Contents

Chapter 3

Discretisation and Solution Methods

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 Variational Formulation

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

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

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

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

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

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

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

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

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

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

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

The variational formulation is therefore equivalent to the problem

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

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

3.1.2 Galerkin Approach

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

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

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

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

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

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

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

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

where u is the solution to (2.2) and u_h to the corresponding finite dimensional problem (2.3). Hence accuracy of our approximation depends in this case on the constants 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 furthemore want to assume that as h tends to zero this implies that $dim(V_h) \Rightarrow \infty$. Additionally let $\{V_h : h > 0\}$ denote a family of finite dimensional subspaces of V, for which we assume that

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

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

3.1.3 Matrix Formulation

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

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

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

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

3.2 Least Squares Finite Element Methods

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

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

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

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

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

We consider the problem and the least squares functional:

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

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

which poses the minimisation problem:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.3 Space-Time Approaches

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 $0 = t_0 < t_1 < .... < t_m = T$. 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

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

can be formulated as follows using two propagation operators:

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

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

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

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

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

In **space-time domain decomposition methods** the idea is to divide the domain 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:

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

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

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

and afterwards the overlapping homogeneous problem is solved:

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

Due to linearity the overall solution can be obtained through summation

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

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

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

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

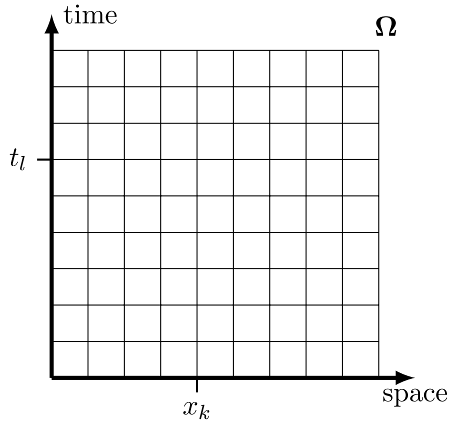
where each subset u_i contains all spatial elements for a particular time interval. In the multigrid iteration they apply a block Jacobi smoother inverting each of the blocks A_j before using a standard restriction operators in space-time to jump to a coarser grid, which is then repeated recursively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

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

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

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

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



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

3.4 Iterative Methods for Nonlinear Systems and Minimisation Problems

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

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

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

3.4.1 Gradient Descent Methods

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

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

If the scaling parameter $\alpha_k > 0$ is chosen sufficiently small, we know that $J(s_{k-1}) \geq J(s_k) \geq J(s_{k+1})$. If $\|\nabla J(s_k)\| = 0$ we have found a local minimum and hence $s_{k+1} = s_k$. There are a number of strategies that try to select a suitable value for 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 $\nabla J(s) = [0, 0, ..., 0]^T$, for which we would like to determine the unknown root s. We consider a Taylor series expansion for an initial guess s_0

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

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

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

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

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

3.4.3 Trust Region Methods

Trust region methods is a generic term that comprises globalisation strategies to approximate minima of potentially non-convex optimisation problems. The objective function J is approximated using a model function in a subset of the domain, the so-called trust region. If the model function seems to be a good local fit to the function, the size of the region is expanded, if it is not, the size of the region is reduced. The fit of the model is assessed by comparing the ratio 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 first set the ground for the overall framework we are looking at. We consider a space-time domain

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

where $\mathcal{T}$ represents the time domain and $\mathcal{S}$ the domain in space, which we require to be Lipschitz regular. We allow a mixture of Dirichlet and Neumann boundary conditions on $\partial\Omega$ which we denote 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:

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

It describes a parabolic partial differential equation with a potentially nonlinear right-hand side where the divergence is defined as

$$\operatorname{div}(\sigma) = \frac{\partial \sigma}{\partial x_1} + ... + \frac{\partial \sigma}{\partial x_N} \text{ for } \sigma = \sigma(x,t) \text{ and } \nabla u = [\frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_N}]^T \text{ for } u = u(x,t).$$

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 (4.2) or equivalence with respect to the solution space of the variational formulation, for a further discussion we refer to [7]. Generally when working with least squares finite element formulations choosing a suitable solution space U and data space Y 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 further considerations need to be taken into account to make the methodology of LSFEMs competitive compared to other approaches like Galerkin approximations. 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] *more here or in previous LSFEM section ...?*. 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 succomb this obstacle we will recast (4.2) as a system of mixed equations only containing first order derivatives to apply the methodologies introduced in section (3.4) at the price of introducing an additional variable. Hence let

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

Rearranging this equation into a vector form we obtain in Ω

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

which we will refer to as the mixed strong form of our problem and 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.

Remark. It is not clear yet in which space we will be looking for a solution. In general when working with strong and weak formulations there is a 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) or (4.3). 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, and later on show that they indeed exist.

So instead of looking for a solution of the strong formulation which is far more restrictive, 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 $J([\sigma, u], f) = 0$ then the orginial problem (4.3) also has to be satisfied if only in a weak sense with respect to the spaces U and Y. We can check that all three properties hold for the functional

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

Furthermore it has shown to be practical to be able to weight the two terms with coefficients [source] which does not affect the solution of the continuous problem but grants us the possibility to numerically give more importance to one term than the other, a further exploration of this will be in section 6.3.2. We also introduce additional scalars of $\frac{1}{2}$ to simplify further computations of the derivatives as we will see in the following section. Therefore the problem then reads

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

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

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

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, and Appendix A for further derivations.

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

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 from the space W.

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

Remark: $\mathcal{L}_{\tilde{f}}(u)([\tau, v])$ is a linear operator in v but nonlinear in u. What remains to be determined are the function spaces U, W, and Y.

4.2 Function Spaces

In order to define U and Y let us consider the optimisation problem (4.6) and its variational formulation (4.9) again. We would like to allow for the broadest class of solutions possible while still ensuring that all terms are well-defined in U, 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 Y. Let us therefore consider the following spaces

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

$$H_{\operatorname{div}}(\Omega) = H^1_{\operatorname{div}}(\mathcal{S}) \times L^2(\mathcal{T}) \quad (4.11)$$

where and when to use $\times$, above it means something different from below ... still both direct sums?

$$U = W = H^1_{\operatorname{div}}(\Omega) \times H^1(\Omega) \quad (4.12)$$

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

And thus we have that

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

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

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

$$\|(\sigma, u)\|^2_U = \|\sigma\|^2_{H_{\operatorname{div}}(\Omega)} + \|u\|^2_{H^1(\Omega)} \quad (4.17)$$

$$(4.18)$$

The direct sum of two Hilbert spaces is again a Hilbert space using the inner product induced by the sum of the respective inner products [23]. We have that $L^2(\mathcal{T})$, $H^1(\Omega)$ and $H_{\operatorname{div}}(\mathcal{S})$ are

Hilbert spaces [24], and hence we obtain that U, W, and Y are as well. 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 chapter 6.

$\mathcal{B}$ induces an inner product and subsequently a norm on U, if we assume homogenous boundary conditions on σ and u. We can check that it is a symmetric bilinear form. In addition we have that $\mathcal{B}([\sigma, u], [\sigma, u]) \geq 0$ for all $[\sigma, u] \in U$, since $\mathcal{B}([\sigma, u], [\sigma, u]) = J([\sigma, u], 0)$ is the sum of two squared L^2–norms. Therefore it only remains to show that

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

If $[\sigma, u] = 0$ we immediately obtain that $\mathcal{B}([\sigma, u], [\sigma, u]) = 0$. Hence only the reverse direction remains to be shown. So let us assume that there exists $0 \neq [\sigma, u] \in U$
to be filled

In order to fully set the theoretical framework that guarantees us all the favourable attributes of the beforementioned Rayleigh–Ritz setting we would like to fulfill the assumptions of *Theorem 1* from section 3.3, which require the usage of conforming discrete subspaces $U^h \subset U$ and the following norm equivalence for some $\alpha, \beta > 0$

$$\alpha \|[\sigma, u]\|_U^2 \leq J([\sigma, u], 0) \leq \beta \|[\sigma, u]\|_U^2 \quad \forall\ [\sigma, u] \in U. \tag{4.20}$$

In order to show the upper bound on J we use that $Y = L^2(\Omega)$, $D(x)$ is bounded, i.e. there exists $d_{\max} \in \mathbb{R}$ such that $\|D(x)\|_Y \leq d_{\max}$ for all $x \in \mathcal{S}$ and the parallelogram identity which holds in Hilbert spaces.

$$\begin{aligned}
J([\sigma, u], 0) &= \|u_t - \operatorname{div}(\sigma)\|_{L^2(\Omega)}^2 + \|\sigma - D(x)\nabla u\|_{L^2(\Omega)}^2 \\
&\leq \|u_t - \operatorname{div}(\sigma)\|_{L^2(\Omega)}^2 + \|u_t + \operatorname{div}(\sigma)\|_{L^2(\Omega)}^2 + \|\sigma - D(x)\nabla u\|_{L^2(\Omega)}^2 + \|\sigma + D(x)\nabla u\|_{L^2(\Omega)}^2 \\
&\leq 2\|u_t\|_{L^2(\Omega)}^2 + 2\|\operatorname{div}(\sigma)\|_{L^2(\Omega)}^2 + 2\|\sigma\|_{L^2(\Omega)}^2 + 2d_{\max}^2\|\nabla u\|_{L^2(\Omega)}^2 \\
&\leq 2\|u\|_{L^2(\Omega)}^2 + 2\|u_t\|_{L^2(\Omega)}^2 + 2d_{\max}^2\|\nabla u\|_{L^2(\Omega)}^2 + 2\|\operatorname{div}(\sigma)\|_{L^2(\Omega)}^2 + 2\|\sigma\|_{L^2(\Omega)}^2 \\
&\leq \max(2, 2d_{\max}^2)(\|u\|_{H^1(\Omega)}^2 + \|\sigma\|_{H_{\mathrm{div}}(\Omega)}^2) \\
&= \beta\|[\sigma, u]\|_U^2 \qquad \text{with } \beta := \max(2, 2d_{\max}^2).
\end{aligned} \tag{4.21}$$

The proof of the coercivity is not straight forward and potentially not even true. In the paper of Z. Cai et al. on "First-order system least squares for second-order partial differential equations: Part l", [25], they show norm equivalence for a similar class of problems that are however elliptic and therefore the terms and spaces involved differ. Nevertheless it might be possible to proceed similarly to their work to show that such an α exists, this is however beyond the scope of this thesis but could be an interesting extension to the topic.

A point that has not really been discussed 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 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 finite-dimensional subspaces U^h and W^h of U and W respectively, that are defined on the entire space-time domain, where U^h contains the solution space for σ^h and u^h, that is $s^h = [\sigma^h, u^h] \in U^h$. However the subspaces for the two do not have to be the same, and can be chosen independently of each other which can potentially be advantageous as the continuous spaces differ as well, due to their different properties. Hence let us assume that $\sigma^h \in \tilde{U}^h$ and $u^h \in \hat{U}^h$, where $U^h = \tilde{U}^h \times \hat{U}^h$.
So suppose we have $\tilde{n} = dim(\tilde{U}^h)$ and let $\{\tilde{\phi}_1, ..., \tilde{\phi}_{\tilde{n}}\}$ be a basis of $\tilde{U}^h$ and similarly $n = dim(\hat{U}^h)$ with $\text{span}\{\phi_1, ..., \phi_n\} = \hat{U}^h$. We furthermore assume U^h to be constructed such that $\inf_{s^h \in U^h} \|s - s^h\|_U \to 0$ as $h \to 0$ for all $s \in U$. *Is this a reasonable assumption, Raviart - Thomas only in space what for time?! Some reference? Direct sums so it works?.* 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^h and u^h as a linear combination of basis functions in $\tilde{U}^h$ or equivalently $\hat{U}^h$ that is

$$\sigma_h(x, t) = \sum_{i=1}^{\tilde{n}} \sigma_i \tilde{\phi}_i(x, t) \qquad u_h(x, t) = \sum_{i=1}^{n} u_i \phi_i(x, t) \tag{4.22}$$

The functional J then looks as follows

$$J(\sigma_h, u_h) = \| \sum_{i=1}^{n} u_i \ (\phi_i)_t - \sum_{i=1}^{\tilde{n}} \sigma_i \ \text{div}(\tilde{\phi}_i) - f(\sum_{i=1}^{n} u_i \phi_i) \|_Y^2 + \| \sum_{i=1}^{\tilde{n}} \sigma_i \tilde{\phi}_i - D(x) \nabla (\sum_{i=1}^{n} u_i \phi_i) \|_Y^2. \tag{4.23}$$

In the arising variational formulation 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 $\tilde{U}^h$ and U^h. The discretised weak form then reads

$$\text{Find } [\sigma_h, u_h] \in U_h \text{ such that } B \cdot [\sigma_h, u_h]^T = L_{\tilde{f}}(u_h) \tag{4.24}$$

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 the discretised right-hand side which we for now assume to contain all nonlinear terms, that is those related to f. Since we assume that the solution $s_h = [\sigma_h, u_h]$ first contains all values corresponding to σ_h and then for u_h we obtain a block structure for B and $L_{\tilde{f}}$ of the following form.

$$B = \begin{bmatrix} B_{\sigma\sigma} & B_{\sigma u} \\ B_{u\sigma} & B_{uu} \end{bmatrix} \qquad L_{\tilde{f}}(u_h) = \begin{bmatrix} (L_{\tilde{f}}(u_h))_\sigma \\ (L_{\tilde{f}}(u_h))_u \end{bmatrix} \tag{4.25}$$

Each entry of each of the blocks of B can be computed explicitly according to the subsequent schemes. For the detailed computation we refer to the next section and appendix A.

For $B_{\sigma\sigma}$: $$B_{ij} = \langle \tilde{\phi}_j, \tilde{\phi}_i \rangle_Y + \langle \text{div}(\tilde{\phi}_j), \text{div}(\tilde{\phi}_i) \rangle_Y \qquad \forall i, j \in \{1, ..., \tilde{n}\} \tag{4.26}$$

For $B_{\sigma u}$: $$B_{ij} = -\langle D(x) \nabla \phi_j, \tilde{\phi}_i \rangle_Y - \langle (\phi_j)_t, \text{div}(\tilde{\phi}_i) \rangle_Y \qquad \forall i \in \{1, ..., \tilde{n}\}, j \in \{\tilde{n}+1, ..., m\} \tag{4.27}$$

For $B_{u\sigma}$: $$B_{ij} = -\langle D(x) \nabla \phi_i, \tilde{\phi}_j \rangle_Y - \langle (\phi_i)_t, \text{div}(\tilde{\phi}_j) \rangle_Y \qquad i \in \{\tilde{n}+1, ..., m\}, \forall j \in \{1, ..., \tilde{n}\} \tag{4.28}$$

For B_{uu} : $$B_{ij} = \langle D(x) \nabla \phi_j, D(x) \nabla \phi_i \rangle_Y + \langle (\phi_j)_t, (\phi_i)_t \rangle_Y \qquad \forall i, j \in \{\tilde{n}+1, ..., m\} \tag{4.29}$$

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 zero and therefore the right-hand side, and which we will denote by $L_{\tilde{f}} = L_{\tilde{f}}(u_h)$ to underline its independence of u. Thus it only contains the following terms

$$(L_{\tilde{f}})_i = 0 \qquad i \in \{1, ..., \tilde{n}\} \tag{4.30}$$

$$(L_{\tilde{f}})_i = \sum_{j=1}^{\tilde{n}} \langle \mathrm{div}(\tilde{\phi_j}), f\rangle_Y - \langle (\phi_i)_t, f\rangle_Y \qquad i \in \{\tilde{n}+1, ..., m\} \tag{4.31}$$

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

$$B \begin{pmatrix} \sigma \\ u \end{pmatrix} = L_{\tilde{f}} \tag{4.32}$$

which can then be solved immediately for $[\sigma, u]$ 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. If $f(u)$ is a linear function we can directly include the additional terms in B. Otherwise we require a nonlinear iteration scheme which considers linearisations of the problem and whose construction will be the topic of the remaining part of the chapter.

4.4 Gradient and Hessian of the Objective

Let us compute the gradient and the Hessian of the functional J, which are needed for the construction of B and the nonlinear iteration scheme. We therefore determine its first and second order Gateaux derivatives. 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 $\langle\cdot,\cdot\rangle = \langle\cdot,\cdot\rangle_Y$ in order to simplify the notation. Let us first assume $f = 0$ to derive the matrix B. That is we have

$$\begin{aligned} &J([\sigma,u],0) = J_1([\sigma,u]) + J_2([\sigma,u]) \quad \text{where} \\ &J_1(\sigma,u) := \frac{1}{2} c_1 \langle u_t - \mathrm{div}(\sigma), u_t - \mathrm{div}(\sigma)\rangle \\ &J_2(\sigma,u) := \frac{1}{2} c_2 \langle \sigma - D(x)\nabla u, \sigma - D(x)\nabla u\rangle \end{aligned} \tag{4.33}$$

Since $u \in H^1(\Omega)$ and $\sigma \in H_{\mathrm{div}}(\Omega)$ they are not defined pointwise but we can nevertheless take the subsequent limits [*something more here, reference?!*], but then also only hold almost everywhere. We then obtain the following directional partial derivative for J_1 using the linearity and the symmetry of the inner product.

$$\begin{aligned} \frac{\partial J_1}{\partial \sigma} &= \lim_{\epsilon\to 0} \frac{J_1(\sigma+\epsilon\tau, u) - J_1(\sigma,u)}{\epsilon} \\ &= \lim_{\epsilon\to 0} \frac{c_1}{2\epsilon}(\langle u_t - \mathrm{div}(\sigma+\epsilon\tau), u_t - \mathrm{div}(\sigma+\epsilon\tau)\rangle - \langle u_t - \mathrm{div}(\sigma), u_t - \mathrm{div}(\sigma)\rangle) \\ &= \lim_{\epsilon\to 0} \frac{c_1}{2\epsilon}(\langle u_t, u_t\rangle - \langle u_t, \mathrm{div}(\sigma)\rangle - \epsilon\langle u_t, \mathrm{div}(\tau)\rangle - \langle \mathrm{div}(\sigma), u_t\rangle + \langle \mathrm{div}(\sigma), \mathrm{div}(\sigma)\rangle \\ &+ \epsilon\langle \mathrm{div}(\sigma), \mathrm{div}(\tau)\rangle - \epsilon\langle \mathrm{div}(\tau), u_t\rangle + \epsilon\langle \mathrm{div}(\tau), \mathrm{div}(\sigma)\rangle + \epsilon^2\langle \mathrm{div}(\tau), \mathrm{div}(\tau)\rangle \\ &- \langle u_t, u_t\rangle + \langle u_t, \mathrm{div}(\sigma)\rangle + \langle \mathrm{div}(\sigma), u_t\rangle - \langle \mathrm{div}(\sigma), \mathrm{div}(\sigma)\rangle) \\ &= \lim_{\epsilon\to 0} \frac{c_1}{2\epsilon}(-2\epsilon\langle u_t, \mathrm{div}(\tau)\rangle + 2\epsilon\langle \mathrm{div}(\sigma), \mathrm{div}(\tau)\rangle + \epsilon^2\langle \mathrm{div}(\tau), \mathrm{div}(\tau)\rangle) \\ &= -c_1\langle u_t, \mathrm{div}(\tau)\rangle + c_1\langle \mathrm{div}(\sigma), \mathrm{div}(\tau)\rangle \end{aligned} \tag{4.34}$$

By proceeding analogously for equation J_2 and the partial directional derivatives for u we end up with

$$\frac{\partial J_2}{\partial \sigma} = c_2\langle \sigma, \tau\rangle - c_2\beta\langle \tau, \nabla u\rangle \tag{4.35}$$

$$\frac{\partial J_1}{\partial u} = c_1\langle u_t, v_t\rangle - c_1\langle v_t, \mathrm{div}(\sigma)\rangle \tag{4.36}$$

$$\frac{\partial J_2}{\partial u} = -c_2\langle \sigma, D(x)\nabla v\rangle + c_2\langle D(x)\nabla u, D(x)\nabla v\rangle \tag{4.37}$$

Hence we obtain the following partial first order directional derivatives.

$$D_\sigma J[\tau] = \frac{\partial}{\partial \sigma}J([\sigma,u])[\tau] = c_2\langle \sigma, \tau\rangle + c_1\langle \mathrm{div}(\sigma), \mathrm{div}(\tau)\rangle - c_2\langle D(x)\nabla u, \tau\rangle - c_1\langle u_t, \mathrm{div}(\tau)\rangle \tag{4.38}$$

$$D_u J[v] = \frac{\partial}{\partial u}J([\sigma,u])[v] = c_1\langle u_t, v_t\rangle - c_1\langle v_t, \mathrm{div}(\sigma)\rangle - c_2\langle \sigma, D(x)\nabla v\rangle + c_2\langle D(x)\nabla u, D(x)\nabla v\rangle \tag{4.39}$$

Following the same principles one can determine the second order partial derivatives.

$$\begin{aligned}
\frac{\partial^2}{\partial \sigma^2}J[\tau][\rho] &= c_2\langle \rho, \tau\rangle + c_1\langle \mathrm{div}(\rho), \mathrm{div}(\tau)\rangle \\
\frac{\partial^2}{\partial \sigma \partial u}[v][\tau] = \frac{\partial^2}{\partial u \partial \sigma}[\tau][v] &= -c_2\langle \tau, D(x)\nabla v\rangle - c_1\langle v_t, \mathrm{div}(\tau)\rangle \\
\frac{\partial^2 J}{\partial u^2}[v][w] &= c_1\langle w_t, v_t\rangle + c_2\langle D(x)\nabla w, D(x)\nabla v\rangle
\end{aligned} \tag{4.40}$$

If we now recast this in the finite dimensional setting using the beforementioned basis functions instead of the τ, ρ, v and w, and we can compute the matrix B *proper name, just call it H for Hessian?* which corresponds exactly to the description given by (4.26)–(4.29).

For the terms involving f we can proceed in a similar way, given that f is sufficiently smooth. The exact assumptions and the corresponding computations can be found in appendix A. This entails first the construction of an entire continuous iteration scheme and then its discretisation. One finally ends up with the following first and second order partial directional derivatives for a $f = f(u)$.

$$D_\sigma J[\tau] = c_2\langle \sigma, \tau\rangle + c_1\langle div(\sigma), div(\tau)\rangle - c_2\langle D(x)\nabla u, \tau\rangle - c_1\langle u_t, div(\tau)\rangle - c_1\langle f(u), div(\tau)\rangle \tag{4.41}$$

$$D_u J[v] = c_1\langle u_t, v_t\rangle - c_1\langle v_t, div(\sigma)\rangle - c_2\langle \sigma, D(x)\nabla v\rangle + c_2\langle D(x)\nabla u, D(x)\nabla v\rangle \tag{4.42}$$

$$- c_1\langle u_t, f'(u)\cdot v\rangle - c_1\langle v_t, f(u)\rangle - c_1\langle div(\sigma), f'(u)\cdot v\rangle + c_1\langle f(u), f'(u)\cdot v\rangle, \tag{4.43}$$

$$\tag{4.44}$$

$$D_{\sigma\sigma} J[\tau][\rho] = c_2\langle \rho, \tau\rangle + c_1\langle \mathrm{div}(\rho), \mathrm{div}(\tau)\rangle \tag{4.45}$$

$$D_{\sigma u} J[v][\tau] = D_{u\sigma} J[\tau][v] = -\langle \tau, D(x)\nabla v\rangle - \langle v_t, div(\tau)\rangle - \langle div(\tau), f'(u)v\rangle \tag{4.46}$$

$$D_{uu} J[v][w] = c_1\langle w_t, v_t\rangle + c_2\langle D(x)\nabla w, D(x)\nabla v\rangle + c_1\langle u_t, w^T f''(u)v\rangle - c_1\langle w_t, f'(u)\cdot v\rangle \tag{4.47}$$

$$- c_1\langle v_t, f'(u)\cdot w\rangle - c_1\langle div(\sigma), w^T f''(u)v\rangle + c_1\langle f(u), w^T f''(u)v\rangle + \langle f'(u)\cdot w, f'(u)\cdot v\rangle \tag{4.48}$$

Another way to formulate the problem is to first restrict the functional to finite dimensional subspaces and then differentiate which is the way we implemented it. For a further discussion of the pros and cons of either approach we can refer to [*find good source?!*]. Hence the terms involvoing f are constructed slightly differently, and will be introduced in the next section, as we derive the iterative schemes.

4.5 Nonlinear Iteration Scheme

As discussed in 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 $J(s_0) \geq J(s_1) \geq ... \geq J(s_k) \geq ...$ 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 in the steepest descent direction. Hence we only need to evaluate the gradient in this case, which is computationally much cheaper but usually leads to a very slow convergence rate.

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 assume 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 $\hat{U}^h$, that is

$$f_h = \sum_{i=1}^{n} f_i \phi_i, \qquad f'_h = \sum_{i=1}^{n} f'_i \phi_i, \qquad f''_h = \sum_{i=1}^{n} f''_i \phi_i \tag{4.49}$$

Therefore f_h, f'_h and f''_h now represent a finite dimensional approximation of f for one particular approximation u_h. 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 $\nabla J = 0$, where

$$\nabla J_k = \nabla J(s_k) = B \begin{pmatrix} \sigma_k \\ u_k \end{pmatrix} - L_{\tilde{f}} \quad \text{with} \tag{4.50}$$

$$(L_{\tilde{f}})_\sigma = \langle \text{div}(\sigma_k), (f'_h) \rangle_Y \tag{4.51}$$

$$(L_{\tilde{f}}(u_k))_u = -\langle u_t, f'_h \rangle_Y - \langle v_t, f_h \rangle_Y + \langle f_h, f'_h \rangle_Y \tag{4.52}$$

I know this is not exactly right yet, need to write it up again exactly, for individual indices. (4.53)

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

$$s_{k+1} = s_k + \alpha(-\nabla J_k) \tag{4.54}$$

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

$$H_k = \nabla^2 J(s_k) = B_{\text{lin}} + Q \tag{4.55}$$

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

$$Q = \begin{bmatrix} 0 & Q_{\sigma u} \\ Q_{u\sigma} & Q_{uu} \end{bmatrix}, \text{ where} \tag{4.56}$$

$$\tag{4.57}$$

$$(Q_{\sigma u})_{ij} = (Q_{u\sigma})_{ji} = -\langle \text{div}(\tilde{\phi}_j), (f'_h)_i \rangle \tag{4.58}$$

$$(Q_{uu})_{ij} = c_1 \langle u_t, w^T f''(u) v \rangle - c_1 \langle w_t, f'(u) \cdot v \rangle \tag{4.59}$$

$$-c_1 \langle v_t, f'(u) \cdot w \rangle - c_1 \langle div(\sigma), w^T f''(u) v \rangle + c_1 \langle f(u), w^T f''(u) v \rangle + \langle f'(u) \cdot w, f'(u) \cdot v \rangle \tag{4.60}$$

again, still need to rewrite this, this time in terms of the discretised everything with indices (4.61)

And this leads to the iteration

$$s_{k+1} = s_k - H_k^{-1}(\nabla J_k) \tag{4.62}$$

of which we would like to solve the linear system of equations

$$e_{k+1} = -H_k^{-1}(\nabla J_k) \quad \text{with} \quad s_{k+1} = s_k + e_{k+1} \tag{4.63}$$

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 (4.63) to solve this linear system of equations in each iteration. The following chapter will provide an insight into the underlying principles of the multigrid methodology before we turn to its actual implementation.