# Chapter 1

# Prologue

In this thesis we discuss and develop *adaptive* multigrid solvers for space-time discretisations of parabolic reaction diffusion equations with a potentially nonlinear forcing term. They present a broad class of partial differential equations that can be written in the form

$$u_t - div(D(x)\nabla u) = f(u) \tag{1.1}$$

for some $u = u(x,t)$ in a space domain over a time interval in addition to a set of boundary conditions. We will postpone more rigorous definitions to the following chapters and for now simply assume the problem to be well posed. This type of equations is used to describe a variety of physical phenomena. In its simplest form we have a zero source term, that is $f = 0$. This heat equation describes the variation of temperature in a particular region over time starting from a set of initial conditions which will eventually reach an equilibrium state. Other important applications are the transformation of one or more chemical substances into another over time, the development of animal populations in biology [1] or the propagation of wavefronts [2]. A particular instance of a traveling wave which we will particularly focus on and which originally motivated the topic of this thesis, is the propagation of electric signals in human heart tissue. It can be modeled using the so-called monodomain equations which are also a reaction-diffusion system [3]. The contraction of our heart is governed by an electric impulse whose charge distribution travels as a wavefront through our cell tissue. When trying to numerically approximate such a process one faces a number of challenges. One of the main difficulties that arises is the multiscale range in space and time [4]. The overall space and time domain are very large compared to the rapid local changes of the current potential in the wavefront which therefore require a high accuracy in time and several space dimensions. Therefore *representative* discretisations result in large systems of equations involving *extensive* numbers of degrees of freedom. Solving them in an accurate, robust and efficient way has been and continuous being an extensive area of interest and research [source].

In general when trying to numerically approximate the solution of a partial differential equation there is no unique way to do so and hence many design choices have to be made. A very important one includes the way of how to discretise the domain [source]. A frequently used possibility is the method of lines approach, where first the space is discretised e.g. using finite elements and where the time variable remains continuous which will give rise to a system of ordinary differential equations that is then to be solved by an appropriate method [source]. A very common approach is to use a time stepping method [source]. That is one computes an approximation for all space elements or nodes at a certain time $t_n$ and then uses those results or even preceding ones to compute the approximate solution at the next time $t_{n+1}$. This is the natural way to perform operations, because this is also how we move through time, sequentially, causality implies that the solution at a given time depends on the previous one but not the other way around. However in current technological development where there is no further significant increase in CPU clockspeed, the only way to really achieve a gain in computational power is through an increase in the number of processors. Therefore for this to actually translate to a computational speed up one requires algorithms to be more and more parallelisable, that is to allow for more operations to be performed at the same time. The approach outlined above contains inherently sequential processes, since only the space dimensions allow for parallelisation. As this saturates [source] there is no possibility for a further speed up. Thus it makes sense to look for methods that utilise a parallelisation in space and time simultaneously. This in turn naturally leads to a space-time discretisation of the equation as a whole [5], which is also what we will be considering in this thesis, a large space-time system which we want to be able to mainly solve in parallel. A short discussion of the research done on this field so far, advantages and difficulties as well as some further references can be found in section 3.1., while the particular discretisation we chose will be introduced in chapter 4.

It has been shown that large linear systems of equations are often most efficiently solved using iterative schemes [source]. Among them, multigrid methods represent an important and powerful class to approximate such solutions. In the case of sparse, symmetric, positive definite systems they even provide optimality in the sense that their complexity can be bounded by $O(N)$, where $N$ is the number of degrees of freedom [6]. Unfortunately the behavior of multigrid algorithms in an indefinite or not symmetric setting is often not yet very well understood or not suitable [source], and convergence is generally not guaranteed [source]. Therefore we would like to aim for the construction of a system that can claim as many of these preferrable properties as possible. However most space-time solution methods do not give rise to symmetric positive definite systems [5] which is why we recast problem (1.1) as an optimisation probelm, an ansatz known as least squares finite element methods [7] and which will be first introduced in section 3.4. It entails the construction of a minimisation problem whose solution coincides with the solution of the differential equation. Instead of solving the original problem we now apply a finite element approach in space-time to solve the auxiliary problem whose value for a given input $u$ denotes an energy that we can minimise. In the linear case we are, due to the symmetry and positive definiteness of the system, guaranteed the existence of a global minimiser. In the nonlinear case we consider linearisations of the system which are generally not positive definite, but the symmtetry is maintained because of the commutativity of derivatives. The problem is non-convex but by successively reducing energy we can still find local minima. Hence for a nonlinear source or reaction term $f$, we additionally require an outer nonlinear iteration scheme which succuessively solves linearisations of the least squares functional. The non-linear solvers that were employed in the implementation section here are a damped Newton method [8] and a trust region method [9], and will be introduced in section 3.2. We have convergence to a global minimiser in a neighbourhood of the solution, that is for a sufficiently good initial starting iterate the solution of the original problem is recovered.

Below we can see a schematic overview of how these beforementioned core concepts are tied together in order to give rise to a comprehensive *solver*.

**Overview of the different Steps towards an Approximate Solution**

1. Reformulate (1.1) as a mimimisation problem $J$ whose solution coincides with the one of the original equation.
2. Discretise the problem using a space-time finite element approach
3. Derive a non-linear iteration scheme (e.g. Newton or Trust Region method) where we solve a linearisation of the problem using the current iterative solution in each step
4. Solve the arising linear system of equations using a multigrid method
5. Repeat step 4 with the updated solution each time until a stopping criterion is met

These are the main ingredients that we will tie together in this thesis in order to develop an efficient, robust and accurate solver to tackle problems of type (1.1). It is a rather novel construction that has, to our knowledge, not been studied in this context and will therefore require further investigations before drawing any final conclusions on its utility. The mathematical methodologies will be introduced more thoroughly in chapter 3, where we will also explain the particular choice for each of them in more detail, attempting to make use of their favourable properties while trying to avoid the pitfalls. In chapter 4 we derive a proper problem formulation, which we will then discretise in order to derive linear systems of equations to be solved iteratively. Afterwards we introduce multrigrid methods in chapter 5, especially discussing the particularities that arise due to the construction presented in chapter 4. Chapter 6 then contains the numerical results we obtained for various test cases and discusses certain behaviors we observed during our work which will then be followed by a conclusion and an outlook in chapter 7.

In order to really obtain a meaningful solution $u$ we need a number of properties to be fulfilled. In each nonlinear iteration step the multigrid solver has to converge to the solution of the linearised least squares minimisation problem. In the outer iteration we need the nonlinear iteration scheme to converge to the minimiser of our non-linear functional whose solution as mentioned above is supposed to correspond to the solution of the original problem. However we are not ensured global convergence since the problem is in general non-convex.

Overall we are aiming for a better understanding of the versatility of space-time least squares finite element approaches in general and in combination with multigrid methods. *A focus will be given to the construction of a particular algebraic multigrid method that takes intrinsic properties related to the monodomain equation into account, developing an equally accurate but more efficient way through an adapted coarse grid construction.* To allow for a better understanding of the processes involved in this particular application the following chapter will give a brief insight into the functioning of the human heart, the transmission of electric potential through tissue, the different charge distribution within or between cells or cellular structures and how this can be turned into a mathematical model.

# Chapter 2

# Cardiac Electrophysiology

Our hearts are absolutely vital for our survival. While it normally functions with an incredible reliability and accuracy that does not even let us begin comprehend the complexity of the mechanisms involved, cardiovascular diseases are estimated to make up for more than 30% of all world wide's death [10]. Often this is related to abnormal heart contractions and thus understanding the processes involved in governing our heart beats is crucial to explaining heart failues. The heart acts as a double pump made out of muscle tissue that provides our bodies with freshly oxygenated blood [11]. A heart has about the size of a fist and sits between our two lungs. It consists of 4 chambers, the upper two atria, that is the left and right atrium and the lower two ventricles which have connections through four heart valves that can open and close respectively but only allow the blood to flow in one direction. The left and the right side of the heart is separated by a wall of tissue known as the atrioventicular septum. An intricate interplay of contraction and relaxation of the chambers governed through electrical stimuli lead to a stable blood flow that enables the replenishment of oxygen levels of the cells in our body. Below we can see a schematic image of the heart, where the arrows indicate the direction of blood flow.

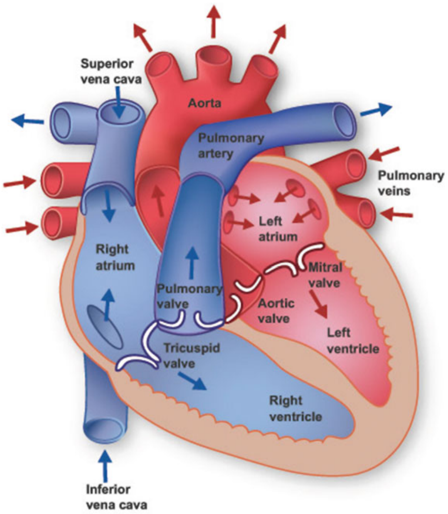



Figure 2.1: Scheme of the Heart [3]

The blood flow through the different chambers of the heart occurs in repeating cycles. After circulating through the body low oxygenated blood flows back into the heart through our veins and enters into the right atrium, which contracts once it is full. This contraction causes a pressure built up and pushes the tricuspid valve open. The blood rushes into the right ventricle, whose walls, once filled, also begin to contract, the pressure within rises again, which shuts tricuspid valve and opens the pulmonary valve to the pulmonary artery from where the blood reaches the lungs and replenishes it oxygen stocks. Afterwards it returns to the left side of the heart from the pulmonary veins to the left atrium, which again, once it is completely filled, contracts and hereby opens the miral valve and forces the blood into the left ventricle. The left ventricle then pumps the oxygenated blood through the aortic valve into the aorta from where it flows into different parts in the body to supply cells with oxygen and nutrients before returning to the right atrium and repeating its cycle. The mitral and tricuspid valves open, and the aortic and pulmonic vavles close while the ventricles fill with blood. In contrast the mitral and tricuspid valves shut, and the aortic and pulmonic valves open during ventricular contraction. This particular sequence makes sure that all ventricles are filled up to capacity before pumping and that blood flows only in one direction. For references and further information see [12], while the following sections are based on [3].

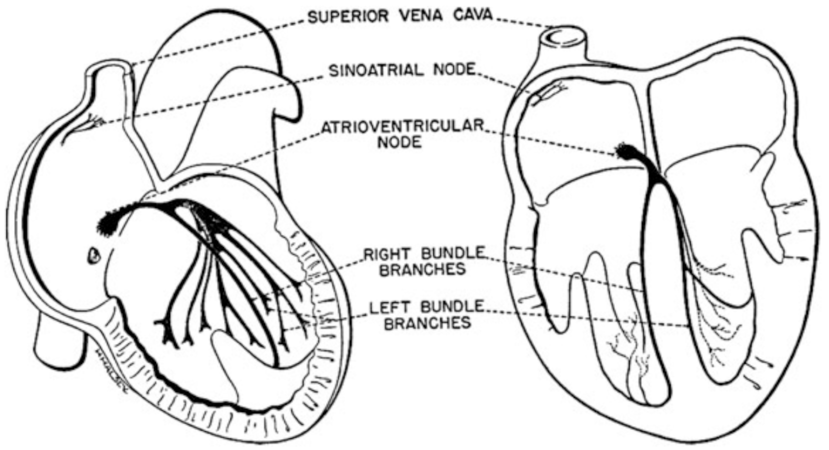



Figure 2.2: Overview of the Hearts Conduction System [3]

The heart contractions are initiated by an electric activiation, that is a depolarizing transistory membrane current which raises the transmembrane potential from its resting value of about -90 to -80 mV to small positive values. This potential describes the difference in the electric potential between the interior and exterior of the cell. The increase is followed by a repolarization current which sends the transmembrane potential back to its resting value. The initial electrical stimulus is generated by the sinoatrial node which is located on the right atrium close to the superior vena cava and possesses the ability to excite its cells autonomously. The frequency of its stimuli is dependent on the parasympathetic nervous system and hormonal factors but under normal health and stress conditions ranges from about 60-100 times per minute. The signal is then transmitted through the surrounding cells and cardiac conduction pathways to the various chambers of the heart. It first propagates to the right atrium and through Bachmann's bundle to the left atrium where it stimulates the cardiac muscle cells of the atria to contract. The activation front then travels to the atrioventricular node situated at the base of the atria. The cells there have a relatively slow conduction velocity and therefore cause a delay in the transmission which is timed this way to achieve optimal pump activity. From the atrioventricular node the stimulus reaches specialised fibres in the bundle of His and the Purkinje network that branch in the left and right bundle onto the inner surface of the ventricals. Again causing a contraction of the cardiac muscle tissue.

## 2.1 Electrical Activity on the Cellular Level

The heart's walls can be subdivided into 3 different layers; the inner endocardium which surrounds the the heart chambers; the outer endocardium which protects and delimits the heart from other parts of the body and the predominant middle layer consisting of cardiac muscle tissue called myocardium. This is where the conduction of the electric potential and the heart contractions mainly take place. Myocardium is made up of sheets of cells, where each one is roughly of a cylindrical shape with a size ranging from 100-150 $\mu$m by 30-40 $\mu$m. (diff source 50-150, 10-20) They are organised in a way similar to a brick wall and joined together at the ends by intercalated disks turning them into long fibres. The disks allow for easy ion movement between the cells and thus allowing for a rapid transmission of electrical impulses. Each cardiomyocyte that is each cardiac muscle cell contains bundles of myofibrils which are protein fibres which can slide past each other, making it possible for the tissue to contract. The cell's membrane is called sarcolemma and contains certain transmembrane channels whose opening and closing is governed through electric stimuli (mainly transversal cell direction?). The intercalated disks allow the transit of ions through channels called gap junctions which are predominantly in longitudinal fiber direction. Due to the varying density of the gap junctions in the different directions there is an anisotropic propagation of the electric potential throughout the tissue which complicates its simulation.

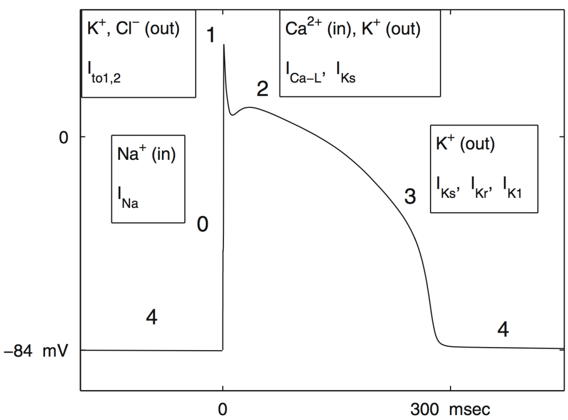



Figure 2.3: Different Phases of Cardiac Action Potential [3]

In this figure we can see a standard ventricular action potential in its main phases, that is the electric charge distribution a cell goes through over time. Following [3] we will have a brief look at the different stages that occur and what they entail.

Phase 0: Depolarization of the cell by opening of $Na^+$ channels of the sarcolemma which leads to rapid inflow of $Na^+$ ions into the cell. Hence, the transmembrance potential passes from negative to positive values.
Phase 1: Outward flow of $K^+$ and $Cl^-$ ions after the inactivaiton of $Na^+$ channels, which causes a rapid decrease of the potential
Phase 2: Governed by an inward as well as an outward current of $Ca^{2+}$ and $K^+$ ions respectively such that there almost is a balance in the potential
Phase 3: Repolarization of the cell by closing of the $Ca^{2+}$ channels while outward current of $K^+$ ions is maintained therefore returning the potential to negative values.
Phase 4: The potential remains at a constant negative value. Some channels are kept open to allow for keeping the right inter-and extracellular charge balance. The cardiomyocyte stays in

this resting state until the next stimulation.

The stimuli are under normal conditions about 0.6-1 seconds apart, which means that the cell is about half the time in its resting phase. They travel as a wave through the cardiac tissue, where one cell excites the next. After this brief description of the processes involved in the functioning of the human heart, let us in the subsequent section turn towards the question of how to adequately represent them in a model and what the particular difficulties are that arise.

## 2.2 The Monodomain Equation

The propagation of these stimuli is usually modeled with either a version of the bidomain equation or monodomain equation. The former was first developed in the late 1970's and is the more comprehensive one of the two still posing many computational challenges in its implementation and execution. Therefore one often relies on a simpler monodomain model which in the vast majority of applications leads to a very similar solution and can therefore be considered as an adequate approximation, [13]. The discrete cellular structure is replaced by an averaged continuous model, giving rise to a parabolic reaction-diffusion equation derived from the cable equation maintaining a conservation of charge. Cable theory is used to calculate the electric current in nerve fibres by modeling them as composed segments with capacitances and resistences combined in parallel. The diffusion term represents the spread of current through gap junctions and cardiac tissue while the reaction terms describe the flux of ions across the myocyte membrane. *say more about internal and external current.*

$$\begin{aligned} \partial_t u - \nabla \cdot (D(x)\nabla u) &= I_{\text{int}}(u) + I_{\text{ext}}(x,t) \quad && (x,t) \in \Omega \\ \nabla u \cdot n &= g(x,t) && (x,t) \in \Gamma_N \end{aligned} \tag{2.1}$$

where the above terms describe the following

$u(x,t)$ : electric potential
$D(x)$ : conductivity tensor
$I_{\text{int}}(u)$: internal current
$I_{\text{ext}}(x,t)$: external current

In the case of the simplified FitzHugh-Nagumo model we have $I_{\text{int}} = u(u-1)(\alpha - u)$, with $0 < \alpha < 1$. *mention gating variables...?*

There are a number of difficulties that arise when trying to numerically approximate a solution to this problem. One is the beforementioned challenging task of dealing with large differences in spatial and temporal scaling due to the complex multiscale structure. As mentioned before the microscopic scale the electric conduction of excitation fronts happens through ion channels of cellular membranes which are on a scale of the order of 0.1 mm. On the other hand the overall size cardiac tissues invovled entails a size of several centimeters which leads to a spatial spread factor of up to $10^3$. Similarly for the time parametrisation we have an even larger spread factor. A normal heartbeat takes about 1 second, that is one full cycle, whereas the step excitation front that is described in phase 0 in the previous section ranges on a much shorter time scale therefore requiring time steps within a range of about 0.1 to 500 milliseconds for accurate representation [4].

But the sheer size of the problem is not the only difficulty, as mentioned in the prologue there are many ways to discretise the domain, various methodologies of how to represent the differential operators and how to approximate the arising linear and nonlinear systems of equations. As we have tried to reason for the particular choices we have made, the following chapter is meant to serve as an introduction to these mathematical methodologies we apply, outline their underlying principles, demonstrate their functioning using standard examples and point out their relevance and applicability for other *problems*.

# Chapter 3

# Mathematical Ingredients

We will begin this chapter by briefly introducing finite element methods and before particularly focusing on least squares finite element methods. The subsequent sections will then give a general overview of space-time solution methods, before going into more detail about the methodology applied in this thesis. Before finally discussing nonlinear iteration schemes. While these concepts or methods are introduced separately here, we will use chapter 4 to tie them together in a *comprehensive* solver.

## 3.1 Finite Element Methods

In order to find a numerical estimate to the solution of a partial differential equation we need a way to approximate the operators involved. And while there are many different ideas of how to do so the one we have chosen to employ is a finite element approach, as they have shown to be one of the most powerful and versatile methodologies for the problem at hand [7]. The purpose of the subsequent section is not to establish the whole finite element framework from scratch but rather to provide the introduction of a unified notation that will be referred to throughout this thesis and a recollection of the most important properties needed. Anything else would be far beyond the scope this thesis, as a full description of the underlying mathematical constructions can quickly become rather involved but we would like to refer to [14] or *[good finite element specific source?!]* for a comprehensive *discussion* of the topic.

### 3.1.1 General Setting

The foundation of every finite element formulation is finding an appropriate weak formulation which includes the choice of suitable trial and solution spaces. This is especially applicable in the case of a least squares approach and will be discussed in further detail in section [...].
Given Banach spaces $X$ and $Y$, a bounded linear operator $\mathcal{A} : X \to Y$, $f \in Y$, we consider the problem:

$$\text{Find } u \in X \text{ such that} \quad \mathcal{A}u = f \text{ in } Y. \tag{3.1}$$

We are interested in the case where $\mathcal{A}$ represents a partial differential operator. As mentioned before the process of discretisation begins with turning (3.6) into a suitable variational equation which is defined in terms of two Hilbert spaces $V$ and $W$, a continuous bilinear form $a(\cdot,\cdot) : V \times W \to \mathbb{R}$, and a bounded linear functional $L_f(\cdot) : W \to \mathbb{R}$ and is given by

$$\text{Find } u \text{ in } V \text{ such that: } a(u, v) = L_f(v) \quad \forall v \in V \tag{3.2}$$

An operator equation such as (3.6) may be reformulated into several different variational equations. We can see that we were originally seeking for a solution $u$ in the space $X$ whereas in the weak formulation one attempts to find a solution in the space $V$, and which generally doesnt lie in $X$, and is therefore often referred to as a weak solution. Hence the relationship between the spaces $X, Y$ and $V, W$, and the operator $\mathcal{A}$ and the bilinear form $a(\cdot,\cdot)$ are of great importance, and while one generally wants the solution of the variational formulation (3.7) to be a "good"

representation of the solution of the original problem (3.6), the definition of what that exactly means varies and usually depends on the nature of the problem and often some practicality issues. One possibility could be ... or too much? Therefore we have denoted them by the same letter but to be precise the solution $u$ appearing in the subsequent paragraphs will always be referring to $u \in V$, because our aim now is to solve the variational formulation.
So let us assume for now that we have found a suitable weak formulation of the operator equation where trial and test space are equal, that is $V = W$. In addition to $a(\cdot,\cdot)$ being linear and bounded, which is equivalent to the continuity, we will also require it to be symmetric, hence we have more specifically that

$a(v_1, v_2) = a(v_2, v_1)$ for all $v_1, v_2 \in V$ (symmetry)
$a(v_1, v_2) \leq \beta ||v_1||_V \cdot ||v_2||_V$, for all $v_1, v_2 \in V$ and $\beta > 0$ (boundedness)
$a(v_1, v_1) \geq \alpha ||v_1||_V^2$, for all $v_1 \in V$ and $\alpha > 0$ (coercivity)

and $f \in V^*$, the dual space of $V$. Furthermore let us have homogeneous Dirichlet boundary conditions, that is $u = v = 0$ on $\partial\Omega$. Then by *Riesz representation theorem/Lax-Milgram* we obtain that there exists a unique solution $u \in V$ that solves (2.2). And additionally the existence of an operator $\tilde{\mathcal{A}} : V \to V^*$ given by

$$a(u, v) = \langle \tilde{\mathcal{A}} u, v \rangle_{V^*,V} \quad \forall u, v \in V \tag{3.3}$$

where $\langle \cdot,\cdot \rangle$ denotes the duality pairing (more...) between $V$ and its dual space $V*$. Likewise we obtain for $L_f(\cdot)$ the existence of an unique (!) element $\tilde{f}$ through the relation

$$L_f(v) = \langle \tilde{f}, v \rangle_{V^*,V} \quad \forall v \in V \tag{3.4}$$

The variational formulation is therefore equivalent to the problem

$$\text{Find } u \in V \text{ such that} \quad \tilde{\mathcal{A}} u = \tilde{f} \quad \text{in } W^* \tag{3.5}$$

In the special case that $X = U$ and $Y = W^*$ we have that $\mathcal{A} = \tilde{\mathcal{A}}$ and $f = \tilde{f}$ but this is generally not the case.

### 3.1.2 Discretisation

A key element to actually finding a good approximation $u^h$ of $u$ is to choose a suitable finite dimensional (sub)space $V_h$ where we search for the solution. We will consider a *Galerkin approach*, where we indeed have $V_h \subset V$, which itself is again a Hilbert space and therefore the projected finite dimensional problem called Galerkin equation looks as follows

$$\text{Find } u_h \text{ in } V_h \text{ such that: } a(u_h, v_h) = L_f(v_h) \quad \forall v_h \in V_h \tag{3.6}$$

and has a unique solution itself. Since (2.2) holds for all $v \in V$ it also holds for all $v \in V_h$, and hence $a(u - u_h, v_h) = 0$, a key property known as Galerkin orthogonality. With respect to the energy norm induced by $a(\cdot,\cdot)$, $u_h$ is a best approximation to $u$, in the sense that

$$\begin{aligned} ||u - u_h||_a^2 &= a(u - u_h, u - u_h) = a(u - u_h, u) + a(u - u_h, v_h) \\ &\leq ||u - u_h||_a \cdot ||u - v_h||_a \quad \forall v_h \in V_h. \end{aligned} \tag{3.7}$$

We derive the third term from the second by using the Galerkin orthogonality. If we now divide both sides by $||u - u_h||_a$, we obtain that $||u - u_h||_a \leq ||u - v_h||_a$ for all $v_h \in V_h$. We also have an estimate on $u - u_h$ in terms of the norm $||\cdot||_V$. Using the coercivity constant $\alpha$ and the bound from above $\beta$, we see that

$$\begin{aligned}\alpha||u-u_h||_V^2 &\leq a(u-u_h,u-u_h) = a(u-u_h,u-u_h) = a(u-u_h,u+v_h-v_h-u_h)\\ &= a(u-u_h,u-v_h)+a(u-u_h,v_h-u_h) = a(u-u_h,u-v_h)\\ &\leq \beta||u-u_h||_V\cdot||u-v_h||_V \quad \forall v_h\in V_h.\end{aligned} \tag{3.8}$$

Dividing by $\alpha||u-u_h||$ we have shown *Céa's lemma*, which states that (accuracy ... constant thing):

$$||u-u_h||_V \leq \inf_{v_h\in V_h}\frac{\beta}{\alpha}||u-v_h||_V, \quad u\in V, u_h\in V_h \tag{3.9}$$

where $u$ is the solution to (2.2) and $u_h$ to the corresponding finite dimensional problem (2.3). Hence accuracy of our approximation depends in this case on the constants $\alpha$ and $\beta$.

If we assume that we have a discretisation $\Omega_h$ of our domain $\Omega$, where $h > 0$ is a parameter depending on the mesh size. We furthemore want to assume that as $h$ tends to zero this implies that $dim(V_h) \Rightarrow \infty$. Additionally let $\{V_h : h > 0\}$ denote a family of finite dimensional subspaces of V, for which we assume that

$$\forall v\in V: \quad \inf_{v_h\in V_h}||v-v_h||_V \to 0 \text{ as } h\to 0. \tag{3.10}$$

That is with a mesh size tending to zero there exist increasingly precise approximations for every $v \in V$, whose infimum tends to zero as the mesh size does. But then we can also conclude by the beforementioned properties (3.24) and (3.25) that $||u - u_h||_V \to 0$ as $h \to 0$. Hence our approximate solution $u_h$ will converge to the weak solution $u$.

### 3.1.3 Matrix Formulation

After establishing these theoretical properties our aim is now to construct a linear system of equations that can be solved efficiently. Since $V_h$ is a finite dimensional Hilbertspace, it has a countable basis $\{\phi_1, \phi_2, ..., \phi_n\}$ and we can write every element in $V_h$ as a linear combination of such, that is we also have $u_h = \sum_{j=1}^n u_j\phi_j$, where $u_1, ..., u_n$ are constant coefficients. Writing (3.21) in terms of the basis we obtain by linearity

$$a(\sum_{j=1}^n u_j\phi_j,\phi_i) = \sum_{j=1}^n u_j a(\phi_j,\phi_i) = L_f(\phi_i) \quad \forall\phi_i,\ i=1,2,...,n \tag{3.11}$$

If we now write this as a system of the form $A_h u_h = L_h$ with entries entries $(A_h)_{ij} = a(\phi_j, \phi_i)$, $(L_h)_i = L_f(\phi_i)$, then this becomes a linear system of equations which we can solve for an unknown vector $u_h$, where each matrix entry represents the evaluation of an integral expression. The question of how to choose favorable subspaces $V_h$, and a suitable basis for it has no trivial answer and depends on many factors and goes hand in hand with the question of how to best discretise the domain. Generally it seems like a sensible aim to opt for easily computable integrals giving rise to a linear system that is in turn as easy as possible to solve. Hence one objective might be to choose the basis $\{\phi_1, ..., \phi_n\}$ such that $supp(\phi_i) \cap supp(\phi_j) = \emptyset$ for as many pairs $(i, j)$ as possible. Since this would ideally give rise to a sparse system of equations. It is also worth noting that due to the symmetry of $a(\cdot, \cdot)$, we have that $a_{ij} = a_{ji}$.

Depending on the operator $\mathcal{A}$, there is not necessarily a straight forward way to translate a strong formulation, that is a problem of the type (...), into a symmetric variational formulation, that is a symmetric bilinear form $a(\cdot, \cdot)$, which can subsequently be restricted to finite-dimensional subspaces and where we search for approximate solutions. However one possibility is through the differentiation of certain energy functionals, because we know by the theorem of Schwarz that order of differentiation with respect to partial derivatives is interchangeable and therefore leads to symmetry. How to construct these functional to be related to particular differential equations will be discussed in the following section.

## 3.2 Least Squares Finite Element Methods

In this section which is based on ([7], mainly ch. 2.1) we would like to introduce least squares finite element methods (LSFEMs), a class of methods for finding the numerical solution of partial differential equations that is based on the minimisation of functionals which are constructed from residual equations. Historically finite element methods were first developed and analysed for problems like linear elasticity whose solutions describe minimisers for convex, quadratic functionals over infinite dimensional Hilbert spaces and therefore emerged in an optimisation setting. A Rayleigh-Ritz approximation of solutions of such problems is then found by minimising the functional over finite dimensional subspaces. For these classical problems the Rayleigh-Ritz setting gives rise to formulations that have a variety of favourable features and therefore have been and continue being highly successful. Among those are that:

1. *general* domains and boundary conditions can be treated relatively easily in a systematic way
2. conforming finite element spaces are sufficient to guarantee stability and optimal accuracy of the approximate solutions
3. all variables can be approximated using the same finite element space, e.g. the space of degree $n$ piecewise polynomials on a particular grid
4. the arising linear systems are
    (i) sparse
    (ii) symmetric
    (iii) positive definite

Hence finite element methods originally emerged in the environment of an optimisation setting but have since then been extended to much broader classes of problems that are not necessarily associated to a minimisation problem anymore and generally lose the desirable features of the Rayleigh-Ritz setting except for 1 and 4 (iii). Least squares finite element methods can be seen as a new attempt to re-establishing as many advantageous aspects of the Rayleigh-Ritz setting as possible, if not all, for more general classes of problems. In the following section we will have a look at a classical straightforward Rayleigh-Ritz setting to familiarise ourselves with the set up before extending it to the more complicated class of problems introduced in [...].

We will consider a similar set up as in the finite element section (3.3.1) but with $X$ and $Y$ being Hilbert spaces, $f \in Y$ and a bounded, coercive linear operator $\mathcal{A} : X \to Y$, that is for some $\alpha, \beta > 0$:

$$\alpha||u||_X^2 \leq ||\mathcal{A}u||_Y^2 \leq \beta||u||_X^2 \quad \forall u \in Y. \tag{3.12}$$

We consider the problem and the least squares functional:

$$\text{Find } u \in X \text{ such that } \mathcal{A}u = f \text{ in } Y \tag{3.13}$$

$$J(u;f) = ||\mathcal{A}u - f||_Y^2 \tag{3.14}$$

which poses the minimisation problem:

$$\text{argmin}_{u\in X} J(u;f) \tag{3.15}$$

where we can see that the least squares functional (3.21) measures the residual of (3.20) in the norm of $Y$ while seeking in for a solution in the space $X$. It follows that if a solution of the the problem (3.20) exists it will also be a solution of the minimisation problem. And a solution of the minimisation problem due to the definition of a norm will be a solution to (3.20) if the minimum is zero. If we consider $f = 0$, and using (3.19) we obtain that

$$\alpha^2||u||_X^2 \leq J(u;0)||_Y^2 \leq \beta^2||v||_X^2 \quad \forall u \in X \tag{3.16}$$

a property of $J(\cdot,\cdot)$ which we will call norm equivalence, which is an important property when defining least squares functionals. We can derive a candidate for a variational formulation of the following form

$$a(u,v) = (\mathcal{A}u, \mathcal{A}v)_Y \text{ and } L_f(v) = (\mathcal{A}v, f)_Y \quad \forall u, v \in X \tag{3.17}$$

where $(\cdot,\cdot)_Y$ again denotes the innerproduct on $Y$, which will turn out to have all the desired properties. The operator form of (3.21) in the least squares setting is equivalent to the normal equations

$$\mathcal{A}^*\mathcal{A}u = \mathcal{A}^*f \quad \text{in } X \tag{3.18}$$

and corresponds to equation (3.9), with $\tilde{A} = \mathcal{A}^*\mathcal{A}$, $\tilde{f} = \mathcal{A}^*f$ and $\mathcal{A}^*$ being the adjoint operator of $\mathcal{A}$. We can then move on to limiting our problem to a finite dimensional setting, where we choose a family of finite element subspaces $X^h \subset X$, parametrised by $h$ tending to zero and restricting the minimisation problem to the subspaces. The LSFEM approximation $u^h \in X^h$ to the solution $x \in X$ of the infinite dimensional problem is the solution of the discrete minimisation problem

$$\min_{u^h\in X^h} J(u^h;f) \tag{3.19}$$

which is due to the fact that $X^h$ is again a Hilbert space and therefore the same properties hold. Similarly to section (3.3.3) we can choose a basis $\{\phi_1, ..., \phi_n\}$ of $X^h$ and will then obtain for the elements of $A^h\mathbb{R}^{n\times n}$, and $L_f^h \in \mathbb{R}^n$ that

$$A_{ij}^h = (\mathcal{A}\phi_j, \mathcal{A}\phi_i)_Y \quad \text{and} \quad (L_f^h)_i = (\mathcal{A}\phi_i, f)_Y \tag{3.20}$$

The following theorem establishes that this problem formulation actually gives rise to finite element set up.

**Theorem 1.** *Let $\alpha||u||_X^2 \leq ||\mathcal{A}u||_Y^2 \leq \beta||u||_X^2$ for all $u \in X$ hold, under the same assumptions as established in this section and let $X^h \subset X$. Then,*

*(i) the bilinear form $a(\cdot,\cdot)$ defined in (3.21) is continuous, symmetric and coercive*

*(ii) the linear functional $L_f(\cdot)$ defined in (3.21) is continuous*

*(iii) the variational formulation (3.21) is of the form (3.9) and has a unique solution $u \in X$ which is also the unique solution of the minimisation problem (3.19)*

*(iv) there exists a constant $c > 0$, such that $u$ and $u_h$ satisfy*

$$||u - u^h||_X \leq c \inf_{v^h \in X^h} ||u - v^h||_X \tag{3.21}$$

*(v) the matrix $A^h$ is symmetric positive definite*

**Idea of Proof:** The properties *(i)* and *(ii)* directly follow from the boundedness and coercivity of $\mathcal{A}$ as well as the linearity of the inner product. Property *(iii)* follows from the theorem of Lax-Milgram while property *(iv)* is a consequence of Céa's lemma. The last property directly follows from the definition of $A^h$.

We therefore obtain that this least squares problem formulation has all the advantageous featurs of the Raleigh-Ritz setting without requiring $\mathcal{A}$ to be self-adjoint or symmetric which was our initial goal. However it is worth noting that the differential operator $\tilde{\mathcal{A}} = \mathcal{A}^*\mathcal{A}$ is of higher order than the one in the original formulation, which therefore requires higher regularity assumptions which might be unpreferrable as well as impractical. Potential ways to overcome this problem will be discussed in the following section as it is also an issue that arises in the problem formulation of the subsequent chapter.

## 3.3 Space-Time Solution Methods

Most solution methods for partial differential equations do not use the time direction for parallelisation. But with increasingly complex models, especially when many small steps in time are required and the rise of massively parallel computers, the idea of a parallelisation of the time axis has experienced a growing interest. Once parallelisation in space saturates it only seems natural to consider this remaining axis for parallelisation, after all, time is just another dimension [5]. However evolution over time behaves differently from the spatial dimensions, in the sense that it follows the causality principle. It means that the solution at later times is determined through earlier times whereas the opposite does not hold. This is not the case in the spatial domain.

The earliest papers on time parallelisation go back more than 50 years now to the 1960's, where it was mostly a theoretical consideration, before receiving an increasingly growing interest in the past two decades due to its computational need and feasibility. As mentioned in [5], on which this section is mainly based on and can be referred to for further details, time parallel methods can be classified into 4 different approaches, methods based on multiple shooting, domain decomposition and waveform relaxation, space-time multigrid and direct time parallel methods. Below a very brief overview of the main ideas behind these methods through some examples before taking a closer look at the strategy employed in this thesis.

**Shooting type time parallel methods** use a decomposition of the space-time domain $\Omega$ into time slabs $\Omega_j$, i.e. $\Omega = \mathcal{S} \times [0, T]$ where $\mathcal{S}$ describes the spatial domain then $\Omega_j = \mathcal{S} \times [t_{j-1}, t_j]$ with $0 = t_0 < t_1 < .... < t_m = T$. Then there is usually an outer procedure that gives a coarse approximated solution $y_j$ for all $x \in \mathcal{S}$ at $t_j$ for all $j$, which are then used to compute solutions in the time subdomains $\Omega_j$ independently and in parallel and give rise to an overall solution. One important example of how this can be done was given by Lions, Maday and Turinici in 2001 [**?**], with an algorithm called parareal. A generalized version of it for a nonlinear problem of the form

$$y' = f(y), \quad y(t_0) = y_0 \tag{3.22}$$

can be formulated as follows using two propagation operators:

1. $G(t_j, t_{j-1}, y_{j-1})$ is a coarse approximation of $y(t_j)$ with initial condition $y(t_{j-1}) = y_{j-1}$

2. $F(t_j, t_{j-1}, y_{j-1})$ is a more accurate approximation of $y(t_j)$ with the initial condition $y(t_{j-1}) = y_{j-1}$.

Starting with a coarse approximation $Y_j^0$ for all points in time $t_j$ using $G$, the algorithm computes a correction iteration

$$Y_j^k = F(t_j, t_{j-1}, Y_{j-1}^{k-1}) + G(t_j, t_{j-1}, Y_{j-1}^k) - G(t_j, t_{j-1}, Y_{j-1}^{k-1}) \tag{3.23}$$

which converges for initial value problems of the beforementioned type (3.1) under a few assumptions and for which we can find the proof in [15].

In **space-time domain decomposition methods** the idea is to divide the domain $\Omega$ into space slabs, that is $\Omega_i = \mathcal{S}_i \times [0, T]$ where $\mathcal{S} = \cup_{i=1}^n \mathcal{S}_i$. Then again an iteration or some other method is used to compute a solution on the local subdomains which can be done in parallel. A major challenge here is how to adequately deal with the values arising on the interfaces of the domain. For examples we can refer to [16] or [17].

**Direct Solvers in Space-Time** employ varying techniques. One example is a method introduced in 2012 by S. Güttel called ParaExp [18], it is only applicable to linear initial value problems and most suitable for hyperbolic equations, where other time parallel solvers often have difficulties. To understand the underlying idea let us consider the following problem:

$$y'(t) = Ay(t) + g(t), \quad t \in [0, T], \quad u(0) = u_0 \tag{3.24}$$

One then considers an overlapping decomposition of the time interval $0 < T_1 < T_2 < .... < T_m = T$ into subintervals $[0, T_m], [T_1, T_m], [T_2, T_m], ..., [T_{m-1}, T_m]$. Now there are two steps to be performed. First one solves a homogenous problem for the initial parts of each subdomain, that is $[0, T_1], [T_1, T_2], ..., [T_{m-1}, T_m]$, which is non-overlapping and can therefore be done in parallel:

$$v_j'(t) = Av_j(t) + g(t), \quad v_j(T_{j-1}) = 0 \quad t \in [T_{j-1}, T_j] \tag{3.25}$$

and afterwards the overlapping homogeneous problem is solved:

$$w_j'(t) = Aw_j(t), \quad w_j(T_{j-1}) = v_{j-1}(T_{j-1}), \quad t \in [T_{j-1}, T_m] \tag{3.26}$$

Due to linearity the overall solution can be obtained through summation

$$y(t) = v_k(t) + \sum_{j=1}^{k} w_j(t) \quad \text{with } k \text{ s.t. } t \in [T_{k-1}, T_k] \tag{3.27}$$

This way we obtain the general solution over the whole time interval. One might wonder why this approach gives a speed up since there is great redundency in the overlapping domains of the homogeneous problems which also need to be computed over big time intervals. The reason behind this is that the homogeneous problems can be computed very cheaply. They consist of matrix exponentials for which methods of near optimal approximations are known [19].

In **space-time multigrid methods**, the parallelisation comes from the discretisation of the space-time domain, that is considered as one, as we will see again in the finite element section 3.2. As a rather recent example of this type we will look at an approach by M. Gander and M. Neumüller [20]. Suppose we are considering a simple heat equation of the form $u_t - \Delta u = f$ and discretise it in a space-time setting using an implicit method like Backward Euler in time and another method, for example a discontinuous Galerkin approach in space. One then obtains a block triangular system of the following form

$$\begin{bmatrix} A_1 & & & & \\ B_2 & A_2 & & & \\ & B_3 & A_3 & & \\ & & \dots & \dots & \\ & & & B_{\tilde{m}} A_{\tilde{m}} & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{\tilde{m}} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \\ f_{\tilde{m}} \end{bmatrix} \tag{3.28}$$

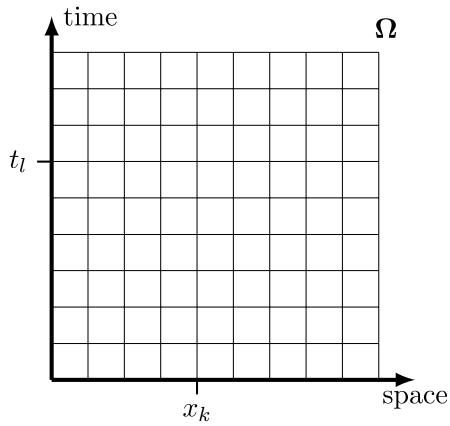
where each subset $u_i$ contains all spatial elements for a particular time interval. In the multigrid iteration they apply a block Jacobi smoother inverting each of the blocks $A_j$ before using a standard restriction operators in space-time to jump to a coarser grid, which is then repeated recursively on each level. For further details on multigrid methods we refer to chapter 5.

Some solution approaches in space-time can be categorized in multiple approaches, for example is a two-level multigrid method starting with an initial guess obtained from the coarse grid and using an upwind smoother the same as a simple parareal approach.

In this thesis we will subsequently consider a space-time multigrid approach but not exactly of the previous type for the beforementioned symmetry reason, see chapter 1, but instead use a continuous Galerkin space-time finite element assembly in addition to a first order least squares formulation which was introduced in the previous section [see 3.1 and 3.2]. The space-time formulation differs from a common finite element approach in the sense that our basis functions $\{\phi_1, ..., \phi_m\}$ are functions of time and space, i.e. $\phi_i = \phi_i(x, t)$, instead of only space, that is $\phi_i = \phi_i(x)$ for each time step or interval. Hence it is possible to assemble one big system of equations that covers the entire space-time domain which can then be solved using a multigrid approach and differs from above system [...] in the sense that there are symmetric upper and lower off-diagonal blocks.

*not really in line with causality principle ...?! say something about that? Has anyone also been using a continuous space-time galerkin approach?*

The discretisation of the domain which will also be referred to in the following chapters can be visualised as shown in figure [...].




where one has $n + 1$ points in space and $m + 1$ points in time. The tuples $(x_k, t_l)$ are then organized in the following manner, the $i$-th entry references $i = (n + 1) \cdot l + k$. That is we first label all elements of a certain time step before moving on to the next time step which will then be assembled into one overall system. Details of how this is done will follow in the implementation section.

## 3.4 Iterative Methods for Nonlinear Systems

Our original problem (1.1) is as mentioned before potentially nonlinear, but so far we have only been discussing ways of how to discretise linear problems. Let us therefore now undertake a short excursion of how to solve non-linear problems, which in each iteration step will include solving linearisations that are of the beforementioned type. We assume that we can write the nonlinear problem in the subsequent form and are therefore interested in solving questions of the following type

$$\text{Find } s \in \Omega \subset \mathbb{R}^m \text{ such that } J(s) = 0. \tag{3.29}$$

for some $J : \Omega \subset \mathbb{R}^m \to \mathbb{R}_{\geq 0}$. Since this describes a very broad class of problems there are of course many different approaches of how to tackle this question. Here we will introduce three well-known possibilities, gradient descent, and Newton's method, as well as a combination of the two in the form a trust region method.

### 3.4.1 Gradient Descent Methods

The method of gradient descent is a computationally inexpensive iterative optimisation algorithm used to find a local minimum of a function $J$. We only have to require that the function $J$ is differentiable in a neighbourhood of each current iterate $s_k$. After an initial guess $s_0$ is chosen, one takes a step in the direction of the negative gradient of $J$ at $s_0$, that is the direction of steepest descent. The iteration then looks as follows

$$s_{k+1} = s_k + \alpha_k(-\nabla J(s_k)) \tag{3.30}$$

If the scaling parameter $\alpha_k > 0$ is chosen sufficiently small, we know that $J(s_{k-1}) \geq J(s_k) \geq J(s_{k+1})$. If $||\nabla J(s_k)|| = 0$ we have found a local minimum and hence $s_{k+1} = s_k$. There are a number of strategies that try to select a suitable value for $\alpha_k$, one of them is for example a line search algorithm using the Wolfe conditions [21].

Under the assumption that $J \in C^1(\Omega)$, bounded and convex and particular choices for the $\alpha_k$, e.g. using an above mentioned line search, the method is guaranteed to converge to a local minimum which is due to the convexity of $J$ the unique global minimiser. However the speed of convergence is dependent on the condition number of the linearised hessian, and can therefore be extremely low if the condition number is high, even when performing an exact line search in every step. Consequently for inexact line search algorithms we cannot expect better convergence rates, and sometimes even have poor convergence rates for relatively well-conditioned problems [21]. Gradient Descent represents a first-order Taylor approximation of $J$ in $s_k$ and gives an updated solution based on this local linear model. A more sophisticated approach than this is for example Newton's method which uses a second order Taylor approximation.

### 3.4.2 Newton's Method

It is one of the most well-known and most commonly used methods to solve non-linear problems of the above type [22]. Let $J \in C^2(\Omega)$, and hence if differentiate both sides of the equation $J(s) = 0$, we obtain $\nabla J(s) = [0, 0, ..., 0]^T$, for which we would like to determine the unknown root $s$. We consider a Taylor series expansion for an initial guess $s_0$

$$\nabla J(s_0 + h) = \nabla J(s_0) + \nabla^2 J(s_0) \cdot h + o(||h||^2), \quad s_0 \in \Omega. \tag{3.31}$$

If we now neglect the higher order terms, setting $\nabla J(s_0 + h) = 0$ and replacing it by its first order Taylor approximation $\nabla J(s_0) + \nabla^2 J(s_0) \cdot h$, which we can then solve for $h$, under the assumption that $\nabla^2 J(s_0)$ is non-singular and use the result to update our initial guess $s_0$. One ends up the with iteration

$$s_{k+1} = s_k - [\nabla^2 J(s_k)]^{-1} \nabla J(s_k). \tag{3.32}$$

In the case of $J$ being convex and a few additional conditions one can achieve a quadratic rate of convergence for Newton's method compared to a linear one for the method of gradient descent. However except for the one-dimensional case, it is usually very hard or impossible to know if these conditions are actually fulfilled [8]. And in addition to the gradient of $J$, one also has to compute the inverse of the Hessian of $J$ in each iteration. For larger systems this is in most cases a rather difficult and computationally expensive problem, which in our case we will try to tackle using a multigrid method [see chapter 5].

Hence, a quadratic approximation to find a minimiser only makes sense for a locally convex neighbourhood, otherwise the Newton iteration might not lead to a decrease but instead an increase in energy as it might take an iteration step towards a local maximum. Therefore in order to make use of the faster rate of convergence in convex neighbourhoods of $J$ it makes sense to use a Newton iteration, while in non-convex neighbourhoods it can be preferrable to use a gradient descent step as it is guaranteed to not increase the value of the functional. One option to combine these two is by using a trust region algorithm which has an additional important parameter, the so-called trust region radius and will be introduced in the following section.

### 3.4.3 Trust Region Methods

Trust region methods is a generic term that comprises globalisation strategies to approximate minima of potentially non-convex optimisation problems. The objective function $J$ is approximated using a model function in a subset of the domain, the so-called trust region. If the model function seems to be a good local fit to the function, the size of the region is expanded, if it is not, the size of the region is reduced. The fit of the model is assessed by comparing the ratio $\rho_k$ of the expected reduction of the objective function by the model and the actual reduction of the function.

A typical iteration step $k$ can be described in the following way. We have a current trust region that is usually defined as a spherical area of radius $\Delta_k$ and a model function $m_k(p)$ that is supposed to locally approximate $J$ and where $p$ describes the update to the current solution, that is the new step to be taken. We therefore want to solve for $p$, hence we minimise over $p$ within the trust region radius to obtain a solution $p_k$ and can thus determine $\rho_k$, the ratio of the expected compared to the actual reduction. Depending on its value and the parameter thresholding we either reduce $\Delta_k$ if the approximation does not seem like a good fit, and then solve $m_k$ again for $p$ with a smaller $\Delta_{k+1}$. Otherwise we either enlarge it, if $||p_k|| = \Delta_k$, i.e. it is maximal or else leave it the same. If $\rho_k$ is not too small we compute the new solution $s_{k+1} = s_k + p_k$.

There are many options what to choose as a model function. Among the simplest approaches is the Cauchy point calculation that stems from gradient descent, other popular ones include steihaug's method or the so-called dogleg method [21]. The former uses a conjugate gradient method for a quadratic model function and the later includes a mixture of using information of the first and second derivative of $J$, which is what we used and which will be explained in more detail in the chapter on the implementation [see 6.6.3].