbol{w}$ of the resulting fluid motion in the channel was modelled using the incompressible Navier-Stokes equation [13]. A characteristic instance in time of the resulting time-dependent velocity field can be seen in figure 3.8. Details on how these equations were solved using the finite element method to generate the velocity field is given in appendix A.

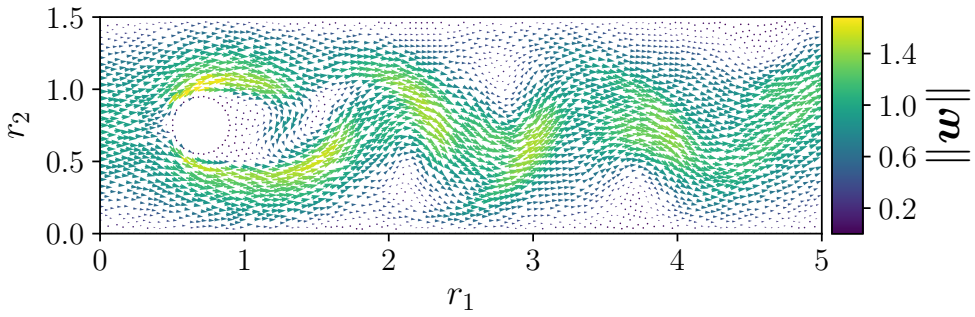


Figure 3.8: Characteristic velocity field $\boldsymbol{w}$ at some time instance of the fluid flow in the channel. The flow exhibits a von Kármán vortex street pattern.

Given the velocity field $\boldsymbol{w}$, the reaction-advection-diffusion equations can be solved to simulate how the chemicals are transported through the channel while they undergo chemical reaction. However, some numerical issues arose when solving the reaction-advection-diffusion equations. Occasionally the resulting chemical concentrations became slightly negative in small regions of the domain. To mitigate this numerical artefact, the concentrations where set to zero in the regions where they were negative. The constants that were used

to simulate the reaction system are given in table 3.2.

description	constant	value
diffusion constant	D	0.02
stoichiometric coefficient	α	1
stoichiometric coefficient	β	2
stoichiometric coefficient	γ	8
Ω_1 centre	$\boldsymbol{r}_1$	$(0.2, 0.6)$
Ω_1 radius	ρ_1	0.1
Ω_2 centre	$\boldsymbol{r}_2$	$(0.2, 0.9)$
Ω_2 radius	ρ_2	0.1
u_1 input interval length	$[m_{\text{min}}, m_{\text{max}}]$	$[5, 10]$
u_1 input value interval	$[u_{\text{min}}, u_{\text{max}}]$	$[5, 10]$
u_2 input interval length	$[m_{\text{min}}, m_{\text{max}}]$	$[5, 10]$
u_2 input value interval	$[u_{\text{min}}, u_{\text{max}}]$	$[5, 10]$
time step	Δt	0.2
number of time steps	N	50

Table 3.2: Values of constants used for simulating the system.

In order to solve the reaction-advection-diffusion equation using the finite element method, the domain is discretized using the triangulation scheme provided by FEniCS. See figure 3.9 for the resulting mesh.

A total of 99 sensors were placed in the domain ordered in a grid with 5 rows. The rows and columns of this grid were placed in a similar procedure as in the one-dimensional system such as to uniformly place the sensors in the domain (see figure 3.9). The noise level was assumed to be $\sigma = 0.01$.

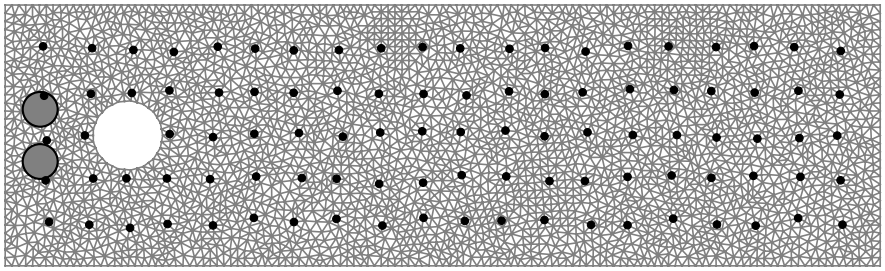


Figure 3.9: Mesh of the domain and the placement of the sensors shown as black circles and the input domains Ω_1 and Ω_2 shown as grey circles.

Just as for the one-dimensional system the training algorithm was carried out with a learning rate of $\eta = 0.1$, but this time 3000 iterations were com-

pleted. In addition, the algorithm was run with several different network initialisations and the one that performed best on the validation set was used for further testing using the test dataset.

Chapter 4

Results

This chapter presents the results from having applied the grey-box modelling method described in section 3.2 to the two reaction systems described in section 3.3.

4.1 One-dimensional reaction-diffusion system

The grey-box modelling method was first applied to the one-dimensional reaction-diffusion system described in section 3.3.2.

In the first test, the system was assumed to be equipped with $M = 8$ sensors with a noise level of $\sigma = 0.1$. After having generated a training, validation and test set, the learning algorithm (algorithm 3) was carried out. For each iteration the loss L (equation 3.26) along with the performance measures $L_{\boldsymbol{c}}$ and $L_{\boldsymbol{f}}$ (equation 3.29 and equation 3.30) are measured on the training data and validation data. As mentioned in section 3.3.2, the algorithm was run with different network initializations. The performance of the resulting models were very similar for the different initializations.

The progress of the training for the best performing model can be seen in figure 4.1. As expected, the loss L measured on the training data decreases for each iteration since this is the loss which the optimization algorithm attempts to minimize. While this is not the case for the losses $L_{\boldsymbol{c}}$ and $L_{\boldsymbol{f}}$ we can see that they nevertheless also decrease significantly. This indicates that the loss L measured at the sensors of the system serves well as a proxy for the error in the neural network estimation. In addition, the losses measured on the validation set closely follows those measured on the training set even though they are

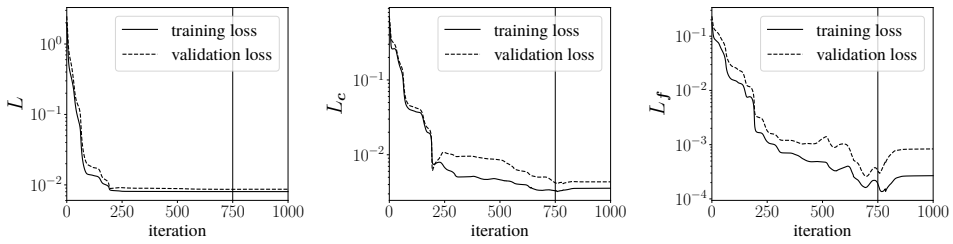


Figure 4.1: Loss functions for each iteration in the training algorithm when applying the grey-box modelling method to the one-dimensional reaction system. The vertical line indicates the iteration corresponding to the smallest loss L measured on the validation data.

slightly larger. The iteration corresponding to the smallest loss L measured on the validation set is indicated by a vertical line in figure 4.1. This iteration did not correspond to the minimal value of $L_{\boldsymbol{c}}$ or $L_{\boldsymbol{f}}$ which highlights the fact that these do not perfectly correlate with L. The grey-box model corresponding to the minimal loss L measured on the validation data during the training, was used for further testing. Firstly, the estimated reaction rate and the estimated stoichiometric coefficients were identified from the black-box component. The estimated stoichiometric coefficients were the following

$$\hat{\alpha} = 1.003, \quad \hat{\beta} = 2.008, \quad \hat{\gamma} = 1.000 \tag{4.1}$$

which is very close to the true values $\alpha = 1$, $\beta = 2$ and $\gamma = 1$. The deviation between the estimated reaction rate and the true reaction rate is shown in figure 4.2. To illustrate which values of c_1 and c_2 were measured a density plot is also shown.

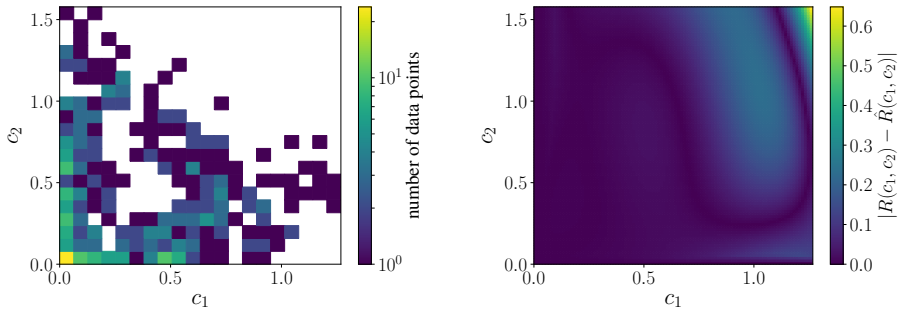


Figure 4.2: The left figure shows a density plot of the measured values of the concentrations c_1 and c_2. The right figure shows the deviation between the estimated reaction rate and the true reaction rate.

Not surprisingly, the reaction rate was best approximated in the domain where most measurement data was available. Lastly, the grey-box model was used to simulate the system given the input signal from the test dataset. A comparison between the estimated and true concentrations for a number of time steps can be seen in figure 4.3. Clearly, the estimated concentrations are very close to the true concentrations.

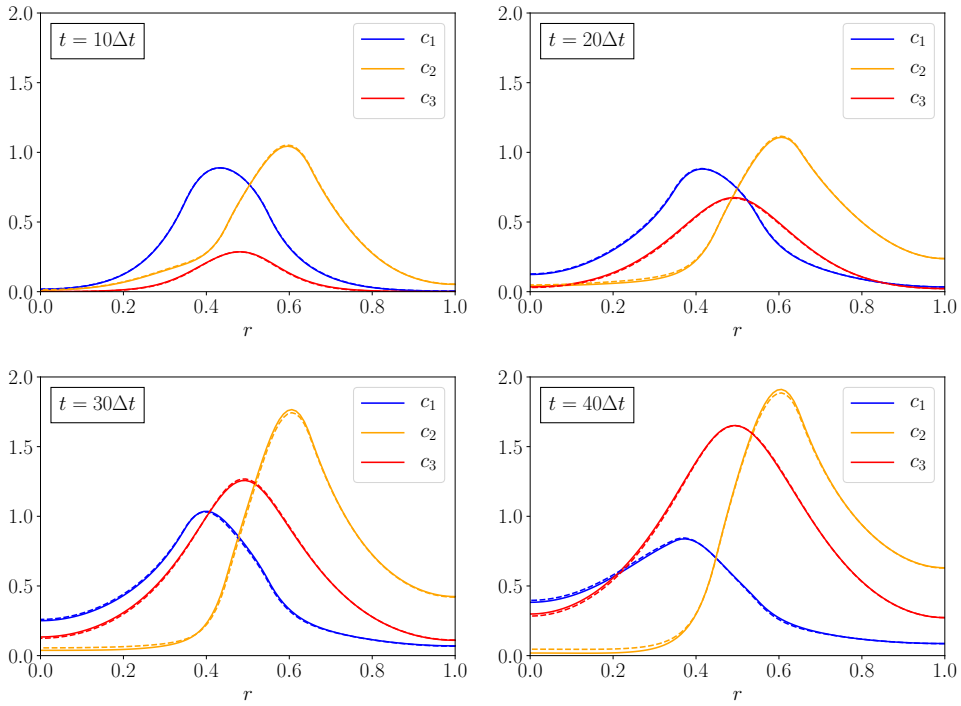


Figure 4.3: Simulation of the one-dimensional reaction-diffusion system using the input signal from the test dataset. The solid lines correspond to the true concentration of each chemical while the dashed lines correspond to the estimated concentrations.

Lastly, the robustness of the grey-box modelling method with respect to the number of sensors and the noise level was tested. The training algorithm was applied to the one-dimensional reaction-diffusion system, but with varying number of sensors and noise levels. The final performance in terms of the loss functions L, L_c and L_f for each of the trained models on the corresponding test datasets is shown in figure 4.4. For all measures, it can be seen that the loss increases as the noise level is increased as expected. Not surprisingly the loss also increases as the number of sensors is decreased. In addition, this decrease becomes more prominent as the noise level is increased. This is also

reasonable since the system should become more sensitive to noise when less measurements can be collected.

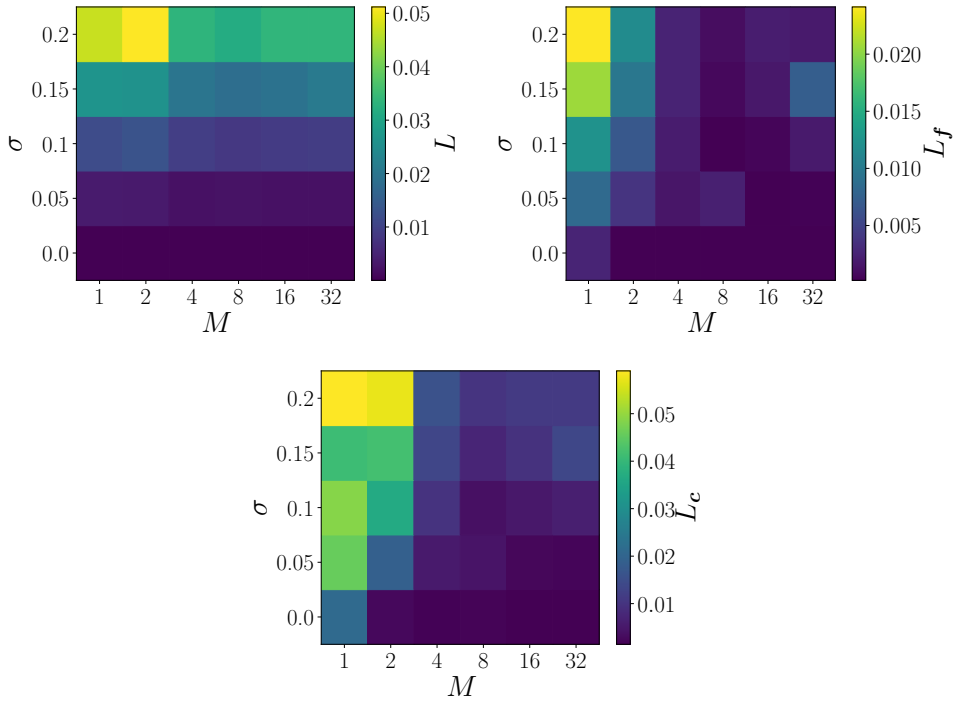


Figure 4.4: Performance of the trained grey-box model in terms of the loss functions L, L_c and L_f measured on the test dataset for different number of sensors M and noise levels σ in the system.

4.2 Two-dimensional reaction-advection-diffusion system

The grey-box modelling method was then applied to the two-dimensional reaction-advection-diffusion system as described in section 3.3.3. Just as for the one-dimensional case, the progress of the training algorithm can be studied by plotting the losses measured on the training and validation set for each iteration. As mentioned in section 3.3.3, the training algorithm was run with different initializations for the neural network. The performance of the resulting models varied. In some cases, the loss stopped decreasing rather early during the training which could be an indication that the algorithm got stuck in a local minima.

The progress during the training for the best performing model is shown in figure 4.5. All losses L, $L_{\boldsymbol{c}}$ and $L_{\boldsymbol{f}}$ decrease significantly during the training even though it is only L that is used for updating the parameters in the neural network. This indicates that the loss L measured in the sensor measurements serves as a good indicator on the overall performance of the grey-box model. A noticeable feature in figure 4.5 is the spike in the loss functions L and $L_{\boldsymbol{c}}$ near iteration 1000. This kind of unstable behaviour is an intrinsic problem with a quasi-Newton method such as L-BFGS which may occasionally cause the minimization scheme to diverge.

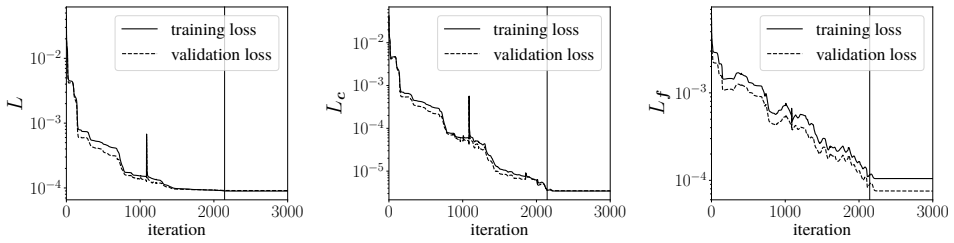


Figure 4.5: Loss functions for each iteration in the training algorithm when applying the grey-box modelling method to the two-dimensional reaction system. The vertical line indicates the iteration corresponding to the smallest loss L measured on the validation data.

The loss functions measured on the test data using the best performing model as measured on the validation were the following

$$L = 9.074 \cdot 10^{-5}, \quad L_{\boldsymbol{c}} = 4.142 \cdot 10^{-6}, \quad L_{\boldsymbol{f}} = 1.117 \cdot 10^{-4}. \tag{4.2}$$

The stoichiometric coefficients and the reaction rate can be estimated from the grey-box model corresponding to the smallest loss measured on the validation set during the training. Through the procedure described in section 3.3.1, the stoichiometric coefficients were identified as

$$\hat{\alpha} = 1.000, \quad \hat{\beta} = 1.975, \quad \hat{\gamma} = 7.693 \tag{4.3}$$

while the true coefficients were $\alpha = 1$, $\beta = 2$ and $\gamma = 8$. In comparison with the corresponding result for the one-dimensional system, we can see that this estimation was not as accurate. The deviation between the estimated and true reaction rate is shown in figure 4.6.

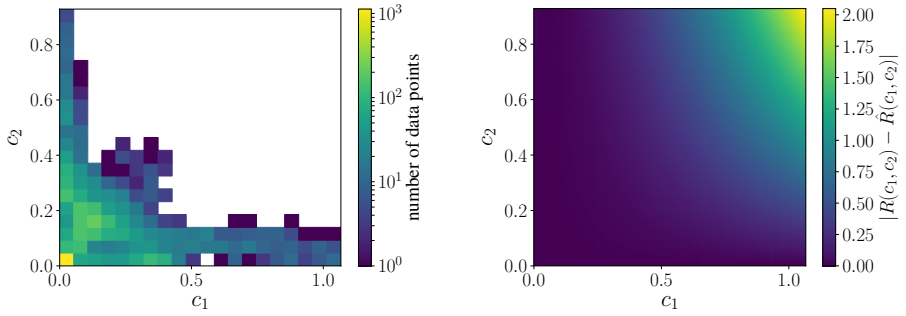


Figure 4.6: The left figure shows a density plot of the measured values of the concentrations c_1 and c_2. The right figure shows the deviation between the estimated reaction rate and the true reaction rate.

Just as for the one-dimensional system, the reaction rate is accurately approximated for chemical concentrations that were present in the system during the simulation corresponding to the training data.

Lastly, the trained grey-box model was used to simulate the reaction system given the inputs from the test dataset. The evolution of the concentration c_3 from the resulting simulation can be seen in figure 4.7. Figure 4.8 shows a comparison between this simulation and the true output in the test dataset. Overall, the estimated concentration distributions are very close to the true distributions. The largest errors are found near the source terms close to the cylinder. This may be explained by the fact that unlike the rest of the domain, the chemicals have not mixed well in these regions. Such concentration distributions are outliers in the data and are likely to be disregarded by the training algorithm.

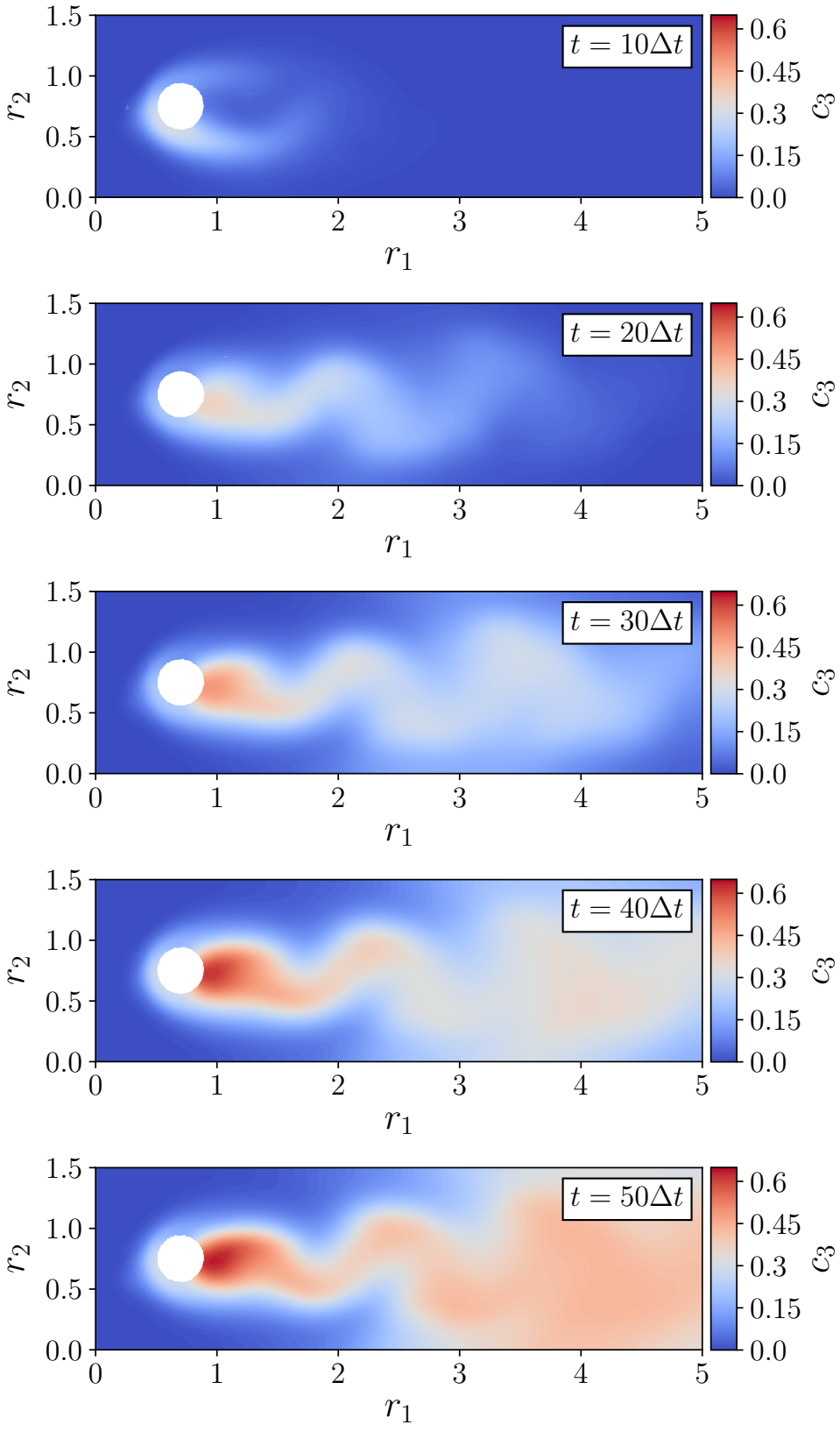


Figure 4.7: Simulation of the two-dimensional reaction-advection-diffusion system using the input signal from the test dataset.

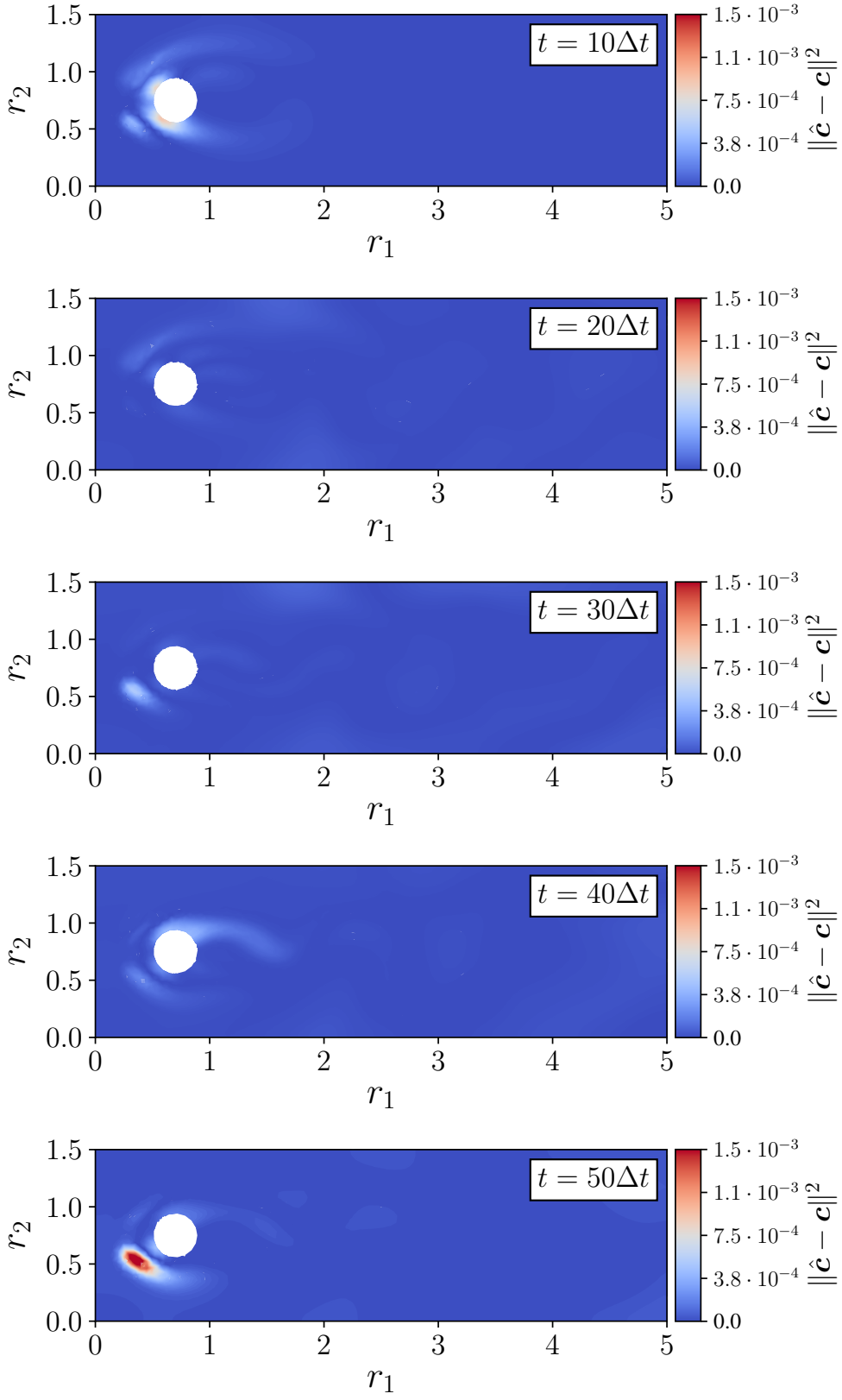


Figure 4.8: Comparison between the estimated concentrations $\hat{\boldsymbol{c}}$ and the true concentrations $\boldsymbol{c}$ in the test dataset.

Chapter 5

Discussion

In this chapter the work carried out in this thesis is discussed. Firstly, the main results presented in chapter 4 are highlighted and discussed. Secondly, the strengths and weaknesses of the grey-box modelling method presented in section 3.2 are discussed. Lastly, an outlook on future work is given followed by some concluding remarks.

5.1 Remarks on the results

The results presented in chapter 4 indicate that the grey-box modelling method was able to accurately capture the two reaction systems that were studied. This can be seen from the low observation loss measured on the test set. However, since we know the model from which the data was generated, we could measure how well the grey-box model was able to recover this model from the data. Firstly this could be done by comparing simulations of the reaction systems carried out with the ground truth reaction dynamics and the trained grey-box model. As can be seen in figure 4.3 and figure 4.8, the simulations closely matched each other. Secondly, the estimated stoichiometric coefficients and reaction rate of the chemical reaction could be computed from the trained grey-box model and compared with the reaction model used for data generation. It could be seen that the reaction rate was accurately recovered and especially for the one-dimensional case the estimated stoichiometric coefficients were very close to the ground truth. It is difficult to speculate on the reason for the lower accuracy on the estimated stoichiometric coefficients for the two-dimensional case. One possibility could simply be the fact that this system involves more complex dynamics due to the fact that it is two-dimensional and includes advection. This in turn causes uncertainty when trying to estimate the reaction

dynamics from sparse observations. Another reason for the lower accuracy could be that the chemicals were less mixed in the two-dimensional case which effectively giving information about the reaction dynamics. Lastly it could be that the threshold used to mitigate numerical artefacts (see section 3.3.3) causes information to be lost during the training process.

Even though these particular reaction systems have not previously been studied in the context of grey-box modelling, the result that the grey-box model was able to approximate their behaviour well is in accordance with previous research. However, the result that the grey-box model was able to recover the underlying reaction dynamics is not necessarily expected. The problem of identifying the dynamics of a system from data (i.e. system identification) is a type of inverse problem. Just like most inverse problems, the collected observations may be ambiguous in the sense that multiple models of the system could lead to the same measured observations. By studying how the different loss functions change during training we can see that this is to some extent the case for the modelling problem in this thesis. As previously noted, we can see that the losses L_c (equation (3.29)) and L_f (equation (3.30) do not necessarily decrease when the loss L (equation (3.26)) decreases. Consequently, this means that the grey-box model temporarily found a way of predicting the output of the system more accurately without becoming more similar to the ground truth model. As noted above, this ambiguity could also be an explanation for the less accurate estimations of the stoichiometric coefficients in the two-dimensional case since the overall behaviour of the system was still predicted rather accurately. Lastly, these ambiguities might also impair the training algorithm in the sense that it may cause the gradient descent scheme to get stuck in undesirable local minima. This was especially observed for the two-dimensional case where the training algorithm was rather sensitive to different initialisations of the network parameters, which could be a consequence of this phenomena.

The difficulties related to solving inverse problems such as system identification tasks may be mitigated by collecting a sufficient amount of data. By collecting more data one can differentiate between models that previously could not be distinguished from each other with less observations. This effect can be observed from the tests on the one-dimensional reaction system where the observability and noise level in the observations were varied. As expected, an increase in noise level or decrease in observability resulted in grey-box models with worse performance (see figure 4.4). That is, the training algorithm was not able to recover the ground truth dynamics of the system given very sparse and noisy data. Interestingly, it can be seen in figure 4.4 that this effect

is most noticeable with very few sensors implying that the gain of adding more sensors successively decreases.

Lastly, it is important to realize that the conclusions from the results in this thesis are somewhat limited due to the fact that we have solely been studying artificial reaction systems. In particular, the dynamics of the white-box component are exactly those that were used to generate the data used for training. The problem with this assumption is twofold. Firstly, it may be the case that the white-box model is flawed. That is, the partial differential equation used to model the distributed properties of the system may be incomplete or erroneous. In reality, this is always the case since it is practically impossible to consider systems without making some assumptions. This means the black-box component has to compensate for this modelling flaw, which may lead to difficulties identifying the parameters of the black-box model. Secondly, as pointed out by Liu and Jacobsen [30] and Liu [31] the process of discretizing a distributed parameter system in the context of grey-box modelling is associated with several pitfalls. The process of reducing a distributed parameter system to a finite dimensional model through discretization is inherently associated with a loss of information. Such errors may not be apparent in the grey-box modelling performance since the black-box component can learn to compensate for such discretization errors. However, this may lead to poor generalization properties of the grey-box model.

5.2 Remarks on the method

The grey-box modelling method presented in section 3.2 is very general. The framework of the method can in principle be used to model any distributed parameter system which contains some unknown dynamics. That is, it can be used to model any quantity or unknown term in a partial differential equation given that one can partially observe the state of the system. This flexibility of the method is made possible by the use of automatic differentiation. Given that one can specify the forward problem, the adjoint equation is solved and intermediate derivatives computed allowing the training algorithm to be carried out. In addition, the usage of the finite element method is also instrumental in being able to describe systems with complex geometries such as the two-dimensional reaction system studied in this thesis.

A major drawback and difficulty in using this method is that it is very computationally demanding. This is due to the fact that many partial differential equations need to be solved in each iteration of the training algorithm. When learning relatively simple dynamics in the system, such as the reaction dynam-

ics considered in this thesis, this might not be so much of a problem. For these cases, it is sufficient to use a rather small neural network such that sophisticated optimization algorithms (e.g. L-BFGS) are effective. However, this might be more of an issue if deeper neural networks are to be used as black-box components in the method. Training algorithms for deep neural networks most often rely on relatively simple gradient descent method, but have to be carried out for a very large number of iterations.

Lastly, another limitation of the grey-box modelling method is that it does not incorporate any of the sensor measurements in order to infer the state of the system. That is, the grey-box estimates the behaviour of the system by starting from a known initial condition and integrating the reaction equations without any feedback from the measurements in the system. Incorporating sensor measurements may not only improve the prediction capabilities of the method, but may also improve the efficiency of the training algorithm. The task of combining observations with model predictions is referred to as data assimilation [44].

5.3 Future work

In terms of continuing the work in this thesis, there are many possible ways forward.

The method can in its current state easily be applied to different distributed parameter systems. Grey-box modelling methods are desirable in almost any modelling problem and thus the potential applications are many. To give some examples, we believe that this method could be useful in areas such as biomedicine, urban planning and weather forecasting. A biomedical application could be that of modelling blood flow in the human heart where the flow through the heart valves is difficult to model from first-principles. In urban planning, it is of interest to know how trees influence the air flow through a city and how they can be used to reduce air pollution. A grey-box model where trees are modelled through a data-driven approach may be viable in this case. Weather forecasting is notoriously difficult and involves many uncertain modelling problems. Using grey-box modelling, prior knowledge about the weather system can be combined with a data-driven method for modelling unknown dynamics such as cloud formation.

There are several parts of the method that may be improved. Currently, we are assuming that the quantity estimated by the neural network must be written in terms of the finite element basis functions (see section 3.2.2). In principle this assumption is redundant since one could rely on the finite element method

to evaluate the neural network when carrying out the numerical integration. Secondly, incorporating data assimilation methods as described in the previous section may also be of great value. Lastly, in order to allow this method to be used with deeper neural networks the training algorithm may have to be optimized. One approach to make it more efficient is to parallelize the gradient computation in each iteration of the algorithm.

5.4 Societal and ethical considerations

The algorithms that constitute the grey-box modelling method proposed in this thesis do no pose any direct ethical issues or impact the society in any harmful way. They are solely meant to solve the related modelling problem and have been developed without any other intention in mind. Nevertheless, since the method is a generic framework for modelling distributed parameter systems it could potentially be used in applications with ethical complications or malicious purposes. This could for example be a problem in the above mentioned biomedical applications where patient specific data must be treated delicately. Rather than causing societal or ethical issues, the developed methods may yield a positive impact on society. As previously noted, modelling is a core component of many fields of science. Consequently, this methodology could potentially enable solutions to or give insights into problems which directly affect society. This is particularly apparent for the above mentioned applications in the domains of biomedicine, urban planning and weather forecasting.

5.5 Conclusion

In this thesis, a grey-box modelling method for describing distributed parameter systems has been developed and evaluated. Through automatic differentiation the method is very flexible and as such it can easily be applied to a wide range of systems. In addition, the finite element method is used to discretize the system enabling the method to handle systems with complex geometries. The method was presented in the context of modelling a reaction-advection-diffusion system and was evaluated on a one-dimensional and a two-dimensional instance of the reaction system. Furthermore, the robustness of the system with respect to observability and measurement noise was investigated. The grey-box model was able to successfully capture the dynamics of the reaction system for the one-dimensional as well as for the two-dimensional

system. The investigation of the robustness of the method indicated that the method was especially sensitive to noise when using very few sensors. In conclusion, the grey-box modelling method is viable to use for modelling the reaction system studied in this thesis and can easily be applicable to other distributed parameter systems.

Bibliography

[1] K.M Hangos and I.T. Cameron. *Process modelling and model analysis*. Vol. 4. Academic Press, 2001. ISBN: 1-281-05413-5.

[2] Moritz von Stosch et al. "Hybrid semi-parametric modeling in process systems engineering: Past, present and future". In: *Computers & Chemical Engineering* (2014). ISSN: 00981354. DOI: 10.1016/j.compchemeng.2013.08.008.

[3] Lennart Ljung. *System identification : theory for the user*. Prentice-Hall information and system sciences series. Englewood Cliffs, NJ: Prentice-Hall, 1987. ISBN: 0-13-881640-9.

[4] John Hertz. *Introduction to the theory of neural computation*. Santa Fe Institute studies in the sciences of complexity Lecture notes volumes, 1. Redwood City, Calif.: Addison-Wesley, 1991. ISBN: 0-201-50395-6.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006. ISBN: 978-0-387-31073-2.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (2015). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00124. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[8] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

[9] Geoffrey Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *Ieee Signal Processing Magazine* November (2012), pp. 82–97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597. arXiv: 1207.0580.

[10] G Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 5.4 (Dec. 1992), p. 455. ISSN: 1435-568X. DOI: 10.1007/BF02134016. URL: https://doi.org/10.1007/BF02134016.

[11] George B. Arfken. *Mathematical Methods for Physicists*. 3rd ed. Elsevier Science, 1985. ISBN: 0120598205.

[12] David Jeffrey Griffiths. *Introduction to electrodynamics*. eng. 3. ed.. Upper Saddle River, N.J.: Prentice Hall, 1999. ISBN: 0-13-805326-x.

[13] Pijush K Kundu. *Fluid mechanics.* 5th ed. / Pijush K. Kundu, Ira M. Cohen, David R. Dowling.. Amsterdam ; London: Academic, 2012. ISBN: 0-12-382101-0.

[14] Claes Johnson. *Numerical solutions of partial differential equations by the finite element method*. Lund: Studentlitteratur, 1987. ISBN: 91-44-25241-2.

[15] Mats G. Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation and Applications*. Springer, 2010. ISBN: 978-3-642-33286-9. DOI: 10.1007/978-3-642-33287-6.

[16] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python: The FEniCS Tutorial I*. Springer, 2017. ISBN: 9783319524610.

[17] Sandip Mazumder. *Numerical Methods for Partial Differential Equations*. Academic Press, 2016. ISBN: 0-12-849894-3.

[18] Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. “Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations”. In: *Applied Numerical Mathematics* 25.2-3 (1997), pp. 151–167. ISSN: 01689274. DOI: 10.1016/S0168-9274(97)00056-1.

[19] Andreas Griewank. “A mathematical view of automatic differentiation”. In: *Acta Numerica* 12.May (2003), pp. 321–398. ISSN: 09624929. DOI: 10.1017/S0962492902000132.

[20] Andreas Griewank. “Who Invented the Reverse Mode of Differentiation?” In: *Documenta Mathematica · Extra Volume ISMP* I (2012), pp. 389–400. ISSN: 1431-0635.

[21] Dimitris C Psichogios and Lyle H Ungar. “A hybrid neural network-first principles approach to process modeling”. In: *AIChE Journal* 11.2 (1992), pp. 337–346. DOI: 10.1016/S0893-6080(98)00005-7. URL: http://onlinelibrary.wiley.com/doi/10.1002/aic.690381003/abstract.

[22] Mark A Kramer, Michael L Thompson, and Phiroz M Bhagat. “Embedding theoretical models in neural networks”. In: *American Control Conference, 1992*. IEEE. 1992, pp. 475–479.

[23] H.-T. Su et al. *Integrating Neural Networks with First Principles Models for Dynamic Modeling*. IFAC, 1992, pp. 327–332. DOI: 10.1016/B978-0-08-041711-0.50054-4. URL: http://dx.doi.org/10.1016/B978-0-08-041711-0.50054-4.

[24] Reinout Romijn et al. “A grey-box modeling approach for the reduction of nonlinear systems”. In: *Journal of Process Control* 18.9 (2008), pp. 906–914. ISSN: 09591524. DOI: 10.1016/j.jprocont.2008.06.007.

[25] S Gupta et al. “Hybrid first-principles neural networks model for column flotation”. In: *AIChE J.* 45.3 (1999), pp. 557–566. ISSN: 00011541. DOI: 10.1002/aic.690450312.

[26] J Schubert et al. “Bioprocess optimization and control - application of hybrid modeling”. In: *J. Biotechnol.* 35.1 (1994), pp. 51–68. ISSN: 01681656. DOI: 10.1016/0168-1656(94)90189-9.

[27] Chenkun Qi et al. “A Grey-Box Distributed Parameter Modeling Approach for a Flexible Manipulator with Nonlinear Dynamics”. In: *IFAC-PapersOnLine* 48.28 (2015), pp. 544–549. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.12.185. URL: http://dx.doi.org/10.1016/j.ifacol.2015.12.185.

[28] H J L van Can et al. “Understanding and applying the extrapolation properties of serial gray-box models”. In: *AIChE Journal* 44.5 (1998), pp. 1071–1089. ISSN: 00011541.

[29] Hua Deng, Han Xiong Li, and Guanrong Chen. “Spectral-approximation-based intelligent modeling for distributed thermal processes”. In: *IEEE Transactions on Control Systems Technology* 13.5 (2005), pp. 686–700. ISSN: 10636536. DOI: 10.1109/TCST.2005.847329.

[30] Yi Liu and Elling W. Jacobsen. “Error Detection and Control in Grey-Box Identification of Distributed Parameter Processes”. In: *IFAC Proceedings Volumes* 37.9 (2004), pp. 841–846. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)31914-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017319146.

[31] Y Liu. “Grey box identification of distributed parameter systems”. PhD thesis. Royal Institute of Technology, 2005.

[32] Michael Hinze. *Optimization with PDE Constraints*. Mathematical Modelling: Theory and Applications, 23. 2009. ISBN: 1-4020-8839-6.

[33] Jens Berg and Kaj Nyström. “Neural network augmented inverse problems for PDEs”. In: (2017), pp. 1–22. arXiv: 1712.09685. URL: http://arxiv.org/abs/1712.09685.

[34] J. Frédéric Bonnans et al. “Numerical optimization: Theoretical and practical aspects”. In: *Numerical Optimization: Theoretical and Practical Aspects* (2006), pp. 1–494. ISSN: 01968092. DOI: 10.1007/978-3-540-35447-5.

[35] Keith James Laidler. *Chemical kinetics*. eng. 3. ed.. New York: Harper & Row, 1987. ISBN: 0-06-043862-2.

[36] T Dupont et al. “The FEniCS project”. In: *Chalmers Finite Element Center Preprint 2003-21, Chalmers University of Technology* (2003).

[37] Martin S Alnaes et al. “The FEniCS Project Version 1.5”. In: *Archive of Numerical Software* 3.100 (2015), pp. 9–23. ISSN: 2197-8263. DOI: 10.11588/ans.2015.100.20553.

[38] Johan Hoffman and Anders Logg. “DOLFIN: Dynamic Object oriented Library for FINite element computation”. In: *Chalmers Finite Element Center Preprint 2002-06, Chalmers University of Technology* (2002).

[39] Anders Logg and Garth N. Wells. “DOLFIN: Automated Finite Element Computing”. In: 37.2 (2011). ISSN: 00983500. DOI: 10.1145/1731022.1731030. arXiv: 1103.6248.

[40] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[41] Patrick E. Farrell et al. “Automated derivation of the adjoint of high-level transient finite element programs”. In: (2012), pp. 1–27. ISSN: 1064-8275. DOI: 10.1137/120873558. arXiv: 1204.5577.

[42] Diederik P Kingma and Jimmy Lei Ba. “Adam: A Method for Stochastic Optimization”. In: (2015), pp. 1–15. arXiv: arXiv:1412.6980v9.

[43] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[44] Kody Law, Andrew Stuart, and Konstantinos Zygalakis. *Data Assimilation: A Mathematical Introduction*. 1st ed. 2015. Vol. 62. Texts in Applied Mathematics. Springer International Publishing, 2015. ISBN: 9783319203249.

Appendix A

Simulating fluid flow

In the two-dimensional reaction system described in section 3.3.3, the advective velocity field corresponds to the motion of a fluid in the domain $\Omega \subset \mathbb{R}^2$. To simulate this flow in the time interval $\mathcal{T}$, the fluid was assumed to be described by the incompressible Navier-Stokes equation

$$\dot{\boldsymbol{w}} + (\boldsymbol{w} \cdot \nabla)\boldsymbol{w} - \nu\nabla^2\boldsymbol{w} = -\nabla p, \tag{A.1}$$

where $p : \mathcal{T} \times \Omega \to \mathbb{R}$ is the kinematic pressure, ν the kinematic viscosity and the velocity field $\boldsymbol{w} : \mathcal{T} \times \Omega \to \mathbb{R}^2$ is divergence free

$$\nabla \cdot \boldsymbol{w} = 0. \tag{A.2}$$

Just as for any partial differential equation, one needs to specify initial and boundary conditions to solve the Navier-Stokes equation. As initial condition it was assumed that the fluid was at rest

$$\boldsymbol{w}(t = 0, \boldsymbol{r}) = \boldsymbol{0}, \quad \boldsymbol{r} \in \Omega. \tag{A.3}$$

To specify the boundary conditions, let $\partial\Omega_{\text{left}}$, $\partial\Omega_{\text{right}}$, $\partial\Omega_{\text{up}}$ and $\partial\Omega_{\text{down}}$ denote the left, right, upper and lower boundaries of the domain. In addition let, $\partial\Omega_{\text{cylinder}}$ denote the boundary of the cylinder. For the upper, lower and cylinder boundaries a no slip-condition was used

$$\boldsymbol{w}(t, \boldsymbol{r}) = \boldsymbol{0}, \quad \boldsymbol{r} \in \partial\Omega_{\text{up}} \cup \partial\Omega_{\text{down}} \cup \partial\Omega_{\text{cylinder}}. \tag{A.4}$$

Fluid was assumed to flow into the domain through the left boundary with a given static velocity profile

$$\boldsymbol{w}(t, \boldsymbol{r}) = \boldsymbol{g}(\boldsymbol{r}) = (g(r_2), 0)^T, \quad \boldsymbol{r} \in \partial\Omega_{\text{left}}. \tag{A.5}$$

where

$$g(r_2) = 4\left(1 - \frac{r_2}{1.5}\right)\frac{r_2}{1.5}, \quad r_2 \in [0, 1.5]. \tag{A.6}$$

The fluid leaves the system through the right boundary, corresponding to a zero pressure condition

$$p(t, \boldsymbol{r}) = 0, \quad \boldsymbol{r} \in \partial\Omega_{\text{right}}. \tag{A.7}$$

The Navier-Stokes equation (equation A.1), the divergence criterion (equation A.2) along with the initial and boundary conditions (equation A.3, A.4, A.5, A.6) defines a boundary value problem which solution corresponds to the desired fluid motion.

To solve this boundary value problem numerically, the finite difference and finite element method was used. The Crank-Nicolson method was used to integrate the boundary value problem in time. It is a finite difference method based on the trapezoidal rule and can be seen as a combination of the forward and backward Euler methods. Denoting $\boldsymbol{w}(n\Delta t, \boldsymbol{r}) = \boldsymbol{w}^n(\boldsymbol{r})$ and $p(n\Delta t, \boldsymbol{r}) = p^n(\boldsymbol{r})$, the Navier-Stokes equation is written as the following difference equation

$$\frac{\boldsymbol{w}^n - \boldsymbol{w}^{n-1}}{\Delta t} + (\overline{\boldsymbol{w}}^n \cdot \nabla)\overline{\boldsymbol{w}}^n - \nu\nabla^2\overline{\boldsymbol{w}}^n = -\nabla p^n. \tag{A.8}$$

where $\overline{\boldsymbol{w}}^n = (\boldsymbol{w}^n + \boldsymbol{w}^{n-1})/2$. We wish to find a weak solution to this equation using the finite element method. The corresponding weak formulation to this equation may be written on residual form as

$$r_{\text{NS}} + r_{\text{div}} = 0 \tag{A.9}$$

where

$$r_{\text{NS}} = \int_\Omega \left(\frac{\boldsymbol{w}^n - \boldsymbol{w}^{n-1}}{\Delta t} + (\overline{\boldsymbol{w}}^n \cdot \nabla)\overline{\boldsymbol{w}}^n + \nabla p^n\right)\cdot\boldsymbol{v} + \nu\nabla\overline{\boldsymbol{w}}^n \cdot \nabla\boldsymbol{v}\,\mathrm{d}r \tag{A.10}$$

and

$$r_{\text{div}} = \int_\Omega (\nabla \cdot \overline{\boldsymbol{w}}^n)\cdot q\,\mathrm{d}r \tag{A.11}$$

where $\boldsymbol{v}$ and q are test functions. Minimizing r_{NS} corresponds to satisfying the Navier-Stokes equation and minimizing r_{div} corresponds to satisfying the divergence criterion. To solve equation (A.11) using the finite element method with piecewise linear basis functions, one must add a stabilizing term to the weak formulation

$$r_{\text{NS}} + r_{\text{div}} + r_{\text{stab}} = 0 \tag{A.12}$$

where

$$r_{\text{stab}} = \int_{\Omega} \delta \left[(\nabla p^n + \nabla \overline{\boldsymbol{w}}^n \cdot \overline{\boldsymbol{w}}^n) \cdot (\nabla q + \nabla \overline{\boldsymbol{w}}^n \cdot \boldsymbol{v}) + (\nabla \cdot \overline{\boldsymbol{w}}^n) \cdot (\nabla \cdot \boldsymbol{v}) \right] \mathrm{d}r. \quad \text{(A.13)}$$

By adding this term and choosing $\delta : \Omega \to \mathbb{R}$ sufficiently small, the stability of the solution can be guaranteed while not significantly modifying the original problem. For the two-dimensioal reaction system $\delta(\boldsymbol{r}) = 0.05h(\boldsymbol{r})$ was used where $h(\boldsymbol{r})$ is the cell diameter of the triangle in the mesh that $\boldsymbol{r}$ belongs to. That is,

$$h(\boldsymbol{r}) = \sup_{\boldsymbol{r}_1, \boldsymbol{r}_2 \in K} \|\boldsymbol{r}_1 - \boldsymbol{r}_2\| \quad \text{such that } \boldsymbol{r} \in K \in \mathcal{K} \quad \text{(A.14)}$$

where $\mathcal{K}$ is the set of triangles in the finite element mesh.

In the finite element method, equation (A.12) corresponds to a large system of non-linear algebraic equations. FEniCS [36, 37] was used through the DOLFIN [38, 39] interface to solve this finite element problem using a Newton method.

Given that equation (A.12) can be solved iteratively starting from $\boldsymbol{w}^0 = \boldsymbol{w}(0, \boldsymbol{r}) = \boldsymbol{0}$ one can generate a time series $\{\boldsymbol{w}^n\}_{n=1}^N$ which can be used as the advective velocity field when solving reaction-advection-diffusion equations. The particular geometry of Ω in the two-dimensional reaction system and boundary conditions described above result in a very characterisitc flow pattern known as a von Kármán vortex street (see figure 3.8).

TRITA EECS-EX-2018:749