0.1 Newton's Method

[source Briggs] One of the most well-known and most commonly used method to solve a non-linear problem is Newton's method which we will also be employed here and which we will derive in the following section and briefly discuss its advantages and disadvantages. So suppose we have an equation $F : \Omega \to \mathbb{R}^n$, with $F(x) = [0, 0, ..., 0]^T$, $x \in \mathbb{R}^n$, and $F \in C^1$, where we would like to determine the unknown root x. We consider a Taylor series expansion for an initial guess x:

$$F(x + h) = F(x) + \nabla F(x) \cdot h + o(||h||^2) \quad x \in \Omega. \tag{1}$$

If we now neglect the higher order terms, setting $F(x+h) = 0$ and replacing it by its first order Taylor approximation $F(x) + \nabla F(x) \cdot h$, which we can then solve for h, assuming that $\nabla F(x)$ is non-singular and use the result to update our initial guess x we obtain

$$x = x - [\nabla F(x)]^{-1} F(x). \tag{2}$$

(maybe better to write as an iteration)

0.1.1 Convergence

The convergence of Newton's method under certain conditions in the one dimensional case is established and proven relatively easily, however it requires much more work to establish the existence of and convergence to a root in the multidimensional case. There are different ways to do so, requiring (slightly) varying assumptions and hence obtaining (slighlty) different results. Here we will restrict ourselves to a classical version which was first shown by L. Kantorovich in 1940 and states the following:

Theorem 1. *Newton-Kantorovich.*
State here? Not too extensive?

While this theoretically establishes conditions for convergence, it is in practice ususally impossible to know if the necessary criteria are met. What we can however say, is that the likelyhood of converging at all and maybe even to the desired solution usually increases drastically if our initial guess is relatively close to the solution, assuming that "close" is defined in a meaningful way. Therefore one often solves a simplified related problem, commonly some sort of linearisation of the original problem which hopefully admits a similar solution that can then serve as a first initial guess. We will see in the following sections that the same idea will be applied here.

Or better something from Deuflhard here?!

— short transition section — So in each Newton step we consider a linearisation of our problem over the entire space - time domain, that we would like to be able to tackle at once and not solve for each time step consecutively. The following section gives an insight of how this can be done and explain why this might be a good idea —–

0.2 Space-Time Solvers

Usually the time direction in partial differential equations is not used 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. However evolution over time behaves differently from the spatial dimensions, in the sense that it follows the causality principle. Meaning 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 [**?**], 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 into time slabs Ω_j, that is if $\Omega = \mathcal{S} \times [0, T]$ 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 an approximated solution y_j for all $x \in \mathcal{S}$ at t_j, with $y_j = y(x, t_j)$, where y denotes solution (careful with approximation and solution...) 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

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

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

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

Convergence? Extra work? General?

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.

Direct Solvers in Space-Time. Here varying techniques are emplyed, one example is a method introduced in 2012 by S. Güttel called ParaExp, 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:

$$y'(t) = Ay(t) + g(t), \quad t \in [0, T], \quad u(0) = u_0 \tag{3}$$

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 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:

$$v_j'(t) = Av_j(t) + g(t), \quad v_j(T_{j-1}) = 0 \quad t \in [T_{j-1}, T_j] \tag{4}$$

and afterwards the overlapping homogeneous problem is solved:

$$w_j'(t) = Aw_j(t), \quad w_j(T_{j-1}) = v_{j-1}(T_{j-1}), \quad t \in [T_{j-1}, T] \tag{5}$$

Due to linearity the overall solution can be obtained through summation

$$y(t) = v_k(t) + \sum_{j=1}^{k} w_j(t) \quad \text{with } k \text{ s.t. } t \in [T_{k-1}, T_k] \tag{6}$$

This way we obtain the general solution over the whole time interval. At first it is not clear 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 [[]].

In **space-time multigrid methods**, the parallelity comes from the discretisation of the space-time domain, that is considered as one, as mentioned before in the section on [FEM discretisation]. As a rather recent example of this type we will consider an approach by M. Gander and M. Neumüller [**?**]. 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

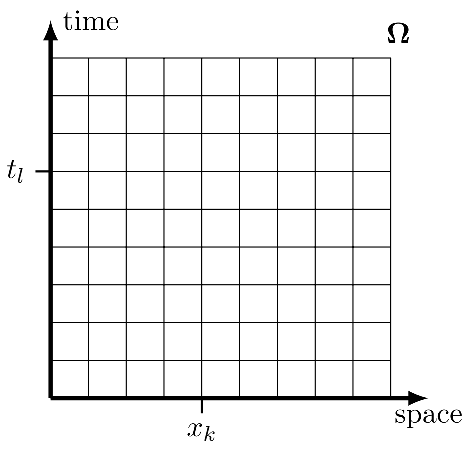
$$\begin{bmatrix} A_1 & & & & \\ B_2 & A_2 & & & \\ & B_3 & A_3 & & \\ & & \cdots & \cdots & \\ & & & B_m A_m & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ u_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdots \\ f_m \end{bmatrix} \tag{7}$$

For the multigrid iteration they apply a block Jacobi smoother inverting each of the space blocks A_j before using a standard restriction operators in space - time to jump to a coarser grid, which is then being applied recursively. Choosing the right parameters one can achieve excellent scaling results for large systems.

Some solution approaches in space - time can be categorized in multiple approaches, for example is a 2 - level multigrid method starting with an initial guess obtained from the coarse grid and using an upwind (?) smoother the same as a simple parareal appraoch []

In this thesis we will consider a space-time multigrid approach but not of the previous type for the beforementioned symmetry reason (see section [...]) but instead go for a more straight forward space-time finite element assembly in addition to a first order least squares formulation that will be introduced in the following sections. For now we will only have a more detailed look at the space-time formulation that differs from a common finite element approach in the sense that our basis functions $\{\phi_1, ..., \phi_z\}$ which were introduced in section [...] are functions of time and space, i.e. $\phi_i = \phi_i(x, t)$. 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 (2.15) in the sense that there are symmetric upper and lower off-diagonal blocks.

future influencing the past? upwind scheme?



The discretisation of the domain can be visualised as shown in figure [...].
Where one has $n+1$ points in space and $m+1$ points in time. The points (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 run through 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 multigrid section. For simplicity we will only consider equidistant grids in each direction, where h_t denotes the size (better word?) of a mesh cell in time direction and h_{x_i} denote the size in the respective space dimension.

——— short transition ——— different ways to discretise the eqn on the mesh ——— one class of possibilities FEM ——

0.3 Finite Element Methods

In order to find a numerical estimate to the solution of our 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 use here is a finite element approach. They have shown to be a very powerful ... and are based on ... [source ?]

0.3.1 General Set Up

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 used throughout this thesis, a recollection of the most important properties needed. 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 [...]. Let us for now assume that we have constructed a proper variational formulation of the following type:

Find u in V such that: $a(u, v) = L_f(v) \quad \forall v \in V$ (8)

where V is a Hilbert space, $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ is a symmetric, bounded and coercive bilinear form thus inducing an innerproduct on V. That is more specifically:

$a(v_1, v_2) = a(v_2, v_1)$ 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 $v_1, v_2 \in V$ (coercivity) and

$a(v_1, v_1) \geq \alpha ||v_1||_V^2$ for all $v_1 \in V$ (boundedness)

$L_f : V \to \mathbb{R}$ is a bounded linear operator on V, that is $f \in V^*$, the dual space of V. And for simplicity we also assume that we 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).
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

$$\text{Find } u_h \text{ in } V_h \text{ such that: } a(u_h, v_h) = L_f(v_h) \quad \forall v_h \in V_h \tag{9}$$

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

$$\begin{aligned} ||u - u_h||_a^2 &= a(u - u_h, u - u_h) = a(u - u_h, u) + a(u - u_h, v_h) \\ &\leq ||u - u_h||_a \cdot ||u - v_h||_a \quad \forall v_h \in V_h. \end{aligned} \tag{10}$$

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

$$\begin{aligned} \alpha ||u - u_h||_V^2 &\leq a(u - u_h, u - u_h) = a(u - u_h, u - u_h) = a(u - u_h, u + v_h - v_h - u_h) \\ &= a(u - u_h, u - v_h) + a(u - u_h, v_h - u_h) = a(u - u_h, u - v_h) \\ &\leq \beta ||u - u_h||_V \cdot ||u - v_h||_V \quad \forall v_h \in V_h. \end{aligned} \tag{11}$$

Dividing by $\alpha ||u - u_h||$ we have shown *Céa's lemma*, which states that:

$$||u - u_h||_V \leq \inf_{v_h \in V_h} \frac{\beta}{\alpha} ||u - v_h||_V, \quad u \in V, u_h \in V_h \tag{12}$$

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

$$\forall v \in V : \quad \inf_{v_h \in V_h} ||v - v_h||_V \to 0 \text{ as } h \to 0. \tag{13}$$

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.

0.3.2 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

$$a(\sum_{j=1}^{n} u_j \phi_j, \phi_i) = \sum_{j=1}^{n} u_j a(\phi_j, \phi_i) = L_f(\phi_i) \quad \forall \phi_i,\ i = 1, 2, ..., n \tag{14}$$

If we now write this as a system of the form $A_h U_h = L_h$ with entries entries $(A_h)_{ij} = a(\phi_j, \phi_i)$, $(L_h)_i = L_f(\phi_i)$, enough for basis Therefore 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 $a_{ij} = a_{ji}$.

Already be more concrete here or wait until least squares section? — If one considers $V = H_0^1(\Omega)$, subspaces of polynomials, requirements on f ...

——- potentially short transition back to Newton —– Now this was considering a linear problem where f is independent of u but we will see later that these linear problems correspond to individual iterates of the Newton method.Hence if $f = f_u$ for a current iterate of u we know that each linearised problem has a unique solution. Therefore if the Newton iteration converges ... ?