

Adaptive Multigrid Solvers for Space-Time Discretisations

Lisa Gaedke-Merzhäuser

January 10, 2019

Contents

1	Prologue	3
2	Cardiac Electrophysiology	6
2.1	Electrical Activity on the Cellular Level	8
2.2	The Monodomain Equation	9
3	Mathematical Ingredients	11
3.1	Finite Element Methods	11
3.1.1	General Setting	11
3.1.2	Discretisation	12
3.1.3	Matrix Formulation	13
3.2	Least Squares Finite Element Methods	14
3.3	Space-Time Solution Methods	16
3.4	Iterative Methods for Nonlinear Systems	19
3.4.1	Gradient Descent Methods	19
3.4.2	Newton's Method	19
3.4.3	Trust Region Methods	20
4	Derivation of a Least Squares Finite Element Space - Time Discretisation	22
4.1	Construction of an Equivalent Minimisation Problem	22
4.2	Function Spaces	24
4.3	Norm Equivalence	25
4.4	A Finite Element Space-Time Formulation	25
4.5	Computation of the Derivatives	27
4.6	Nonlinear Iteration Scheme	30
5	Multigrid Methods	32
5.1	Core Ideas	32
5.2	Basic Algorithm	33
5.3	Convergence Properties and Complexity	34
5.4	Coarse Space Construction Based on Eigenfunctions	34
6	Implementation and Numerical Results	35
6.1	Discretisation Scheme	35
6.2	Multigrid Implementation	36
6.2.1	Smoothers	37
6.3	Numerical Test Cases	40
6.3.1	Heat Equation	40
6.3.2	Linearisation of a Monodomain Equation	41

Contents	6.3.3 Monodomain Equation	42
7	Conclusions and Outlook	42
8	Epilogue	43

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

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

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 spatial derivatives are discretised and the time variables remain continuous which will give rise to a system of ordinary differential equations that is then to be solved by an appropriate method [source]. Another very common approach is to use a time stepping method in combination with for example a finite element discretisation in space [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, additionally the solution at a

given time usually depends on the previous one but not the other way around. However in current technological development where there is no further significant increase in 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. Both methodologies mentioned above contain inherently sequential processes. As for example for the latter only the space dimensions allow for parallelisation and as this saturates [source] there is no possibility for a further speed up. Thus it computationally only makes sense to look for methods that utilise a parallelisation in space and time simultaneously. This in turn naturally invites for 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 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 preferable properties as possible. However most space-time solution methods do not give rise to symmetric positive definite systems [5] which is why we decided to recast problem (1.1) as an optimisation problem, 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 over. 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 symmetry 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 successively 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 to 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 minimisation 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 an adaptive 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 multigrid 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. But then an additional 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 atrioventricular 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.

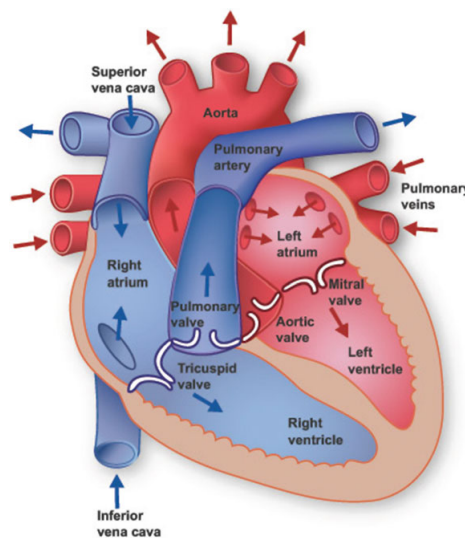


Figure 2.1: Scheme of the Heart [3]

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 its 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 mitral 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 valves 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].

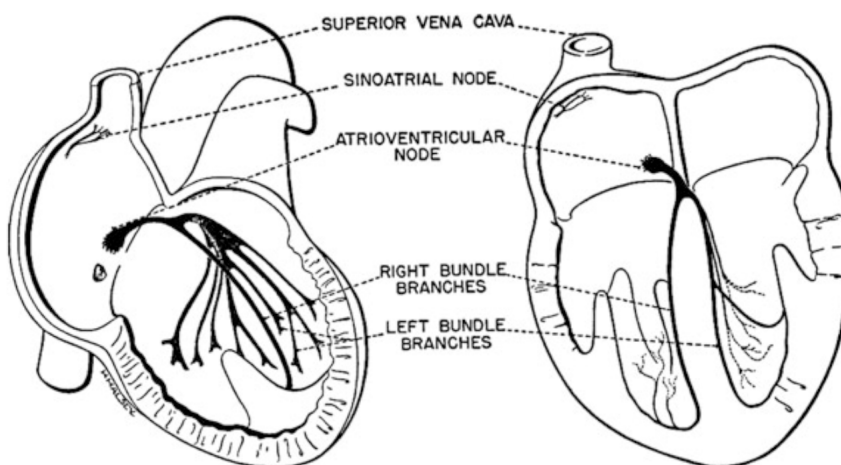


Figure 2.2: Overview of the Hearts Conduction System [3]

The heart contractions are initiated by an electric activation, that is a depolarizing transitory 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 ventricles. 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 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 μm by 30-40 μm . (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.

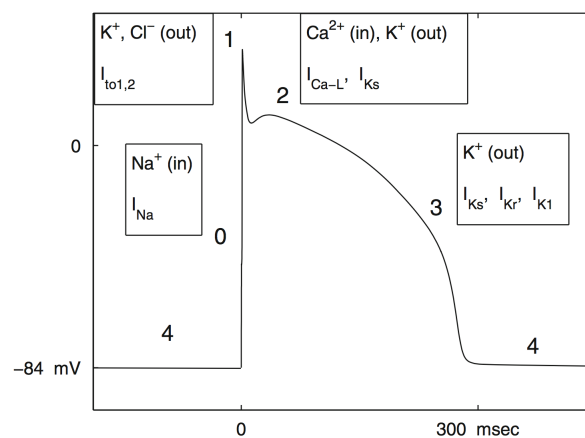


Figure 2.3: Different Phases of Cardiac Action Potential [3]

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 transmembrane potential passes from negative to positive values.

Phase 1: Outward flow of K^+ and Cl^- ions after the inactivation 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 resistances 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.*

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

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 involved 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 $\mathcal{A} : X \rightarrow Y$, $f \in Y$, we consider the problem:

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

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 $a(\cdot, \cdot) : V \times W \rightarrow \mathbb{R}$, and a bounded linear functional $L_f(\cdot) : W \rightarrow \mathbb{R}$ and is given by

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

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 doesn't 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

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

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 $\tilde{\mathcal{A}} : V \rightarrow V^*$ given by

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

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 a unique (!) element \tilde{f} through the relation

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

The variational formulation is therefore equivalent to the problem

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

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

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

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

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

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 α and the bound from above β , we see that

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

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

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

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 α and β .

If we assume that we have a discretisation Ω_h of our domain Ω , where $h > 0$ is a parameter depending on the mesh size. We furthermore 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

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

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 $\|u - u_h\|_V \rightarrow 0$ as $h \rightarrow 0$. 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 $\{\phi_1, \phi_2, \dots, \phi_n\}$ 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, \dots, u_n are constant coefficients. Writing (3.21) in terms of the basis we obtain by linearity

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

If we now write this as a system of the form $A_h u_h = L_h$ with 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 $\{\phi_1, \dots, \phi_n\}$ such that $\text{supp}(\phi_i) \cap \text{supp}(\phi_j) = \emptyset$ 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 $\mathcal{A} : X \rightarrow Y$, that is for some $\alpha, \beta > 0$:

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

We consider the problem and the least squares functional:

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

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

which poses the minimisation problem:

$$\operatorname{argmin}_{u \in X} J(u; f) \quad (3.15)$$

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

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

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

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

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

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

and corresponds to equation (3.9), with $\tilde{\mathcal{A}} = \mathcal{A}^* \mathcal{A}$, $\tilde{f} = \mathcal{A}^* f$ 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

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

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 $\{\phi_1, \dots, \phi_n\}$ 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

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

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

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

- (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 features 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 $\hat{\mathcal{A}} = \mathcal{A}^* \mathcal{A}$ is of higher order than the one in the original formulation, which therefore requires higher regularity assumptions which might be unpreferable 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 Ω into time slabs Ω_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 $0 = t_0 < t_1 < \dots < t_m = T$. 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 Ω_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

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

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

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

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 Ω 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:

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

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

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

and afterwards the overlapping homogeneous problem is solved:

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

Due to linearity the overall solution can be obtained through summation

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

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 redundancy 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

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

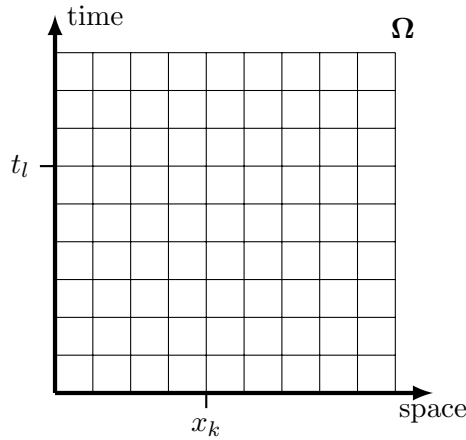
where each subset u_i contains all spatial elements for a particular time interval. In the multigrid iteration they apply a block Jacobi smoother inverting each of the blocks A_j before using a standard restriction operators in space-time to jump to a coarser grid, which is then repeated recursively 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 $\{\phi_1, \dots, \phi_m\}$ 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

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

for some $J : \Omega \subset \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$. 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

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

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 α_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 α_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 $\nabla J(s) = [0, 0, \dots, 0]^T$, for which we would like to determine the unknown root s . We consider a Taylor series expansion for an initial guess s_0

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

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

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

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 preferable 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 ρ_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 Δ_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 ρ_k , the ratio of the expected compared to the actual reduction. Depending on its value and the parameter thresholding we either reduce Δ_k if the approximation does not seem like a good fit, and then solve m_k again for p with a smaller Δ_{k+1} . Otherwise we either enlarge it, if $\|p_k\| = \Delta_k$, i.e. it is maximal or else leave it the same. If ρ_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].

Chapter 4

Derivation of a Least Squares Finite Element Space - Time Discretisation

4.1 Construction of an Equivalent Minimisation Problem

In this chapter we will tie the beforementioned concepts together to derive a discretised problem formulation that can subsequently be turned into an algorithm capable of approximating reaction diffusion equations. In order to do so let us firstly set the ground for the overall framework we are looking at. We consider a space-time domain

$$\Omega = \mathcal{S} \times \mathcal{T}, \quad \mathcal{T} = (0, T), \quad T > 0 \text{ and } \mathcal{S} \subset \mathbb{R}^N, \quad N = 1, 2, 3 \quad (4.1)$$

where \mathcal{T} represents the time domain and \mathcal{S} is the domain in space which we require to be Lipschitz regular. We may have a mixture of Dirichlet and Neumann boundary conditions on the boundary of Ω which we will denote by Γ and are labeled as $\Gamma_D, \Gamma_N \subset \Gamma$ respectively. We further assume them to be such that the problem is well posed. *Is this enough?* The class of partial differential equations introduced in the prologue that we would like to solve for then reads as the following:

$$\begin{aligned} \partial_t u - \operatorname{div}(D(x)\nabla u) &= f(u) & (x, t) \in \Omega \\ u &= g_D & (x, t) \in \Gamma_D \\ \nabla u \cdot n &= g_N & (x, t) \in \Gamma_N \end{aligned} \quad (4.2)$$

It describes a parabolic partial differential equation with a non-linear right hand side where the divergence is defined as $\operatorname{div}(\sigma) = \frac{\partial \sigma}{\partial x_1} + \dots + \frac{\partial \sigma}{\partial x_N}$ for $\sigma = \sigma(x, t)$ and $\nabla u = [\frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_N}]^T$ for $u = u(x, t)$. And where we further assume that $D(x)$ is a bounded, symmetric positive definite matrix of size $N \times N$ with functions in $L^2(\mathcal{S})$ for almost all $x \in \bar{\mathcal{S}}$. *Are these sufficient assumptions on $D(x)$?* Typically we will have that $u(x, 0) = u_0 = g_D$ for all $x \in \mathcal{S}$ and Neumann boundary conditions on the boundary of \mathcal{S} for $t \in (0, T)$.

The next step will be to derive an equivalent optimisation problem whose solution therefore then coincides with the solution of (4.2) at least in a weak sense. Since we are entirely working with finding solutions in Sobolev spaces in the least squares setting we can generally only require equivalence to a primal weak formulation of (4.3) and (4.2) or equivalence with respect to the solution space of the variational formulation, for a further discussion we refer to [source]. Generally when working with least squares finite element formulations choosing a suitable solution space V and data space Z is often non trivial as there are a number of difficulties that can arise. One usually faces a trade off between constructing a mathematically well-defined problem and

allowing for a relatively simple, efficient, robust while still accurate implementation. Therefore to make the methodology of LSFEMs competitive compared to other approaches like Galerkin approximations further considerations need to be taken into account. We saw in the previous chapter that it is possible to derive least squares formulations that recover the properties of the Rayleigh- Ritz setting, however one hindrance one encounters in this setting as well as in many others is the higher order operator arising in (3.25), that would require a solution space of higher regularity. When considering a simple Poisson equation with Dirichlet boundary conditions this would for example imply that we would require the solution u to be from H_0^2 , instead of H_0^1 [7]. This does not only heavily limit the set of admissible solutions but it is additionally much harder to construct appropriate finite dimensional subspaces for and is therefore impractical to use. In order to succumb this obstacle we will recast (4.2) as a system of coupled equations only containing first order derivatives to apply the methodologies introduced in section (3.4) at the price of introducing an additional variable.

$$\begin{aligned} \partial_t u - \operatorname{div}(\sigma) &= f(u) & (x, t) \in \Omega \\ \sigma &= D(x) \nabla u & (x, t) \in \Omega \\ u &= g_D & (x, t) \in \Gamma_D \\ \nabla u \cdot n &= g_N & (x, t) \in \Gamma_N \end{aligned} \quad (4.3)$$

Rearranging this equation into a vector form we obtain in Ω

$$\begin{pmatrix} I & -D(x) \nabla \\ -\operatorname{div} & \frac{\partial}{\partial t} \end{pmatrix} \begin{pmatrix} \sigma \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ f(u) \end{pmatrix} \quad (4.4)$$

which can be shortened to $\mathcal{A}([\sigma, u]) = \tilde{f}(u)$, where \mathcal{A} denotes the differential operator and $\tilde{f}(u)$ the right hand side.

Show that this operator is linear and bounded? but then norm ... spaces ... bla bla bla ...

Remark. It is not clear yet though in which space we will be looking for a solution. In general when working with strong and weak formulations there is a number of function spaces involved, problem formulations that are similar but not the same and it is difficult to represent them in clear but short notation. This can easily lead to some confusion. In the current case the situation is even more complicated since we are trying to derive a weak formulation in a least squares setting, while additionally there obviously exist the regular or so-called primal weak formulations of (4.2) which we will sometimes refer to. Therefore we will first derive a least squares functional J with a corresponding variational formulation, postponing the definition of appropriate spaces for now but simply assuming them to already be well-defined.

So instead of looking for a solution of the strong formulation, let us now turn to the derivation of an optimisation problem that we would like to satisfy in a weak sense. The properties that we would like to be fulfilled are the following. We want a solution of (4.3) to be a global minimum of the optimisation problem, independently of the choice of the spaces that we are using and its associated norm, hence we additionally want the least squares functional J to be zero for a solution of (4.3). On the other hand if $J(\sigma, u) = 0$ then the original problem (4.3) also has to be satisfied if only in a weak sense with respect to the spaces U and Z . We can check that all three properties hold for the functional

$$\tilde{J}(\sigma, u) = \|u_t - \operatorname{div}(\sigma) - f(u)\|_Z^2 + \|\sigma - D(x) \nabla u\|_Z^2 \quad (4.5)$$

Furthermore it has shown to be practical to be able to weight the two terms with coefficients [source] which does not affect the overall solution but grants us the possibility to numerically give more importance to one term than the other. We also introduce additional scalars of 0.5 to

simplify further computations of the derivatives as we will see in the following section. Therefore the the minimisation problem then reads

$$\min_{(\sigma, u) \in U} J(\sigma, u) = \frac{1}{2} c_1 \|u_t - \operatorname{div}(\sigma) - f(u)\|_Z^2 + \frac{1}{2} c_2 \|\sigma - D(x) \nabla u\|_Z^2. \quad (4.6)$$

J now defines an energy we can minimise over. If we consider the functional for $f = 0$ we obtain

$$J([\sigma, u], 0) = \frac{1}{2} c_1 \langle u_t - \operatorname{div}(\sigma), u_t - \operatorname{div}(\sigma) \rangle_Z + \frac{1}{2} c_2 \langle \sigma - D(x) \nabla u, \sigma - D(x) \nabla u \rangle_Z. \quad (4.7)$$

which more specifically gives rise to the following bilinear form if we differentiate J with respect to directional derivatives τ and v , and subsequently sum over them [see section 4.5 for the derivation].

$$\mathcal{B}([\sigma, u], [\tau, v]) = \left\langle \begin{pmatrix} I & -D\nabla \\ -\operatorname{div} & \frac{\partial}{\partial t} \end{pmatrix} \begin{pmatrix} \sigma \\ u \end{pmatrix}, \begin{pmatrix} I & -D\nabla \\ -\operatorname{div} & \frac{\partial}{\partial t} \end{pmatrix} \begin{pmatrix} \tau \\ v \end{pmatrix} \right\rangle_Z \quad (4.8)$$

The resulting candidate for a variational formulation can be derived as usual [source] that is as the *sum* of the directional derivatives of J , where the *directions* become the testfunctions.

$$\begin{aligned} &\text{Find } (\sigma, u) \in U \text{ such that } \mathcal{B}([\sigma, u], [\tau, v]) = L_{\tilde{f}}(u)(v) \quad \forall (\tau, v) \in W \\ &\text{with } \mathcal{B}([\sigma, u], [\tau, v]) = (\mathcal{A}[\sigma, u]^T, \mathcal{A}[\tau, v]^T) \text{ and } \mathcal{L}_{\tilde{f}}(u)([\tau, v]) = (\mathcal{A}[\tau, v], \tilde{f}(u))_Z \end{aligned} \quad (4.9)$$

Remark: $L_{\tilde{f}}(u)(v)$ is a linear operator in v but non-linear in u . What remains to be determined are the function spaces V , W , Z which potentially even lead us to a setting where can show norm equivalence in order for *Theorem 3* to hold.

4.2 Function Spaces

In order to define V and Z let us consider the optimisation problem [...] and its variational formulation [...] again. We would like to allow for the broadest class of solutions possible while still ensuring that all terms are well-defined in V , and staying away from Sobolev spaces of negative or fractional powers due to the beforementioned practicality reasons. Additionally to guarantee the existence of all terms involved we have to make sure that the present weak derivatives of σ and u exist in the induced inner product of Z . Let us therefore consider the following spaces

$$H_{\operatorname{div}}^1(\mathcal{S}) = \{\sigma \in (L^2(\mathcal{S}))^n : \operatorname{div}(\sigma) \in L^2(\mathcal{S})\} \quad (4.10)$$

$$H_{DIV}^1(\Omega) = H_{\operatorname{div}}^1(\mathcal{S}) \times L_2(\mathcal{T}) \quad (4.11)$$

where and when to use \times , overall notation

$$U = W = H_{DIV}^1(\Omega) \times H^1(\Omega) \quad (4.12)$$

$$Z = L^2(\Omega) \quad (4.13)$$

that is we require that

$$\sigma \in H_{DIV}^1(\Omega) \text{ and } u \in H^1(\Omega) \text{ with} \quad (4.14)$$

$$\|\sigma\|_{H_{DIV}^1(\Omega)} = \|\sigma\|_{L^2(\Omega)} + \|\operatorname{div}(\sigma)\|_{L^2(\Omega)}, \quad (4.15)$$

$$\|u\|_{H^1(\Omega)} = \|u\|_{L^2(\Omega)} + \|\nabla u\|_{L^2(\Omega)} + \|u_t\|_{L^2(\Omega)} \text{ and hence} \quad (4.16)$$

$$\|(\sigma, u)\|_U = \|\sigma\|_{H_{DIV}^1(\Omega)} + \|u\|_{H^1(\Omega)} \quad (4.17)$$

$$\text{CORRECT?!} \quad (4.18)$$

We can then check that all above terms in $J([\sigma, u], 0)$ are well defined. If we additionally assume $f(u) \in L^2(\Omega)$ for all u , the problem remains to be well-posed. Theoretically one could potentially only require $f \in H^{-1}(\Omega)$ but for the scope of this thesis, we will restrict ourselves to the former. Especially as we will later on have to require f to be twice differentiable to actually solve the problem numerically, [see Newton's method, section ...].

what do we get on \mathcal{A} with respect to what space? bounded? in previous section would have the bounds from X , how to do this here? how does this even make sense, hilbert space, strong form?

Another point that has not been mentioned so far but will have to be taken into account is the way of how to treat the boundary conditions in least-squares formulations. One possibility is to also include them in the functional as an additional term while another one would be to directly include them in the discretised system of the space. The former one entails the additional definition of an appropriate norm on the boundary while also requiring the treatment of the additional term. Since we have assumed it to be at least L_2 -regular the appropriate conditions can directly be imposed as part of the discretised system which will be discussed in more detail in the implementation section.

4.3 Norm Equivalence

\mathcal{B} induces an inner product on V and subsequently a norm. It is clearly a symmetric bilinear form. We also have that $\mathcal{B}([\sigma, u], [\sigma, u]) \geq 0$ for all $[\sigma, u] \in V$, it only remains to show that $\mathcal{B}([\sigma, u], [\sigma, u]) = 0 \Rightarrow [\sigma, u] = 0$. Show positive definiteness :

We clearly have $\mathcal{B}([\sigma, u], [\sigma, u]) \geq 0$ for all $(\sigma, u) \in V$ since $J([\sigma, u], 0)$ is the sum of two squared L^2 -norms. It remains to show that

$$\mathcal{B}([\sigma, u], [\sigma, u]) = 0 \iff (\sigma, u) = 0 \quad (4.19)$$

Can't be right, the point is that J is zero at the solution, so unless the solution is zero ...

In order to set a theoretical framework for the space induced by the scalarproduct \mathcal{B} we would like to establish norm equivalence comparable to the one in section 3.3 or in the paper of Z. Cai et al. on "First-order system least squares for second-order partial differential equations: Part I", [23], where they show ellipticity of their derived functional with respect to their *solution* space. That is we would like to know if there exists constants $\alpha, \beta > 0$ such that

$$\alpha \|(\sigma, u)\|_U \leq J([\sigma, u], 0) \leq \beta \|(\sigma, u)\|_U \quad \forall [\sigma, u] \in U. \quad (4.20)$$

The spaces differ from [23] in terms of the H_{DIV} definition, therefore we cannot apply their result. However it might still be possible to proceed similarly to them to show that these constants exists, it is however beyond the scope of thesis to fully check this but could be an interesting extension to the topic.

4.4 A Finite Element Space-Time Formulation

After having derived a continuous least squares formulation let us turn towards deriving a finite element discretisation of the problem. We want to consider conforming subspaces of U and W , hence we will be considering finite dimensional spaces $U_h \subset U$ and $W_h \subset W$ that are also defined on the entire space-time domain.

U_h contains the solution space for σ and u . They can be chosen independently from each other which can potentially be advantageous due to the different occurrences of their partial derivatives. Hence let us assume that $\sigma \in \tilde{V}_h$ and $u \in V_h$, where $U_h = \tilde{V}_h \times V_h$.

Suppose we have $\tilde{n} = \dim(\tilde{V}_h)$ and let $\{\tilde{\phi}_1, \dots, \tilde{\phi}_{\tilde{n}}\}$ be a basis of \tilde{V}_h and similarly $n = \dim(V_h)$ and $\text{span}\{\phi_1, \dots, \phi_n\} = V_h$. We furthermore assume V_h to be constructed such that $\inf_{u_h \in V_h} \|v - u_h\|_V \rightarrow 0$ as $h \rightarrow 0$ for all $v \in U$ and respectively the same for σ_h and \tilde{V}_h . It is also worth noting that since we are in a space-time setting we have $\tilde{\phi} = \tilde{\phi}(x, t)$ and $\phi = \phi(x, t)$.

We can then represent σ and u as a linear combination of basis functions in V_h or equivalently \tilde{V}_h , that is

$$\sigma_h(x, t) = \sum_{i=1}^{\tilde{n}} \sigma_i \tilde{\phi}_i(x, t) \quad u_h(x, t) = \sum_{i=1}^n u_i \phi_i(x, t) \quad (4.21)$$

The functional J then looks as follows

$$J(\sigma_h, u_h) = \left\| \sum_{i=1}^n u_i (\phi_i)_t - \sum_{i=1}^{\tilde{n}} \sigma_i \text{div}(\tilde{\phi}_i) - f \left(\sum_{i=1}^n u_i \phi_i \right) \right\|_Z^2 + \left\| \sum_{i=1}^{\tilde{n}} \sigma_i \tilde{\phi}_i - D(x) \nabla \left(\sum_{i=1}^n u_i \phi_i \right) \right\|_Z^2 \quad (4.22)$$

which will give rise to a vector-valued system once we discretise the domain.

In the arising variational formulation that we will also have to consider finite dimensional subspaces of W_h of W . In the scope of this thesis we restrict ourselves to the assumption that $W_h = U_h$. That is we introduce a set of test functions consisting of the basis vectors of \tilde{V}_h and V_h . That is the discretised weak form then reads

$$\text{Find } (\sigma_h, u_h) \in U_h \text{ such that } B \cdot [\sigma_h, u_h]^T = L_{\tilde{f}}(u_h) \quad (4.23)$$

where $B \in \mathbb{R}^{m \times m}$, with $m = \tilde{n} + n$, be the matrix arising from the discretised bilinear operator, and $L_{\tilde{f}}(u_h) \in \mathbb{R}^m$ being a discretised right-hand side. Since we assume that the solution $s_h = [\sigma_h, u_h]$ first contains all values corresponding to σ and then for u we obtain a block structure for B and $L_{\tilde{f}}$ of the following form.

$$B = \begin{bmatrix} B_{\sigma\sigma} & B_{\sigma u} \\ B_{u\sigma} & B_{uu} \end{bmatrix} \quad L_{\tilde{f}}(u_h) = \begin{bmatrix} (L_{\tilde{f}}(u_h))_{\sigma} \\ (L_{\tilde{f}}(u_h))_u \end{bmatrix} \quad (4.24)$$

Each entry of each of the blocks of B can be computed explicitly according to the subsequent schemes.

$$\text{For } B_{\sigma\sigma} : \quad B_{ij} = \langle \tilde{\phi}_j, \tilde{\phi}_i \rangle_Z + \langle \text{div}(\tilde{\phi}_j), \text{div}(\tilde{\phi}_i) \rangle_Z \quad \forall i, j \in \{1, \dots, \tilde{n}\} \quad (4.25)$$

$$\text{For } B_{\sigma u} : \quad B_{ij} = -\langle D(x) \nabla \phi_j, \tilde{\phi}_i \rangle_Z - \langle (\phi_j)_t, \text{div}(\tilde{\phi}_i) \rangle_Z \quad \forall i \in \{1, \dots, \tilde{n}\}, j \in \{\tilde{n} + 1, \dots, m\} \quad (4.26)$$

$$\text{For } B_{u\sigma} : \quad B_{ij} = -\langle D(x) \nabla \phi_i, \tilde{\phi}_j \rangle_Z - \langle (\phi_i)_t, \text{div}(\tilde{\phi}_j) \rangle_Z \quad i \in \{\tilde{n} + 1, \dots, m\}, \forall j \in \{1, \dots, \tilde{n}\} \quad (4.27)$$

$$\text{For } B_{uu} : \quad B_{ij} = \langle D(x) \nabla \phi_j, D(x) \nabla \phi_i \rangle_Z + \langle (\phi_j)_t, (\phi_i)_t \rangle_Z \quad \forall i, j \in \{\tilde{n} + 1, \dots, m\} \quad (4.28)$$

In the case that f is independent of u , that is $f = 0$ or $f = f(x, t)$, the derivative of f with respect to u , is clearly zero and therefore the right-hand side, which we will denote by

$L_{\tilde{f}} = L_{\tilde{f}}(u_h)$ to underline its independence of u only contains the following terms

$$(L_{\tilde{f}})_i = 0 \quad i \in \{1, \dots, \tilde{n}\} \quad (4.29)$$

$$(L_{\tilde{f}})_i = P\left(\sum_{j=1}^{\tilde{n}} \langle \operatorname{div}(\tilde{\phi}_j), f \rangle_Z\right)_i - \langle (\phi_i)_t, f \rangle_Z \quad i \in \{\tilde{n} + 1, \dots, m\} \quad (4.30)$$

where P is some sort of projection in case $\tilde{n} \neq n$, how to really write this up?

Thus we now know how compute each term we can assemble one large linear system of equations

$$B \begin{pmatrix} \sigma \\ u \end{pmatrix} = L_{\tilde{f}} \quad (4.31)$$

which can then be solved immediately for $[\sigma, u]^T$ using for example a multigrid method. In the case of $f = f(u)$ the situation is more complicated, since we also have to take the derivatives of f with respect to u into account, and then construct a nonlinear iteration scheme and consider linearisations of the problem to solve for u unless we have that $f(u)$ is a linear function. In that case we would only obtain additional terms that can be added to B_{uu} .

4.5 Computation of the Derivatives

In order to formulate a nonlinear iteration scheme we will first derive the first and second order Gateaux derivatives of J to be able to find concrete formulations for the gradient and hessian. We formulate them here now in their continuous form and will discuss the discretisation in the subsequent section. Furthermore for the remainder of this chapter we will assume that $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_Z$ in order to simplify the notation. For brevity and clarity we split the functional J in three different terms that will be considered separately dividing it into the linear and nonlinear terms which is possible due to the linearity of the inner product.

$$\begin{aligned} J_1(\sigma, u) &= \frac{1}{2} c_1 \langle u_t - \operatorname{div}(\sigma), u_t - \operatorname{div}(\sigma) \rangle \\ J_2(\sigma, u) &= \frac{1}{2} c_1 \langle 2u_t - 2\operatorname{div}(\sigma) - f(u), -f(u) \rangle \\ J_3(\sigma, u) &= \frac{1}{2} c_2 \langle \sigma - D(x)\nabla u, \sigma - D(x)\nabla u \rangle \end{aligned} \quad (4.32)$$

We can check that $J(\sigma, u) = J_1(\sigma, u) + J_2(\sigma, u) + J_3(\sigma, u)$.

Since $u \in H^1(\Omega)$ and $\sigma \in H_{DIV}(\Omega)$ they are not defined pointwise which we have to take into account for the subsequent computations.

What changes, what doesn't? Weak derivatives.

The partial directional derivatives of J_1 can be determined by once again taking the linearity

of the inner product as well as its symmetry into account

$$\begin{aligned}
\frac{\partial J_1}{\partial \sigma} &= \lim_{\epsilon \rightarrow 0} \frac{J_1(\sigma + \epsilon \tau, u) - J_1(\sigma, u)}{\epsilon} \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (\langle u_t - \operatorname{div}(\sigma + \epsilon \tau), u_t - \operatorname{div}(\sigma + \epsilon \tau) \rangle - \langle u_t - \operatorname{div}(\sigma), u_t - \operatorname{div}(\sigma) \rangle) \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (\langle u_t, u_t \rangle - \langle u_t, \operatorname{div}(\sigma) \rangle - \epsilon \langle u_t, \operatorname{div}(\tau) \rangle - \langle \operatorname{div}(\sigma), u_t \rangle + \langle \operatorname{div}(\sigma), \operatorname{div}(\sigma) \rangle \\
&\quad + \epsilon \langle \operatorname{div}(\sigma), \operatorname{div}(\tau) \rangle - \epsilon \langle \operatorname{div}(\tau), u_t \rangle + \epsilon \langle \operatorname{div}(\tau), \operatorname{div}(\sigma) \rangle + \epsilon^2 \langle \operatorname{div}(\tau), \operatorname{div}(\tau) \rangle \\
&\quad - \langle u_t, u_t \rangle + \langle u_t, \operatorname{div}(\sigma) \rangle + \langle \operatorname{div}(\sigma), u_t \rangle - \langle \operatorname{div}(\sigma), \operatorname{div}(\sigma) \rangle) \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (-2\epsilon \langle u_t, \operatorname{div}(\tau) \rangle + 2\epsilon \langle \operatorname{div}(\sigma), \operatorname{div}(\tau) \rangle + \epsilon^2 \langle \operatorname{div}(\tau), \operatorname{div}(\tau) \rangle) \\
&= -c_1 \langle u_t, \operatorname{div}(\tau) \rangle + c_1 \langle \operatorname{div}(\sigma), \operatorname{div}(\tau) \rangle
\end{aligned} \tag{4.33}$$

By proceeding analogously for equation J_2 and J_3 we obtain in these cases:

$$\frac{\partial J_2}{\partial \sigma} = c_1 \langle \operatorname{div}(\tau), f(u) \rangle \tag{4.34}$$

$$\frac{\partial J_3}{\partial \sigma} = c_2 \langle \sigma, \tau \rangle - c_2 \beta \langle \tau, \nabla u \rangle \tag{4.35}$$

Let us now turn to the partial derivatives with respect to u . Here we obtain the following for J_1 and J_3 :

$$\begin{aligned}
\frac{\partial J_1}{\partial u} &= \lim_{\epsilon \rightarrow 0} \frac{J_1(\sigma, u + \epsilon v) - J_1(\sigma, u)}{\epsilon} \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (\langle (u + \epsilon v)_t - \operatorname{div}(\sigma), (u + \epsilon v)_t - \operatorname{div}(\sigma) \rangle - \langle u_t - \operatorname{div}(\sigma), u_t - \operatorname{div}(\sigma) \rangle) \\
&= c_1 \langle u_t, v_t \rangle - c_1 \langle v_t, \operatorname{div}(\sigma) \rangle
\end{aligned} \tag{4.36}$$

$$\frac{\partial J_3}{\partial u} = -c_2 \langle \sigma, D(x) \nabla v \rangle + c_2 \langle D(x) \nabla u, D(x) \nabla v \rangle$$

Repeat assumptions on $D(x)$. But can leave it where it is for now?

In the case of J_2 , we have to take the non-linearity of f into account. If we assume that f sufficiently smooth (what do we need exactly?!) that is

$$\lim_{\epsilon \rightarrow 0} f(u + \epsilon v) = f(u) \text{ and } \tag{4.37}$$

$$\lim_{\epsilon \rightarrow 0} \langle f(u + \epsilon v), f(u + \epsilon v) \rangle - \langle f(u), f(u) \rangle = \langle f'(u) \cdot v, f(u) \rangle + \langle f(u), f'(u) \cdot v \rangle \tag{4.38}$$

which can be added due to symmetry.

$$\begin{aligned}
\frac{\partial J_2}{\partial u} &= \lim_{\epsilon \rightarrow 0} \frac{J_2(\sigma, u + \epsilon v) - J_2(\sigma, u)}{\epsilon} \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (\langle 2(u + \epsilon v)_t - 2\operatorname{div}(\sigma) - f(u + \epsilon v), -f(u + \epsilon v) \rangle - \langle 2u_t - 2\operatorname{div}(\sigma) - f(u), -f(u) \rangle) \\
&= \lim_{\epsilon \rightarrow 0} \frac{c_1}{2\epsilon} (-2\langle u_t, f(u + \epsilon v) \rangle + 2\langle u_t, f(u) \rangle \\
&\quad - 2\epsilon \langle v_t, f(u + \epsilon v) \rangle \\
&\quad + 2\langle \operatorname{div}(\sigma), f(u + \epsilon v) \rangle - 2\langle \operatorname{div}(\sigma), f(u) \rangle \\
&\quad + \langle f(u + \epsilon v), f(u + \epsilon v) \rangle - \langle f(u), f(u) \rangle) \\
&= -c_1 \langle u_t, f'(u) \cdot v \rangle - c_1 \langle v_t, f(u) \rangle + c_1 \langle \operatorname{div}(\sigma), f'(u) \cdot v \rangle + c_1 \langle f(u), f'(u) \cdot v \rangle
\end{aligned}$$

(4.39)

Hence we obtain the following partial first order directional derivatives.

$$J_\sigma[\tau] = \frac{\partial}{\partial \sigma} J(\sigma, u)[\tau] = c_2 \langle \sigma, \tau \rangle + c_1 \langle \operatorname{div}(\sigma), \operatorname{div}(\tau) \rangle - c_2 \langle D(x) \nabla u, \tau \rangle - c_1 \langle u_t, \operatorname{div}(\tau) \rangle - c_1 \langle f(u), \operatorname{div}(\tau) \rangle \quad (4.40)$$

$$\begin{aligned} J_u[v] = \frac{\partial}{\partial u} J(\sigma, u)[v] = & c_1 \langle u_t, v_t \rangle - c_1 \langle v_t, \operatorname{div}(\sigma) \rangle - c_2 \langle \sigma, D(x) \nabla v \rangle + c_2 \langle D(x) \nabla u, D(x) \nabla v \rangle \\ & - c_1 \langle u_t, f'(u) \cdot v \rangle - c_1 \langle v_t, f(u) \rangle - c_1 \langle \operatorname{div}(\sigma), f'(u) \cdot v \rangle + c_1 \langle f(u), f'(u) \cdot v \rangle \end{aligned} \quad (4.41)$$

Following the same principles one can determine the second order partial derivatives whose derivation will only be briefly outlined here for the most difficult terms which are those including f .

$$\frac{\partial^2}{\partial \sigma^2} J[\tau][\rho] = c_2 \langle \rho, \tau \rangle + c_1 \langle \operatorname{div}(\rho), \operatorname{div}(\tau) \rangle \quad (4.42)$$

$$\frac{\partial^2}{\partial \sigma \partial u} [v][\tau] = \frac{\partial^2}{\partial u \partial \sigma} [\tau][v] = - \langle \tau, \nabla v \rangle - \langle v_t, \operatorname{div}(\tau) \rangle - \langle \operatorname{div}(\tau), f'(u) v \rangle \quad (4.43)$$

$$\begin{aligned} \frac{\partial^2 J}{\partial u^2} [v][w] &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (J_u(\sigma, u + \epsilon w)[v] - J_u(\sigma, u)[v]) \\ &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (c_1 \langle (u + \epsilon w)_t, v_t \rangle + c_2 \langle \nabla(u + \epsilon w), \nabla v \rangle - c_1 \langle (u + \epsilon w)_t, f'(u + \epsilon w) \cdot v \rangle \\ &\quad - c_1 \langle v_t, f(u + \epsilon w) \rangle - c_1 \langle \operatorname{div}(\sigma), f'(u + \epsilon w) \cdot v \rangle + c_1 \langle f(u + \epsilon w), f'(u + \epsilon w) \cdot v \rangle \\ &\quad - J_u(\sigma, u)[v]) \\ &= c_1 \langle w_t, v_t \rangle + c_2 \langle \nabla w, \nabla v \rangle \\ &\quad + \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (c_1 \langle u_t, f'(u + \epsilon w) \cdot v \rangle - c_1 \langle u_t, f'(u) \cdot v \rangle \\ &\quad - \epsilon \cdot c_1 \langle w_t, f'(u + \epsilon w) \cdot v \rangle \\ &\quad - c_1 \langle v_t, f(u + \epsilon w) \rangle + c_1 \langle v_t, f(u) \rangle \\ &\quad - c_1 \langle \operatorname{div}(\sigma), f'(u + \epsilon w) \cdot v \rangle + c_1 \langle \operatorname{div}(\sigma), f'(u) \cdot v \rangle \\ &\quad + c_1 \langle f(u + \epsilon w), f'(u + \epsilon w) \cdot v \rangle - c_1 \langle f(u), f'(u) \cdot v \rangle) \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 J}{\partial u^2} [v][w] &= c_1 \langle w_t, v_t \rangle + c_2 \langle \nabla w, \nabla v \rangle + c_1 \langle u_t, w^T f''(u) v \rangle - c_1 \langle w_t, f'(u) \cdot v \rangle - c_1 \langle v_t, f'(u) \cdot w \rangle \\ &\quad - c_1 \langle \operatorname{div}(\sigma), w^T f''(u) v \rangle + c_1 \langle f(u), w^T f''(u) v \rangle + \langle f'(u) \cdot w, f'(u) \cdot v \rangle \end{aligned} \quad (4.44)$$

Yet some more on weak derivatives etc.

Thus we have now computed the necessary terms needed for the gradient as well as the Hessian of J . So let us now construct the nonlinear iteration scheme in order to minimise J .

4.6 Nonlinear Iteration Scheme

We only require the whole thing to be differentiable in a weak sense not like mentioned before.

As discussed in the previous chapter [see section 3.4] the general approach to finding a minimiser of the functional J is to search for a tuple $[\sigma, u]$ for which $\nabla J([\sigma, u]) = 0$, while $\nabla^2 J([\sigma, u])$ is positive definite. We start with an initial guess $s_0 = s_{\text{init}} = [\sigma_{\text{init}}, u_{\text{init}}]$ and then successively try to decrease energy. In case of a Newton step by finding a quadratic approximation of J at the value of the current iterate s_k , where the updated solution s_{k+1} is the minimiser of the quadratic approximation. However if J is not convex, and hence $\nabla^2 J([\sigma, u])$ not positive definite, the extremum of the quadratic approximation can actually lead us to a maximum of J . Therefore if we want to use a Newton iteration and maintain a decrease or at least no increase of energy in every step, that is $J(s_0) \geq J(s_1) \geq \dots \geq J(s_k) \geq \dots$ we need to be checking for convexity. In order to perform a Newton step we have to compute the Hessian of J , as well as its gradient in each iterate. The other option that was discussed previously to obtain a reduction in energy is to use a gradient descent method where we simply take an iteration step into the steepest descent direction. Hence we only need to evaluate the gradient in this case, which is computationally much cheaper but only leads to a linear convergence rate, instead of a quadratic one in the convex Newton case.

There are different ways to linearise and discretise in f , that is we have to decide how to represent f in our finite dimensional subspace formulation. For simplicity we assumed that $f \in C^2$, and denote the first and second derivative with respect to u by f' and f'' . This is in line with the forcing term stemming from the FitzHugh-Nagumo model as well as many other physical applications [source]. Hence for each degree of freedom in u we can determine coefficients for f, f', f'' and represent them in the basis of V_h , that is

$$(f_h)_i = \sum_{i=1}^n f_i \phi_i, \quad (f'_h)_i = \sum_{i=1}^n f'_i \phi_i, \quad (f''_h)_i = \sum_{i=1}^n f''_i \phi_i \quad (4.45)$$

Therefore f_h, f'_h and f''_h now represent a finite dimensional approximation of f for one particular approximation of u . With this representation we can now proceed to formulate a discretisation of the gradient of J , linearised in $s_k = [\sigma_k, u_k]$ where we know that inner products are still well-defined since we are working in conforming subspaces.

We want $\nabla J = 0$

$$(L_{\tilde{f}}(u_h))_\sigma = \langle \text{div}(\sigma_h), f'_h \rangle_Z \quad (4.46)$$

$$(L_{\tilde{f}}(u_h))_u = -\langle u_t, f'_h \rangle_Z - \langle v_t, f_h \rangle_Z + \langle f_h, f'_h \rangle_Z \quad (4.47)$$

$$\nabla J_k = \nabla J(s_k) = B \begin{pmatrix} \sigma_k \\ u_k \end{pmatrix} - L_{\tilde{f}}(u_k) \quad (4.48)$$

For one step of gradient descent we therefore compute the update by

$$s_{k+1} = s_k + \alpha(-\nabla J_k) \quad (4.49)$$

where $\alpha > 0$ is a scaling parameter that can be chosen in different ways, for example a line search algorithm and which will be discussed in more detail in the implementation chapter.

From the gradient and previous section we can determine the discretised Hessian which is needed for a Newton step. We obtain

$$H_k = \nabla^2 J(s_k) = B_{\text{lin}} + Q \quad (4.50)$$

where $B_{\text{lin}} = B$ from before and Q contains the nonlinear part, that is

$$Q = \begin{bmatrix} 0 & Q_{\sigma u} \\ Q_{u\sigma} & Q_{uu} \end{bmatrix} \text{ where} \quad (4.51)$$

$$(Q_{\sigma u})_{ij} = (Q_{u\sigma})_{ji} = -\langle \text{div}(\tilde{\phi}), f'_h \rangle \quad (4.52)$$

$$s_{k+1} = s_k - H_k^{-1}(\nabla J_k) \quad (4.53)$$

that is we would like to solve the linear system of equations

$$e_{k+1} = -H_k^{-1}(\nabla J_k) \quad \text{with} \quad s_{k+1} = s_k + e_{k+1} \quad (4.54)$$

But since it is generally expensive to compute the inverse of a large matrix, even if H_k is sparse and symmetric, because this property does in general not translate to the inverse we apply a multigrid method to solve (4.33). That is we need to solve a linear system of equations in each Newton step.

put derivative to zero therefore if we assume local convexity must be an extremum, positive definite, means minimum, therefore variational formulation makes sense

As mentioned in the prologue we consider an iterative approach to solve this problem and will therefore consider linearisations of the problem. But before going into more detail about that, let us consider the variational formulation of the coupled reaction-diffusion system (4.3)

A solution to (4.3) will always be a minimiser of the functional J independent on the choice of Z , however the sequence of iterates is dependent on the choice of the norm.

After having derived this discrete least squares space-time finite element formulation we will in the next chapter be looking at a concrete implementation which contains numerical results for some sample problems.

Chapter 5

Multigrid Methods

5.1 Core Ideas

Multigrid Methods are an important class of algorithms to iteratively approximate linear systems of equations of the form $Au = f$ where $A \in \mathbb{R}^n$ is usually a sparse symmetric positive definite matrix, that arises from the discretisation of a differential equation. They were first developed in the early 1960s by the soviet mathematicians Fedorenko and Bachlwalow but only became more well-known and *developed* in the late 1970s through the works of Hackbusch [24] and Brandt [6]. The core idea behind them is to approximate the error between an estimate and the unknown actual solution "on different frequencies". Zerlegung. Due to the structure of the matrix of discretised differential equations we only have a local transport of information, therefore simple iterative solvers such as a Jacobi or Gauss-Seidel method damp high frequency components of the error fastest, which leads to a much smoother, lower frequency error after only a few iterations. However the low frequency error on the fine grid becomes high frequency error on the coarse grid and can therefore be damped there more efficiently also applying a smoother there. That is the usage of coarser grids accelerates the transport of low frequency error across the discretised domain. Additionally the coarser grids have significantly less degrees of freedom which makes them computationally much cheaper to handle, until we can solve the problem directly on the coarsest grid. The key is to then restrict and interpolate effectively between the series of nested grids to approximate the error the current and the actual solution.

Through this combination of coarse grid correction and smoothing one obtains a fast, mesh-size independent convergence rate *for elliptic problems* [source].

SMOOTHER

COARSENING STRATEGIES, cite that λ parameter paper

INTERPOLATION AND RESTRICTION OPERATOR

which allows us to consider the solution u , the approximate solution w as well as the difference between the two, the error $e = u - w$ as a linear combination of this basis. Furthermore ... frequency ... eigenfunctions ... eigenvectors. Therefore it is possible to differentiate between high and low frequency error where one generally assumes ... (further discussion of what high and low frequency is later ... physical connection to solution, $AMG \iff GMG$)

nested meshes

elliptic problems

In geometric multigrid we have a predefined sequence of nested meshes of specific coarsening factors that can vary in different directions and on different levels. The coarse level spaces still represent the original problem just at a lower resolution.

Algebraic multigrid on the other hand does not have predefined coarse level spaces but instead they are chosen according to a given rule, that takes the values of A (or other known properties of the problem) into account. This is favorable if ... but also more expensive to compute.

5.2 Basic Algorithm

Hence we need the following main ingredients to construct a multigrid method with $l + 1$ levels, where $l + 1$ represents the finest grid; a smoother S , a set of interpolation and restriction matrices I_l, I_{l-1}, \dots, I_1 and R_l, R_{l-1}, \dots, R_1 , representations of the matrix A on all levels, that is $A^l, A^{l-1}, A^{l-2}, \dots, A^0$ with $A_l = A$, an initial guess s_{init} . These we can then put together in a multigrid V-cycle who looks as follows.

Multigrid V-cycle

Let w^J be the initial guess (on the finest grid level). Then repeat the following until convergence criterium is met or number of iterations exceeds a certain threshold:

- do ν_{J_a} smoothing steps on $A^J u^J = f^J$ with initial guess w^J
- compute $f^{J-1} = I_J^{J-1} r^J$
- do ν_{J-1_a} smoothing steps on $A^{J-1} u^{J-1} = f^{J-1}$ with initial guess $w^{J-1} = 0$ (vector)
- compute $f^{J-2} = I_{J-1}^{J-2} r^{J-1}$
- do ν_{J-2_a} smoothing steps on $A^{J-2} u^{J-2} = f^{J-2}$ with initial guess $w^{J-2} = 0$ (vector)
- compute $f^{J-3} = I_{J-2}^{J-3} r^{J-2}$
- ...
- ...
- solve $A^0 u^0 = f^0$
- ...
- ...
- correct $w^{J-2} = w^{J-2} + I_{J-3}^{J-2} w^{J-3}$
- do ν_{J-2_b} smoothing steps on $A^{J-2} u^{J-2} = f^{J-2}$ with initial guess w^{J-2}
- correct $w^{J-1} = w^{J-1} + I_{J-2}^{J-1} w^{J-2}$
- do ν_{J-1_b} smoothing steps on $A^{J-1} u^{J-1} = f^{J-1}$ with initial guess w^{J-1}
- correct $w^J = w^J + I_{J-1}^J w^{J-1}$
- do ν_{J_b} smoothing steps on $A^J u^J = f^J$ with initial guess w^J

picture V-cycle?

5.3 Convergence Properties and Complexity

strengthened Cauchy - Schwarz necessary?

Theorem 2. *Convergence.*

5.4 Coarse Space Construction Based on Eigenfunctions

Practically important extensions of multigrid methods include techniques where no partial differential equation nor geometrical problem background is used to construct the multilevel hierarchy.[16] Such algebraic multigrid methods (AMG) construct their hierarchy of operators directly from the system matrix. In classical AMG, the levels of the hierarchy are simply subsets of unknowns without any geometric interpretation.

Definition 1. *Strong dependence.*

As briefly mentioned before

Concept of strong dependence. Different ways to define this, most commonly ...?

This is where later on the adaption to the monodomain equation is made because here we can take specific knowledge about the equation into consideration.

Chapter 6

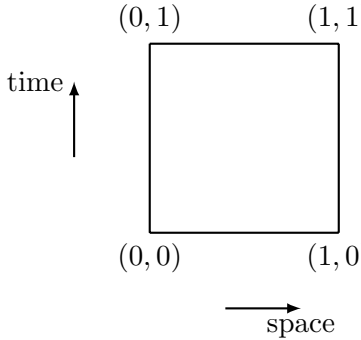
Implementation and Numerical Results

6.1 Discretisation Scheme

In this chapter we will look at how the *solver* we derived actually performs in some test cases. We looked at problems with a one-dimensional space domain $\mathcal{S} = (0, S)$ and a time-interval $\mathcal{T} = (0, T)$, that is

$$\Omega = (0, S) \times (0, T), \quad S, T > 0 \quad (6.1)$$

and which we discretised using uniformly-sized rectangular elements. Let N_x be the number of elements in space and N_t the number of elements in time. As a finite-dimensional approximation space we used piecewise linear polynomials in space and time for the approximation of σ_h as well as u_h and chose the local basis



$$\begin{aligned}
 \phi_{11}(x, t) &= (1 - x) \cdot (1 - t) \\
 \phi_{12}(x, t) &= x \cdot (1 - t) \\
 \phi_{21}(x, t) &= (1 - x) \cdot t \\
 \phi_{22}(x, t) &= x \cdot t
 \end{aligned} \quad (6.2)$$

If we translate this from local to the global coordinates and scale appropriately according to the mesh size with an appropriate projection, we have that

$$\phi_{ij}(x_k, t_l) = \begin{cases} 1 & \text{if } (k, l) = (i, j) \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

We arrange the grid points the same way the local basis is labeled, that is for a fixed t_j all elements in space with $x_i < x_{i+1}$, and subsequently all space elements for t_{j+1} . As previously mentioned the we arranged σ_h and u_h such that they are two concatenated vectors. Hence one obtains a system of equations with $m = (N_x + 1) \cdot (N_t + 1)$ degrees of freedom for σ_h and the same again for u_h . The solution vector is therefore of $2m$ while the matrices have a size of $2m \times 2m$. It is possible to use this local basis for σ_h and u_h because we are in the one-dimensional case and therefore $\nabla u = \partial_x u = \sigma$ and $\text{div}(\sigma) = \partial_x \sigma$. For $\dim(\mathcal{S}) > 1$ we would have to treat this problem

differently using for example Raviart-Thomas elements [source], which is however beyond the scope of this thesis *but could be of interest in future investigations*. In order to compute the individual integral terms we used a quadrature rule of degree three, that is we get the exact integral for the linear approximations of the functions. The admissible boundary conditions are a mixture of Dirichlet and Neumann type and are directly imposed in the system, either in u or σ , respectively.

6.2 Multigrid Implementation

We implemented a geometric multigrid V-cycle, where the number of elements on the coarse grid and the number of levels are chosen and then all other grids, interpolation operators, and finer level operators are generated automatically. The reason behind this is to guarantee nested meshes. As discussed previously there is no unique best problem independent coarsening strategy, and especially in the case of space-time discretisations it has a great effect on the overall performance. Unfortunately the literature on what could be a favourable strategy in a least squares space-time set up was very sparse and therefore we decided to use a classical space-time coarsening approach, where the size of each rectangle grows by a factor of 4 with each coarsening because the length of the element is doubles in each direction, see figure [...] below. It means that with each coarsening step the number of elements reduces by a factor of (2^{-2}) and therefore also the degrees of freedom. This seemed like a reasonable approach, since we are dealing with a coupled first order system, and hence considerations like in [20], that *showed* it would be preferable to keep the quotient $\lambda = \frac{d\Delta t}{\Delta x^2}$ (where d describes the diffusion constant and Δt and Δx the respective step sizes) close to one on all levels, did not really apply here because we have no second order derivative, that would lead to the square term in space. So under the assumption that the diffusion term d is not too different from one and we choose a similar stepsize in space and time, the standard space-time coarsening would lead to a coefficient that is the same on all levels as well as close to one. More extensive testing on this beyond the scope of this thesis. *Rough estimates showed that for the linear case this seemed to be the case as a interpolation operators of type [20] seemed to lead to slower convergence.*

We then chose to use interpolation weights according to the following scheme, where filled dots represent coarse grid points and the blank circles fine grid points

$$\begin{aligned}
 s(y_{C_i}) &= s(y_{C_i}) \quad \forall i \\
 s(y_{F_1}) &= \frac{1}{2}y_{C_1} + \frac{1}{2}y_{C_2} \\
 s(y_{F_2}) &= \frac{1}{2}y_{C_1} + \frac{1}{2}y_{C_3} \\
 s(y_{F_3}) &= \frac{1}{4}y_{C_1} + \frac{1}{4}y_{C_2} + \frac{1}{4}y_{C_3} + \frac{1}{4}y_{C_4} \\
 s(y_{F_4}) &= \frac{1}{2}y_{C_2} + \frac{1}{2}y_{C_4} \\
 s(y_{F_5}) &= \frac{1}{2}y_{C_3} + \frac{1}{2}y_{C_4}
 \end{aligned} \tag{6.4}$$

They are constructed respectively to go recursively from one level to the next, and this is for either σ_h or u_h . If \tilde{I} is an interpolation matrix of the above type then we will have that the overall interpolation operator will have the following form, that is we interpolate independently

for the two variables.

$$I = \begin{bmatrix} \tilde{I} & 0 \\ 0 & \tilde{I} \end{bmatrix} \quad (6.5)$$

The interpolation matrix from level $k - 1$ to level k is as before denoted by I_{k-1}^k and we have that $I_{k-1}^k \in \mathbb{R}^{2m_k \times 2m_{k-1}}$, where m_k and m_{k-1} denotes the number of points on the space time grid on the respective level.

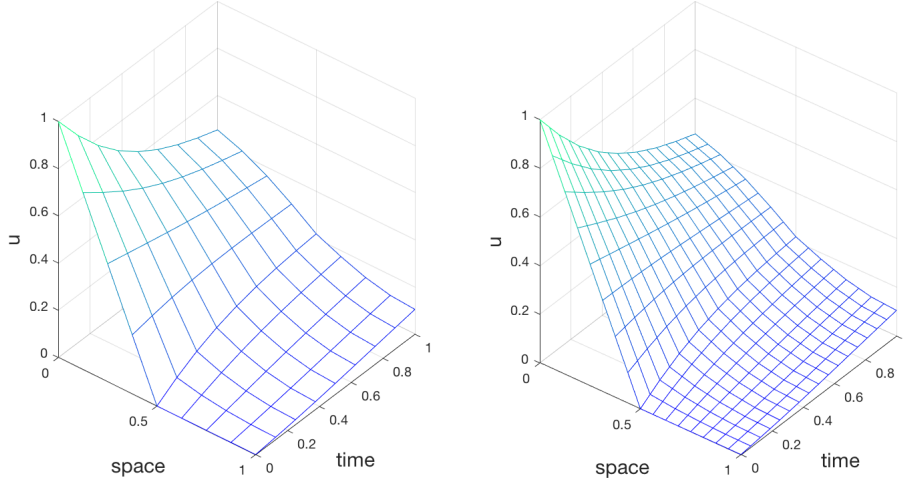


Figure 6.1: Interpolation of the solution to the heat equation with $u(x, 0) = \max(1 - 2x, 0)$, $D(x) = 0.1$ and homogeneous Neumann boundary conditions in time on a unit square from 8×8 to 16×16 elements. The interpolated solution of σ is not shown here as \tilde{I} simply applied to σ and u independently.

Visually the function on the right really appears to be a good interpolation of the function on the left, transmitting *global* information. *In the multigrid implementation we remove the influence of the boundary conditions.*

6.2.1 Smoothers

Smoothers are a key part of an efficient multigrid algorithm. And while many of them may theoretically be a possibility, in practice the choice of a suitable smoother is often not an easy one. We would like that the combination of coarse grid correction and smoother reduces the error efficiently for all frequencies. In the case of a geometric multigrid applied to an elliptic problem the coarse grid correction captures the low frequency error quickly while most smoothers like (block-) Jacobi or Gauss-Seidel reduce the high frequency error well and therefore the two are a favourable synthesis. But as we are also trying to develop a highly parallelisable solver it is of course important to also be taking that into account when choosing a smoother, therefore a regular Gauss-Seidel iteration is for example not a suitable choice as it works sequentially. In our set up we also have the additional feature of the two coupled variables σ_h and u_h , which are separated in the sense that they are two concatenated vectors but have the contentual connection. Therefore it might be favorable to use a (block-) smoother that takes this relationship into account. Hence we decided to use a Vanka type [source?] block Jacobi relaxation. That is a block Jacobi smoother where each block contains all degrees of freedom associated to one grid point in the space-time domain. That is we choose rectangular patches of size $p \times q$ on the

domain, that is p points in space and q in time, construct a submatrix containing all entries of A associated to those nodes for σ_h and u_h and all of the corresponding coupling terms and directly invert the small submatrix. We partition the entire domain like that, while potentially having smaller patches on the boundary, and finally assemble a preconditioner P containing the inverted submatrices. Due to the organisation of σ_h and u_h , P will not be a block diagonal matrix. For $p = q = 2$ we would have the following patches, where \tilde{c}_i describes the set of indices associated with the patch, and C_i the corresponding submatrix, which are each of size 8×8 .

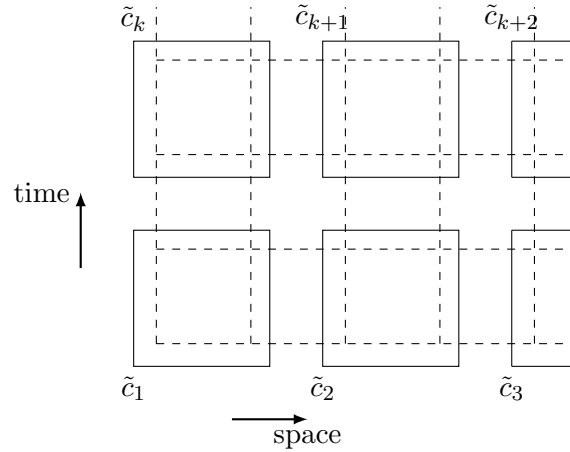


Figure 6.2: Schematic overview of the construction of the blocks of damped Jacobi smoother

damped Jacobi

Below we can see how the smoother acts on a random initial guess for the linear system $As = f$, where A is the least squares finite element matrix arising from the functional $J([\sigma, u], 0)$ with the discretisation chosen as described in chapter 4 and the previous parts of this chapter, and f is a zero right-hand side that only contains the boundary conditions. That is we look at the iteration

$$e_{k+1} = e_k + \omega P(f - Ae_k) \quad (6.6)$$

where P is the preconditioner and ω stands for the damping factor. Since we have a zero forcing term, the exact solution is zero everywhere if we have homogenous boundary conditions, and hence e_k describes the remaining error in each iteration starting from an initial guess.

We can see that the damped block Jacobi smoother really leads to a fast reduction of the high frequency error whereas the low frequency error is only reduced very slowly. *something about computationally very cheap*. Again we only plotted u as σ behaves in the same way, simply with different boundary conditions.

Block Jacobi 2x2 DIVERGES ?! can that be?

In this case they were chosen to be We used a ... grid and a patch size of ...

PICTURES AFTER ... MANY ITERATIONS

We can see that ... maybe another patch size ... difference?

Another type of smoother that we tested is a Gauss-Seidel *line smoother* [25] which can also be categorised as a Vanka smoother *write more about that reference?*. One considers each spatial degree of freedom individually but together for all times as well as the values for σ and u . We extract all matrix values for each of the x_i , to construct submatrices. We then first solve exactly for all odd subsets of \tilde{c}_i , here coloured in black and update the solution before solving for all

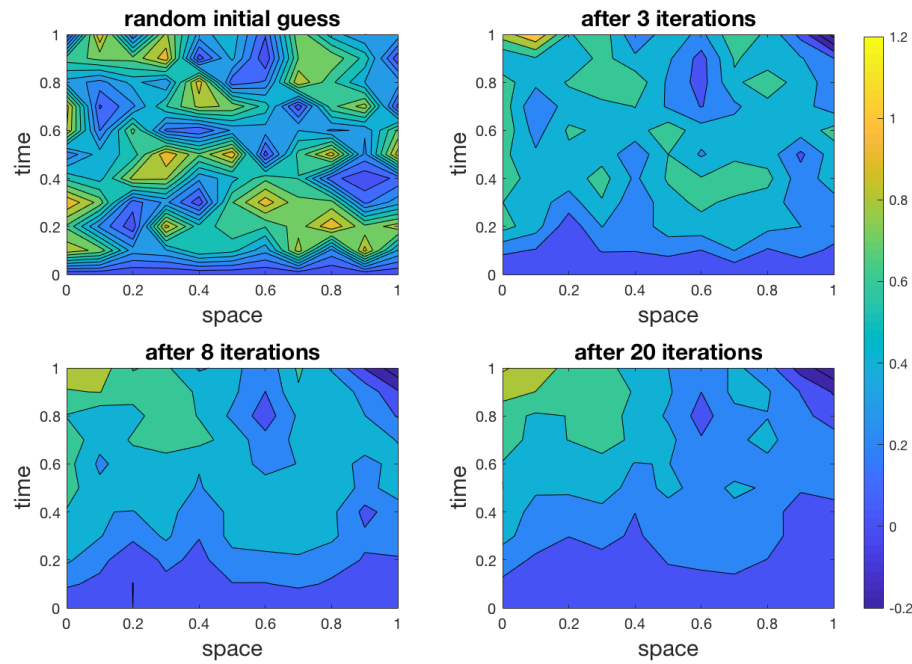


Figure 6.3: Iterates after $k = 0, 3, 8, 20$ iterations for a patch size of 1×1 , and a grid with 10×10 elements. needed more than 8000 iterations in order for the norm of the residual to be $< 10^{-9}$.

even subsets, here coloured in red. Hence for all subsets labeled in black we can solve in parallel, and subsequently for all subsets in red.

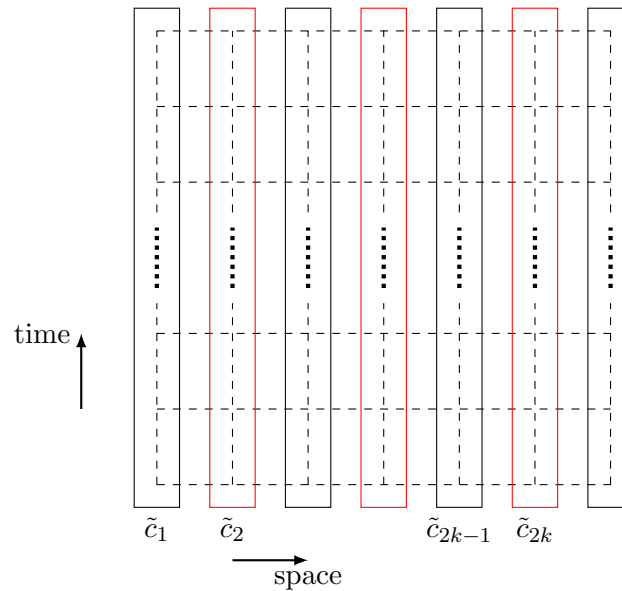


Figure 6.4: Schematic overview of the construction blocks in the Gauss Seidel line smoother

We repeated the same test as before where the preconditioner P is now defined by the above description.

As one might expect the line smoother converges much faster than the block Jacobi iteration,

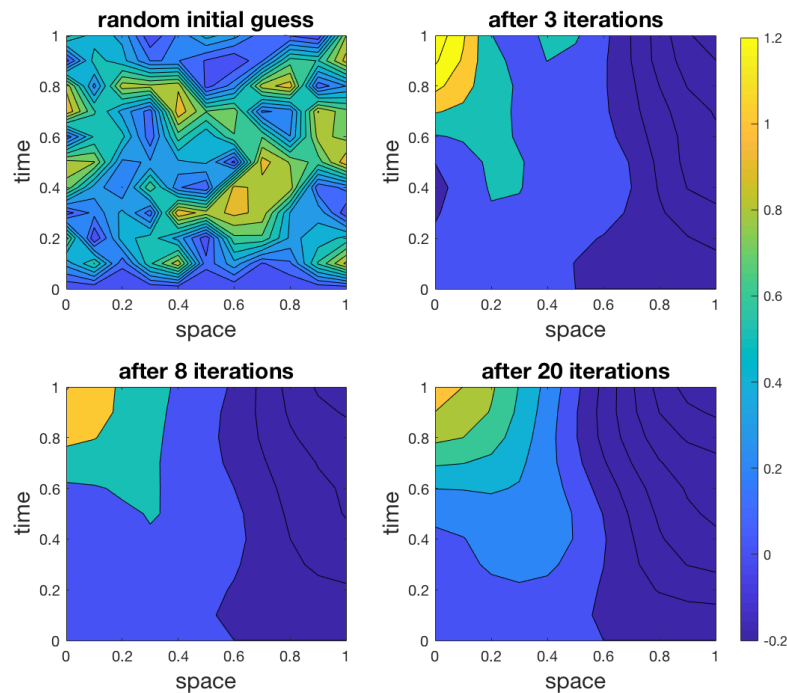


Figure 6.5: Iterates after $k = 0, 3, 8, 20$ iterations for the described line smoother, using a grid with 10×10 elements. It needed more than about 1300 iterations in order for the norm of the residual to be $< 10^{-9}$, where A, f as before

since the individual blocks we solve for exactly are larger. We can also see that the regions become more homogenous more quickly. *there are quite large areas where we go below 0, why?*. But obviously this comes with an increase in the computational cost.

6.3 Numerical Test Cases

6.3.1 Heat Equation

After having discussed the individual multigrid terms let us look at the overall multigrid performance for a heat equation using the derived LSFEM space-time discretisation.

parameters, boundary conditions, grid sizes, ...

pictures, this is how the solution looks like

some tables with convergence results also how many steps do we need until convergence of the smoothers by itself

further analysis? test laplace and u_t independently, what can we see from there?

Galerkin not Galerkin assembly, try with both

for the particular problem one is looking

6.3.2 Linearisation of a Monodomain Equation**6.3.3 Monodomain Equation**

talk about non convexity

Show what solution looks like, wavefront with initial conditions

- general construction
- talk about non-convexity
- then about nonlinear solvers

Chapter 7

Conclusions and Outlook

Potential Reasons Why Doesn't Work Well

- maybe a space-time approach where future influencing the past isn't a great idea, should follow causality principle?
-

Chapter 8

Epilogue

Bibliography

- [1] C. Cosner, “Reaction–diffusion equations and ecological modeling,” in *Tutorials in Mathematical Biosciences IV*, pp. 77–115, Springer, 2008.
- [2] P. A. Zegeling and H. Kok, “Adaptive moving mesh computations for reaction–diffusion systems,” *Journal of Computational and Applied Mathematics*, vol. 168, no. 1-2, pp. 519–528, 2004.
- [3] P. C. Franzone, L. F. Pavarino, and S. Scacchi, *Mathematical cardiac electrophysiology*, vol. 13. Springer, 2014.
- [4] P. Colli Franzone and L. F. Pavarino, “A parallel solver for reaction–diffusion systems in computational electrocardiology,” *Mathematical models and methods in applied sciences*, vol. 14, no. 06, pp. 883–911, 2004.
- [5] M. J. Gander, “50 years of time parallel time integration,” in *Multiple Shooting and Time Domain Decomposition Methods*, pp. 69–113, Springer, 2015.
- [6] A. Brandt, “Multi-level adaptive solutions to boundary-value problems,” *Mathematics of computation*, vol. 31, no. 138, pp. 333–390, 1977.
- [7] P. B. Bochev and M. D. Gunzburger, *Least-squares finite element methods*, vol. 166. Springer Science & Business Media, 2009.
- [8] P. Deuffhard, *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, vol. 35. Springer Science & Business Media, 2011.
- [9] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*, vol. 1. Siam, 2000.
- [10] “World health organisation, cardiovascular diseases,” Oct. 2018.
- [11] W. Harvey, *Exercitatio anatomica de motu cordis et sanguinis in animalibus*. Frankfurt, 1628.
- [12] A. C. of Cardiology, “cardiosmart, how the heart works,” Oct. 2018.
- [13] M. Potse, B. Dubé, J. Richer, A. Vinet, and R. M. Gulrajani, “A comparison of monodomain and bidomain reaction–diffusion models for action potential propagation in the human heart,” *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2425–2435, 2006.
- [14] L. C. Evans, “Partial differential equations,” 2010.
- [15] M. J. Gander and E. Hairer, “Nonlinear convergence analysis for the parareal algorithm,” in *Domain decomposition methods in science and engineering XVII*, pp. 45–56, Springer, 2008.

- [16] M. J. Gander, “Overlapping schwarz for linear and nonlinear parabolic problems,” 1996.
- [17] M. J. Gander, L. Halpern, and F. Nataf, “Optimal convergence for overlapping and non-overlapping schwarz waveform relaxation,” 1999.
- [18] S. Güttel, “A parallel overlapping time-domain decomposition method for odes,” in *Domain decomposition methods in science and engineering XX*, pp. 459–466, Springer, 2013.
- [19] M. J. Gander and S. Guttel, “Paraexp: A parallel integrator for linear initial-value problems,” *SIAM Journal on Scientific Computing*, vol. 35, no. 2, pp. C123–C142, 2013.
- [20] M. J. Gander and M. Neumuller, “Analysis of a new space-time parallel multigrid algorithm for parabolic problems,” *SIAM Journal on Scientific Computing*, vol. 38, no. 4, pp. A2173–A2208, 2016.
- [21] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 1999.
- [22] W. L. Briggs, S. F. McCormick, *et al.*, *A multigrid tutorial*, vol. 72. Siam, 2000.
- [23] Z. Cai, R. Lazarov, T. A. Manteuffel, and S. F. McCormick, “First-order system least squares for second-order partial differential equations: Part i,” *SIAM Journal on Numerical Analysis*, vol. 31, no. 6, pp. 1785–1799, 1994.
- [24] W. Hackbusch, *Multi-grid methods and applications*, vol. 4. Springer Science & Business Media, 2013.
- [25] M. F. Adams, “A distributed memory unstructured gauss-seidel algorithm for multigrid smoothers,” in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, pp. 4–4, ACM, 2001.