UNIVERSITY OF LATVIA

DOCTORAL THESIS

Parameter optimization and pattern recognition for combustion and reaction kinetics applications

Author:
Maksims MARINAKI

Supervisor:
Dr. math. prof. Uldis STRAUTIŅŠ

Department of Mathematics
Faculty of Physics, Mathematics and Optometry

September 14, 2021

UNIVERSITY OF LATVIA

Abstract

Department of Mathematics
Faculty of Physics, Mathematics and Optometry

Doctor of Philosophy

Parameter optimization and pattern recognition for combustion and reaction kinetics applications

by Maksims MARINAKI

In this thesis we consider the simplified combustion models and describe the chemical kinetics via different reaction mechanisms. We solve the resulting problems by the finite element techniques. Then we consider some model reduction techniques and verify the results with the experimental ones by doing the parameter optimization. The leftover patterns of the optimization method are stored as fixed points of the discrete dynamical system in order to be reproduced when the experimental data is modified.

Keywords: combustion, chemical kinetics, mathematical modelling, finite element method, PSO optimization, model reduction, pattern storage.

Acknowledgements

Author would like to acknowledge the financial support from the Latvian Research Cooperation Project of the Latvian Council of Science Nr. 623/2014, the ERAF project Nr. 1.1.1.1/16/A/004 and SAM project Nr. 8.2.2.0/18/A/010.

Author would like to express his sincere gratitude to his thesis supervisor Uldis Strautiņš, project supervisor Harijs Kalis, colleagues Andrejs Reinfelds, Andrejs Cibulis and all the others who supported the accomplishment of the thesis.

Contents

List of Symbols

In order of appearance with several reorderings:

q	scalar quantity (unless stated vectorial)
$\mathbf{v}$	vectorial quantity/column-vector
N	number of species/PDEs/vector-function's components
M	number of chemical reactions
d	number of spatial variables
n	finite dimensional function space's dimension/number of experimental quantities/any natural number
D	mixture molecular diffusivity/ the dimension of the pattern
S	number of particles in a swarm
D_p	number of parameters
L	number of snapshots
K	number of elements in the reduced basis
P	number of patterns
N_E	number of experiments
N_S	number of recalculation stages
N	number of iterations
T	number of triangles
t	time variable/triangle matrix
p	node matrix/any natural number
$j = 1..M$	for j from 1 till M
$\frac{\mathrm{d}u}{\mathrm{d}t},\ u'(t),\ u'$	first derivative
$\frac{\partial u}{\partial t},\ u_t,\ \partial^i u$	partial derivative
$\frac{\mathrm{D}}{\mathrm{D}t}$	total derivative
∇	nabla operator
$\nabla\cdot$	divergence operator

$\mathbf{u}\cdot\mathbf{v},\ (\mathbf{u},\mathbf{v})$	dot product
$\|\cdot\|$	element norm (type specified)
$\|\cdot\|_2$	$\mathbb{R}^n$ element Euclidean norm
$L(\cdot)$	differential operator
$\mathbf{v}^T$	transpose vector/matrix
det	determinant of a matrix
E	unit matrix
$\mathbb{N}$	set of natural numbers
$\mathbb{N}_0$	set of natural numbers and zero
$\mathbb{R}$	set of real numbers
$\mathbb{R}^+$	set of positive real numbers
$\partial\Omega$	set boundary
$\Omega_1 \cup \Omega_2$	set union
$\Omega_1 \cap \Omega_2$	set intersection
$\overline{\Omega}$	set closure
$\Omega_1 \times \Omega_2$	set Cartesian product
$\mathrm{int}\Omega$	set interior
$\emptyset$	empty set
$\inf\Omega$	set infimum
$H,\ \mathcal{H},\ G,\ \mathcal{G}$	function space/Hilbert space
C^∞	space of infinitely many times differentiable functions
C_0^∞	space of infinitely many times differentiable functions with compact support
$H^{-1}(\Omega)$	the dual space
$I,\ I_h$	interpolation operator
$\dim H$	space dimension
span	linear hull
Π_m	space of m-th order polynomials
$\sim U(a,b)$	uniformly distributed random numbers
$\mathrm{tr}A$	trace operator
$\mathbf{e_i}$	i-th unit vector
$\mathrm{diag}[A_1,\ldots,A_n]$	block diagonal matrix
$\vee$	logical OR
$:=$	is defined as
$\Rightarrow$	logical consequence (implication)
$\widehat{x}$	solution representative (snapshot)

x^p	p-th vector (pattern)

Abbreviations:

ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
BCs	Boundary Conditions
FEM	Finite Element Method
PSO	Particle Swarm Optimization
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
DDS	Discrete Dynamical System
PC	Pattern Creation
Prop.	proposition (lemma or theorem)
Def.	definition
Alg.	algorithm
cgs	centimetre–gram–second system of units

Chapter 1

Introduction

The idea to write thesis on this particular topic has been a direct consequence of the collaboration between two research institutes: Institute of Mathematics and Computer Science of University of Latvia (the one where the author's employed) and the Institute of Physics of University of Latvia. In the latter institution the researchers have been widely interested in topics such as an efficient combustion of biomass for a very long time and been facilitating the experiments involving the combustion and gasification of different fuels.

The author has participated in the corresponding collaboration projects arised in this field with the mathematical modelling contribution and decided to write thesis that would serve as a supplementary material for the published works in these projects, containing the results in topics of author's scientific interests only.

In [Mar18b] and [Mar19] the modelling and experimental results have been published for experiments such as straw co-firing with peat or propane.

In [Mar18a] and [Mar16] for similar experiments the impact of electric and magnetic fields have been discussed and corresponding experiments have been facilitated.

In [Vey05] the modern theory and applications of combustion and gasification processes is introduced. More classical theory is found in [Wil85] and [Bor01].

These processes are mostly modelled by systems of partial differential equations (PDEs) and the choice of suitable numerical methods along with the theory of PDEs is to be found in [Hac17], [Ben13], [Ang03].

The optimization of parameters, mainly the technical part of the process itself for any optimization problem found in [BAE05], [Cle06].

When the problem becomes a high dimensional one, the principal component analysis (PCA) comes to rescue and the results compatible with the models to be proposed in this thesis are found in [Wil10], [Gre08]. Another aspect is the pattern recognition out of high dimensional problem in order to minimize number of calling out the costly procedure of obtaining the numerical solution thereof and general theory and applications of recurrent networks can be found in [Cha01].

The main idea of this thesis is to develop the mathematical model that governs the main basic physical processes, occurring in the experiments, facilitated by the team, and at the same time uses the experimental data in order to improve itself by using several modern techniques.
If one wants the concise formulation of thesis' aims, these would be:

1) Find a way how to model chemical reactions with PDEs. The models should be as simple as possible, but in two or three dimensions and the underlying chemistry

should be described thoroughly and the transition between the design of chemical reaction and the design of PDE should be visible.

2) Provide the necessary mathematics for the chosen set of models - the existence and uniqueness results, briefly discuss approximation and choose the numerical methods - universal for the obtained models.

3) Develop concepts of measure between experimental and modelling data and formulate all the necessary problems.

4) Perform several numerical experiments with either solution obtaining process as it is or the parameter optimization with the pattern storage procedures - as proposed in the abstract.

The necessity to introduce the concept of measure between modelling and experimental data in 3) is the main motivation to write this thesis. Along the way the author came up with several techniques of solutions' post-processing and storage due to the repetitive nature of the experiments.

Due to the nature of author's work, the main course of the following narration is going to be aspects of implementation of well-known instruments in applied math along with less-known ones for the sake of reaching goals arising throughout the work. These instruments might be used for different purposes separately but they do form a network in this thesis. Each chapter 2-4 shall represent one of them. Chapter 5 shall represent the network along with the computer implementation. In particular:

In Chapter 2 we catalogue the equations to be solved in further chapters in their dimensional forms along with the main laws of physics and combustion chemistry that underlie in every term of these equations.

In Chapter 3 we build the theory till the point where one freely classifies the partial differential equation and chooses the appropriate and universal methods to solve one numerically.

Chapter 4 is the core of the thesis. Author considers several modern techniques for optimizing the model: whether it is a matter of optimizing the parameters of the parametrized model - making the model 'better' in a way - closer to reality so to speak, or a matter of reducing the dimension of the discretized model or fully reformulating the problem as the pre-stored pattern recognition problem - something of a vast importance when it comes to solve the problem by using the personal computer.

Since nowadays most of the modelling working hours are in front of the computer developing a decent code for necessary computations, Chapter 5 is present and contains all the necessary information on what are the main peculiarities when it comes to code in Matlab software: either create user's own script or uses the graphical user interfaces.

Chapter 6 uses the results from the previous chapters to solve some applied problems. These are the ones, developed in Chapter 2 with parametrization and source terms described in details for several different situations. The total scheme on how to effectively solve the resulting problems in the area of applications chosen derived in this thesis is implemented for each case and explicit step explanations with references on results in previous chapters is provided.

The subdivision onto the aforementioned chapters is more or less natural when it comes up to thorough description of each tool to be used. Now each model is to be analysed by sticking to one and the same framework which means all the methods described in each chapter combined. The order of the actions corresponds to the order of the chapters in the thesis and is something that is going to be discussed in the beginning of each chapter.

All the codes used in the text are created by the author using the MATLAB software. All the narrative is the result of the consistent work throughout the years of PhD studies, teaching natural sciences and the participation in various projects and in many cases, some well-known proofs are recreated and interpreted by the author without straightforwardly using any literature, however, since the act of recreation is still implying the fact that some pieces of bibliography were once used, we provide references and discuss their values in prefaces of the corresponding topics. Whereas a well-known algorithm or a widely-known proof is used, but its representation is beyond the scope of this thesis, the author provides the reference links to the items of bibliography as well.

In the end we formulate the conclusions on what has been done, provide the list of literature and list of publications and conferences the author has been involved in, see chapter 7 and appendices A, B.

Chapter 2

Combustion models

As stated in introduction, this chapter takes into account main laws of physics, describing the combustion process in the combustion chamber; formulates them in their differential form and makes a lot of simplifications. Moreover, models of different complexities are constructed: different levels of complexity used for different purposes in chapter 6.

Modelling stage, where we address the laws of physics in order to formulate the concise mathematical model is a case of calling on the gathered common knowledge throughout the history of science. Thus we seek the mathematical formulations of the empirically proven laws. These are widely used and to be found in literature along with the necessary derivations.

Whilst participating in projects, author had to analyse several books containing mathematical models of combustion process and reacting flows. [Vey05] along with its author's lecture notes gave the best insight on topic and is the most recent and decent one containing all the modern theory and applications. More classical ones such as [Wil85] and [Bor01] were of a great value for achieving goals of our researches as well.

Main purpose is to sort the detailed chemistry descriptions out of these models presented in specialized literature rather than convection and diffusion aspects since the latter is modelled mathematically in a more or less traditional way. Hence the main stress is on reaction kinetics.

The idea is to pin down the terms of the combustion equations in their dimensional form. Then to make simplifications and to neglect some terms. The goal of the chapter is to obtain the parametrized equations to be solved for temperature T and the mass fraction of the k-th specie Y_k for each k in the reacting flow in the combustion chamber and verify the dimensions. Then the solvability and the solution process discussion will start in the next chapter 3.

2.1 Equations solved for mass fractions

The goal is to obtain the simplifications of the general mass fraction equation, which simplifies the diffusion and convection terms whilst describes the underlying chemistry quite thoroughly. When we say that, we seek the form where one takes a certain reaction (or the reaction mechanism) and easily plugs it in the model.

Following the monographs stated in the introduction of this chapter we shall present the species equation for each specie in its general form. Then make simplifications in order to make it solvable in terms of theory presented in chapter 3.

We use the notation Y_k for the mass fraction of k-th specie in the mixture and define it as a partial density and mixture of gases density ratio $Y_k := \dfrac{\rho_k}{\rho}$.

We want to solve nothing but a continuity equation for Y_k. So it is necessary to set it up first.

Def. 2.1.1. *We call an equation*

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = \sigma \tag{2.1}$$

a *continuity equation* in differential form for quantity q. Here ρ is the density of a quantity q as an amount of q per cubic meters, $\mathbf{j}$ represents a flux of q and σ is the generation of quantity q as an amount of that quantity per cubic meter per second. Intuitively it is positive when the quantity q gets generated, negative when it gets driven out and zero when the quantity gets conserved. In the latter case we might as well call it a *conservation equation*.

In case of combustion the mass of each particular specie is not conserved, might disappear and might also be created for each particular k, thus we model it with non-zero σ, whilst the total mass clearly has to be conserved, where σ is neglected.

The mass continuity equation in fluid dynamics is defined by considering ρ as a density of mass and setting $\mathbf{j} = \rho\mathbf{u}$:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.2}$$

where ρ is the density as an amount of kilograms per cubic meter and $\mathbf{u}$ represents the velocity field of the reacting flow. It is clearly modelled with a zero valued generation rate σ due to conservation of mass. We wish to write down the equations that are true for each individual specie Y_k, $k = 1..N$, defined above and observe the consistency rules such that (2.2) holds at the same time.

For each partial density $\rho_k = \rho \cdot Y_k$ we can write down the continuity equation (2.1) by setting non-zero generation rate (usually denoted by $\dot{\omega}_k$ also referred to as a *reaction term*) and flux $\mathbf{j} = \rho_k(\mathbf{u} + \mathbf{v}_k)$, where $\mathbf{v}_k$ is the diffusion velocity of k-th specie:

$$\frac{\partial}{\partial t}(\rho Y_k) + \nabla \cdot \Big(\rho Y_k(\mathbf{u} + \mathbf{v}_k)\Big) = \dot{\omega}_k. \tag{2.3}$$

Along with an obvious consistency relation

$$\sum_{k=1}^{N} Y_k = \frac{\sum_{k=1}^{N} \rho_k}{\rho} = 1, \tag{2.4}$$

by summing up N equations (2.3) and comparing it with the continuity equation (2.2), we obtain two more:

$$\sum_{k=1}^{N} \dot{\omega}_k = 0 \; - \; \text{due to conservation of mass} \tag{2.5}$$

and

$$\sum_{k=1}^{N} Y_k \mathbf{v}_k = 0 \; - \; \text{the consistency in divergence terms.} \tag{2.6}$$

Regarding the last one, usually the extension of Fick's law does the job [Bet16; Bro13]. This is the first simplification we would like to make:

$$\mathbf{v}_k Y_k = -D \nabla Y_k, \tag{2.7}$$

where D is the molecular diffusivity in the mixture. This relation is the most common simplification which would allow us to consider more traditional models in sense of mathematical theory to be discussed in chapter 3 and at the same time it does satisfy the consistency relations.

The equation (2.3) now can be reformulated by using this simplification and the total derivative notation. We split the linear divergence operator into two parts:

$$\frac{\partial}{\partial t}(\rho Y_k) + \nabla \cdot (\rho Y_k \mathbf{u}) = \nabla \cdot (\rho D \nabla Y_k) + \dot{\omega}_k, \tag{2.8}$$

and simplify the left hand side:

$$\frac{\partial}{\partial t}(\rho Y_k) + \nabla \cdot (\rho Y_k \mathbf{u}) = \rho \frac{\partial Y_k}{\partial t} + \rho \mathbf{u} \cdot \nabla Y_k + Y_k \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right]. \tag{2.9}$$

The square brackets in the end is our mass continuity equation (2.2), thus these terms are getting neglected. The remaining part divided by ρ is something we call a *total derivative* and use the notation

$$\frac{\mathrm{D} Y_k}{\mathrm{D} t} := \frac{\partial Y_k}{\partial t} + \mathbf{u} \cdot \nabla Y_k. \tag{2.10}$$

Thus the species equation for the specie $Y_k,\ k = 1..N$, has its simplified form:

$$\rho \frac{\mathrm{D} Y_k}{\mathrm{D} t} = \nabla \cdot (\rho D \nabla Y_k) + \dot{\omega}_k. \tag{2.11}$$

In the next section we pin down the reaction term $\dot{\omega}_k$.

2.2 The reaction term

So far the equation (2.11) reads that the mass fraction of the k-th specie changes in time due to convection and diffusion processes plus the rate of change due to chemical reactions occurring in the flow. We wish to write down the reaction term $\dot{\omega}_k$ as a sort of a template which holds for each reaction mechanism and to be fulfilled with the parameters that are different for each mechanism. We split it into different parts first and present it in one row afterwards.

In the previous section 2.1 we've numbered species such that they're indexed by k and it changes from 1 till N (total number of species considered in the model is N and an individual k-th specie is considered in the equation).

At the same time we have to number the chemical reactions involved in the reacting flow. We do it in the same manner and number the reaction with an index j and the total number of reactions is M and adopt the empirical laws commonly used to model the reacting flow [Vey05].

The term $\dot{\omega}_k$ is the sum over all reactions:

$$\sum_{j=1}^{M} \dot{\omega}_{kj}, \tag{2.12}$$

where $\dot{\omega}_{kj}$ is expressed as $\dot{\omega}_{kj} = q_j w_k \nu_{kj}$.

The explanation of this form starts with the ν_{kj} multiplier. This is constructed as following: we consider M reactions in our model, through which N species are reacting. We consider forward reactions here and each j-th reaction, $j = 1..M$, usually has the form

$$\sum_{k=1}^{N} \nu'_{kj} M_k \to \sum_{k=1}^{N} \nu''_{kj} M_k,\ j = 1..M, \tag{2.13}$$

where ν_{kj} is the number of moles of the k-th specie in the j-th reaction, and M_k is the nomenclature of the corresponding specie.

Then we denote $\nu_{kj} := \nu''_{kj} - \nu'_{kj}$. So we have only one stoichiometric coefficient describing the reaction instead of two, which can be either positive or negative. The w_k is the molecular weight of specie k, while the q_j is the multiplier that covers the *Arrhenius law.*

The Arrhenius law lies in the core of the reaction term and states the compound exponential or quasipolynomial rate with respect to temperature and polynomial with respect to mass fractions. For the forward reaction it has the following form:

$$q_j = K_j \prod_{k=1}^{N} N_k^{\nu'_{kj}}, \tag{2.14}$$

$$K_j = A_j e^{\frac{-E_j}{RT}} T^{\beta_j}. \tag{2.15}$$

Here N_k is the concentration of the specie k as the amount of moles per cubic meter. The reactivity parameters for the reaction j are the pre-exponential factor A_j, the activation energy E_j and the temperature exponent β_j and are either to be determined from the existing tables or their values would be estimated and afterwards optimized which is the course of chapter 4. R is the ideal gas constant.

There's an explicit temperature dependence and an implicit mass fraction dependence in (2.14),(2.15). The concentration - mass fraction relation to use is

$$N_k = \frac{\rho Y_k}{w_k}. \tag{2.16}$$

Now we are ready for the final formula of the reaction term $\dot{\omega}_k$. In each k- th mass fraction equation we construct the parametrized source term by considering the reaction mechanism (2.13):

$$\dot{\omega}_k = \sum_{j=1}^{M} A_j e^{\frac{-E_j}{RT}} T^{\beta_j} \prod_{k=1}^{N} N_k^{\nu'_{kj}} w_k \nu_{kj}. \tag{2.17}$$

Thus we have a framework where one takes the reaction mechanism as a list of formulas (2.13), constructs the matrix ν by doing the arithmetics to a stoichiometry coefficients and in a very convenient manner plugs everything into formula (2.17) along with the parameters to be discussed in chapter 4.

2.3 Temperature equation

One of the simplified forms of temperature equation for deflagrations is constructed similarly to (2.3) by setting up the continuity equation for energy flow and reformulating it for temperature:

$$\rho c_p \frac{\mathrm{D}T}{\mathrm{D}t} = \nabla \cdot (\lambda \nabla T) + \omega_T', \tag{2.18}$$

where ρ, c_p and λ are respectively mass density, heat capacity and the thermal conductivity.

The flux term, called a heat flux, similarly to (2.7) is given by $\lambda \nabla T$ and is due to Fourier's law [Bet16; Bro13].

The source term for this equation has the form [Vey05]

$$\omega_T' = -\sum_{k=1}^{N} (\triangle H_{f,k} + h_{s,k})\dot{\omega}_k, \tag{2.19}$$

where $\triangle H_{f,k}$ stands for the enthalpy of formation and the $h_{s,k}$ for the sensible enthalpy.

One can see that the source term consists of the aforementioned enthalpies and the same source term $\dot{\omega}_k$ as in (2.3) and ends up with enthalpies of formation only if uses heat capacity for mixture c_p and not for the individual specie $c_{p,k}$.

The heat capacity we shall consider independent of the temperature as well. The source term we divide into two sums:

$$\omega_T' = -\sum_{k=1}^{N} \triangle H_{f,k}\dot{\omega}_k - \sum_{k=1}^{N} h_{s,k}\dot{\omega}_k. \tag{2.20}$$

Under the assumption that the heat capacity doesn't depend on the temperature

$$h_{s,k} = \int_{T_0}^{T} c_{pk} \mathrm{d}T = (T - T_0)c_{pk}, \tag{2.21}$$

or the species

$$\sum_{k=1}^{N} h_{s,k}\dot{\omega}_k = (T - T_0)c_p \sum_{k=1}^{N} \dot{\omega}_k = 0, \tag{2.22}$$

due to consistency relation (2.5), the only enthalpies we consider are the enthalpies of formation.

The enthalpy of formation is to be found in the table for each specie in the reacting mixture, which concludes the parametrization of the equation, since the $\dot{\omega}_k$ multiplier is the same as in the species equations. Thus the weighted sum of the right hand sides from all N species equations is considered as the right hand side of the temperature equation.

At the end of the day we are going to work with the reaction-diffusion partial differential equation system (2.11), (2.18).

The next chapter 3 provides classical results in classification and well-posedness of these equations; in other words the results of fields such as mathematical physics and numerics for problems of mathematical physics we are going to address there.

Chapter 3

Partial differential equations and the FEM

The description of physical laws in their differential forms in the previous chapter 2 led to parametrized system of partial differential equations (2.11), (2.18). The present chapter provides the definitions and the common classification of the partial differential equations (PDEs). Then we address the aspects of the solution space discretization which leads to an equivalence of the discretized problem to a problem of linear algebra. These definitions and proofs of the results are more or less common for every book in the corresponding field - the author tried to make it as laconic as possible by including the necessary stuff only and jump to the procedures part since the thesis is mainly concerned on applications aspect.

These aspects are included in one and the same present chapter since every chapter in the paper represents one step of processing the input information: in this case the equations written down in their differential form will be reformulated as a problem of linear algebra or series of such problems. Series of such problems can be solved subsequently by using a personal computer. The scheme used in the next chapter will use these solutions as a function of parameter inputs as a part of a more superior procedure which calls out the solution procedure many times.

But in order to obtain the desired solution, we consider the necessary theory first. Author mostly inspired by books [Fou03], [Hac17], [Ben13], [Ang03]. All well-known results to be presented are mostly taken from them. When the technical aspects of the particular proof require more theory to be brought to the table these are omitted and to be found in the literature presented. But in many cases the proofs recreated by the author are to be present since the technical peculiarities and the intermediate results thereof are of importance when building the necessary theory.

3.1 The classification of PDEs

We shall start with the classification of PDEs which is required for choosing the right tool to solve it either analytically or numerically. Classification provided here built in correspondence with the classification of second order curves in the analytical geometry.

We consider the second order linear partial differential equation

$$Lu = -\nabla \cdot (A\nabla u) + \mathbf{b}^T \nabla u + cu = f, \tag{3.1}$$

where the unknown function $u : \Omega \to \mathbb{R}$, $A \in \mathbb{R}^{d\times d}$ and elements of vector $\mathbf{b}$ along with c and f - are bounded functions $\Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$.

The classification of (3.1) depends only on the *principal part* of the equation, i.e $-\nabla \cdot (A\nabla u)$.

Def. 3.1.1. *We call an equation (3.1) elliptic at point* $\mathbf{x}$ *if all the eigenvalues of* A $\lambda_i > 0,\ i = 1..d$ *or* $\lambda_i < 0,\ i = 1..d$ *at* $\mathbf{x}$.
We call an equation (3.1) parabolic at point $\mathbf{x}$ *if all but one eigenvalues of* A $\lambda_i > 0,\ i = 1..d-1$ *or* $\lambda_i < 0,\ i = 1..d-1$ *whilst* $\lambda_d = 0$ *at* $\mathbf{x}$.

Def. 3.1.2. *We call an equation (3.1) elliptic/parabolic in* Ω *if it's elliptic/parabolic* $\forall \mathbf{x} \in \Omega$.

Def. 3.1.3. *We call a differential operator in (3.1) uniformly elliptic if* $\xi^T A \xi \geq \gamma \|\xi\|^2$, $\gamma > 0,\ \xi \in \mathbb{R}^d$. *The matrix* A *in this case is called uniformly positive definite.*

Remark 3.1.1. *The equations (2.11) and (2.18) in two or three spatial variables are elliptic if made stationary, linearised and decoupled since* A *is in form* $cE,\ c \in \mathbb{R}^+$ *- a positive function of positive physical constants.*

Remark 3.1.2. *The equations (2.11) and (2.18) in two or three spatial and one time variable are parabolic when linearised and decoupled since there's no second order derivatives with respect to time involved, thus* A *has a zero-valued row and column.*

3.2 Boundary and initial value problems

Well-posedeness of a problem containing equation (3.1) depends on an appropriate choice of boundary conditions. Several types of boundary conditions are going to be considered. Further definitions will deal with a boundary $\partial\Omega$ of a certain degree of smoothness - a *Lipshitz boundary*.

Def. 3.2.1. *We say that a bounded domain* $\Omega \subset \mathbb{R}^d$ *has a Lipshitz boundary, if*

$$\forall x \in \partial\Omega \ \ \exists B(x) \subset O_i :$$

$$O_i \cap \Omega = O_i \cap \Omega_i, \tag{3.2}$$

where $B(x)$ *is a sphere,* $O_i,\ i = 1..m$ *- open sets,*

$$\Omega_i = \{(x_1, x_2) \in \mathbb{R}^d : x_1 \in \mathbb{R}^{d-1}, x_2 \in \mathbb{R}, x_2 < \phi_i(x_1)\}, \tag{3.3}$$

$\phi_i,\ i = 1..m$ *- Lipshitz functions:*

$$\exists L > 0 \ \ \forall x, y \in \partial\Omega \ \ |\phi_i(x) - \phi_i(y)| \leq L|x - y|. \tag{3.4}$$

Now that we have a boundary, the procedure is to define two types of boundary conditions separately and afterwards mix them together in order to obtain the model-type problem suitable for applications to be discussed further on.

Def. 3.2.2. *We call a problem*

$$\begin{cases} Lu = -\nabla \cdot (A\nabla u) + \mathbf{b}^T \nabla u + cu = f, \Omega, \\ u = 0, \ \ \partial\Omega \end{cases} \tag{3.5}$$

a Dirichlet boundary value problem (BVP) for an elliptic PDE.

Def. 3.2.3. *We call a problem*

$$\begin{cases} Lu = -\nabla \cdot (A\nabla u) + \mathbf{b}^T \nabla u + cu = f, \Omega, \\ A\nabla u \cdot \widehat{n} = h, \ \ \partial\Omega \end{cases} \tag{3.6}$$

a Neumann boundary value problem (BVP) for an elliptic PDE.

Here $\widehat{n}$ is an outer normal of Ω, h is a scalar function of spatial variables.

The Dirichlet boundary conditions here and in further claims are considered homogeneous. Any inhomogeneity can be eliminated by redefining the right hand side of the problem by considering the auxiliary function.

For one and the same problem one can prescribe Dirichlet and Neumann boundary conditions on different portions of the boundary Γ_D and Γ_N respectively such that $\Gamma_D \cap \Gamma_N = \emptyset$ and $\bar{\Gamma}_D \cup \bar{\Gamma}_N = \partial\Omega$.

Def. 3.2.4. *We call a problem*

$$\begin{cases} Lu = -\nabla \cdot (A\nabla u) + \mathbf{b}^T \nabla u + cu = f, \Omega, \\ u = 0, \quad \Gamma_D, \\ A\nabla u \cdot \widehat{n} = h, \quad \Gamma_N \end{cases} \tag{3.7}$$

a mixed boundary value problem (BVP) for an elliptic PDE.

Instead of considering zero row and column in matrix A due to nature of applications of parabolic equations such as heat and diffusion equations the first order derivative with respect to time is written down as a separate term and not a part of the gradient term.

In similar fashion we define the problem of finding the solution $u(t, \mathbf{x})$ of parabolic equation with mixed boundary conditions.

Def. 3.2.5. *We call a problem*

$$\begin{cases} \dfrac{\partial u}{\partial t} + Lu = f, (0, T) \times \Omega, \\ u(t, \mathbf{x}) = 0, \quad (0, T) \times \Gamma_D, \\ A\nabla u \cdot \widehat{n} = h, \quad (0, T) \times \Gamma_N, \\ u(0, \mathbf{x}) = u_0, \quad \Omega \end{cases} \tag{3.8}$$

an initial boundary value problem for parabolic PDE with mixed boundary conditions.

Such approach lets us build the theory in such fashion that we start with the elliptic part and regarding the time-dependence the problem gets split into sequence of such problems.

In theory of classical solutions the smoothness required for the solution is such as second order derivatives exist in the interior or the domain Ω. Numerics we are going to apply usually deal with solutions of less smoothness requirements. In the next section we address the weak formulations of boundary value problems (3.7), (3.8) that seek the solutions with weakened requirements and discuss the solvability afterwards.

3.3 Weak formulations

For elliptic PDEs the aim is to reformulate the BVP (3.7) in operator equation form

$$a(u, v) = f(v) \tag{3.9}$$

in appropriate function spaces for u and v elements.

For parabolic PDEs the aim is to reformulate the IBVP (3.8) in operator equation form

$$\frac{\mathrm{d}}{\mathrm{d}t}(u,v) + a(u,v) = f(v) \tag{3.10}$$

in appropriate function spaces for u and v elements.

In this section we set and examine the nature of right and left hand sides of these equations.

Def. 3.3.1. *We call a mapping*
$a(u,v): \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ *a bilinear form, if the following linearity axioms hold:*

$\forall \alpha \in \mathbb{R}\ \forall u, v, u_1, v_1 \in \mathcal{H}:$

1. $a(\alpha u, v) = a(u, \alpha v) = \alpha a(u,v)$,
2. $a(u + u_1, v) = a(u,v) + a(u_1, v)$,
3. $a(u, v + v_1) = a(u,v) + a(u, v_1)$.

Def. 3.3.2. *We call a mapping* $f(v): \mathcal{H} \to \mathbb{R}$ *a linear functional, if the following linearity axioms hold:*

$\forall \alpha \in \mathbb{R}\ \forall v, v_1 \in \mathcal{H}$*:*

1. $f(\alpha v) = \alpha f(v)$,
2. $f(v + v_1) = f(v) + f(v_1)$.

First we obtain the weak formulation for BVP (3.7).

Def. 3.3.3. *We call a problem:*

Find the function $u \in \mathcal{H}$, *that satisfies*

$$a(u,v) = f(v) \quad \forall v \in \mathcal{G}, \tag{3.11}$$

where $a(u,v)$ *- bilinear form,* $f(v)$ *- linear functional,*
a weak formulation of a problem (3.7).

There are several existence requirements on $a(u,v)$ and $f(v)$. We ask u and v along with their partial derivatives to be square integrable.

Def. 3.3.4. *For* $1 \leq p < \infty$, *the space* $L^p(\Omega)$ *is defined as follows:*

$$L^p(\Omega) = \{u : \int_\Omega |u(x)|^p \mathrm{d}\mathbf{x} < \infty\}. \tag{3.12}$$

We define a norm in space $L^p(\Omega)$:

$$\|u\|_{L^p} := \|u\|_{0,p} := (\int_\Omega |u(x)|^p \mathrm{d}\mathbf{x})^{\frac{1}{p}}. \tag{3.13}$$

The space $L^2(\Omega)$ is a Hilbert space when equipped with a scalar product

$$(u,v)_{L^2} := (u,v)_0 := \int_\Omega uv \mathrm{d}\mathbf{x}. \tag{3.14}$$

Def. 3.3.5. *We call a function $w \in L^2(\Omega)$ a weak derivative of a function $u \in L^2(\Omega)$ if*

$$(u, \partial^\alpha v)_0 = (-1)^{|\alpha|}(w, v)_0 \text{ for each test function } v \in C_0^\infty(\Omega), \tag{3.15}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$ *- multi-index,* $|\alpha| := \sum_{i=1}^{n} \alpha_i$, $\partial^\alpha v := \frac{\partial^{\alpha_1}}{\partial_{x_1}^{\alpha_1}} \cdots \frac{\partial^{\alpha_n}}{\partial_{x_n}^{\alpha_n}}$.

The defined derivatives of u exist inside of the integration operator with respect to test functions. The other concise notation to use is $D^\alpha u$. Piecewise linear functions we are about to consider when dealing with finite element procedures are weakly but not strongly differentiable.

Def. 3.3.6. *For $m > 0$ and $p \geq 1$, Sobolev space $W^{m,p}(\Omega)$ is defined as follows:*

$$W^{m,p}(\Omega) = \{u : u \in L^p(\Omega),\ D^\alpha u \in L^p(\Omega),\ |\alpha| \leq m\}. \tag{3.16}$$

The space $W^{m,p}(\Omega)$ is a normed spaced when equipped with a norm:

$$\|u\|_{m,p} := \Big(\sum_{|\alpha| \leq m} \|D^\alpha u\|_{0,p}^p \Big)^{\frac{1}{p}}.$$

The space $W^{m,2}(\Omega)$ is Hilbert space, when equipped with a scalar product

$$(u, v)_m := \sum_{|\alpha| \leq m} (D^\alpha u, D^\alpha v)_0, \tag{3.17}$$

and the notation often used is $H^m(\Omega) := W^{m,2}(\Omega)$.

Def. 3.3.7. *The space $H^1(\Omega)$ defined as follows:*

$$H^1(\Omega) = \{u : u \in W^{1,2}(\Omega)\}. \tag{3.18}$$

The space H^1 contains the square integrable functions along with their square integrable weak derivatives.
Now we are ready for the technical part of obtaining the weak formulation of (3.7). Both sides of its equation

$$-\nabla \cdot (A \nabla u) + \mathbf{b}^T \nabla u + cu = f$$

get multiplied by the sufficiently smooth test function v. The necessary order of smoothness of u and v to be determined later.

The resulting expression

$$-\nabla \cdot (A \nabla u) v + \mathbf{b}^T \nabla u v + cuv = vf \tag{3.19}$$

we integrate over Ω:

$$-\int_\Omega \nabla \cdot (A \nabla u) v \mathrm{d}\mathbf{x} + \int_\Omega \mathbf{b}^T \nabla u v \mathrm{d}\mathbf{x} + \int_\Omega cuv \mathrm{d}\mathbf{x} = \int_\Omega f v \mathrm{d}\mathbf{x}. \tag{3.20}$$

We apply partial integration for the principal part $-\int_\Omega \nabla \cdot (A \nabla u) v \mathrm{d}\mathbf{x}$ such that there's no longer second order smoothness requirement.

For partial integration the divergence theorem is required in order reduce the smoothness order.

Prop. 3.3.1. *(Divergence theorem. For proof see [Pfe12].) For the domain $\Omega \subset \mathbb{R}^d$ with a boundary $\partial\Omega$, fulfilling requirements (3.2.1) and the continuously differentiable vector field* $\mathbf{u}$ *the following holds:*

$$\int\limits_{\Omega} \nabla \cdot \mathbf{u} \, \mathrm{d}\mathbf{x} = \int\limits_{\partial\Omega} (\mathbf{u}^T \widehat{n}) \mathrm{d}\sigma,$$

where $\widehat{n}$ is the outer normal of Ω.

In order to extend the notion of function's restriction to the boundary for Sobolev functions, we need the following result.

Prop. 3.3.2. *(Trace mapping theorem. For proof see [Fou03].) For the domain $\Omega \subset \mathbb{R}^d$ with a boundary $\partial\Omega$, fulfilling requirements (3.2.1) there exists a unique continuous linear mapping*
$\gamma_0 : \; H^1(\Omega) \to L^2(\partial\Omega)$ such that for all $u \in C^1(\overline{\Omega})$ the following holds:

$$\gamma_0(u) = u|_{\partial\Omega}.$$

Prop. 3.3.3. *(Partial integration of the principal part)*
Under the conditions (3.18) for ∇u and v and a boundary $\partial\Omega$, fulfilling requirements (3.2.1), we can integrate the principal part of (3.1) as follows:

$$-\int\limits_{\Omega} \nabla \cdot (A\nabla u) v \mathrm{d}\mathbf{x} = -\int\limits_{\partial\Omega} v(A\nabla u) n \mathrm{d}\sigma + \int\limits_{\Omega} \nabla v^T A \nabla u \mathrm{d}\mathbf{x}. \tag{3.21}$$

Proof. Partial integration procedure can be performed due to the following differential relation:

$$\nabla \cdot (vA\nabla u) = \nabla \cdot \begin{pmatrix} v(a_{11}u_{x_1} + \cdots + a_{1d}u_{x_d}) \\ \vdots \\ v(a_{d1}u_{x_1} + \cdots + a_{dd}u_{x_d}) \end{pmatrix} =$$

$$= (v(a_{11}u_{x_1} + \cdots + a_{1n}u_{x_d}))_{x_1} + \cdots + (v(a_{d1}u_{x_1} + \cdots + a_{dd}u_{x_d}))_{x_d} =$$
$$= v_{x_1}(a_{11}u_{x_1} + \cdots + a_{1d}u_{x_d}) + \cdots + v_{x_d}(a_{d1}u_{x_1} + \cdots + a_{dd}u_{x_d}) +$$
$$+ v(a_{11}u_{x_1x_1} + \cdots + a_{1d}u_{x_dx_1}) + \cdots + v(a_{d1}u_{x_1x_d} \cdots + a_{dd}u_{x_dx_d}) =$$
$$= \nabla v^T A \nabla u + \nabla \cdot (A\nabla u) v.$$

Thus we can integrate

$$-\int\limits_{\Omega} \nabla \cdot (A\nabla u) v \mathrm{d}\mathbf{x} = -\int\limits_{\Omega} \nabla \cdot (vA\nabla u) \mathrm{d}\mathbf{x} + \int\limits_{\Omega} \nabla v^T A \nabla u \mathrm{d}\mathbf{x}. \tag{3.22}$$

The divergence theorem (3.3.1) transforms the first summand in the right-hand side:

$$\int\limits_{\Omega} \nabla \cdot (vA\nabla u) \mathrm{d}\mathbf{x} = \int\limits_{\partial\Omega} v(A\nabla u) n \mathrm{d}\sigma. \tag{3.23}$$

This yields the result.

□

The conditions for weak formulation (3.7) thus can be smoothened to $u, \; v \in H^1$.

The operator equation for the weak formulation in problems (3.5), (3.6), (3.7) now reads:

$$-\int\limits_{\partial\Omega} v(A\nabla u)n d\sigma + \int\limits_{\Omega} \nabla v^T A\nabla u \mathrm{d}\mathbf{x} + \int\limits_{\Omega} b^T \nabla u v \mathrm{d}\mathbf{x} + \int\limits_{\Omega} cuv \mathrm{d}\mathbf{x} = \int\limits_{\Omega} fv \mathrm{d}\mathbf{x}. \tag{3.24}$$

For Dirichlet problem (3.5)

$$\begin{aligned} a(u,v) &= \int\limits_{\Omega} \nabla v^T A\nabla u \mathrm{d}\mathbf{x} + \int\limits_{\Omega} b^T \nabla u v \mathrm{d}\mathbf{x} + \int\limits_{\Omega} cuv \mathrm{d}\mathbf{x}, \\ f(v) &= \int\limits_{\Omega} fv \mathrm{d}\mathbf{x}. \end{aligned} \tag{3.25}$$

For Dirichlet problems we want boundary values to be incorporated into space definition. Therefore we define the space for functions u and v for the homogeneous Dirichlet problem.

Def. 3.3.8. *The space $H_0^1(\Omega)$ is defined as follows:*

$$H_0^1(\Omega) = \{u : u \in H^1(\Omega);\ \ \gamma_0(u) = 0, \partial\Omega\}.$$

We are all set for the weak formulation for Dirichlet problem:

Find function $u \in H_0^1(\Omega)$ that satisfies the equation

$$a(u,v) = f(v)\ \ \forall v \in H_0^1, \tag{3.26}$$

where $a(u,v)$, $f(v)$ are in form (3.25).

Boundary conditions for Neumann problems (3.6) should be incorporated in the right-hand functional:

$$\begin{aligned} a(u,v) &= \int\limits_{\Omega} \nabla v^T A\nabla u \mathrm{d}\mathbf{x} + \int\limits_{\Omega} b^T \nabla u v \mathrm{d}\mathbf{x} + \int\limits_{\Omega} cuv \mathrm{d}\mathbf{x}, \\ f(v) &= \int\limits_{\Omega} fv \mathrm{d}\mathbf{x} + \int\limits_{\partial\Omega} h\gamma_0(v) \mathrm{d}\sigma. \end{aligned} \tag{3.27}$$

Thus the weak formulation for Neumann problem (3.6) reads:

Find function $u \in H^1(\Omega)$, that satisfies the equation

$$a(u,v) = f(v)\ \ \forall v \in H^1, \tag{3.28}$$

where $a(u,v)$, $f(v)$ are in form (3.27).

Now we are ready to mix two together and in similar fashion obtain the weak formulation for the mixed problem (3.7).

The boundary conditions for mixed problems should be incorporated in the right-hand functional and into function space:

$$\begin{aligned} a(u,v) &= \int\limits_{\Omega} \nabla v^T A\nabla u \mathrm{d}\mathbf{x} + \int\limits_{\Omega} b^T \nabla u v \mathrm{d}\mathbf{x} + \int\limits_{\Omega} cuv \mathrm{d}\mathbf{x}, \\ f(v) &= \int\limits_{\Omega} fv \mathrm{d}\mathbf{x} + \int\limits_{\Gamma_N} h\gamma_0(v) \mathrm{d}\sigma. \end{aligned} \tag{3.29}$$

The space H for the solution and the test functions is defined as follows:

$$H = \{u \in H^1(\Omega),\ \gamma_0(u) = 0, \Gamma_D\}. \tag{3.30}$$

Thus the weak formulation for the mixed boundary value problem (3.7) reads:

Find function $u \in H$, that satisfies the equation

$$a(u,v) = f(v) \ \ \forall v \in H, \tag{3.31}$$

where $a(u,v)$, $f(v)$ are in form (3.29).

Since it is a mixture of two, further on we are going to refer to the *mixed problem*. The other reason for considering it is that exactly this type of boundary conditions is going to be considered mostly in the applications section of this thesis.

For the initial boundary value problems (3.8) the solution space should be redefined first.

Def. 3.3.9. *For Banach space H we define the space $L^2(0,T;H)$ as the space of functions $u: \ (0,T) \to H$ equipped with the norm $\|u\|_{L^2(0,T;H)} := \left(\int\limits_0^T \|u(t)\|_H^2 \mathrm{d}t\right)^{\frac{1}{2}}$.*

In order to interpret the weak time derivative we need an appropriate definition as well. This one is similar to (3.3.5) but with respect to $L^2(0,T;H)$. We present the following weak formulations in the corresponding function spaces. More on choice of these spaces e.g in [Ang03].

Def. 3.3.10. *We call a function $w \in L^2(0,T;H^{-1})$ a weak derivative of a function $u \in L^2(0,T;H)$ if*

$$\int\limits_0^T u(t)v'(t)\mathrm{d}t = -\int\limits_0^T w(t)v(t)\mathrm{d}t \ \forall v \in C_0^\infty(0,T). \tag{3.32}$$

Def. 3.3.11. *The space $W(0,T;H)$ is defined as follows:*

$$\{u: \ u \in L^2(0,T;H), \ u' \in L^2(0,T;H^{-1})\}. \tag{3.33}$$

As for the operator equation for the weak formulation of (3.8) the procedure is similar to (3.20) with the new term corresponding to the time derivative:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int\limits_\Omega uv\mathrm{d}\mathbf{x} - \int\limits_\Omega \nabla \cdot (A\nabla u)v\mathrm{d}\mathbf{x} + \int\limits_\Omega \mathbf{b}^T\nabla uv\mathrm{d}\mathbf{x} + \int\limits_\Omega cuv\mathrm{d}\mathbf{x} = \int\limits_\Omega fv\mathrm{d}\mathbf{x}, \tag{3.34}$$

and after the partial integration procedure due to (3.21) one gets the weak formulation for the initial boundary value problem with mixed boundary conditions (3.8).

Find function $u \in W(0,T;H)$, that satisfies the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}(u,v)_0 + a(u,v) = f(v) \ \ \forall v \in H, \tag{3.35}$$

along with the initial condition

$$u(0) = u_0 \in L^2(\Omega), \tag{3.36}$$

where $a(u,v)$, $f(v)$ are in form (3.29).

With the reduced smoothness requirements the solution existence results from classical theory don't apply anymore, thus we have to consider different results that propose the existence and uniqueness of solution for weakly formulated problems.

3.4 Existence and uniqueness results

There are several lemmas that guarantee the solution existence and uniqueness for problems (3.31) and (3.35), (3.36) . The aim of this section is to present the appropriate results found in literature that state that several properties of objects used in (3.31) hold and afterwards to check these properties.

Def. 3.4.1. *Given H a Hilbert space. Defined in $H \times H$ bilinear form $a(u,v) : H \times H \to \mathbb{R}$ is called continuous, if*

$$\exists \lambda > 0,\ \forall u,v \in H\ |a(u,v)| \leq \lambda \|u\|_H \|v\|_H. \tag{3.37}$$

Def. 3.4.2. *Given H a Hilbert space. Defined in $H \times H$ bilinear form $a(u,v) : H \times H \to \mathbb{R}$ is called coercive or H-elliptic, if*

$$\exists \gamma > 0,\ \forall u \in H\ \gamma \|u\|_H^2 \leq a(u,u). \tag{3.38}$$

Def. 3.4.3. *Defined in a normed space H linear functional $f(v) : H \to \mathbb{R}$ is called continuous, if*

$$\exists M > 0,\ \forall v \in H\ |f(v)| \leq M \|v\|_H. \tag{3.39}$$

The next results guarantee the existence and uniqueness for the solutions of weakly-formulated problems (3.31) and (3.35),(3.36). The conditions to fulfill rely on properties of the 'principal' part of the bilinear form.

Prop. 3.4.1. *(Lax-Milgram theorem. For proof see [Hac17], [Ben13], [Ang03].)*
Given Hilbert space H. If defined in $H \times H$ bilinear form
$a(u,v) : H \times H \to \mathbb{R}$ is continuous and H-elliptic,
and a linear functional $f(v) : H \to \mathbb{R}$ - continuous, then the weakly-formulated problem (3.31) has a unique solution.

Prop. 3.4.2. *Problem (3.31) fulfills the conditions of the Lax-Milgram theorem 3.4.1 assuming $L(\cdot)$ is uniformly elliptic.*

Proof. Several times we are going to use the Cauchy-Schwarz inequality in the appropriate Hilbert space H:

$$\forall\ u,v\ \in H:\ \ (u,v)_H \leq \|u\|_H \cdot \|v\|_H. \tag{3.40}$$

Since we need to take account of our first derivatives, the notation $\partial^i u := \dfrac{\partial u}{\partial x_i}$ is going to be widely used as well.

- The first property is the continuity of the bilinear form.
  In order to show that $|a(u,v)| \leq \lambda \|u\|_H \|v\|_H$, the bilinear form's expression $a(u,v)$ may first be rewritten in form:

$$a(u,v) = \sum_{0 \leq i,j \leq d} (\tilde{a}_{ij} \cdot \partial^i u, \partial^j v)_{L_2}, \tag{3.41}$$

where the new matrix $\tilde{A}$ has the structure $\begin{pmatrix} c & 0 & \dots & 0 \\ b_1 & a_{11} & \dots & a_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ b_d & a_{d1} & \dots & a_{dd} \end{pmatrix}$. Elements of vector $\mathbf{b}$ and the function c are bounded in Ω. By using the Cauchy-Schwarz and letting go the tilde notation, one can write down

$$|a(u,v)| = |\sum_{0\le i,j\le d} (a_{ij}\cdot \partial^i u, \partial^j v)_{L_2}| \le \lambda \sum_{0\le i,j\le d} \|\partial^i u\|_{L^2}\|\partial^j v\|_{L^2} \le$$
$$\le \lambda \sqrt{\int_\Omega (|u|^2+|u_{x_1}|^2+\dots+|u_{x_d}|^2)\mathrm{d}\mathbf{x}} \sqrt{\int_\Omega (|v|^2+|v_{x_1}|^2+\dots+|v_{x_d}|^2)\mathrm{d}\mathbf{x}} =$$
$$= \lambda \|u\|_{H^1}\|v\|_{H^1}, \tag{3.42}$$

where $\lambda = \max_{0\le i,j\le d}\{C_{ij} : |a_{ij}(\mathbf{x})| \le C_{ij}\}$.

- The second property is the H-ellipticity of the bilinear form $a(u,v)$.
  We have to show, that $a(u,u) \ge \gamma\|u\|^2_{H^1}$. We use the uniform ellipticity (3.1.3) $\xi^T A\xi \ge \gamma\|\xi\|^2$, $\gamma > 0$, $\xi \in \mathbb{R}^d$ and assume that $\tilde{A}$ is uniformly positive definite as well. Then we have

$$a(u,u) = \sum_{0\le i,j\le d} (a_{ij}\cdot\partial^i u, \partial^j u)_{L_2} \ge \gamma \int_\Omega (|u|^2+|u_{x_1}|^2+\dots+|u_{x_d}|^2)dx = \gamma\|u\|^2_{H^1},$$

  where $\gamma = \min_{0\le i\le d}\{C : |a_{ii}(\mathbf{x})| \ge C\}$.

- The last property is the continuity of the linear functional $f(v)$.
  When all boundary conditions are homogeneous, all we have to do is apply the Cauchy-Schwarz:

$$|f(v)| = |(f,v)_{L^2}| \le \|f\|_{L^2}\|v\|_{L^2} \le M\|v\|_{H^1}. \tag{3.43}$$

  For problems with mixed boundary conditions (3.31) the functional has an extra term:

$$f(v) = \int_\Omega f v \mathrm{d}x + \int_{\partial\Omega} h v \mathrm{d}x. \tag{3.44}$$

  Bearing in mind the trace-mapping theorem 3.3.2, one can use the relation between the norms:
  for Lipshitz boundaries $\partial\Omega$ 3.2.1

$$\exists K > 0 : \forall v \in H^1(\overline{\Omega}) \quad \|v\|_{L^2(\partial\Omega)} \le K\|v\|_{H^1(\Omega)}. \tag{3.45}$$

  Using the estimate for homogeneous case along with Cauchy-Schwarz one obtains:

$$|f(v)| \le |(f,v)_{L^2(\Omega)}| + |(h,v)_{L^2(\partial\Omega)}| \le$$
$$\le M\|v\|_{H^1(\Omega)} + \|h\|_{L^2(\partial\Omega)}\|v\|_{L^2(\partial\Omega)} = M\|v\|_{H^1(\Omega)} + K_1\|v\|_{L^2(\partial\Omega)} \le$$
$$\le M\|v\|_{H^1(\Omega)} + K_1\cdot K\|v\|_{H^1(\Omega)} \le \max\{M, K_1\cdot K\}\|v\|_{H^1(\Omega)}. \tag{3.46}$$

This concludes the proof. □

The last technique in the latter proof can be applied whenever one wants to use the continuity argument for boundary integral term in the right-hand side functional. However in the applications section the Neumann boundary condition is going to be homogeneous thus in further results the function h is going to be neglected.

There is a result with the conditions similar to ones that the Lax-Milgram theorem uses and it serves as an extension to parabolic case. It shall serve us as an existence-uniqueness result thereof.

Prop. 3.4.3. *Given Hilbert space H. If defined in $H \times H$ bilinear form*
$a(u, v) : H \times H \to \mathbb{R}$ is continuous and H-elliptic,
$f \in L^2(0, T; L^2(\Omega))$ and $u_0 \in L^2(\Omega)$, then the weakly-formulated problem (3.35), (3.36) has a unique solution.

Proof. For problem (3.35), (3.36) with homogeneous boundary conditions one first obtains the smoothing estimate of the solution's norm

$$\|u(t)\|_{L^2} \leq \|u_0\|_{L^2} \cdot e^{-\gamma t} + \int_0^t \|f(s)\|_{L^2} \cdot e^{-\gamma(t-s)} \mathrm{d}s, \; t \in (0, T), \tag{3.47}$$

where γ is the coerciveness constant for our bilinear form (3.4.2).
The argument is shown by setting $v = u(t)$.
In that case

$$(u'(t), u(t))_{L^2} = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u(t)\|^2_{L^2} = \|u(t)\|_{L^2}\frac{\mathrm{d}}{\mathrm{d}t}\|u(t)\|_{L^2}. \tag{3.48}$$

The H-ellipticity of the bilinear form gives

$$a(u(t), u(t)) \geq \gamma\|u(t)\|^2_H \geq \gamma\|u(t)\|^2_{L^2} \tag{3.49}$$

and the Cauchy-Schwarz for linear functional

$$(f(t), u(t))_{L^2} \leq \|f(t)\|_{L^2}\|u(t)\|_{L^2}. \tag{3.50}$$

One obtains

$$\|u(t)\|_{L^2}\frac{\mathrm{d}}{\mathrm{d}t}\|u(t)\|_{L^2} + \gamma\|u(t)\|^2_{L^2} \leq \|f(t)\|_{L^2}\|u(t)\|_{L^2}. \tag{3.51}$$

For non-trivial solutions after dividing by $\|u(t)\|_{L^2}$ and multiplying by $e^{\gamma t}$ one gets

$$e^{\gamma t}\frac{\mathrm{d}}{\mathrm{d}t}\|u(t)\|_{L^2} + \gamma e^{\gamma t}\|u(t)\|_{L^2} = \frac{\mathrm{d}}{\mathrm{d}t}(e^{\gamma t}\|u(t)\|_{L^2}) \leq e^{\gamma t}\|f(t)\|_{L^2}. \tag{3.52}$$

The result is now integrated from 0 to t:

$$e^{\gamma t}\|u(t)\|_{L^2} - 1 \cdot \|u(0)\|_{L^2} \leq \int_0^t \|f(s)\|_{L^2} \cdot e^{\gamma s} \mathrm{d}s. \tag{3.53}$$

In order to obtain the desired result (3.47) one multiplies the relation by $e^{-\gamma t}$ and applies the initial condition $u(0) = u_0$.

The uniqueness can now be proven by contradiction. Suppose there are two solutions u_1 and u_2. The function $w = u_1 - u_2$ solves the homogeneous problem with

$f \equiv 0$ and $u_0 \equiv 0$. By applying the smoothing estimate (3.47) one now concludes that $\|w\|_{L^2} = 0$ and thus $u_1 = u_2$.

$\square$

Since we have constructed the parabolic differential operator in form $u_t + Lu$, the checking process of properties for objects involved in the variational formulation for parabolic equations is somewhat similar to elliptic case bearing in mind that the smoothness requirement for the initial condition has to hold as well. In fact, it is the same for the bilinear form and time-independent linear form in the right-hand sides of the equations resulting from (2.11), (2.18) we are going to consider in the applications section.

Since there exists only one solution for each problem (3.31) and (3.35),(3.36) we are all set to jump on to the techniques of obtaining the numerical solution thereof. The next section shall address the discretization aspects of these problems.

3.5 Finite dimensional problems and the FEM

The approach behind all the techniques we are about to mention is the function space discretization. If we consider the finite amount of basis elements, our weak formulations (3.31) and (3.35),(3.36) would become equivalent to problems of linear algebra. This approach is closely related to *Galerkin method* and would involve full matrices in case if our basis functions each is defined and mainly non-zero in the whole interior of the geometrical space. In order to obtain sparse matrices for convenience in numerics, one can combine the solution and test function space-discretization approach with the geometrical space discretization such that the basis functions each have a local support i.e each is zero in most of the interior of the geometrical space and locally non-zero. This approach is closely related to the *Finite element method*.

We now are going to work with the so-called discrete weak formulation and obtain the existence and uniqueness results thereof. We will present the general formulation when one does not at all or specifies only some of the basis functions whilst the formulation stays universal and leaves space for wide variety of such. The nature of chosen basis shall appear in the coding chapter 5 and shall be explained there in addition.

We are going to work in finite dimensional spaces that are subspaces of the infinite dimensional ones

$$H_h \subset H^1, \; W(0,T;H_h) \subset W(0,T;H^1). \tag{3.54}$$

If n is the dimension of H_h then we clearly have $n \sim \frac{1}{h}$ and when $h \to 0$, $n = \dim H_h \to \infty$.

The discrete analogue to our weak formulation (3.31) shall look like this:

Find the function $u_h \in H_h(\Omega)$, that satisfies the equation

$$a(u_h, v_h) = f(v_h) \;\; \forall v_h \in H_h(\Omega). \tag{3.55}$$

The discrete analogue to our weak formulation (3.35), (3.36) shall look like this:

Find the function $u_h(t) \in W(0,T;H_h(\Omega))$ such that $u_h(0) = I_h u_0 \in H_h$, that satisfies the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t), v_h)_0 + a(u_h(t), v_h) = f(v_h) \ \ \forall v_h \in H_h(\Omega). \tag{3.56}$$

We need the representation and the existence-uniqueness result for the discrete case.

Prop. 3.5.1. *The formulation (3.55) is equivalent to linear system $A\xi = f$; $\xi, f \in \mathbb{R}^N$. The formulation (3.56) is equivalent to initial-value problem for the ODE system*
$\dot{\xi}(t) + B\xi(t) = f(t)$, $\xi(0) = \xi_0$; $\xi, f : \mathbb{R} \to \mathbb{R}^N$, $\xi_0 \in \mathbb{R}^N$.
If the bilinear form is continuous and H-elliptic then each problem has a unique solution.

Proof. We are going to use the notion of basis in our finite dimensional subspace H_h as a set of n linearly independent elements $S := \{\phi_1, \dots, \phi_n\}$ with the property

$$\forall \phi \in H_h \ \exists \beta \in \mathbb{R}^n : \ \phi = \sum_{i=1}^{n} \beta_i \phi_i. \tag{3.57}$$

In an n-dimensional space H_h we choose a basis S and consider n equations. For every $i, i \in \{1 \dots n\}$ we substitute our test function $v_h \in H_h$ by $\phi_i \in H_h$:

$$a(u_h, \phi_i) = f(\phi_i) \ \forall i \in \{1 \dots n\}. \tag{3.58}$$

The element u_h then is represented by $\sum_{j=1}^{n} \xi_j \phi_j$:

$$a(\sum_{j=1}^{n} \xi_j \phi_j, \phi_i) = f(\phi_i) \ \forall i \in \{1 \dots n\}. \tag{3.59}$$

Due to bilinearity and linearity of $a(u_h, v_h)$ and $f(v_h)$:

$$\sum_{j=1}^{n} a(\phi_j, \phi_i) \cdot \xi_j = f(\phi_i) \ \forall i \in \{1 \dots n\}. \tag{3.60}$$

This $n \times n$ linear system can be rewritten in matrix form:

$$A\xi = f, \tag{3.61}$$

where $a_{ij} = a(\phi_j, \phi_i)$, $f_i = f(\phi_i)$.

The uniqueness of the finite dimensional weak formulation (3.55) is the direct consequence of the infinite dimensional case. This can be also shown by proving the fact that the system matrix is not singular.

Since our bilinear form $a(u, v)$ is H^1-elliptic in $H_h \subset H^1$, the $a(u_h, v_h)$ H_h-ellipticity yields that the matrix A is positive definite: $\xi^T A \xi > 0 \ \forall \xi \in \mathbb{R}^n \setminus \{0\}$, where $A = (a(\phi_j, \phi_i))$, $1 \le i, j \le n$, since

$$\xi^T A \xi = \sum_{i,j=1}^{n} a(\phi_j, \phi_i) \xi_j \xi_i \tag{3.62}$$

and due to bilinearity and H^1-ellipticity for non-trivial solutions:

$$\sum_{i,j=1}^{n} a(\phi_j, \phi_i) \xi_j \xi_i = a(\sum_{j=1}^{n} \xi_j \phi_j, \sum_{i=1}^{n} \xi_i \phi_i) = a(u_h, u_h) \ge \gamma \|u_h\|^2_{H^1} > 0. \tag{3.63}$$

The positive definite matrix has all positive eigenvalues and thus $\det A > 0$ yielding that the linear system has a unique solution.

As for the parabolic case $A\xi = f$ transforms to $A\xi(t) = f(t)$ by assuming the solution and the data are time-dependent but there is one more term $\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t), v_h)_0$ to add to the left-hand side.

If we choose the same space discretization for test functions

$$H_h = \operatorname{span}\{\phi_1, \dots, \phi_n\}, \tag{3.64}$$

the solution's representation would be

$$u_h(t) = \sum_{j=1}^{n} \xi_j(t)\phi_j. \tag{3.65}$$

For each linear equation in the system the test function is again $v_h = \phi_i,\ i = 1..N$.
Thus, due to linearity, for $i = 1..n$ we have

$$\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t), v_h)_0 = \frac{\mathrm{d}}{\mathrm{d}t}(\sum_{j=1}^{n} \xi_j(t)\phi_j, \phi_i)_0 = \frac{\mathrm{d}}{\mathrm{d}t}\sum_{j=1}^{n}(\phi_j, \phi_i)_0 \cdot \xi_j(t) = M\dot{\xi}(t), \tag{3.66}$$

where $M_{ij} = (\phi_j, \phi_i)_0$ - is commonly called a *mass matrix*.
Thus we have a formulation

$$M\dot{\xi}(t) + A\xi(t) = f(t). \tag{3.67}$$

The initial condition $u_h(0) = I_h u_0 \in L^2(\Omega)$ should be interpreted in weak sense. In order to write it in form $\xi(0) = \xi_0 \in \mathbb{R}^n$, by using the same argument, as for all t, one rewrites for the initial value

$$\left(\sum_{j=1}^{N} \xi_j(0)\phi_j, \phi_i\right)_0 = M\xi(0), \tag{3.68}$$

which should be equal to $(u_0, \phi_i) =: \varphi$. Thus in order to get coefficients ξ_0 one has to solve the linear system

$$M\xi(0) = \varphi. \tag{3.69}$$

By showing that the mass matrix is invertible, one gets the desired formulation. The mass matrix is again positive definite since

$$\forall \xi \in \mathbb{R}^n \setminus \{0\} \quad \xi^T M \xi = \sum_{i,j=1}^{n} (\phi_j, \phi_i)_0 \xi_j \xi_i = (\sum_{j=1}^{n} \xi_j \phi_j, \sum_{i=1}^{n} \xi_i \phi_i)_0 = (u_h, u_h)_0 = \|u_h\|_{L^2}^2 > 0 \tag{3.70}$$

and thus it is not singular.
The uniqueness of the finite dimensional weak formulation (3.56) is the direct consequence of the uniqueness result for infinite dimensional case.

□

The system matrix $B = -M^{-1}A$ for parabolic case is the multiplication of two positive definite matrices with the minus sign and thus is negative definite, real parts of all the eigenvalues are strictly negative and thus the system is asymptotically stable. This property is of importance in our applications section when one would like to connect the stationary and non-stationary case.

The nature of objects involved in our formulation is now clear. In order to examine them in details, we are left with a choice of our basis functions. If we want our matrices to be sparse, this choice is to be closely related to a partition of domains. Here we have arrived to a definition of the *finite element*.

This means that the basis functions have to be defined locally, on Ω subsets or *elements*. The plan is to subdivide Ω onto such elements. In this case the number of elements is closely related to the number of basis functions.

Another requirement is that the basis functions are piecewise linear or polynomial, which would guarantee the existence of quantities $a_{ij} = a(\phi_j, \phi_i)$, $f_i = f(\phi_i)$, since these types of functions are square integrable and possess their weak derivatives, necessary to compute these quantities.

One of the opinions is that the main aspect of the method nevertheless is the subdivision of the domain onto subdomains or the introduction of the *triangulation* for the domain $\overline{\Omega}$.

Therefore we arrive to the definition of triangulation.

Def. 3.5.1. *The bounded domain's $\Omega \subset \mathbb{R}^n$ subdivision $\mathcal{T}_h$ onto elements $T \subset \mathbb{R}^n$, $T \in \mathcal{T}_h$ is called the triangulation of set Ω if the following holds:*

1. $\overline{\Omega} = \bigcup\limits_{T \in \mathcal{T}_h} T$.
2. $\forall T \in \mathcal{T}_h$*: T - closed set, $int(T) \neq \emptyset$, ∂T - Lipshitz boundary according to definition (3.2.1).*
3. $\forall T_i, T_j \in \mathcal{T}_h,\ i \neq j:\ int(T_i) \cap int(T_j) = \emptyset$.

   For polyhedral subdivisions we require an admissibility condition:

4. $\forall T_i \in \mathcal{T}_h$ *: each T_i face is either other element's T_j face or it belongs to set's Ω boundary $\partial\Omega$.*

We shall use the polynomial finite elements hence the definition.

Def. 3.5.2. *The space*

$$S_h^{(m)} := \{u \in C^0(\bar{\Omega}) :\ u|_{T_i} \in \Pi_m \ \forall T_i \in \mathcal{T}_h\} \tag{3.71}$$

is called the m-th order polynomial finite element space with respect to triangulation $\mathcal{T}_h$.

One of the most popular special cases is the *Linear Lagrangian elements*:

$$S_h^{(1)} := \{u \in C^0(\bar{\Omega}) :\ u|_{T_i} \in \Pi_1 \ \forall T_i \in \mathcal{T}_h\}. \tag{3.72}$$

This one is to be used in our applications chapter 6.

One way to refer to a finite element is to refer to a $\mathcal{T}_h$ element, but there exists more general definition, which takes into account several method's aspects.

Def. 3.5.3. *We call a finite element in $\mathbb{R}^n$ a triplet (T, P, Σ), where:*

1. *T - closed set, $int(T) \neq \emptyset$, ∂T - Lipshitz boundary.*
2. *P - defined on T finite dimensional function $T \to \mathbb{R}$ space.*
3. *Σ - finite set of linearly independent functionals φ_i with the unisolvency property: $\forall \alpha_i \in \mathbb{R}\ \exists p \in P: \quad \varphi_i(p) = \alpha_i,\ i = 1..\dim P$.*

Thus, the finite element is characterized by the basis functions $p \in P$, defined on this element, and the degrees of freedom φ_i.

The property of the set Σ gives us a chance to unambiguously set the linear or polynomial functions, defined on element T.

In case when these degrees of freedom are in form $p \to p(a_i)$, where a_i are the mesh points, the task of finding the basis functions is a simple interpolation task.

We need to say some words on quality of approximation.

It is clear that the quality of approximation depends on all the aspects in our abstract definition: quality of subdivision of domain and the degree of polynomial basis functions.

But there's a result, which would say that the Galerkin solution u_h itself, without saying anything about the nature of shape functions, is as close to an analytical solution as any other function from H_h.

Prop. 3.5.2. *(Céa's lemma [Hac17], [Ben13], [Ang03].)*

Under the conditions that the bilinear form a is continuous and H^1-elliptic, the solutions of problems (3.31) and (3.55) u and u_h do satisfy the following:

$$\|u - u_h\|_{H^1} \leq c \inf_{v_h \in H_h} \|u - v_h\|_{H^1}, \tag{3.73}$$

where $c > 0$.

Proof. We work in the subspace $H_h \subset H^1$.

Our weak formulation (3.31) reads $a(u, v_h) = F(v_h)$.

Due to linearity we've got

$$a(u, v_h) - a(u_h, v_h) = a(u - u_h, v_h) = F(v_h) - F(v_h) = 0\ \forall v_h \in H_h. \tag{3.74}$$

v_h is an arbitrary H_h element and by setting $v_h = u_h$, one obtains

$$a(u - u_h, u_h) = 0,\ a(u - u_h, v_h) = 0 \tag{3.75}$$

and by subtracting them both from $a(u - u_h, u)$, due to linearity one gets

$$a(u - u_h, u - u_h) = a(u - u_h, u - v_h). \tag{3.76}$$

Hence by using the continuity and H^1- ellipticity with the corresponding continuity and coerciveness constants λ and γ as in (3.4.1), (3.4.2), the estimate holds:

$$a(u - u_h, u - u_h) \geq \gamma \|u - u_h\|^2_{H^1} \tag{3.77}$$

and

$$a(u - u_h, u - v_h) \leq \lambda \|u - u_h\|_{H^1} \|u - v_h\|_{H^1} \tag{3.78}$$

and therefore

$$\gamma \|u - u_h\|^2_{H^1} \leq \lambda \|u - u_h\|_{H^1} \|u - v_h\|_{H^1} \tag{3.79}$$

and for non-zero differences one gets

$$\|u - u_h\|_{H^1} \leq \frac{\lambda}{\gamma} \|u - v_h\|_{H^1} \forall v_h \in H^1, \tag{3.80}$$

or in other terms

$$\|u - u_h\|_{H^1} \leq c \inf_{v_h \in H_h} \|u - v_h\|_{H^1}, \tag{3.81}$$

where $c := \dfrac{\lambda}{\gamma} \neq 0,\ c > 0$.

This concludes the proof. $\square$

This basically means that the approximation u_h is the best amongst all the candidates $v_h \in H_h$. The result is the key to the approximation and convergence results in different norms which we do not present here as they are beyond the scope of this thesis but one can find them in literature such as [Hac17], [Ben13], [Ang03].

What we can conclude from all of the above is that there is always choice on the partition of geometrical domain, type of basis functions and the degrees of freedom. Along with the choice of Linear Lagrangian elements (3.72), the methods for solving the resulting linear system and the linear ODE system are going to be chosen depending on application. Some more comments on that in the next section.

Now we have some techniques on how to obtain a certain-type numerical solution of problems (3.31) and (3.35), (3.36) - weak formulations of single linear differential equation, either elliptic or parabolic. Since the problems (2.11), (2.18) about to be considered in our applications chapters shall be coupled in a system and would involve non-linear source terms, the next section shall consider these aspects in order to conclude the chapter.

3.6 Treatment of the non-linear time-dependent system

There is a reason why we considered decoupled equations first and now about to jump to a coupled system properties.

First of all, some algorithms would use the decoupling whereas a single equation to be solved at each step of the solution process.

Secondly, we'll show that the weak formulation's extension to system form would require a simple summation and thus the existence-uniqueness proofs would use the results obtained for the bilinear form and linear functional formed out of single equation.

Thus we do a somewhat straightforward extension to several dimensions for our functions to become the vector-valued ones and discuss the underlying aspects along the way.

The capital N notation for the number of components for our vector functions is going to be used and is in consistence with the number of species from chapter 2 and thus the number of the equations and unknown functions in the system (actually $N + 1$ if we solve for N mass fractions and 1 temperature, but due to conservation of mass, one mass fraction can be excluded from the system thus the consistency still holds).

We define the vector functions $\mathbb{R}^d \to \mathbb{R}^N$ as $\mathbf{u} := (u_1, \ldots, u_N)^T$, $\mathbf{v} := (v_1, \ldots, v_N)^T$. The components are either elements of $L^2(\Omega)$ or $H^1(\Omega)$ thus $[L^2(\Omega)]^N$ and $[H^1(\Omega)]^N$ are still Hilbert spaces equipped with scalar product $(\mathbf{u}, \mathbf{v})_k = \sum\limits_{i=1}^{N} (u_i, v_i)_k$ and the induced norm $[(\mathbf{u}, \mathbf{u})_k]^{\frac{1}{2}}$ for $k = 0, 1$.

The construction of our elliptic or parabolic PDE system is in accordance with models (2.11), (2.18) from chapter 2 and has a somewhat diagonal structure. Thus, more generally they look like this:

$$\begin{cases} L_1 u_1 = f_1(u_1, \ldots, u_N), \Omega, \\ \ldots \\ L_n u_n = f_n(u_1, \ldots, u_N), \Omega, \\ u_i(\mathbf{x}) = 0, \quad \Gamma_{D_i}, \\ \nabla u_i \cdot \widehat{n} = 0, \quad \Gamma_{N_i}, \end{cases} \tag{3.82}$$

$i = 1..N$.
The notation L_i stands for indexed elliptic differential operators that have the same construction as a single operator L for each i as in (3.1).

Similar construction is obtained for parabolic case. This one is going to be in correspondence with the system (2.11), (2.18) from chapter 2, thus, again the diagonal structure and only one time derivative for each equation.

$$\begin{cases} \dfrac{\partial u_i}{\partial t} + L_{ii} u_i = f_i(u_1, \ldots, u_N), \quad (0, T) \times \Omega, \\ u_i(t, \mathbf{x}) = 0, \quad (0, T) \times \Gamma_{D_i}, \\ \nabla u \cdot \widehat{n} = 0, \quad (0, T) \times \Gamma_{N_i}, \\ u_i(0, \mathbf{x}) = u_{0,i}, \quad \Omega, \end{cases} \tag{3.83}$$

$i = 1..N$.
In order to obtain the weak formulations one scalarly multiplies both sides of our system with a test vector function $\mathbf{v}$. We arrive to weak formulations similar to (3.29):

$$a(\mathbf{u}, \mathbf{v}) = \int\limits_{\Omega} [L_1 u_1 v_1 + \cdots + L_n u_n v_n] \mathrm{d}\mathbf{x}, \; f(\mathbf{v}) = \int\limits_{\Omega} [f_1 v_1 + \cdots + f_n v_n] \mathrm{d}\mathbf{x}. \tag{3.84}$$

Partial integration to be applied to all principal parts similarly to (3.21). We assume that the right-hand side function is linearised and more on that below in this chapter.

The coerciveness and continuity would again guarantee the solvability and the uniqueness of solution for weak formulations of (3.82) and (3.83) (if the vector-valued initial condition is also in $[L^2(\Omega)]^N$) due to the results (3.4.1) - (3.4.3). Thus we formulate the following proposition.

Prop. 3.6.1. *The bilinear form in (3.84) is $[H^1(\Omega)]^N$ continuous and coercive, the linear form in (3.84) is $[H^1(\Omega)]^N$ continuous.*

Proof. The proof is analogous to the case with a single unknown function (Prop. 3.4.2). Sobolev norms are constructed in a way such that the integral has new summands out of the derivatives and vector function norms introduce even more summands in the same fashion.

Now we check whether the three properties are true in the same way as (3.4.2).

- The continuity of the bilinear form.
  In order to show that $|a(u,v)| \le \lambda \|u\|_H \|v\|_H$, the bilinear form's expression $a(u,v)$ may first be rewritten in form same as in (3.4.2) with the new index $k = 1..N$ taking into account the vector function component numeration. The coefficients are stored in the 3-dimensional object a_{ijk}:

  $$a(u,v) = \sum_{\substack{0\le i,j\le d \\ 1\le k\le N}} (a_{ijk} \cdot \partial^i u_k, \partial^j v_k)_{L_2}.$$

  Again, by applying the Cauchy-Schwarz, one can write down

  $$|a(u,v)| = |\sum_{\substack{0\le i,j\le d \\ 1\le k\le N}} (a_{ijk} \cdot \partial^i u_k, \partial^j v_k)_{L_2}| \le \lambda \sum_{\substack{0\le i,j\le d \\ 1\le k\le N}} \|\partial^i u_k\|_{L^2} \|\partial^j v_k\|_{L^2} \le$$

  $$\le \lambda \sqrt{\int_\Omega (|u_1|^2 + |u_{1_{x_1}}|^2 + \cdots + |u_{1_{x_d}}|^2 + \cdots + |u_N|^2 + |u_{N_{x_1}}|^2 + \cdots + |u_{N_{x_d}}|^2) d\mathbf{x}} \cdot$$

  $$\cdot \sqrt{\int_\Omega (|v_1|^2 + |v_{1_{x_1}}|^2 + \cdots + |v_{1_{x_d}}|^2 + \cdots + |v_N|^2 + |v_{N_{x_1}}|^2 + \cdots + |v_{N_{x_d}}|^2) d\mathbf{x}} =$$

  $$= \lambda \|u\|_{[H^1]^N} \|v\|_{[H^1]^N}, \tag{3.85}$$

  where $\lambda = \max\limits_{\substack{1\le k\le N \\ 0\le i,j\le d}} \{C_{ijk} : |a_{ijk}(\mathbf{x})| \le C_{ijk}\}$.

- The second property is the H-ellipticity of the bilinear form $a(u,v)$.
  We have to show, that

  $$a(u,u) \ge \gamma \|u\|^2_{[H^1]^N}. \tag{3.86}$$

  For each vector-function component k one uses the coerciveness result from (3.4.2):

  $$a_k(u_k,u_k) \ge \gamma_k \|u_k\|^2_{H^1}, \tag{3.87}$$

  where $\gamma_k = \min\limits_{0\le i\le d} \{C : |a_{iik}(\mathbf{x})| \ge C\}$.
  Since $a(u,u) = \sum\limits_{1\le k\le N} a_k(u_k,u_k)$, we've got the estimate

  $$a(u,u) \ge \gamma(\|u_1\|^2_{H^1} + \cdots + \|u_N\|^2_{H^1}) = \gamma \|u\|^2_{[H^1]^N}, \tag{3.88}$$

  where $\gamma = \min\limits_{1\le k\le N} \{|\gamma_k|\}$.

- The last property is the continuity of the linear functional $f(v)$.
  As we'd agreed, we only consider the homogeneous case - the one that is going to be considered in the applications chapters. Inhomogeneous case had been proven in (3.4.2) for a single function and the extension to the vector functions is straightforward. We apply the Cauchy-Schwarz as usual along with the fact that each component of the vector function f is bounded:

$$|f(v)| = \sum_{1\le k\le N} |(f_k, v_k)_{L^2}| \le \sum_{1\le k\le N} \|f_k\|_{L^2} \|v_k\|_{L^2} \le \sum_{1\le k\le N} M_k \|v_k\|_{H^1} \le M \|v\|_{[H^1]^N}, \tag{3.89}$$

where $M = \|(M_1, \ldots, M_N)^T\|_2$.

□

In the similar fashion, since the uniqueness is proven, we can jump to the discretization aspects. These are again a slight generalization of what has already been done.

The choice of test vector-functions for the application section is the following:

$$\{v_1\mathbf{e}_1, v_2\mathbf{e}_1, \ldots, v_N\mathbf{e}_1, v_1\mathbf{e}_2, v_2\mathbf{e}_2, \ldots, v_N\mathbf{e}_2 \ldots, v_N\mathbf{e}_N\}. \tag{3.90}$$

This leads to larger stiffness matrix that has a block-diagonal structure:

$$\mathrm{diag}[A_1, \ldots A_N]. \tag{3.91}$$

The system is again solvable in linear algebra sense since each block has the origin already discussed for single unknown function and the uniqueness proven in (3.4.2). The block-diagonal matrix constructed out stiffness matrices for each individual elliptic differential operator L_i is still positive definite. The linearization would add extra non-zero elements, which doesn't affect the positive-definiteness. The nature of these extra elements and how to decompose the non-linear problem onto sequence of linear problems is what we are going to discuss till the end of this chapter.

Here we address the notion of linearization, which naturally would come together with the time discretization. In case if the problem is given linear or time-independent, we only consider the special one-step case of what we are going to consider.

All our results so far were obtained for linear PDEs or systems of PDEs. Some notes were made that we consider the linearised equations. The problems from the chapter 2 are non-linear. Such approach is due to the fact that the linearisation theory would transform the non-linear problem onto sequence of linear problems, thus the solution process for uniquely solvable linear problems is to be called several times.

Recall the objects from the proposition 3.5.1. The discretization of a weak formulation for elliptic BVP (3.31) leads to system $A\xi = g$.

We have intentionally changed f to g so we get f in a final result.

The discretization of a weak formulation for parabolic IBVP (3.35), (3.36) leads to ODE system $M\dot{\xi} + A\xi = g$.

If one takes a closer look, the non-linearity appears in the right hand side only and for models from chapter 2 is time independent.

If we want to extend the theory up to a non-linear right hand side case, in the equivalence result 3.5.1 instead of g we get $g(\xi)$:
respectively $A\xi = g(\xi)$ and $M\dot{\xi} + A\xi = g(\xi)$.

In more general form for both problems (the IBVP as well after time discretization for each time step we're about to discuss) we have to solve the non-linear system of algebraic equations $\mathbf{f}(\xi) = 0$, where $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$ - nonlinear, assumed differentiable vector function.

This is due to the fact that by using a suitable time-stepping scheme the linear first term of $M\dot{\xi} + A\xi = g(\xi)$ with respect to $\xi(t)$ now becomes linear with respect to all introduced quantities in discrete time.

One of the choices is the *backward Euler* scheme [Hac17], [Ang03].

Def. 3.6.1. *We call the scheme* $\dot{\xi} \approx \dfrac{\xi_k - \xi_{k-1}}{\tau_k}$ *a backward Euler scheme*

with the time steps τ_k, $k = 1..\mathrm{N}$.
Then, by substituting that in (3.67), we obtain

$$M\frac{\xi_k - \xi_{k-1}}{\tau_k} + A\xi_k = g(\xi_k), \tag{3.92}$$

or in more compact way $\mathbf{f}(\xi) = 0$ with

$$\mathbf{f}(\xi_k) = (M + \tau_k A)\xi_k - M\xi_{k-1} - \tau_k g(\xi_k). \tag{3.93}$$

The scheme is selected as a more or less universal one, but other implicit schemes would yield the same model - a system of non-linear algebraic equations.

Thus in the following we omit the vectorial notation and present some results on linearisation of general non-linear systems
$f(x) = 0$.

First approach is to make a set of equivalent algebraic operations which leads to simple iterations. This trick at the same time proves that if the convergence is obtained, the limit element is indeed one of the solutions of the system. One can easily construct a simple iteration technique for this problem:

$$\begin{aligned} f(x) &= 0, \\ hf(x) &= 0, \\ hf(x) + x &= x, \end{aligned} \tag{3.94}$$

with the iteration process

$$x_{k+1} = x_k + h_k f(x_k),\ x_0 = \tilde{x_0},\ h_k \in \mathbb{R} \setminus \{0\},\ k = 0..\mathrm{N}. \tag{3.95}$$

Def. 3.6.2. *We call the method (3.95) a simple iteration method.*

One can observe that if the convergence has been obtained, i.e
$x^* = x^* + h_k f(x^*)$, then $f(x^*) = 0$, which means that the fixed point is one of the solutions of the given non-linear system.

But this fact is not enough since one need to exclude the divergent iterations, therefore one requires the notion of the *contraction map*.

We define our function f in $\Omega \subset \mathbb{R}^n$ - convex, bounded set, so $f : \Omega \to \Omega$.
We consider the results quite abstractly - in metric spaces.

Let Ω be a metric space (Ω, ρ) with standard metric space axioms:
$\rho : \Omega \times \Omega \to [0, \infty)$:

1) $\rho(x, y) \geq 0$,

2) $\rho(x, y) = 0 \Leftrightarrow x = y$,

3) $\rho(x,y) = \rho(y,x)$,

4) $\rho(x,y) \le \rho(x,z) + \rho(z,y)\ \forall z \in \Omega$.

We can choose $\rho(x,y) = \|x-y\|$ if Ω - linear normed space.
Once we've got the metrics we need the notion of *convergence*.

Def. 3.6.3. *(Convergence) A sequence $\{x_n\}$ of metric space's (Ω,ρ) elements does converge to an element $x^* \in \Omega$ iff $\rho(x_n, x^*) \to 0,\ n \to \infty$.*

Def. 3.6.4. *(Contraction) We call a mapping $f : \Omega \to \Omega$ a contraction when*
$\rho(f(x), f(y)) \le q \cdot \rho(x,y)$,
$\forall x, y, f(x), f(y) \in \Omega,\ 0 \le q < 1$.

Prop. 3.6.2. *Assume that $\Omega \subset \mathbb{R}^n$ - convex, bounded set.*
Then for each contraction $f:\ \Omega \to \Omega\ \exists!\ x^* \in \Omega :$
$f(x^*) = x^*$
- a fixed point.

Proof. 1) Existence
For an arbitrary element $x_0 \in \Omega$ we can make a sequence of Ω elements $\{x_n\}$:
$x_{k+1} = f(x_k),\ k = 0, 1, 2, \ldots,$

we show that $\{x_n\} \to x^*$, by using the convergence definition.
$f(x^*) = x^*$ means that $\rho(f(x^*), x^*) = 0$ or $\rho(x_k, x^*) \to 0$.

We construct the iteration process:
$x_0 \in \Omega$,
$f(x_0) = x_1 \in \Omega$,
$\rho(x_0, x_1) \le C$ (due to convex and bounded set).

$$x_2 = f(x_1) \in \Omega. \tag{3.96}$$
$$\rho(x_1, x_2) = \rho(f(x_0), f(x_1)) \le q \cdot \rho(x_0, x_1) \le q \cdot C,$$

$$x_3 = f(x_2) \in \Omega. \tag{3.97}$$
$$\rho(x_2, x_3) = \rho(f(x_1), f(x_2)) \le q \cdot \rho(x_1, x_2) \le q^2 \cdot C,$$
$$\vdots$$
$$x_{k+1} = f(x_k) \in \Omega. \tag{3.98}$$

$$\rho(x_k, x_{k+1}) = \rho(f(x_{k-1}), f(x_k)) \le q \cdot \rho(x_{k-1}, x_k) \le q^k \cdot C,\ k = 1, 2 \ldots. \tag{3.99}$$

The limit x^* exists since

$$\rho(x_k, x^*) \le \sum_{n=k}^{\infty} \rho(x_n, x_{n+1}) \le C \sum_{n=k}^{\infty} q^n = C \frac{q^k}{1-q} \to 0,\ k \to \infty. \tag{3.100}$$

We have used the formula for sum of geometric series

$$\sum_{k=0}^{\infty} q^k = \frac{q^0}{1-q} = \frac{1}{1-q}, \; |q| < 1. \tag{3.101}$$

2) Uniqueness.

We assume that there exists x_1^* and x_2^* - two fixed points of the contraction $f(x)$:

$$f(x_1^*) = x_1^*, \; f(x_2^*) = x_2^*, \; \rho(x_1^*, x_2^*) > 0. \tag{3.102}$$

By using the definition of the contraction (3.6.4),

$$\rho(f(x_1^*), f(x_2^*)) \leq q \cdot \rho(x_1^*, x_2^*), \tag{3.103}$$

or

$$\rho(x_1^*, x_2^*) \leq q \cdot \rho(x_1^*, x_2^*), \tag{3.104}$$

thus

$$\rho(x_1^*, x_2^*) = 0, \tag{3.105}$$

which leads to a contradiction. □

Thus, by using the simple iterations $x_{k+1} = x_k + h_k f(x_k)$, or any other iteration method in form $x_{k+1} = g(x_k)$, $x_0 = \tilde{x_0}$, in order for method to converge (or a fixed point to exist and be unique), the following has to hold:
$g(x)$ - a contraction, $\Omega \to \Omega$ (since the proof held for an arbitrary $x_0 \in \Omega$). Hence the next result.

Prop. 3.6.3. $g : \Omega \to \Omega, \; \Omega \subset \mathbb{R}^n, \; g \in C^1(\Omega)$.
The iteration method in form $x_{k+1} = g(x_k)$ *does converge (*g *is contraction) if*
$\|J(g(x))\| < 1, \forall\, x \in \Omega$.

Proof. We apply the finite increment formula [Fin]:
$g : \Omega \to \Omega, \; \Omega \subset \mathbb{R}^n, \; g \in C^1(\Omega)$. Ω - convex, bounded set.
Then

$$\|g(x) - g(y)\| \leq \sup_{\xi \in \Omega} \|J(g(\xi))\| \cdot \|x - y\|. \tag{3.106}$$

That is done by choosing a suitable matrix norm ($1, 2$ or ∞ - all are equivalent in a finite dimensional space). In our terms:

$$\|g(x) - g(y)\| = \|J(g(\xi)) \cdot (x-y)\| \leq \|J(g(\xi))\| \cdot \|x-y\| \leq \max_{\xi \in [a,b]} \|J(g(\xi))\| \cdot \|x-y\|. \tag{3.107}$$

g would become contraction when $\max\limits_{\xi \in [a,b]} \|J(g(\xi))\| < 1$ or

$$\|J(g(x))\| < 1, \; \forall\, x \in \Omega. \tag{3.108}$$

This concludes the proof. □

The special case of this condition is the *Newton method* - the one that guarantees that the condition (3.6.3) holds by neglecting the growth factor $J(g(x)) \equiv \theta_{n \times n}$ since $\|J(g(x))\| = 0 \Leftrightarrow J(g(x)) \equiv \theta_{n \times n}$.

For our simple iterations (3.95) that means:

$$\begin{aligned} g(x) &= x + hf(x), \\ x_{k+1} = g(x_k) &= x_k + h_k f(x_k), \\ J(g(x_k)) &= E + h_k J(f(x_k)), \end{aligned} \tag{3.109}$$

thus

$$h_k = -J^{-1}(f(x_k)). \tag{3.110}$$

Therefore the method has the form

$$x_{k+1} = x_k - J^{-1}(f(x_k)) \cdot f(x_k),\ k = 0..\mathrm{N}. \tag{3.111}$$

Thus whenever one wishes to solve the non-linear (or linear) system (3.83) with iteration techniques one considers the process (3.6.2) to reach the predefined tolerance and moreover if the constructed function $g(x)$ is differentiable one considers the Newton method (3.111) to annihilate the growth factor entirely or one of the other parameter optimization techniques.

In the applications' chapter 6 the methods to be chosen depending on situation. Numerical simulations will show that Newton's method or the one with the approximate derivatives in it lead to convergence in case of non-linearities in our models.

The treatment of the non-linearity has now been pinned down and this concludes the chapter since now one knows how to consider the linear stationary problem in its differential form, discretize the solution space and obtain the discretized problem and regarding the time stepping and linearization, one transforms the non-linear time dependent problem onto sequence of linear time-independent ones.

The next chapter deals with special techniques on solution post-processing such as parameters' optimization, model reductions and the information storage, which are to be much more closely related with the main topic of this thesis and real life applications.

Chapter 4

Parameter optimization and model reduction strategies

This chapter is the core of the thesis. Models obtained in Chapter 2 are parametrized and the parametrization is to be discussed more thoroughly in the applications Chapter 6 for each particular application. In this chapter however we address the general procedures arising in the topic of treatment of the parametrization aspect and consider several modern techniques.

The idea to introduce the topic in the first place comes from the fact that there won't be enough reactions introduced in the model or won't be enough data in general. If we work with an adequate dimension (for author to program and simulate in a short amount of time) but assume that the model parameters can be varied, one can teach the model to be close enough to the experimental data.

Now, 'close enough' interprets such that the quantities have to be minimized, thus we consider a class of optimization problems. Since the numerical solution obtaining procedure is rather complex, we shall consider the class of optimization techniques that cannot be solved with an exact algorithm. The concept of heuristics is crucial here - by doing research on that and how the notion is related to the optimization problems, one can find different results in publications and books [BAE05], [Cle06], [Wil10], [Gre08].

Several techniques amongst those the author has found shall be considered here, given in their classical forms. Then the modification of each of them shall be introduced and motivated by the class of application problems.

Whether one or another optimization procedure is called out, it is a part of a scheme, which uses the procedures from the previous chapter. It takes the vector of parameters as an input by calling out the solution process many times, teaches itself how to reach the optimization goal. Then in the applications chapter, for each of the applications this goal is formulated. Finally, the result of the last solution procedure can be interpreted as the final one and brought further for the post-processing.

For the applications considered in chapter 6 one or several techniques presented in the next sections shall be applied. A brief analysis of each such application such as dimensionality and the type of the problem allows us to pick the appropriate tool amongst the following techniques.

The main motivation to consider the optimization technique is that the modelling results are not necessarily consistent with the experimental results. Thus we need a concept to minimize the distance between the chosen quantities.

The reason to choose the stochastic numerical method is that the data might be not smooth enough and the functional to optimize appears as a sort of black box in complicated cases even though in simple cases the structure can be visible.

In general the expression to minimize shall look like this:

$$f(\mathbf{p}) = \|\mathbf{Q}_{exp} - \mathbf{Q}_{mod}(\mathbf{p})\|^2, \tag{4.1}$$

where $\mathbf{Q}_{exp} \in \mathbb{R}^n$ is the given vector of experimental values and $\mathbf{Q}_{mod} \in \mathbb{R}^n$ is the result of post-processing of the PDE system (2.11), (2.18) solution. The minimimization problem then should look like this

$$\min_{\mathbf{p}} \|\mathbf{Q}_{exp} - \mathbf{Q}_{mod}(\mathbf{p})\|^2. \tag{4.2}$$

Also $\mathbf{p}$ is the vector of parameters for our parametrized model - dimension to be denoted as D_p depends on the nature of the task.

In our models (2.11), (2.18) it might be maximal and outlet temperatures comparison to name the least:

$$\min_{\mathbf{p}} \|T_{max_{exp}} - T_{max_{mod}}\|^2,$$

$$\min_{\mathbf{p}} \left\| \begin{pmatrix} T_{max_{exp}} \\ T_{out_{exp}} \end{pmatrix} - \begin{pmatrix} T_{max_{mod}} \\ T_{out_{mod}} \end{pmatrix} \right\|^2.$$

These and other choices of cost functional to be discussed in the applications chapter 6.

One good feature of the functionals constructed is that they directly represent the measure between experimental and modelling data in a certain way to a certain extent. After one stops the optimization procedure, the value of the functional can be claimed as such measure and it can easily be reported.

In the first section we discuss the choice of the optimization technique, present its algorithm, convergence criterion and prove the convergence criterion.

4.1 Particle swarm optimization

The choice of this numerical non-gradient optimization method is due to the fact that in general we don't have an explicit expression of the functional. This is either due to the iterative nature of finding the solution of the PDE system (2.11), (2.18) when we code the solver by ourselves or to the complete black-box nature of our solving process when we use an industrial package for solving it. Thus we look onto the set of gradient-free methods.

For several other applications throughout the PHD studies author had made his choice on the PSO due to simplicity, universality and since it has been showing good result in terms of leading to optimal directions.

In this section we present the method (4.3), (4.4) and the convergence criterion 4.1.1. The results are adopted out of several books and publications [BAE05], [Cle06].

We present the algorithm in its classical form. It uses a modification of discrete velocities and positions of particles and for each particle determines winning states

$\mathbf{p}^i \in \mathbb{R}^{D_p}$ and for the entire swarm - the winning state $\mathbf{g} \in \mathbb{R}^{D_p}$ and depends on method's parameters $w,\ c_1,\ c_2 \in \mathbb{R}$ and $r_1,\ r_2 \in \mathbb{R}$:

$$v_d^i(k+1) = wv^i(k) + c_1 r_1 (p_d^i - x_d^i(k)) + c_2 r_2 (g_d - x_d^i(k)), \tag{4.3}$$

$$x_d^i(k+1) = x_d^i(k) + v_d^i(k+1), \tag{4.4}$$

$k = 1..\mathrm{N},\ d = 1..D_p,\ i = 1..S,$ where N - number of iterations, D_p - number of parameters, S - number of particles.
$c_1 r_1$ and $c_2 r_2$ are usually combined and the notations $\phi_1 := c_1 r_1$ and $\phi_2 := c_2 r_2$ are widely used and is due to the fact that $r_{1,2}$ are usually picked as two uniformly distributed random numbers $\sim U(0,1)$.

The choice of parameters to be discussed below.

The process (4.3), (4.4) iterates the quantity $x_d^i(k)$ in upper and lower bounds $\mathbf{b}_{lo},\ \mathbf{b}_{up}$.

In order to program this method as we are going to in chapter 5, we need to construct the algorithm out of our formulas (4.3), (4.4) along with the initialization of data.

We come up with the simplest way how to do it and the algorithm looks as follows:

Alg. 4.1.1.

1) The initial cycle.
Randomize the swarm's best known position $\mathbf{g}$ and the position of each particle

$$\mathbf{x}^i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up}),\ \mathbf{g} \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up}),\ i = 1..S, \tag{4.5}$$

and the particle's best known position set to its initial position $\mathbf{p}^i := \mathbf{x}^i$.
If $f(\mathbf{p}^i) < f(\mathbf{g})$ then set $\mathbf{g} := \mathbf{p}^i$.

Initialize the particle's velocity:

$$\mathbf{v}^i \sim U(-|\mathbf{b}_{up} - \mathbf{b}_{lo}|,\ |\mathbf{b}_{up} - \mathbf{b}_{lo}|) \tag{4.6}$$

The randomization and absolute value taking is happening component-wise.

2) The main cycle.
For $k = 1..\mathrm{N},\ d = 1..D_p,\ i = 1..S$:
implement the formulas (4.3), (4.4).

Update the winning states:

$$\begin{aligned} f(\mathbf{x}^i) < f(\mathbf{p}^i) \ &\Rightarrow\ \mathbf{p}^i := \mathbf{x}^i, \\ f(\mathbf{p}^i) < f(\mathbf{g}) \ &\Rightarrow\ \mathbf{g} := \mathbf{p}^i. \end{aligned} \tag{4.7}$$

The implementation of this algorithm in Matlab is going to be pinned down in the next chapter 5.

Since upper and lower bounds $\mathbf{b}_{up},\ \mathbf{b}_{lo}$ are present, we need to apply some boundary controls. The linear operations in (4.3), (4.4) can easily compute the new state $x_d^i(k)$ out of bounds. This thing should be taken care of a-priori.

The author came up with the following naive set of actions, which so far have shown to be consistent with all the author's experiments and in particular the ones from the applications chapter 6.

In order to perform a constrained optimization between $\mathbf{b}_{lo}$, $\mathbf{b}_{up}$, we apply the following actions.

The first action is for each k (round brackets omitted):

$$x_d^i + v_d^i < b_{lo_d} \lor x_d^i + v_d^i > b_{up_d} \Rightarrow v_d^i := -v_d^i, \tag{4.8}$$

which is to switch the velocity sign in case if the generated velocity at the corresponding step k is about to make the particle move through the boundary. This would lead us to bouncing backwards from the boundary, but doesn't guarantee that we don't move beyond the opposite boundary by applying this action. Thus one more set of controls is to be applied:

$$\begin{aligned} x_d^i + v_d^i < b_{lo_d} &\Rightarrow v_d^i := b_{lo_d} - x_d^i, \\ x_d^i + v_d^i > b_{up_d} &\Rightarrow v_d^i := b_{up_d} - x_d^i, \end{aligned} \tag{4.9}$$

which guarantees that we never move beyond any boundary by staying at the corresponding boundary if the velocity sign switching action gives no luck.

The convergence to at least a local minimum is what we wish to discuss now. In the applications section we need to make our quantities closer to each other to a certain extent - due to the nature of our problems (4.1). The algorithm (4.5)-(4.7) has appeared to be trustworthy in an empirical way - so far it has been consistent with all the experiments - in our case - test problems with the a-priori known optimum.

And the other thing is - the nature of our problems (4.1) is always to tend distances to zero - therefore in our rather complex models in the applications chapter 6 the parameter optimization procedure always knows the optimal value of the functional but the *argmin* value is to be determined which eases the life in this particular aspect to a certain extent. And finally, for sake of applications considered we are totally fine with finding only one of argmins in case if there are several.

Thus we present the convergence criterion - the choice of method's parameters, that guarantees us to tend the value of our functional to the optimal one and as we see experimentally - for our applications in the applications chapter 6, indeed find the argmin of the approximated zero value of our functional.

Prop. 4.1.1. *The choice of parameters such as* $\max\{|\lambda_1|, |\lambda_2|\} < 1$,
where $\lambda_1 = \dfrac{1 + w - \phi_1 - \phi_2 + \gamma}{2}$ *and* $\lambda_2 = \dfrac{1 + w - \phi_1 - \phi_2 - \gamma}{2}$,
$\gamma = \sqrt{(1 + w - \phi_1 - \phi_2)^2 - 4w}$
guarantees the convergence of (4.3), (4.4) to the best state of their population $\mathbf{g}$:

$$\lim_{k\to\infty} x_d^i(k) = g_d, \; d = 1..D_p, \; i = 1..S. \tag{4.10}$$

The design of the criterion is rather counter-intuitive but let's see why it is the case by proving the statement.

Proof. In order to talk about convergence, first of all we need to obtain a recurrence relation as a function of previous states for $x_d^i(k)$, $k \in \mathbb{N}_0$ for each $d = 1..D_p$ and $i = i..S$ denoted as $x(k)$ from now on since the dimensions are independent and we only work with linear operations on vectors. Same with velocities and winning states:

$$v(k+1) = wv(k) + \phi_1(p - x(k)) + \phi_2(g - x(k)), \tag{4.11}$$

$$x(k+1) = x(k) + v(k+1). \tag{4.12}$$

By iterating the system and using common notation for difference equations $x_k := x(k),\ k \in \mathbb{N}_0$, one obtains

$$x_{k+1} = x_k - \phi_1 x_k - \phi_2 x_k + wv_k + \phi_1 p + \phi_2 g. \tag{4.13}$$

Since $v_k = x_k - x_{k-1}$, one obtains

$$x_{k+1} = x_k - \phi_1 x_k - \phi_2 x_k + wx_k - wx_{k-1} + \phi_1 p + \phi_2 g, \tag{4.14}$$

or

$$x_{k+1} + Ax_k + Bx_{k-1} = f, \tag{4.15}$$

where

$$A = \phi_1 + \phi_2 - w - 1,\ B = w,\ f = \phi_1 p + \phi_2 g. \tag{4.16}$$

In order to iterate the difference equation (4.15) we should set the initial conditions

$$x_0 = \tilde{x_0},\ x_1 = \tilde{x_1},\ \tilde{x_0},\ \tilde{x_1} \in \mathbb{R}. \tag{4.17}$$

The problem (4.15), (4.17) describes the trajectory of each particle's each component up until either particle's winning state p or global winning state g gets updated. After each update the process continues with the new values. Since the difference equation's coefficients A and B do not depend on p and g, one can show convergence for every $p, g \in \mathbb{R}$.

The solution for this problem for second order difference equation with constant coefficients (4.15) is sought in form

$$x_k = C_1\lambda_1^k + C_2\lambda_2^k + x_{part}, \tag{4.18}$$

where $C_1,\ C_2 \in \mathbb{R},\ \lambda_i \in \mathbb{C},\ i = 1..2$ and it is done by means of substitution

$x_k = C\lambda^{k-1},\ C \in \mathbb{R} \setminus \{0\}, \lambda \in \mathbb{C} \setminus \{0\}$ - a discrete analogue of Euler substitution for differential equations.
Thus for the homogeneous part we obtain the characteristic equation

$$\lambda^2 + A\lambda + B = 0. \tag{4.19}$$

Thus λ_1 and λ_2 are the roots of the quadratic equation (4.19):

$$\lambda_{1,2} = \frac{-A \pm \sqrt{A^2 - 4B}}{2}, \tag{4.20}$$

from where

$$\lambda_1 = \frac{w + 1 - \phi_1 - \phi_2 + \sqrt{(\phi_1 + \phi_2 - w - 1)^2 - 4w}}{2}, \tag{4.21}$$

$$\lambda_2 = \frac{w + 1 - \phi_1 - \phi_2 - \sqrt{(\phi_1 + \phi_2 - w - 1)^2 - 4w}}{2}. \tag{4.22}$$

Regarding the particular solution $x_{part} = C,\ C \in \mathbb{R}$, it can be found by plugging it in the equation (4.15):

$$C(\cancel{1+}\phi_1 + \phi_2 - \cancel{w} - \cancel{1} + \cancel{w}) = \phi_1 p + \phi_2 g, \tag{4.23}$$

hence

$$C = x_{part} = \frac{\phi_1 p + \phi_2 g}{\phi_1 + \phi_2}. \tag{4.24}$$

and by tending k to infinity in the representation (4.18) one gets

$x_k \underset{k \to \infty}{\to} x_{part}$ when the condition $\max\{|\lambda_1|, |\lambda_2|\} < 1$ holds.

The weighted average (4.24) of the global and local best states g and p obviously would tend to g at point when all the particles don't seem to find any better p_i in comparison with the state g. $\square$

Thus we have obtained that there exists a parameter region whereas particles would stop moving in the iteration process (4.3), (4.4) while each moving step is either stagnant or indeed minimizes the functional (4.1).

What we don't want to do in formulas (4.21) and (4.22) is go to complex values such as $(\phi_1 + \phi_2 - w - 1)^2 > 4w$ (even though it is still an option and the absolute value in Prop.4.1.1 can be interpreted as the length of a complex number) or these quantities to be larger or equal to unity.

The popular parameter choice, that can be found in different sources and to be used in this thesis since it has been consistent with all author's numerical experiments so far is $w = 0.3,\ \phi_i \in (0,\ 1.2),\ i = 1..2$. As we stated in the beginning, ϕ_i is decomposed onto $c_i r_i$, where $r_i \sim U(0,\ 1),\ c_i = 1.2,\ i = 1..2$.

If one plugs in the numbers it is easy to see that this gives somewhat extreme values in order not to go to a complex plane as well as the ratios not to exceed the unity.

The other aspect is the functional values' comparison operation that also lies within.

The practice shows that the optimization process runs rather fast but there obviously is one costly part, which is the functional evaluation process. This is clearly the running of the PDE solution process and the post-processing of the solution.

The PDE solution process is the part that should be taken care of since the inverting of large matrices or many matrix-vector multiplications are brought to the table and moreover this complex of actions should be done many times in the optimization process.

In the next sections we discuss how to overcome these difficulties to a certain extent and simplify these underlying procedures.

4.2 Reduction due to principal components

The present model reduction technique naturally has to be introduced for general models, discussed briefly in chapter 2, where we solve for $5 + N$ equations, where we take the account of temperature, velocities, density and N is the number of species, which can be measured in hundreds. One can see how these models look like e.g in [Vey05] or [Wil85].

As for the models introduced in this thesis, the idea is to reduce the dimension for the discretized and linearized models (3.67). There's a necessity for that since our optimization procedures described in previous chapter lead to high computational times.

We do the reduction to the certain extent: the dimension should be reduced as much as possible, whilst at the same time measure of difference between the results should be admissible.

The author is mainly inspired by the published results [Wil10], [Gre08].

So main technique we are going to use here is the *Principal Component Analysis*.

A principal component analysis is required in order to speed up the evaluation of the functional whilst performing the optimization. Since we are going to mostly deal with vectorial quantities, the bold notation for vectors is going to be omitted till the end of the chapter.

The procedure requires making of the matrix of snapshots first:

$$X = [\hat{x}_1, \ldots, \hat{x}_L]. \tag{4.25}$$

This matrix should contain the representative solutions $\hat{x}_k,\ k = 1..L$ at different time steps. This should be calculated once with full dimension. Afterwards this matrix is going to be used to form a suitable basis.

If we seek the representation of the snapshots in the reduced basis with the introduced measure of quality of the representation, we wish to minimize the quantity

$$\sum_{k=1}^{L} \|\hat{x}_k - \hat{V}C\|_2^2 \tag{4.26}$$

subject to orthogonal bases of K elements, stored as the columns of matrix $\hat{V} \in \mathbb{R}^{n\times K},\ K < L$. Small n here is in correspondence with the number of basis elements for Galerkin or FEM solution from chapter 3.

C represents the vector of coefficients in this basis - for each $\hat{V}$ to be uniquely determined.

Matrix of snapshots is rectangular and there's a necessity to introduce the generalization of matrices eigenrepresentation: the *Singular Value Decomposition*.

Def. 4.2.1. *The SVD - a (compact) singular value decomposition is a factorization of a matrix $X \in \mathbb{R}^{n\times L}$ in form $X = V\Sigma U^T$, where $V \in \mathbb{R}^{n\times L}$ is an orthogonal matrix, $\Sigma \in \mathbb{R}^{L\times L} = diag([\sigma_1,\ \sigma_2, \ldots, \sigma_L])$ with positive diagonal elements, $U \in \mathbb{R}^{L\times L}$ - an orthogonal matrix.*

If we consider the left singular vector matrix V in the SVD $X = V\Sigma U^T$, where $X = [\hat{x}_k]$ - matrix of the solution representatives (4.25), we need to know the representation of the basis vectors as well as the quality of approximation of snapshots. As usual in this thesis we formulate one vast proposition and the technical peculiarities throughout the proof are going to be of importance.

Prop. 4.2.1. *For a matrix of snapshots $X \in \mathbb{R}^{n\times L}$ the left singular vector matrix V is the orthogonal eigenvectors matrix of XX^T.*
The right singular vectors in U are the orthogonal eigenvectors of X^TX.
The singular values are the square roots of eigenvalues of X^TX.

If we choose the set of $K < L$ left singular vectors to form a reduced basis in (4.26), the value of (4.26) shall be $\|\sigma\|_2^2$, where $\sigma \in \mathbb{R}^{L-K}$ - the vector of leftover singular values, corresponding to $L - K$ left singular vectors.

Proof. We are going to widely use the property of the transpose of the product of two matrices $(AB)^T = B^T A^T, A \in \mathbb{R}^{n\times m},\ B \in \mathbb{R}^{m\times p}$ as well as the dot product property for $A \in \mathbb{R}^{n\times n},\ x, y \in \mathbb{R}^n :\ (Ax, y) = (x, A^T y)$
since

$$(Ax, y) := (Ax)^T y = x^T A^T y = (x, A^T y). \tag{4.27}$$

The first part is a link between the eigenrepresentation and the singular representation:

$$XX^T = V\Sigma U^T (V\Sigma U^T)^T = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T \tag{4.28}$$

and

$$X^T X = (V\Sigma U^T)^T V\Sigma U^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T \tag{4.29}$$

since U and V are orthogonal and Σ is diagonal.

The Σ^2 representation is $\mathrm{diag}([\sigma_1^2, \dots, \sigma_L^2])$ and in comparison with the regular eigenvalue representation $V\Lambda V^{-1}$,
we would have the coincidence of V and U as well as the relation $\lambda_k = \sigma_k^2,\ k = 1..L$.
iff the eigenvectors for both $X^T X$ and XX^T were orthogonal.

Both $X^T X$ and XX^T are symmetric since $(X^T X)^T = X^T X$ and $(XX^T)^T = XX^T$ and assuming there are no eigenvalues of multiplicity > 1 (in this case the representatives can be reconstructed), one can choose the orthonormal eigenvector basis since we have for $\lambda_i \neq \lambda_j$
and $v_i,\ v_j,\ i, j = 1..L :$
$\lambda_i(v_i, v_j) = (\lambda_i v_i, v_j) = (Av_i, v_j)$.

Due to (4.27) and symmetry we have $(Av_i, v_j) = (v_i, Av_j) = (v_i, \lambda_j v_j) = \lambda_j(v_i, v_j)$.

Therefore $(\lambda_i - \lambda_j)(v_i, v_j) = 0$ when the scalar product is zero.

As for the last part:
for $X^T X$ we have

$$\sum_{k=1}^{L} \lambda_k(X^T X) = \mathrm{tr} X^T X = \sum_{k=1}^{L} \|X_k\|_2^2 \tag{4.30}$$

since on the diagonal of $X^T X$ we've got
(X_k, X_k), where the lower index stands for column number.

Therefore, by representing each snapshot in full basis $\hat{x}_k = VV^T \hat{x}_k$ and computing the coefficients in the reduced basis $C = \hat{V}^T \hat{x}_k$ in (4.26), we've got

$$\sum_{k=1}^{L} \|\hat{x}_k - \hat{V}C\|_2^2 = \sum_{k=1}^{L} \|VV^T \hat{x}_k - \hat{V}\hat{V}^T \hat{x}_k\|_2^2 = \mathrm{tr} Y^T Y \tag{4.31}$$

where

$$Y = VV^T X - \hat{V}\hat{V}^T X \tag{4.32}$$

and since

$$VV^T = \sum_{k=1}^{L} V_k V_k^T,\ \hat{V}\hat{V}^T = \sum_{k=1}^{K} V_k V_k^T, \tag{4.33}$$

$$Y = \Big[\sum_{k=K+1}^{L} V_k V_k^T\Big] X = V_L V_L^T X, \tag{4.34}$$

where V_L is the matrix of leftover basis vectors, not taken into reduced basis.

Due to orthogonality one gets

$$Y^TY = (V_LV_L^TX)^T(V_LV_L^TX) = X^TV_L\cancel{V_L^TV_L}V_L^TX = X^TV_LV_L^TX \tag{4.35}$$

and by taking SVD of $X = V\Sigma U^T$, one obtains

$$U\Sigma V^TV_LV_L^TV\Sigma U^T. \tag{4.36}$$

Since, due to orthogonality, ΣV^TV_L and $V_L^TV\Sigma$ pick out only leftover singular values, one ends up with the eigenrepresentation $U\Lambda_LU^T$, where $\Lambda_{L,kk} = \sigma_k^2,\ k = K+1..L$ if it corresponds to leftover basis vector $V_{L,k}$ or zero otherwise (when $k = 1..K$).

Hence by using the above statements,

$$\sum_{k=1}^{L}\|\hat{x}_k - \hat{V}C\|_2^2 = \mathrm{tr}(Y^TY) = \sum_{k=1}^{L}\lambda_k(Y^TY) = \sum_{k=1}^{L}[\sigma_k(Y)]^2 = \sum_{k=K+1}^{L}[\sigma_k(X)]^2 = \|\sigma\|_2^2, \tag{4.37}$$

with $\sigma \in \mathbb{R}^{L-K}$.
This concludes the proof. $\square$

Thus, in order to form a suitable reduced basis, one should take the most 'energetic' singular vectors - the ones corresponding to largest singular values.

Whenever we execute the PDE solution process, described in Chapter 3 and arrive to a system of ODEs after spatially discretizing the system of parabolic PDEs with FEMs or to a linear system in steady case, we introduce the vector of reduced states $x_r,\ Vx_r \approx x$ and $x_{0,r},\ Vx_{0,r} \approx x_0$. It is easy to invert the orthogonal matrix by transposing and the spatially discretized system (recall (3.67) with non-linearity in the RHS) then becomes:

$$\dot{x_r} = V^TBVx_r + V^Tg(Vx_r),\ x_{0,r} = V^Tx_0 \tag{4.38}$$

and after iterating the system in time as in Chapter 3, the closer we are to the snapshot representation the better and at a single step if one of the snapshots is not represented at the reduced basis, we iterate with an introduced error since the iterations of the reduced system would perturb the representative vectors.

Overall, as we see in the applications' chapter 6, leaving behind the vectors corresponding to small singular values and taking into basis the energetic ones shall show very good results and the dimension shall be reduced and the quality of approximation is given by Prop. 4.2.1.

After time discretization and applying the back substitution $x = Vx_r$ one does the regular solution post-processing in order to compute the value of the functional (4.1) in the optimization procedure.

Thus we end up with the approximated zero of the functional. To tackle this approximation, the author came up with the algorithm for the applications section:

Alg. 4.2.1.

1) Before running but after initializing the PSO (4.3), (4.4), compute the snapshot matrix for initial vector of parameters.

2) Run the optimization procedure with the reduced states.

3) Converge to an approximated zero of the functional.

4) Compute the value of the functional with the p=argmin for the solution in full basis. At the same time get the new snapshot matrix.

5) If the tolerance not reached, repeat the procedure.

The new snapshot matrix in 4) would clearly continue the procedure such as we are even closer to zero at all further steps since the movement of the best particle is possible only if the better value is found and even if robustness perturbs the values, running the whole thing several times would apply the control at step 4) as many times as required.

The ways of simplifying the functional evaluation procedure has now been considered and the choice to reduce the dimension by using the technique as long with its properties' choice shall be used separately for each application in the applications' chapter 6.

Now we are ready for the last section in the present chapter, which deals with treatment of the unstored information appearing in the optimization cycles in our general optimization scheme.

4.3 Information storage and pattern recognition

The goal is to reduce the number of evaluations of the functional (4.1).

Consider the sequence of experiments $E_i,\ i = 1..N_E$. If the new experimental data is considered for some i, one can use the gathered knowledge from the previous optimization stages for previous experiments since the optimization technique processes a lot of solutions not feasible for the present experiment, but that might be feasible for any subsequent experiment.

The idea is to store some of the information from previous experiments in a form of network - a discrete dynamical system with a certain convergence-guaranteed properties, which can be addressed as a first step of the optimization strategy. Due to these properties its evaluation shall be much less costly than the functional evaluation.

In a realm of applications of a certain nature (likewise considered in the application chapter 6 for this thesis) it might be possible to reduce the complex differential parametrized problem onto such network such as for some experiments there is a need to address the latter only.

Experimental values are either provided by the team that actually facilitates the experiments (and in this case the application has some value in terms of an actual interdisciplinary collaboration) or are somewhat intuitively set by an author and in this case the application has a potential value (in case someone provides an accurate

numerical value to be compared with). The origin of each, to be chosen in the applications chapter, is going to be discussed individually for each application.
The optimization method (4.3), (4.4) considered processes a lot of feasible or not feasible solutions. The techniques used leave lots of data unstored whilst most of the solutions, especially if the initial guess isn't close enough, most likely should be stored. The motivation to store the data for further usages is quite natural since there might be a situation where the new experimental data is brought to the table. And lots of contemporary techniques such as perceptrons and recurrent networks is something that we here about a lot nowadays and some of their features go together quite well with the fact that we might want to seek the solution close to one of the many that has been already processed and do not want to run the costly computational procedure from the very beginning. Before doing that one would address different types of models first such as pattern recognition and in case if those don't give any good results, run the procedure (4.3), (4.4) once again, at the same time storing the new data. Gradually if we repeat this procedure necessary amount of times, we naturally arrive at the point where there is no necessity to initiate the stage of costly procedures, and simply either run the solution process once with all the parametrization fully determined or, if we get an approximation out of it, use it as a starting point in our costly procedures which would substantially reduce the number of iterations.
Thus till the end of the present chapter we address the technique of storing the information in a way such that the resulting structure reproduces it in a smart way - given an arbitrary piece of information the system produces the closest one in terms of the defined measure. Similar approach is to be found in theory of recurrent systems and finite automation [Cha01]. Then the author adopts it for the considered applications and couples it with the optimization procedure (4.3), (4.4).

The idea is to introduce the discrete dynamical system

$$x(k+1) = f(x(k)),\ x(0) = x_0, \tag{4.39}$$

where

$$f(x) = \mathbf{sign}(Ax),\ k = 0..\mathrm{N},\ x_0, x(k) \in \{-1, 1\}^D,\ D \in \mathbb{N} \tag{4.40}$$

and the bold-case sign stands for the sign function component-wise with zero mapped to one. Each pattern $x^i,\ i = 1..P$ to be stored can be represented as a vector of binary values thus the number D can be large.

One of the ways to define the matrix in (4.39),(4.40) is the following:

$$A = \sum_{i=1}^{P} x^i [x^i]^T. \tag{4.41}$$

The latter representation allows to dynamically add or remove patterns.

We need a system to converge to a fixed point and the best scenario is that this fixed point is one of the patterns.

As usual we prove one vast proposition whereas the proof itself explains a lot on this matter.

Prop. 4.3.1. *The system (4.39),(4.40) with A constructed as in (4.41) with $[Ax]_d \neq 0,\ d = 1..D,\ x \in \{-1, 1\}^D$ always converges to a fixed point $x^* :\ f(x^*) = x^*$.
Moreover if the condition $\sum_{\substack{i,j=1 \\ i \neq j}}^{P} |(x^i, x^j)| < D$ holds, then it converges to a fixed point, which*

is one of the patterns: $x^* \in \{x^1, \ldots, x^P\}$.

Proof. First observation is that the system matrix A is symmetric and positive semi-definite.

We are going to widely use the transpose array property for the product of two $[ab]^T = b^T a^T,\ a, b \in \mathbb{R}^D$.

The symmetry holds due to the fact that $A^T = A$ since for $D \times P$ matrix X of patterns entered column-wise

$$A^T = [XX^T]^T = [X^T]^T X^T = XX^T = A, \tag{4.42}$$

and the positive semi-definite property $\xi^T A \xi \geq 0,\ \xi \in \mathbb{R}^D$ holds due to

$$\xi^T XX^T \xi = [X^T \xi]^T X^T \xi = (X^T \xi, X^T \xi) \geq 0. \tag{4.43}$$

Now it is convenient to address the optimization problem equivalent to (4.39).

We define the quadratic energy functional

$$\phi(x) = x^T A x \tag{4.44}$$

and seek the fulfilment of the property

$$x \neq f(x) \Rightarrow \phi(f(x)) > \phi(x) \tag{4.45}$$

such that the iterations solve the maximization problem for the quadratic form in a finite set:

$$\max_{x \in \{-1,1\}^D} \phi(x). \tag{4.46}$$

In order to prove that this property holds we consider the difference $\phi(f(x)) - \phi(x)$ in a finite set and show that it is always positive.

For that we require a piece of arithmetics of the quadratic forms - specifically the difference of two.

We've got the result

$$x^T A x - y^T A y = (x-y)^T A (x-y) + 2(x-y)^T A y \tag{4.47}$$

for $x, y \in \mathbb{R}^D, A \in \mathbb{R}^{D \times D}$ - a symmetric matrix
as a multi-dimensional generalization of the one-dimensional

$$ax^2 - ay^2 = a(x-y)^2 + 2a(x-y)y,\ a, x, y \in \mathbb{R}. \tag{4.48}$$

This is shown by opening brackets and using symmetry of A:

$$(x-y)^T A (x-y) + 2(x-y)^T A y = \underline{x^T A x} - y^T A x - x^T A y + \underline{y^T A y} + 2x^T A y - \underline{2y^T A y}. \tag{4.49}$$

The underlined terms form the necessary statement, while the remaining ones cancel out due to symmetry of A. Thus for our case we have

$$\phi(f(x)) - \phi(x) = (f(x) - x)^T A (f(x) - x) + 2(f(x) - x)^T A x > 0 \tag{4.50}$$

when $x \neq f(x)$.

This is due to the fact that the first summand is non-negative for positive semi-definite A and since $f(x) = \mathbf{sign}[Ax]$, the second summand is strictly positive since for $x \neq f(x)$ or $f(x) - x \neq \theta_D$, only two variants of the non-zero d-th component contribution to scalar product $(f(x) - x, Ax)$ are possible with the non-zero $[Ax]_d$:

$$[f(x) - x]_d = 2 \Rightarrow [f(x)]_d = 1 \Rightarrow [Ax]_d > 0, \tag{4.51}$$

$$[f(x) - x]_d = -2 \Rightarrow [f(x)]_d = -1 \Rightarrow [Ax]_d < 0, \tag{4.52}$$

$d = 1..D$.

In a finite set the existence of the monotonically increasing or decreasing (in our case increasing) energy functional clearly implies the existence of a fixed point.

As for the second part:

we want to use the condition $\sum\limits_{\substack{i,j=1 \\ i \neq j}}^{P} |(x^i, x^j)| < D$ in order to prove that each pattern x_p is a fixed point of the system, i.e $\mathbf{sign}[Ax^p] = x^p,\ p = 1..P$.

By using the representation of A (4.41) we've got the estimate

$$Ax^p = \sum_{i=1}^{P} x^i [x^i]^T x^p = \sum_{i=1}^{P} x^i (x^i, x^p) = Dx^p + \sum_{\substack{i,p=1 \\ i \neq p}}^{P} x^i (x^i, x^p) \tag{4.53}$$

and or each d-th component we get

$$|[Ax^p - Dx^p]_d| \leq \sum_{\substack{i,p=1 \\ i \neq p}}^{P} |(x^i, x^p)| < D \tag{4.54}$$

hence

$$\text{sign}[Ax^p]_d = [x^p]_d,\ d = 1..D,\ p = 1..P \tag{4.55}$$

or

$$\mathbf{sign}[Ax^p] = x^p,\ p = 1..P, \tag{4.56}$$

or, in other words, each pattern is a fixed point.
This concludes the proof. □

The system matrix A is constructed out of the patterns arising from the data left over of the optimization procedure for (4.3), (4.4). Each time one has to find the parameters for the system (2.11), (2.18) given some experimental data regarding the solution, the vector x has to be constructed. One of the ways is to put the given data inside and regarding the unknown values put the deterministic function such as the extreme or average value.

Then running the procedure (4.39) several times will find the closest pattern in terms of the predefined measure. The value of the measure is to be chosen separately for each application in 6.

One can come up with the following algorithm for this matter. It is to be implemented in Matlab in chapter 5, second part of the section 5.5.

Alg. 4.3.1.

1) Analyse the dimensions of the given solution and corresponding parameter data and create the patterns such that they fulfil or are close to condition in (4.3.1).

2) Convert the given initial value in pattern form. In case if some parameter values are left to be unknown, use the deterministic or random values in the given intervals.

3) Call out the DDS (4.39) with the reasonable amount of iterations.

4) Decode the obtained values.

In 3) the whole idea of binary design of the patterns justifies itself and hence strolling around the corners of corresponding hypercube as in 4.3.1 only, usually yields a sufficiency of small amount of iterations.

In 4) in case if the produced unknown parameters are out of bound, the whole procedure can be repeated. This can happen due to presence of spurious fixed points and can be easily eliminated by simple bound control, usually same as we used in PSO in section 4.1 but might be even simpler by considering the reiteration technique only due to the introduced non-costly procedures.

Then the PDE system (2.11), (2.18) is to be solved with the determined parameters and if the result is close enough in terms of the measure (4.1) from the first section of this chapter, one stops, or alternatively the result can be used as a good starting point for further optimization.

Concluding this chapter, one can say that the measure between the experimental and modelling results has been introduced, defined and in terms of this measure the solution close enough to the experimental data can been found by using several model reduction techniques when necessary.

We have described the necessary techniques for the application section in details to a certain extent. They all are prepared in a way such that it is easy to code the underlying algorithms. In the next chapter we are going to discuss some programming aspects in terms of *Matlab* programming language, which shall naturally come together with more detailed problem description, whereas the aspects of computation, not discussed whilst setting the rather abstract theory in the first chapters, are now going to be discussed in details by using the computational approach.

Chapter 5

Test problem treatment with MATLAB

We intentionally left some parts of the modelling undiscussed since it is more convenient to illustrate them by showing the sample computer codes at the same time. These would be:

What domain and which partition of the domain to choose in order to define shape functions.

How to assemble the stiffness matrix, mass matrix and the load vector.

Incorporation of the boundary conditions.

Parametrization and criteria of the optimization method.

Coding and decoding the information for pattern storage task.

MATLAB [Tla] is considered as a very useful tool when it comes to matrix operations.

Our goal here is to create a code which consists of several parts, corresponding to different modelling aspects in previous chapters such that the pieces were to be used either simultaneously or separately.

The motivation of mostly not using the built-in codes lies in a necessity to fully access all the transitional data arrays, required to be controllable when implementing the model reduction (4.38) and the optimization techniques (4.3), (4.4).

The code will consist of several part: - the PDE solver - implementation of the results from Chapters 2, 3. Second - the parameter optimizer - the implementation of the algorithms from Chapter 4. The last one - storage of patterns to a new model - a dynamical system from Chapter 4 and some coding/decoding aspects to be explained there as well.

Each aspect of modelling in the following sections shall be illustrated with a concise piece of code - a function which depends on a set of parameters and returns a set of values. This function shall be called an *m-file* - and the reason for that is that the software produces the file with an extension $*.m$.

These sample m-files to be extended for each applications to some extent, but the main common coding ideas for all considered applications are something we want to present in this chapter.

In order to make it readable (usually the code goes last in the paper and hard to interpret), the author decided to make it different this time and simplify the general code by splitting it onto several parts, making comments and considering simplified test models only.

The author has learned the aspects of MATLAB coding throughout the years of studies in masters and PhD and the tutorials on Mathworks web page [Tla] along with the provided forums were always of great help.

5.1 Triangulation settings

The goal is to produce the grid data in form used by many packages which is somewhat 'equidistant'. This is useful whenever we want to solve the models where we do not want to refine grid in subdomains but are ok with grid of same quality in all domain. In this case we use the fact that each element of the stiffness matrix doesn't have to be treated separately, but there's a pattern to be used in order to assemble one.

Grid data is a set of arrays, and, mathematically speaking, several matrices we want to find. The main two are the node matrix $p \in \mathbb{R}^{n\times m}$, where n is the dimension (of physical space) and m is the number of points. And the other one is the triangle matrix $t \in \{1, 2 \dots m\}^{(n+1)\times \mathrm{T}}$, where T is the number of triangles.

We illustrate the coding process with two dimensional equidistant triangulation of the rectangle $[0, L1] \times [0, L2]$. The author came up with the algorithm that uses the step size h_1 in x direction and the step size h_2 in y direction and produces the grid data $p \in \mathbb{R}^{2\times m}$ and $t \in \{1, 2 \dots m\}^{3\times \mathrm{T}}$.

Taking the lengths and the steps as the inputs, it produces the triangulation data

```
function [trmx,nodemx]=grd(l_1_2,h_1_2)
n=l_1_2(1)/h_1_2(1)+1; m=l_1_2(2)/h_1_2(2)+1;
```

We obtain the triangulation out of the equidistant grid in x and y directions by connecting each north-west and south-east point with the hypotenuse.

```
trmx=[];
for i=1:m*n

    if mod(i,n)~=0&&i+n<=m*n
trmx=[trmx;[i,i+1,i+n]];
trmx=[trmx;fliplr([i+1,i+n,i+n+1])];
    end


end
```

Afterwards we construct the node matrix, bearing in mind the geometry and the numeration.

```
for i=1:n*m
        nodemx(i,:)=[mod(i-1,n)*h_1_2(1),floor((i-1)/n)*h_1_2(2)];
end
end
```

5.2 Assembly of stiffness, mass matrix and a load vector

Within the main function we start entering the data. The dimension of the entered data would determine how the code reacts - either solves a single PDE or a system of PDEs. The one below illustrates the two equations case. For that we use the three-dimensional arrays.

```
A=[eye(2),eye(2)];A(:,:,2)=[0*eye(2),eye(2)];
 b=[0;0];
 b(:,:,2)=[0;0];
  c=0;c(2)=0;c(:,:,2)=[0 0];
   src=1;src(:,:,2)=0;
```

The integrals in weak formulations (3.31), (3.35), (3.36) to be computed locally on a unit triangle by means of variable affine transformations and, since several triangles are adjacent to one node, the element of any matrix in a system would be a sum of the local results. The function to be considered now deals with these local results.

```
function [lsm,lmm]=lsmx(BT,bt,A,b,c)

   gphi=[-1 -1;1 0;0 1];
 phiphi=1/24*ones(3,3)+1/24*eye(3);
  for ii=1:3
    for jj=1:3

lsm(ii,jj)=0.5*abs(det(BT))*(((A*(gphi(jj,:))')'*(inv(BT))*...
    (gphi(ii,:)*(inv(BT)))')+...
   abs(det(BT))*b.'*(gphi(jj,:)*(inv(BT)))'*(1/6)+...
   abs(det(BT))*c*phiphi(jj,ii);

    end
 end
      lmm(ii,jj)=abs(det(BT))*phiphi(jj,ii);

    end
```

In this way we've computed the elements of local stiffness, local mass matrix by considering three linear functions $\phi_1 = 1 - \xi - \eta$, $\phi_2 = \xi$, $\phi_3 = \eta$ and computing integrals locally on a unit cell $\{(\xi, \eta) : \xi > 0,\ \eta > 0,\ \eta < 1 - \xi\}$. This approach easily to be extended to 3 dimensions.

Now we are ready for the big cycle, which computes the elements of each block of our large sparse block-diagonal matrix (3.91). The cycle is fashioned in a way that it solves a single PDE as a special case when the corresponding data is entered.

The main idea here is to deal with node and triangle matrices obtained in the beginning, call out the *lsm* and *lmm* functions - stands for local stiffness and local mass matrices - and apply the main idea of assembly of a global matrix *gmx* out of several local matrices *lm*, namely the following algorithm (5.1).

for each $i-$th triangle in the triangle matrix $trmx$ and $j,k = 1..3$ do

$$gmx(trmx(i,j), trmx(i,k)) = gmx(trmx(i,j), trmx(i,k)) + lm(j,k). \tag{5.1}$$

Thus we start with initializing the arrays and coding the main cycle.

```
gsmxs=[];gsmx2=[];fs=[];intrs=[];sols=[];sol=[];
remove3s=[];periods=[];remove1s=[];


for d=1:size(A,3)
```

```
    f=zeros(size(nodemx,1),1);
    for g=1:size(A,2)/size(A,1)
%
gsmx=zeros(size(nodemx,1),size(nodemx,1));msmx=gsmx;
for i=1:size(trmx,1)
     BT=[nodemx(trmx(i,2),1),nodemx(trmx(i,3),1);
     nodemx(trmx(i,2),2),nodemx(trmx(i,3),2)]...
           -[nodemx(trmx(i,1),1),nodemx(trmx(i,1),1);
           nodemx(trmx(i,1),2),nodemx(trmx(i,1),2)];
       bt=[nodemx(trmx(i,1),1);nodemx(trmx(i,1),2)];
[lsm,lmm]=lsmx(BT,bt,A(:,(g-1)*size(A,1)+1:g*size(A,1),d),b(:,:,d),c(:,g,d));


    for j=1:3


     if g==size(A,2)/size(A,1)
      f(trmx(i,j))=f(trmx(i,j))
  +abs(det(BT))*src(:,:,d)*(1/6); end


     for k=1:3

       gsmx(trmx(i,j),trmx(i,k))=gsmx(trmx(i,j),trmx(i,k))+lsm(j,k);
            msmx(trmx(i,j),trmx(i,k))=msmx(trmx(i,j),trmx(i,k))+lmm(j,k);

        end
    end
end
```

The last lines directly implement the local-to-global algorithm (5.1).

The illustration is two dimensional, but can easily be extended to three dimensional by working with the corresponding structured triangulations and augmenting the upped bounds in the cycle. The next part deals with boundary conditions. This would modify our created arrays depending on type of BCs.

5.3 Boundary conditions

The types of boundary conditions considered here are Dirichlet (case 1 in the code) and Neumann (case 2) for two dimensional Lipshitz boundary (3.2.1), but other types can be easily coded. Given functions or normal derivatives on a boundary are coded as function-handles $@(x, y)$ [Tla], but can be easily extended in three dimensions if needed.

```
bnd1=1;bnd2=1;bnd3=1;bnd4=1;
bnd1(:,:,2)=1;bnd2(:,:,2)=1;bnd3(:,:,2)=1;bnd4(:,:,2)=1;
bnd1=bnd1(:,:,g);bnd2=bnd2(:,:,g);bnd3=bnd3(:,:,g);bnd4=bnd4(:,:,g);


dir={{@(x,y)-1,@(x,y)1,@(x,y)2,@(x,y)3},{@(x,y)0,@(x,y)0,@(x,y)0,@(x,y)0}};
```

```
neum={{@(x,y)0,@(x,y)0,@(x,y)0,@(x,y)0},{@(x,y)0,@(x,y)0,@(x,y)0,@(x,y)0}};
dir=dir{g};neum=neum{g};

intr=[];
for i=1:size(nodemx,1)


    l_1_2=[1 1];
    if nodemx(i,2)==0&&nodemx(i,1)~=0&&nodemx(i,1)~=l_1_2(1) bnd3=[bnd3;i];
    elseif nodemx(i,2)==l_1_2(2)&&nodemx(i,1)~=0&&nodemx(i,1)~=l_1_2(1) bnd4=[bnd4;i];
        elseif nodemx(i,1)==0 bnd1=[bnd1;i];
             elseif nodemx(i,1)==l_1_2(1) bnd2=[bnd2;i];
    else intr=[intr;i];end;
end
```

The last part creates the *bnd* arrays - the sets of nodes, which belong to the boundaries, coded by identifying the nodes' positions.

Once these have been identified, one can modify the system matrices or the right hand side depending on the type of BC.

Start with creating the boundary matrix.

```
bndmx=zeros(max([size(bnd1,1),size(bnd2,1),size(bnd3,1),size(bnd4,1)]),4);
   bndmx(1:size(bnd1,1),1)=bnd1;
    bndmx(1:size(bnd2,1),2)=bnd2;
     bndmx(1:size(bnd3,1),3)=bnd3;
      bndmx(1:size(bnd4,1),4)=bnd4;

remove3=[];remove11=[];period=[];nodrem=[];remove1b=[];remove1c=[];%remove1s=[];
```

Then goes the loop where we modify the right hand side and identify the positions to remove - all depending on type of boundary conditions. Illustation for Dirichlet (case 1) and Neumann (case 2). The loop can be easily extended to other types by adding other cases.

```
for j=1:size(bndmx,2)
    for i=2:size(bndmx,1)
        if bndmx(i,j)==0 break; end

         if bndmx(1,j)==1
 f=f-dir{j}(nodemx(bndmx(i,j),1),nodemx(bndmx(i,j),2))*gsmx(:,bndmx(i,j));
 remove1=[remove1,bndmx(i,j)];
 remove11=[remove11,bndmx(i,j)];
nodrem=[nodrem;j];
         end
if bndmx(1,j)==2
f=f+neum{j}(nodemx(bndmx(i,j),1),nodemx(bndmx(i,j),2))*msmx(:,bndmx(i,j));
intr=[intr;bndmx(i,j)];
           end

        end
    end
```

Afterwards the modification of arrays happens - cutting off and concatenations for large system matrix with the block diagonal structure.

```
 remove1_1=[remove1,remove1b];
          gsmx(:,remove1_1)=[];
        remove1_1=[];

  remove1=[remove1,remove1c];


%end_boundaries



intr=sort(intr);
period=sort(period);




      nodrem(:,2:3)=nodemx(remove11,:);

      dirn=dir(nodrem(:,1));


 %system
 gsmx2=[gsmx2,gsmx];
 if d==size(A,3) intrs=[intrs;intr+(g-1)*size(nodemx,1)];


     for i=1:size(remove11,2)
  sol(remove11(i))=dirn{i}(nodrem(i,2),nodrem(i,3));
     end

     remove3s=[remove3s;remove3+(g-1)*size(nodemx,1)];
     periods=[periods;period+(g-1)*size(nodemx,1)];
     remove1s=[remove1s,remove1+(g-1)*size(nodemx,1)];

    if size(sols,1)>1 zrs=size(nodemx,1)-size(sols',2);
    else zrs=0; end
     sols=[sols;zeros(zrs,1);sol'];
     sol=[];
     end
 end

    gsmxs=[gsmxs;gsmx2];


    fs=[fs;f];gsmx2=[];end
 %
 gsmxs(remove1s,:)=[];
fs(remove1s)=[];
```

Then we finally solve the linear system - the illustration is the backslash in Matlab - a built in elimination method, but can be replaced by any iterative method from numerical linear algebra, such as gradient method [Mei08] if need be.

```
solintrs=gsmxs\fs;

 sols(intrs)=solintrs;

sols(remove3s)=sols(periods);

if size(sols,1)==1 sols=sols'; end
```

After that the array *sols* is to be stored for post-processing. It can be plotted straight away in Matlab by using it together with the node and triangle matrix. Averaged or minimal-maximal values can be easily computed.

The line with the backslash operator can be adopted for the time dependent problems, non-linear problems or problems with the reduced dimension. This is usually a matter of modifying this line according to formulas obtained in chapter 4 for the proposed cases and calling out the whole function described in this section several times. Due to the repetitive nature we don't present them here.

5.4 Optimization of the parameters

In this section we are going to code the PSO method described in 4. In shall be as usual in m-file-Matlab function form: the function will take another function as an argument, as well as a stopping criterion, and produce all the necessary data, such as the optimal value, value of the arguments, where the optimum is achieved, terminal state of the particles, as well as in case of necessity it shall tell us the information about all the intermediate states of the particles, functions values and arguments.

The author has created an m-file *ps_try.m*, which has two arguments: k_1 - the number of iterations and k_2 - an initial state - either user defined or randomized. It produces the optimal value g of the functional $f(p)$, where p is the vector of parameters.

```
function g=ps_try(k1,k2)
```

The function optimizes the outcome of any function or procedure $f(p)$ - all we have to do is create the separate m-file $f(p)$ in the same folder.

We choose the method's parameters w, φ_p and φ_g - the choice is in accordance with w, φ_1 and φ_2 from the result (4.1.1) from chapter 4.

```
w=0.3;phi_p=1.2;phi_g=1.2;
```

The number of particles in the method (4.3), (4.4) is denoted as S. Many experiments have shown that 8 is a workable choice.

```
S=8;
```

Then we define two vectors of boundary values: $\mathbf{b}_{lo} \in \mathbb{R}^{D_p}$ and $\mathbf{b}_{up} \in \mathbb{R}^{D_p}$, where D_p is the number of parameters. What we enter here is crucial and the dimension will be determined out of this data and should be consistent with the dimension of the vector of parameters $\mathbf{p}$.

```
sz=size(b_lo); Dp=sz(1);
```

Now we use our second input k_2 - the initial condition. If there's no input, it should produce the vector of uniformly distributed random numbers - each component in interval $[b_{lo_i},\ b_{up_i}],\ i = 1..D_p$.

```
g=k2;
if length(k2)==0 g=b_lo+(b_up-b_lo).*rand(Dp,1); end;
```

Thus the first value of the functional $f(g)$ can be computed for the further comparison:

```
fg=f(g);
```

Now we are ready for the initial loop that creates the initial values of the necessary quantities for the first and all the subsequent loops. For each particle one randomly chooses the position vector and sets the winning states **p** as **x** for the first time:

```
for i=1:S

x(:,i)=b_lo+(b_up-b_lo).*rand(Dp,1);

p(:,i)=x(:,i);
```

Then we compute the value of the functional and compare it with the first value. It is necessary to use the format $fpi < fg$ instead of $f(p(:,i)) < f(g)$ to reduce the number of functional evaluations as for our applications it can be too costly.

```
fpi=f(p(:,i));
if fpi<fg g=p(:,i); end
```

Finally the initial velocity is chosen to be randomized as well:

```
a=-abs(b_up-b_lo);
b=abs(b_up-b_lo);
v(:,i)=a+(b-a).*rand(Dp,1);

end
```

Now we are ready for the main loop. Evaluate till we reach the number of iterations $k1$. For each particle initiate the parameters r_p and r_g as random uniformly distributed numbers $U(0,1)$.

```
k=0;
while k<k1

for i=1:S

r_p=rand(1,1);
r_g=rand(1,1);
```

Now implement the formula (4.3):

```
for d=1:Dp

   v(d,i)=w*v(d,i)+phi_p*r_p*(p(d,i)-x(d,i))+phi_g*r_g*(g(d)-x(d,i));



end
```

Here goes the bound control as in (4.8), (4.9). First try the opposite direction trick:

```
for d=1:Dp

if x(d,i)+v(d,i)<b_lo(d)||x(d,i)+v(d,i)>b_up(d)
v(d,i)=-v(d,i);

end
```

And if it doesn't work, try the boundary value trick:

```
if x(d,i)+v(d,i)<b_lo(d)
v(d,i)=b_lo(d)-x(d,i);

end

if x(d,i)+v(d,i)>b_up(d)
v(d,i)=b_up(d)-x(d,i);

end

end
```

After it's done we update the particle position, evaluate and compare functional values in order to determine the new best state and the winning state, as well as increase the iteration number by 1:

```
x(:,i)=x(:,i)+v(:,i);

 fxi=f(x(:,i));

if fxi<fpi p(:,i)=x(:,i);fpi=fxi;

 if  fpi<fg g=p(:,i);fg=fpi; end

end

end

k=k+1

end
```

This ends the while loop and the computer implementation of the method. In case of necessity it can solve the maximization problems (by changing the functional comparison procedure to the opposite inequality sign) and all the parameters are fully controllable. This setting chosen by author by analysing many sources has appeared to be consistent with all the experiments so far.

Now we are ready for the pattern storage part in order to conclude the present chapter.

5.5 Pattern storage

We store the patterns to form a system matrix of our discrete dynamical system.

We take the theory set in 4.3 to a programming level.

For this matter we create the test m-file *storage.m* that converts the real number into sequence of ones and minus ones and afterwords constructs the system matrix. Then the backwards conversion is applied.

We create a test m-file, which has two arguments: c stands for the integer to be converted to binary and tested as a pattern and $step$ represents the one-dimensional distance between two adjacent patterns to be stored.

```
function storage(c,step)
```

Then we allocate memory for the matrix A - in the test example let be $A \in \mathbb{R}^{20\times 20}$:

```
 A=zeros(20,20);
```

Now we consider the double loop - the one where we store patterns - elements of the grid $\{j \cdot \text{step}\}_1^{20}$ and converts them into binary vectors with minus ones and ones.
For this matter one can use the built-in operator 'decimalToBinaryVector' in order to obtain zeros and ones and afterwards replace zeros by minus ones.
The outer loop starts with:

```
for j=1:20
    x(:,j)=decimalToBinaryVector(step*j,20)';
```

and the inner loop is

```
for i=1:20
    if x(i,j)==0 x(i,j)=-1; end
end
```

and to end the outer loop the matrix is constructed by (4.41):

```
A=A+x(:,j)*x(:,j)';
 end
```

Then the pattern c - an input which we want to converge to a nearest fixed point is converted as well and becomes an initial value of our discrete dynamical system:

```
y(:,1)=decimalToBinaryVector(c,20)';
for i=1:20
    if y(i,1)==0 y(i,1)=-1; end
end
```

Then the system itself gets initiated with an update of zero-component to minus one. For the test example there are 10 iterations.

```
for k=1:10
    y(:,k+1)=sign(A*y(:,k));
  for i=1:20
    if y(i,k+1)==0 y(i,k+1)=1; end
end
end
```

And finally we take the last value and convert it back. For this matter we use Matlab operator 'binaryVectorToDecimal'.

```
k=y(:,10);
for i=1:20
    if k(i)==-1 k(i)=0; end
end

binaryVectorToDecimal(k')
end
```

The binary representation is not the best one when it comes to finding the closest pattern since the difference in value in certain positions yields the big difference when decoded. Hence the author came up with the following algorithm for binary representation that works with data from optimization procedure, codes it and feeds to the discrete dynamical system above.

For this matter we adopt the above for our patricular types of applications and create the function that takes the optimization data mx, $mx5$, M, N, win and the initial value of the (4.39) dynamical system vec as arguments and produces the closest value KK by means of this system.

Here N is the dimension of the particle, M is the number of criteria, arrays mx and $mx5$ have S columns, where S is the number of particles and mx is the concatenation of $\mathrm{N}+1$ particle positions and $mx5$ is the concatenation of $\mathrm{N}+1$ outputs. N stands for number of iterations. Augmented due to presence of initialization in PSO. win is the winning column to be stored. All columns can be stored if we reshape the matrices to form single vectors, but usually one winning particle trajectory with its solution outputs storage is enough.

```
function KK=stor_last_rand(mx,mx5,M,N,win,vec)

global mxx
mx_par=mx(:,win); mx_val=mx5(:,win);

n=M*45+N*45;
if length(mxx)==0
VV=zeros(n,1); idt=zeros(3*(M+N),1);
```

We start the loop that creates the patterns:

```
for j=1:2^n/100
    P=size(VV,2);
    if length(mx_val)<M||length(mx_par)<N break;end

    A=[mx_val(1:M);mx_par(1:N)]; D=[]; idtt=[];
```

This particular storage approach uses the standard form of the number $in.dec \cdot 10^{ten}$.

```
    for i=1:length(A)
        ten=0; in=0; dec=0; B=abs(A(i));
          if B>=1 && B<10
              in=floor(B)*sign(A(i)); dec=floor(10*(vpa(B,2)-floor(B))); end
        if B>=10
           while B>=10 B=B/10; ten=ten+1; end
           in=floor(B)*sign(A(i)); dec=floor(10*(vpa(B,2)-floor(B)));
        end

    if B<1 && B~=0
           while B<1 B=B*10; ten=ten-1; end
           in=floor(B)*sign(A(i)); dec=floor(10*(vpa(B,2)-floor(B)));
    end
```

It is important not to save the repeating patterns after rounding up hence the counter *kk*.

```
    idtt=[idtt; in;dec;ten]

kk=0;
      if i==length(A)
          for k=1:size(idt,2)

              double(double(idt(:,k))==double(idtt))

              if double(double(idt(:,k))==double(idtt))==ones(3*(M+N),1) kk=kk+1
           break; end
          end
      end
      if kk>0 mx_val(1:M)=[];mx_par(1:N)=[]; end


     if i==length(A)&&kk==0  idt=[idt idtt]

     end
```

The idea of one building block is to locate randomly ones (if positive) or minus ones (if negative) amongst their negative values for *in* and *dec* values and same in the middle for *dec* which is always positive. This is quite good for orthogonality or closeness to it if we want to fulfil the condition such as (4.3.1). The number of elements of the same sign then represents the integer we wish to code.

```
    if kk==0
     DD=[-sign(in)*ones(18,1);-ones(9,1);-sign(ten)*ones(18,1)];
     if in~=0 DD(randperm(18,9+abs(in)))=sign(in); end
     if in==0 DD(1:9)=-1;DD(10:18)=1; end
     if dec~=0 DD(randperm(9,double(dec))+18)=1; end
     if ten~=0 DD(randperm(18,9+abs(ten))+27)=sign(ten); end
     if ten==0 DD(28:36)=-1;DD(37:45)=1; end
     D=[D;DD];
```

Then the condition from (4.3.1) comes, but one can put here any other condition or, as concluded experimentally, let it loose a little bit for higher dimensional problems not to take ages to construct the patterns.

```
if  i==length(A)&&norm(VV'*D,1)<n
norm(VV'*D,1)
VV=[VV D]
mx_val(1:M)=[];mx_par(1:N)=[];
end
    end
end
    end

    mxx=VV;
end
```

The matrix *mxx* were defined as a global variable in the beginning so for one application one can run the costly procedure of constructing the patterns only once. Hence 'if length(mxx)==0' in the beginning.
Now the *vec* input to be coded as well:

```
vec_bin=[];
for i=1:size(vec,1)
 DD=[-sign(vec(i,1))*ones(18,1);-ones(9,1);-sign(vec(i,3))*ones(18,1)];
     if vec(i,1)~=0 DD(randperm(18,9+abs(vec(i,1))))=sign(vec(i,1)); end
     if vec(i,1)==0 DD(1:9)=-1;DD(10:18)=1; end
     if vec(i,2)~=0 DD(randperm(9,double(vec(i,2)))+18)=1; end
     if vec(i,3)~=0 DD(randperm(18,9+abs(vec(i,3)))+27)=sign(vec(i,3)); end
     if vec(i,3)==0 DD(28:36)=-1;DD(37:45)=1; end
     vec_bin=[vec_bin;DD];
end
```

Afterwords the procedure as in 'storage(c,step)' above is called out but it has to be rewritten so it takes the pattern matrix and the initial value of the discrete dynamical system as inputs.

```
k=stor_pat_bin(mxx(:,2:size(mxx,2)),vec_bin);
```

It produces the fixed point *k*. Then we decode it:

```
K=reshape(k,[45 M+N]); KK=[];
for i=1:M+N
    int=0;dec=0;ten=0;
    bb=K(1:18,i);

    if sum(bb>0)>sum(bb<0)
    int=sum(bb>0)-9;end
 if sum(bb>0)<sum(bb<0)
    int=-(sum(bb<0)-9);end
bb=K(19:27,i);
dec=sum(bb>0);
bb=K(28:45,i);

    if sum(bb>0)>sum(bb<0)
    ten=sum(bb>0)-9;end
 if sum(bb>0)<sum(bb<0)
    ten=-(sum(bb<0)-9);end
KK=[KK;(int+sign(int)*dec/10)*10^ten];
end
vpa(KK)

end
```

The obtained fixed point can be used now for further purposes. As we've already discussed in the end of chapter 4, for our applications - solve the PDE system (2.11), (2.18) with the obtained parameters only once (instead of many times if no pattern storage were ever introduced), check the measure (4.1) and make decision on whether it is close enough or the additional optimization is still required.

These simple steps will serve as the base for all the complex applications - number of iterations can be easily increased as well as the nature of the patterns as well as the stopping criterion.

The procedure can be called out several times in order to serve the purposes of our problems, when only part of the vector is given. The stopping criterion is then the closeness to the known values. Afterwords the solution process is to be called with the determined values out of this system only once.

This ends the description of the 'Pattern storage' coding aspects by means of a test code, as usual in the present chapter.

At the end of the day, in this chapter we have described different stages of our complex model with the short test codes. All the codes are fully controllable and can be further developed when necessary. The optimization and the pattern storage sections can be applied as user-created post-processing stages of the solution produced by some commercial industrial package or can be applied elsewhere even for non-differential problems.

Now we are finally ready for the last chapter with the applications - or in other words - several worked examples with the data given or to be determined - as states the nature of our problems. For each example a certain model-reduction procedure is going to be considered as in (4.38), as well as the optimization and data determination procedure as in (4.3), (4.4) as well as the pattern storage procedure as in (4.39), (4.41).

Chapter 6

Applications

In the last chapter the author obviously wants to show the numerical results out of frameworks presented in the previous chapters in form of pictures and tables.

We start with the solution of two problems with one or three reactions. For these problems the parameters are taken to be determined. This modelling stage has been discussed in chapters 2 and 3.

Then run the optimization process - choose several experimental results, and minimize the functional (4.1) by applying PSO.

Finally by using the stored intermediary results of the previous section, construct the discrete dynamical system as in (4.39) and run couple more experiments by addressing it first before considering the PDE solution framework.

Author has made a lot of numerical experiments in this topic and would like to present only small part of them in the thesis. Other results can be found in author's published works, see Appendix A.

The plots are going to be one, two and three-dimensional. All three settings can obviously be obtained out of the framework, discussed in previous chapters either by solving the 3D first and then consider cross-sectional plots or by solving 1D and 2D as separate problems, and whenever it is more convenient to use cross-sectional plots to show some specific results, we do that.

In all the following experiments the geometry of the combustion chamber is going to be represented either by a cylinder of radius 1 meter and height 4 meters or the axial symmetry may be considered depending on situation and then it gets solved in two dimensions on the rectangle. The inhomogeneous Dirichlet boundary conditions are to be set at inlet and the homogeneous Neumann boundary conditions to be set at all the remaining portions of the boundary. The plots of PDE solutions are going to be in 3 dimensions.

First section deals with one single-step reaction in a three-dimensional setting.

6.1 Solving PDEs for the single reaction model

For the beginning the model with one standard combustion reaction

$$2H_2 + O_2 \rightarrow 2H_2O \tag{6.1}$$

is considered.

The same framework can be applied for single reaction with different species (the determined parameter values would change).

This leads to 4 PDEs (3 species equations and one temperature equation). The species equation for the specie Y_k has it's form (2.11), as in chapter 2:

$$\rho\frac{DY_k}{Dt} = \nabla\cdot(\rho D\nabla Y_k) + \dot{\omega}_k. \tag{6.2}$$

The term $\dot{\omega}_k$ has its form as in (2.12)

$$\sum_{j=1}^{M}\dot{\omega}_{kj}. \tag{6.3}$$

The numeration for the species is as follows:

$$Y_1 := Y(H_2),\ Y_2 := Y(O_2),$$
$$Y_3 := Y(H_2O). \tag{6.4}$$

Then the matrix ν in the reaction mechanism (2.13) is $\begin{pmatrix}-2\\-1\\2\end{pmatrix}$, and the matrix $\dot{\omega}$ is $\begin{pmatrix}-2q_1w_1\\-q_1w_2\\2q_1w_3\end{pmatrix}$.

According to the Arrhenius law (2.14), (2.15),

$$q_1 = A_1e^{\frac{-E_1}{RT}}\left(\frac{\rho Y_1}{w_1}\right)^2\left(\frac{\rho Y_2}{w_2}\right). \tag{6.5}$$

Also we consider the axial velocity component w only. The diffusivity D is considered as a constant and equal for every specie in the system.

In this case, the species equations are (terms are divided by ρ):

$$\begin{cases}\dfrac{\partial Y_1}{\partial t} + w\dfrac{\partial Y_1}{\partial x} = D\triangle Y_1 - 2A_1e^{\frac{-E_1}{RT}}\dfrac{\rho^2}{w_1w_2}Y_1^2Y_2,\\ \dfrac{\partial Y_2}{\partial t} + w\dfrac{\partial Y_2}{\partial x} = D\triangle Y_2 - A_1e^{\frac{-E_1}{RT}}\dfrac{\rho^2}{w_1^2}Y_1^2Y_2,\\ \dfrac{\partial Y_3}{\partial t} + w\dfrac{\partial Y_3}{\partial x} = D\triangle Y_3 + 2A_1e^{\frac{-E_1}{RT}}\dfrac{\rho^2w_3}{w_1^2w_2}Y_1^2Y_2.\end{cases} \tag{6.6}$$

The temperature equation is subject to one more simplification: the heat capacity we shall consider independent of the specie or the temperature as in (2.19)-(2.22). The right hand side of the temperature equation is

$$-\sum_{k=1}^{N}(\triangle H_{f,k} + h_{s,k})\dot{\omega}_k, \tag{6.7}$$

which we divide into two sums:

$$-\sum_{k=1}^{N}\triangle H_{f,k} - \sum_{k=1}^{N}h_{s,k}\dot{\omega}_k. \tag{6.8}$$

Under the assumption that the heat capacity doesn't depend on the temperature

$$h_{s,k} = \int_{T_0}^{T}c_{pk}dT = (T-T_0)c_{pk} \tag{6.9}$$

or the species

$$\sum_{k=1}^{N} h_{s,k}\dot{\omega}_k = (T - T_0)c_p \sum_{k=1}^{N} \dot{\omega}_k = 0, \tag{6.10}$$

the only enthalpies we need are the enthalpies of formation to be determined from the tables [Mil00].

The temperature equation is as in (2.18):

$$\rho c_p \frac{DT}{Dt} = \nabla \cdot (\lambda \nabla T) - \sum_{k=1}^{N} \triangle H_{f,k}\dot{\omega}_k, \tag{6.11}$$

and now we rewrite it explicitly by using the right hand sides of the species equations (the diffusivity λ is considered as a constant):

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p w \frac{\partial T}{\partial x} = \lambda \triangle T + A_1 e^{\frac{-E_1}{RT}} \frac{\rho^2}{w_1^2 w_2} Y_1^2 Y_2 \left[2w_1 \triangle H_{f,1} + w_2 \triangle H_{f,2} - 2w_3 \triangle H_{f,3} \right]. \tag{6.12}$$

The resulting system of partial differential equations in its dimensional form now is:

$$\begin{cases} \dfrac{\partial Y_1}{\partial t} + w\dfrac{\partial Y_1}{\partial x} = D\triangle Y_1 - 2A_1 e^{\frac{-E_1}{RT}} \dfrac{\rho^2}{w_1 w_2} Y_1^2 Y_2, \\ \dfrac{\partial Y_2}{\partial t} + w\dfrac{\partial Y_2}{\partial x} = D\triangle Y_2 - A_1 e^{\frac{-E_1}{RT}} \dfrac{\rho^2}{w_1^2} Y_1^2 Y_2, \\ \dfrac{\partial Y_3}{\partial t} + w\dfrac{\partial Y_3}{\partial x} = D\triangle Y_3 + 2A_1 e^{\frac{-E_1}{RT}} \dfrac{\rho^2 w_3}{w_1^2 w_2} Y_1^2 Y_2, \\ \rho c_p \dfrac{\partial T}{\partial t} + \rho c_p w \dfrac{\partial T}{\partial x} = \lambda \triangle T + A_1 e^{\frac{-E_1}{RT}} \dfrac{\rho^3}{w_1^2 w_2} Y_1^2 Y_2 \left[2w_1 \triangle H_{f,1} + w_2 \triangle H_{f,2} - 2w_3 \triangle H_{f,3} \right]. \end{cases} \tag{6.13}$$

We subject the vector of unknown functions $\mathbf{X} = (Y_1, Y_2, Y_3, T)^T$ to the boundary and the initial conditions:

$$\begin{aligned} \mathbf{X}|_{\Gamma_D} &= \mathbf{X}_D, \\ \mathbf{X}|_{\Gamma_N} &= 0, \\ \mathbf{X}(0) &= \mathbf{X}_D e^{-x}, \end{aligned} \tag{6.14}$$

where Γ_D is the Dirichlet portion of the boundary (the inlet), $\mathbf{X}_D$ are the prescribed values at Γ_D and Γ_N is the Neumann portion of the boundary. The smooth exponential decay initial condition is considered.

In order to start the simulation one has to specify all the parameters values, e.g. to answer the question, what values are inside this vector of unknowns:

$$\left(\rho, c_p, D, \lambda, w_1, w_2, w_3, w, A_1, E_1, R, \triangle H_{f,1}, \triangle H_{f,2}, \triangle H_{f,3}, Y_{1,D}, Y_{2,D}, Y_{3,D}\right)^T. \tag{6.15}$$

The parameters are to be taken from different tables [Mil00], [Woo02], [20 04], [Tab20].

For the simulation we consider the density of mixture ρ to be $1[\frac{kg}{m^3}]$. The heat capacity for the mixture to be $1000[\frac{J}{kg \cdot K}]$.

The molecular diffusivity D is $5 \cdot 10^{-5}[\frac{m^2}{s}]$ and the thermal conductivity λ is $5 \cdot 10^{-2}[\frac{J}{m \cdot s \cdot K}]$. The atomic weights $w_{1,2,3}$ are straightforwardly to be taken from the periodic table: $w_1 = 0.002[\frac{kg}{mol}]$, $w_2 = 0.032[\frac{kg}{mol}]$, $w_3 = 0.018[\frac{kg}{mol}]$.

The constant axial velocity is set to $1[\frac{m}{s}]$.

The pre-exponential factor and the activation energy for the reaction (6.1) are something to be discussed in the optimization section. If we are lucky, the parameters can be found in Chemkin format table or found in publications [Woo02], [20 04]. These are the empirical results and not necessarily suit our experiments. As for now we choose $A_1 = 1.7 \cdot 10^{13}[cgs]$ and $E_1 = 47780[\frac{cal}{mol}] = 47780 \cdot 4.18[\frac{J}{mol}]$ as we've seen some estimates in [Wil10] or values for similar but not exact same reactions in [Woo02], [20 04].

In order to convert the pre-exponential factor A_1 accurately into the SI units, one has to see what the dimension of this factor is for the corresponding reaction. Each term of the dimensional species equation has a dimension of $[s^{-1}]$. By considering the reaction term of, for example, first equation, e.g $-2A_1 e^{\frac{-E_1}{RT}} \dfrac{\rho^2}{w_1 w_2} Y_1^2 Y_2$, the only dimensional multiplier left is $\dfrac{\rho^2}{w_1 w_2}$, which is in $\left[\dfrac{\cancel{kg^2} mol^2}{m^6 \cancel{kg^2}}\right]$. Thus the dimension of the pre-exponential factor is $\left[\dfrac{m^6}{s \cdot mol^2}\right]$. If it is given in the cgs units, 12 digits are to be taken off, which for the factor of 10^{13} is good news from the numerical point of view.

The ideal gas constant $\mathcal{R} = 8.314[\frac{J}{mol \cdot K}]$ should be divided by the mean atomic weight $\bar{w}$ in order to use it in the right hand sides of our equations: $R = \frac{\mathcal{R}}{\bar{w}}$.

The enthalpies of formation for the species H_2 and O_2 are by convention zero at the reference temperature. Then the enthalpy of formation for the H_2O is set to $-285800[\frac{J}{mol}]$.

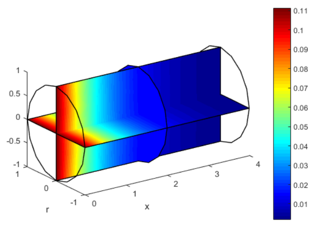
Finally, to prescribe the constant values on the Dirichlet portion of the boundary, we use the stoichiometric ratio $s = \dfrac{0.032}{2 \cdot 0.002} = 8$, and then solve the system for the mass fractions at stoichiometry:

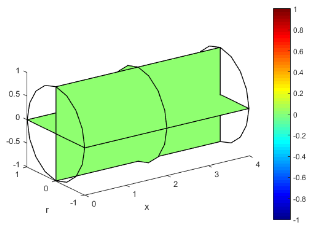
$$\begin{cases} \frac{Y_1}{Y_2} = \frac{1}{8}, \\ Y_1 + Y_2 = 1, \end{cases} \tag{6.16}$$

which gives us $Y_{1,D} = \frac{1}{9}$ and $Y_{2,D} = \frac{8}{9}$.
The product $Y_{3,D}$ is set to zero on this portion of the boundary.

By means of the spatial linear finite element discretization (3.56), (3.72) and the treatment of non-linear time depending system (3.84), (3.93), one gets the 4-second process graphical description with the time step of $\tau = 0.01$.

In figures 6.1-6.10 we present the snapshots for the distributions of $Y(H_2)$, $Y(O_2)$, $Y(H_2O)$ and T at times $t = 0, 1, 2, 3, 4$ s.


(A) $Y(H_2)$


(B) $Y(H_2O)$

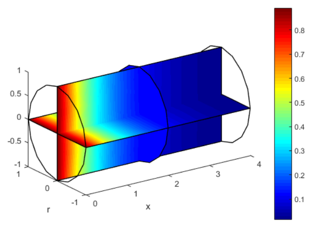

(C) $Y(O_2)$

FIGURE 6.1: Mass fractions at $t = 0$ s

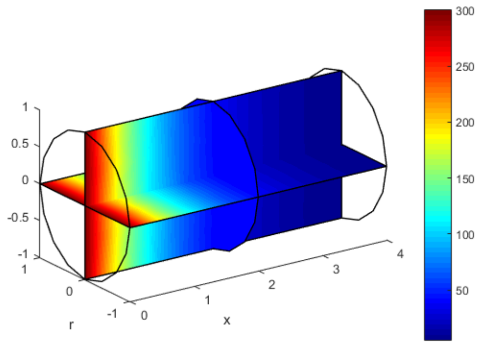
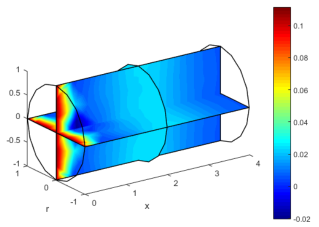
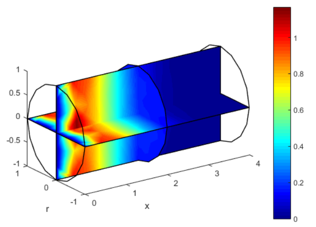


FIGURE 6.2: T at 0 s


(A) $Y(H_2)$


(B) $Y(H_2O)$

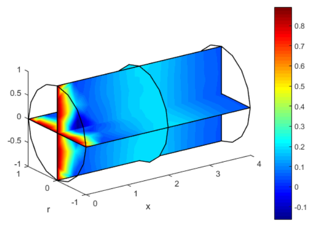

(C) $Y(O_2)$

FIGURE 6.3: Mass fractions at $t = 1$ s

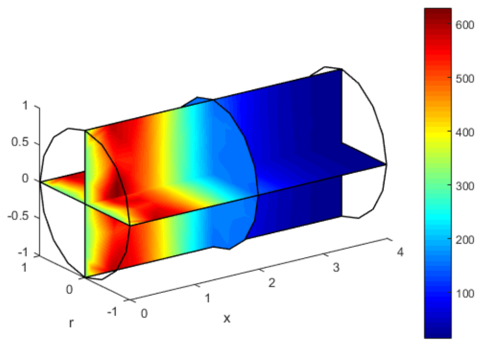
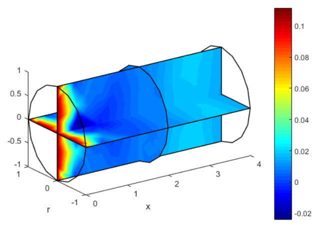
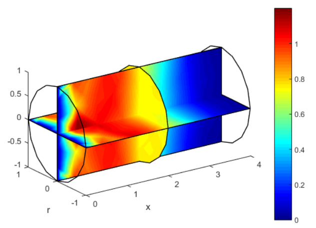


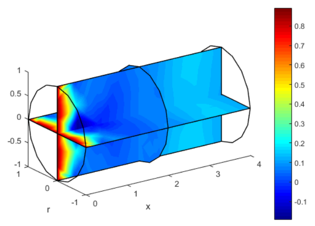
FIGURE 6.4: T at 1 s



(A) $Y(H_2)$



(B) $Y(H_2O)$



(C) $Y(O_2)$

FIGURE 6.5: Mass fractions at $t = 2$ s

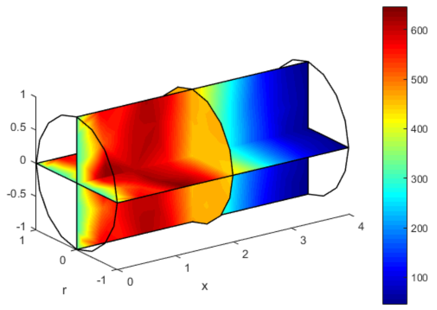
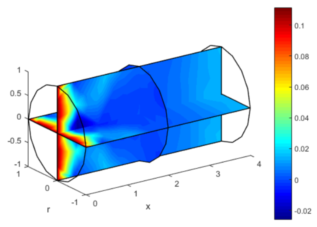
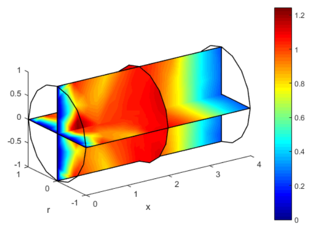


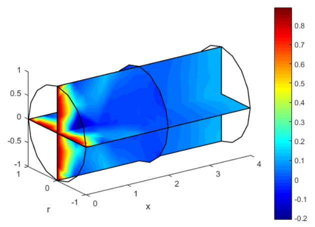
FIGURE 6.6: T at 2 s



(A) $Y(H_2)$



(B) $Y(H_2O)$



(C) $Y(O_2)$

FIGURE 6.7: Mass fractions at $t = 3$ s

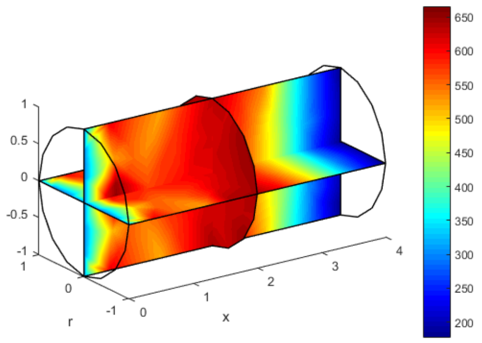
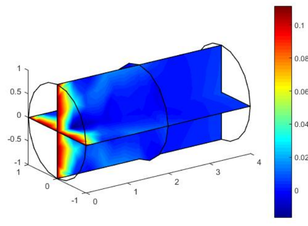
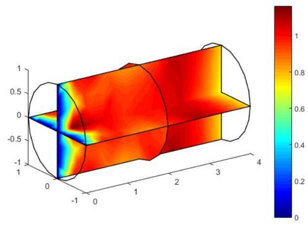


FIGURE 6.8: T at 3 s


(A) $Y(H_2)$


(B) $Y(H_2O)$

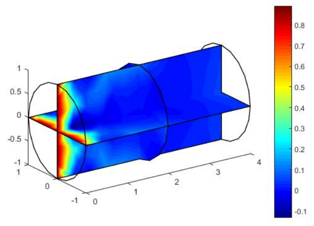

(C) $Y(O_2)$

FIGURE 6.9: Mass fractions at $t = 4$ s

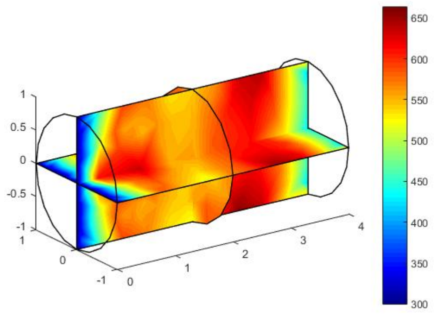


FIGURE 6.10: T at 4 s

According to the rates of corresponding reactions in (6.6) we obtain the decay of the first and the third mass fraction, growth (tendency to unity) of the second and the distribution of temperatures according to (6.12). For the chosen parameters (fully determined in this example), at 4 seconds, the maximal temperature is close to 600 K and the outlet temperature is close to 450 K.

The numerical simulation takes several minutes to perform. More on computational speeds in the last section of this chapter.

This application is good start for considering the next one 6.2, where we extend the number of reacting species. Also it motivates us for considering the application in section 6.3, whereas we have to optimize the parameters since the parameters in this one are chosen to be determined only for sake of actually being able to perform the computations and aren't at all commonly approved. The parameter region usually is given. More on that in sections 6.3, 6.4. But in the next section we consider the three reaction model. This transition would show us that the framework from chapter 2 works for any amount of species and reactions.

6.2 Solving PDEs for the several reaction model

The model with three combustion reactions

$$O_2 + CO \to O + CO_2,$$

$$CO_2 + H \to CO + OH,$$

$$H_2 + OH \to H_2O + H \tag{6.17}$$

is now considered. Same framework can be applied for any other three reaction mechanism of the same template but for different species (the determined parameter values would change).

By using the framework from chapter 2, this leads to 9 PDEs (8 species equations and one temperature equation).

The species equation for the specie Y_k has it's form (2.11) as in chapter 2:

$$\rho \frac{DY_k}{Dt} = \nabla \cdot (\rho D \nabla Y_k) + \dot{\omega}_k. \tag{6.18}$$

The term $\dot{\omega}_k$ has its form as in (2.12):

$$\sum_{j=1}^{M} \dot{\omega}_{kj}. \tag{6.19}$$

The numeration for the species is as follows:

$$\begin{gathered} Y_1 := Y(O_2), \\ Y_2 := Y(CO), \\ Y_3 := Y(O), \\ Y_4 := Y(CO_2), \\ Y_5 := Y(H), \\ Y_6 := Y(OH), \\ Y_7 := Y(H_2), \\ Y_8 := Y(H_2O). \end{gathered} \tag{6.20}$$

Then the matrix ν in the reaction mechanism (2.13) is $\begin{pmatrix} -1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$, and the matrix $\dot{\omega}$ is $\begin{pmatrix} -q_1 w_1 & 0 & 0 \\ -q_1 w_2 & q_2 w_2 & 0 \\ q_1 w_3 & 0 & 0 \\ q_1 w_4 & -q_2 w_4 & 0 \\ 0 & -q_2 w_5 & q_3 w_5 \\ 0 & q_2 w_6 & -q_3 w_6 \\ 0 & 0 & -q_3 w_7 \\ 0 & 0 & q_3 w_8 \end{pmatrix}$.

According to the Arrhenius law (2.14), (2.15),

$$q_1 = A_1 e^{\frac{-E_1}{RT}} \left(\frac{\rho Y_1}{w_1}\right)^1 \left(\frac{\rho Y_2}{w_2}\right)^1,$$

$$q_2 = A_2 e^{\frac{-E_2}{RT}} \left(\frac{\rho Y_4}{w_4}\right)^1 \left(\frac{\rho Y_4}{w_4}\right)^1,$$

$$q_3 = A_3 e^{\frac{-E_3}{RT}} \left(\frac{\rho Y_6}{w_7}\right)^1 \left(\frac{\rho Y_6}{w_7}\right)^1. \tag{6.21}$$

First power have been intentionally left as they are to show the nature of the formulas and compare with the previous case.
Also we consider the axial velocity component w only. The diffusivity D is considered as a constant and equal for every specie in the system:

In this case, the species equations are (terms are divided by ρ):

$$\begin{cases} \frac{\partial Y_1}{\partial t} + w\frac{\partial Y_1}{\partial x} = D\triangle Y_1 - \frac{q_1 w_1}{\rho}, \\ \frac{\partial Y_2}{\partial t} + w\frac{\partial Y_2}{\partial x} = D\triangle Y_2 + \frac{-q_1 w_2 + q_2 w_2}{\rho}, \\ \frac{\partial Y_3}{\partial t} + w\frac{\partial Y_3}{\partial x} = D\triangle Y_3 + \frac{q_1 w_3}{\rho}, \\ \frac{\partial Y_4}{\partial t} + w\frac{\partial Y_4}{\partial x} = D\triangle Y_4 + \frac{q_1 w_4 - q_2 w_4}{\rho}, \\ \frac{\partial Y_5}{\partial t} + w\frac{\partial Y_5}{\partial x} = D\triangle Y_5 + \frac{-q_2 w_5 + q_3 w_5}{\rho}, \\ \frac{\partial Y_6}{\partial t} + w\frac{\partial Y_6}{\partial x} = D\triangle Y_6 + \frac{q_2 w_6 - q_3 w_6}{\rho}, \\ \frac{\partial Y_7}{\partial t} + w\frac{\partial Y_7}{\partial x} = D\triangle Y_7 - \frac{q_3 w_7}{\rho}, \\ \frac{\partial Y_8}{\partial t} + w\frac{\partial Y_8}{\partial x} = D\triangle Y_8 + \frac{q_3 w_8}{\rho}. \end{cases} \tag{6.22}$$

The temperature equation is as in (2.18) and due to considerations of the previous section 6.1:

$$\rho c_p \frac{DT}{Dt} = \nabla \cdot (\lambda \nabla T) - \sum_{k=1}^{N} \triangle H_{f,k} \dot{\omega}_k. \tag{6.23}$$

We subject the vector of unknown functions $\mathbf{X} = (Y_i, T)^T,\ i = 1..8$ to the boundary and the initial conditions:

$$\begin{aligned} \mathbf{X}|_{\Gamma_D} &= \mathbf{X}_D, \\ \mathbf{X}|_{\Gamma_N} &= 0, \\ \mathbf{X}(0) &= \mathbf{X}_D e^{-x}, \end{aligned} \tag{6.24}$$

where Γ_D is the Dirichlet portion of the boundary (the inlet), $\mathbf{X}_D$ are the prescribed values at Γ_D and Γ_N is the Neumann portion of the boundary. The smooth exponential decay initial condition is considered.

As in the previous section 6.1, in order to start the simulation one has to specify all the parameters values:

$$(\rho, c_p, D, \lambda, w_i, w, A_j, E_j, R, \triangle H_{f,i}, Y_{i,D}), \tag{6.25}$$

$i = 1..8,\ j = 1..3.$
Same as in the previous section 6.1, the parameters are to be taken from different tables [Mil00], [Woo02], [20 04], [Tab20].

For the simulation we consider the density of mixture ρ to be $1[\frac{kg}{m^3}]$. The heat capacity for the mixture to be $1000[\frac{J}{kg \cdot K}]$.
The molecular diffusivity D is $5 \cdot 10^{-5}[\frac{m^2}{s}]$ and the thermal conductivity λ is $5 \cdot 10^{-2}[\frac{J}{m \cdot s \cdot K}]$.

The atomic weights $w_{1,..,8}$ are straightforwardly to be taken from the periodic table: $w_1 = 0.032[\frac{kg}{mol}]$, $w_2 = 0.028[\frac{kg}{mol}]$, $w_3 = 0.016[\frac{kg}{mol}]$, $w_4 = 0.044[\frac{kg}{mol}]$, $w_5 = 0.001[\frac{kg}{mol}]$, $w_6 = 0.017[\frac{kg}{mol}]$, $w_7 = 0.002[\frac{kg}{mol}]$, $w_8 = 0.018[\frac{kg}{mol}]$.
The constant axial velocity is set to $1[\frac{m}{s}]$.
The pre-exponential factor and the activation energy for the reactions (6.17) are:
$A_1 = 2.5 \cdot 10^{12}[cgs]$ and $E_1 = 47690[\frac{cal}{mol}] = 47690 \cdot 4.18[\frac{J}{mol}]$.
$A_2 = 1.5 \cdot 10^{7}[cgs]$ and $E_2 = -497[\frac{cal}{mol}] = -497 \cdot 4.18[\frac{J}{mol}]$.
$A_3 = 2.2 \cdot 10^{12}[cgs]$ and $E_3 = 3430[\frac{cal}{mol}] = 3430 \cdot 4.18[\frac{J}{mol}]$.

For this model we were lucky enough to find the same reaction mechanism in [20 04]. But this result is an empirical one suitable for some applications and the mechanism itself might be an idealisation for different models hence additional parameter optimization still should be considered.

The ideal gas constant $\mathcal{R} = 8.314[\frac{J}{mol \cdot K}]$ should be divided by the mean atomic weight $\bar{w}$ in order to use it in the right hand sides of our equations: $R = \frac{\mathcal{R}}{\bar{w}}$.
The enthalpies of formation for the species are $\triangle H_{f,1} = 0[\frac{J}{mol}]$, $\triangle H_{f,2} = -110435[\frac{J}{mol}]$, $\triangle H_{f,3} = 248919[\frac{J}{mol}]$, $\triangle H_{f,4} = -393129[\frac{J}{mol}]$, $\triangle H_{f,5} = 217778[\frac{J}{mol}]$, $\triangle H_{f,6} = 38957[\frac{J}{mol}]$, $\triangle H_{f,7} = 0[\frac{J}{mol}]$, $\triangle H_{f,8} = -241604[\frac{J}{mol}]$.

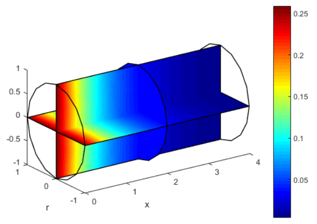
Finally, to prescribe the constant values on the Dirichlet portion of the boundary, we use the atomic weights' ratios and then solve the system for the mass fractions at stoichiometry:

$$\begin{cases} \frac{Y_1}{Y_2} = \frac{8}{7}, \\ \frac{Y_1}{Y_4} = \frac{8}{11}, \\ \frac{Y_4}{Y_5} = 44, \\ \frac{Y_4}{Y_7} = 22, \\ \frac{Y_7}{Y_6} = \frac{2}{17}, \\ Y_1 + Y_2 + Y_4 + Y_5 + Y_6 + Y_7 = 1. \end{cases} \tag{6.26}$$

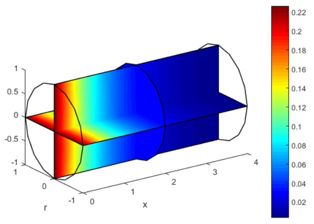
The pure products in this chain of reactions are set to zero on this portion of the boundary.

By means of the spatial linear finite element discretization (3.56), (3.72) and the treatment of non-linear time depending system (3.84), (3.93), one gets the 4-second process graphical description with the time step of $\tau = 0.01$.

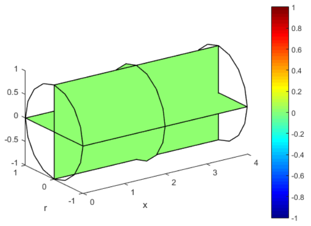
In figures 6.11-6.30 we present the snapshots for the distributions of $Y(O_2)$, $Y(CO)$, $Y(O)$, $Y(CO_2)$, $Y(H)$, $Y(OH)$, $Y(H_2)$, $Y(H_2O)$ and T at times $t = 0, 1, 2, 3, 4$ s.



(A) $Y(O_2)$

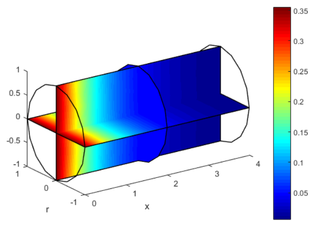


(B) $Y(CO)$

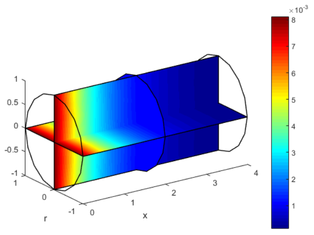


(C) $Y(O)$

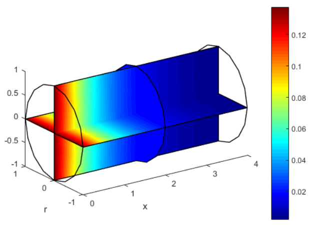
FIGURE 6.11: Mass fractions at $t = 0$ s



(A) $Y(CO_2)$

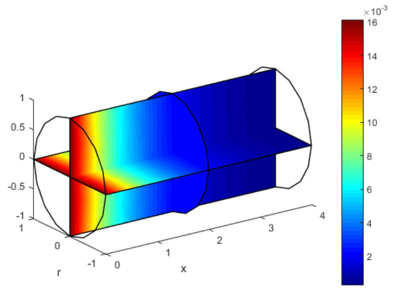


(B) $Y(H)$

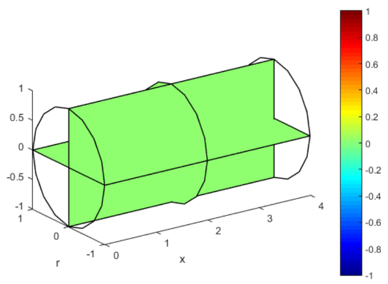


(C) $Y(OH)$

FIGURE 6.12: Mass fractions at $t = 0$ s



(A) $Y(H_2)$



(B) $Y(H_2O)$

FIGURE 6.13: Mass fractions at $t = 0$ s

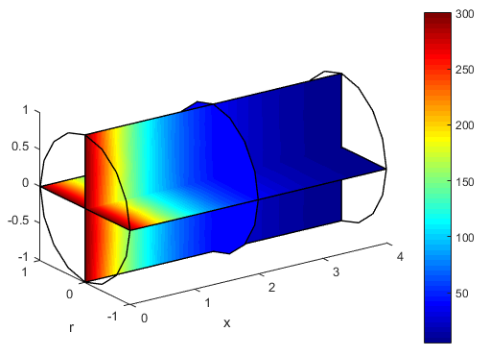
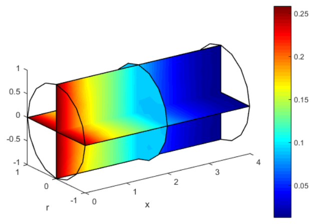
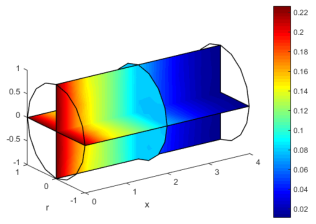


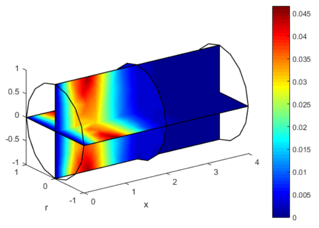
FIGURE 6.14: T at 0 s



(A) $Y(O_2)$

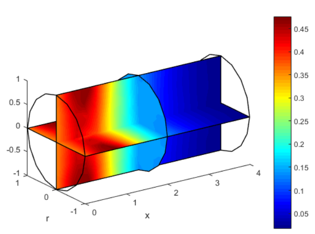


(B) $Y(CO)$

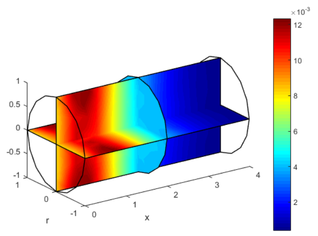


(C) $Y(O)$

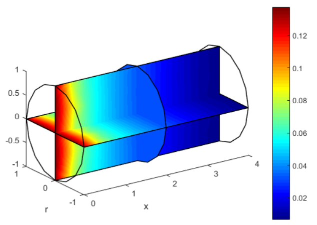
FIGURE 6.15: Mass fractions at $t = 1$ s



(A) $Y(CO_2)$

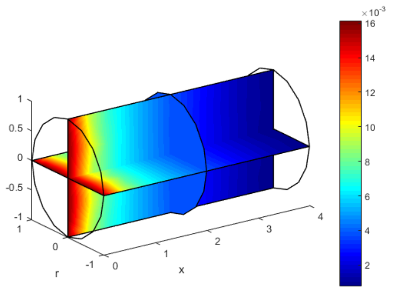


(B) $Y(H)$

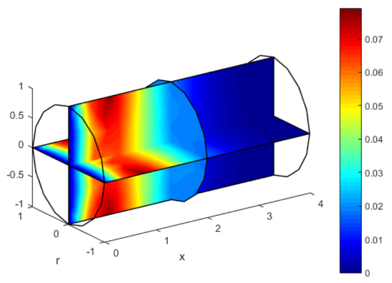


(C) $Y(OH)$

FIGURE 6.16: Mass fractions at $t = 1$ s



(A) $Y(H_2)$



(B) $Y(H_2O)$

FIGURE 6.17: Mass fractions at $t = 1$ s

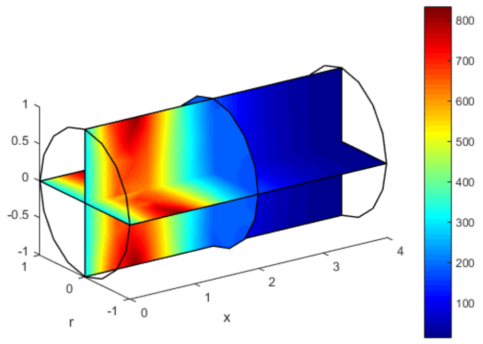
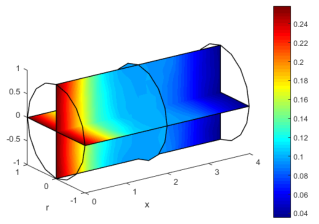
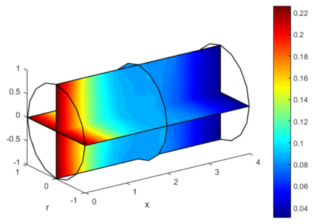


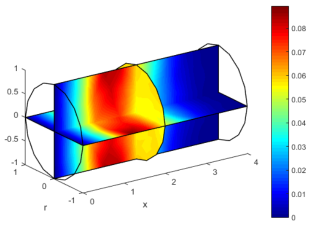
FIGURE 6.18: T at 1 s



(A) $Y(O_2)$

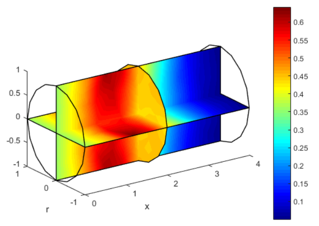


(B) $Y(CO)$

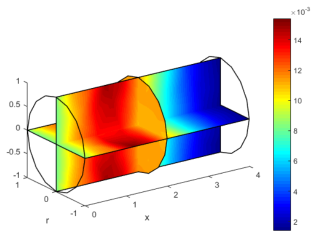


(C) $Y(O)$

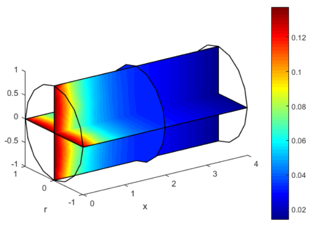
FIGURE 6.19: Mass fractions at $t = 2$ s



(A) $Y(CO_2)$

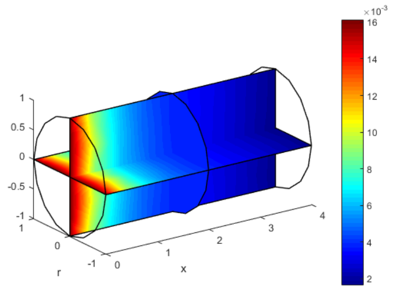


(B) $Y(H)$

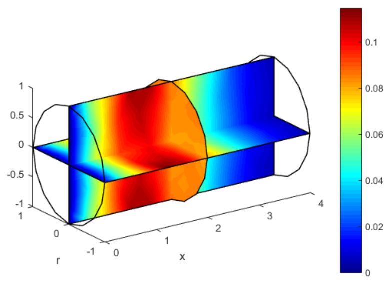


(C) $Y(OH)$

FIGURE 6.20: Mass fractions at $t = 2$ s



(A) $Y(H_2)$



(B) $Y(H_2O)$

FIGURE 6.21: Mass fractions at $t = 2$ s

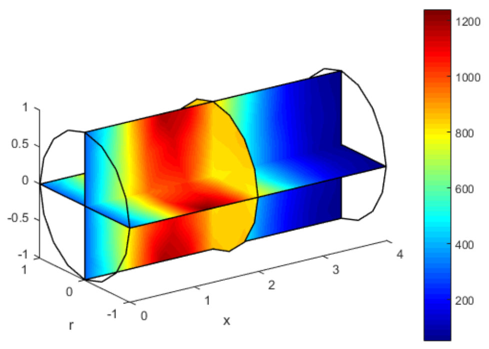
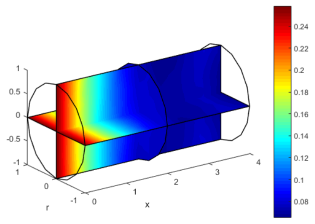
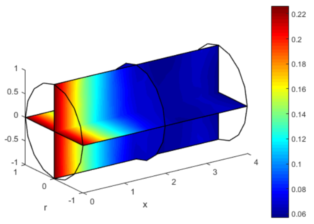


FIGURE 6.22: T at 2 s


(A) $Y(O_2)$


(B) $Y(CO)$

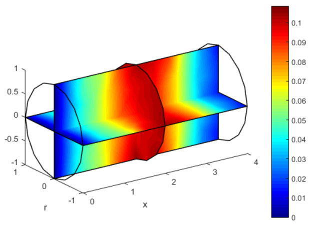
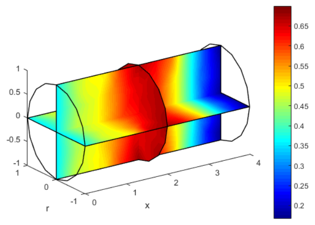
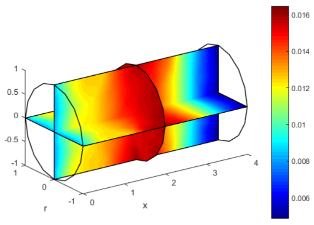

(C) $Y(O)$

FIGURE 6.23: Mass fractions at $t = 3$ s


(A) $Y(CO_2)$


(B) $Y(H)$

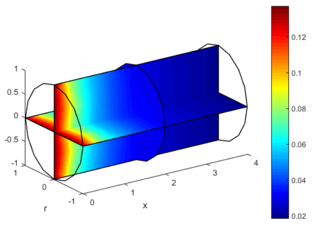
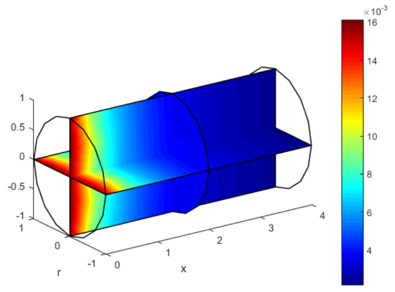

(C) $Y(OH)$

FIGURE 6.24: Mass fractions at $t = 3$ s


(A) $Y(H_2)$

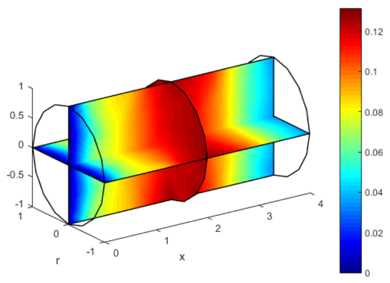

(B) $Y(H_2O)$

FIGURE 6.25: Mass fractions at $t = 3$ s

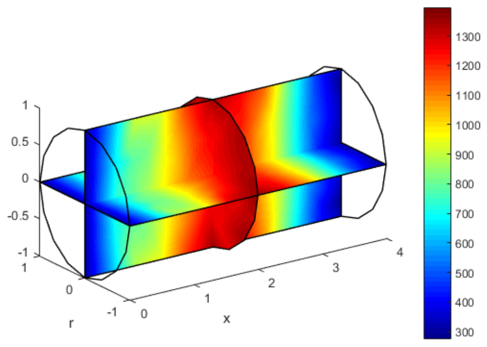
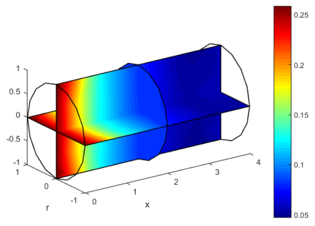
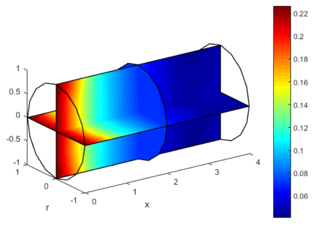


FIGURE 6.26: T at 3 s


(A) $Y(O_2)$


(B) $Y(CO)$

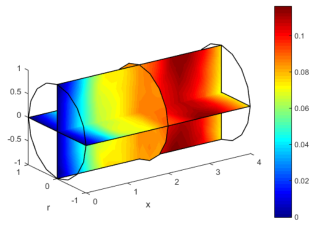
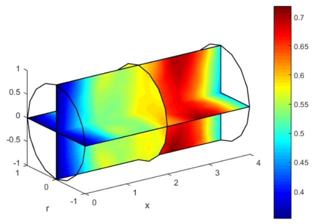
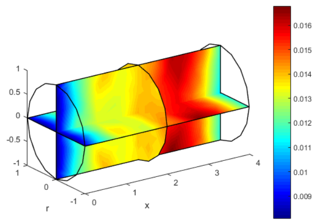

(C) $Y(O)$

FIGURE 6.27: Mass fractions at $t = 4$ s


(A) $Y(CO_2)$


(B) $Y(H)$

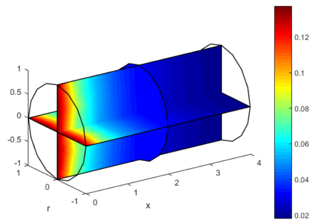
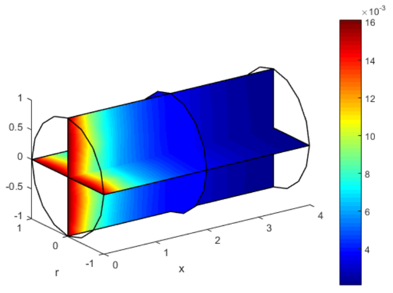

(C) $Y(OH)$

FIGURE 6.28: Mass fractions at $t = 4$ s


(A) $Y(H_2)$

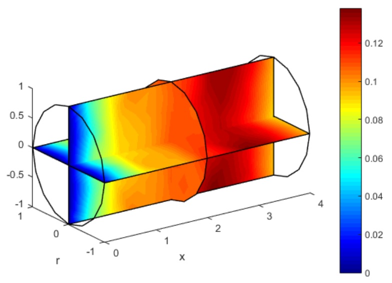

(B) $Y(H_2O)$

FIGURE 6.29: Mass fractions at $t = 4$ s

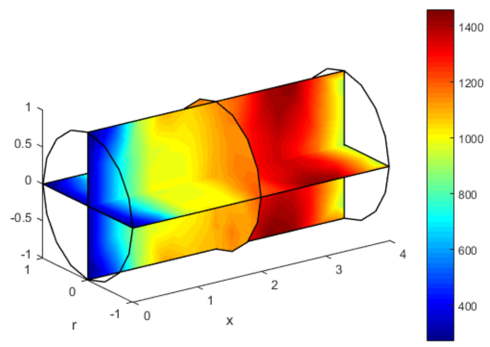


FIGURE 6.30: T at 4 s

According to the rates of corresponding reactions in (6.22) we obtain the mass fraction evolutions and the distribution of temperatures according to (6.23). For the chosen parameters (fully determined in this example), at 4 seconds, the maximal temperature is close to 1400 K and the outlet temperature is close to 1000 K.

The numerical simulation takes more than several minutes but still less than an hour to perform. More on computational speeds in the last section of this chapter.

This application along with the previous one motivates us for considering the next section, which is the parameter optimization.

6.3 Parameter optimization for 3D model with one or several criteria

We would like to start with the first optimization application. This would be the three dimensional setting with a single reaction as in section 6.1, but the parameters are not explicitly given this time and the interval thereof is estimated.

For each experiment we consider the reduced order solutions as in 4.2 and perform it in a way that is is easy to store the data as in 4.3.

The region is deducted from the fact that there are similar reactions (with similar list of species, but not exactly the same), that can be found in Chemkin format table [20 04] or other published works, which are empirically obtained. The author then gathers data from these tables of works and estimates the interval. In figures 6.31a - 6.33b the one is represented by the red square. Following the notation from Chapter 5, our vectors of boundary values $\mathbf{b}_{lo}$ and $\mathbf{b}_{up}$ are respectively $\begin{pmatrix} 1.7 \cdot 10^{-3} \\ 47780/100 \end{pmatrix}$ and $\begin{pmatrix} 1.7 \cdot 10^{1} \\ 47780 \cdot 100 \end{pmatrix}$.

The division, multiplication by a hundred as well as the power of one have intentionally been left as they are so the nature of obtaining the parameter interval is visible.

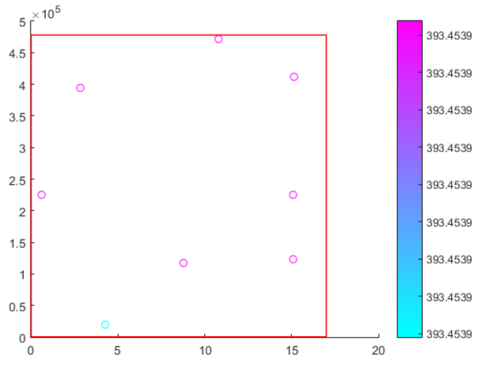
This estimates are obtained by gathering as much as possible data for similar reactions since this particular reaction hasn't been found there in its very form. For different reaction mechanisms this approach has given good results.

For this particular application we compare the maximal temperature $\max_{\mathbf{x} \in \Omega} T(\mathbf{x})$ with the experimental value of 740 K.

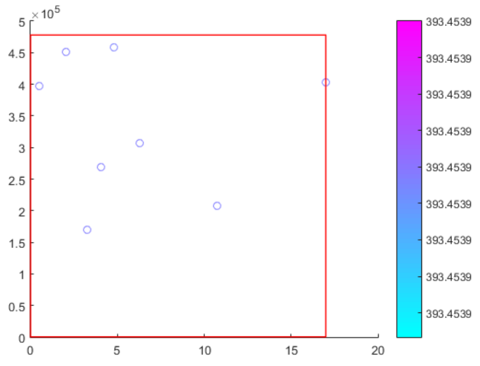
The functional choices had been described in Chapter 4 and is due to the formula (4.1).

One can see in figures 6.31a - 6.33b the motion of two dimensional particles. The blue dot represents the winning state. The winning parameter values are approximately $\begin{pmatrix} 10.44 \\ 5235.25 \end{pmatrix}$.

The measure in this example has been obtained and kept small (less than 1 K).

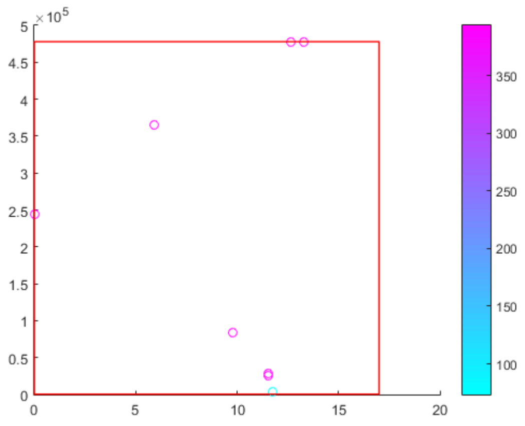


(A) 1st iteration

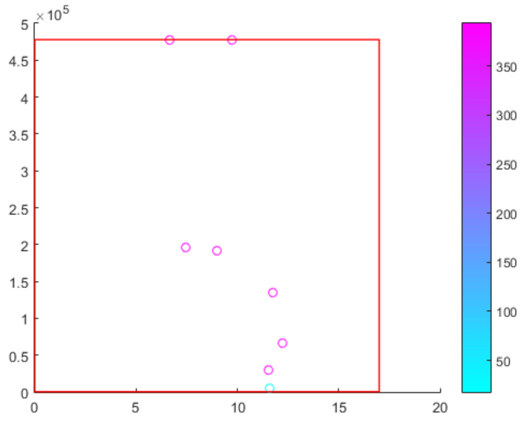


(B) 3rd iteration

FIGURE 6.31: Particle positions at $k = 1, 3$

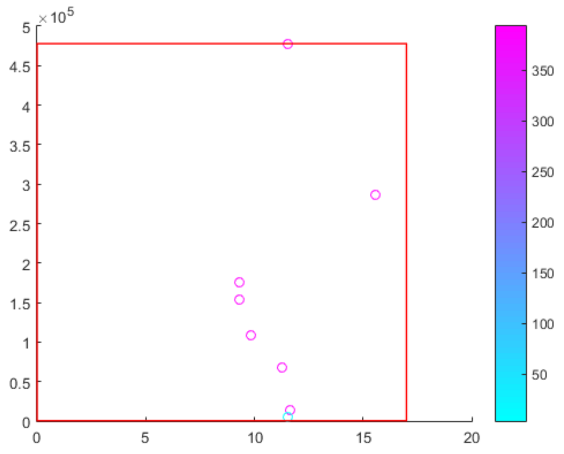


(A) 5th iteration

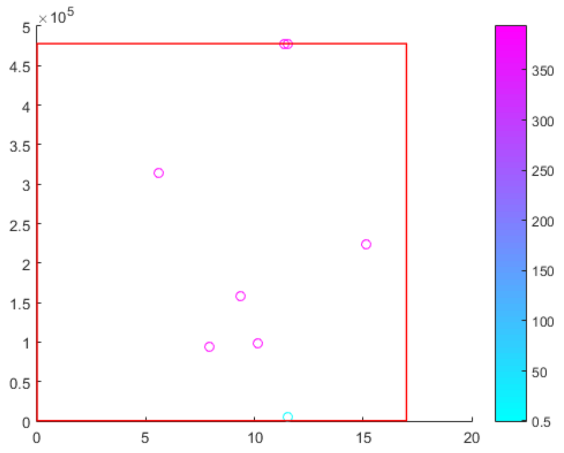


(B) 7th iteration

FIGURE 6.32: Particle positions at $k = 5, 7$



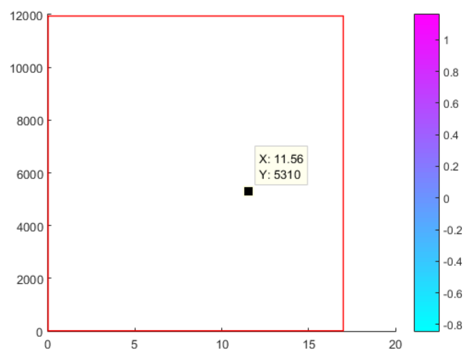
(A) 15th iteration



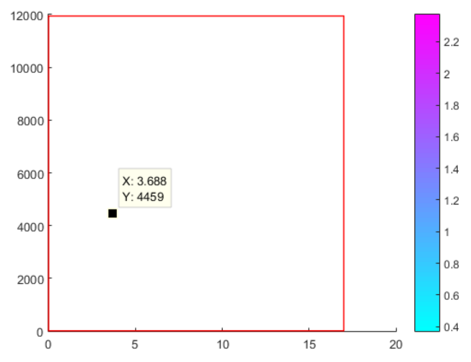
(B) 30th iteration

FIGURE 6.33: Particle positions at $k = 15, 30$

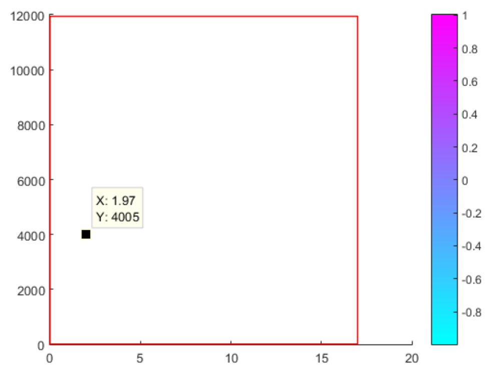
As discussed previously, for the proposed goals we are totally fine with only one argmin that brings the modelling data close to experimental data. The series of further numerical experiments 6.34a-6.34c can show that the close to zero distance can be obtained at different points. The region at these figures is localised. The nature of the region depends on how the initial problem had been parametrized and is not a matter of current discussion since finding of one feasible variant usually stops the costly procedure and the obtained vector of parameters is proposed as the optimal one and further stages of modelling are considered. It is so mostly due to the fact that the global reaction mechanism, regulating the source term, itself is an approximation of the real-life situation of much more complex reaction mechanisms.



(A) winning particle I



(B) winning particle II



(C) winning particle III

FIGURE 6.34: Winning states for different initializations

The process of obtaining this measure took around 6-10 hours when the initial guess wasn't close enough. When the initial guess was close enough, the time could be substantially reduced, however for this application the random positions are chosen at the beginning. This motivates us to use the model reduction techniques when necessary from chapter 4 section 4.2 and the pattern storage application from 4 section 4.3. The former would reduce the amount of hours whilst keeping the resulting quantities close and the latter shall define the auxiliary problem to be called out beforehand.

The algorithm 4.2.1 introduced in chapter 4 for the optimization with the reduced states can be applied when necessary. The motivation is to compute the PDE solutions for the optimization purposes in the reduced basis most of the times and come up with several stages of the functional value rechecking. This should be a time saver to a certain extent. The figure 6.35 illustrates its behaviour.

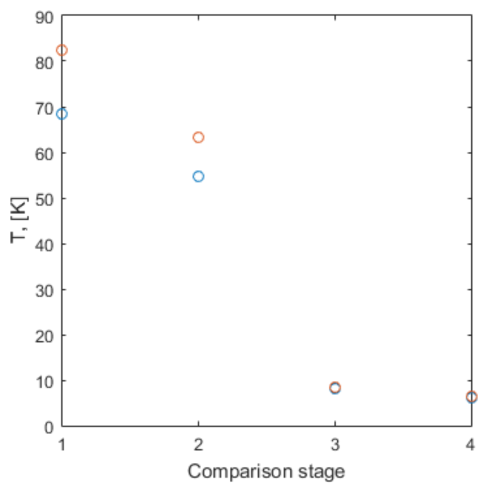


FIGURE 6.35: Four stages of comparison

This is a simple illustration on how the optimization procedure can solve the PDE system mostly with reduced states. The number of PSO iterations (experimentally, usually N is around 30 is enough) is split into several parts (N_S - number of stages - four on 6.35) in order to apply the algorithm 4.2.1. The matrix of snapshots and its SVD is recalculated at each stage. According to the algorithm these are the only times when the solution gets calculated in full basis and thus for N $\gg$ N_S it gets calculated in reduced bases only.

For the considered types of models, experimentally, it is possible to reduce the dimension by $10\% - 20\%$ and number of stages N_S around $4 - 6$ is enough. This is a time saver, but the whole procedure still takes hours to perform and motivates us to consider the pattern storage concept in 6.4. However when the experiments are not connected, we consider completely new model, or it gets substantially modified, etc, the PCA is always a good way to start.

The problem can become more complex when we introduce several criteria. Theoretically this concept can be used to solve the whole closest curve or surface finding task by using the proposed techniques. This would be the topic of further researches and to show that this works in the present paper we extend the number of criteria by one. Same task will be solved in storage pattern section 6.4 by using present results.
So we consider the PSO iterations with 2 criteria:
$T_{max} = 740$ K and $T_{out} = 450$ K.
30 iterations of minimizing the functional (4.1) takes the same time as in case with one criterion but since the dimension has been increased the value of the functional can be not as close to zero as in case with one criterion. Hence additional iterations are required if one wishes to keep the measure between experimental and modelling data small and this motivates us even more to consider pattern storage in 6.4 in case if there are more parameter optimization tasks with similar model to go.
The convergence obtained obviously mean that we approximately reach the desired temperature values. We'll show the one dimensional projections of these 3D graphics after the pattern storage procedure in the next section.

6.4 Pattern storage results

The nature of figures 6.31a - 6.33b for section 6.3 is the processing of the optimization intermediate states. This has good geometrical representation when the dimension and the number of particles is kept small. However when this in not the case (lots of degrees of freedom to be compared amongst in our functional) there is a lot of unstored and unprocessed data.

The idea that author came up with for the last type of applications is that this data can be stored in a certain smart way for later applications.

In this simple example we store the data of the evolution of the winning (blue) particle from 6.31a - 6.33b. Form a discrete dynamical system as in (4.39). Then call it out with the new experimental data and unknown (but guessed as an initial state) parameter values. Then see how close is the result obtained by the system (4.39) in terms of measure (4.1). In case of necessity it can be used as a starting guess in the PSO (4.3), (4.4) and not apply the randomization in the beginning. We show how the computational time can be substantially reduced.

The first experiment is as follows: we store the winning trajectory in (4.3), (4.4) along with the maximal temperatures at last time step.

We create the discrete dynamical system by applying (4.41) and running 5.5.

Then assume then the next experiment has been held with the new value of 640 K. Remember that 30 iterations for 740 K took around $6 - 10$ machine computation hours depending on the initial randomization.

The computational time of creating the patterns depends on condition in the result 4.3.1 and randomization but usually takes several minutes for chosen dimension.

The iterations of the discrete dynamical system takes several seconds.

We don't know the parameters - all we know is the new maximal temperature value at last time step, which is 640 K. We choose the values in the allowed intervals for two parameters and feed the system the matrix $\begin{pmatrix} 6 & 4 & 2 \\ 1 & 0 & 1 \\ 4 & 1 & 4 \end{pmatrix}$. Since $\mathbf{b}_{lo}$ and $\mathbf{b}_{up}$ are taken from section 6.3 and respectively are $\begin{pmatrix} 1.7 \cdot 10^{-3} \\ 47780/100 \end{pmatrix}$ and $\begin{pmatrix} 1.7 \cdot 10^{1} \\ 47780 \cdot 100 \end{pmatrix}$,

this is a good place to start.

After several iterations of the processes 5.5 the machinery has produced the vector $\begin{pmatrix} 640 \\ 10 \\ 8200 \end{pmatrix}$, which is too good to be true so solving the system of PDEs (2.11), (2.18) with these parameters was necessary which gave the close enough value between 635 and 645 Kelvins. The figure (6.36) shows the closeness to the temperature profile.

In table 6.4 we see how we have substantially reduced the computational time for the new experiment. The times are in seconds as results of tic and toc in Matlab.

T value/method	PSO	DDS with PC	DDS	PDE
740 K	37412	-	-	-
640 K	-	160	-	119.4
700 K	-	-	32.2	119.3

It all started (first row in table 6.4) with the full PSO for 740 K which took around 10 hours to perform on a powerful but not super computer.

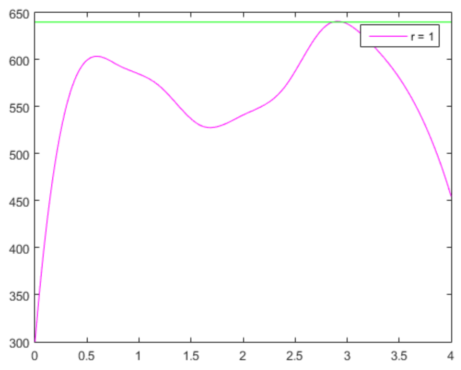


FIGURE 6.36: T at 4 s - max T around 640 K

The sequence of experiments afterwards is considered whereas we do not wish to wait that long and use the leftover information of the optimization trajectories.

For illustration we considered two subsequent experiments with finding the parameters for solution with maximal temperatures of 640 and 700 K (second and third rows of table 6.4).

DDS stands for iterating the discrete dynamical system and the one with PC (stands for pattern creation) is when we create the patterns as in 5.5 first. This took between 2 and 3 minutes and gave an approximation since the standard forms of numbers had been considered and by feeding to the system the trajectory of only one particle there is no guarantee the chosen maximal values are close to the values on the trajectory.

Therefore the one last solution of PDE (the last column) were called out with the wait time of around about 2 minutes.

For the experiment in the 3rd row and all the hypothetical subsequent experiments there is no need for pattern creation so it takes around 30 seconds to iterate the DDS several times and find the approximated parameter values.

In comparison with lots of hours if we had to optimize the parameters every time this is a significant time saver.

In the next application we consider the two criteria for optimization - maximal and outlet temperature with the model as in the last part of section 6.3. This would demonstrate that for more complicated problem the pattern storage is a life saver for the sequence of subsequent experiments and if there are even more sophisticated potential applications the framework can be applied as well.

We consider the PSO iterations with 2 criteria:
$T_{max} = 740$ K and $T_{out} = 450$ K. This is possible scenario to converge onto out of the solutions with determined parameters in sections 6.1, 6.2. And by running the optimization procedure wisely (applying the pattern storage procedure at the same time) this has given us lots of trajectories to store.

Therefore, analogously to the first part of the section, we consider the next possible parameter optimization task with $T_{max} = 670$ K and $T_{out} = 430$ K.

Again: the parameters can be chosen somewhat randomly or according to the last good experiment if any. For this illustration we choose the matrix $\begin{pmatrix} 6 & 7 & 2 \\ 4 & 3 & 2 \\ 1 & 0 & 1 \\ 4 & 1 & 3 \end{pmatrix}$.

This matrix sets standard forms of three numerical values: first two rows - what we wish to obtain and last two rows - the chosen values of the parameters. These we wish to be detrmined after we feed it to the DDS. Again: parameters are obviously chosen to belong to the feasible interval.

What our machinery has to obey is that any produced decoded value has to belong to its feasible interval. This is obtained by a simple bound control as in (4.8), (4.9) but when it comes to DDS, which has been designed to work very fast, the bound control can discard all the values beyond the given interval and run the process which lasts several seconds only several times till it gives out a set of feasible values.

Experiments show that the more we teach the system with values in interval, the better it stays inside of it.

After several iterations and running the PDE solver we have found the necessary close enough projection. The figure (6.37) graphically compares the temperature hypersurface projection (magenta curve) with the lines corresponding to the desired values.

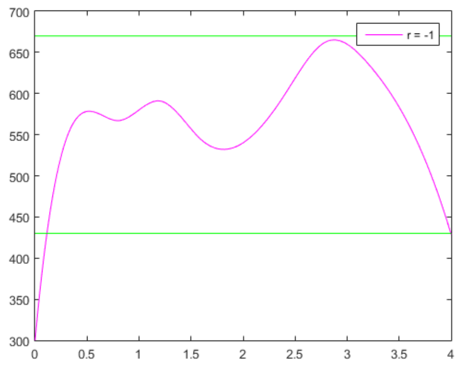


FIGURE 6.37: T at 4 s - max T around 670 K, out T around 430 K

We construct the table with tics tocs from Matlab as well.

T values/method	PSO	DDS with PC	DDS	PDE
740, 450 K	25816.8	-	-	-
670, 430 K	-	868.9	-	105.9
700, 440 K	-	-	277.7	107.5

The explanation of this table is similar to the previous one, but instead of one criterion there are two.

Again, we begin with the first row in table with the full PSO for 740 K, 450 K this time, which took around 7-8 hours to perform on a powerful but not super computer. The few hour difference (as it is if we compare it with the first table) can always be explained by the random nature of the initial guess in (4.3), (4.4).

Again, the sequence of experiments afterwards is considered whereas we do not wish to wait that long and use the leftover information of the optimization trajectories.

For illustration we considered two subsequent experiments with finding the parameters for solution with temperatures pairs of 670, 430 and 700, 440 Kelvins (second and third rows of the table).

DDS stands for iterating the discrete dynamical system and the one with PC (stands for pattern creation) is when we create the patterns as in 5.5 first. This took about 15 minutes and gave an approximation since the standard forms of numbers had been considered and by feeding to the system the trajectory of only one particle there is no guarantee the chosen maximal values are close to the values on the trajectory. Here the dimension doubled and the binary representation took a little bit more time to create.

Therefore the one last solution of PDE (the last column) were called out with the wait time of around about 2 minutes.

For the experiment in the 3rd row and all the hypothetical subsequent experiments there is no need for pattern creation so it takes around 4 minutes to iterate the DDS several times and find the approximated parameter values.

Again, in comparison with lots of hours if we had to optimize the parameters every time this is a significant time saver.

One can conclude that for two criteria the timings haven't been significantly increased (even decreased at some point but this is as usual a matter of an initial guess) and in case of necessity one can increase the number of criteria and the codes in 5.5 have been created such as we can easily implement it.

Formulation of the problem whereas the number of criteria is greater than the number of parameters is also possible after examining the behaviour of the solutions for different values of parameters. This can lead to values of measure not as close to zero. That is also the case for the 'improper' choice of criteria values when the numbers coincide or the number of criteria is smaller than the number of parameters. The nature of our considered problems however (comparison of modelling and experimental data and modification of models therefore) makes us analyse the experimental data first and then formulate the optimization problem. In case when the measure is obtained inadmissible one may consider reformulating the optimization problem. The figure 6.38 shows how the measure can still be obtained in case when the number of criteria is larger (we've added the third one - the temperature value in the middle) but the formulation of the corresponding optimization problem and the execution of the algorithms is still possible.

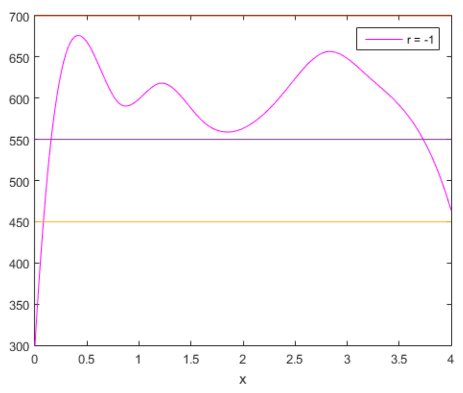


FIGURE 6.38: T at 4 s - desired max T - 700 K, out T - 450 K, mid T - 550 K.

In similar fashion we can continue the experiments for the chosen types of models, which shall be a matter of author's further researches, or can apply the obtained machinery for different types of models (not even necessarily described by PDEs).

This ends the applications section. We have chosen illustrative examples to show how the machinery works and given pictorial and tabular representations of our results.

The formulation of the conclusions of the whole thesis is the subject of the last pages as well as the list of author's publications and the list of literature.

Chapter 7

Conclusions

Sticking to the plan proposed in chapter 1 led to results applicable in the field of combustion and chemical kinetics experiments. The techniques used for that are obviously applicable elsewhere as well since the code had been created such that its separate parts can be used as a part of another code for the field or in other various fields.

In particular, the framework when we know the list of chemical reactions in the model and don't know some of their parameters but do know how to estimate their regions has been created. Very briefly at the end of the day:

In chapter 2 we've learned how to take the list of forward chemical reactions and construct the source terms in the equations of the PDE system. The other classical terms of the equations of the PDE system have been discussed as well.

In chapter 3 we've chosen universal methods to be safe to obtain the feasible solutions of the PDE system and have presented the necessary theory of weak solutions, function space approximation, finite element method and the linearisation.

In chapter 4 we went step further, considering the process of obtaining the numerical solution has been developed, we have presented the parameter optimization technique and at the same time concluded that the process is time demanding and thus considered the model reduction and pattern storage and recognition techniques.

In chapter 5 the simple test codes in Matlab have been presented to serve as building blocks for the general code governing all the techniques presented in the thesis.

In chapter 6 some graphical and numerical results along with the tables have been presented where one can see that proposed techniques in the thesis work very well in simple cases.

The main author's contribution is the application of different modern techniques not yet ever used in this particular combination for the combustion and reaction kinetics applications along with their modifications and adopting them for this particular matter.

The pattern storage results are something the author had come up within the last steps of gathering data for the thesis and the simple examples such as single and two criteria recognition has only been considered, but theoretically this concept can be used to solve the whole closest curve or surface finding task by using the proposed machinery. This would be the topics of author's further researches.

Appendix A

Author's participation in published works

Publications presented here are the ones author has been involved in for the last couple of years whilst researching the topic.

These have a lot of co-authors too since there have been a lot of colleagues in several projects the author has been involved in.

Amongst the wider list of publications and proceedings with author's contributions on similar but not exactly the same topic, we present the list of thematically connected publications on the topic of this particular thesis only and omit most of the conference proceedings.

- On the numerical simulation of the vortex breakdown in the combustion process with simple chemical reaction and axial magnetic field, International Journal of Differential Equations and Applications, Volume 14, No. 3, 2015, p. 235-250. Coauthors: H. Kalis, U.Strautiņš, O. Lietuvietis.
- Magnetic Field Control of Combustion Dynamics. Latvian Journal of Physics and Technical Sciences, Vol. 53, N4, p. 36-47, 2016. Coauthors: I. Barmina, R. Valdmanis, M. Zaķe, H. Kalis, U. Strautiņš.
- On numerical simulation of electromagnetic field effects in the combustion process. Mathematical Modelling and Analysis, 23(2), p. 327-343, 2018. Coauthors: H. Kalis, U. Strautiņš, M. Zaķe.
- Experimental Study and Mathematical Modelling of Straw Co-Firing with Peat. Chemical Engineering Transactions, Vol. 65, p. 91-96, 2018. Coauthors: I. Barmina, A. Kolmičkovs, R. Valdmanis, M. Zaķe, H. Kalis.
- Experimental study and mathematical modelling of straw co-firing with propane. Chemical Engineering Transactions, Vol.74, p. 19-24, 2019. Coauthors: I. Barmina, A. Kolmičkovs, R. Valdmanis, M. Zaķe, H. Kalis.
- Mathematical Modelling and Experimental Study of Straw co-Firing with Gas. Mathematical Modelling and Analysis, Vol. 24, N4, p. 505-529, late 2019. Coauthors: I. Barmina, H. Kalis, A. Kolmičkovs, L. Ozola, U. Strautiņš, R. Valdmanis, M. Zaķe.
- Mathematical Modelling and Experimental Study of Straw co-Firing with Gas Using Electric Field Control of Combustion Characteristics. Contents of Proceedings of 19th International Scientific Conference Enginering for Rural Development, p. 1059-1064, 20.-22.05.2020, Jelgava, Latvia. Coauthors: H. Kalis, A. Kolmičkovs, R. Valdmanis.

The present results are to be published in 2021.

Appendix B

Author's participation in international conferences

The list contains all the events during the last couple of years, whereas the author participated with the thematically connected presentations.

- On the vortex formation in the combustion process with simple chemical reaction and axial magnetic field // Mathematical Modelling and Analysis (MMA 2015) : 20th International Conference, Sigulda, Latvia, May 26-29, 2015 : abstracts Riga : University of Latvia, 2015. p. 43.
- On the numerical simulation of the combustion process with simple chemical reaction, 7th Baltic Heat Transfer Conference, August 24-26, 2015, Tallinn, Estonia.
- Mathematical modeling of MHD flow in the combustion process . Acta Societatis Mathematicae Latviensis, Abstr. of the 11-th Latvian Mathematical Conference, 15.04.2016, Daugavpils, Latvia, p. 37.
- On mathematical modelling of the combustion process of biomass. Acta Societatis Mathematicae Latviensis, Abstr. of the 11-th Latvian Mathematical Conference, 15.04.2016, Daugavpils, Latvia, p. 38.
- On numerical modelling of swirl flow in the combustion process // Mathematical Modelling and Analysis (MMA 2016) : 21st International Conference, Tartu, Estonia, June 01-04, 2016 : abstracts Tartu : University of Tartu, 2016. p. 36.
- Experimental and numerical study of the development of swirling flow and flame dynamics and combustion characteristics at biomass thermo-chemical conversion. 16th International Scientific Conference Enginering for Rural Development, 24.-26.05.2017, Jelgava, Latvia.
- Effects of gradient magnetic field on swirling flame dynamics, 16th International Scientific Conference Enginering for Rural Development 24.-26.05.2017, Jelgava, Latvia.
- Electric field effects on gasification/combustion at thermo-chemical conversion of biomass mixtures. 16th International Scientific Conference Enginering for Rural Development, 24.-26.05.2017, Jelgava, Latvia.
- On numerical simulation of electromagnetic field effects in the combustion process. Abstr. of MMA2017, May 30-June 02, 2017, Druskininkai, Lithuania, p. 41.

- Mathematical Modelling on Electromagnetic Field Control of the Combustion Process, Modelling for Materials Processing, Riga, September 21-22, 2017.
- On mathematical modelling of the chemical reactions for two-dimensional diffusion flames. Acta Societatis Mathematicae Latviensis, Abstr. of the 12-th Latvian Mathematical Conference, 13.04.2018, Ventspils, Latvia, p. 40.
- Development of combustion dynamics at co-combustion of straw with wood. 17th International Scientific Conference Enginering for Rural Development, 23.-25.05.2018, Jelgava, Latvia.
- Influence of electric field on thermo-chemical conversion of mixtures of straw pellets with coal. 17th International Scientific Conference Enginering for Rural Development, 23.-25.05.2018, Jelgava, Latvia.
- Mathematical modelling and experimental study of co-firing straw with gas. Abstr. of MMA2018, May 29-June 1, 2018, Sigulda, Latvia.
- Numerical study of electrodynamic control of straw co-firing with propane. Engineering for Rural Development: 18th International Scientific Conference, May 22-24, 2019, Jelgava, Latvia.
- Mathematical Modelling and Experimental Study of Straw co-Firing with Gas Using Electric Field Control of Combustion Characteristics. Engineering for Rural Development: 19th International Scientific Conference, 20.-22.05.2020, Jelgava, Latvia.

Bibliography

[20 04] Kinetic Mechanism for 20 torr. In: (2004). URL: http://web.mit.edu/anish/www/pub20torr.mec.

[Ang03] P. Knabner, L. Angermann. “Numerical Methods for Elliptic and Parabolic Partial Differential Equations”. In: *Springer* (2003).

[BAE05] F. van den Bergh, A.P. Engelbrecht. “A study of particle swarm optimization particle trajectories”. In: *Information Sciences* (2005).

[Ben13] M.G. Larson, F. Bengzon. “The Finite Element Method Theory, Implementation, and Applications”. In: *Springer* (2013).

[Bet16] A. Bettini. “A Course in Classical Physics 2 — Fluids and Thermodynamics”. In: *Springer* (2016).

[Bor01] E.S. Oran, J.P. Boris. “Numerical Simulation of Reactive Flow”. In: *Cambridge University Press, NY USA* (2001).

[Bro13] R.L. Brooks. “The Fundamentals of Atomic and Molecular Physics”. In: *Springer* (2013).

[Cha01] D. Mandic, J. Chambers. “Recurrent neural networks for prediction: learning algorithms, architectures, and stability”. In: *John Wiley* (2001).

[Cle06] M. Clerc. “Particle Swarm Optimization”. In: *Iste* (2006).

[Fin] In: *Encyclopedia of Math* (2016). URL: https://encyclopediaofmath.org/wiki/Finite-increments_formula.

[Fou03] R.A. Adams, J.J.F. Fournier. “Sobolev Spaces”. In: *Elsevier* (2003).

[Gre08] M.A. Singer, W.H. Green. “Using adaptive proper orthogonal decomposition to solve the reaction–diffusion equation”. In: *Applied Numerical Mathematics* (2008).

[Hac17] W. Hackbusch. “Elliptic Differential Equations Theory and Numerical Treatment”. In: *Springer* (2017).

[Mar16] I. Barmina, R. Valdmanis, M. Zaķe, H. Kalis, U. Strautiņš, M. Marinaki. “Magnetic Field Control of Combustion Dynamics”. In: *Latvian Journal of Physics and Technical Sciences, Vol. 53, N4, p. 36-47* (2016).

[Mar18a] H. Kalis, U. Strautiņš, M. Zaķe, M. Marinaki. “On numerical simulation of electromagnetic field effects in the combustion process”. In: *Mathematical Modelling and Analysis, 23(2), p. 327-343* (2018).

[Mar18b] I. Barmina, A. Kolmičkovs, R. Valdmanis, M. Zaķe, H. Kalis, M. Marinaki. “Experimental Study and Mathematical Modelling of Straw Co-Firing with Peat”. In: *Chemical Engineering Transactions, Vol. 65, p. 91-96* (2018).

[Mar19] I. Barmina, A. Kolmičkovs, R. Valdmanis, M. Zaķe, H. Kalis, M. Marinaki. “Experimental study and mathematical modelling of straw co-firing with propane”. In: *Chemical Engineering Transactions, Vol.74, p. 19-24* (2019).

[Mei08] A. Meister. “Numerik linearer Gleichungssysteme”. In: *Vieweg* (2008).

[Mil00] R.J. Kee, F.M. Rupley, J.A. Miller. “The Chemkin Thermodynamic database”. In: *Reaction design, Sandia National Laboratories Report SAND87-8215B* (2000).

[Pfe12] W.F. Pfeffer. “The Divergence Theorem and Sets of Finite Perimeter”. In: *Taylor and Francis* (2012).

[Tab20] Dynamic Periodic Table. In: (2020). URL: `https://ptable.com/`.

[Tla] In: *Mathworks* (2021). URL: `https://se.mathworks.com/help/matlab/getting-started-with-matlab.html`.

[Vey05] T. Poinsot, D. Veynante. “Theoretical and Numerical Combustion”. In: *Edwards, PA USA* (2005).

[Wil10] M. Buffoni, K. Willcox. “Projection-based model reduction for reacting flows”. In: *40th Fluid Dynamics Conference and Exhibit* (2010).

[Wil85] F.A. Williams. “Combustion Theory. The Fundamental Theory of Chemically Reacting Flow Systems”. In: *Taylor and Francis, FI USA* (1985).

[Woo02] D.M.T. Hall, D.L. Torek, P.V. Schrock, C.R. Wooldridge. “H2/O2 reaction mechanism”. In: *M.S., Proc. Combust. Inst. 29 (2002), in press* (2002).