Universität Hamburg

Mathematik Diplomarbeit

Principles of Optimal Residual Formulations and Optimal Residual Finite Element Methods for Solving Partial Differential Equations

Dipl. Phys., cand. Math. Eike Michael Scholz
eikescholz@gmx.de
Department Mathematik, University of Hamburg
Matrikel-Nr. 5386245

February 20, 2014

Erstgutachter: Prof Dr. Jens Struckmeier (Universität Hamburg)
Zweitgutachter: Prof. Dr. Reiner Lauterbach (Universität Hamburg)

To My Wonderful Wife Hanyu

Versicherung an Eides statt

Ich versichere an Eides statt, das ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe.

Eike (Michael) Scholz, Wuppertal den 17.12.2012

Contents

Nomenclature

$(v_i)_{i \in I} \subset V$ — A sequence (maybe finite), i.e. a collection of elements $v_i \in V$ for all $i \in I$. The sequence is said to be complete with resp. to V if

$$\overline{span(\{v_i | i \in I\})} = V \ .$$

1_V — The indicator function, i.e.

$$1_V(x) := \begin{cases} 1 & \text{if } x \in V \\ 0 & \text{otherwise} \end{cases} \ .$$

$< \cdot, \cdot >$ — The inner product of a Hilbert space.

$\#M$ — The cardinal number of the set M, i.e. the number of elements in M

δ_{ij} — The Kronecker delta, i.e.

$$\delta_{ij} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{0.1}$$

$\text{exker}(\mathfrak{F})$ — The extended kernel of the mapping $\mathfrak{F}$, see definition 3.1.5

$\frac{\partial}{\partial n} f$ — The directional directive of $f : \Omega \to V$ along the direction of the vector field n, where n is orthogonal to $\partial\Omega$ on $\partial\Omega$

$\frac{\partial}{\partial z_u}$ — Differential geometric partial derivative operator regarding component u of coordinate system z

$\ker(\mathfrak{F})$ — The kernel of the mapping $\mathfrak{F}$, see definition 3.1.4

$\mathbb{N}$ — The C-programmers natural numbers, i.e. $\{0, 1, 2, 3, \dots\}$

$\overline{M}$ — The topological closure of the set M

$\overset{\circ}{V}$ — The topological interior of the set V

$\overset{(w)}{\text{exker}}(\mathfrak{F})$ — The extended kernel of $\mathfrak{F}$, respectively the weak extended kernel of $\mathfrak{F}$

$\overset{(w)}{\lim_{n\to\infty}} v_n = b$	The sequence $(v_n)_{n\in\mathbb{N}}$ converges to b, respectively it converges weakly to b
∂M	The topological boundary of the set M
∂^k	A general partial derivation operator, see equation 3.3
∂_i	A partial derivation into the direction of component i
$\mathbb{R}^+$	The set of non negative real numbers, i.e. $\mathbb{R}^+ := \{r \in \mathbb{R} \mid r \geq 0\}$
$\mathbb{R}^{N\times M}$	The set of $N \times M$ matrices with real entries
$\lvert k\rvert$	The order of a multiindex k, see definition 3.1.1
$\lVert\cdot\rVert_B$	Specifically the norm of the Banach space B
$\lVert\cdot\rVert$	The norm of a Banach space. $\lVert x\rVert = \sqrt{< x, x >}$ if x is an element of a Hilbert space
$\lVert\cdot\rVert_\partial$	The norm on the boundary space, i.e. the norm on $codom(id_\partial)$ for some solution space, see definition 3.1.12 and 3.1.13
$\mathrm{ac}(j)$	The the set of affected cells of degree of freedom j of some finite element space, see definition 3.9.4
$\mathrm{eff}(i)$	The set of effective degrees of freedom of the $i-th$ degree of freedom of some finite element space, see definition 3.9.1
$\mathrm{ineff}(i)$	The set of ineffective degrees of freedom of the $i-th$ degree of freedom of some finite element space, see definition 3.9.2
$\underline{N}$	The set $\{0, 1, \ldots, N-1\}$ of $N \in \mathbb{N}$ elements.
$\vec{\cdot}$	Explicit decoration of vectors in $\mathbb{R}^3$, $\vec{a}, \vec{b}$, ... etc. are by notation elements of $\mathbb{R}^3$
$\vec{e}_0, \vec{e}_1, \vec{e}_2$	The canonical base of $\mathbb{R}^3$
$\wedge$	Depending on the context either the exterior product or the logical operator "and"
$\overset{w}{\mathrm{exker}}(\mathfrak{F})$	The weakly extended kernel of the mapping $\mathfrak{F}$, see definition 3.1.6
$\overset{w}{\lim_{n\to\infty}} v_n$	The weak limes of v_n, i.e. if V is a Banach space, V^* is its dual space and $(v_n)_{n\in\mathbb{N}} \subset V$ then $\overset{w}{\lim_{n\to\infty}} v_n = v_\infty \Leftrightarrow \forall v' \in V^* : \lim_{n\to\infty} v'(v_n) = v'(v_\infty)$

$a * b$	The plain multiplication of real, natural, rational, etc. Numbers, used for readability
$a \cdot b$	The scalar product of $a, b \in \mathbb{R}^3$
$a \times b$	If a, b are sets the Cartesian product, if $a, b \in \mathbb{R}^3$ the cross product
$C^\infty(\Omega)$	The set of all infinitely differentiable functions from Ω to $\mathbb{R}$
$codom(\mathfrak{F})$	The codomain of the mapping $\mathfrak{F}$
d	The exterior derivative
$d(v, S)$	The distance between a element v in a Banach space B to a Subset $S \subset B$ in the following sense $$d(v, S) := \inf\{\|v - s\| \mid s \in S\}$$
$d(v, w)$	The distance between two vectors v, w in a metric space. If the space is a Banach space B, then $d(v, w) := \|v - w\|$
$dom(\mathfrak{F})$	The domain of the mapping $\mathfrak{F}$
e_i	The i-th unit vector of the canonical base of a vector space where e_0 is the first vector
$f \overset{B}{=} g$	The equality regarding the norm in the Banach space B, i.e. $$f \overset{B}{=} g \quad \Leftrightarrow \quad \|f - g\|_B = 0$$
$f \circ g$	The function composition, i.e. $f \circ g := x \mapsto f(g(x))$
$f\|_B$	The restriction of the function f to the set B
id_∂	The mapping that restricts functions $f : \Omega \to \mathbb{R}$ to $\partial\Omega$, also known as the trace operator, see definition 3.1.12
$j \nparallel k$	The degrees of freedom j and k are not independent, see definition 3.9.6
$j \parallel k$	The degrees of freedom j and k are independent, see definition 3.9.6
$L^2(\Omega)$	The Hilbert space of all square integrable functions from Ω to $\mathbb{R}$
$span(M)$	The subspace generated by M
v^T	The transposed of v, where v is either a vector or a matrix

1. Overview

> Its modest aim is to elaborate the point, that informal quasi-empirical, mathematics [...] [grows] through the incessant improvement of guesses by speculation and criticism, by the logic of proofs and refutations.
>
> *Imre Lakatos (about and in [15] p.5)*

1.1. Motivation

From a practical point of view solving a nonlinear partial differential equation numerically usually reduces to a problem of finding roots of some nonlinear function. I.e. find x with

$$F(x) = 0 \ . \tag{1.1}$$

It is a common practice to use a variation of Newtons method to solve the above problem:

$$x_{n+1} = x_n - ((dF)(x_n))^{-1} F(x_n) \tag{1.2}$$

The primary numerical problem that commonly arises when this equation is used, is that $(dF)(x_n)$ can not be guaranteed to be well conditioned. In some cases $(dF)(x_n)$ might not be regular or it might not even exist. It is therefore natural to ask if there is a more direct way to find a solution to equation 1.1 that does not necessarily involve solving a linear equation in each step.

This work will discuss approaches around the reformulation of equation 1.1 into a global optimization problem:

$$F(x) = 0 \tag{1.3}$$

$$\Leftrightarrow ||F(x)|| = 0 \tag{1.4}$$

$$\Leftrightarrow ||F(x)|| \stackrel{!}{=} min \tag{1.5}$$

Even though this reformulation itself is not complicated, I was not able to find literature on using this approach specifically to solve nonlinear partial differential equations. There is, of course, a lot of literature about solving $I(x) \stackrel{!}{=} min$ in very general cases. The whole field of functional analysis "revolves" around

such problems. However, I did not find anything examining the implications of choosing $I(x) = \|F(x)\|$ where $F(x)$ is a representation of a partial differential equation (pde). The formulation 1.5 is of course general and not restricted to finite dimensional cases.

The reason for my inability to find literature might be that a reformulation as the one above does make the initial problem even more complex. Thus it should be expected that algorithms based on this kind of reformulation are slow. However optimization algorithms do not require the "Jacobian matrix" dF to be regular, and thus the above approach could very well be much more robust, than approaches using variations of Newtons method.

Beside the aspect of robustness, I supposed that formulation 1.5 allows to construct algorithms that are almost embarrassingly parallel in each step, which could, using modern parallel computing hardware, compensate for the intrinsic slowness of optimization algorithms with respect to solving pdes. I did not find any specific literature and were curious, which lead to this work. The algorithms, that I supposed to exist, do indeed exist and are referred to as optimal residual finite element methods later in this work. Further I will in the following pages try to answer or at least cast some light on the following questions:

1. How can the above approach be rigorously formalized for solving nonlinear partial differential equations?

2. What properties must a partial differential equation (pde) at least have, to justify the expectation that the solution to an optimal residual formulation, like equation 1.5, approximates the solution of the pde?

3. How can an explicit algorithm be constructed using a finite element approach?

4. Since it is not strictly necessary to solve a linear equation system in 1.5, does this really increase the robustness of the algorithm?

1.2. Acknowledgments

As it is almost always the case, publications are not possible without the help of other people. First of all I have to thank my parents for making this work financially possible and providing a lot of support regarding almost everything. Next I have to thank Prof. Dr. M. Clemens from the University of Wuppertal for making this work possible by providing a workplace to write this work and computer hardware to test the algorithms. I have to thank Prof Dr. J. Struckmeier form the University of Hamburg for making it possible to Work on this topic as my Mathematics diploma thesis and for his open minded motivations. Special thanks go to Prof. Dr. R. Lauterbach from the University of Hamburg, to whom I specifically dedicate the dictum of this chapter, for his discussion with me of my analytic approaches. Without his help there would be some minor errors and at least one

mayor error in this work, as well as a lot of hints, regarding possibly problematic constructions, less. Further some hints from Prof. Dr. Tibken from the University of Wuppertal regarding multivariate polynomials and varieties helped me to refocus on important parts of this work, for that I want to thank him to. Regarding to my Implementation efforts I have to thank M.Sc. M. Saviz, who has been a visiting scientist at the University of Wuppertal, during the time this work was created, for his notes and paper collation about numerical quadratures. I have, as well, to thank the whole staff from Prof. Dr. M. Clemens for their patience and support, because my reluctance to switch from a half-time job into a full time job before I finished this work caused several delays.

Almost at last I have to thank my wonderful wife M.Sc H. Ye very much for her patience and support. She never lost her patience with me, even when we where in China to celebrate our marriage with her relatives and friends, and I used almost every spare hour to debug the test implementation of the algorithm explained at the end of the work.

At last I want to thank every one else who helped and supported me and whom I did not explicitly mention.

2. Introduction - Galerkin Methods in Optimal Residual Formulation

> Few things are harder to put up with than a good example.
>
> *Mark Twain*

Instead of starting directly with some non obvious definitions that will later be required for the full formalism, I start by informally showing, that the optimal residual approach of equation 1.5 for linear pdes is tightly connected to Galerkin methods. This introduction should therefore informally clarify the way in that solving a linear pde reduces to solving some n-dimensional affine linear equation $F(x) = 0$. We start by defining partial differential equations (almost) identical to Evans[1] p. 1.

Definition 2.0.1: Partial Differential Equation (pde)
An expression

$$F(\partial^k u(x), \partial^{k-1} u(x), ..., \partial u(x), u(x), x) = 0 \ \ (\forall x \in U) \tag{2.1}$$

is called a $|k|$-th order partial differential equation, where

$$F : \mathbb{R} \times ... \times \mathbb{R} \to \mathbb{R} \tag{2.2}$$

is given and $u : U \to \mathbb{R}$ is called the unknown. U is an open subset of $\mathbb{R}^n$. F is called the structural function of the pde. Further the k should be interpreted as a multiindex (this will be defined later).

The key to working out the connection between optimal residual formulations and Galerkin methods is to interpret the later as a result of finding a kind of optimal numerical approximation of the pde's solution.

2.1. Optimal Approximations of Functions

One central scheme in approximation is Hilbert space based. Its basic idea can nicely be illustrated in the three dimensional geometrical case. Assume, as shown

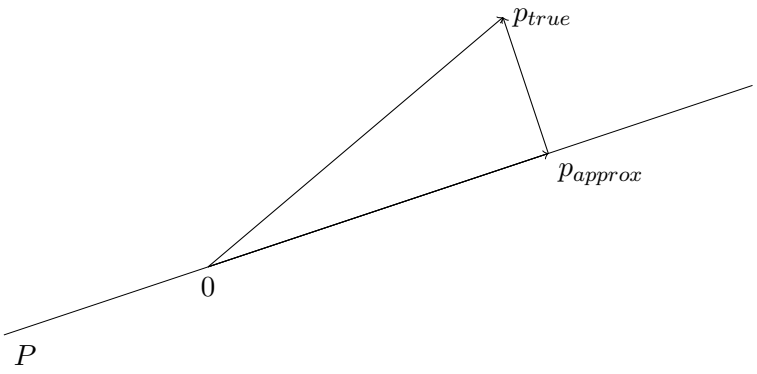


Figure 2.1.: Simple sketch of a best approximation of point p_{true} in a subspace P of a Hilbert space.

in Fig. 2.1, that in the three dimensional Euclidean space, a point p_{true} and a 2D plane P are given. The plane P contains all points - including the origin - that are usable as approximations for p_{true}. Then the best approximation point $p_{approx} \in P$ is the point nearest to p_{true}. That is p_{approx} is the solution of

$$||p_{true} - p|| \stackrel{!}{=} min \text{ for } p \in P \ .$$

From the geometrical perspective this is equivalent to stating that $p_{true} - p_{approx}$ is orthogonal to P. Which is equivalent to:

$$\forall p_{test} \in P : < p_{true} - p_{approx}, p_{test} >= 0 \ .$$

For the approximation of functions as elements of Hilbert spaces over real numbers there is a corresponding theorem:

Theorem 2.1.1: The Classical Projection Theorem (from Luenberger[2] p.51)
Let H be a Hilbert space and P be a closed subspace of H. Corresponding to any vector $p_{true} \in H$, there is a unique vector $p_{approx} \in P$ such that

$$||p_{true} - p_{approx}|| \leq ||p_{true} - p_{test}|| \ \ \forall p_{test} \in P \ . \tag{2.3}$$

Furthermore, a necessary and sufficient condition that $p_{approx} \in P$ be the unique minimizing vector is that $p_{true} - p_{approx}$ be orthogonal to P.

For numerical approximations of solutions of a pde a finite dimensional subspace P from a Hilbert space H is selected, where H contains all relevant solutions. For example, P could be a finite element space. Let $(e_i)_{i \in \mathbb{N}}$ be a complete orthonormal

sequence of H, for which $(e_0, ..., e_{N-1})$ is a orthonormal base of P. Then every vector p in H including the solutions of the pde can be written as

$$p = \sum_{i=0}^{\infty} < p, e_i > e_i \tag{2.4}$$

therefore

$$\begin{aligned} p_{approx} &= \sum_{i=0}^{\infty} < p_{approx}, e_i > e_i \\ &= \sum_{i \in \underline{N}} < p_{approx}, e_i > e_i \\ &= \sum_{i \in \underline{N}} < p_{true} - (p_{true} - p_{approx}), e_i > e_i \\ &= \left(\sum_{i \in \underline{N}} < p_{true}, e_i > e_i \right) - \left(\sum_{i \in \underline{N}} < p_{true} - p_{approx}, e_i > e_i \right) \\ &= \sum_{i \in \underline{N}} < p_{true}, e_i > e_i \ . \end{aligned}$$

The above equation is often used to numerically solve a pde, by first selecting a suitable approximation space P and then compute - to the precision possible - the coefficients $< p_{true}, e_i >$. Often used methods to do this are the Galerkin methods.

2.2. Galerkin Methods

To derive Galkerin methods it is assumed that the residual operator

$$\mathfrak{F} := u \mapsto (x \mapsto F(\partial^k u(x), \partial^{k-1} u(x), ..., \partial u(x), u(x), x)) \tag{2.5}$$

is an affine linear mapping. I.e. there exist a b_{true} such that $\mathfrak{L} := u \mapsto b_{true} + \mathfrak{F}(u)$ is linear. This implies that $b_{true} = -\mathfrak{F}(0)$. As its name suggests $\mathfrak{F}(u)$ yields a residual, because rearranging the definition of $\mathfrak{L}$ yields

$$\mathfrak{F}(u) = \mathfrak{L}(u) - b_{true} \tag{2.6}$$

where

$$\mathfrak{L}(u) - b_{true} = 0 \tag{2.7}$$

yields the solutions of the problem.

Beside the affine linearity, it is important to note, that the operator $\mathfrak{F}$ maps functions to functions. The residual operator $\mathfrak{F}$ has the property, that for all sufficiently differentiable functions u of type $U \to \mathbb{R}$ the result $\mathfrak{F}(u)$ is as well of type $U \to \mathbb{R}$. It is therefore reasonable to require $dom(\mathfrak{F}) = codom(\mathfrak{F})$. With

that formulation finding all solutions of the pde becomes the problem of finding all roots of $\mathfrak{F}$, i.e.: Finding all solutions to

$$\mathfrak{F} \quad : \quad H \longrightarrow H \tag{2.8}$$

$$\mathfrak{F}(u) \quad = \quad 0 \ . \tag{2.9}$$

To derive Galerkin methods for this problem it is further required that

$$H = dom(\mathfrak{F}) \tag{2.10}$$

is a Hilbert space. Then a finite dimensional subspace $P = \overline{span(e_0, ..., e_{N-1})}$ of H is chosen as space of possible approximations of the pdes solution. The best approximation in P to the solution is due to theorem 2.1.1

$$u_{approx} = \sum_{i \in \underline{N}} < u_{true}, e_i > e_i \ . \tag{2.11}$$

Under the assumption, that u_{approx} is itself a solution, this leads to

$$\begin{aligned} & \mathfrak{F}(u_{approx}) = 0 \\ \Leftrightarrow \quad & \mathfrak{F}(\sum_{i \in \underline{N}} < u_{true}, e_i > e_i) = 0 \\ \Leftrightarrow \quad & \mathfrak{F}(\sum_{i \in \underline{N}} < u_{true}, e_i > e_i) + b_{true} - b_{true} = 0 \\ \Leftrightarrow \quad & \sum_{i \in \underline{N}} < u_{true}, e_i > \mathfrak{L}(e_i) = b_{true} \ . \end{aligned}$$

It is not in general possible to numerically determine whether the function on the left-hand side is the function on the right hand side[1], therefore the right-hand side is approximated by choosing an other subspace $\tilde{P} = \overline{span(\tilde{e}_0, ..., \tilde{e}_{\tilde{N}-1})}$ of H to approximate b_{true}. This leads to the equation

$$b_{approx} \quad = \quad \sum_{j \in \underline{\tilde{N}}} < b_{true}, \tilde{e}_j > \tilde{e}_j \tag{2.12}$$

$$= \quad \sum_{j \in \underline{\tilde{N}}} < \sum_{i \in \underline{N}} < u_{true}, e_i > \mathfrak{L}(e_i), \tilde{e}_j > \tilde{e}_j \tag{2.13}$$

$$= \quad \sum_{j \in \underline{\tilde{N}}} \sum_{i \in \underline{N}} < \mathfrak{L}(e_i), \tilde{e}_j >< u_{true}, e_i > \tilde{e}_j \ . \tag{2.14}$$

This is equivalent to finding solutions for the linear equation

$$Ax = b \tag{2.15}$$

[1] This is due to the halting problem. The class of all mappings contains the set of mappings representable by the lambda calculus. The set of all functions generated by the lambda calculus is isomorphic to the set of all Turing machines. Thus, comparing computable functions is equivalent to the problem of deciding whether two Turing machines yield equal output for every equal input.

with

$$A_{ij} = < \mathfrak{L}(e_i), \tilde{e}_j > \tag{2.16}$$
$$x_j = < u_{true}, e_j > \tag{2.17}$$
$$b_i = < b_{true}, \tilde{e}_i > \ . \tag{2.18}$$

The actual approximate numerical solution is then:

$$u_{approx} = \sum_{i \in \underline{N}} x_i \, e_i = \sum_{i \in \underline{N}} < u_{true}, e_i > e_i \tag{2.19}$$

Boundary conditions are used to select an unique solution. For brevity it will not be discussed how this is done.

2.3. Galerkin Methods in Optimal Residual Formulation

The use of the classical projection theorem in the prior description of Galerkin methods does already give a hint on how numerically solving a pde can involve an optimization problem. With the definitions of the previous section we will derive Galerkin Methods from an optimal residual formulation. First it should be made explicit why 1.5 is referred to as an optimal *residual* formulation. Again, let the linear operator part of $\mathfrak{F}$ be

$$\mathfrak{L}(u) := \mathfrak{F}(u) + b_{true} \ . \tag{2.20}$$

With this the root problem 2.9 is equivalent to

$$\mathfrak{F}(u) = \mathfrak{L}(u) - b_{true} = 0 \tag{2.21}$$

so the idea in the motivation chapter can be applied. However before doing that, b_{true} is again, for practical reasons, approximated by

$$b_{approx} = \sum_{j \in \underline{\tilde{N}}} < b_{true}, \tilde{e}_j > \tilde{e}_j \tag{2.22}$$

where $(\tilde{e}_i)_{i \in \mathbb{N}}$ is a complete orthonormal sequence of $\overline{span(codom(\mathfrak{L}))}$ and $\tilde{N} \in \mathbb{N}$ is chosen big enough to approximate b_{true} well. With this the following approximate operator is defined:

$$\tilde{\mathfrak{F}}(u) := \mathfrak{L}(u) - b_{approx} \ . \tag{2.23}$$

For this operator the (informal) optimal residual problem is given by solving

$$||\tilde{\mathfrak{F}}(u)|| \stackrel{!}{=} min \ . \tag{2.24}$$

Then idea of the motivation chapter is applied backwards:

$$\begin{aligned} & ||\tilde{\mathfrak{F}}(u)|| \stackrel{!}{=} min \\ \Leftrightarrow & ||\tilde{\mathfrak{F}}(u)|| = 0 \\ \Leftrightarrow & \tilde{\mathfrak{F}}(u) = 0 \end{aligned}$$

Further instead of solving the optimization problem 2.24 exact a finite dimensional approximation space $P = span(e_0, ..., e_{N-1})$ is selected for representing the solutions. Thus we solve as approximation

$$\tilde{\mathfrak{F}}(u) = 0 \text{ for } u \in P$$

which is equivalent to

$$\forall u \in P : j \in \mathbb{N} : < \tilde{\mathfrak{F}}(u), \tilde{e}_j > \; = 0 \; .$$

This is equivalent to the two conditions

$$\begin{aligned} \forall u \in P : j \in \mathbb{N} \setminus \underline{\tilde{N}} \quad &: \quad < \mathfrak{L}(u), \tilde{e}_j > = 0 \\ \forall u \in P : j \in \mathbb{N} \quad &: \quad < \mathfrak{L}(u) - b_{approx}, \tilde{e}_j > = 0 \; . \end{aligned}$$

The second of the above equations expressed as a vector equation yields:

$$\sum_{j \in \underline{\tilde{N}}} < \mathfrak{L}(u) - b_{approx}, \tilde{e}_j > \tilde{e}_j = 0 \tag{2.25}$$

Again, due to the classical projection theorem, the best approximation of the solution in P is

$$u_{approx} = \sum_{i \in \underline{N}} < u_{true}, e_i > e_i \; .$$

With the trivial consequence of equation 2.22 that $< b_{true}, \tilde{e}_j > = < b_{approx}, \tilde{e}_j >$ for all $j \in \underline{\tilde{N}}$ and with equation 2.25 this yields:

$$\begin{aligned} 0 \;&=\; \sum_{j \in \underline{\tilde{N}}} < \mathfrak{L}(u_{approx}) - b_{approx}, \tilde{e}_j > \tilde{e}_j \\ &=\; \sum_{j \in \underline{\tilde{N}}} < \mathfrak{L}(\sum_{i \in \underline{N}} < u_{true}, e_i > e_i) - b_{approx}, \tilde{e}_j > \tilde{e}_j \\ &=\; \sum_{j \in \underline{\tilde{N}}} \sum_{i \in \underline{N}} < \mathfrak{L}(e_i), \tilde{e}_j > < u_{true}, e_i > \tilde{e}_j - \sum_{j \in \underline{\tilde{N}}} < b_{true}, \tilde{e}_j > \tilde{e}_j \end{aligned}$$

Therefore we finally arrive at the equation:

$$\sum_{j \in \underline{\tilde{N}}} < b_{true}, \tilde{e}_j > \tilde{e}_j = \sum_{j \in \underline{\tilde{N}}} \sum_{i \in \underline{N}} < \mathfrak{L}(e_i), \tilde{e}_j > < u_{true}, e_i > \tilde{e}_j \tag{2.26}$$

This is, by comparison with equation 2.12 and equation 2.14, equivalent to the Galerkin method of the previous section.

This summaries the connection between residual algorithms and Galerkin methods. Again, for brevity, the handling of boundary conditions is not discussed in this case. In the rest of this work the optimal residual formulations will be investigated with more rigor including treatment of non linear pdes and boundary conditions.

3. Mathematical Framework

> This may seem wasted or misguided effort, but in fact mathematicians are like theologians: we regard existence as the prime attribute of what we study. But unlike theologians, we need not always rely upon faith alone.
>
> *Lawrence C. Evans ([1] p. 9)*

A deeper discussion of optimal residual formulations and derived algorithms requires a sufficient mathematical framework, whose initial development has been performed for this work.

This chapter lays out the basic framework used to formulate optimal residual formulations. While I tried to avoid much of the complexities that arise in functional analysis, this was only partially successful. However the framework provides a constructive approach well suited for applied and computational mathematics, but it can not hide the functional analytic background it is based on.

In the following text corollaries will have no proof, since corollaries are meant to be, at least in this work, statements whose poof is trivial.

3.1. Partial Differential Equations and Their Solutions

Since this work is about partial differential equations, these should be defined first. To do this the following notation (following Forster[6] p. 55) is required:

Definition 3.1.1: Multiindex

A multiindex k of order N and dimension d is a d-tuple

$$k = (k_0, ..., k_{d-1}) \text{ where } \sum_{i=0}^{d-1} k_i = N \ . \tag{3.1}$$

Let $x \in \mathbb{R}^d$ and ∂_i for $i = 0, ..., d-1$ partial derivations, then the following further notations are defined for brevity:

$$x^k := x_0^{k_0} x_1^{k_1} x_2^{k_2} ... x_{d-1}^{k_{d-1}} \tag{3.2}$$

$$\partial^k := \partial_0^{k_0} \partial_1^{k_1} \partial_2^{k_2} ... \partial_{d-1}^{k_{d-1}} \tag{3.3}$$

The order of k can be written as $|k|$, i.e. $|k| = N$. Multiindices may be enumerated. This is indicated by using a symbolically meant sequence $k, k-1, ..., 1$, where the actually selected order is irrelevant for this work.

Following Evans[1] p. 1 partial differential equations can be defined as:

Definition 3.1.2: Partial Differential Equation (pde)
Let $U \subset \mathbb{R}^n$ be open and k be a multiindex of dimension n, then

$$F(\partial^k u(x), \partial^{k-1} u(x), ..., \partial^1 u(x), u(x), x) = 0 \ \ (\forall x \in U) \tag{3.4}$$

is called a $|k|^{th}$ order partial differential equation, where the structural function

$$F : \mathbb{R} \times ... \times \mathbb{R} \to \mathbb{R} \tag{3.5}$$

is given and $u : U \to \mathbb{R}$ is the unknown. In this work F is called structural function of the pde. Any function u realizing equation 3.4 is called a solution of the pde, where the partial derivations $\partial_i, i \in \underline{n}$ are usually understood as weak partial derivations.

To reformulate the partial differential equation in a neat way, suitable for the functional analytic setting used for optimal residual formulations, we will introduce the following definition:

Definition 3.1.3: Residual Operator of a pde
Let F be the structural function of a k^{th} order pde, then its residual operator[1] $\mathfrak{F}$ is defined as:

$$\mathfrak{F} := u \mapsto (x \mapsto F(\partial^k u(x), \partial^{k-1} u(x), ..., \partial u(x), u(x), x)) \tag{3.6}$$

Obviously all functions in the kernel of the residual operator of a pde are solutions of that pde, where the term kernel is extended to arbitrary functions:

[1] The quite simple definitions of residual operator, extended kernel and weak extended kernel, are results of this work. Since they allow to speak concisely of several aspects of pdes and optimal residual formulations their introduction should be justified.

Definition 3.1.4: Kernel
Let V, W be sets with $0 \in W$ and $\mathfrak{F} : V \to W$, than the kernel of $\mathfrak{F}$ is defined as

$$\ker(\mathfrak{F}) := \mathfrak{F}^{-1}(\{0\}) := \{v \in V | \mathfrak{F}(v) = 0\} \ . \tag{3.7}$$

Further, as the name "residual operator" suggests, $\mathfrak{F}(v) : U \to \mathbb{R}$ is a mapping that maps each point in U to its associated point-wise residual error for any "approximate" solution v. However in practice solutions are usually approximated. Therefore one would like to characterize all possible solutions reachable by sequences of approximations. For this we will introduce the following definition:

Definition 3.1.5: Extended Kernel
Let V, W be Banach spaces and $\mathfrak{F} : V \to W$. The extended kernel[1] is defined as

$$\operatorname{exker}(\mathfrak{F}) := \{v_\infty \in V | \exists (v_n)_{n\in\mathbb{N}} \subset V : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} \mathfrak{F}(v_n) = 0\}$$

However the extended Kernel is associated with the concept of strong convergence. Since often weak solutions are required, this still can be a too restrictive set for identifying possible solutions for pdes. Thus we define:

Definition 3.1.6: Weakly Extended Kernel
Let V, W be Banach spaces and $\mathfrak{F} : V \to W$. The weakly extended kernel[1] is defined as

$$\overset{w}{\operatorname{exker}}(\mathfrak{F}) := \{v_\infty \in V | \exists (v_n)_{n\in\mathbb{N}} \subset V : \overset{w}{\lim_{n\to\infty}} v_n = v_\infty \ \wedge \overset{w}{\lim_{n\to\infty}} \mathfrak{F}(v_n) = 0\} \ .$$

For the above definition of a (weakly) extended kernel it is crucial to be aware that the following is true:

$$\exists \mathfrak{F} : \exists v \in \overset{(w)}{\operatorname{exker}}(\mathfrak{F}) : \mathfrak{F}(v) \neq 0 \tag{3.8}$$

In fact we have the relations described by the following corollary:

Corollary 3.1.7:
Let V, W be Banach spaces, then for all $\mathfrak{F} : V \to W$:

$$\ker(\mathfrak{F}) \subset \operatorname{exker}(\mathfrak{F}) \subset \overset{w}{\operatorname{exker}}(\mathfrak{F}) \tag{3.9}$$

Beside the fact, that describing real world problems requires weak solutions in some cases, they provide the further convenience, that the partial derivation operators are continuous with respect to weak convergence in $L^1_{loc}(\Omega)$ (Struwe[5] p. 263).

For most problems the set $\overset{w}{\operatorname{exker}}(\mathfrak{F})$ can be seen as the set of all possible solutions. That is, if they actually are solutions. To speak about that, we introduce the following notions:

Definition 3.1.8: Minimal Extended Kernel
A function $\mathfrak{F}$ is said to have a minimal extended kernel if

$$\ker(\mathfrak{F}) = \operatorname{exker}(\mathfrak{F}) \ . \tag{3.10}$$

Definition 3.1.9: Minimal Weakly Extended Kernel
A function $\mathfrak{F}$ is said to have a minimal weakly extended kernel if

$$\ker(\mathfrak{F}) = \overset{w}{\operatorname{exker}}(\mathfrak{F}) \ . \tag{3.11}$$

With these definitions it can be stated that residual operators that have a minimal (weakly) extended kernel have the following nice property, that allows to construct a solution from a (weakly) convergent sequence of approximations:

Lemma 3.1.10:
If $\mathfrak{F}$ is a function with a minimal extended kernel, then

$$\lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} \mathfrak{F}(v_n) = 0 \Rightarrow \mathfrak{F}(v_\infty) = 0 \tag{3.12}$$

and if $\mathfrak{F}$ is a function with a minimal weakly extended kernel, then

$$\overset{w}{\lim_{n\to\infty}} v_n = v_\infty \wedge \overset{w}{\lim_{n\to\infty}} \mathfrak{F}(v_n) = 0 \Rightarrow \mathfrak{F}(v_\infty) = 0 \ . \tag{3.13}$$

Proof. We proof the second case. First $\overset{w}{\lim_{n\to\infty}} v_n = v_\infty$ and $\overset{w}{\lim_{n\to\infty}} \mathfrak{F}(v_n) = 0$ implies that $v_\infty \in \overset{w}{\operatorname{exker}}(\mathfrak{F})$. Since $\mathfrak{F}$ has a minimal weakly extended kernel, this implies that $v_\infty \in \ker(\mathfrak{F})$ and therefore $\mathfrak{F}(v_\infty) = 0$. The poof for the strong case is analogous. □

From a practical point of view the whole nomenclature seems not yet to be very useful. It is, by no means clear, if any real world partial differential equations have residual operators with minimal (weakly) extended kernels. However the restriction to residual operators with minimal (weakly) extended kernels is not very strong. In fact it is possible to relate the property of having a minimal (weakly) extended kernel to some notion of well posedness of a partial differential equation problem. So it should be expected that a wide range of practical problems will yield residual operators with minimal (weakly) extended kernel. However well posedness is usually not formally defined, which causes additional problems. To show the connection I will propose a (quite strong) formal definition of well posedness. Before that, we start with an informal definition of well posedness:

Definition 3.1.11: Well Posed Problem (from Evans[1] p. 7)
A partial differential equation is well-posed if

1. *The problem has in fact a solution.*
2. *The solution is unique.*
3. *The solution depends continuously on the data given in the problem.*

The first most obvious issue that needs to be addressed is, what can point three mean in a formal sense. To clarify this we will need some additional definitions regarding the (boundary) data. In this work I will try to avoid details of functional analysis, so instead of laying out the theory of Sobolev- and associated spaces, I will use a definition to catch essential properties of this spaces and assume their existence as known. First we define a space suitable to contain solutions and where boundary data can be properly defined.

Definition 3.1.12: Solution Space
A set S of functions over a domain $\Omega \subset \mathbb{R}^n$ to $\mathbb{R}$ is a (real) solution space if:

1. *S is a Banach space.*
2. *There exists a Banach space S_∂ of mappings from $\partial\Omega$ to $\mathbb{R}$, such that the linear mapping*
$$id_\partial : S \longrightarrow S_\partial \tag{3.14}$$

defined by the property

$$\forall v \in C^{\infty}(\overline{\Omega}) : id_{\partial}(v) = v|_{\partial\overline{\Omega}} \tag{3.15}$$

exists and is continuous.

Definition 3.1.13: Boundary Norm

Let S be a solution space, then the norm of the space S_{∂}, denoted as

$$\|\cdot\|_{\partial} : S_{\partial} \longrightarrow \mathbb{R}^{+} \ , \tag{3.16}$$

is called the boundary norm of the space.

Solution spaces exist, of course. For most applications Sobolev spaces are adequate solution spaces, and the well known trace theorems proof that. See for example Dobrowolski[3] p. 109 and 219, Knabner[4] et al. p. 89 or Evans[1] p. 258. After having solution spaces defined, it is necessary to describe in formal, what is meant by the term data in the informal definition of a well posed problem. For the sake of describing optimal residual formulations we need to define boundary conditions in terms of operators.

Definition 3.1.14: Boundary Operator

Let S be a solution space and $W \subset S_{\partial}$. Then any operator

$$\Gamma : S \to W \tag{3.17}$$

is called a boundary operator.

Definition 3.1.15: Boundary Data

Let $\Gamma : S \to W$ be a boundary operator than any function $\gamma \in W$ is called boundary data for Γ.

Definition 3.1.16: Boundary Condition
Let $\gamma \in W$ be boundary data for the boundary operator $\Gamma : S \to W$. Then

$$\Gamma(\cdot) = \gamma \tag{3.18}$$

is a boundary condition and a function f realizes the condition if $\Gamma(f) = \gamma$.

With this operator centric definitions the well known Dirichlet- and Neumann-boundary conditions can be defined:

Definition 3.1.17: Dirichlet Boundary Condition
A boundary condition $\Gamma(\cdot) = \gamma$ is called Dirichlet boundary condition if

$$\Gamma = id_{\partial} \ . \tag{3.19}$$

Definition 3.1.18: Neumann Boundary Condition
A boundary condition $\Gamma(\cdot) = \gamma$ is called Neumann boundary condition if

$$\Gamma = id_{\partial} \circ \frac{\partial}{\partial n} \ . \tag{3.20}$$

However these conditions define Dirichlet or Neumann conditions on the whole boundary. For a combination of different conditions the boundary operator needs to be constructed piece wise. Before a well posed pde problem can be defined formally, a pde problem needs to be defined formally:

Definition 3.1.19: Partial Differential Equation Problem
A partial differential problem is given by a triple $(\mathfrak{F}, \Gamma, \gamma)$, (or optional a quadruple $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$) where $\mathfrak{F}$ is the residual operator of a partial differential equation and and γ is boundary data of a boundary Operator Γ. Finding all functions $f \in S$ for the solution space $S = dom(\mathfrak{F})$ that realize

$$\mathfrak{F}(f) = 0 \tag{3.21}$$
$$\Gamma(f) = \gamma \tag{3.22}$$
$$\left[\mathfrak{C}(f) = 0\right] \tag{3.23}$$

is a partial differential equation problem[2]*, where* $\mathfrak{C}(f) = 0$ *is an optional constraint condition, that is required to select relevant solutions. For example unique entropy solutions for weak formulations.*

For practical purposes of computability the notion of well posedness of a problem is necessary. This notion is motivated by the fact that boundary conditions are often given by measurements of some experiment. Measurements are never exact. Well posedness of a pde models the condition, that despite of this, solutions of the pde with the measured data are similar to the solution with the true boundary data. The idea can be clarified by the following thought experiment: A perfect experimenter can infinitely increase the preciseness of its measurements, so that the resulting sequence of measured data is convergent. A well posed pde problem should then have the property, that the solutions, for the measured boundary data, exist and converge to the unique solution the true boundary data would yield.

Definition 3.1.20: Well Posed Partial Differential Equation Problem.
A partial differential equation problem $(\mathfrak{F}, \Gamma, \gamma)$ *or* $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ *is well posed*[3] *if there exist a neighborhood* $D \subset codom(\Gamma)$ *of* γ *such that*

1. *A solution exits for all boundary data* $d \in D$
2. *The solution is unique for all boundary data* $d \in D$
3. *For all (weakly) convergent series of boundary data* $(d_j)_{j\in\mathbb{N}} \subset D$ *the following holds:*
   a) *The sequence of solutions* $f_{sol,j}$ *is a (weakly) convergent sequence, where* $f_{sol,j}$ *is the unique solution to the problem with boundary data* d_j*.*
   b) *Let* $f_{sol,\infty} := \overset{(w)}{\lim_{j\to\infty}} f_{sol,j}$*, then*

$$\mathfrak{F}(f_{sol,\infty}) = 0 \wedge \Gamma(f_{sol,\infty}) = d_\infty \tag{3.24}$$

   and if $\mathfrak{C}$ *is given:* $\mathfrak{C}(f_{sol,\infty}) = 0$

[2] There are certainly more problems regarding pdes then finding a set of solutions realizing a boundary condition. However this is a common nomenclature, so we stick to it.

[3] This is a rather restrictive notion of well posedness, since it conceptually requires that every measurement data, having at least a certain precision, can be used as boundary data that yields a unique solution. However I did not find the time to examine a more general notion of well posedness. Further this concept formalization requires that the boundary operator (and constraint operator) can be applied to every function in the solution space. This should be kept in mind with regard to the results obtained with this formalization.

With these definitions we can examine well posed problems regarding their (weakly) extended kernels.

Lemma 3.1.21:
The operator $\mathfrak{B} := v \mapsto \Gamma(v) - d$ *of a well posed problem* $(\mathfrak{F}, \Gamma, \gamma)$ *or* $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ *has a (weakly) minimal extended kernel in a neighborhood of the solution.*

Proof. Assume that this were not true, then there would exist a (weakly) convergent sequence of solutions $(f_n)_{n\in\mathbb{N}}$ with $\overset{(w)}{\lim_{n\to\infty}} \mathfrak{B}(f_n) = 0$ and $\mathfrak{B}(f_\infty) \neq 0$. Let $(d_n)_{n\in\mathbb{N}}$ be a (weakly) convergent sequence of boundary conditions such that $\Gamma(f_n) = d_n$, this sequence exists due to the well posedness of the problem, that implies that in a neighborhood there is an unique solution for every boundary data. Then

$$0 = \overset{(w)}{\lim_{n\to\infty}} \mathfrak{B}(f_n) = \overset{(w)}{\lim_{n\to\infty}} \Gamma(f_n) - d \Rightarrow \overset{(w)}{\lim_{n\to\infty}} \Gamma(f_n) = d \ , \tag{3.25}$$

but the well posedness implies that

$$\Gamma(f_\infty) = d_\infty = \overset{(w)}{\lim_{n\to\infty}} d_n = \overset{(w)}{\lim_{n\to\infty}} \Gamma(f_n) \tag{3.26}$$

and thus $d = d_\infty$. This however implies that $\mathfrak{B}(f_\infty) = 0$. □

Further, for an optional constraint operator $\mathfrak{C}$ we get the following lemma:

Lemma 3.1.22:
The constraint operator $\mathfrak{C}$ *of a well posed problem* $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ *has a minimal (weakly) extended kernel in a neighborhood of the solution.*

Proof. Assume that, this where not true, then there would exist a convergent sequence of solutions f_n with $\overset{(w)}{\lim_{n\to\infty}} \mathfrak{C}(f_n) = 0$ and and $\mathfrak{C}(f_\infty) \neq 0$. Let d_n be a (weakly) convergent sequence of boundary conditions such that $\Gamma(f_n) = d_n$. Then we have as shown in the proof of the previous lemma $d_\infty = d$. Therefore f_∞ is the unique and existing solution of the problem. This however implies that $\mathfrak{C}(f_\infty) = 0$. □

Finally we get an analogous result for the residual operator:

Lemma 3.1.23:
The residual operator $\mathfrak{F}$ of a well posed problem $(\mathfrak{F}, \Gamma, \gamma)$ or $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ has a minimal (weakly) extended kernel in a neighborhood of the solution.

Proof. Let D be a neighborhood on which the boundary operator of the well posed problem has a minimal extended kernel. Assume the lemma is wrong, then there exists a convergent sequence of functions $\lim\limits_{i\to\infty}^{(w)} f_i = f_{sol}$ in $dom(\mathfrak{F})$ with $\lim\limits_{n\to\infty}^{(w)} \mathfrak{F}(f_n) = 0$ and $\mathfrak{F}(f_\infty) \neq 0$. Further let $d_n := \Gamma(f_n)$, then well posedness implies that $\mathfrak{F}(f_\infty) = 0$. □

With this we conclude the part about partial differential equations and their solutions. So far we have shown that the classical solutions of a well posed problem are all functions in the minimal extended kernel of the residual operator, whereas the weak solutions are in the minimal weakly extended kernel. Here having a (weakly) extended kernel assures that the solutions can, in principle, be approximated by (weakly) convergent sequences of approximations.

3.2. Optimal Residual Formulations

Coming back to the motivational examples in the introduction one could simply apply the idea to the now introduced notations. That is if $(\mathfrak{F}, \Gamma, \gamma)$ or $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ is a well posed pde problem, solve

$$\mathfrak{G}(f) := ||\mathfrak{F}(f)|| \stackrel{!}{=} min$$

for some small enough neighborhood D of γ. Doing this does not consider boundary conditions and constraints, like for example entropy conditions. The next obvious idea would be to solve

$$\mathfrak{G}(f) := ||\mathfrak{F}(f)|| + ||\Gamma(f) - \gamma||_\partial \stackrel{!}{=} min \tag{3.27}$$

or

$$\mathfrak{G}(f) := ||\mathfrak{F}(f)|| + ||\Gamma(f) - \gamma||_\partial + ||\mathfrak{C}(f)|| \stackrel{!}{=} min\ . \tag{3.28}$$

This, from a practical point of view, is still not very good[4]. There are unnecessary square roots built in the norms, and finding the minimum might be faster if some positive scaling factors $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{R}^+$ are introduced. For example:

$$\mathfrak{G}(f) := \alpha_0||\mathfrak{F}(f)||^2 + \alpha_1||\Gamma(f) - \gamma||_\partial^2 + \alpha_2||\mathfrak{C}(f)||^2 \stackrel{!}{=} min \tag{3.29}$$

[4]Well, we simply ignore that, from a practical point of view, we made the problem only complexer. To this point it is not clear, if this approach is practical at all. But it was the idea of this work to examine that. Later I will show, that this is not really as bad as one might guess.

One even might for convergence-speedup imagine some more complicated expressions. However in any case the idea is to define an operator $\mathfrak{G}$ with the property, that if f_{min} is a solution to $\mathfrak{G}(f) \stackrel{!}{=} min$ this implies that $\mathfrak{F}(f_{min}) = 0$, $\Gamma(f_{min}) - \gamma = 0$ and, if $\mathfrak{C}$ is given, $\mathfrak{C}(f_{min}) = 0$. From the above examples it is clear that in an algorithm solving the above optimization problem will yield a minimizing sequence. Then we must require that these sequence actually approaches the solution. This motivates the following definition:

Definition 3.2.1: (Weak) Residual Gauge

Let S be a Banach space. A (weak) residual gauge $\mathfrak{G}$ on S is a mapping $\mathfrak{G} : S \to \mathbb{R}^+$ with the following properties:

1. *For all sequences $(u_n)_{n\in\mathbb{N}}$ it holds that*

$$\lim_{n\to\infty} \mathfrak{G}(u_n) = 0 \Longrightarrow \exists u_\infty \in S : \overset{(w)}{\lim_{n\to\infty}} u_n = u_\infty \wedge \mathfrak{G}(u_\infty) = 0 \tag{3.30}$$

2. *$ker(\mathfrak{G}) \neq \varnothing$*

This definition leaves open, whether the examples above actually are residual gauges. The theorems for the construction of residual gauges will be discussed in the next section. Note that these gauges are not intended to have any relation to the residual gauges used in field theory. They just have the same name by coincidence. Having this definition at hand, it is readably definable what an optimal residual formulation is:

Definition 3.2.2: (Weak) Optimal Residual Formulation

Regarding a partial differential equation problem $(\mathfrak{F}, \Gamma, \gamma)$ (resp. $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$) a optimal residual formulation is given by the problem

$$\mathfrak{G}(f) \stackrel{!}{=} min \ , \tag{3.31}$$

where $\mathfrak{G}$ is an additional (weak) residual gauge with the following property: For all sequences $(x_n)_{n\in\mathbb{N}}$ the following holds:

$$\lim_{n\to\infty} \mathfrak{G}(x_n) = 0 \implies \mathfrak{F}(x_\infty) = 0 \ \wedge \ \Gamma(x_\infty) = \gamma \ \big(\wedge \ \mathfrak{C}(x_\infty) = 0 \big) \tag{3.32}$$

The section will be concluded with some important results on residual gauges. First of all they exist:

Lemma 3.2.3:
For any fixed $u_{min} \in S$ the mapping

$$u \mapsto ||u - u_{min}|| \tag{3.33}$$

is a residual gauge .

Proof. The defining property 1 in definition 3.2.1 of that specific residual gauge is the definition of a sequence convergent to u_{min}. □

The second important property is that the kernel of a residual gauge contains exactly one element:

Lemma 3.2.4:
Let S be a Banach space. If $f : S \longrightarrow \mathbb{R}^+$ is a (weak) residual gauge then, $\#ker(f) = 1$.

Proof. Assume that $\{u_e, u_o\} \subset f^{-1}(\{0\})$. The sequence

$$u_n = \begin{cases} u_e \text{ if n is even} \\ u_o \text{ if n is odd} \end{cases} \tag{3.34}$$

has no (weak) limes by construction. However it is a minimizing sequence, therefore it should have a (weak) limes, since f is a (weak) residual gauge. Therefore $f^{-1}(\{0\})$ contains only one point. □

Further residual gauges are a huge class of mappings because:

Lemma 3.2.5:
A residual gauge is not necessarily continuous

Proof. The function $f : \mathbb{R} \longrightarrow \mathbb{R}^+$

$$f(x) = \begin{cases} -x + 2 \text{ for } x < 1 \\ x - 1 \text{ for } x \geq 1 \end{cases} \tag{3.35}$$

is not continuous at $x = 1$ but is a residual gauge. Because: If $(x_n)_{n\in\mathbb{N}}$ is a sequence with $\lim_{n\to\infty} f(x_n) = 0$ then there exists a $N \in \mathbb{N}$ so that $x_n \geq 1$ for all

$n > N$. Therefore, without loss of generality, it can be assumed that $x_n \in [1, \infty)$. However f is continuous on $[1, \infty)$ and has a continuous inverse there. Therefore

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} f^{-1}(f(x_n)) = f^{-1}(\lim_{n\to\infty} f(x_n)) = f^{-1}(\{0\}) = 1 \ .$$

□

At this point we have defined residual gauges by properties we wished it had regarding to convergence. To show the usefulness of the definition it is required to derive some tools for residual gauge construction, that make at least some of the initial examples of this section usable.

3.3. Construction of Residual Gauges

In the previous section there were some examples given, how residual gauges for pde problems might be constructed. For example: Is the suggested residual gauge

$$\mathfrak{G}(f) := \|\mathfrak{F}(f)\| + \|\Gamma(f) - \gamma\|_\partial + \|\mathfrak{C}(f)\|$$

indeed a residual gauge regarding the definition of the above section? In this section the theorems to proof, that such constructions are indeed residual gauges, are given. However, for the task of constructing residual gauges definition 3.2.1 is not very suitable. From a practical point of view an optimization algorithm, applied to an optimal residual formulation, will yield a minimizing sequence, that should have the property, that it approaches the solution of the problem. This notion of "approaching" can be formalized by the introduction of the following property:

Definition 3.3.1: (Weakly) Introversive Function

Let V, W be Banach spaces, $D \subset V$ and $\mathfrak{T} : D \to W$ then $\mathfrak{T}$ is called (weakly) introversive[5] *if for all sequences $(v_n)_{n\in\mathbb{N}} \subset D$*

$$\overset{(w)}{\lim_{n\to\infty}} \mathfrak{T}(v_n) = 0 \Rightarrow \lim_{n\to\infty} d(v_n, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) = 0 \tag{3.36}$$

(where $d(v, S) := \inf\{\|v - s\| \,|\, s \in S\}$).

With this definition residual gauges can indeed be characterized in a different way:

[5] I do not know if functions of this property have been examined elsewhere and what name has been used for them. I did not find any literature on that, if there is any, it would be nice to let me know.

Lemma 3.3.2:
Let S be a Banach space and $\mathfrak{G} : S \longrightarrow \mathbb{R}^+$. Then $\mathfrak{G}$ is a (weak) residual gauge if and only if all the following conditions hold:

1. *$\mathfrak{G}$ is (weakly) introversive*

2. *$\mathfrak{G}$ has a minimal (weakly) extended kernel, i.e. $\overset{(w)}{\operatorname{exker}}(\mathfrak{G}) = \ker(\mathfrak{G})$*

3. *$ker(\mathfrak{G})$ contains exactly one element, i.e. $\#ker(\mathfrak{G}) = 1$*

Proof. To proof the if part, let $(v_i)_{i\in\mathbb{N}} \subset V$ be a sequence with $\lim_{n\to\infty} \mathfrak{G}(v_n) = 0$, then, since $\mathfrak{G}$ is introversive, $\lim_{n\to\infty} d(v_n, \overset{(w)}{\operatorname{exker}}(\mathfrak{G})) = 0$. Further $\overset{(w)}{\operatorname{exker}}(\mathfrak{G})$ has only one element p, therefore $d(v_n, \overset{(w)}{\operatorname{exker}}(\mathfrak{G})) = d(v_n, \{p\}) = d(v_n, p)$. The sequence $(v_i)_{i\in\mathbb{N}} \subset V$ is therefore a convergent sequence with $\lim_{n\to\infty} v_n = p$. By lemma 3.1.10 this implies that $\mathfrak{G}(p) = 0$. To proof the only if part, that is that introversiveness implies 1., 2. and 3. assume the opposite. That is, assume that $\mathfrak{G}$ is a residual gauge, and there exists a case where 1., 2. or 3. is not true, then

1. if $\mathfrak{G}$ where not (weakly) introversive, then there exists a minimizing sequence with $\lim_{n\to\infty} d(v_n, \overset{(w)}{\operatorname{exker}}(\mathfrak{G})) \neq 0$. Then either v_n is not (weakly) convergent or v_n is (weakly) convergent and $\mathfrak{G}(v_\infty) \neq 0$.

2. if $\mathfrak{G}$ had not a minimal (weakly) extended kernel, then there is a (weakly) convergent minimizing sequence with $\lim_{n\to\infty} \mathfrak{G}(v_n) = 0$ and $\mathfrak{G}(v_\infty) \neq 0$.

3. if $ker(\mathfrak{G})$ did not contain only one element, that would directly contradict lemma 3.2.4.

□

This characterizations of a residual gauge decomposes it into properties that can investigated much more easily. Since residual gauges are usually constructed by taking norms of some operator the following lemma is foundational.

Lemma 3.3.3:
If $\mathfrak{T}$ is a (weakly) introversive function, then $v \mapsto \|\mathfrak{T}(v)\|$ is a (weakly) introversive function.

Proof.

$$\begin{aligned} & \lim_{n\to\infty} \|\mathfrak{T}(v_n)\| = 0 \\ \Rightarrow\quad & \lim_{n\to\infty} \mathfrak{T}(v_n) = 0 \\ \Big(\quad \Rightarrow\quad & \lim_{n\to\infty} \overset{w}{\mathfrak{T}}(v_n) = 0 \quad \Big) \\ \Rightarrow\quad & \lim_{n\to\infty} d(v_n, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) = 0 \ . \end{aligned}$$

□

To be able to use optimal residual formulations for at least linear partial differential equations, it is necessary to prove that at least weakly continuous affine linear operators are introversive. Further to be able to construct the examples of residual gauges it should be investigated under which conditions a weighted sum of introversive functions with positive weights is introversive. I will start with the discussion of the second case. To do this we will need further a property of the extended kernels of the weighted functions:

Definition 3.3.4: Clean Intersection

Let I be a countable index set and $(A_i)_{i\in I}$ be subsets of a Banach space H. The $(A_i)_{i\in I}$ have a clean intersection if for all $(x_n)_{n\in\mathbb{N}} \subset H$:

$$\left(\forall i \in I : \lim_{n\to\infty} d(x_n, A_i) = 0\right) \Longrightarrow \lim_{n\to\infty} d(x_n, \bigcap_{i\in I} A_i) = 0 \tag{3.37}$$

Lemma 3.3.5:

Let H be a Hilbert space, I an index set and $\forall i \in I : A_i \subset H$ be affine subspaces with $\bigcap_{i\in I} A_i \neq \varnothing$, then $(A_i)_{i\in I}$ has a clean intersection.

Proof. Without loss of generality we can assume that $0 \in \bigcap_{i\in I} A_i$, that is, that the A_i are subspaces. The intersection of subspaces is a subspace. For each two subspaces A_i, A_j a canonically defined angle α_{ij} between this subspaces is given. Now assume that $\lim_{n\to\infty} d(x_n, A_i) = 0$ and $\lim_{n\to\infty} d(x_n, A_j) = 0$. This is, due to the angle α_{ij} between the subspaces, only possible if $\lim_{n\to\infty} d(x_n, A_i \cap A_j) = 0$. Then the lemma follows by transfinite induction. □

With this we can answer the second problem:

Lemma 3.3.6:
Let $\omega_0, ..., \omega_{n-1} \in \mathbb{R}^+ \setminus \{0\}$, H be a Hilbert space and $f_0 : H \to \mathbb{R}^+, ..., f_{n-1} : H \to \mathbb{R}^+$ (weakly) introversive functions whose (weakly) extended kernels have a clean intersection. Then

$$f := \sum_{i \in \underline{n}} \omega_i f_i \tag{3.38}$$

is (weakly) introversive.

Proof. Obviously $\omega_i f_i$ is introversive if and only if f_i is introversive. We therefore only need to proof that

$$g := \sum_{i \in \underline{n}} f_i \tag{3.39}$$

is introversive. Let $(x_j)_{j \in \mathbb{N}}$ be a minimizing sequence of g, i.e. $\lim_{j \to \infty} g(x_j) = 0$. Since the f_i are non negative, this implies that $\forall i \in \underline{n} : \lim_{j \to \infty} f_i(x_j) = 0$. Since the (weakly) extended kernels are subspaces, they have a clean intersection (lemma 3.3.5) and the statement follows. □

To apply optimal residual formulations to partial differential equations, it remains to show introversiveness for at least affine linear operators.

Theorem 3.3.7: Affine Linear Functional Introversiveness
Let H be a Hilbert space, then every non constant affine linear[6] mapping

$$p : H \to \mathbb{R} \tag{3.40}$$

is introversive.

Proof. The following decomposition of p is used:

$$p \quad = \quad x \mapsto L(x) + c \tag{3.41}$$

[6] This theorem might seem more general than it is, in the sense that it uses a construction similar to building a weak closure of an operator. Thus it might be true only for densely defined operators and weak closures thereof, thus p might implicitly be weakly continuous. However I could not identify an error here, so I left the theorem formulated in the way it is. The theorem seems to be false for Banach spaces. An insight I owe to Prof. Dr. R. Lauterbach from the university of Hamburg, who derived quite convincingly, that it is possible to construct a counter example for some Banach spaces.

where c is a constant and $L \neq 0$, since p is not a constant mapping. Further let

$$\begin{aligned}(x_i)_{i\in\mathbb{N}} &\subset H \text{ with } \lim_{i\to\infty} p(x_i) = 0\\ H_i &:= \overline{span(\{x_j | j \in \{0,...,i\}\})}\\ \tilde{L}_i &:= L|_{H_i}\\ \tilde{p}_i &:= \tilde{L}_i + c\ .\end{aligned}$$

Then by definition $\tilde{p}_i(x_i) = p(x_i)$. The $\tilde{L}_i$ are defined on finite dimensional Hilbert spaces and thus are continuous. The representation theorem of Frechet-Riesz implies that there is a non zero vector $v_i \in H_i$ such that $\tilde{L}_i(x) = < v_i, x >$. This implies that $\tilde{p}_i(x)$ has the representation

$$\tilde{p}_i(x) = < v_i, x > +c\ .$$

The sequence of $(||v_i||)_{i\in\mathbb{N}}$ is nowhere vanishing and monotone increasing, because $v_0 \neq 0$ and

$$\forall x \in H_i : \tilde{L}_{i+1}(x) = \tilde{L}_i(x) \Longleftrightarrow \forall x \in H_i : < x, v_{i+1} - v_i >= 0\ .$$

That is, $v_{i+1} - v_i$ is orthogonal to H_i and therefore

$$||v_{i+1}||^2 = ||v_{i+1} - v_i + v_i||^2 = ||v_i||^2 + ||v_{i+1} - v_i||^2\ .$$

Let $x_{oi} := -\frac{cv_i}{<v_i,v_i>}$, then $< v_i, x_{oi} > = -c$ and $p_i(x_{oi}) = < v_i, x_{oi} > +c = 0$, thus $x_{oi} \in ker(\tilde{p}_i)$. Using x_{oi} the affine linear mappings $\tilde{p}_i$ can be rewritten as

$$\tilde{p}_i(x) \ = \ < v_i, x - x_{oi} >\ .$$

This implies that the kernel of $\tilde{p}_i$ is a hyper plane on which v_i is orthogonal. Therefore it follows as a consequence from the classical projection theorem, that

$$d(x, ker(\tilde{p}_i)) \ = \ < \frac{v_i}{||v_i||}, x - x_{oi} >\ .$$

Further it should be noted that by construction

$$\ker(\tilde{p}_i) \subset \ker(p) \subset \text{exker}(p) \subset \overset{w}{\text{exker}}(p) \tag{3.42}$$

and therefore

$$d(x_i, \overset{(w)}{\text{exker}}(p)) \leq d(x_i, \ker(p)) \leq d(x_i, \ker(\tilde{p}_i))\ .$$

Finally we can proof that:

$$\begin{aligned}
& \lim_{i\to\infty} p(x_i) = 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} \tilde{p}_i(x_i) = 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} < v_i, x_i - x_{oi} >= 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} < \frac{v_i}{||v_i||}, x_i - x_{oi} > ||v_i|| = 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} < \frac{v_i}{||v_i||}, x_i - x_{oi} >= 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} d(x_i, ker(\tilde{p}_i)) = 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} d(x_i, ker(p)) = 0 \\
\Longrightarrow\quad & \lim_{i\to\infty} d(x_i, \text{exker}(p)) = 0 \ .
\end{aligned}$$

□

Lemma 3.3.8: Affine Linear Mapping Introversiveness
Let H, B be Hilbert spaces, then every non constant[7] affine linear mapping

$$\mathfrak{T} : H \to B \tag{3.43}$$

is introversive and weakly introversive.

Proof. Let $(e_i)_{i\in\mathbb{N}}$ be a complete orthonormal sequence of B. Let $(x_j)_{j\in\mathbb{N}}$ be a sequence with

$$\overset{(w)}{\lim_{j\to\infty}} \mathfrak{T}(x_j) = 0 \ . \tag{3.44}$$

This then implies with $p_i(x_j) :=< \mathfrak{T}(x_j), e_i >$ that

$$\forall i \in \mathbb{N} : \lim_{j\to\infty} p_i(x_j) = \lim_{j\to\infty} < \mathfrak{T}(x_j), e_i >= 0 \ .$$

Due to the affine linearity of $\mathfrak{T}$, the kernels of the functionals p_i are affine subspaces of H, and therefore have a clean intersection. Further, theorem 3.3.7 implies that $\forall i \in \mathbb{N} : d(x_j, \overset{(w)}{\text{exker}}(p_i)) = 0$. Then with lemma 3.3.5 it follows that

$$d(x_j, \bigcap_{i\in\mathbb{N}} \overset{(w)}{\text{exker}}(p_i)) = 0 \ .$$

[7] Here, as in theorem 3.3.7, this might not be as general as it seems, thus a weak kind of continuity might be build in there somewhere.

With having in mind that

$$\mathfrak{T}(x_j) = \sum_{i\in\mathbb{N}} < \mathfrak{T}(x_j), e_i > e_i = \sum_{i\in\mathbb{N}} p_i(x_j)e_i \ .$$

we can derive that

$$\begin{aligned}
& \overset{(w)}{\operatorname{exker}}(\mathfrak{T}) \\
= \ & \{x_\infty \in H | \exists (x_n)_{n\in\mathbb{N}} \subset H : \overset{(w)}{\lim_{n\to\infty}} x_n = x_\infty \wedge \overset{(w)}{\lim_{n\to\infty}} \mathfrak{T}(x_n) = 0\} \\
= \ & \{x_\infty \in H | \exists (x_n)_{n\in\mathbb{N}} \subset H : \overset{(w)}{\lim_{n\to\infty}} x_n = x_\infty \wedge \overset{(w)}{\lim_{n\to\infty}} \sum_{i\in\mathbb{N}} p_i(x_n)e_i = 0\} \\
= \ & \{x_\infty \in H | \exists (x_n)_{n\in\mathbb{N}} \subset H : \overset{(w)}{\lim_{n\to\infty}} x_n = x_\infty \wedge \forall i \in \mathbb{N} : \lim_{n\to\infty} p_i(x_n) = 0\} \\
= \ & \bigcap_{i\in\mathbb{N}} \{x_\infty \in H | \exists (x_n)_{n\in\mathbb{N}} \subset H : \overset{(w)}{\lim_{n\to\infty}} x_n = x_\infty \wedge \lim_{n\to\infty} p_i(x_n) = 0\} \\
= \ & \bigcap_{i\in\mathbb{N}} \overset{(w)}{\operatorname{exker}}(p_i) \ .
\end{aligned}$$

□

To construct residual gauges as suggested at the beginning of the section, we need to investigate the combination of functions with minimal (weakly) extended kernel. By any analogous argument to lemma 3.3.6 one gets the following corollary:

Corollary 3.3.9:
Let $\omega_0, ..., \omega_{n-1} \in (0, \infty)$, H be a Hilbert space and $f_0 : H \to \mathbb{R}^+, ..., f_{n-1} : H \to \mathbb{R}^+$ with minimal (weakly) extended kernels that have a clean intersection. Then

$$f := \sum_{i\in\underline{n}} \omega_i f_i \tag{3.45}$$

has a minimal (weakly) extended kernel.

With the above results it is now possible to construct a wide range or residual gauges for at least linear partial differential equations. This is exemplary shown in the following lemma:

Lemma 3.3.10: Residual Gauge Construction Example

For a well posed linear partial differential equation problem $(\mathfrak{F}, \Gamma, \gamma, \mathfrak{C})$ *on a Hilbert solution space, with affine linear constraint operator* $\mathfrak{C}$

$$\mathfrak{G}(f) := \|\mathfrak{F}(f)\| + \|\Gamma(f) - \gamma\|_\partial + \|\mathfrak{C}(f)\|$$

is a (weak) residual gauge on a neighborhood of γ*.*

Proof. Kernels of affine linear operators are affine subspaces, and thus have a clean intersection due to lemma 3.3.5, therefore

1. lemma 3.3.6 implies, as a consequence of the lemmas 3.3.8 and 3.3.3 that $\mathfrak{G}$ is (weakly) introversive and
2. corollary 3.3.9 implies, as a consequence of lemma 3.1.21, lemma 3.1.22 and lemma 3.1.23, that $\mathfrak{G}$ has a minimal (weakly) extended kernel on a neighborhood of γ.
3. Further, the well posedness of the problem implies, that the solution f_{sol} of the problem exists and is unique, therefore $\ker(\mathfrak{G}) = \{f_{sol}\}$.

Then 1, 2 and 3 imply by lemma 3.3.2 that $\mathfrak{G}$ is a residual gauge on a neighborhood of γ. □

This proofs that the proposed formalism is suitable for solving at least linear partial differential problems.

The section will be concluded with theorems that show, that introversiveness is associated to the notion of coerciveness[8] in the following sense:

Theorem 3.3.11:

Let A, B *be Banach spaces (where* A *is reflexive). Further let* $\mathfrak{T} : A \to B$ *be (weakly) continuous with nonempty weakly extended kernel, then*

$$\overset{(w)}{\lim_{i\to\infty}} \mathfrak{T}(x_i) = 0 \wedge \lim_{i\to\infty} d(x_i, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) \neq 0 \tag{3.46}$$

$$\implies \lim_{i\to\infty} \|x_i\| = \infty \ . \tag{3.47}$$

Proof. Let $\overset{(w)}{\lim_{i\to\infty}} \mathfrak{T}(x_i) = 0$ and $\lim_{i\to\infty} d(x_i, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) \neq 0$, then (x_i) is not a (weakly) convergent sequence. If it were, then $\overset{(w)}{\lim_{i\to\infty}} \mathfrak{T}(x_i) = \mathfrak{T}(\overset{(w)}{\lim_{i\to\infty}} x_i) = 0$ due to the (weak)

[8] Where f is coercive means that $\lim_{n\to\infty} \|x_n\| = \infty \Rightarrow \lim_{n\to\infty} f(x_n) = \infty$ for all sequences $(x_n)_{n\in\mathbb{N}}$ in a Banach space.

continuity of $\mathfrak{T}$, and that implies $\lim_{i\to\infty} d(x_i, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) = 0$. Further

$$\exists r > 0 : \forall z \in \overset{(w)}{\operatorname{exker}}(\mathfrak{T}) : \forall i \in \mathbb{N} : \exists n > i : \|z - x_n\| > r \ . \tag{3.48}$$

This implies that there exist a sub-sequence $(x_{n_i})_{i\in\mathbb{N}}$ with

$$\forall z \in \overset{(w)}{\operatorname{exker}}(\mathfrak{T}) : \forall i \in \mathbb{N} : \|z - x_{n_i}\| > r \ . \tag{3.49}$$

If the sequence $(x_{n_i})_{i\in\mathbb{N}}$ were contained in a sphere $B_R(0)$ i.e. bounded, then there would be an (weak) accumulation point (due to the reflexiveness of A) and therefore there would exist a (weakly) convergent subsequence $(x_{n_{i_j}})_{j\in\mathbb{N}}$ of $(x_{n_i})_{i\in\mathbb{N}}$ and of $(x_n)_{n\in\mathbb{N}}$ that converges (weakly) to the accumulation point x_{acc}. However $\overset{(w)}{\lim_{n\to\infty}} \mathfrak{T}(x_n) = 0$ and therefore $\overset{(w)}{\lim_{j\to\infty}} \mathfrak{T}(x_{n_{i_j}}) = 0$. I.e. $(x_{n_{i_j}})_{j\in\mathbb{N}}$ converges (weakly) to a point in the (weakly) extended kernel, which is a contradiction to equation 3.49. This establishes that $\lim_{i\to\infty} \mathfrak{T}(x_i) = 0$ and $\lim_{i\to\infty} d(x_i, \overset{(w)}{\operatorname{exker}}(\mathfrak{T})) \neq 0$ implies that $(x_n)_{n\in\mathbb{N}}$ is a strictly divergent series, i.e.:

$$\lim_{n\to\infty} \|x_n\| = \infty \ . \tag{3.50}$$

□

With this we get the following corollary for introversive functions.

Corollary 3.3.12:
Let A be a reflexive Banach space, $\mathfrak{T} : A \to \mathbb{R}$ be (weakly) continuous and not (weakly) introversive, then there exists a sequence $(x_n)_{n\in\mathbb{N}}$ with $\overset{(w)}{\lim_{n\to\infty}} \mathfrak{T}(x_n) = 0$, that $(x_n)_{n\in\mathbb{N}}$ is not bounded.

However, for applications weak continuity of the residual operator of a pde is not often given, but an analogous result can be proven for weakly lower semi-continuous functions:

Theorem 3.3.13:
Let A be a reflexive Banach space. Further let $\mathfrak{T} : A \to \mathbb{R}^+$ be weakly lower semi-continuous with nonempty weakly extended kernel, then

$$\overset{w}{\lim_{i\to\infty}} \mathfrak{T}(x_i) = 0 \wedge \lim_{i\to\infty} d(x_i, \overset{w}{\operatorname{exker}}(\mathfrak{T})) \neq 0 \tag{3.51}$$

$$\implies \lim_{i\to\infty} \|x_i\| = \infty \tag{3.52}$$

Proof. Let $\overset{w}{\lim_{i\to\infty}}\mathfrak{T}(x_i) = 0$ and $\lim_{i\to\infty} d(x_i, \overset{w}{\operatorname{exker}}(\mathfrak{T})) \neq 0$, then $(x_i)_{i\in\mathbb{N}}$ is not a weakly convergent sequence. If it were, then $\mathfrak{T}(\overset{w}{\lim_{i\to\infty}} x_i) \leq \overset{w}{\lim_{i\to\infty}}\mathfrak{T}(x_i) = 0$ due to the weak continuity of $\mathfrak{T}$, and that implies that $\mathfrak{T}(\overset{w}{\lim_{i\to\infty}} x_i) = 0$ and therefore $\lim_{i\to\infty} d(x_i, \overset{w}{\operatorname{exker}}(\mathfrak{T})) = 0$. The rest of the argument is identical to the corresponding part of the proof of the previous theorem. □

And again this results in a corollary for introversive functions:

Corollary 3.3.14:
Let A be a reflexive Banach space, $\mathfrak{T} : A \to \mathbb{R}$ be weakly lower semi-continuous and not weakly introversive, then there exists a sequence $(x_n)_{n\in\mathbb{N}}$ with $\overset{w}{\lim_{n\to\infty}}\mathfrak{T}(x_n) = 0$, that is not bounded.

3.4. Optimal Residual Algorithms

In the previous sections some basic principles for building optimal residual formulations where discussed. To that point, this was only concerned with developing an analytic formulation of how the solution of a (well posed) partial differential equation problem can be found.

As the examples in the introduction about the Galerkin methods suggest, discretization can be done by solving optimal residual formulations on finite dimensional approximation subspaces. Of course, for the discretization, the optimal residual formulation might already be modified to some similar problem more suitable for the selected approximation space.

The approach then results in the following fundamental problem of finding a solution for

$$\mathfrak{G}(x) \overset{!}{=} min \text{ for } x \in P$$

where P is the selected solution approximation subspace. Of course, for a proper optimal residual algorithm, it should be required, that solving the above problem for an increasing sequence of approximation subspaces P_n, that converges to the hole solution space, yields a sequence of solutions that converges to the solution of the optimal residual formulation over the whole solution space. That is, the sequence of solutions for each approximation subspace, converges to the solution of the associated partial differential problem.

Definition 3.4.1: Optimal Residual Algorithm

Let S be a solution space and $\mathfrak{G}$ be a (weak) residual gauge on S. An optimal residual algorithm $\mathfrak{A}$ for $\mathfrak{G}$ is any algorithm $\mathfrak{A}_i$ in a sequence $(\mathfrak{A}_n)_{n\in\mathbb{N}}$, called optimal residual algorithm sequence, where

- *$(e_i)_{i\in\mathbb{N}} \subset S$ is a complete sequence of functions of the solution space, generating the approximation spaces*

$$U^n := span(\{e_0, ..., e_{n-1}\}) \ .$$

- *each algorithm $\mathfrak{A}_n : (S \to \mathbb{R}^+) \times U^n \to \mathbb{R}^+$ for $n \in \mathbb{N}$ yields solutions $u_{n+1} \in U_{n+1}$ of equation 3.54 for a given start point $u_n \in U^n$, in short*

$$u_{n+1} = \mathfrak{A}_{n+1}(\mathfrak{G}, u_n) \ . \tag{3.53}$$

- *$\mathfrak{A}_{n+1}(\mathfrak{G}, u_n)$ is an approximate solution of the problem*

$$\mathfrak{G}(x) \stackrel{!}{=} min \text{ for } x \in U^{n+1} \tag{3.54}$$

using the start-point[9] u_n.

Having this definition we can proof that making, in a reasonable way, the approximating spaces bigger, e.g. making the mesh finer in a finite element space case, results in a sequence of solutions that will converge to the solution of the optimal residual formulation on the whole solution space. To catch the effect of the notion of "making approximations spaces bigger in a reasonable way" in a preciser way, we introduce the following term:

Definition 3.4.2: Optimizing Sequence of Algorithms

A sequence of algorithms $(\mathfrak{A}_n)_{n\in\mathbb{N}}$ is optimizing regarding to $\mathfrak{G}$, if

1. $\forall n \in \mathbb{N} : \forall u \in U^n : \exists \lambda_n(u) \in [0,1] : \mathfrak{G}(\mathfrak{A}_n(f,u)) = \lambda_n(u)\mathfrak{G}(u)$
2. *for every start point $u_1 \in U_1$, there exists a monotone increasing sequence of indices $(n_i)_{i\in\mathbb{N}}$ and a fixed $\lambda_{cap} \in [0,1)$ so that*

$$\prod_{j=n_i}^{n_{i+1}-1} \lambda_j(u_j) \leq \lambda_{cap} \ .$$

This definition can then be used to proof a quite general convergence result.

[9] This is for algorithms, that require start-points, otherwise u_n could be omitted.

Theorem 3.4.3: Optimal Residual Algorithm Convergence
Let $(u_n)_{n\in\mathbb{N}}$ be the result of an optimizing[10] , optimal residual algorithm sequence $(\mathfrak{A}_n)_{n\in\mathbb{N}}$ then

$$u_\infty := \overset{(w)}{\lim_{n\to\infty}} u_n \tag{3.55}$$

exists and is the global minimum of the associated residual gauge $\mathfrak{G}$, i.e.

$$\mathfrak{G}(u_\infty) = 0 \ . \tag{3.56}$$

Proof. By property 1 of the optimizing algorithms $\mathfrak{G}(u_{n+1}) \leq \mathfrak{G}(u_n)$ holds. Further, by definition, $\mathfrak{G}$ has the global lower boundary of 0. The sequence

$$r_n := \mathfrak{G}(u_n) \tag{3.57}$$

has therefore a lower boundary of 0 and is, since it is decreasing, convergent to a value r_∞. The subsequence $(r_{n_i})_{i\in\mathbb{N}}$, using the n_i from the property 2 of the optimizing algorithms definition 3.4.2 , does therefore converge to the same limit, i.e

$$\lim_{i\to\infty} r_{n_i} = r_\infty \ .$$

Further:

$$\begin{aligned}
r_{n_{i+1}} &= \mathfrak{G}(u_{n_{i+1}}) \\
&= \mathfrak{G}(\mathfrak{A}_{n_{i+1}}(\mathfrak{G}, u_{n_{i+1}-1})) \\
&= \lambda_{n_{i+1}}(u_{n_{i+1}-1})\mathfrak{G}(u_{n_{i+1}-1}) \\
&= \mathfrak{G}(u_{n_i}) \prod_{j=n_i}^{n_{i+1}-1} \lambda_j(u_j) \\
&= \mathfrak{G}(u_{n_{i-1}}) \prod_{j=n_{i-1}}^{n_i-1} \lambda_j(u_j) \prod_{j=n_i}^{n_{i+1}-1} \lambda_j(u_j) \\
&= \mathfrak{G}(u_1) \prod_{k=1}^{i} \left(\prod_{j=n_k}^{n_{k+1}-1} \lambda_j(u_j) \right) \\
&\leq \mathfrak{G}(u_1) \prod_{k=1}^{i} \lambda_{cap} \\
&= \mathfrak{G}(u_1)\, \lambda_{cap}^i \\
\Longrightarrow r_\infty &\leq 0 \\
\Longrightarrow r_\infty &= 0 \ .
\end{aligned}$$

Since $\mathfrak{G}$ is a (weak) residual gauge, this implies that the u_n converge (weakly) against a unique value u_∞ and $\mathfrak{G}(u_\infty) = 0$. u_∞ is therefore the global minimizer of $\mathfrak{G}$. □

[10] That is the sequence of algorithms of the optimal residual algorithm is optimizing.

The above theorem might seem a bit strange, because it is not required for the optimizing algorithms to actually find the optimum in each step. The rationale for that is the following: Initially the optimizing algorithms $\mathfrak{A}_n$ where indeed intended to yield global minima in each step. However, because of the few assured properties of a residual gauge, the $\mathfrak{A}_n$ can not be said to be optimization algorithms that find global minima, because regarding to $\mathfrak{G}$ restricted to an U_n a minimum might not exist. Further from a practical point of view, floating point numbers are usually not capable of representing exact solutions for most problems. The requirements for $(\mathfrak{A}_n)_{n\in\mathbb{N}}$ are therefore selected to guarantee sufficient greediness to enforce convergence. That is, property one guaranties that the algorithm sequence is greedy. This means the following: If the approximation space is expanded to a new approximation space, then the new approximation on the new space is at least as good as the old one. This is a reasonable assumption, since the old approximation space is contained in the new one. The second property ensures, that the algorithms are, at least, for the given $\mathfrak{G}$, greedy enough to yield convergent sequences of approximations.

It should be noted, that these requirements are very general. The approximation spaces are constructed from a complete sequence. There is specifically no orthogonality required. Even linear independence of elements of subsequences is not required. The only thing required is, that the sequence is complete, that is, the closure of its span is the complete solution space. Beside this requirement arbitrary functions can be used. However, this work will only consider polynomial finite element spaces without any additional functions for pathological case improvement.

3.5. Polynomial Structural Functions

For linear problems optimal residual algorithms do not seem to have any advantage over existing ones. They will reformulate a linear problem into a nonlinear one. However for the nonlinear case is not so obvious, that optimal residual formulations seems to be unpractical.

Polynomial structural functions have nice properties. Further they are common, since many nonlinearities of real world pdes are results from nonlinear material properties, which usually can be approximated by some high degree polynomial. The first of the interesting properties of polynomial structural functions is given by the following theorem:

Theorem 3.5.1:
Let $(\mathfrak{F}, \Gamma, \gamma)$ *be a well posed problem over a domain* Ω *for the partial differential equation*

$$F(\partial^k u(x), \partial^{k-1} u(x), ..., \partial u(x), u(x), x) = 0 \ ,$$

where F is a polynomial, $\mathfrak{F}$ is introversive, the boundary conditions are combinations of Dirichlet- and Neumann-conditions and $\partial\Omega$ be piecewise parametrized by polynomials over the standard simplex. Further, let P be a n-dimensional approximation space ($n < \infty$) that consists of polynomials. Let $(e_i)_{i\in\mathbb{N}}$ be a base of P, then there exists a residual gauge

$$\mathfrak{G}(u) := \|\mathfrak{F}(u)\|^2_{L^2(\Omega)} + \|\Gamma(u) - \gamma\|^2_{\partial,L^2} \ , \tag{3.58}$$

such that the mapping

$$\alpha \mapsto \mathfrak{G}\left(\sum_{i\in\underline{n}} \alpha_i e_i\right) \tag{3.59}$$

is a polynomial.

Proof. The residual gauge $\mathfrak{G}$ is one of the residual gauges that have been investigated before. The first term of equation 3.58 obeys, by definition of the norm, the following relation

$$\|\mathfrak{F}(u)\|^2_{L^2} = \int_\Omega \mathfrak{F}(u)^2(x)dV(x) = \int_\Omega F(\partial^k u(x), \partial^{k-1}u(x), ..., \partial u(x), u(x), x)^2 dV(x) \ .$$

Then, for all polynomials $u \in P$

$$F(\partial^k u(x), \partial^{k-1}u(x), ..., \partial u(x), u(x), x)^2$$

is a polynomial, since composition and derivation of polynomials yields polynomials.

This allows to compute the integral in $\|\mathfrak{F}(u)\|^2_{L^2}$ exact by well known quadrature formulas for a huge class of domains, including the domains, that can be parametrized by polynomials on the standard simplex. Tables for lower degree quadrature formulas can be found in many textbooks, for example Ern[9] et al. p. 359-361 or Zienkiewics[10] et al. p. 165-166[11]. A lot of papers show, that quadrature formulas for exact integration of arbitrary degree polynomials can often be constructed, see for example Rathod[11] et al. or Keast[12]. These quadrature formulas yield a finite number M of sample points x and weights ω so that

$$\sum_{j\in\underline{M}} \omega_j \mathfrak{F}(u)^2(x_j) = \int_\Omega \mathfrak{F}(u)^2(x)dV(x) \ . \tag{3.60}$$

In the left hand of the above equation $\mathfrak{F}$ is evaluated at fixed points. However, for a fixed x_j the mappings $\alpha \mapsto \partial^k \sum_{i\in\underline{n}} \alpha_i e_i(x_j)$ are polynomials. The mapping

$$\alpha \mapsto \sum_{j\in\underline{M}} \omega_j \mathfrak{F}(\sum_{i\in\underline{n}} \alpha_i e_i)^2(x_j) = \alpha \mapsto \|\mathfrak{F}(\sum_{i\in\underline{n}} \alpha_i e_i)\|^2_{L^2(\Omega)} \tag{3.61}$$

[11] One should be a little cautious with the later reference (i.e. Zienkiewics[10]). The weights in the table need to be multiplied with the volume of the cells over which the integration occurs. This is not directly obvious, when reading the surrounding text.

is a linear combination of polynomials, and therefore a polynomial itself. The second part of the argument is analog to the prior one. The mapping Γ is build "only" from Dirichlet- and Neumann boundary conditions which applied to polynomials yield polynomials. Then, since the boundary is parametrized by polynomials, the resulting integrals in the analog to equation 3.60 are integrals over polynomials. Which leads to the conclusion that $\mathfrak{G}$ is a sum of polynomials, and therefore a polynomial itself. □

While the above theorem itself is remarkable, approximating solutions with a global very high order degree polynomial is not very practical. However many finite element methods are using such approximation in every mesh cell - with low order polynomials. First I thought that I could proof introversiveness of $\mathfrak{F}$ for all polynomial structural functions. This however leaded to questions about the shape of the kernels of polynomials, i.e. varieties. However, after some time of studying the required theory of varieties in Cox[13] et al. without any results, I gave up on that for the time being.

However leaving the issue of introversiveness aside, the remaining question is how the above result, describing a residual gauge of one cell of a polynomial based finite element approximation, can be carried over to a whole finite element mesh. Before this can be done, finite element spaces have to be discussed.

3.6. Finite Element Spaces

To construct implementable algorithms with above concepts viable approximation spaces and residual gauges are needed. In this work only finite element spaces are considered due to time and space limitations, and because, as it was shown in the previous section, they reduce many nonlinear problems to problems regarding polynomials. Furthermore, to apply the concepts introduced so far without much problems, this section will define finite element spaces in a suitable way.

The basic idea of finite elements is to make a problem of functional analysis, as for example finding the root of some residual operator, computable by using solution approximations on a "meshed" domain.

Historically, finite elements where constructed for structural mechanic problems, where the shape of a finite element, more specifically its deformation under stress, was searched for. However in other finite element applications the shape of the geometries involved does not change. This caused some inconsistent naming conventions, where some times the functions describing the shape (resp. shape deformation) of the mesh cells are called the finite elements (for example in Zienkiewicz[10] et. al. p. 21), and sometimes base functions, that approximate a field on mesh cells, are called the finite elements (for example in Dobrowolski[3] p. 149). A very abstract definition of a finite element, that conceptually includes both approaches is given in Ern[9] et al. p. 19[12]. While the last abstract definition is very neat

[12]Where that definition follows a definition by P. Ciarlet.

and well suited to describe a wide range of algorithms, its abstractness hides some features that can be used for constructing optimal residual algorithms. Thus the following definitions came from a more constructive - implementation focused - mindset, and are my own combination of the above approaches:

Definition 3.6.1: Finite Element Space

A finite element space is a function space

$$E := \overline{\left\{ \sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{J_i}} w(\alpha, i, j)\, p_{ij} \;\middle|\; \alpha \in \mathbb{R}^{N_g} \right\}} \tag{3.62}$$

where:

1. $N \in \mathbb{N}$ *is the number of cells in the mesh.*
2. $(V_i)_{i \in \underline{N}}$ *is a family, called the mesh, of compact, connected, Lipschitz subsets* $V_i \subset \Omega$*, called cells (of the mesh), with*

$$\Omega = \bigcup_{i \in \underline{N}} V_i \text{ and } \forall i \neq j : \overset{\circ}{V}_i \cap \overset{\circ}{V}_j = \varnothing \ . \tag{3.63}$$

3. $J_i \in \mathbb{N}$ *the number of local ansatz functions (see below) in mesh cell* i*.*
4. $(p_{ij})_{ij \in \underline{N} \times \underline{J_i}}$ *is a family of functions, called local ansatz functions, where*

$$p_{ij} : V_i \longrightarrow \mathbb{R},\ j \in \underline{J_i} \ . \tag{3.64}$$

5. $N_g \in \mathbb{N}$ *is the number of global degrees of freedom.*
6. w *is a function*

$$w : \mathbb{R}^{N_g} \times \underline{N} \times \underline{J} \longrightarrow \mathbb{R} \tag{3.65}$$

called welding function, that will "weld" together local ansatz functions using global degrees of freedoms, where $J = max(\{J_i | i \in \underline{N}\})$*.*

Definition 3.6.2: Finite Element

Using the notations of a finite element space, the mapping

$$\alpha \mapsto \left(x \mapsto 1_{V_i}(x) \sum_{j \in \underline{J_i}} w(\alpha, i, j) p_{ij}(x) \right) \tag{3.66}$$

is called the finite element of (the mesh cell) V_i*.*

Definition 3.6.3: Local Degrees of Freedom

The results of the welding function w of a finite element space, are called local degrees of freedom. More precisely $w(\alpha, i, j)$ is the j-th local degree of freedom of cell i.

These definitions have been assembled with the application to optimal residual formulations in mind. There is no requirement of orthogonality (or even independence) regarding the local ansatz functions, since this is not in general necessary for optimal residual algorithms. This might be useful, when trying to extend a given finite element space by functions to handle pathological cases that the prior selection of local ansatz functions could not approximate well. Further, global properties, like global continuity, are dependent on the choice of the welding function, thus the above definition should be usable to include discontinuous Galerkin approaches.

Evaluation of the objective function, i.e. the residual gauge, is, for many optimization algorithms, the most expensive part. However, evaluation is embarrassingly parallel for optimal residual formulations on finite element spaces:

Conjecture 3.6.4:

Let $\mathfrak{F}$ be an introversive residual operator constructed from a structural function F. Let E be a finite element space over a domain Ω. Then the mapping

$$f := \alpha \mapsto \|\mathfrak{F}(\sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{J_i}} w(\alpha, i, j)\, p_{ij})\|^2_{L^2(\Omega)} \tag{3.67}$$

decomposes into an integration over the cells of the mesh, that is

$$f(\alpha) = \sum_{i \in \underline{N}} \|\mathfrak{F}(\sum_{j \in \underline{J_i}} w(\alpha, i, j)\, p_{ij})\|^2_{L^2(V_i)} \ . \tag{3.68}$$

Proof. (Well, almost a proof.) First, the residual operator should be investigated for any $u \in E$, with

$$u = \sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{J_i}} w(\alpha, i, j) p_{ij} \ . \tag{3.69}$$

This yields :

$$\mathfrak{F}(u) = x \mapsto F(..., \partial^l u(x), ...) = x \mapsto F\big(..., \sum_{i \in \underline{N}} \sum_{j \in \underline{J_i}} w(\alpha, i, j) \partial^l (1_{V_i} p_{ij}), ...\big) \ .$$

For points in the interior of a cell we get:

$$\begin{aligned}\forall x \in \overset{\circ}{V_h}: \qquad & F\big(..., \sum_{i \in \underline{N}} \sum_{j \in \underline{J_i}} w(\alpha, i, j) \partial^l (1_{V_i}(x) p_{ij}(x)), ...\big) \\ = \; & F\big(..., \sum_{j \in \underline{J_h}} w(\alpha, h, j) \partial^l p_{hj}(x), ...\big) \\ = \; & 1_{\overset{\circ}{V_h}}(x) F\big(..., \sum_{j \in \underline{J_h}} w(\alpha, h, j) \partial^l p_{hj}(x), ...\big)\end{aligned}$$

Further, let

$$\mathfrak{F}_i(u) := x \mapsto F\big(..., \sum_{j \in \underline{J_h}} w(\alpha, h, j) \partial^l p_{hj}(x), ...\big) \; . \tag{3.70}$$

Since boundaries are sets of measure 0 regarding the norm in $L^2(\Omega)$, we have with the above definitions[13]

$$\mathfrak{F}(u) \overset{L^2(\Omega)}{=} \sum_{i \in \underline{N}} 1_{V_i} \mathfrak{F}_i(u) \; . \tag{3.71}$$

Using this with the definition of the norm yields

$$\begin{aligned}\|\mathfrak{F}(u)\|^2_{L^2(\Omega)} &= \|\sum_{i \in \underline{N}} 1_{V_i} \mathfrak{F}_i(u)\|^2_{L^2(\Omega)} && (3.72)\\ &= \int_\Omega \sum_{i \in \underline{N}} 1_{V_i} \mathfrak{F}_i(u)^2 dV && (3.73)\\ &= \sum_{i \in \underline{N}} \int_\Omega 1_{V_i} \mathfrak{F}_i(u)^2 dV && (3.74)\\ &= \sum_{i \in \underline{N}} \int_{V_i} \mathfrak{F}_i(u)^2 dV && (3.75)\\ &= \sum_{i \in \underline{N}} \|\mathfrak{F}_i(u)\|^2_{L^2(V_i)} \; . && (3.76)\end{aligned}$$

□

The above conjecture combined with the conceptional flexibility of this approach had been the reason for me to start with this work.

There is however a subtle problem with the above proof. The omission of the behavior at cell boundaries makes the above a kind of a weak formulation. Indeed, if u is continuous but not differentiable at the cell boundaries, its first weak derivations will have jumps at the boundary. Thus any second weak derivation of u will result in a weak derivation of a jump and thus would yield a scaled delta distribution (see for example Reed[14] et al. p. 138). This will void the assumption that the boundaries can be neglected. This means, that in the case of residual

[13] This uses the assumption, that integrals of a set of measure 0 are zero. This is not necessary true for the used "functions". Derivations over jumps yield distribution like "functions". The implications are discussed below.

operators the integration of them over a boundary (i.e. a set of measure 0), will not necessarily yield 0.

Then naturally, if it is assumed anyway like above, that the integration over a boundary yields 0, the solution for a well posed problem is not anymore unique, since the partial differential equation is not forced to be realized on the cell boundaries. Therefore an optimal residual algorithm might converge to a non physical solution. As the numerical results in the last chapter show, this indeed happens. The standard way to avoid this, is to apply some sort of "entropy condition" to select the physically right solution. For optimal residual algorithms, this can be done, either by finding a suitable welding function w that will automatically guarantee realization of the entropy condition, or by using an addition to the constraint condition, in the residual gauge construction, that will enforce convergence to the entropy solution. Selecting a minimal energy condition as entropy condition works, for example, in simple cases.

However there is a lot more to be discussed before the actual numerical results can be meaningfully presented. A direct corollary conjecture of the above conjecture and of theorem 3.5.1 is that

Conjecture 3.6.5:

Let $\mathfrak{F}$ be a introversive residual operator constructed from a polynomial structural function F. Let E be a finite element space over a domain Ω with polynomial local ansatz functions. Then the mapping

$$f := \alpha \mapsto \|\mathfrak{F}(\sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{J_i}} w(\alpha, i, j)\, p_{ij})\|^2_{L^2(\Omega)} \tag{3.77}$$

is a polynomial.

Before actual results can be discussed, a practical pde needs to be explicitly written in an optimal residual formulation on a finite element mesh. However the definitions above are quite too general for that. Therefore some further definitions are introduced to make the task of implementation simpler:

Definition 3.6.6: Nodal Finite Element Space

A finite element space E is a nodal finite element space, if there is a family $(s_l)_{l \in \underline{N_g}} \subset \Omega$ of node vectors such that

$$\forall v \in E : \forall \alpha \in \mathbb{R}^{N_g} : \forall l \in \underline{N_g} : \alpha_l = v(s_l)\ . \tag{3.78}$$

For nodal finite element spaces, the degrees of freedom have a direct interpretation as the value of the finite element at some sample point called "node", for a

given array of degrees of freedom.

This definition does not specify how a node value is computed in a given cell. For example let $s_l \in V_i$ then

$$\forall \alpha \in \mathbb{R}^{N_g} : \forall v \in E : \alpha_l = v(s_l) = \sum_{j \in J_i} w(\alpha, i, j) p_{ij}(s_l) \ . \tag{3.79}$$

This is still not as simple as one would like for a "simple" implementation. Therefore one additionally requires that on each cell the local ansatz functions behave like Lagrange interpolation functions:

Definition 3.6.7: Lagrange Finite Element Space

Let E be a nodal finite element space, then E is a Lagrange finite element space, if

$$\forall \alpha \in \mathbb{R}^{N_g} : \forall (i,j) \in (I, J_i) : \exists ! \, l \in \underline{N_g} : \alpha_l = w(\alpha, i, j) \ . \tag{3.80}$$

The simplest way to implement a Lagrange finite element space is to reduce the welding function to a mapping on indices. That is in terms of a implementation, create an "array" $(l_{ij})_{(i,j) \in I \times J_i}$ such that:

$$\forall \alpha \in \mathbb{R}^{N_g} : \forall (i,j) \in (I, J_i) : w(\alpha, i, j) = \alpha_{l_{ij}} \tag{3.81}$$

There is a last commonly used simplification of finite element spaces. All J_i are chosen to have the same size, and the cells V_i can be parametrized by mapping from a standard cell V_{std}, for example the standard simplex, to V_i. This leads to the following definition:

Definition 3.6.8: Standard Finite Element Spaces

A finite element space E is a standard finite element space if there is a standard cell V_{std} along with homeomorphisms $L_i : V_{std} \to V_i \ \forall i \in \underline{N}$, a fixed number J of local degrees of freedom and local ansatz functions $p_j : V_{std} \to \mathbb{R}$ for all $j \in \underline{J}$, so that

$$\forall i \in \underline{N} : J_i = J \ \textit{and} \ \forall i \in \underline{N} : \forall j \in \underline{J} : p_{ij} = p_j \circ L_i^{-1} \ . \tag{3.82}$$

This is the amount of definitions regarding finite element spaces that are required for this work. To illustrate the usefulness of these definitions the following lemma for the numerical computation of a cell wise residual measure in spirit of conjecture 3.6.4 is given :

Lemma 3.6.9:
Let E be a standard Lagrange finite element space over a domain Ω with M local ansatz functions p_{ij} and mesh $(V_h)_{h\in\underline{N}}$. Further $\mathfrak{F}$ be a linear operator. Then with the notation

$$\phi_\lambda(\vec{x}) := \sum_{i=0}^{N-1} 1_{V_i}(\vec{x}) \sum_{j=0}^{M-1} \lambda_{(i*M+j)} p_{ij}(\vec{x}) \tag{3.83}$$

there exist a matrix $A \in \mathbb{R}^{(M*N)\times(N*M)}$ so that for all $\phi_\lambda \in E$ it holds that

$$\sum_{h\in\underline{N}} \|\mathfrak{F}(\phi_\lambda)\|^2_{L^2(\overset{\circ}{V}_h)} = \lambda^T A \lambda \tag{3.84}$$

where

$$(A)_{(i*M+j)(k*M+l)} = \delta_{ik} \int_{V_i} \mathfrak{F}(p_{ij})\mathfrak{F}(p_{il}) dV_x \tag{3.85}$$

for all $i,k \in \underline{N}$ and $j,l \in \underline{M}$

Proof.

$$\begin{aligned}
& \sum_{h\in\underline{N}} \|\mathfrak{F}(\phi_\lambda)\|^2_{L^2(\overset{\circ}{V}_h)} \\
= & \sum_{h\in\underline{N}} \|1_{\overset{\circ}{V}_h} \mathfrak{F}(\phi_\lambda)\|^2_{L^2(\overset{\circ}{V}_h)} \\
= & \sum_{h\in\underline{N}} < 1_{\overset{\circ}{V}_h} \mathfrak{F}(\phi), 1_{\overset{\circ}{V}_h} \mathfrak{F}(\phi) >_h \\
= & \sum_{h\in\underline{N}} < 1_{\overset{\circ}{V}_h} \mathfrak{F}(\sum_{i\in\underline{N}} 1_{V_i} \sum_{j\in\underline{M}} \lambda_{(i*M+j)} p_{ij}), 1_{\overset{\circ}{V}_h} \mathfrak{F}(\sum_{k\in\underline{N}} 1_{V_k} \sum_{l\in\underline{M}} \lambda_{(k*M+l)} p_{kl}) > \\
= & \sum_{h\in\underline{N}} < 1_{\overset{\circ}{V}_h} \mathfrak{F}(\sum_{j\in\underline{M}} \lambda_{(h*M+j)} p_{hj}), 1_{\overset{\circ}{V}_h} \mathfrak{F}(\sum_{l\in\underline{M}} \lambda_{(h*M+l)} p_{hl}) > \\
= & \sum_{h\in\underline{N}} \sum_{j\in\underline{M}} \sum_{k\in\underline{N}} \lambda_{(h*M+j)} \lambda_{(h*M+l)} < 1_{\overset{\circ}{V}_h} \mathfrak{F}(p_{hj}), 1_{\overset{\circ}{V}_h} \mathfrak{F}(p_{hl}) > \\
= & \sum_{h\in\underline{N}} \sum_{j\in\underline{M}} \sum_{k\in\underline{N}} \lambda_{(h*M+j)} \lambda_{(h*M+l)} \int_{V_h} \mathfrak{F}(p_{hj})\mathfrak{F}(p_{hl}) dV_x \\
= & \sum_{h\in\underline{N}} \sum_{j\in\underline{M}} \sum_{k\in\underline{N}} \sum_{l\in\underline{M}} \lambda_{(h*M+j)} \lambda_{(k*M+l)} \delta_{hk} \int_{V_h} \mathfrak{F}(p_{hj})\mathfrak{F}(p_{hl}) dV_x \\
= & \sum_{i\in\underline{N}} \sum_{j\in\underline{M}} \sum_{k\in\underline{N}} \sum_{l\in\underline{M}} \lambda_{(i*M+j)} \lambda_{(k*M+l)} (A)_{(i*M+j)(k*M+l)} \\
= & \sum_{m\in\underline{N*M}} \sum_{n\in\underline{N*M}} \lambda_m \lambda_n (A)_{mn} \\
= & \lambda^T A \lambda
\end{aligned}$$

□

The above lemma hides the welding function completely, by using only local degrees of freedoms denoted with λ. There are of course neat uses for full non-trivial welding functions. They can be used to construct finite element spaces that automatically realize boundary conditions. This will be discussed in the next two sections.

3.7. Welding in Standard Lagrange Finite Element Spaces

In the prior section, the welding function was given without much explanation. However to make use of this notion in applications some more details should be discussed. This is necessary to arrive at a simple implementation of welding functions. Further it is useful since welding functions might be chosen in a way, so that all elements of the finite element space, that was constructed with certain welding function, obey a given set of boundary conditions.

Since this section is application - or more precise - implementation focused, we only consider standard Lagrange finite element spaces, which are still general enough for many applications.

Lemma 3.7.1:
Let N_g be the number of global degrees of freedom of a standard Lagrange finite element space with N cells, where each cell has M local degrees of freedom. Further let w be the welding function of that space, and $\lambda \in \mathbb{R}^{N*M}$ the local degree of freedom vector, defined as

$$\lambda := \sum_{i \in \underline{N}} \sum_{j \in \underline{M}} \alpha_{l_{ij}} e_{(M*i+j)} := \sum_{i \in \underline{N}} \sum_{j \in \underline{M}} w(\alpha, i, j) e_{(M*i+j)} \ . \tag{3.86}$$

Then there exists a matrix $C \in \mathbb{R}^{(N*M) \times N_g}$, called connection matrix, with

$$\lambda = C\alpha \tag{3.87}$$

where

$$(C)_{hk} := \delta_{\hat{l}_h k} \tag{3.88}$$

$$\hat{l}_h := l_{\left(\frac{h-(h \bmod M)}{M}\right)(h \bmod M)} \ . \tag{3.89}$$

Proof. On a standard Lagrange finite element space each degree of freedom is bound to a single node. As it was noted right after the definition 3.6.7 the welding

functions can, in that case, be expressed by an array l_{ij} that maps local to global degrees of freedom. Therefore we have

$$\lambda_{ij} = w(\alpha, i, j) = \alpha_{l_{ij}} = \sum_{k=0}^{N_g-1} \delta_{k l_{ij}} \alpha_k \ . \tag{3.90}$$

Local degrees are identified by a multiindex, consisting of two indices, one for the cell and on for the ansatz function on the standard element, however, local degrees of freedom can canonically be enumerated by a single number via the relation

$$h = i * M + j \ . \tag{3.91}$$

This mapping can be inverted with

$$j = h \bmod M \tag{3.92}$$

$$i = \frac{h-j}{M} = \frac{h-(h \bmod M)}{M} \ . \tag{3.93}$$

Thus we have

$$\lambda_h = w\left(\alpha, \frac{h-(h \bmod M)}{M}\right)_{(h \bmod M)} = \alpha_{l_{\left(\frac{h-(h \bmod M)}{M}\right)(h \bmod M)}} \ . \tag{3.94}$$

This equation can of course be written as a matrix multiplication, with

$$\hat{l}_h := l_{\left(\frac{h-(h \bmod M)}{M}\right)(h \bmod M)} \tag{3.95}$$

$$\lambda_h = \sum_{k=0}^{N_g-1} \delta_{\hat{l}_h k} \alpha_k \tag{3.96}$$

where the matrix coefficients of the matrix C are obviously given by

$$(C)_{hk} = \delta_{\hat{l}_h k} \ . \tag{3.97}$$

□

This Lemma proofs, that the welding function can be implemented as a simple matrix, that has exactly one entry set to 1 in each row. However, there where not many restrictions imposed on the welding function, so the natural question arises, if it is possible to select welding functions in a way, that will have the result, that all elements of a finite element space automatically realize a given set of boundary conditions.

3.8. Boundary Conditions for Standard Lagrange Finite Element Spaces

In the previous chapters it has been shown, that solving an optimal residual formulation on a finite element space E will result in a problem like

$$\mathfrak{G}(u) := \|\mathfrak{F}(u)\|^2_{L^2(\Omega)} + \|\Gamma(u) - \gamma\|^2_{\partial} \stackrel{!}{=} min \quad \forall u \in E \ .$$

Since there is a lot of freedom to choose the welding function w of the finite element space E, it is natural to investigate whether there is a modified welding function $\tilde{w}$, that force all elements of the finite element space $\tilde{E}$ it generates to obey a given set of boundary conditions. I.e. find a welding function so that

$$\forall u \in \tilde{E} : \Gamma(u) = \gamma \ .$$

This would then simplify the evaluation of the residual gauge, since then

$$\forall u \in \tilde{E} : \mathfrak{G}(u) = \|\mathfrak{F}(u)\|^2_{L^2(\Omega)} + \|\Gamma(u) - \gamma\|^2_{\partial} = \|\mathfrak{F}(u)\|^2_{L^2(\Omega)} \ .$$

If the welding function remains simple, this should be speed up algorithms applied to optimal residual formulations. Due to time and space limitations for this work, the discussion will be restricted to Dirichlet and Neumann boundary conditions for a standard Lagrange finite element space E.

We will construct $\tilde{w}$ in two stages, first we will define operators, that will modify a given vector of global degrees of freedoms into a vector of global degrees of freedom that realize a given boundary condition, second we will use this operators to construct the needed welding function.

Constructing an operator that enforces Dirichlet conditions is trivial:

Corollary 3.8.1: Dirichlet Condition Enforcement

For a standard Lagrange finite element space E, let $\mathcal{D}$ be the set of Dirichlet degrees of freedom[14] with Dirichlet data $(d_n)_{n\in\mathcal{D}}$. Then the affine linear mapping

$$C_{\mathcal{D}} := \alpha \mapsto d_{\mathcal{D}} + B_{\mathcal{D}}\alpha \tag{3.98}$$

where

$$\begin{aligned} d_{\mathcal{D}} &\in \mathbb{R}^{N_g} \\ (d_{\mathcal{D}})_n &= \begin{cases} \tilde{d}_n & \text{if } n \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \\ B_{\mathcal{D}} &\in \mathbb{R}^{N_g \times N_g} \\ (B_{\mathcal{D}})_{ln} &= \begin{cases} 0 & \text{if } n \in \mathcal{D} \\ \delta_{ln} & \text{otherwise} \ . \end{cases} \end{aligned} \tag{3.99}$$

[14] Dirichlet degrees of freedoms are degrees of freedom that are in the Dirichlet part of the boundary of the domain over which the standard Lagrange finite element space is constructed.

The mapping $C_{\mathcal{D}}$ maps any input global degree of freedom vector α to an output global degree of freedom vector $C\alpha = d_{\mathcal{D}} + B_{\mathcal{D}}\alpha$ whose associated element in E obeys the Dirichlet (boundary) conditions.

The above corollary follows the following construction: First apply a linear mapping that sets all Dirichlet degrees of freedom to zero. Then add a vector that is the Dirichlet data on the Dirichlet degrees of freedom and 0 otherwise.

This operator is obviously cheap to compute, so its use should make algorithms faster. Sadly in the case of Neumann Conditions this is not so simple to figure out. Indeed one can proof the following lemma:

Lemma 3.8.2: Neumann Condition Enforcement

For a standard Lagrange finite element space E, let $\mathcal{N}$ be the set of Neumann degrees of freedom[15] *with data $(\mathfrak{d}_n)_{n\in\mathcal{N}}$. Then there exists a linear mapping $B_{\mathcal{N}}^{-1}$ such that the affine linear mapping*

$$C_{\mathcal{N}} := \alpha \mapsto d_{\mathcal{N}} + B_{\mathcal{D}}\alpha \tag{3.100}$$

maps any input global degree of freedom vector α to an output global degree of freedom vector $C\alpha = d_{\mathcal{D}} + B_{\mathcal{D}}\alpha$ whose associated element in E obeys the Neumann (boundary) conditions.

Proof. The Neumann boundary condition is given by

$$\frac{\partial \phi_\alpha}{\partial \mathfrak{n}_n}(\vec{v}_n) = \mathfrak{d}_n \tag{3.101}$$

where $\frac{\partial \phi_\alpha}{\partial \mathfrak{n}_n}$ denotes the normal dSerivation for boundary node $\vec{v}_n$. Let $\mathrm{nc}(n, \mathfrak{n})$ denote the index of the cell adjacent to $\vec{v}_n$, such that $\exists \delta > 0 : \forall \epsilon < \delta : (\vec{v}_n + \epsilon\mathfrak{n}) \in cellvol(nc(n, \mathfrak{n}))$ Further as a short notation we will write $i_n := \mathrm{nc}(n, -\mathfrak{n}_n)$ and let $\phi \in E$ denote an element of E that is constructed by the degree of freedom array α. Last let $\phi_i = 1_{V_i}(x) \sum_{j\in\underline{M}} \alpha_{l_{ij}} p_{ij}(x)$ be the finite element in E over cell i. Then we can

[15] That is, degrees of freedom that are in the Neumann part of the boundary of the domain over which the standard Lagrange finite element space is constructed.

derive

$$\begin{aligned}
\mathfrak{d}_n &= \left(\frac{\partial}{\partial \mathfrak{n}_n}\phi\right)(\vec{v}_n) \\
&= \lim_{\epsilon\to 0}\frac{\phi(\vec{v}_n) - \phi_{\mathrm{nc}(n,-\mathfrak{n}_n)}(\alpha)(\vec{v}_n - \epsilon\mathfrak{n}_n)}{\epsilon} \\
&= \left(\frac{\partial}{\partial \mathfrak{n}_n}\phi_{\mathrm{nc}(n,-\mathfrak{n}_n)}(\alpha)\right)(\vec{v}_n) \\
&= \left(\frac{\partial}{\partial \mathfrak{n}_n}\sum_{j=0}^{M-1}\alpha_{l_{in}j}p_{in}j\right)(\vec{v}_n) \\
&= \sum_{j=0}^{M-1}\alpha_{l_{in}j} < grad(p_{in}j)(\vec{v}_n), \mathfrak{n}_n >
\end{aligned}$$

This is a linear relation between the degrees of freedom and the nodes, therefore an affine linear mapping as stated exists. □

With these results it is possible to construct a new welding function, that will cause obedience of boundary conditions for all elements in the derived finite element space. From the above constructions, it is clear, that the boundary mappings $C_{\mathcal{D}}$ and $C_{\mathcal{N}}$ have the property that:

$$\forall n \in \mathcal{D} : \frac{\partial}{\partial e_n}C_{\mathcal{D}} = 0 \tag{3.102}$$

$$\forall n \in \mathcal{N} : \frac{\partial}{\partial e_n}C_{\mathcal{N}} = 0 \tag{3.103}$$

That is, both mappings are constant for the degrees of freedom on the boundaries they are defined for. The degrees of freedom on these boundaries can therefore be removed from the formulation. The combination of the above results yields the following theorem.

Theorem 3.8.3: Enforcement of Boundary Conditions

Let E be a standard Lagrange finite element space with welding function w. Further, let $\mathcal{D}$ be the set of Dirichlet degrees of freedom with Dirichlet data $(d_n)_{n\in\mathcal{D}}$ and $\mathcal{N}$ be the set of Neumann degrees of freedom with Neumann data $(\mathfrak{d}_n)_{n\in\mathcal{N}}$. Then there exists an affine linear mapping

$$\alpha \mapsto \lambda^0 + P\alpha \tag{3.104}$$

such that all functions in the standard Lagrange finite element space $\tilde{E}$, with

$$N_{g0} = N_g - \#(\mathcal{D}\cup\mathcal{N}) \tag{3.105}$$

global degrees of freedom constructed by the welding function

$$\tilde{w} \quad : \quad \mathbb{R}^{N_{g0}} \times \underline{N} \times \underline{J} \to \mathbb{R} \tag{3.106}$$

$$\tilde{w}(\alpha, i, j) \quad := \quad \left(\lambda^0 + P\alpha\right)_{(i*M+j)} \ , \tag{3.107}$$

obey the boundary conditions.

Proof. Corollary 3.8.1 and lemma 3.8.2 imply the equations 3.102 and 3.103 that implicitly states, that the result of the enforcement of boundary conditions will not depend on the input global degrees of freedoms, that are members of a Neumann or Dirichlet boundary. Thus the degrees of freedom that are in the Neumann or Dirichlet boundaries can be removed from the degrees of freedom that will construct the new finite element space. This deletion of the boundary conditions can be done by the linear mapping given by a matrix

$$S \in \mathbb{R}^{N_g \times N_{g0}} \ ,$$

that maps N_{g0} degrees of freedom to the degrees of freedom not in $\mathcal{D} \cup \mathcal{N}$, and sets the degrees of freedom in $\mathcal{D} \cup \mathcal{N}$ to 0. Now there are two ways of enforcing both conditions, using lemma 3.7.1, corollary 3.8.1 and lemma 3.8.2, to construct local degrees of freedom arrays, whose associated approximation functions will realize the boundary conditions:

$$\lambda_\alpha = (C \circ C_{\mathcal{D}} \circ C_{\mathcal{N}} \circ S)(\alpha)$$
$$\lambda_\alpha = (C \circ C_{\mathcal{N}} \circ C_{\mathcal{D}} \circ S)(\alpha)$$

If $\mathcal{D} \cap \mathcal{N} = \varnothing$ these two constructions are equivalent. W.l.o.g. we investigate the first construction, where

$$\begin{aligned} C \circ C_{\mathcal{D}} \circ C_{\mathcal{N}} \circ S &= C \circ C_{\mathcal{D}} \circ (d_{\mathcal{N}} + B_{\mathcal{N}}) \circ S \\ &= (C \circ C_{\mathcal{D}})(d_{\mathcal{N}}) + C \circ C_{\mathcal{D}} \circ B_{\mathcal{N}} \circ S \\ &= (C \circ (d_{\mathcal{D}} + B_{\mathcal{D}}))(d_{\mathcal{N}}) + C \circ (d_{\mathcal{D}} + B_{\mathcal{D}}) \circ B_{\mathcal{N}} \circ S \\ &= (2C(d_{\mathcal{D}}) + (C \circ B_{\mathcal{D}})(d_{\mathcal{N}})) + C \circ B_{\mathcal{D}} \circ B_{\mathcal{N}} \circ S \ . \end{aligned}$$

In the above equations matrices were treated as mappings. If we introduce the following abbreviations

$$\lambda^0 := 2Cd_{\mathcal{D}} + CB_{\mathcal{D}}d_{\mathcal{N}}$$
$$P := CB_{\mathcal{D}}B_{\mathcal{N}}S$$

with standard matrix notations this yields the equation

$$\lambda_\alpha = \lambda^0 + P\alpha$$

that induces the welding function $\tilde{w}$, by

$$\tilde{w}(\alpha, i, j) \coloneqq (\lambda_\alpha)_{(i*M+j)} \ .$$

Then by using the mesh of E and the local ansatz functions of E the welding function $\tilde{w}$ generates a finite element space $\tilde{E}$ whose elements realize the boundary conditions. □

The above theorem might not be very useful in all cases, since obtaining the actual matrices for the enforcement of the Neumann boundary conditions is not trivial. Therefore it is reasonable to investigate the case where only Dirichlet conditions are given. In that case we have, using the notations of the previous proof,

$$\begin{aligned} \lambda = CC_{\mathcal{D}}S\alpha &= C(d_{\mathcal{D}} + B_{\mathcal{D}})S\alpha \\ &= Cd_{\mathcal{N}} + CB_{\mathcal{D}}S\alpha \end{aligned}$$

and therefore:

$$\begin{aligned} \lambda^0 &\coloneqq Cd_{\mathcal{D}} \\ P &\coloneqq CB_{\mathcal{D}}S \ . \end{aligned} \tag{3.108}$$

Before this section ends a lemma is presented to show the practical usefulness of the introduced concepts. Assume, for example that, in the spirit of conjecture 3.6.4 and lemma 3.6.9, one tries to solve an optimal residual problem by finding a solution to

$$f(\alpha) \coloneqq \sum_{h \in \underline{N}} \|\mathfrak{F}(\phi_{\lambda_\alpha})\|^2_{L^2(\overset{\circ}{V}_h)} = \lambda_\alpha^T A \lambda_\alpha \overset{!}{=} min \text{ for } \phi_{\lambda_\alpha} \in E \tag{3.109}$$

for some finite element space E, that enforces boundary conditions in the sense of theorem 3.8.3, i.e. the local degrees of freedom obey the condition

$$\lambda_\alpha = \lambda^0 + P\alpha \ . \tag{3.110}$$

Then the following lemma transforms the above optimization problem into the problem of linear algebra, that is, into the problem of solving a linear equation.

Lemma 3.8.4:
Let, in the spirit of the text above, $A \in \mathbb{R}^{(N*M)\times(N*M)}$, $P \in \mathbb{R}^{(N*M)\times N_{g0}}$, $\lambda^0 \in \mathbb{R}^{N*M}$ *and*

$$f(\alpha) \coloneqq (\lambda^0 + P\alpha)^T A(\lambda^0 + P\alpha) \tag{3.111}$$

where f can only yield non negative numbers. Then the necessary and sufficient condition for the solution α of the problem

$$f(\alpha) \stackrel{!}{=} min \quad \forall \alpha \in \mathbb{R}^{N_{g0}} \tag{3.112}$$

is

$$P^T A P \alpha = -P^T A \lambda^0 \ . \tag{3.113}$$

Proof. The mapping f is a non negative quadric therefore

$$grad(f) = 0$$

is a sufficient and necessary condition for optimality. To compute this condition, the computation of the partial derivations of f are required:

$$\begin{aligned}
& \frac{\partial}{\partial \alpha_n}(\lambda^0 + P\alpha)^T A(\lambda^0 + P\alpha) \\
= \ & \frac{\partial}{\partial \alpha_n}(\lambda^0 + P\alpha)^T (\sum_i (\sum_j (A)_{ij}(\lambda^0_j + \sum_k (P)_{jk}\alpha_k))e_i) \\
= \ & \frac{\partial}{\partial \alpha_n} \sum_i [((\lambda^0)_i + \sum_m (P)_{im}\alpha_m)(\sum_j (A)_{ij}(\lambda^0_j + (P)_{jk}\alpha_k))] \\
= \ & \sum_i [(P)_{in}(\sum_j (A)_{ij}(\lambda^0_j + \sum_k (P)_{jk}\alpha_k)) \\
& \quad + ((\lambda^0)_i + \sum_m (P)_{im}\alpha_m)(\sum_j (A)_{ij}((P)_{jn})] \\
= \ & [P^T A(\lambda^0 + P\alpha)]_n + \sum_i [(\lambda^0 + P\alpha)_i (AP)_{in}] \\
= \ & [P^T A(\lambda^0 + P\alpha)]_n + [(AP)^T(\lambda^0 + P\alpha)]_n \\
= \ & [P^T A(\lambda^0 + P\alpha) + P^T A^T(\lambda^0 + P\alpha)]_n \\
= \ & 2[P^T A(\lambda^0 + P\alpha)]_n
\end{aligned}$$

Thus

$$grad(f)(\alpha) = 2P^T A(\lambda^0 + P\alpha)$$

and therefore

$$\begin{aligned}
& grad(f)(\alpha) = 0 \\
\Leftrightarrow \ & 2P^T A(\lambda^0 + P\alpha) = 0 \\
\Leftrightarrow \ & 2P^T A\lambda^0 + 2P^T AP\alpha = 0 \\
\Leftrightarrow \ & P^T AP\alpha = -P^T A\lambda^0
\end{aligned}$$

□

With the mathematical framework so far, it is possible to actually examine concrete optimal residual algorithms, which will be done in the next chapter. However, the actual optimal residual algorithms investigated will use some further properties to define embarrassingly parallel steps. These properties will be investigated in the following last section of this chapter.

3.9. Finite Elements and Relations Between Degrees of Freedom

When developing parallel algorithms it is crucial to identify computationally independent parts of a problem. Finite element meshes have the innate property that their degrees of freedom only describe well localized aspects of an approximation function. To be able to use this property for parallel algorithm construction, this section elaborates on the question of what effect a change in a specific degree of freedom has for elements of a finite element space. Much aspects of the question get clearer, when using some carefully crafted definitions:

Definition 3.9.1: Effective Degree of Freedom

Let u_i be a finite element on cell V_i of a finite element space E and N_g be the number of global degrees of freedom of E. Then $j \in \underline{N_g}$ is an effective degree of freedom of element i if

$$\exists \alpha \in \mathbb{R}^{N_g} : \exists h \in \mathbb{R} : u_i(\alpha) \neq u_i(\alpha + he_j) \ . \tag{3.114}$$

The set of all effective degrees of freedom element i is denoted $\mathit{eff}(i)$.

Definition 3.9.2: Ineffective Degree of Freedom

Let u_i be a finite element on cell V_i of a finite element space E and N_g be the number of global degrees of freedom of E. Then $j \in \underline{N_g}$ is an ineffective degree of freedom of element i if

$$\forall \alpha \in \mathbb{R}^{N_g} : \forall h \in \mathbb{R} : u_i(\alpha) = u_i(\alpha + he_j) \ . \tag{3.115}$$

The set of all ineffective degrees of freedoms of element i is denoted $\mathit{ineff}(i)$.

To understand these definitions it is crucial to remember definition 3.6.2 of a finite element. Finite elements are defined as mappings from a global degree of freedom array to some approximation function with a support only on a specific mesh cell. With the above definitions we have the simple corollary that:

Corollary 3.9.3:

$$\mathit{eff}(i) = \underline{N_g} \cap \mathit{ineff}(i) = \mathit{ineff}(i)^c \tag{3.116}$$

Further it will be necessary to investigate cells affected by a given degree of freedom, which motivates the following definition:

Definition 3.9.4: Affected Cells

Let $(u_i)_{i\in\underline{N}}$ be the family of finite elements of a finite element space E. Then the affected cells of degree of freedom j are defined as

$$ac(j) := \{i | j \in \mathit{eff}(i)\} \tag{3.117}$$

With the above definition we get the following duality:

Corollary 3.9.5:

$$i \in ac(j) \quad \Leftrightarrow \quad j \in \mathit{eff}(i) \tag{3.118}$$
$$i \notin ac(j) \quad \Leftrightarrow \quad j \in \mathit{ineff}(i) \tag{3.119}$$

These concepts allow us to define a notion of independence for degrees of freedom:

Definition 3.9.6: Independent Degrees of Freedom

Let $(u_i)_{i\in\underline{N}}$ be the family of finite elements of a finite element space E. Then j is independent of k, if[16]

$$ac(j) \cap ac(k) = \varnothing \ . \tag{3.120}$$

This relation is written as

$$j \parallel k \ . \tag{3.121}$$

The negation of the relation is written with the symbol $\nparallel$.

[16] Well, this definition might need to be revised to handle aspects at cell boundaries better, in the sense that degrees of freedoms are independent if the affected cells are disjoint and have no cell face in common, i.e. if $\overline{\mathit{volume}(\mathrm{ac}(j))} \cap \overline{\mathit{volume}(\mathrm{ac}(k))} = \varnothing$.

The notion of independence comes from the observation, that independent degrees of freedom have no affected cells in common. This results in a computational independence when computationally calculating the effects of changing independent degrees of freedoms for a finite element approximation function. In short, computing changes on independent degrees of freedom is parallel, and thus the following definition is of much interest when constructing parallel algorithms:

Definition 3.9.7: Decomposition into Sets of independend degrees of freedom

For a given finite element space E let $\underline{N_g}$ denote the set of all degrees of freedom. Then a finite family $(K_v)_{v \in \underline{N_\cap}}$, where $K_v \subset \underline{N_g}$ for all $v \in \underline{N_\cap}$ and $N_\cap \in \mathbb{N}$, is a decomposition into independent sets of degrees of freedom, if

1. $\forall v \in \underline{N_\cap} : \ (\ \forall j,k \in K_v : \ j \parallel k \ \Leftrightarrow \ j \neq k \ )$
2. $\forall v,u \in \underline{N_\cap} : \ (\ K_v \cap K_u = \varnothing \ \Leftrightarrow \ u \neq v \ )$
3. $\bigcup_{v \in \underline{N_\cap}} K_v = \underline{N_g}$

4. The Scalar Conservation Law in Optimal Residual Formulation

> The struggle itself towards the heights is enough to fill a man's heart. One must imagine Sisyphus happy.
>
> *Albert Camus*

The central motivation for the development of the theory of optimal residual algorithms was, that it seemed due to conjecture 3.6.4, that the L_2-norm in finite element solution spaces can be computed embarrassingly parallel for a lot of residual operators of real world non linear problems. The concepts of the previous chapter are here investigated in depth for the (simplified) scalar conservation law[1]

$$div(\kappa(grad(\phi))grad(\phi)) = 0 \tag{4.1}$$

to make a test implementation possible. The scalar conservation law is a nonlinear partial differential equation and has many applications, for example in stationary current problems and electrostatic problems. In the following section we will explicitly derive the equations, that are needed to implement optimal residual algorithms.

4.1. Residual Gauges for the Scalar Conservation Law

To implement an optimal residual algorithm, a residual gauge for the scalar conservation law is needed. The evaluation of this gauge should decompose into computationally independent integrations as guessed in conjecture 3.6.4. There are some subtle details in constructing general solutions for scalar conservation laws. To keep things short, despite of that, a simple kind of piecewise solution will be defined:

[1] Well, to be precise, this is a very special form of the scalar conservation law, that is usually written $\frac{\partial \rho}{\partial t} + divJ = 0$. Here it is assumed, that the flow J can be represented by a gradient field, i.e. $J = -\kappa\, grad(\phi)$ and the problem is static. The equation above could also be called nonlinear Poisson equation and still has many practical applications, however, the term nonlinear Poisson equation is used for an other equation in Evans[1] p.5.

Definition 4.1.1: Scalar Conservation Law Mesh Solution
Let

$$\mathfrak{F} := \phi \mapsto (x \mapsto div(\kappa(grad(\phi))grad(\phi))(x)) \tag{4.2}$$

and for any given finite element space mesh $(V_i)_{i\in\mathbb{N}}$ *of* Ω *let*

$$F_{kh} := V_k \cap V_h \text{ for } k \neq h \tag{4.3}$$

be the face[2] *connecting cell* k *with cell* h*, where an implicit orientation is given by the normal vector on* F_{kh} *pointing from cell* k *into cell* h*. The flow* I_{kh} *from cell* k *through face* F_{kh} *is defined as*

$$I_{kh}(\phi) := \int_{F_{kh}} 1_{V_k}\kappa(grad(\phi))grad(\phi)\cdot d\vec{A} \ . \tag{4.4}$$

Then ϕ *is a mesh solution if*

$$\forall i \in \underline{N} : \mathfrak{F}|_{V_i}(\phi) = 0 \tag{4.5}$$

$$\text{and} \quad \forall (k,h) \in \underline{N}\times\underline{N} \text{ with } k \neq h : I_{kh}(\phi) + I_{hk}(\phi) = 0 \ . \tag{4.6}$$

For this notion of a solution we need to construct a residual gauge:

Conjecture 4.1.2: Scalar Conservation Law Mesh Solution Residual Gauge
Let $\mathfrak{F}$ *the classical residual operator of the scalar conservation law, and assume that the boundary condition* $\Gamma(\cdot) = \gamma$ *in conjunction with the constraint* $\mathfrak{C}(\cdot) = 0$ *gives a well posed problem regarding the mesh solution, then*

$$\mathfrak{G} := \phi \mapsto \left(\sum_{i\in\underline{N}} ||1_{A_i}\mathfrak{F}(\phi)||^2\right) + \left(I_{ik}(\phi) + I_{ki}(\phi)\right)^2 + ||\Gamma(\phi)-\gamma||_\partial^2 + ||\mathfrak{C}(\phi)||^2 \ . \tag{4.7}$$

This a conjecture, since proving this would require, to proof introversiveness for the general nonlinear conservation law, which I assume is true, at least for many cases, but did not proof. For the linear case this construction is just a linear combination of residual gauges on each mesh cell with patching conditions, and should not be problematic. However, it will be left open, if there are indeed well

[2] This is a very awkward definition of faces, since it will include empty sets, points and lines as what should be two dimensional faces. These non 2D-faces are formalization artefact's and are not meant to be relevant. Besides of being insulting to the mathematicians perception of a beautiful definition, these artefact's do not yield practical problems, since empty sets, points or 1D-lines, have a surface measure of 0 and thus the corresponding flow integrals will yield 0. In short this definition should be revised and not used in later works.

posed problems regarding the above mesh solution concept. I will, with regard to numerical testing optimal residual problems, assume, that this is the case, and proceed with a description of how the several parts of the above residual gauge can be computed in an implementation.

Lemma 4.1.3:
Let E be a standard Lagrange finite element space with N cells, M local ansatz functions on the standard element and mesh $(V_i)_{i\in\mathbb{N}}$, then, with the notations of definition 4.1.1, for any ϕ_λ where λ is a local degree of freedom array, so that

$$\phi_\lambda = \sum_{i\in\underline{N}} 1_{V_i} \sum_{j\in\underline{M}} \lambda_{(i*M+j)} p_{ij} \tag{4.8}$$

then, for a cell wise constant kappa, i.e.

$$\kappa = \sum_{i\in\underline{N}} 1_{V_i}\kappa_i \ \ (\kappa_i \in \mathbb{R} \ \forall i \in \underline{N}) \tag{4.9}$$

the condition for flow conservation across cell boundaries can be expressed as

$$(I_{kh}(\phi_\lambda) + I_{hk}(\phi_\lambda))^2 = \lambda^T \left(w_{kh} w_{kh}^T\right) \lambda \tag{4.10}$$

where for all $i \in \underline{N}$ and $j \in \underline{M}$

$$(w_{kh})_{(i*M+j)} := \delta_{ki} \int_{F_{kh}} \kappa_k grad(p_{kj}) \cdot d\vec{A} + \delta_{hi} \int_{F_{hk}} \kappa_h grad(p_{hj}) \cdot d\vec{A} \tag{4.11}$$

Proof. First note that

$$\begin{aligned}
I_{kh}(\phi_\lambda) &:= \int_{F_{kh}} \kappa_k grad(\phi_\lambda) \cdot d\vec{A} \\
&= \int_{F_{kh}} \kappa_k grad(\sum_{i\in\underline{N}} 1_{V_i} \sum_{j\in\underline{M}} \lambda_{(i*M+j)} p_{ij}) \cdot d\vec{A} \\
&= \int_{F_{kh}} \kappa_k grad(\sum_{j\in\underline{M}} \lambda_{(k*M+j)} p_{kj}) \cdot d\vec{A} \\
&= \sum_{j=0}^{M-1} \lambda_{(k*M+j)} \int_{F_{kh}} \kappa_k grad(p_{kj}) \cdot d\vec{A} \ .
\end{aligned}$$

With this we have

$$\begin{aligned}
& I_{kh}(\phi_\lambda) + I_{hk}(\phi_\lambda) \\
= & \sum_{j\in\underline{M}} \lambda_{(k*M+j)} \int_{F_{kh}} \kappa_k grad(p_{kj}) \cdot d\vec{A} + \sum_{j\in\underline{M}} \lambda_{(h*M+j)} \int_{F_{hk}} \kappa_h grad(p_{hj}) \cdot d\vec{A} \\
= & \sum_{i\in\underline{N}} \sum_{j\in\underline{M}} \lambda_{(i*M+j)} \left(\delta_{ki} \int_{F_{kh}} \kappa_k grad(p_{kj}) \cdot d\vec{A} + \delta_{hi} \int_{F_{hk}} \kappa_h grad(p_{hj}) \cdot d\vec{A}\right) \\
= & \lambda^T w_{kh} \ .
\end{aligned}$$

Finally

$$\left(I_{kh}(\phi_\lambda) + I_{hk}(\phi_\lambda)\right)^2 = \lambda^T w_{kh} \lambda^T w_{kh} = \lambda^T \left(w_{kh} w_{kh}^T\right) \lambda \ .$$

□

Welding functions yield local degree of freedom arrays. The above result is expressed using only local degrees of freedom, thus the above result is easily combined with any kind of welding function. Most notably, this approach can simply be used with the enforcement of boundary conditions, that has been discussed in the mathematical framework chapter. In the next section it will be shown, how the residual operator on each cell can be computed. Regarding to a fast implementation there is, however, a problem: The enforcement of Neumann conditions is complicated, thus, instead of implementing this an additional constraint term is introduced to find, what is in many applications, a minimal energy solution.

Definition 4.1.4: Conservation Law Cell Energy
The abstract[3] *conservation law cell energy on cell V_i is given by*

$$E_k(\phi) := \int_{V_k} \kappa(grad(\phi)) \left(grad(\phi)\right)^2 dV \tag{4.12}$$

With this we get the following result for a cell-wise constant κ:

Lemma 4.1.5:
With the notation of lemma 4.1.3 the energy is given by

$$E(\phi_\lambda) := \sum_{k \in \underline{N}} E_k(\phi_\lambda) = \lambda^T A_E \lambda \tag{4.13}$$

where for all $k, i \in \underline{N}$ and $j, h \in \underline{M}$

$$(A_E)_{(k*M+j)(i*M+h)} = \delta_{ki} \int_{V_k} \kappa_k grad(p_{kj}) \cdot grad(p_{kh}) dV \ . \tag{4.14}$$

[3]The defined quantity is proportional to the real energy in a cell in many applications.

Proof.

$$\begin{aligned}
E(\phi_\lambda) &= \sum_{k\in\underline{N}} E_k(\phi_\lambda) \\
&= \sum_{k\in\underline{N}} \int_{V_k} \kappa_k \left(grad(\phi)\right)^2 dV \\
&= \sum_{k\in\underline{N}} \int_{V_k} \kappa_k \left(grad(\sum_{i\in\underline{N}} 1_{V_i} \sum_{j\in\underline{M}} \lambda_{(i*M+j)} p_{ij}) \right)^2 dV \\
&= \sum_{k\in\underline{N}} \int_{V_k} \kappa_k \left(grad(\sum_{j\in\underline{M}} \lambda_{(k*M+j)} p_{kj}) \right)^2 dV \\
&= \sum_{k\in\underline{N}} \int_{V_k} \kappa_k grad(\sum_{j\in\underline{M}} \lambda_{(k*M+j)} p_{kj}) \cdot grad(\sum_{h\in\underline{M}} \lambda_{(k*M+h)} p_{kh}) dV \\
&= \sum_{k\in\underline{N}} \sum_{j\in\underline{M}} \sum_{i\in\underline{N}} \sum_{h\in\underline{M}} \lambda_{(k*M+j)} \lambda_{(i*M+h)} \delta_{ki} \int_{V_k} \kappa_k grad(p_{kj}) \cdot grad(p_{kh}) dV \\
&= \lambda^T A_E \lambda
\end{aligned}$$

□

With these results there are two possible and very simple candidates for residual gauges for testing optimal residual algorithms with the scalar conservation law. First, simplicity can be attained by only allowing Dirichlet boundary conditions on some boundary areas while leaving everything else on the boundary open and search for a minimal energy solution. Then a simple residual gauge should, for a finite element space that enforces Dirichlet boundary conditions, be given, with the notations above, by

$$\mathfrak{G}(\phi_\lambda) := \left(\sum_{i\in\underline{N}} \|1_{V_i} \mathfrak{F}(\phi_\lambda)\|^2 \right) + \left(\sum_{k,h\in\underline{N}} \lambda^T \left(w_{kh} w_{kh}^T \right) \lambda \right) + \lambda^T A_E \lambda \ . \tag{4.15}$$

The second possible gauge, comes from the observation that many physical configurations are already determined by having a minimal energy from all possible solutions that realize the boundary conditions. Thus, when a Dirichlet boundary condition enforcing finite element space is used, a very simple plausible residual gauge could be

$$\mathfrak{G}(\phi_\lambda) := \left(\sum_{i\in\underline{N}} \|1_{V_i} \mathfrak{F}(\phi_\lambda)\|^2 \right) + \lambda^T A_E \lambda \ . \tag{4.16}$$

As it will be shown later, the alleged gauge in equation 4.16 gives correct results in some cases, while in other cases this might not be true. However due to time and space limitations, no rigor construction of a residual gauge for the scalar conservation law has been performed. In any case however, using the above alleged gauges for testing the viability of optimal residual algorithms, requires the numerical evaluation of $\|1_{V_i} \mathfrak{F}(\phi_\lambda)\|^2$. This will be discussed in the next section.

4.2. Evaluation of the Residual Operator

Regarding to our prior definitions the structural operator of the scalar conservation law is

$$\mathfrak{F} = \phi \mapsto (\vec{x} \mapsto div(\kappa(grad(\phi))grad(\phi))(\vec{x})) \ . \tag{4.17}$$

To actually implement a residual gauge it is necessary to know how to calculate $\|1_{A_i}\mathfrak{F}(\phi)\|^2$ for any ϕ in a given finite element space.

Like in the above cases we restrict this part of the work to standard Lagrange finite element spaces, to make this simpler some abbreviations are used and only the L^2-norm is considered.

Definition 4.2.1: Standard Lagrange Finite Element Space Abbreviations

Let E be a standard Lagrange finite element space, then

$$\phi_\alpha \in E \tag{4.18}$$

denotes the element of E, that is constructed by the global degrees of freedom array α, i.e. with the notations from definition 3.6.1, 3.6.7 and 3.6.8

$$\begin{aligned} \phi_\alpha(\vec{x}) &:= \sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{M}} \alpha_{l_{ij}} \ p_j \circ L_i^{-1} && (4.19)\\ &= \sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{M}} \alpha_{l_{ij}} \ p_{ij} && (4.20)\\ &= \sum_{i \in \underline{N}} 1_{V_i} \phi_{\alpha i}(\vec{x}), && (4.21) \end{aligned}$$

where

$$\begin{aligned} p_{ij} &:= p_j \circ L_i^{-1} && (4.22)\\ \phi_{\alpha i}(\vec{x}) &:= \sum_{j \in \underline{M}} \alpha_{l_{ij}} \ p_{ij} \ . && (4.23) \end{aligned}$$

With the above notations we will simplify the expression $\|1_{V_i}\mathfrak{F}(\phi)\|^2$ step by step and finally arrive at an implementable formula. The first step is to "remove" the norm for the $L^2(\Omega)$ case:

Lemma 4.2.2:

Let E be a finite element space over the domain Ω, then, with the notations of definition 4.2.1,

$$\|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} = \|\mathfrak{F}(\phi_{\alpha k})\|^2_{L^2(V_k)} = \int_{V_k} \left(\mathfrak{F}(\phi_{\alpha k})(\vec{x})\right)^2 dV_x \ . \tag{4.24}$$

Proof. Following the above theory we get

$$\begin{aligned}
\|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} &= \|1_{V_k}\mathfrak{F}(\sum_{i\in\underline{N}} 1_{V_i}\phi_{\alpha i})\|^2_{L^2(\Omega)} \\
&= \int_\Omega \left(1_{V_k}\mathfrak{F}(\sum_{i\in\underline{N}} 1_{V_i}\phi_{\alpha i})(\vec{x})\right)^2 dV_x \\
&= \int_\Omega 1_{V_k}\sum_{i\in\underline{N}} 1_{V_i}(\vec{x})\left(\mathfrak{F}(\phi_{\alpha i})(\vec{x})\right)^2 dV_x \\
&= \int_{V_k}\left(\mathfrak{F}(\phi_{\alpha k})(\vec{x})\right)^2 dV_x \\
&= \|\mathfrak{F}(\phi_{\alpha k})\|^2_{L^2(V_K)} \ .
\end{aligned}$$

□

For the case of a polynomial κ the structural function of the scalar conservation law is a polynomial. In that case the integral in the above lemma can be computed by a weighted sum for many standard Lagrange finite element spaces.

Lemma 4.2.3:
Let E be a finite element space over the domain Ω, then with the notations of definition 4.2.1, further let the standard element V_{std} be the standard simplex and κ be a polynomial, then there exists a number $H \in \mathbb{N}$, weights $(\omega_l)_{l\in\underline{H}}$ and nodes $(\vec{y}_l)_{l\in\underline{H}}$ so that

$$\|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} = \sum_{l\in\underline{H}} \omega_l \left(\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}_l))\right)^2 |det(dL_k(\vec{y}_l))| \ . \tag{4.25}$$

Proof. In the proof of theorem 3.5.1 it has been proven, that the evaluation of the residual operator is an evaluation of a polynomial under the given presuppositions. For a fixed maximal degree d, there exists a number $H \in \mathbb{N}$, weights $(\omega_l)_{l\in\underline{H}}$ and nodes $(\vec{y}_l)_{l\in\underline{H}}$ so that every integration of a polynomial over the standard simplex V_{std} of smaller or equal degree than d can be by exactly represented by a weighted sum using these weights and nodes (see for example Rathod[11] et al. or Keast[12]). With this we can compute

$$\begin{aligned}
\|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} &= \int_{V_k}\left(\mathfrak{F}(\phi_{\alpha k})(\vec{x})\right)^2 dV_x \\
&= \int_{V_{std}} \left(\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}))\right)^2 |det(dL_k(y))| dV_y \\
&= \sum_{l\in\underline{H}} \omega_l \left(\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}_l))\right)^2 |det(dL_k(\vec{y}_l))|
\end{aligned}$$

□

The the next step towards an implementable formula is to derive an explicit calculation formula for the operator evaluations $\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}_l))$. For the scalar conservation law the residual operator at a given point $\vec{y}$ in standard (element) coordinates in element k is

$$\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y})) \quad = \quad div(\kappa(grad(\phi_{\alpha k}))grad(\phi_{\alpha k}))(L_k(\vec{y})) \ .$$

This is a real value, and the $\phi_{\alpha k}$ have a discrete representation via the global degree of freedom array α. This allows to derive implementable computation formulas for all of the differential operators in the above equation:

Lemma 4.2.4:
Let E be a finite element space over the domain Ω and let y denote the coordinates in the standard element, then, with the notations of definition 4.2.1,

$$\mathfrak{F}(\phi_{\alpha i}) = \kappa'^{T}_{\alpha i} Q_{\alpha i} \vec{g}_{\alpha i} \ + \ \kappa_{\alpha i} tr(Q_{\alpha i}) \tag{4.26}$$

or more explicitly

$$\mathfrak{F}(\phi_{\alpha i})(\vec{x}) = \left(\vec{\kappa'}^{T}_{\alpha i}(\vec{x})\right) (Q_{\alpha i}(\vec{x})) \, (\vec{g}_{\alpha i}(\vec{x})) + (\kappa_{\alpha i}(\vec{x})) \, tr(Q_{\alpha i}(\vec{x})) \tag{4.27}$$

where

$$\begin{aligned}
Q_{\alpha i} \quad &: \quad \mathbb{R}^3 \to \mathbb{R}^{3\times 3} && (4.28)\\
Q_{\alpha i} \quad &:= \quad d(grad(\phi_{\alpha i})) && (4.29)\\
\kappa_{\alpha i} \quad &: \quad \mathbb{R}^3 \to \mathbb{R} && (4.30)\\
\kappa_{\alpha i} \quad &:= \quad \kappa \circ grad(\phi_{\alpha i}) && (4.31)\\
\kappa'_{\alpha i} \quad &: \quad \mathbb{R}^3 \to \mathbb{R}^3 && (4.32)\\
\kappa'_{\alpha i} \quad &:= \quad grad(\kappa) \circ grad(\phi_{\alpha i}) && (4.33)\\
\vec{g}_{\alpha i} \quad &: \quad \mathbb{R}^3 \to \mathbb{R}^3 && (4.34)\\
\vec{g}_{\alpha i} \quad &:= \quad grad(\phi_{\alpha i}) \ . && (4.35)
\end{aligned}$$

Proof. First we can rewrite the residual operator application in the following way:

$$
\begin{aligned}
& \mathfrak{F}(\phi_{\alpha i}) \\
= \;& div((\kappa \circ grad(\phi_{\alpha i}))grad(\phi_{\alpha i})) \\
= \;& \sum_{u=0}^{2} \frac{\partial}{\partial x_u} \left((\kappa \circ grad(\phi_{\alpha i}) grad(\phi_{\alpha i})) \right)_u \\
= \;& \sum_{u=0}^{2} \frac{\partial}{\partial x_u} \left(\kappa \circ grad(\phi_{\alpha i}) \frac{\partial}{\partial x_u} \phi_{\alpha i} \right) \\
= \;& \sum_{u=0}^{2} \left(\left(\frac{\partial}{\partial x_u} \kappa \circ grad(\phi_{\alpha i}) \right) grad(\phi_{\alpha i}) + \kappa \circ grad(\phi_{\alpha i}) \left(\frac{\partial^2}{\partial x_u^2} \phi_{\alpha i} \right) \right) \\
= \;& \sum_{u=0}^{2} \left(\left(d(\kappa \circ grad(\phi_{\alpha i})) \right)_u grad(\phi_{\alpha i}) + \kappa \circ grad(\phi_{\alpha i}) \left(\frac{\partial^2}{\partial x_u^2} \phi_{\alpha i} \right) \right) \\
= \;& \sum_{u=0}^{2} \left(\left(d\kappa \circ grad(\phi_{\alpha i}) \, d(grad(\phi_{\alpha i})) \right)_u grad(\phi_{\alpha i}) + \kappa \circ grad(\phi_{\alpha i}) \left(\frac{\partial^2}{\partial x_u^2} \phi_{\alpha i} \right) \right)
\end{aligned}
$$

The equation can be simplified using $grad(\kappa)^T = d\kappa$ and the definitions above:

$$
\begin{aligned}
\mathfrak{F}(\phi_{\alpha i}) &= \sum_{u=0}^{2} \left(\left({\kappa'}_{\alpha i}^{T} Q_{\alpha i} \right)_u \vec{g}_{\alpha i} + \kappa_{\alpha i} (Q_{\alpha i})_{uu} \right) \\
&= \sum_{u=0}^{2} \left(\left(\left(\sum_{v=0}^{2} (\vec{\kappa'}_{\alpha i})_v (Q_{\alpha i})_{vu} \vec{e}_u \right)^T \right) \vec{g}_{\alpha i} + \kappa_{\alpha i} (Q_{\alpha i})_{uu} \right) \\
&= \sum_{u=0}^{2} \left(\left(\sum_{v=0}^{2} ({\kappa'}_{\alpha i})_v (Q_{\alpha i})_{vu} (\vec{g}_{\alpha i})_u \right) + \kappa_{\alpha i} (Q_{\alpha i})_{uu} \right) \\
&= \left(\sum_{u=0}^{2} \left(\sum_{v=0}^{2} ({\kappa'}_{\alpha i})_v (Q_{\alpha i})_{vu} (\vec{g}_{\alpha i})_u \right) \right) + \left(\kappa_{\alpha i} \sum_{u=0}^{2} (Q_{\alpha i})_{uu} \right) \\
&= \left(\sum_{v=0}^{2} ({\kappa'}_{\alpha i})_v \left(\sum_{u=0}^{2} (Q_{\alpha i})_{vu} (\vec{g}_{\alpha i})_u \right) \right) + \kappa_{\alpha i} \, tr(Q_{\alpha i}) \\
&= {\kappa'}_{\alpha i}^{T} Q_{\alpha i} \, \vec{g}_{\alpha i} \; + \; \kappa_{\alpha i} \, tr(Q_{\alpha i})
\end{aligned}
$$

□

The above equations still depend on terms that have no computation rules derived. This will be done in the following text. For the gradient the following rule can be derived:

Lemma 4.2.5:
Let E be a finite element space over the domain Ω and let y denote the coordinates in the standard element, then with the notations of definition 4.2.1 and lemma 4.2.4,

$$\vec{g}_{\alpha i} = \left(dL_i^{-1}\right)^T \sum_{j \in \underline{M}} \alpha_{l_{ij}} G_j \circ L_i^{-1} \tag{4.36}$$

where

$$\begin{aligned} G_j &: \mathbb{R}^3 \to \mathbb{R}^3 \\ (G_j)_n &:= \frac{\partial p_j}{\partial y_n} \ . \end{aligned} \tag{4.37}$$

Proof.

$$\begin{aligned}
& grad(\phi_{\alpha i}) \\
= \ & grad(\sum_{j \in \underline{M}} \alpha_{l_{ij}} \ p_j \circ L_i^{-1}) \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} grad(p_j \circ L_i^{-1}) \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \sum_{m=0}^{2} \left(\frac{\partial}{\partial x_m} p_j \circ L_i^{-1} \right) \vec{e}_m \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \sum_{m=0}^{2} \left((d(p_j \circ L_i^{-1}))_m \right) \vec{e}_m \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \sum_{m=0}^{2} \left((dp_j \circ L_i^{-1} dL_i^{-1}))_m \right) \vec{e}_m \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \sum_{m=0}^{2} \left(\sum_{n=0}^{2} (dL_i^{-1})_{nm} \frac{\partial p_j}{\partial y_n} \circ L_i^{-1} \right) \vec{e}_m \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \sum_{m=0}^{2} \left(\sum_{n=0}^{2} (dL_i^{-1})_{nm} (G_j)_n \circ L_i^{-1} \right) \vec{e}_m \\
= \ & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \left((dL_i^{-1})^T G_j \circ L_i^{-1} \right) \\
= \ & \left(dL_i^{-1}\right)^T \sum_{j \in \underline{M}} \alpha_{l_{ij}} G_j \circ L_i^{-1}
\end{aligned}$$

□

The abbreviations might seem a bit strange, but were selected carefully. They have the advance, that when using the above formulas in an implementation, the mapping L_i^{-1} will never be evaluated, since it will cancel out with the L_i in the integration equation 4.25. However, the above equation can still not be easily

computed, since there is a dL_i^{-1} that needs to be computed. This problem can be solved by the following lemma:

Lemma 4.2.6:
Let $L : \mathbb{R}^3 \to \mathbb{R}^3$ be a diffeomorphism and $y = L^{-1}(x)$ then

$$dL^{-1}(x) = \left(dL \circ L^{-1}\right)^{-1}(x) \tag{4.38}$$

and, if L is sufficiently smooth,

$$\frac{\partial^2 L^{-1}}{\partial x_v \partial x_u} = \left(dL \circ L^{-1}\right)^{-1} Z_{vu} \circ L^{-1} \tag{4.39}$$

where

$$Z_{vu} = -\sum_{k=0}^{2}\left(\sum_{h,l=0}^{2}\frac{\partial^2 L_k}{\partial y_l \partial y_h}(dL)^{-1}_{lv}(dL)^{-1}_{hu}\right)e_k \ . \tag{4.40}$$

Proof. Let $y = L^{-1}(x)$. Equation 4.38 is a simple consequence of applying the chain rule to $L \circ L^{-1} = id$. Further $L \circ L^{-1} = id$ implies that

$$\frac{\partial}{\partial x_u} L_k \circ L^{-1} = \sum_{h=0}^{2}\frac{\partial L_k}{\partial y_h} \circ L^{-1}\frac{\partial L_h^{-1}}{\partial x_u} = \delta_{ku}$$

and therefore

$$0 = \frac{\partial^2 L_k^{-1}}{\partial x_v \partial x_u} = \sum_{h=0}^{2}\left[\left(\frac{\partial}{\partial x_v}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\right)\frac{\partial L_h^{-1}}{\partial x_u} + \left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\frac{\partial^2 L_h^{-1}}{\partial x_v \partial x_u}\right]$$

this is equivalent to

$$\begin{aligned}
&\sum_{h=0}^{2}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\frac{\partial^2 L_h^{-1}}{\partial x_v \partial x_u} = -\sum_{h=0}^{2}\left(\frac{\partial}{\partial x_v}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\right)\frac{\partial L_h^{-1}}{\partial x_u} \\
\Leftrightarrow\quad &\sum_{k=0}^{2}\sum_{h=0}^{2}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\frac{\partial^2 L_h^{-1}}{\partial x_v \partial x_u}e_k = -\sum_{k=0}^{2}\sum_{h=0}^{2}\left(\frac{\partial}{\partial x_v}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\right)\frac{\partial L_h^{-1}}{\partial x_u}e_k \\
\Leftrightarrow\quad &dL \circ L^{-1}\frac{\partial^2 L^{-1}}{\partial x_v \partial x_u} = -\sum_{k=0}^{2}\sum_{h=0}^{2}\left(\frac{\partial}{\partial x_v}\left(\frac{\partial L_k}{\partial y_h} \circ L^{-1}\right)\right)(dL^{-1})_{hu}e_k \\
\Leftrightarrow\quad &dL \circ L^{-1}\frac{\partial^2 L^{-1}}{\partial x_v \partial x_u} = -\sum_{k=0}^{2}\sum_{h=0}^{2}\left(\sum_{l=0}^{2}\frac{\partial^2 L_k}{\partial y_l \partial y_h} \circ L^{-1}\frac{\partial L_l^{-1}}{\partial x_v}\right)(dL^{-1})_{hu}e_k \\
\Leftrightarrow\quad &dL \circ L^{-1}\frac{\partial^2 L^{-1}}{\partial x_v \partial x_u} = -\sum_{k=0}^{2}\left(\sum_{h,l=0}^{2}\frac{\partial^2 L_k}{\partial y_l \partial y_h} \circ L^{-1}(dL^{-1})_{lv}(dL^{-1})_{hu}\right)e_k
\end{aligned}$$

Now by using equation 4.38 and multiplying with $(dL \circ L^{-1})^{-1}$ the lemma follows. □

The above lemma yields again mappings that combine nicely with the integration formula 4.25. To implement equation a formula for the computation of the Hessian matrix is required:

Lemma 4.2.7:
Let E be a finite element space over the domain Ω and let y denote the coordinates in the standard element, then with the notations of definition 4.2.1, lemma 4.2.4, lemma 4.2.6 and lemma 4.2.5

$$Q_{\alpha i} = \sum_{j \in \underline{M}} \alpha_{l_{ij}} \left(\left(dL_i^{-1} \right)^T H_j \circ L_i^{-1} dL_i^{-1} + \left(\left(dL \circ L^{-1} \right)^{-1} (Z_i)_{vu} \circ L^{-1} \right)^T G_j \circ L_i^{-1} \right) \tag{4.41}$$

where

$$\begin{aligned} H_j &: \mathbb{R}^3 \to \mathbb{R}^{3\times 3} \\ (H_j)_{mn} &:= \frac{\partial^2 p_j}{\partial y_m \partial y_n} \ . \end{aligned} \tag{4.42}$$

Proof. First we can derive that

$$\begin{aligned} Q_{\alpha i} &= d(grad(\phi_{\alpha i})) \\ &= \sum_{v=0}^{2} \frac{\partial}{\partial x_v} grad(\phi_{\alpha i}) dx_v \\ &= \begin{pmatrix} \frac{\partial^2 \phi_{\alpha i}}{\partial x_0 \partial x_0} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_1 \partial x_0} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_2 \partial x_0} \\ \frac{\partial^2 \phi_{\alpha i}}{\partial x_0 \partial x_1} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_1 \partial x_1} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_2 \partial x_1} \\ \frac{\partial^2 \phi_{\alpha i}}{\partial x_0 \partial x_2} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_1 \partial x_2} & \frac{\partial^2 \phi_{\alpha i}}{\partial x_2 \partial x_2} \end{pmatrix} , \end{aligned}$$

where

$$\begin{aligned} \frac{\partial^2 \phi_{\alpha i}}{\partial x_v \partial x_u} &= \frac{\partial^2}{\partial x_v \partial x_u} \left(\sum_{j \in \underline{M}} \alpha_{l_{ij}} \ p_{ij} \right) \\ &= \sum_{j \in \underline{M}} \alpha_{l_{ij}} \frac{\partial^2 p_{ij}}{\partial x_v \partial x_u} \ . \end{aligned}$$

To simplify this the following transformations are used

$$\begin{aligned}
\frac{\partial^2 p_{ij}}{\partial x_v \partial x_u} &= \frac{\partial}{\partial x_v}\left(\frac{\partial}{\partial x_u}\left(p_j \circ L_i^{-1}\right)\right) \\
&= \frac{\partial}{\partial x_v}\left(\sum_{n=0}^{2}(dL_i^{-1})_{nu}\frac{\partial p_j}{\partial y_n}\circ L_i^{-1}\right) \\
&= \sum_{n=0}^{2}\left[\left(\frac{\partial}{\partial x_v}(dL_i^{-1})_{nu}\right)\frac{\partial p_j}{\partial y_n}\circ L_i^{-1} + (dL_i^{-1})_{nu}\frac{\partial}{\partial x_u}\left(\frac{\partial p_j}{\partial y_n}\circ L_i^{-1}\right)\right] \\
&= \sum_{n=0}^{2}\frac{\partial^2 (L_i^{-1})_n}{\partial x_v \partial x_u}\left(\frac{\partial p_j}{\partial y_n}\circ L_i^{-1}\right) + \sum_{n=0}^{2}(dL_i^{-1})_{nu}\sum_{m=0}^{2}(dL_i^{-1})_{mv}\frac{\partial^2 p_j}{\partial y_m \partial y_n}\circ L_i^{-1} \ .
\end{aligned}$$

Now using the defined abbreviations yields

$$\begin{aligned}
&\sum_{n=0}^{2}(dL_i^{-1})_{nu}\sum_{m=0}^{2}(dL_i^{-1})_{mv}\frac{\partial^2 p_j}{\partial y_m \partial y_n}\circ L_i^{-1} \\
= &\left(\sum_{n=0}^{2}(dL_i^{-1})_{nu}\left(\sum_{m=0}^{2}(dL_i^{-1})_{mv}(H_j)_{mn}\circ L_i^{-1}\right)\right) \\
= &\left(\sum_{n=0}^{2}(dL_i^{-1})_{nu}\left(\left(dL_i^{-1}\right)^T H_j \circ L_i^{-1}\right)_{vn}\right) \\
= &\left(\left(dL_i^{-1}\right)^T\left(\left(dL_i^{-1}\right)^T H_j \circ L_i^{-1}\right)^T\right)_{uv} \\
= &\left(\left(dL_i^{-1}\right)^T H_j^T \circ L_i^{-1} dL_i^{-1}\right)_{uv}
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{n=0}^{2}\frac{\partial^2 (L_i^{-1})_n}{\partial x_v \partial x_u}\left(\frac{\partial p_j}{\partial y_n}\circ L_i^{-1}\right) \\
= &\sum_{n=0}^{2}\frac{\partial^2 (L_i^{-1})_n}{\partial x_v \partial x_u}\left(\frac{\partial p_j}{\partial y_n}\circ L_i^{-1}\right) \\
= &\left(\frac{\partial^2 L_i^{-1}}{\partial x_v \partial x_u}\right)^T G_j \circ L_i^{-1} \\
= &\left(\left(dL_i \circ L_i^{-1}\right)^{-1}(Z_i)_{vu}\circ L_i^{-1}\right)^T G_j \circ L_i^{-1} \ .
\end{aligned}$$

Substituting this back yields

$$\begin{aligned}
&d(grad(\phi_{\alpha i})) \\
&= \sum_{j=1}^{M}\alpha_{l_{ij}} d(grad(p_{ij})) \\
&= \sum_{j\in\underline{M}}\alpha_{l_{ij}}\left(\left(dL_i^{-1}\right)^T H_j \circ L_i^{-1} dL_i^{-1} + \left(\left(dL_i \circ L_i^{-1}\right)^{-1}(Z_i)_{vu}\circ L_i^{-1}\right)^T G_j \circ L_i^{-1}\right) \ .
\end{aligned} \tag{4.43}$$

□

With all the results in this section a computation formula can be given. To keep this short we will assume that, κ is constant on each cell.

Lemma 4.2.8:
For a cell wise constant κ, there exists a number $C \in \mathbb{N}$, weights $(\omega_l)_{l \in \underline{C}}$ and nodes $(\vec{y}_l)_{l \in \underline{C}}$ so that, with all of the above notations for the scalar conservation law, the following holds:

$$\begin{aligned}
&\|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} \\
&= \sum_{l \in \underline{C}} \omega_l \left(tr\left(\kappa_k \sum_{j \in \underline{M}} \alpha_{l_{kj}} ((dL_k)^{-1})^T H_j (dL_k)^{-1} + \left((dL_k)^{-1}(Z_k)_{vu}\right)^T G_j \right)^2 |det(dL_k)| \right)(\vec{y}_l)
\end{aligned} \tag{4.44}$$

Proof. With the above results we can derive that

$$\begin{aligned}
& Q_{\alpha i}(L_i(\vec{y}_l)) \\
= & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \left(\left(dL_i^{-1}\right)^T H_j \circ L_i^{-1} dL_i^{-1} + \left(\left(dL_i \circ L_i^{-1}\right)^{-1} (Z_i)_{vu} \circ L_i^{-1} \right)^T G_j \circ L_i^{-1} \right)(L_i(\vec{y}_l)) \\
= & \sum_{j \in \underline{M}} \alpha_{l_{ij}} \left(((dL_i)^{-1}))^T H_j (dL_i)^{-1} + \left((dL_i)^{-1}(Z_i)_{vu}\right)^T G_j \right)(\vec{y}_l) \ .
\end{aligned}$$

And therefore using that a cell wise constant κ implies $\kappa'_{\alpha i} = 0$ we can derive that

$$\begin{aligned}
& \|1_{V_k}\mathfrak{F}(\phi_\alpha)\|^2_{L^2(\Omega)} \\
= & \int_{V_k} \left(\mathfrak{F}(\phi_{\alpha k})(\vec{x})\right)^2 dV_x \\
= & \int_{V_{std}} \left(\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}))\right)^2 |det(dL_k(y))| dV_y \\
= & \sum_{l \in \underline{C}} \omega_l \left(\mathfrak{F}(\phi_{\alpha k})(L_k(\vec{y}_l))\right)^2 |det(dL_k(\vec{y}_l))| \\
= & \sum_{l \in \underline{C}} \omega_l \left(\kappa_k tr(Q_{\alpha k}(L_k(\vec{y}_l))\right)^2 |det(dL_k(\vec{y}_l))| \\
= & \sum_{l \in \underline{C}} \omega_l \left(\kappa_k tr\left(\sum_{j \in \underline{M}} \alpha_{l_{kj}} ((dL_k)^{-1})^T H_j (dL_i)^{-1} + \left((dL_i)^{-1}(Z_i)_{vu}\right)^T G_j \right) \right)^2 (\vec{y}_l) |det(dL_k(\vec{y}_l))| \\
= & \sum_{l \in \underline{C}} \omega_l \left(tr\left(\kappa_k \sum_{j \in \underline{M}} \alpha_{l_{kj}} ((dL_k)^{-1})^T H_j (dL_k)^{-1} + \left((dL_k)^{-1}(Z_k)_{vu}\right)^T G_j \right)^2 |det(dL_k)| \right)(\vec{y}_l) \ .
\end{aligned}$$

□

The above lemma is notable, since it states that the computation of an integral of the residual operator over a cell does not require any evaluations of L_i^{-1} for any i despite the fact that ϕ_α is piecewise constructed using these mappings. This section ends with the assumption, that L_i and p_j and their derivations are computable. How these last computations can be done for a second order tetrahedrons based finite element mesh is shown in the next section.

4.3. Required Calculations on Second Order Tetrahedron Meshes

The idea of the chapter is to show a full application of the optimal residual formulation to a problem, where full application means, that it should be evident how an optimal residual algorithm can be implemented using this formalism. The evaluation of residual gauges will require evaluation of equations like equation 4.44 for a selected finite element space. However, this equation will evaluate to 0 if a first order finite element space is used (since $(Z_k)_{vu} = 0$ and $H_j = 0$ in that case).

To test the optimal residual formulation for a general case the approximation space should be selected as a second order space. A very common mesh type is based on second order tetrahedrons, where each tetrahedron is parametrized over the standard simplex. It is therefore reasonable to implement a test optimal residual algorithm on a standard Lagrange finite element space where the standard element $V_{std} \subset \mathbb{R}^3$ is the standard simplex, and all cell parametrizations L_i map V_{std} to the i-th cell of the mesh. There are multiple ways to choose the mappings L_i. For this work we use the method that is used by the Abaqus software, where each cell is defined by 10 nodes $\vec{P}_{i0}, ..., \vec{P}_{i9} \in \mathbb{R}^3$ and L_i is defined by (see the Abaqus Theory Manual[20] p. 354)

$$\begin{aligned} L_i(\vec{y}) := (1-2y_0-2y_1-2y_2)(1-y_0-y_1-y_2)\vec{P}_{i0} \\ +(2y_0-1)y_0\vec{P}_{i1} \\ +(2y_1-1)y_1\vec{P}_{i2} \\ +(2y_2-1)y_2\vec{P}_{i3} \\ +4(1-y_0-y_1-y_2)y_0\vec{P}_{i4} \\ +4y_0y_1P_{i5} \\ +4(1-y_0-y_1-y_2)y_1\vec{P}_{i6} \\ +4(1-y_0-y_1-y_2)y_2\vec{P}_{i7} \\ +4y_0y_2\vec{P}_{i8} \\ +4y_1y_2\vec{P}_{i9}\ . \end{aligned} \tag{4.45}$$

This mapping gives the parametrization for the cells. For a full standard Lagrange finite element space the local ansatz functions and their derivations are

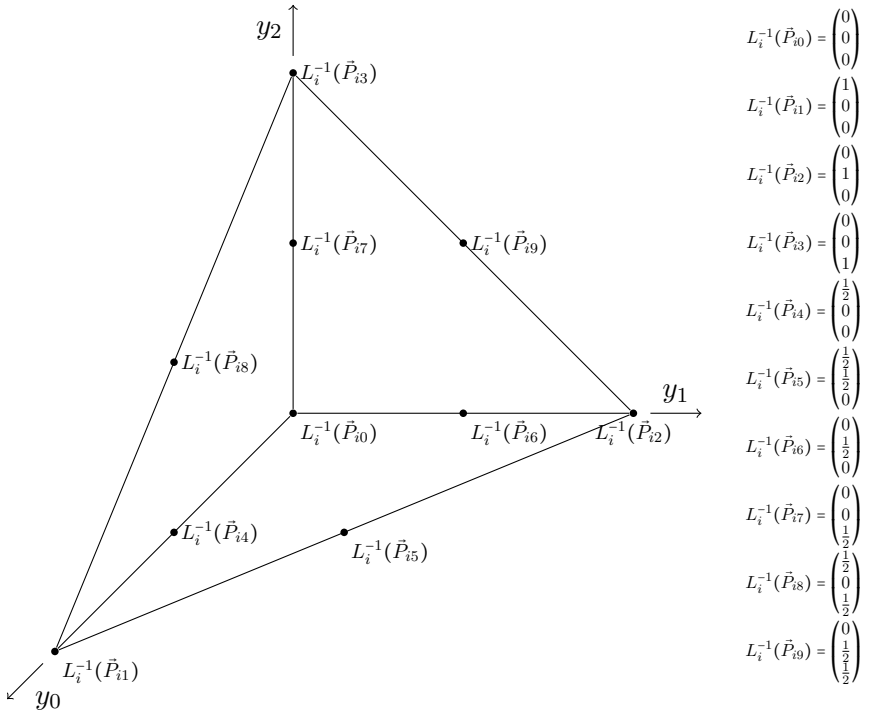


Figure 4.1.: The degrees of freedom of a second order tetrahedron on its associated standard simplex. The enumeration scheme is from Abaqus (see [20] p. 354).

required as well. To write down the functions more concisely, barycentric coordinates and their derivations are introduced:

$$\lambda_0(\vec{y}) := 1 - y_0 - y_1 - y_2 \tag{4.46}$$
$$\lambda_1(\vec{y}) := y_0 \tag{4.47}$$
$$\lambda_2(\vec{y}) := y_1 \tag{4.48}$$
$$\lambda_3(\vec{y}) := y_2\ . \tag{4.49}$$

$$\frac{\partial \lambda_i}{\partial y_u} = \begin{cases} -1 & \text{for } i = 0 \\ \delta_{(i-1)u} & \text{otherwise} \end{cases} \tag{4.50}$$

For the definition of local ansatz functions p_j on the standard element we use the

equations from Ern[9] et al. p. 22, with a special enumeration scheme, that is:

$$(h_j)_{j\in\underline{10}} = (0,1,2,3,0,1,0,0,1,2) \tag{4.51}$$
$$(k_j)_{j\in\underline{10}} = (0,0,0,0,1,2,2,3,3,3) \tag{4.52}$$
$$p_j = \begin{cases}(2\lambda_{h_j}-1)\lambda_{h_j} & \text{for } j\le 3\\ 4\lambda_{h_j}\lambda_{k_j} & \text{otherwise}\end{cases} \tag{4.53}$$
$$\frac{\partial p_j}{\partial y_u} = \begin{cases}(4\lambda_{h_j}-1)\frac{\partial\lambda_{h_j}}{\partial y_u} & \text{for } j\le 3\\ 4\left(\frac{\partial\lambda_{h_j}}{\partial y_u}\lambda_{k_j}+\lambda_{h_j}\frac{\partial\lambda_{k_j}}{\partial y_u}\right) & \text{otherwise}\end{cases} \tag{4.54}$$
$$\frac{\partial^2 p_j}{\partial y_v\partial y_u} = \begin{cases}4\frac{\partial\lambda_{h_j}}{\partial y_v}\frac{\partial\lambda_{h_j}}{\partial y_u} & \text{for } j\le 3\\ 4\left(\frac{\partial\lambda_{h_j}}{\partial y_u}\frac{\partial\lambda_{k_j}}{\partial y_v}+\frac{\partial\lambda_{h_j}}{\partial y_v}\frac{\partial\lambda_{k_j}}{\partial y_u}\right) & \text{otherwise .}\end{cases} \tag{4.55}$$

This enumeration scheme is chosen to be compatible with the Abaqus element descriptions in the following way. With the above definitions we have

$$L_i(\vec{y}) = \sum_{j\in\underline{10}} p_j(\vec{y})\vec{P}ij \ . \tag{4.56}$$

This defines everything necessary to implement an optimal residual algorithm for the scalar conservation law with the (alleged) simple residual gauge of equation 4.16. However to implement a test of the (alleged) residual gauge 4.15, it must be derived, how integrations on faces between cells can be done, to compute the flows in and out of a cell.

Let $\hat{F}_h$ be the face opposite of node h in the standard simplex, $K_h : V^2_{std} \to \hat{F}_h$ for $h \in \underline{4}$ denotes parametrization of the faces of the 3D standard simplex, where $V^2_{std} \subset \mathbb{R}^2$ is the two dimensional standard simplex. Explicitly the K_h for $h \in \underline{4}$ and their derivations are given by

$$K_0(z_0,z_1) := \vec{e}_0 + z_0(\vec{e}_1-\vec{e}_0) + z_1(\vec{e}_2-\vec{e}_0) \tag{4.57}$$
$$K_1(z_0,z_1) := z_0\vec{e}_2 + z_1\vec{e}_1 \tag{4.58}$$
$$K_2(z_0,z_1) := z_0\vec{e}_0 + z_1\vec{e}_2 \tag{4.59}$$
$$K_3(z_0,z_1) := z_0\vec{e}_1 + z_1\vec{e}_0 \tag{4.60}$$

and

$$dK_0\vec{e}_0 = \vec{e}_1-\vec{e}_0 \tag{4.61}$$
$$dK_0\vec{e}_1 = \vec{e}_2-\vec{e}_0 \tag{4.62}$$
$$dK_1\vec{e}_0 = \vec{e}_2 \tag{4.63}$$
$$dK_1\vec{e}_1 = \vec{e}_1 \tag{4.64}$$
$$dK_2\vec{e}_0 = \vec{e}_0 \tag{4.65}$$
$$dK_2\vec{e}_1 = \vec{e}_2 \tag{4.66}$$
$$dK_3\vec{e}_0 = \vec{e}_1 \tag{4.67}$$
$$dK_3\vec{e}_1 = \vec{e}_0 \ . \tag{4.68}$$

Then we get the following transformation lemma:

Lemma 4.3.1:
With the definitions above in this section, we have for every face[4] F_{ih} *in a second order tetrahedron mesh the following integration formula*

$$\int_{F_{ih}} \vec{f} \cdot d\vec{A} = \int_{\hat{F}_h} \left(\vec{f} \circ K_{ih}\right) \cdot \left(\frac{\partial K_{ih}}{\partial z_0} \times \frac{\partial K_{ih}}{\partial z_1}\right) dz_0 \wedge dz_1 \tag{4.69}$$

where

$$K_{ih} := L_i \circ K_h \tag{4.70}$$

and

$$\frac{\partial K_{ih}}{\partial z_u} = dL_i \circ K_h dK_h \vec{e}_u \ . \tag{4.71}$$

Proof. By using the definitions we can derive that

$$\begin{aligned}
\int_{F_{ih}} \vec{f} \cdot d\vec{A} &= \int_{F_{ih}} f_0 dx_1 \wedge dx_2 + f_1 dx_2 \wedge dx_0 + f_2 dx_0 \wedge dx_1 \\
&= \int_{\hat{F}_h} {K_{ih}}^* (f_0 dx_1 \wedge dx_2 + f_1 dx_2 \wedge dx_0 + f_2 dx_0 \wedge dx_1) \\
&\quad \int_{\hat{F}_h} f_0 \circ K_{ih} d(K_{ih})_1 \wedge d(K_{ih})_2 \\
&= + \int_{\hat{F}_h} f_1 \circ K_{ih} d(K_{ih})_2 \wedge d(K_{ih})_0 \\
&\quad + \int_{\hat{F}_h} f_2 \circ K_{ih} d(K_{ih})_0 \wedge d(K_{ih})_1 \ .
\end{aligned}$$

Then with the following coordinate system notations $x = L_i(y)$ and $y = K_h(z)$ it follows that

$$\begin{aligned}
d(K_{ih})_u \wedge d(K_{ih})_v &= \left(\sum_{n=0}^{1} \frac{\partial (K_{ih})_u}{\partial z_n} dz_n\right) \wedge \left(\sum_{m=0}^{1} \frac{\partial (K_{ih})_v}{\partial z_m} dz_m\right) \\
&= \sum_{n,m=0}^{1} \frac{\partial (K_{ih})_u}{\partial z_n} \frac{\partial (K_{ih})_v}{\partial z_m} dz_n \wedge dz_m \\
&= \left(\frac{\partial (K_{ih})_u}{\partial z_0} \frac{\partial (K_{ih})_v}{\partial z_1} - \frac{\partial (K_{ih})_v}{\partial z_0} \frac{\partial (K_{ih})_u}{\partial z_1}\right) dz_0 \wedge dz_1 \ ,
\end{aligned}$$

[4] This faces, have not the same enumeration scheme, that was used before, where the face was defined by the cells on both sides. Here the face is defined by a cell i, specifically a tetrahedron, and the node with the local index k that is at the "opposite" of the face.

and therefore

$$
\begin{aligned}
&\int_{F_{ih}} \vec{f} \cdot d\vec{A} \\
&\quad \int_{\hat{F}_h} f_0 \circ K_{ih} \left(\frac{\partial (K_{ih})_1}{\partial z_0} \frac{\partial (K_{ih})_2}{\partial z_1} - \frac{\partial (K_{ih})_2}{\partial z_0} \frac{\partial (K_{ih})_1}{\partial z_1} \right) dz_0 \wedge dz_1 \\
&= + \int_{\hat{F}_h} f_1 \circ K_{ih} \left(\frac{\partial (K_{ih})_2}{\partial z_0} \frac{\partial (K_{ih})_0}{\partial z_1} - \frac{\partial (K_{ih})_0}{\partial z_0} \frac{\partial (K_{ih})_2}{\partial z_1} \right) dz_0 \wedge dz_1 \qquad (4.72) \\
&\quad + \int_{\hat{F}_h} f_2 \circ K_{ih} \left(\frac{\partial (K_{ih})_0}{\partial z_0} \frac{\partial (K_{ih})_1}{\partial z_1} - \frac{\partial (K_{ih})_1}{\partial z_0} \frac{\partial (K_{ih})_0}{\partial z_1} \right) dz_0 \wedge dz_1 \\
&= \int_{\hat{F}_h} \left(\vec{f} \circ K_{ih} \right) \cdot \left(\frac{\partial K_{ih}}{\partial z_0} \times \frac{\partial K_{ih}}{\partial z_1} \right) dz_0 \wedge dz_1 \ .
\end{aligned}
$$

Further we get by applying basic differentiation rules

$$
\begin{aligned}
\frac{\partial K_{ih}}{\partial z_u} &= dK_{ih} \vec{e}_u \\
&= d(L_i \circ K_h) \vec{e}_u \\
&= dL_i \circ K_h dK_h \vec{e}_u \ .
\end{aligned}
$$

□

The above integral can again be calculated using an according quadrature formula. With these computation rules at hand testing an optimal residual formulation for the scalar conservation law is possible.

5. Optimal Residual Finite Element Methods

> Premature optimization is the root of all evil (or at least most of it) in programming.
>
> *Donald Knuth*

In this section we will discuss how direct search optimization algorithms can be used to solve an optimal residual formulation of a problem, in a massively parallel way.

There are a lot of options to choose the approximation spaces U^n and optimization algorithms $\mathfrak{A}$ to solve an optimal residual problem. To make a short summary a list of some reasonable choices for the algorithm is given:

1. Newton's method
   - \+ Very fast convergence per step
   - \- Not globally convergent
   - \- Requires information about the Hessian matrix of the objective function
   - \- Requires that the Hessian matrix is not singular in each step
   - \- Slow per step - each step solves a linear equation
2. Quasi-Newton methods, for example L-BFGS (e.g. Liu[16] et al.)
   - \+ Well suited for large scale problems
   - \- Requires information about the gradient of the objective function
   - \- Not necessarily globally convergent
3. Descend methods, for example the Barzilai-Borwein method (initially proposed in Barzilai[17] et al.)
   - \+ Usually globally convergent
   - \+ Usually simple implementation
   - \- Requires information about the gradient of the objective function
   - \- Usually not fast
4. Direct search methods, for example compass search (see Kolda[8] at al.)
   - \+ Usually globally convergent

+ Usually simple implementation
+ Usually very suitable for parallel computing
+ No information about derivations of the objective function is required
- Slow

Since Newton methods and quasi-Newton methods are well researched and require much more implementation effort then descend methods or direct search methods, I decided to test constructing optimal residual algorithms with either the Barzilai-Borwein method or compass search. The method by Barzilai-Borwein is a method with interesting convergence properties (e.g Raydan[18] and Narushima[19] et al.) that might be well suited for large scale optimization. However it requires information about the gradient of the objective function, resp. the residual gauge. This would lead to additional complexity for an implementation and analytic computations. Since the first choice of the optimization algorithm is used for testing the hole approach using optimal residual formulations and I decided to investigate an optimization algorithm, that can easily be implemented and whose convergence is secure. Therefore, the algorithm introduced later in this chapter will be based on a direct search method. Direct search methods have the additional advantage that they are not very complicated and well parallelizable (see for example Dennis[22] et al. for an overview and Hough[21] et al. for an interesting algorithm). A very nice overview and a framework for direct search methods is given in Kolda[8] et al.. To avoid the problems of implementing complexer algorithms, a very simple direct search algorithm will be used.

5.1. Nodal Search Finite Element Methods

One very simple approach to find a solution to an optimal residual formulation on a standard Lagrange finite element space is to stepwise find for each node's degree of freedom a better value. These kind of algorithms will called nodal search algorithms, a very simple reference algorithm is the compass search algorithm described in Kolda[8] et al.. Improving the result of the objective function, i.e. minimizing the residual gauge, by changing one node's degree of freedom at a time, has the advantage that independent degrees of freedom can be computed parallel.

To make the investigations somewhat simpler we introduce definitions to abbreviate the handling of gauges like the ones in equation 4.15 and 4.16.

Definition 5.1.1: Nodal Search Algorithm Nomenclature

Let E be a standard Lagrange finite element space with the notations of above. Let N_F be the number of faces of the mesh of E. For nodal search algorithms let f be the objective function created by a residual gauge $\mathfrak{G}$ that has the following

decomposition[1]

$$f(\alpha) = \mathfrak{G}(\phi_\alpha) = \sum_{i \in \underline{N}} f_i(\alpha) + \sum_{h \in \underline{N_F}} g_h(\alpha) \tag{5.1}$$

where the f_i are functions whose values solely depend on $\phi_\alpha|_{V_i}$ and g_i are functions that solely depend on $\phi_\alpha|_{F_h}$ where F_h is the face h of the mesh. The f_i are called cell objective functions and the g_h are called face objective functions. Further the functions

$$\hat{f}_k(\alpha) \quad := \quad \sum_{i \in ac(k)} f_i(\alpha) + \sum_{i \in ef(k)} g_i(\alpha) \tag{5.2}$$

are called the nodal objective functions, where ef are affected faces and defined analogous to the affected cells. Last let

$$(K_u)_{u \in \underline{N_\cap}} \tag{5.3}$$

be a decomposition into independent degrees of freedom like in definition 3.9.7 and for all $u \in \underline{N_\cap}$ let

$$\mathcal{D}_u \quad := \quad \{e_i | i \in K_u\} \tag{5.4}$$

$$d_u \quad \in \quad span(\mathcal{D}_u) \tag{5.5}$$

$$R_u \quad := \quad \underline{N} \cap \left(\bigcup_{k \in K_u} ac(k) \right) . \tag{5.6}$$

The name nodal objective function was selected because of the property described in the following theorem.

Theorem 5.1.2:

Let E be a standard Lagrange finite element space with N cells and N_g global degrees of freedom, f be an objective function with the definitions 5.1.1, then for all $u \in \underline{N_\cap}$, for all $d_u \in span(\mathcal{D}_u)$, for all $k \in K_u$, and for all $\alpha \in \mathbb{R}^{N_g}$ it holds that

$$\hat{f}_k(\alpha + d_u) = \hat{f}_k(\alpha + (d_u)_k e_k) . \tag{5.7}$$

Proof. For the proof it is sufficient to assume that $f = \sum_{i \in \underline{N}} f_i$, that is that all face objective functions are ignored, since all derivation with the cell objective functions can be equivalently performed with the face objective functions, where $ac(i) \cap ac(j) = \varnothing \Rightarrow ef(i) \cap ef(j) = \varnothing$ should be kept in mind[2]. Thus to save

[1] The rationale behind this decomposition is that it is assumed that the residual gauge is constructed by summing up residual gauges (the f_i) for each cell and that abidance of the partial differential equation on cell boundaries is enforced by penalty functions on the faces (the g_h).

[2] This is true, because cells include faces

space and avoid writing the same derivations twice it is henceforth without loss of generality assumed that

$$f(\alpha) = \sum_{i \in \underline{N}} f_i(\alpha) \ . \tag{5.8}$$

With this we can derive that

$$\begin{aligned}
\hat{f}_k(\alpha + d_u) &= \sum_{i \in \mathrm{ac}(k)} f_i(\alpha + d_u) \\
&= \sum_{i \in \mathrm{ac}(k)} f_i\left(\alpha + \sum_{l \in K_u} (d_u)_l e_l\right) \\
&= \sum_{i \in \mathrm{ac}(k)} f_i\left(\alpha + (d_u)_k e_k + \sum_{l \in K_u \setminus \{k\}} (d_u)_l e_l\right) \ .
\end{aligned}$$

Now with the presupposition of $k \in K_u$ we can rewrite the condition under the second summation sign of the above equation in the following way

$$\begin{aligned}
& k \in K_u \wedge i \in \mathrm{ac}(k) \ \wedge \ l \in K_u \setminus \{k\} \\
\Leftrightarrow \ & i \in \mathrm{ac}(k) \ \wedge \ l, k \in K_u \wedge k \neq l \\
\Leftrightarrow \ & i \in \mathrm{ac}(k) \ \wedge \ l, k \in K_u \wedge k \parallel l \\
\Leftrightarrow \ & i \in \mathrm{ac}(k) \ \wedge \ l, k \in K_u \wedge \mathrm{ac}(k) \cap \mathrm{ac}(l) = \varnothing \\
\Leftrightarrow \ & i \in \mathrm{ac}(k) \ \wedge \ l, k \in K_u \wedge i \notin \mathrm{ac}(l) \wedge \mathrm{ac}(k) \cap \mathrm{ac}(l) = \varnothing \\
\Leftrightarrow \ & i \in \mathrm{ac}(k) \ \wedge \ l, k \in K_u \wedge l \in \mathrm{ineff}(i) \wedge l \neq k \\
\Leftrightarrow \ & k \in K_u \wedge i \in \mathrm{ac}(k) \ \wedge \ l \in \ (K_u \setminus \{k\}) \cap \mathrm{ineff}(i) \ .
\end{aligned}$$

Inserting this condition back into the initial calculation yields

$$\begin{aligned}
\hat{f}_k(\alpha + d_u) &= \sum_{i \in \mathrm{ac}(k)} f_i(\alpha + (d_u)_k e_k + \sum_{l \in K_u \setminus \{k\}} (d_u)_l e_l) \\
&= \sum_{i \in \mathrm{ac}(k)} f_i(\alpha + (d_u)_k e_k + \sum_{l \in (K_u \setminus \{k\}) \cap \mathrm{ineff}(i)} (d_u)_l e_l) \\
&= \sum_{i \in \mathrm{ac}(k)} f_i(\alpha + (d_u)_k e_k) \\
&= \hat{f}_k(\alpha + (d_u)_k e_k) \ .
\end{aligned}$$

□

The above theorem states that, for a change in any combination of independent degrees of freedom $d_u \in span(\mathcal{D}_u)$, the change of the nodal objective function $\hat{f}_i$ solely depend on the degree of freedom at node i.

The above definition allows to derive a rule for the cheap computation of the objective function f for a change in one degree of freedom. To proof this we begin with a lemma showing how to compute a set of changes on independent degrees of freedom.

Lemma 5.1.3:
Let E be a standard Lagrange finite element space with N cells and N_g global degrees of freedom, f be an objective function with the definitions 5.1.1, $\alpha \in \mathbb{R}^{N_g}$ and for all $l \in \underline{N_g}$, it holds that

$$f(\alpha + d_u) = \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) - \hat{f}_k(\alpha) \right) + f(\alpha) \ . \tag{5.9}$$

Proof. W.l.o.g we can assume that

$$f(\alpha) = \sum_{i \in \underline{N}} f_i(\alpha) \ . \tag{5.10}$$

for the same reasons as stated in the proof of theorem 5.1.2. Then, since

$$\forall k, v \in K_u : (ac(k) \cap ac(v) = \varnothing \Leftrightarrow k \neq v)$$

and since d_u is constructed from (remaining) ineffective degrees of freedom regarding all f_i with $i \in R_u$, the following holds

$$\begin{aligned}
& f(\alpha + d_u) \\
= \ & \sum_{i \in \underline{N}} f_i(\alpha + d_u) \\
= \ & \left(\sum_{k \in K_u} \left(\sum_{i \in \mathrm{ac}(k)} f_i(\alpha + d_u) \right) \right) + \left(\sum_{i \in R_u} f_i(\alpha + d_u) \right) \\
= \ & \left(\sum_{k \in K_u} \left(\sum_{i \in \mathrm{ac}(k)} f_i(\alpha + d_u) \right) \right) + \left(\sum_{i \in R_u} f_i(\alpha) \right) \\
= \ & \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) \right) + \left(\sum_{i \in R_u} f_i(\alpha) \right) \\
= \ & \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) \right) - \left(\sum_{k \in K_u} \hat{f}_k(\alpha) \right) + \left(\sum_{k \in K_u} \hat{f}_k(\alpha) \right) + \left(\sum_{i \in R_u} f_i(\alpha) \right) \\
= \ & \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) - \hat{f}_k(\alpha) \right) + \left(\sum_{k \in K_u} \left(\sum_{i \in \mathrm{ac}(k)} f_i(\alpha) \right) \right) + \left(\sum_{i \in R_u} f_i(\alpha) \right) \\
= \ & \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) - \hat{f}_k(\alpha) \right) + f(\alpha) \ .
\end{aligned}$$

□

The above lemma already makes the computation of changes on a set of independent degrees of freedom cheap and parallel. To make this clearer, the above result can be used to derive the following lemma regarding changes on one degree of freedom.

Lemma 5.1.4:
Let E be a standard Lagrange finite element space with N cells and N_g global degrees of freedom and let f be a objective function as in definition 5.1.1 then for all $\alpha \in \mathbb{R}^{N_g}$ and for all $l \in \underline{N_g}$ it holds that

$$f(\alpha + \theta e_l) = \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha) + f(\alpha) \ . \tag{5.11}$$

Proof. W.l.o.g we can assume that

$$f(\alpha) = \sum_{i \in \underline{N}} f_i(\alpha) \ . \tag{5.12}$$

for the same reasons as stated in the proof of theorem 5.1.2. Let $(K_u)_{u \in \underline{N_\cap}}$ be any decomposition into independent degrees of freedom. Then for any $l \in \underline{N_g}$ there exists a $\mathcal{D}_u$ such that $e_l \in \mathcal{D}_u$. Let $d_u := \theta e_l$, then using theorem 5.1.2 and lemma 5.1.3 we can derive that

$$\begin{aligned}
f(\alpha + \theta e_l) &= f(\alpha + d_u) \\
&= \left(\sum_{k \in K_u} \hat{f}_k(\alpha + d_u) - \hat{f}_k(\alpha) \right) + f(\alpha) \\
&= \hat{f}_l(\alpha + d_u) - \hat{f}_l(\alpha) + \left(\sum_{k \in K_u \setminus \{l\}} \hat{f}_k(\alpha + d_u) - \hat{f}_k(\alpha) \right) + f(\alpha) \\
&= \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha) + \left(\sum_{k \in K_u \setminus \{l\}} \hat{f}_k(\alpha + (d_u)_k e_k) - \hat{f}_k(\alpha) \right) + f(\alpha) \\
&= \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha) + \left(\sum_{k \in K_u \setminus \{l\}} \hat{f}_k(\alpha) - \hat{f}_k(\alpha) \right) + f(\alpha) \\
&= \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha) + f(\alpha) \ .
\end{aligned}$$

□

The above result is remarkable since it holds in very general conditions, even for nonlinear problems. Further it yields a huge improvement in performance, since, given the value of $f(\alpha)$, the evaluation of $f(\alpha + \theta e_l)$ essentially reduces to an evaluation of $\hat{f}_l(\alpha)$, which is a lot cheaper to evaluate. This is caused by the fact, that a direct evaluation of f at $\alpha + \theta e_l$ would have to evaluate all cell objective functions f_i at $\alpha + \theta e_l$, while the evaluation of $\hat{f}_l$ will only require the evaluation of the cell objective functions of the affected cells of degree of freedom l. Further the average cost of evaluating the node objective functions usually does not increase with the size of the finite element mesh, since the number of the affected cells depends on the quality of the generated mesh and not on the size of the generated rated mesh.

The idea for a nodal search algorithm is to stepwise improve the degree of freedom value on each node. To do it is necessary to investigate when a line search is successful. As a "minimal" criterion it is required that a line search at least decreases the objective function, i.e.

$$f(\alpha + \theta e_l) \leq f(\alpha) \tag{5.13}$$
$$\Leftrightarrow \quad f(\alpha + \theta e_l) - f(\alpha) \leq 0 \tag{5.14}$$
$$\Leftrightarrow \quad \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha)) \leq 0 \ . \tag{5.15}$$

This motivates the following definition,

Definition 5.1.5: Nodal Search Improvement

$$(\Delta_{\theta;l} f)(\alpha) \coloneqq \hat{f}_l(\alpha + \theta e_l) - \hat{f}_l(\alpha) \tag{5.16}$$

As usual in optimization improvements must be large enough to ensure algorithm convergences. Without going into details of that requirement, that are well described in Kolda[8] at al., the following definition is introduced.

Definition 5.1.6: (Proper) Nodal Search Step
Any $\theta \in \mathbb{R}$ with

$$(\Delta_{\theta;l} f)(\alpha) \leq 0 \tag{5.17}$$

is a nodal search step on node l starting from α. θ is a proper nodal search step if $|\theta|$ is big enough to ensure algorithm convergence.

The definition of the line search improvement has the central property, that the calculation of it does not depend on the original objective function. Only the much cheaper nodal objective function needs to be investigated.

With this definition nodal search algorithms can be defined.

Definition 5.1.7: Nodal Search Finite Element Method
A Nodal search finite element method is given by an optimal residual problem solved

on a Lagrange[3] finite element approximation space, with the iteration

$$\begin{aligned} \alpha_{m;0} &:= \alpha_m && (5.18)\\ \alpha_{m+1} &:= \alpha_{m+1;N_\cap} && (5.19)\\ \alpha_{m;u+1} &:= \alpha_{m;u} + \sum_{l \in K_u} \theta_{\alpha_{m;u};l} e_l && (5.20) \end{aligned}$$

starting from some selected start point α_0, where $(K_u)_{u \in \underline{N_\cap}}$ is a decomposition into independent degrees of freedom and the

$$\theta_{\alpha_{m;u};l} \tag{5.21}$$

are nodal search steps along the node l starting from $\alpha_{m;u}$. Further each step

$$\alpha \to \alpha + \theta_{\alpha_{m;u};l} e_l \tag{5.22}$$

is called a micro step in direction l, each step

$$\alpha_{m;u} \to \alpha_{m;u+1} \tag{5.23}$$

is called a parallel step, and each step

$$\alpha_m \to \alpha_{m+1} \tag{5.24}$$

is called a macro step.

The nodal search algorithms are in each parallel step embarrassingly parallel, as it will be proven in the next lemma:

Lemma 5.1.8: Stepwise Embarrassingly Parallel Computation of Nodal Search Finite Element Methods

The sequence $(\alpha_m)_{m \in \mathbb{N}}$ generated by a nodal search algorithm, can be computed by

$$\alpha_{m+1} = \alpha_m + \sum_{u \in \underline{N_\cap}} \left(\sum_{l \in K_u} \theta_{\alpha_{m;u};l} e_l \right) \tag{5.25}$$

where for each $u \in \underline{N_\cap}$ the computation of a parallel step

$$\alpha_{m;u+1} := \alpha_{m;u} + \sum_{l \in K_u} \theta_{\alpha_{m;u};l} e_l \tag{5.26}$$

is embarrassingly parallel.

[3] For non Lagrange finite element approximation spaces, the degrees of freedom don't have a one to one association with the nodes of the mesh, thus the term "Nodal Search" would be misleading in that case. However, despite the name, the whole concept does not require the finite element approximation spaces to be Lagrange finite element spaces.

Proof. The derivation of equation 5.25 is a simple induction using definition 5.1.7. To proof that the steps are embarrassingly parallel, it is first important, that each expression $\theta_{\alpha_{m;u};l}e_l$ calculates an update of the global degree of freedom l, and thus the updates itself are embarrassingly parallel. Further the calculation of a nodal search step involves finding a $\theta_{\alpha_{m;u};l}$ such that

$$(\Delta_{\theta_{\alpha_{m;u};l};l}f)(\alpha_{m;u}) = \hat{f}_l(\alpha_{m;u} + \theta_{\alpha_{m;u};l}e_l) - \hat{f}_l(\alpha_{m;u}) \leq 0 \ . \tag{5.27}$$

The above condition only involves the nodal objective functions in one set of independent degrees of freedom. The independence implies that no two nodal objective functions $\hat{f}_l, \hat{f}_k$ for $l, k \in K_u$ with $k \neq l$ have any effective degrees of freedom in common. Since only effective degrees of freedom are required to calculate any $\hat{f}_l$ for $l \in K_u$ all $\hat{f}_l$ for $l \in K_u$ can be calculated in parallel where each calculation depends on its own set of effective degrees of freedom. Thus a parallel calculation of the nodal search steps for one set of independent degrees of freedom, does not involve any read or write access to the same memory, that is, it does not require any synchronization and therefore it is embarrassingly parallel. However after the calculation of all nodal search steps in a set of independent degree of freedom, i.e. after a parallel step, a synchronization has to occur, until all computations of micro steps have updated their degree of freedom in the degree of freedom array. □

Definition 5.1.7 does not imply that the nodal search finite element method is convergent, i.e. it does not require that enough nodal search steps are proper search steps. The simplest way to assure that is to compute the optimal steps. This motivates the following definition:

Definition 5.1.9: Optimal Nodal Search Finite Element Method

A nodal search finite element method is a optimal nodal search finite element method if each micro step solves the problem

$$(\Delta_{\theta_{\alpha_{m;u};l};l}f)(\alpha_{m;u}) \stackrel{!}{=} min \text{ for } \theta_{\alpha_{m;u};l} \in \mathbb{R} \ . \tag{5.28}$$

For differential objective functions, resp. residual gauges, a necessary optimality condition can be derived:

Lemma 5.1.10: Optimal Line Search Necessary Micro Step Length Necessary Condition

For being an optimal nodal search finite element method, it is, in case of a differential residual gauge, necessary, that for each micro step in direction l the following holds

$$\frac{\partial}{\partial \theta_{\alpha_{m;u}}} \hat{f}_l(\alpha_{m;u} + \theta_{\alpha_{m;u};l}e_l) = 0 \ . \tag{5.29}$$

Proof. Optimal nodal search finite element methods solve

$$(\Delta_{\theta_{\alpha_{m;u};l};l} f)(\alpha_{m;u}) \stackrel{!}{=} min \text{ for } \theta_{\alpha_{m;u};l} \in \mathbb{R} \tag{5.30}$$

in each micro step. If the objective function (resp. the residual gauge) is differentiable this is equivalent to find a search step that is a solution to

$$\frac{\partial}{\partial \theta}(\Delta_{\theta;l} f)(\alpha) = 0 \tag{5.31}$$

which is equivalent to solve

$$\frac{\partial}{\partial \theta} \hat{f}_l(\alpha + \theta_{\alpha;l} e_l) = 0 \ . \tag{5.32}$$

□

The optimal nodal search finite element method indeed solves the problem of finding the best approximation of a solution in a given Lagrange finite element space for a lot of differentiable objective function resp. residual gauges.

A proof of convergence will be omitted since the algorithm constructed as a generating set search algorithm with maximal step lengths whose discussion is beyond the scope of this work. However the very general convergence results derived in Kolda[8] et al. (e.g. Theorem 3.11 p. 421 , Theorem 3.14 p. 423) apply.

The above derivation of the optimal node search finite element algorithm, as an instance of an optimal residual finite element method, is still quite general. It is specifically able to solve nonlinear problems, provided that an according residual gauge is known.

However, for the start well understood linear problems are better for initial testing to avoid errors in the implementation. Further micro steps do work on a very local region of a finite element space mesh, so that in many problems, even nonlinear ones, many local problems will be linear. Therefore a specific discussion of linear problems will be given in the next section.

5.2. Nodal Search for Linear Partial Differential Equations

For linear problems it is of huge interest to derive fast computation rules since many problems are linear, or at least are in part linear.

For a linear partial differential equation with (then linear) residual operator $\mathfrak{F}$ lets assume, that the boundary conditions can be realized by choosing a sufficient finite element space E that enforces the realization of the boundary conditions (like described in theorem 3.8.3). For the scalar conservation law we discussed two alleged residual gauges (equation 4.15 and 4.16) that in conjunction with lemma 3.6.9 state, that the residual gauge is a quadric, i.e. there exists a matrix A such that

$$\mathfrak{G}(\phi_\lambda) = \lambda^T A \lambda \ . \tag{5.33}$$

where λ is a local degree of freedom array, and ϕ_λ the associated element in a finite element space. With this in mind the following representation lemmas can be derived:

Lemma 5.2.1:
Let E be a standard Lagrange finite element space of a domain Ω with M local ansatz functions p_{ij} per cell, so that the objective function in terms of definition 5.1.1 is given by

$$f(\lambda) = \sum_{h \in \underline{N}} \|\mathfrak{F}(\phi_\lambda)\|^2_{L^2(\overset{\circ}{V}_h)} \tag{5.34}$$

where $\mathfrak{F}$ be a linear operator, and

$$\phi_\lambda(\vec{x}) := \sum_{i \in \underline{N}} 1_{V_i}(\vec{x}) \sum_{j \in \underline{M}} \lambda_{(i*M+j)} p_{ij}(\vec{x}) \ . \tag{5.35}$$

Then the objective function is given as sum of cell objective functions

$$f(\lambda) = \sum_{h \in \underline{N}} f_h(\lambda) \tag{5.36}$$

where

$$\begin{aligned}
f_h &:= \lambda^T \tilde{A}_h \lambda && (5.37)\\
\left(\tilde{A}_h\right)_{mn} &= \delta_{i_m h} \int_{V_{i_m}} \mathfrak{F}(p_{i_m j_m}) \mathfrak{F}(p_{i_m j_n}) dV_x && (5.38)\\
j_m &= m \ mod \ M && (5.39)\\
i_m &= \frac{m - j_m}{M} = \frac{m - (m \ mod \ M)}{M} && (5.40)\\
j_n &= n \ mod \ M && (5.41)\\
i_n &= \frac{n - j_n}{M} = \frac{n - (n \ mod \ M)}{M} \ . && (5.42)
\end{aligned}$$

Proof. The proof is analogous to the proof of 3.6.9, that is, in short we have

$$\begin{aligned}
& \|\mathfrak{F}(\phi_\lambda)\|^2_{L^2(\overset{\circ}{V}_h)} \\
= \ & < 1_{\overset{\circ}{V}_h} \mathfrak{F}\Big(\sum_{i \in \underline{N}} 1_{V_i} \sum_{j \in \underline{M}} \lambda_{(i*M+j)} p_{ij}\Big), 1_{\overset{\circ}{V}_h} \mathfrak{F}\Big(\sum_{k \in \underline{N}} 1_{V_k} \sum_{l \in \underline{M}} \lambda_{(k*M+l)} p_{kl}\Big) > \\
= \ & \sum_{j \in \underline{M}} \sum_{k \in \underline{N}} \lambda_{(h*M+j)} \lambda_{(h*M+l)} \int_{V_h} \mathfrak{F}(p_{hj}) \mathfrak{F}(p_{hl}) dV_x \\
= \ & \sum_{m \in \underline{N}} \sum_{n \in \underline{M}} \lambda_m \lambda_m \delta_{i_m h} \int_{V_{i_m}} \mathfrak{F}(p_{i_m j_m}) \mathfrak{F}(p_{i_m j_n}) dV_x \\
= \ & \sum_{m \in \underline{N}} \sum_{n \in \underline{M}} \lambda_m \lambda_n \left(\tilde{A}_h\right)_{mn} \\
= \ & \lambda^T \tilde{A}_h \lambda \ .
\end{aligned}$$

□

Indeed an analogs computation can be done for the minimal energy conditions above. That yields the following representation lemma.

Lemma 5.2.2:
The evaluation of the energy minimizing objective function

$$f(\lambda) := \mathfrak{G}(\phi_\lambda) = \left(\sum_{i \in \underline{N}} \| 1_{A_i} \mathfrak{F}(\phi_\lambda) \|^2 \right) + \lambda^T A_E \lambda \ , \tag{5.43}$$

for an optimal residual formulation with a linear residual operator[4]*, regarding local degrees of freedom λ, can be decomposed in to cell objective functions in the following way*

$$f(\lambda) = \sum f_k(\lambda) \tag{5.44}$$

where

$$\begin{aligned}
f_h(\lambda) &:= \lambda^T \tilde{A}_h \lambda \\
\left(\tilde{A}_h\right)_{mn} &= \delta_{i_m h} \int_{V_{i_m}} \mathfrak{F}(p_{i_m j_m}) \mathfrak{F}(p_{i_m j_n}) \\
&\qquad + \kappa_{i_m} grad(p_{i_m j_m}) \cdot grad(p_{i_m j_n}) dV \\
j_m &= m \ mod \ M \\
i_m &= \frac{m - j_m}{M} = \frac{m - (m \ mod \ M)}{M} \\
j_n &= n \ mod \ M \\
i_n &= \frac{n - j_n}{M} = \frac{n - (n \ mod \ M)}{M} \ .
\end{aligned}$$

Proof. This is a simple combination of lemma 5.2.1 and lemma 4.1.5. □

The objective function above is not in itself useful, since it (allegedly) finds minimal energy solutions, without using any boundary conditions. However as it has been discussed in the mathematical frame work section this can easily be corrected by using boundary condition enforcement. Thus let the boundary conditions be enforced by an affine linear mapping

$$\lambda = \lambda^0 + P\alpha \ . \tag{5.45}$$

Then the above objective function can be reformulated to

$$f(\alpha) := (\lambda^0 + P\alpha)^T A (\lambda^0 + P\alpha) \tag{5.46}$$

[4]For example f could be derived from the scalar conservation law with a cell wise constant κ and the alleged residual gauge of equation 4.16.

where the above decomposition into cell objective function yields:

$$f(\alpha) = \sum_{i \in \underline{N}} f_i(\alpha) = \sum_{i \in \underline{N}} (\lambda^0 + P\alpha)^T \tilde{A}_i (\lambda^0 + P\alpha) \tag{5.47}$$

For this objective function the optimal nodal search finite element method is given by the following theorem:

Theorem 5.2.3: Linear PDE Minimal Energy Optimal Nodal Search Finite Element Method

Let $(\mathfrak{F}, \Gamma, \gamma)$ be a well posed pde problem with linear residual operator $\mathfrak{F}$. Let E be a standard Lagrange finite element space over a domain Ω with M local ansatz functions p_{ij} per cell, that enforces the boundary conditions by using the affine linear relation

$$\lambda = \lambda^0 + P\alpha \ . \tag{5.48}$$

Let the objective function f be an objective function exactly as in lemma 5.2.2, that is

$$f(\alpha) = \sum_{i \in \underline{N}} (\lambda^0 + P\alpha)^T \tilde{A}_i (\lambda^0 + P\alpha) \ . \tag{5.49}$$

Then the step length of each micro step of the optimal nodal search finite element is

$$\theta_{\alpha_{m;u};l} = -\frac{\xi_l + v_l^T \alpha_{m;u}}{\chi_l} \tag{5.50}$$

where

$$\tilde{v}_l^T := \sum_{i \in ac(l)} e_l^T P^T \tilde{A}_i \tag{5.51}$$

$$\xi_l := \tilde{v}_l^T \lambda^0 \tag{5.52}$$

$$v_l^T := \tilde{v}_l^T P \tag{5.53}$$

$$\chi_l := v_l^T e_l \tag{5.54}$$

and the vectors v_l are sparse vectors for all $l \in \underline{N_{g0}}$.

Proof. Since f is obviously a nowhere negative quadric, the necessary condition of lemma 5.1.10 is a sufficient condition too. To compute this condition explicitly

one derives

$$
\begin{aligned}
& \hat{f}_l(\alpha_{m;u} + \theta_{\alpha_{m;u};l} e_l) \\
= & \sum_{i \in \mathrm{ac}(l)} f_i(\alpha_{m;u} + \theta_{\alpha_{m;u};l} e_l) \\
= & \sum_{i \in \mathrm{ac}(l)} \left((\lambda^0 + P(\alpha_{m;u} + \theta_{\alpha_{m;u};l} e_l))^T \tilde{A}_i (\lambda^0 + P(\alpha_{m;u} + \theta_{\alpha_{m;u};l} e_l)) \right) \\
= & \sum_{i \in \mathrm{ac}(l)} \left(((\lambda^0 + P\alpha_{m;u}) + \theta_{\alpha_{m;u};l} P e_l)^T \tilde{A}_i ((\lambda^0 + P\alpha_{m;u}) + \theta_{\alpha_{m;u};l} P e_l) \right) \\
= & \sum_{i \in \mathrm{ac}(l)} [((\lambda^0 + P\alpha_{m;u})^T \tilde{A}_i ((\lambda^0 + P\alpha_{m;u}) + \theta_{\alpha_{m;u};l} P e_l) \\
& \qquad + (\theta_{\alpha_{m;u};l} P e_l)^T \tilde{A}_i ((\lambda^0 + P\alpha_{m;u}) + \theta_{\alpha_{m;u};l} P e_l)] \\
= & \sum_{i \in \mathrm{ac}(l)} [(\lambda^0 + P\alpha_{m;u})^T \tilde{A}_i (\lambda^0 + P\alpha_{m;u}) \\
& \qquad + (\lambda^0 + P\alpha_{m;u})^T \tilde{A}_i \theta_{\alpha_{m;u};l} P e_l \\
& \qquad + (\theta_{\alpha_{m;u};l} P e_l)^T \tilde{A}_i (\lambda^0 + P\alpha_{m;u}) \\
& \qquad + (\theta_{\alpha_{m;u};l} P e_l)^T \tilde{A}_i (\theta_{\alpha_{m;u};l} P e_l)] \\
= & \sum_{i \in \mathrm{ac}(l)} [f_i(\alpha_{m;u}) + \theta_{\alpha_{m;u};l} 2 e_l^T P^T \tilde{A}_i (\lambda^0 + P\alpha_{m;u}) + \theta^2_{\alpha_{m;u};l} e_l^T P^T \tilde{A}_i P e_l] \; .
\end{aligned}
$$

This equation is just a parabola regarding to $\theta_{\alpha_{m;u};l}$, thus the computation of the condition in lemma 5.1.10 reduces to finding a minimal point of a parabola:

$$
\theta_{\alpha_{m;u};l} = - \frac{\sum_{i \in \mathrm{ac}(l)} e_l^T P^T \tilde{A}_i (\lambda^0 + P\alpha_{m;u})}{\sum_{i \in \mathrm{ac}(l)} e_l^T P^T \tilde{A}_i P e_l}
$$

Using the given abbreviations the theorem follows. □

The above theorem is a sufficient example of a optimal residual implementation test algorithm. To keep the work "short" it will not be discussed how flow conditions through cell faces are correctly established. Instead the next chapter will show results of tests of this algorithm for the scalar conservation law.

6. Numerical Results

> Nearly every man who develops an idea works it up to the point where it looks impossible, and then he gets discouraged. That's not the place to become discouraged.
>
> *Thomas A. Edison*

In this chapter some results of testing the introduced concepts are shown. For the tests the scalar conservation law

$$div(\kappa(grad(\phi))grad(\phi)) = 0 \tag{6.1}$$

from the above chapter has been used, in conjunction with the nodal search finite element method on a second order isoparametric 3D standard Lagrangian finite element space. Performing the test with first order elements would not be advisable, since for a ϕ-approximation constructed from first order elements, equation 6.1 would always be realized on each cell. Thus, testing with first order elements would not be very conclusive. Especially, since, due to time limitations, the proper patching/flow conditions described in the chapters above have not been implemented[1]. To compensate for this the minimal energy condition, that has been discussed in the previous chapters has been used. This further allowed to compensate for the, due to time limitations, not implemented Neumann conditions. A minimal energy condition will (usually) always yield a unique solution, even if there are open boundaries. However it could not be determined, weather the minimal energy condition is correct for all problems. For the tested cases it seems to be correct and converges to something similar to a solution of a problem using homogeneous Neumann conditions for the open boundaries. However, this is not the kind of solution one would like to get for open boundaries.

6.1. The Nodal Search Finite Element Method Test Implementation

Due to time and space limitations the details of the implementation will not be discussed. By the end of the deadline for this work the code was not sufficiently tidied up to be of any didactic worth.

[1] Well, they have been implemented, but I didn't get it working in time, most probably because of some face orientation/sign bugs.

The C programming language has been used as implementation language, since that allows the implementation to be fast enough to test real world problems. While doing the implementation in C, I aimed toward using OpenCL or CUDA to run the algorithm on GPGPU-Hardware. However, only an ad-hoc parallelization with OpenMP has been done so far. The program CUBIT 12.2 has been used to create the test cases - including the real world test case. The Abacus file format has been used for exporting the problems from CUBIT. The program Paraview 3.10 was used for visualizing the results. I did not use any third party libraries other then the C and POSIX standard libraries. The development was done on Ubuntu 11.10 Linux systems using the gnu compiler collection 4.6.1 . In a simple list the implementation of the solver for the scalar conservation law, with cell-wise constant κ, is given by:

1. Read a mesh from an Abaqus-file generated by CUBIT with an ad-hoc self written parser.

2. Solve the problem with the algorithm of theorem 5.2.3 for the scalar conservation law of chapter 4 by

   - using isoparametric second order tetrahedron elements as described in 4.3
   - using an ad-hoc self written iterative mark and sweep algorithm to generate sets/arrays of independent degrees of freedom.
   - using hard coded boundary conditions
   - using hard coded material properties
   - using the computation formulas derived in chapter 4
   - aborting when the step size is lower than a preset (again hard-coded) tolerance

3. Generating legacy VTK output format 1.0 for visualization with Paraview

Further I used simple ad-hoc self written unit testing to test components and scons 2.0.1 as build system.

6.2. Tests of a Simple Cylindrical Geometry

The tests begun with testing a simple cylindrical geometry with two layers of material. This geometry is shown in Fig. 6.1. The analytical solution to the

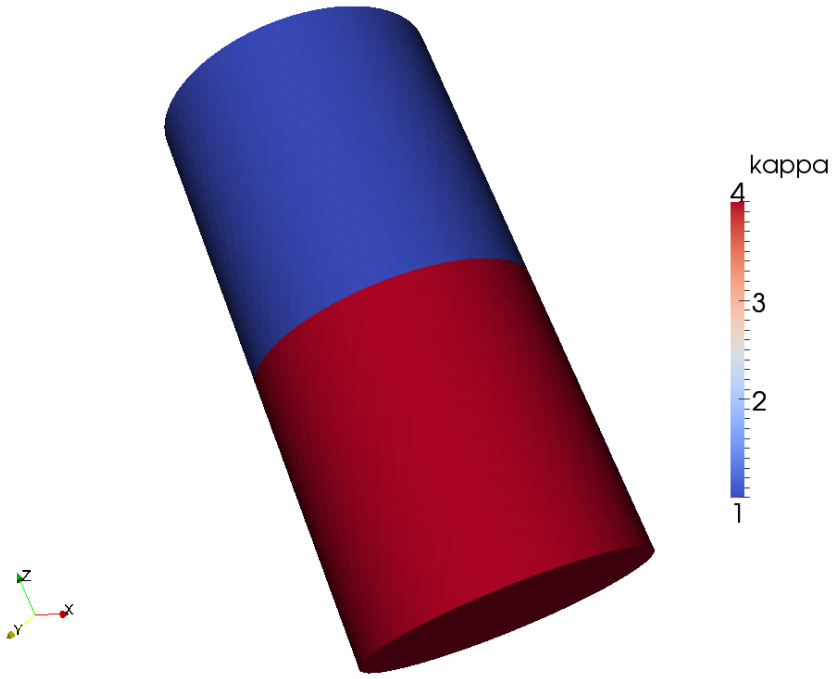
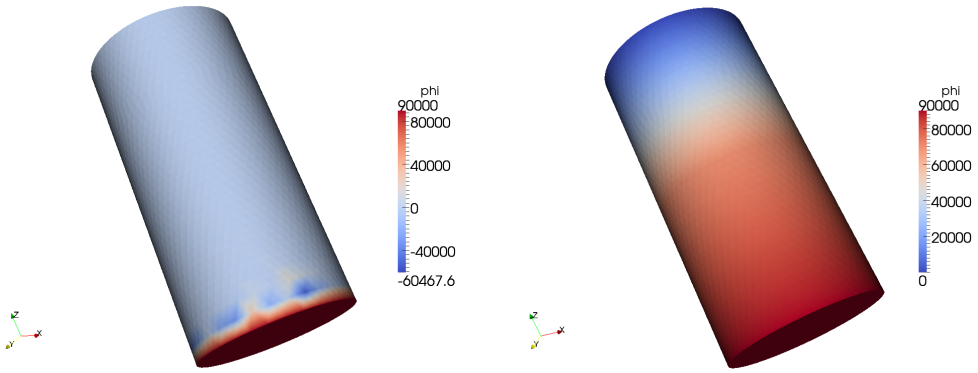


Figure 6.1.: The simple cylindrical test geometry displaying the materials. (16586 Nodes, 17 sets of independent dofs, min. set size 6, max. set size 1321, avg. set size 934.47 with std. dev. 438.16)

problem is given by

$$\begin{aligned}
\phi(z) &= 1_{[z_0-z_1]}(z)\frac{z-z_0}{z_1-z_0}(\phi_1-\phi_0)+\phi_0 && (6.2)\\
&\quad +1_{[z_2-z_1]}(z)\frac{z-z_1}{z_2-z_1}(\phi_2-\phi_1)+\phi_1 && (6.3)\\
\phi_0 &= 90000 && (6.4)\\
\phi_2 &= 0 && (6.5)\\
\phi_1 &= \frac{\phi_2\kappa_2+\phi_0\kappa_1}{\kappa_1+\kappa_2} && (6.6)\\
\kappa_1 &= 4 && (6.7)\\
\kappa_2 &= 1 && (6.8)
\end{aligned}$$

where ϕ_2 is the Dirichlet boundary data on the top face and ϕ_0 is the Dirichlet boundary data on the bottom face. First it was tested if the continuity of the elements in the second order finite element space is enough to ensure convergence to a physical solution. That is, test if the minimal energy solution can be omitted. Here "physical solution" means, that the cylinder could be interpreted as a capacitor or



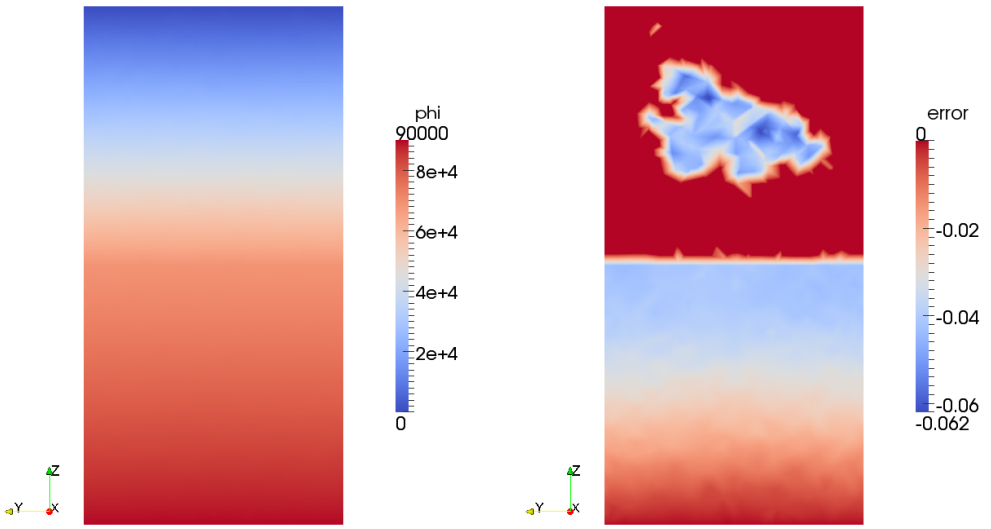
(a) Wrong result of the optimal nodal search FEM when omitting minimal energy and patching conditions.

(b) Correct result of the optimal nodal search FEM with minimal energy condition and omitting patching conditions.

Figure 6.2.: Results for the cylindrical geometry with and without the minimal energy condition. (stopping after macro 117840 steps on step size 0.1 (maximums norm).

a conductor, where in both cases ϕ would be the electric potential. As shown in Fig. 6.2(a) the solution of the nodal search finite element method converges to an incorrect "solution", if the minimal energy condition is omitted. Fig. 6.2(b) shows that the result is correct when using the minimal energy condition.

However I did not investigate the dependence between being a minimal energy solution and realizing correct patching/flow conditions, since the later where not (correctly) implemented. In Fig. 6.3 the solution and the relative error in the interior of the cylindrical geometry problem are shown.



(a) The field in the interior.

(b) The relative error in the interior.

Figure 6.3.: Results for the cylindrical geometry in the interior.

6.3. Convergence and Robustness of the Nodal Search FEM

The simple cylindrical geometry test problem with a well understood analytic solution can be used well to examine some convergence and robustness aspects. The first interesting property regarding convergence is the behavior of the step length which is shown in Fig. 6.4. As the figures show the step length will converge to zero fast, that is linear in a logarithmic plot. The other most interesting feature regarding the convergence is the behavior of the error over the time (i.e. the number of macro steps). This is shown in Fig. 6.5. The result that the step length approaches zero fast is not surprising. Descend methods are well known for their slow convergence. However, in practice, since computing the steps fast is parallel, this seems not to be too bad. To examine the robustness, in respect to what is the condition for classical finite element methods, κ_1 has been increased and the number of steps until the abort condition was reached have been counted. The result is shown in Fig. 6.6

This result is interesting. After a somewhat linear "dose-response" line, where the dose is the "ill-contentedness" and the response is the number of macro steps, the response seems to switch into saturation and then plunges. Here one has to say that for the extreme values of κ after the plunge, the result yield is still a

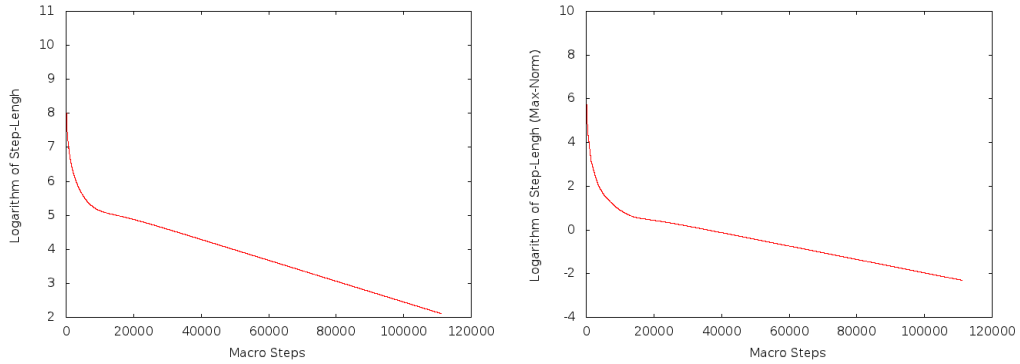


Figure 6.4.: Results for the steplengh of the cylindrical geometry.

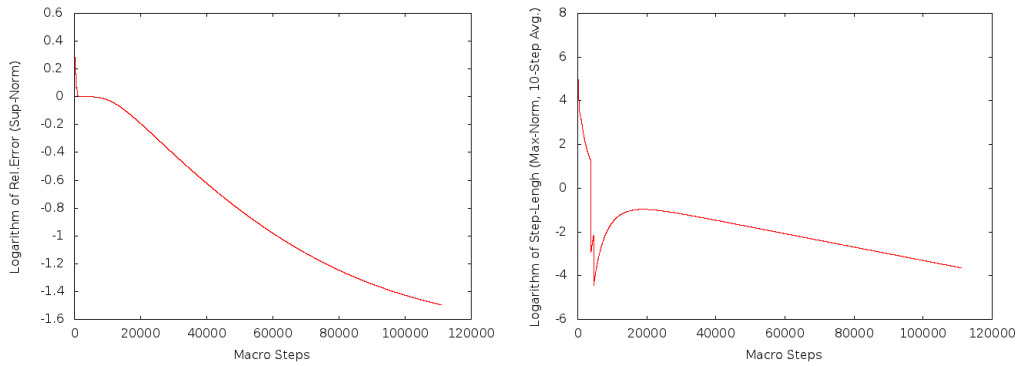


Figure 6.5.: Results for the convergence behavior of the cylindrical geometry in the interior.

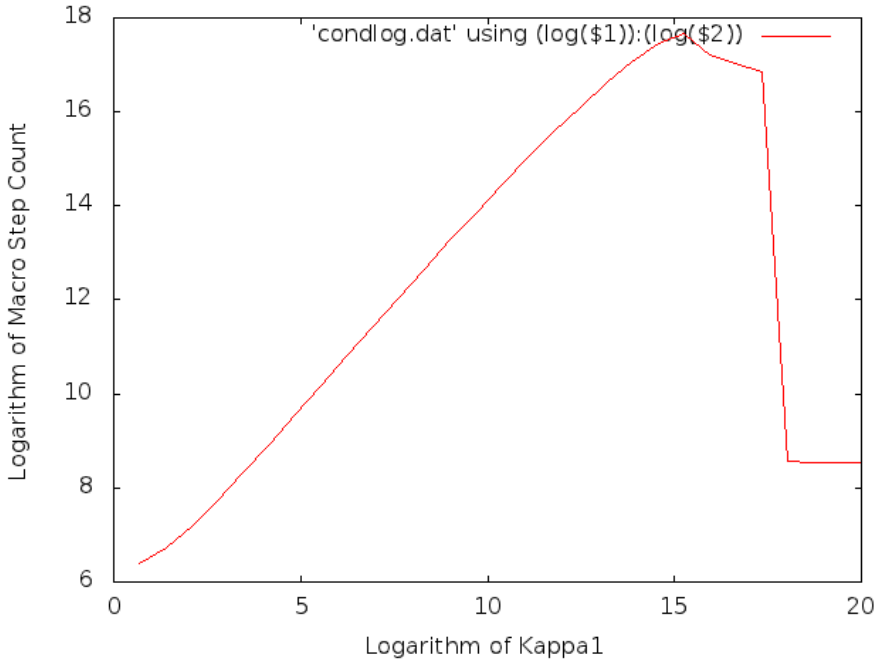


Figure 6.6.: A test of condition sensibility of the algorithm by increasing κ_1. The step count is the count of macro steps needed to reach a step-size smaller than 0.1 (max-norm).

reasonable approximation, but not as good as the prior ones. This behavior is interesting and should be investigated in later works.

6.4. Simulating a Static Dipole

To investigate some more complex geometry and test the open boundaries a simple dipole geometry (Fig. 6.7) has been simulated. For time and space reasons no quantitative analysis has been performed. The results of this simulation are given in figure 6.8 where the abort condition had been that the step size is smaller then 10^{-3}. As one can see around poles the results seem to be correct. However at the open boundary the results seem to be not correct for an open boundary. They are similar to homogeneous Neumann conditions. However they differ from homogeneous Neumann conditions in the sense, that the equipotential-lines are not exactly orthogonal to the boundary of the computation box. This is not the kind of solution one would like to have for open boundary conditions. The reason for this is possibly the following. Technically the energy is proportional to the squared gradient of the potential ϕ. The gradient is however usually small if the equipotential lines don't have much curvature. Thus minimizing the energy in a bounded box will also tend to minimize the curvature of the equipotential lines in

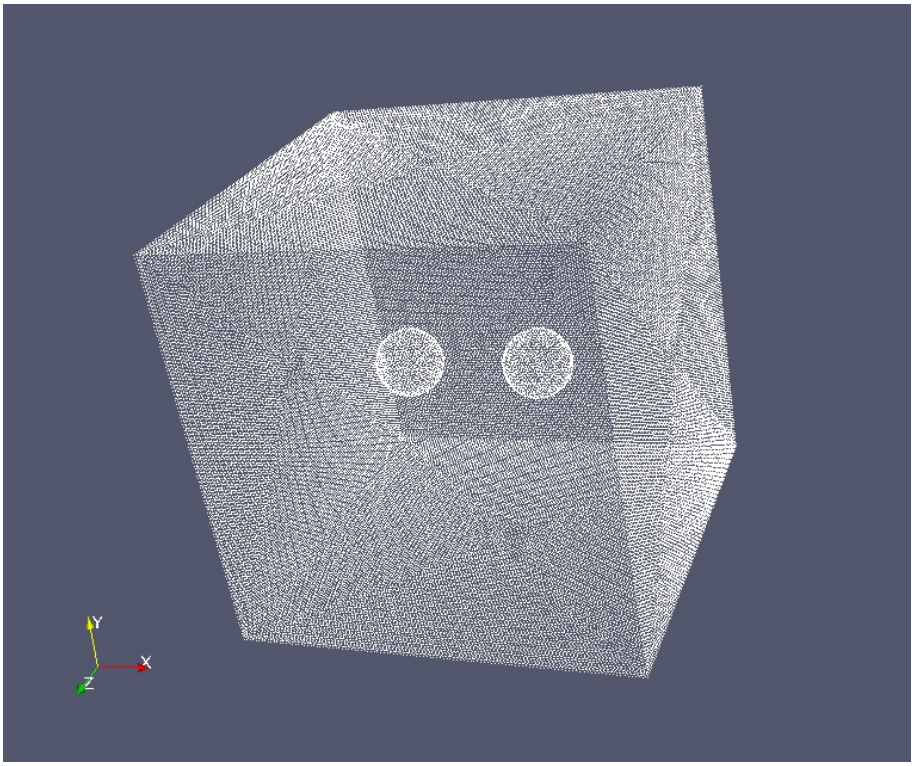


Figure 6.7.: A 3D visualization of the volume used to perform the dipole study. It was meshed second order with about 1.5M degrees of freedom.

that box. Thus minimal energy seems not to be a sufficient condition to model open boundaries. This behavior occurs for real world simulation examples as well.

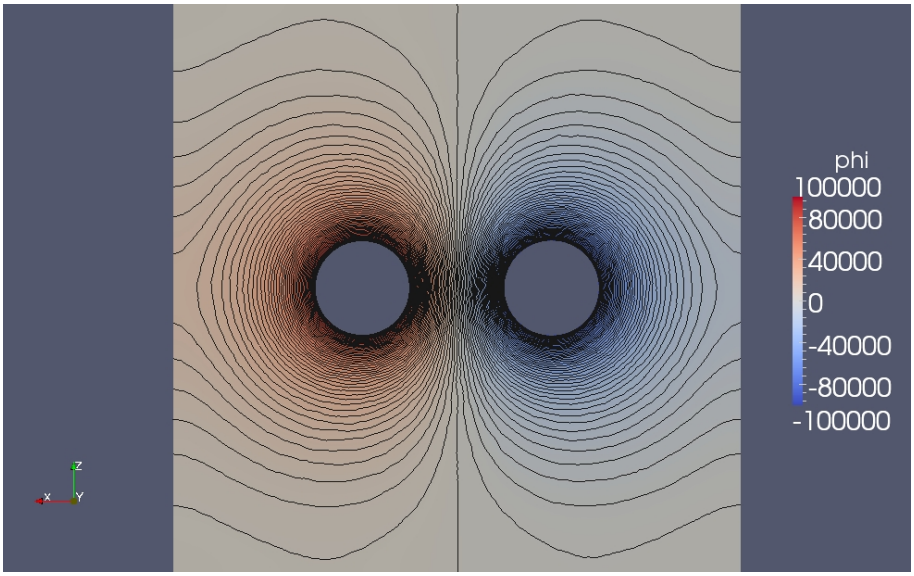


Figure 6.8.: A qualitative static dipole simulation. .

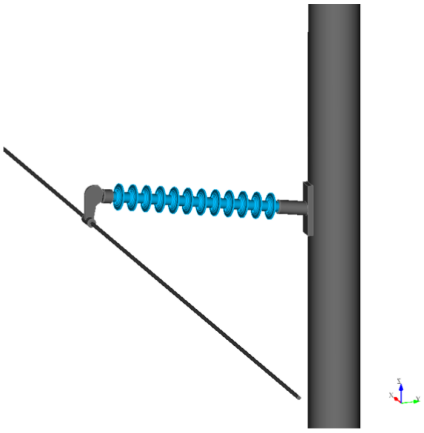
6.5. Simulating Real World High Voltage Insulators

As a last test for the nodal search finite element algorithm I have tried to simulate a real world structure. The structure is shown in Fig. 6.9 and was used in a study[2] for LAPP Insulator[3] supervised by Prof. Dr. M. Clemens (Bergische Universität Wuppertal).

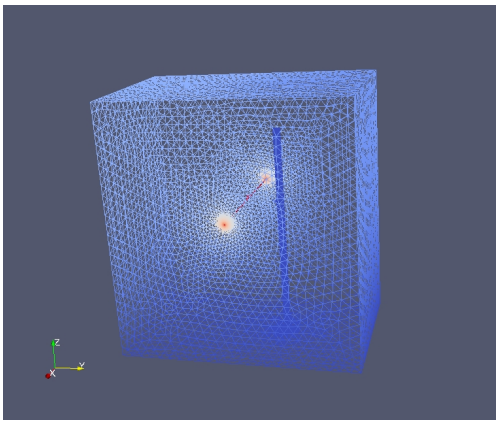
The solutions are at least correct within the limits of this examination. However the simulation takes really a long time, roughly 3 days on a 48-CPU machine with 2.8GHz per CPU. The abort condition was that the macro step size in the max.-norm has to be lower then 0.1. It took 108740 macro steps to compute the solution, where the second order mesh had 4806834 nodes/dofs. The degrees of freedoms where decomposed into 18 sets of independent degrees of freedom, with a minimal set size of 78, a maximal set size of 376646 and a average set size of 258236.28 with a standard deviation of 132376.50.

Even after a "short" time of computation the solution approximates the final solution globally. This shown in Fig. 6.9(d), where the simulation was aborted after about an hour. However, as is shown in Fig. 6.10(b) in comparison with Fig.6.9(d) the solution is locally (still) wrong. However globally this short time solution is not that bad and thus it probably could be used for a starting solution for an other nonlinear finite element algorithm, that uses e.g. Newtons method. Finally the solution of the real world problem seems to be correct when waiting

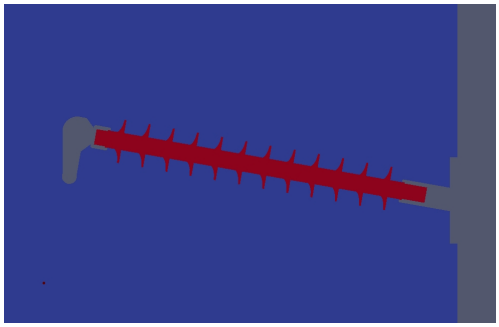
[2]The study is 'Simulation Report „CL2-061-21-100-A without Ring line post" Rev. 1.0', however it will probably never be publicly available.
[3]LAPP Insulator GmbH & Co. KG, Bahnhofstrasse 5, D-95632 Wunsiedel Germany



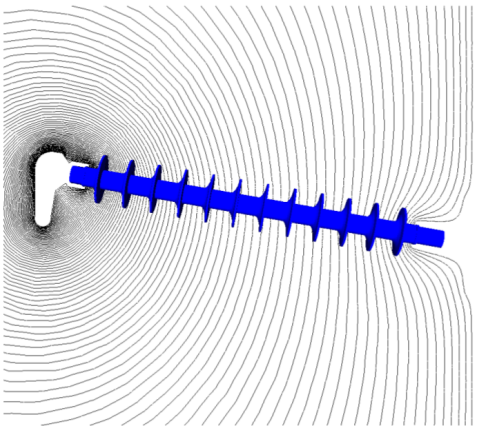
(a) The CAD Model



(b) The Meshed Model

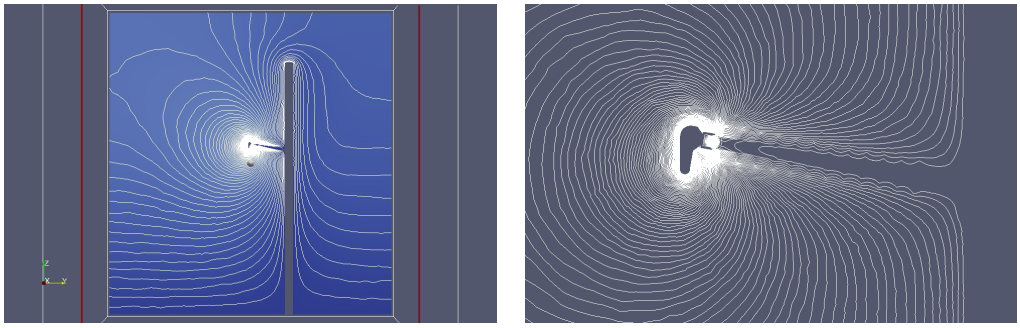


(c) Visualization of the material properties. (Red with a value of 4, blue with a value of 1 and gray is not in the mesh.)



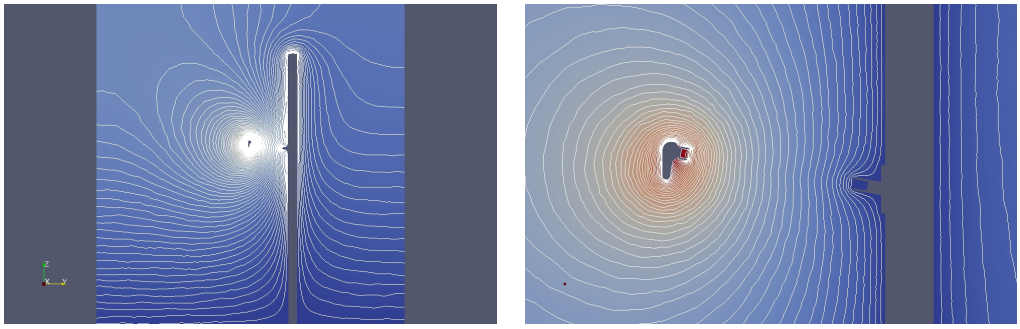
(d) A visualization of the solution for the study. It was generated with the Meqsico code that was developed under supervision of Prof. Dr. M. Clemens.

Figure 6.9.: The the real world test problem, meshed second order with 4.8M nodes. The model is from a study for the LAPP Insulator GmbH & Co. KG.



(a) Global Solution (b) Local Solution

Figure 6.10.: Wrong solutions for a short computation time (one hour) for the tested insulator.



(a) Global Solution (b) Local Solution

Figure 6.11.: Correct solutions for the tested insulator. (The insulator geometry is not shown always due to some quirk with the Paraview visualization, it was definitely in the data, which I checked twice.)

long enough. This is shown in Fig. 6.11.

However the comparison of the results can only be qualitative, since the boundary conditions and material properties where hard coded in the test implementation and not the same as in the study[4].

However the comparison of the sort time simulation result with the long time simulation result give an interesting hint to a further property of the nodal search finite element method. The boundary information seems to move slowly through the cells. That is why the solution looks fine fast for the big cells in the outer space of the mesh but is not correct at the insulator which is meshed with a lot of small cell. It might be possible to reduce the this effect and making the algorithm faster by applying a multi-grid scheme to it. The reason is that the algorithm is (unintentionally) very similar to a multiplicative Schwartz method. Every micro

[4]Where, beside that, it is not clear weather I would be allowed to make details of that study public.

step solves the optimal residual formulation for a nodal objective function, that is for the domain of the affected cells of the associated node. Some micro step in the next parallel step, i.e. some micro step using a degree of freedom from a different set of independent degrees of freedom, will be performed on a degree of freedom whose affected domain overlaps with the affected domain of the prior micro step. For multiplicative Schwarz methods multi-grid approaches are known to yield significant improvement. This will probably be investigated in some later work.

6.6. Scaling and Speed

The algorithm was implemented with an ad-hoc OpenMP parallelization. This causes a lot of overhead, so that small examples are actually computed slower. On a 48 core system (with 2.8GHz clock speed) the speedup for the simple cylindrical geometry of section 6.2 between using one and all cores is a about a factor of 3. Further for very small problems the speedup by parallelization becomes a penalty of about factor 100 or more. This indicates that OpenMP causes a significant overhead when entering and leaving parallel sections. For small problems, the sets of independent degrees of freedom are very small and almost nothing is to be computed in that case. A systematic examination of the speed in wall clock time could not yet be done, since the machine used to test the algorithm, was under heavy load from other projects. For the small cylindrical examples described above Fig. 6.12 shows some preliminary results.

At the current state the results of my experiments are quite chaotic regarding the used abort criteria and the system load caused by other simulations. Thus a systematic review of the speed will probably be done in a later publication. However there is one further remarkable result when comparing the number of steps needed to reach the aborting criteria of a step size 0.1 (in max.norm) in two different cases: For the cylinder mesh with about 20k degrees of freedom it took about 110k macro steps. For the insulator mesh with about 4800k degrees of freedom it took about 110k macro steps, too. This is remarkable and gives a hint, that as it was expected, the geometric condition of the mesh might not have a significant impact on the convergence speed in terms of macro steps. Each macro step takes, of course, a lot longer to compute for the bigger problem. However

Dofs	Ind. DoF Sets (set size min/max/avg)	SP Time	Time MP	Macro Steps
173	14 (1/16/3.64±0.72)	0.72s	0.22s	570
16586	17 (6/1321/934.47±438.1)	7m30s	3m38s	111010

Figure 6.12.: Preliminary simple speed metrics for the Cylindrical geometry above. Measured on a 48 Core System (2.8GHz clock speed). The abort condition was that the step size in maximums norm is smaller than 0.1.

since the algorithm is stepwise embarrassingly parallel, this might be a solvable problem. This property certainly calls for further investigation.

6.7. Conclusion and Outlook

The conclusion will start with summarizing what has been archived in this work:

- A new mathematical framework to formulate partial differential problems has been developed around the, in this work developed, notion of optimal residual formulations. A basic initial theory has been developed that proofs the applicability of this formulation to linear problems.

- Based on the introduced optimal residual formulations optimal residual algorithms have been defined. Optimal residual algorithms have the advantage, that they require almost no structure for the approximation spaces used to approximate the solutions of partial differential equations.

- The introduced framework includes results necessary to formulate optimal residual algorithms using finite element spaces for linear and non linear problems with Neumann and Dirichlet boundary conditions.

- A constructive definition of finite element spaces has been developed that is well suited for implementation and building finite element spaces of functions that already realize boundary conditions, like Neumann or Dirichlet boundary conditions.

- The way, the framework can be used, has been demonstrated by applying it to a scalar conservation law.

- The class of optimal residual finite element methods has been discussed, that is optimal residual algorithms using finite element spaces as approximation spaces.

- The nodal search finite element method has been developed. It has the nice property that its implementation decomposes into a sequence of embarrassingly parallel steps. Further it is globally convergent, that is they don't need a good start point estimation, like for example Newtons method based algorithms. Last it seems, regarding to the numerical results, robust but slow with promising scaling properties.

- The nodal search algorithm has been (partially) implemented - for time and space reasons - only on an isoperimetrical, second order, tetrahedron based, finite element space. Even so, this partial implementation, lacking proper treatment of cell boundaries, yields good results for the tested problems.

This work, as its title states, was done to introduce some principles, not a complete theory. Therefore I introduced a lot of general definitions that would allow to investigate a lot more than what has been done and could be done in this work. There are now a lot more open questions that the ones stated in the beginning of the work. However, I cold answer my initial questions or at least cast light on the answer of them, as I intended. Thus the end of this work I will give an outlook on what might be promising ways to proceed:

- Test an implementation with correct residual gauge or proof the tested alleged residual gauge to be correct.
- Test complete Neumann and Dirichlet Boundary conditions.
- Test GPGPU implementations.
- Investigate nonlinear problems which was one of the motivations for this work.
- Test other optimal residual finite element methods (e.g. based on Newton, Quasi-Newton, BB, CG, etc. methods)
- Investigate nodal search finite element multi-grid methods.
- Investigate how to building good DoF decompositions for nodal search finite element methods. The current ad-hoc one is obviously bad.
- Use the nodal search finite element method as a start value generator for other non globally convergent algorithms. (E.g. algorithms depending on variations of Newtons method.)
- Investigate non finite element optimal residual algorithms.
- Investigate constraint enforcement of other things than boundary conditions, for example try to enforce energy conservation of a system.

A. Additional Results Regarding Extended Kernels

This appendix lists some additional results I removed from the main work, since they where at some point not longer required. In this respect one should be wary regarding their correctness, because I did not give much attention to them, after they where moved to the appendix.

Lemma A.0.1:
A function with minimal (weakly) extended kernel is not necessarily continuous at points in the kernel.

Proof. The function $f : \mathbb{R} \longrightarrow \mathbb{R}^+$

$$f(x) = \begin{cases} -x + 2 \text{ for } x < 1 \\ x - 1 \text{ for } x \geq 1 \end{cases} \tag{A.1}$$

has $\ker(f) = \{1\} = \operatorname{exker}(f)$ and therefore has a minimal (weak) extended kernel, but it is not continuous at $x = 1$. □

Lemma A.0.2: Extended Kernel Completeness
Let V, W be Banach spaces, $F : V \to W$ with nonempty extended kernel, then the extended kernel is complete.

Proof. Let $(v_i)_{i\in\mathbb{N}} \subset exker(F)$ be a Cauchy sequence. V is complete, thus there exists $v_\infty \in V$ such that $\lim_{i\to\infty} v_i = v_\infty$. Since all v_i are elements of the extended kernel, there exist sequences $(w_{ij})_{j\in\mathbb{N}} \subset V$ with $\lim_{j\to\infty} F(w_{ij}) = 0$ and $\lim_{j\to\infty} w_{ij} = v_i$. Then $W_{i;n} := \{w_{ij} | F(w_{ij}) < \frac{1}{n} \wedge ||w_{ij} - v_i|| < \frac{1}{n}\}$ are non empty sets. The axiom of choice implies that there is a sequence $(h_n)_{n\in\mathbb{N}} \subset V$ such that $\forall n \in \mathbb{N} : h_n \in W_{n;n}$. By construction $\lim_{n\to\infty} F(h_n) = 0$ and $\lim_{j\to\infty} ||h_j - v_j|| = 0$. Further $||h_n - v_\infty|| = ||h_n - v_n + v_n - v_\infty|| \leq ||h_n - v_n|| + ||v_n - v_\infty||$ and therefore $\lim_{n\to\infty} ||h_n - v_\infty|| = 0$. Finally, by construction the sequence $(h_n)_{n\in\mathbb{N}} \subset V$

has therefore the property, that $\lim_{n\to\infty} F(h_n) = 0$ and $\lim_{n\to\infty} h_n = v_\infty$, thus $v_\infty \in exker(F)$. □

Lemma A.0.3: Extended Kernel Closedness

Let V, W be Banach spaces. If $F : V \to W$ is a function with non empty extended kernel, then for every $v \in V$ there exist $z \in exker(F)$ with $d(v, exker(F)) = \|v - z\|$.

Proof. By definition $d(v, exker(F)) = \inf\{\|v - z\| \mid z \in exker(F)\}$. Thus, there exists a sequence $(w_i)_{i\in\mathbb{N}} \subset exker(F)$ and a value $z_{inf} \in V$ with $\lim_{i\to\infty} w_i = z_{inf}$ and $d(v, exker(F)) = \|v - z_{inf}\|$. The extended kernel is complete, therefore $z_{inf} \in exker(F)$. □

Lemma A.0.4:

There exist functions is with nonempty extended kernel, but empty kernel.

Proof. For example: $f(x) = \begin{cases} \|x\| \text{ for } x \neq 0 \\ 1 \text{ for } x = 0 \end{cases}$ □

Lemma A.0.5:

Let V, W, Z be Banach spaces. If $F : V \to W$ and $G : V \to Z$ are functions with minimal extended kernel and $\alpha, \beta \in (0, \infty)$ then

$$H := v \mapsto \alpha\|F(v)\|_V^2 + \beta\|G(v)\|_Z^2 \tag{A.2}$$

has a minimal extended kernel.

Proof.

$$\begin{aligned} exker(H) &:= \{v \in V | \lim_{n\to\infty} v_n \in V = v \text{ with } \lim_{n\to\infty} H(v_n) = 0 \text{ exists}\} \\ &= \{v \in V | \ldots \text{ with } \lim_{n\to\infty} F(v_n) = 0 \wedge \lim_{n\to\infty} G(v_n) = 0 \text{ exists}\} \\ &= \{v \in V | v \in exker(F) \wedge v \in exker(G)\} \\ &= \{v \in V | v \in ker(F) \wedge v \in ker(G)\} \\ &= ker(H) \end{aligned}$$

The last step is true since by construction $H(v) = 0 \Leftrightarrow F(v) = 0 \wedge G(v) = 0$. □

Lemma A.0.6: Extended Kernels of Linear Operators with Adjoints
Let D, H, W be Hilbert spaces and $T : D \to H$ be a linear operator with adjoint operator $T^* : W \to D$ where W is dense in H, then T has a minimal extended kernel.

Proof. Let $v_\infty \in exker(T)$. Then, by definition of the extended kernel, there exist $(v_n)_{n\in\mathbb{N}} \subset D$ with $\lim_{n\to\infty} v_n = v_\infty$ and $\lim_{n\to\infty} T(v_n) = 0$. Then:

$$\lim_{n\to\infty} T(v_n) = 0 \tag{A.3}$$
$$\Leftrightarrow \quad \forall w \in W :< \lim_{n\to\infty} T(v_n), w >= 0 \tag{A.4}$$
$$\Leftrightarrow \quad \forall w \in W : \lim_{n\to\infty} < T(v_n), w >= 0 \tag{A.5}$$
$$\Leftrightarrow \quad \forall w \in W : \lim_{n\to\infty} < v_n, T^*(w) > \tag{A.6}$$
$$\Leftrightarrow \quad \forall w \in W :< \lim_{n\to\infty} v_n, T^*(w) > \tag{A.7}$$
$$\Leftrightarrow \quad \forall w \in W :< v_\infty, T^*(w) > \tag{A.8}$$
$$\Leftrightarrow \quad \forall w \in W :< T(v_\infty), w >= 0 \tag{A.9}$$
$$\Leftrightarrow \quad T(v_\infty) = 0 \tag{A.10}$$

In short $v_\infty \in exker(T) \Rightarrow v_\infty \in ker(T)$. Since $ker(T) \subset exker(T)$ this proofs $ker(T) = exker(T)$ □

Corollary A.0.7:
The derivation ∂^α operator on $L^2(\Omega)$ where $\Omega \subset \mathbb{R}^n$ is open, has a minimal extended kernel for all multiindices α.

Lemma A.0.8:
Let D, H, W be Hilbert spaces and $T : D \to H$ be a linear operator with adjoint operator $T^* : W \to D$ where W is dense in H. Further let $g \in H$ be a function such that there exists a $\tilde{g} \in D$ so that $T(\tilde{g}) = g$. Then $L := f \mapsto T(f) - g$ has a minimal extended kernel.

Proof. Since T is linear L can be expressed as $L = f \mapsto T(f - \tilde{g})$. Let $v_\infty \in exker(L)$. Then by definition of the extended kernel, there exist $(v_n)_{n\in\mathbb{N}} \subset D$ with $\lim_{n\to\infty} v_n = v_\infty$

and $\lim_{n\to\infty} L(v_n) = 0$. Then analog to the above case

$$\begin{aligned}
& \lim_{n\to\infty} L(v_n) = 0 && \text{(A.11)}\\
\Leftrightarrow\quad & \forall w \in W : \lim_{n\to\infty} < T(v_n - \tilde{g}), w >= 0 && \text{(A.12)}\\
\Leftrightarrow\quad & \forall w \in W : \lim_{n\to\infty} < v_n - \tilde{g}, T^*(w) > && \text{(A.13)}\\
\Leftrightarrow\quad & \forall w \in W :< v_\infty - \tilde{g}, T^*(w) > && \text{(A.14)}\\
\Leftrightarrow\quad & \forall w \in W :< T(v_\infty - \tilde{g}), w >= 0 && \text{(A.15)}\\
\Leftrightarrow\quad & L(v_\infty) = 0\ . && \text{(A.16)}
\end{aligned}$$

□

This theorem motivates further definitions:

Definition A.0.9: Extended Fiber
Let V, W be Banach spaces and $f : V \to W$ then

$$exfib(f, y) := \{v_\infty | \exists (v_n)_{n\in\mathbb{N}} \subset V : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} f(v_n) = y\} \tag{A.17}$$

Definition A.0.10: Minmal Extended Fiber
A fiber is minimal extended if

$$fib(f, y) = exfib(f, y) \tag{A.18}$$

Lemma A.0.11:
Let D, H, W be Hilbert spaces and $T : D \to H$ be a linear operator and $g \in H$ such that $L := f \mapsto T(f) - g$ has a minimal extended kernel. Then $exfib(T, g)$ is minimal extended.

Proof.

$$\begin{aligned}
exfib(T,g) &= \{v_\infty | \exists (v_n)_{n\in\mathbb{N}} \subset V : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} T(v_n) = g\} \\
&= \{v_\infty | \exists (v_n)_{n\in\mathbb{N}} \subset V : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} T(v_n) - g = 0\} \\
&= \{v_\infty | \exists (v_n)_{n\in\mathbb{N}} \subset V : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} L(v_n) = 0\} \\
&= exker(L) \\
&= ker(L) = \{v | L(v) = 0\} = \{v | T(v) - g = 0\} \\
&= \{v | L(v) = 0\}
\end{aligned}$$

□

Lemma A.0.12: Extended Kernel by Parts Construction

Let B be a Banach space, H be a Hilbert space, $(e_i)_{i\in\mathbb{N}}$ a complete orthonormal sequence of H and $f : B \to H$ a function with non empty extended Kernel. Then

$$exker(f) = \bigcap_{i\in\mathbb{N}} exker(v \mapsto < f(v), e_i >) \tag{A.19}$$

Proof.

$$\begin{aligned}
exker(f) &= \{v_\infty | \exists (v_n)_{n\in\mathbb{N}} \subset B : \lim_{n\to\infty} v_n = v_\infty \wedge \lim_{n\to\infty} f(v_n) = 0\} && \text{(A.20)} \\
&= \{v_\infty | ... \wedge \lim_{n\to\infty} \sum_{i\in\mathbb{N}} < f(v_n), e_i > e_i = 0\} && \text{(A.21)} \\
&= \{v_\infty | ... \wedge \forall i \in \mathbb{N} : \lim_{n\to\infty} < f(v_n), e_i >= 0\} && \text{(A.22)} \\
&= \bigcap_{i\in\mathbb{N}} \{v_\infty | ... \wedge \lim_{n\to\infty} < f(v_n), e_i >= 0\} && \text{(A.23)} \\
&= \bigcap_{i\in\mathbb{N}} exker(v \mapsto < f(v), e_i >) && \text{(A.24)}
\end{aligned}$$

□

Index

Bibliography

[1] Lawrence C. Evans, Partial Differential Equations (2002), The American Mathematical Society, Graduate Studies in Mathematics Volume 19

[2] David. G. Luenberger, Optimization by Vector Space Methods, John Wiley & Sons Inc 1969

[3] Manfred Dobrowolski, Angewandte Funktionalanalysis, Springer-Verlag Berlin-Heidelberg 2006

[4] Peter Knabner, Lutz Angermann, Numerik partieller Differentialgleichungen, Springer-Verlag Berlin Heidelberg New York 2000

[5] Michael Struwe, Variational Methods (Fourth Edition), Springer-Verlag Berlin Heidelberg 2008

[6] Otto Forster, Analysis 2 (5. Auflage), Friedr. Vierweg & Sohn Verlagsgesellschaft mbH 1999

[7] Dirk Werner, Funktionalanalysis (5. Erweiterte Auflage), Springer-Verlag Berlin-Heidelberg 2005

[8] Tamara G. Kolda, Robert M. Lewis, Virginia Torczon, Optimization By Direct Search: New Perspectives on Some Classical and Modern Methods, SIAM REVIEW Vol 45, NO 3 pp 385-482. Society for Industrial and Applied Mathematics

[9] Alexandre Ern, Jean-Luc Guermond, Theory and Practice of Finite Elements, Springer-Verlag New York 2004

[10] O. C. Zienkiewicz, R. L. Taylor, J.Z. Zhu, The Finite Element Method Its Basis & Fundamentals (6th Edition),Elsevier Ltd, 2005

[11] H. T. Rathod, B. Venkatesudu, K. V. Nagaraja, Gauss Legendre Quadrature Formulas over a Tetrahedron, Wiley InterScience (www.interscience.wiley.com) 10.06.2005

[12] Patrick Keast, Moderate-degree tetrahedral quadrature formulas, Computer Methods in Applied Mechanics and Engineering, Volume 55, Issue 3, May 1986

[13] David Cox, John Little, Donal O'Shea, "Ideals, Varieties, And Algorithms" Third Edition, Springer Science+Business Media 2007

[14] Michael Reed, Barry Simon, Methods of Modern Mathematical Physics I: Functional Analysis, Revised and Enlarged Edition, Academic Press 1980

[15] Imre Lakatos, Proofs and Refutations, Cambridge University Press, 1976

[16] Dong C. Liu, Jorge Nocedal, On the Limited Memory BFGS Method for Large Scale Optimization, Mathematical Programming (45) 1989, p. 503-528

[17] Jonathan Barzilai, Jonathan M. Borwein, Two-Point Step Size Gradient Method, IMA Journal of Numerical Analysis (1988), p. 141-148

[18] Marcos Raydan, On the Barzilai and Borwein Choice of Steplength for the Gradient Method, October 1991, IMA Journal of Numerical Analysis Volume13, Issue3, Pp. 321-326.

[19] Yasushi Narushima, Takahiko Wakamatsu, Hiroshi Yabe, Extended Barzilai-Borwein method for unconstrained minimization problems, August 17, 2007, www.optimization-online.org

[20] ABAQUS Inc, Theory Manual ABAQUS Version 6.4, 2003

[21] Patricia D. Hough, Tamara G. Kolda, Virginia J. Torczon, Asynchronous Parallel Pattern Search for Nonlinear Optimization, SIAM Journal on Scientific Computing, Vol. 23 No. 1 pp 134-156, Society for Industrial and Applied Mathematics 2001

[22] J. E. Dennis, Jr. and Virginia Torczon, Direct Search Methods On Parallel Machines, SIAM Journal on Optimization Vol. 1, p. 448-474, 1991