TECHNISCHE
UNIVERSITÄT
WIEN

DIPLOMARBEIT

Robust Algebraic Solvers for Electromagnetics

ausgeführt am

Institut für
Analysis und Scientific Computing
TU Wien

unter der Anleitung von

Univ.Prof. Dipl.-Ing. Dr.techn. Joachim SCHÖBERL

durch

Bernd SCHWARZENBACHER, BSc
Matrikelnummer: 01326919
Springergasse 3 / 2a
1020 Wien

Wien, am 12. Februar 2020

Kurzfassung

Ein Algorithmus zur effizienten Lösung der mit Nédélec-Elementen diskretisierten Maxwell-Gleichungen im magnetostatischem Regime wird präsentiert. Er basiert auf einer Algebraischen Mehrgitter (AMG) Methode, die als Vorkonditionierer für die Methode der konjugierten Gradienten verwendet wird. Eine entscheidende Komponente ist die Prolongation von [RS02], welche den Kern des curl-Operators korrekt auf den gröberen Gittern erhält. Diese Prolongation wird außerdem mit Techniken angelehnt an [BGH$^+$03] geglättet um besser Konvergenz und Robustheit im Regularisierungsparameter zu erhalten. Der Beitrag dieser Arbeit ist ein neuer Vergröberungsalgorithmus, der zu einer verbesserten Robustheit für große Sprünge in der Permeabilität führt.

Abstract

An algorithm to efficiently solve the magnetostatic case of Maxwell's equations discretized by Nédélec elements [Né86] is presented. It is based on an algebraic multigrid (AMG) method used as a preconditioner to the conjugate gradient method. One main component is the prolongation proposed in [RS02] to properly treat the kernel of the curl operator. This prolongation is then smoothed with techniques similar to [BGH$^+$03] to obtain better convergence and robustness in the regularization parameter of the magnetostatic problem. The main contribution of this thesis is to obtain improved robustness with respect to big jumps in permeability by introducing a new coarsening algorithm.

Acknowledgment

First and foremost I want to thank my supervisor Joachim Schöberl for the years of support.

I am very grateful to all my study colleagues for the friendships we developed during studying at the Technische Universität Wien. Special thanks go to two of them, Daniel Herold and Tobias Danczul. They stepped in the last minute to proofread this work.

Thanks also go to Matthias Hochsteger for providing the `Python` bindings and helping with memory optimizations.

Finally, I want to thank my family for always supporting me throughout my education.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 12. Februar 2020

Bernd Schwarzenbacher

Contents

List of Symbols

$\mathbb{1}$	one vector
$\boldsymbol{A}$	magnetic vector potential
$\boldsymbol{B}$	magnetic induction field, magnetic flux
$\boldsymbol{\lambda}$	Lagrange parameters
$\boldsymbol{H}$	magnetic field intensity
$\boldsymbol{j}$	current density
$.^T$	transpose of a vector, matrix or an operator
$\mathrm{cond}(A)$	condition number of a matrix A
$\boldsymbol{D}$	electric displacement field
$\boldsymbol{E}$	electric field intensity
ε	electric permittivity
$\mathcal{E}$	set of edges
$\mathcal{F}$	set of faces
γ_1	minimal eigenvalue
γ_2	maximal eigenvalue
$L_2(\Omega)$	space of square-integrable functions over Ω
μ	magnetic permeability
$\boldsymbol{n}$	unit normal vector
$\mathcal{N}_0^I(K)$	Space of Nédélec edge elements of first kind and order 0
Ω	domain
$\partial\Omega$	Boundary of the domain
ρ	charge density
σ	electrical conductivity
$C^\infty(\Omega)$	space of infinite differentiable functions over Ω
e	edge
F	face
h	mesh size, discretization parameter
I	identity matrix
K	element
w_e	edge weight
w_F	face weight
DOF	degree of freedom
EVP	eigenvalue problem
SPD	sparse positive definite
SPSD	sparse positive semi-definite

1. Introduction

In this thesis we are concerned with numerically solving the magnetostatic case of Maxwell's equations. The equations can be stated in a general form as

$$\operatorname{curl}\operatorname{curl}\boldsymbol{u} + \kappa\boldsymbol{u} = \boldsymbol{j}_i,$$

with appropriate boundary conditions. Since the magnetostatic case with $\kappa = 0$ is ill-posed, we need to regularize the problem with a small $\kappa > 0$. The finite element discretization for such problems is typically done with so called Nédélec edge elements [Né86]. To solve big linear systems of equations arising from the discretization, typically the preconditioned conjugate gradient method is employed. With typical Jacobi-like preconditioners, the method leads to a condition number of [Zag06]

$$\operatorname{cond}(C^{-1}A) = \frac{1}{h^2\kappa}.$$

[Hip99] and [AFW00] proposed smoothers to make geometric multigrid methods also robust in the model parameter κ. Of essential consideration is that the exact sequence property of the de Rham complex is conserved on all levels of the multigrid hierarchy. To retain the same property in the Algebraic Multigrid setting [RS02] introduced a prolongation honoring the exact sequence property. We present an Algebraic Multigrid method based on this prolongation. To further improve the condition number we adapt the smoothed prolongation proposed in [BGH$^+$03].

Our main contribution is to introduce a new coarsening algorithm, based on computing collapse weights. The main point is to demonstrate improved convergence for cases with big jumps in the parameters. Those jumps in parameters can occur for example in setups where an electromagnetic coil has an iron core. The magnetic permeability in such cores is magnitudes bigger then the permeability in air. In the numerical tests for this thesis we apply the proposed preconditioner to such examples and show its performance and robustness.

Implementation

All implementations were done in `C++` on top of Netgen/NGSolve[1] [Sch97][Sch14] and exposed as a `Python` library to be used together with Netgen/NGSolve.

[1] `https://ngsolve.org`

2. Fundamental Electromagnetics

We mostly follow the introduction in [Zag06]. For a related treatment with a focus on inverse problems see [Mon03].

2.1. Maxwell's Equations

Electromagnetic phenomena are modeled by Maxwell's equations, introduced by and named after Clark Maxwell. The equations describe the relation between the electric field and the magnetic field. They are given as the electric and magnetic field intensities $\boldsymbol{E}$ and $\boldsymbol{H}$ respectively. Further useful are the electric displacement field $\boldsymbol{D}$ (also called the electric flux) as well as the magnetic induction field $\boldsymbol{B}$ (also called the magnetic flux). We give their relationship by the Maxwell's equations in differential form:

Faraday's law of induction describes how a change in the magnetic field produces an electric field

$$\operatorname{curl} \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}.$$

Ampère's circuital law describes an analogous reverse statement. A change in the electric field or a current will produce a magnetic field

$$\operatorname{curl} \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} + \boldsymbol{j}_t,$$

where $\boldsymbol{j}$ is the current density.

Gauß' law describes that charges (given as a scalar field of current density ρ) are sources (positive charge) and sinks (negative charge) of electric fields:

$$\operatorname{div} \boldsymbol{D} = \rho.$$

Gauß' law for magnetostatics describes that the magnetic field is divergence free, meaning there are no sources for the magnetic field. Vector fields with this property are also called solenoidal fields.

$$\operatorname{div} \boldsymbol{B} = 0$$

To relate the intensities and fluxes of the electric and magnetic fields we can use the following material equations, where ε is the permittivity and μ is the permeability of the material:

$$\boldsymbol{D} = \varepsilon \boldsymbol{E}$$

$$\boldsymbol{B} = \mu \boldsymbol{H}$$

Furthermore the total current density $\boldsymbol{j}_t$ is given as a sum by the conduction current $\boldsymbol{j}_c$ and the impressed current $\boldsymbol{j}_i$:

$$\boldsymbol{j}_t = \boldsymbol{j}_c + \boldsymbol{j}_i$$

The impressed current is introduced by some external current source and the conduction current is a result of the potential difference and is obtained by **Ohm's law** as

$$\boldsymbol{j}_c = \sigma \boldsymbol{E},$$

where σ is the electrical conductivity.

In this thesis we only concentrate on cases where the material parameters ε, μ and σ are time independent, isotropic, and bounded. Thus they can be modeled by scalar fields over the bounded domain Ω and are elements of $L^\infty(\Omega)$.

2.1.1. Interface Conditions

It is instructive for the further treatment to understand the behavior of the introduced fields across interfaces of domains. Considering a simply-connected domain $\Omega \subset \mathbb{R}^3$ split into two disjoint domains Ω_1 and Ω_2, we are interested in the behavior of the fields $\boldsymbol{E}$ and $\boldsymbol{B}$ across the shared interface $\Gamma \coloneqq \overline{\Omega}_1 \cap \overline{\Omega}_2$. Here $\boldsymbol{n}_\Gamma$ is the unit normal vector on Γ pointing from Ω_2 to Ω_1 and $\boldsymbol{B}_i$ is the field $\boldsymbol{B}$ restricted to the respective volume Ω_i. First we use the integral formulation of Gauß' law to derive a condition for the jump term $[\boldsymbol{B} \cdot \boldsymbol{n}_\Gamma] = (\boldsymbol{B}_2 - \boldsymbol{B}_1) \cdot \boldsymbol{n}_\Gamma$:

$$\begin{aligned} 0 &= -\int_{\partial\Omega} \boldsymbol{B} \cdot \boldsymbol{n} \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_1} \boldsymbol{B}_1 \cdot \boldsymbol{n} \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_2} \boldsymbol{B}_2 \cdot \boldsymbol{n} \,\mathrm{d}\boldsymbol{x} \\ &= -\int_{\partial\Omega_1 \cap \Gamma} \boldsymbol{B}_1 \cdot \boldsymbol{n}_\Gamma \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_2 \cap \Gamma} \boldsymbol{B}_2 \cdot \boldsymbol{n}_\Gamma \,\mathrm{d}\boldsymbol{x} \\ &= \int_\Gamma [\boldsymbol{B} \cdot \boldsymbol{n}_\Gamma] \,\mathrm{d}\boldsymbol{x} \end{aligned}$$

Since this derivation holds for an arbitrary interface Γ between arbitrary volumes V_1, V_2, we can conclude that the normal component for the magnetic flux $\boldsymbol{B}$ across interfaces needs to be continuous.

A similar derivation is used for the electric field $\boldsymbol{E}$. We use Faraday's law in integral form with a surface Σ crossing the interface Γ. The interface splits Σ into two disjoint surfaces Σ_1 and Σ_2. Here $\boldsymbol{\tau}$ denotes a tangential vector along respective boundaries and $L \coloneqq \partial\Sigma_1 \cap \partial\Sigma_2$ is the boundary curve shared by Σ_1 and Σ_2. Again we obtain a condition for the corresponding jump term $[\boldsymbol{E} \cdot \boldsymbol{\tau}_L] = (\boldsymbol{E_2} - \boldsymbol{E_1}) \cdot \boldsymbol{\tau}_L$:

$$\begin{aligned} 0 &= -\int_{\partial\Sigma} \boldsymbol{E} \cdot \boldsymbol{\tau} \,\mathrm{d}s + \int_{\partial\Sigma_1} \boldsymbol{E}_1 \cdot \boldsymbol{\tau}_1 \,\mathrm{d}s + \int_{\partial\Sigma_2} \boldsymbol{E}_2 \cdot \boldsymbol{\tau}_2 \,\mathrm{d}s \\ &= \int_{\partial\Sigma_1 \cap L} \boldsymbol{E}_1 \cdot \boldsymbol{\tau}_1 \,\mathrm{d}s + \int_{\partial\Sigma_2 \cap L} \boldsymbol{E}_2 \cdot \boldsymbol{\tau}_2 \,\mathrm{d}s \\ &= \int_L [\boldsymbol{E} \cdot \boldsymbol{\tau}_L] \,\mathrm{d}s \end{aligned}$$

Now this implies, that the tangential component of the electric field needs to be continuous across interfaces.

2.1.2. Boundary Conditions

To obtain a boundary value problem we still need boundary conditions. For reference we only note the two simplest types: Neumann- and Dirichlet-type. For other kind of boundary conditions see [Zag06].

Neumann-type boundary conditions correspond to so called *perfect electric conductors*.

$$\boldsymbol{E} \times \boldsymbol{n} = 0 \qquad \text{on } \Gamma_N$$

Dirichlet-type boundary conditions correspond to so called *perfect magnetic conductors*.

$$\boldsymbol{H} \times \boldsymbol{n} = 0 \qquad \text{on } \Gamma_D$$

2.2. Vector Potential Formulation

The formulation we will use depends on the representation of $\boldsymbol{B}$ by a vector potential $\boldsymbol{B} = \operatorname{curl} \boldsymbol{A}$ with *Coulamb-gauging* $\operatorname{div} \boldsymbol{A} = 0$. Further we can choose $\boldsymbol{A}$ such that $\boldsymbol{E} = -\frac{\partial \boldsymbol{A}}{\partial t}$. Thus we arrive at the *vector potential formulation* of Maxwell's equations:

$$\operatorname{curl} \mu^{-1} \operatorname{curl} \boldsymbol{A} + \sigma \frac{\partial \boldsymbol{A}}{\partial t} + \varepsilon \frac{\partial^2 \boldsymbol{A}}{\partial t^2} = \boldsymbol{j}_i$$

For the details of the derivation see [Zag06].

The thesis is mainly concerned with the magnetostatic case. In this regime Maxwell's equations simplify to

$$\operatorname{curl} \boldsymbol{H} = \boldsymbol{j}_i, \quad \operatorname{div} \boldsymbol{B} = 0, \quad \text{and } \boldsymbol{B} = \mu \boldsymbol{H}. \tag{2.1}$$

With the mentioned vector potential $\boldsymbol{A}$ we obtain the simplified *magnetostatic vector potential problem*

$$\operatorname{curl} \mu^{-1} \operatorname{curl} \boldsymbol{A} = \boldsymbol{j}_i. \tag{2.2}$$

A similar argument as in Section 2.1.1 leads to the essential boundary conditions

$$\boldsymbol{A} \times \boldsymbol{n} = 0 \qquad \text{on } \Gamma_B,$$

which imply that the magnetic flux through the boundary is zero:

$$\Rightarrow \operatorname{div}(\boldsymbol{A} \times \boldsymbol{n}) = \operatorname{curl} \boldsymbol{A} \cdot \boldsymbol{n} - \boldsymbol{A} \cdot \underbrace{\operatorname{curl} \boldsymbol{n}}_{=0} = \boldsymbol{B} \cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_B.$$

The natural boundary conditions

$$\mu^{-1} \operatorname{curl} \boldsymbol{A} \times \boldsymbol{n} = -\boldsymbol{j}_S \qquad \text{on } \Gamma_H$$

model impressed surface currents

$$\boldsymbol{H} \times \boldsymbol{n} = -\boldsymbol{j}_S \qquad \text{on } \Gamma_H.$$

Other regimes like time-harmonic, eddy-current, and time-stepping formulations lead to similar equations, which can be summarized in the general curl-curl problem:

Problem 1. Let $\Omega \subset \mathbb{R}^3$ be a bounded domain with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$. Find $\boldsymbol{u}$ such that

$$\operatorname{curl} \mu^{-1} \operatorname{curl} \boldsymbol{u} + \kappa \boldsymbol{u} = \boldsymbol{f} \qquad \text{in } \Omega \subset \mathbb{R}^3, \tag{2.3}$$

$$(\boldsymbol{u} \times \boldsymbol{n}) \times \boldsymbol{n} = g_D \qquad \text{on } \Gamma_D, \tag{2.4}$$

$$\mu^{-1} \operatorname{curl} \boldsymbol{u} \times \boldsymbol{n} = g_N \qquad \text{on } \Gamma_N. \tag{2.5}$$

From this, the magnetostatic case is obtained again by setting $\kappa = 0$. We will see that this leads to an ill-posed problem. This issue can be alleviated by regularizing with a small contribution of the L^2-term. So effectively we work with $0 < \kappa \ll 1$.

3. Finite Element Framework

3.1. Variational Formulation

As a next step we show, that the Problem 1 is actually solvable. Again we mostly follow [Zag06].

First we put a restriction on the domain we use. Specifically, it should be a bounded Lipschitz domain as defined below.

Definition 1. (Lipschitz domain [Zag06]) The boundary of a domain $\Omega \subset \mathbb{R}^3$ is called *Lipschitz continuous* if there exists a finite number of domains ω_i, local coordinate systems (ξ_i, η_i, ζ_i), and Lipschitz continuous functions $b(\xi_i, \eta_i)$ such that

- $\partial\Omega \subset \bigcup \omega_i$ with $\partial\Omega \cap \omega_i = \big\{(\xi_i, \eta_i, \zeta_i) \mid \zeta_i = b(\xi_i, \eta_i)\big\}$,
- $\Omega \cap \omega_i = \big\{(\xi_i, \eta_i, \zeta_i) \mid \xi_i > b_i(\xi_i, \eta_i)\big\}$

Ω is then called a *Lipschitz domain*.

So the boundary $\partial\Omega$ is piecewise Lipschitz continuous and the domain lies on one side of it. For such domains we get a few important properties:

- For every point on the boundary we can define an outward pointing unit normal vector $\boldsymbol{n}$ almost anywhere. (See [Mon03] and references therein.)
- The following results hold [GR86]:

$$H^1(\Omega) = \overline{C^\infty(\overline{\Omega})}^{\|\cdot\|_1}, \quad H(\mathrm{curl}, \Omega) = \overline{C^\infty(\overline{\Omega})}^{\|\cdot\|_{\mathrm{curl}}}, \quad H(\mathrm{div}, \Omega) = \overline{C^\infty(\overline{\Omega})}^{\|\cdot\|_{\mathrm{div}}}$$

Here $C^\infty(\overline{\Omega})$ is the space of infinitely differentiable functions over the closure of Ω and the other spaces are defined as

$$\begin{aligned} H^1(\Omega) &= \{v \in L_2(\Omega) \mid \mathrm{grad}\, v \in [L_2(\Omega)]^3\}, \\ H(\mathrm{curl}, \Omega) &= \{\boldsymbol{v} \in [L_2(\Omega)]^3 \mid \mathrm{curl}\, \boldsymbol{v} \in [L_2(\Omega)]^3\}, \\ H(\mathrm{div}, \Omega) &= \{\boldsymbol{v} \in [L_2(\Omega)]^3 \mid \mathrm{div}\, \boldsymbol{v} \in L_2(\Omega)\}, \end{aligned}$$

where the norms $\|\cdot\|_1$, $\|\cdot\|_{\mathrm{curl}}$, and $\|\cdot\|_{\mathrm{div}}$ are induced by the corresponding scalar products:

$$\begin{aligned} (u, v)_1 &= \int_\Omega \mathrm{grad}\, u \cdot \mathrm{grad}\, u \,\mathrm{d}\boldsymbol{x} + \int_\Omega u \cdot v \,\mathrm{d}\boldsymbol{x} \\ (\boldsymbol{u}, \boldsymbol{v})_{\mathrm{curl}} &= \int_\Omega \mathrm{curl}\, \boldsymbol{u} \cdot \mathrm{curl}\, \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} + \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} \\ (\boldsymbol{u}, \boldsymbol{v})_{\mathrm{div}} &= \int_\Omega \mathrm{div}\, \boldsymbol{u} \cdot \mathrm{div}\, \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} + \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} \end{aligned}$$

Theorem 1 (Trace theorem and integration by parts in $H(\mathrm{curl},\Omega)$)**.** *Let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz-domain.*

1. *The classical trace map*

$$\mathrm{tr}_{\boldsymbol{\tau}}(\boldsymbol{v})(\boldsymbol{x}) \coloneqq \boldsymbol{v}(\boldsymbol{x}) \times \boldsymbol{n}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Gamma$$

can be extended from $[C^\infty(\overline{\Omega})]^d$ to a continuous and linear map (still denoted by $\mathrm{tr}_{\boldsymbol{\tau}}$)

$$\mathrm{tr}_{\boldsymbol{\tau}} : H(\mathrm{curl},\Omega) \to [H^{-1/2}(\partial\Omega)]^3, \quad \textit{and} \quad \left\|\mathrm{tr}_{\boldsymbol{\tau}}(\boldsymbol{v})\right\|_{-1/2} \preceq \|\boldsymbol{v}\|_{\mathrm{curl}} \, \forall \boldsymbol{v} \in H(\mathrm{curl},\Omega),$$

2. *There holds the integration by parts formula*

$$\int_\Omega \mathrm{curl}\,\boldsymbol{u} \cdot \boldsymbol{\varphi}\,\mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{u} \cdot \mathrm{curl}\,\boldsymbol{\varphi}\,\mathrm{d}\boldsymbol{x} - \int_\Gamma \mathrm{tr}_{\boldsymbol{\tau}}(\boldsymbol{u}) \cdot \boldsymbol{\varphi}\,\mathrm{d}\boldsymbol{x} \quad \forall \boldsymbol{x} \in H(\mathrm{curl},\Omega), \forall \boldsymbol{\varphi} \in [H^1(\Omega)]^3 \tag{3.1}$$

Proof. See [Zag06] Theorem 3.6 and [Mon03] Theorem 3.29. □

With this prerequisites we choose to look for solutions to Problem 1 in $H(\mathrm{curl},\Omega)$. First we multiply Equation (2.3) with a test function $v \in H(\mathrm{curl},\Omega)$, integrate over the domain and then apply the integration by parts formula (Equation (3.1)) to arrive at:

Problem 2. Find $\boldsymbol{u} \in H(\mathrm{curl},\Omega)$

$$\int_\Omega \mu^{-1}\,\mathrm{curl}\,\boldsymbol{u} \cdot \mathrm{curl}\,\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \int_\Omega \kappa \boldsymbol{u} \cdot \boldsymbol{v}\,\mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{j} \cdot \boldsymbol{v}\,\mathrm{d}\boldsymbol{x} \qquad \forall \boldsymbol{v} \in H(\mathrm{curl},\Omega) \tag{3.2}$$

and appropriate boundary conditions.

Then we can prove, that Problem 2 gives rise to a continuous and coercive bilinear form on $H(\mathrm{curl},\Omega)$ and a continuous linear functional on the right hand side. So following from Theorem 2, the problem admits a unique solution in $H(\mathrm{curl},\Omega)$.

Theorem 2 (Lax-Milgram)**.** *Given a Hilbert space $(V,(\cdot,\cdot))$, a continuous, coercive bilinear form $a(\cdot,\cdot)$ and a continuous linear functional $F \in V'$, there exists a unique $\boldsymbol{u} \in V$ such that*

$$a(\boldsymbol{u},\boldsymbol{v}) = F(\boldsymbol{v}), \qquad \forall \boldsymbol{v} \in V. \tag{3.3}$$

Proof. See [Bre] Theorem (2.7.7). □

3.2. The De Rham Complex

A very useful tool in the context of computational electromagnetics is the *de Rham complex*:

$$\mathbb{R} \xrightarrow{\mathrm{id}} H^1(\Omega) \xrightarrow{\mathrm{grad}} H(\mathrm{curl},\Omega) \xrightarrow{\mathrm{curl}} H(\mathrm{div},\Omega) \xrightarrow{\mathrm{div}} L^2(\Omega) \to 0. \tag{3.4}$$

Theorem 3. *The chain in Equation (3.4) forms an exact sequence, meaning that the range of each operator on the left of a space is exactly the kernel of the operator to the right of the same space.*

Proof. See [Zag06][Mon03] and the references therein. □

The theorem holds true for simply connected domains and also with Dirichlet boundary conditions. For other boundary conditions and non-simply connected domains, the exact sequence property is violated by a finite dimensional subspace, the so called *cohomology space* [Mon03]. We will see, that it is a guiding principle to try to honor the exact sequence when deriving subspace and discrete operators. For a modern treatment see [AFW06].

The most important part of the de Rham complex for electromagnetics is the space $H(\mathrm{curl}, \Omega)$ and the surrounding operators:

$$\mathrm{grad}\, H^1(\Omega) = \ker(H(\mathrm{curl}, \Omega)), \tag{3.5}$$

where the classical analog is a well-known vector calculus identity.

A closely related property is the Helmholtz decomposition, which tells us, that a vector field can be represented by the *curl* of another vector field and the gradient of a scalar field.

Theorem 4 (Helmholtz decomposition)**.** *Every vector field* $\boldsymbol{u} \in L_2(\Omega)^3$ *admits an orthogonal decomposition*

$$\boldsymbol{u} = \mathrm{grad}\, \varphi + \mathrm{curl}\, \boldsymbol{\psi}, \quad \textit{with } \varphi \in H^1(\Omega) \textit{ and } \boldsymbol{\psi} \in H(\mathrm{curl}, \Omega)$$

Proof. See [Zag06] and the references therein. □

3.3. Discretization

In this thesis we concern ourselves only with meshes discretized by tetrahedrons and lowest order elements. For related elements on other primitives and also higher order elements see [Zag06]. Therefore only the Nédélec Element of first kind [Né86] of order 0 is presented. Often they are referred to as edge elements.

Definition 2 (Nédélec element of first kind of order 0 [Zag06])**.** The lowest order edge-element on a tetrahedron K is given by

- the local space $\mathcal{N}_0^I(K)$ defined as

$$\mathcal{N}_0^I(K) \coloneqq \{\boldsymbol{a} + \boldsymbol{b} \times \boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3\} \quad \text{with } \dim(\mathcal{N}_0^I(K)) = 6.$$

- the *edge-based* degrees of freedom:

$$N_\alpha^{\mathcal{N}_0} : \boldsymbol{v} \to \int_{e_\alpha} \boldsymbol{v} \cdot \boldsymbol{\tau}\, \mathrm{d}\boldsymbol{x} \quad \text{for } \alpha = 1, \ldots, \left|\right\rceil_K\left|\right.$$

  i.e. the line integrals of the tangential component over each edge $e_\alpha \in \mathcal{E}_K$.

Since the degrees of freedom of these elements are tangential components we can construct tangential continuous fields. Therefor the finite element space, spanned by these elements over a discretization, forms a conforming subspace $\boldsymbol{V_h} \subset H(\mathrm{curl}, \Omega)$ [Zag06]. This

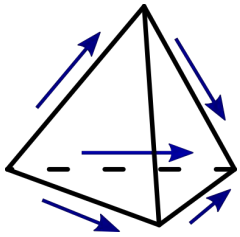


Figure 3.1.: Nédélec Element type 0 for a tetrahedron

association of degrees of freedom with the tangential components of the edges is schematically depicted in Figure 3.1.

With the same reasoning as in the previous chapter, we obtain a unique solution of Problem 2 where $V = \boldsymbol{V_h}$. The viability of the discretization is shown with Céa's lemma. It bounds the discretization error by the approximation error. We can directly apply this lemma to our situation with $V = H(\mathrm{curl}, \Omega)$ and $\boldsymbol{V_h}$.

Theorem 5 (Céa's Lemma)**.** *Suppose the same conditions for the Hilbert space $(V, (\cdot,\cdot))$ and the bilinear form $a(\cdot,\cdot)$ hold as in the Lax-Milgram theorem. Furthermore V_h is a closed subspace of V. Then we have*

$$\|u - u_h\|_V \leq \frac{C}{\alpha} \min_{v_h \in V_h} \|u - v_h\|_V, \tag{3.6}$$

where C is the continuity constant and α is the coercivity constant of $a(\cdot,\cdot)$ on V.

Proof. See [Bre] Theorem (2.8.1). □

We summarize the other used spaces in the global discrete sequence [Zag06, Theorem 4.28 and chapter 4.3]

$$\mathbb{R} \xrightarrow{\mathrm{id}} \boldsymbol{Q}_h(\Omega) \xrightarrow{\mathrm{grad}} \boldsymbol{V}_h \xrightarrow{\mathrm{curl}} \boldsymbol{W}_h \xrightarrow{\mathrm{div}} \boldsymbol{S}_h(\Omega) \to 0, \tag{3.7}$$

where $\boldsymbol{Q}_h$ is the FE-space spanned by continuous first order nodal elements, $\boldsymbol{W}_h$ is the FE-space spanned by lowest order Raviart-Thomas elements, and $\boldsymbol{S}_h$ is the space spanned by discontinuous lowest order elements. All mentioned finite element spaces are conforming subspaces of the corresponding spaces in equation (3.4).

4. Iterative Solving of Maxwell's Equations

With a suitable finite element subspace we arrive by the Galerkin approximation at the linear system

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad \text{with } \boldsymbol{b} \in \mathbb{R}^n \text{ given.} \tag{4.1}$$

Typical direct solvers like LU- or Cholesky-decompositions to compute the inverse A^{-1} have an asymptotic behavior for the amount of work of $\mathcal{O}(n^3)$, where n is the number of degrees of freedom of our problem. Therefor one resorts to iterative methods to solve problems with a high number of degrees of freedom, since they often provide better asymptotic behavior in cases where they are applicable. A short discussion of this motivation can be found in the introductions of [Hac16, Chapter 1.5] and [vdV03, Chapter 1].

4.1. Conjugate Gradient Method

For a symmetric positive definite (SPD) matrices the first choice for an iterative method is the conjugate gradient (CG) method.

Theorem 6. *Conjugate Gradient Convergence. Let A be SPD with the minimal and maximal eigenvalues $\gamma_1 := \gamma_{\min}(A)$, $\gamma_2 := \gamma_{\max}(A)$. We abbreviate the spectral condition number by $\operatorname{cond}(A) = \gamma_2/\gamma_1$. The errors $\boldsymbol{e}^m = \boldsymbol{x} - \boldsymbol{x}^m$ (where $\boldsymbol{x}$ is the solution to Equation (4.1)) of the CG iterates $\boldsymbol{x}^m$ after m steps satisfy the estimate*

$$\|\boldsymbol{e}^m\|_A \leq \frac{2c^m}{1+c^{2m}}\|\boldsymbol{e}^0\|_A \tag{4.2}$$

with $c := \frac{\sqrt{\operatorname{cond}(A)}-1}{\sqrt{\operatorname{cond}(A)}+1} = \frac{\sqrt{\gamma_{\max}}-\sqrt{\gamma_{\min}}}{\sqrt{\gamma_{\max}}+\sqrt{\gamma_{\min}}}$.

Proof. See [Hac16, Theorem 10.14]. □

The convergence thus only depends on the spectral condition number. To improve the convergence, the spectral condition number of the problem can be tweaked by applying preconditioners as described in the next chapter.

4.2. Preconditioner

To further improve the speed of convergence one idea is to instead solve the preconditioned system

$$C^{-1}A\boldsymbol{x} = C^{-1}\boldsymbol{b}, \tag{4.3}$$

where $C^{-1}A$ hopefully has a better condition number.

A good preconditioner should fulfill the following conditions as presented in [vdV03, Chapter 13].

1. The cost for the setup of the preconditioner C shouldn't be too expensive.
2. The inverse of the preconditioner matrix C^{-1} should be a good approximation to the system matrix inverse A^{-1} in some sense.
3. The preconditioned system $C\boldsymbol{y} = \boldsymbol{z}$ should be cheaper to solve than the original equation.

We see that conditions 2) and 3) work against each other, as the optimal choice for the second condition would be the matrix A itself as the preconditioned Equation (4.3) reduces to $\boldsymbol{x} = A^{-1}\boldsymbol{b}$, but then we are back at the problem of having to invert the matrix. Contrary the ideal choice for the third condition would be the identity matrix I since it is trivially invertible.

Typical preconditioners are Jacobi and Gauß-Seidel preconditioners. They are also often called smoothers and performing one application is called smoothing or relaxation.

For Problem 1 they suffer from a condition number of

$$\operatorname{cond}(C^{-1}A) = \frac{1}{\kappa h^2}, \tag{4.4}$$

where h is a measure for the mesh size [Zag06, Section 6.2.1]. For the magnetostatic problems, with a small κ for regularization, we end up with a high condition number. So the first goal is to get rid of this dependency on κ. Preconditioners without this dependency are called κ-robust.

Definition 3. We call a preconditioner C κ-robust, if its spectral bounds are independent of the problem parameter κ, i.e.,

$$\gamma_1 C(\boldsymbol{u}, \boldsymbol{u}) \leq a(\boldsymbol{u}, \boldsymbol{u}) \leq \gamma_2 C(\boldsymbol{u}, \boldsymbol{u}). \tag{4.5}$$

Two popular preconditioners to solve this were proposed in the setting of geometric multigrid. They deal with the kernel of the curl operator in the smoother.

1. A preconditioner based on treating the kernel of $H(\operatorname{curl}, \Omega)$ [Hip99]
2. A block Gauß-Seidel preconditioner based on edge patches [AFW00]

Applying the second preconditioner directly leads to a condition number [Zag06, Section 6.2.1] of

$$\operatorname{cond}(C_{\text{Arnold}}^{-1}A) = \frac{1}{h^2}.$$

4.3. Multigrid

It was shown for various discretizations of elliptic problems that the multigrid algorithm is an optimal preconditioner. This means that the number of iterations does not increase with increasing number of degrees of freedom and the amount of work and memory is linear in the number of unknowns [Bre]. Such a solver makes a prime candidate to use as a preconditioner for the CG method as introduced Section 4.1.

4.3.1. Motivation for Multigrid

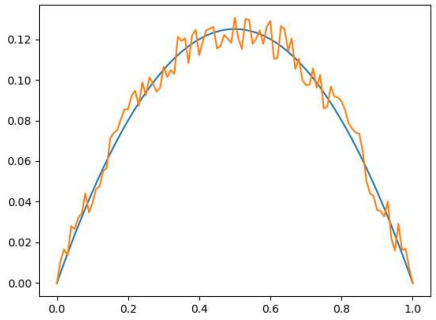
The general idea of multigrid solvers is to solve a linear system arising from a finite element discretization on several so called levels. This allows to better approximate and relax errors of frequencies corresponding to those levels. This principle is illustrated by the following example.

Problem 3. Let $I = [0, 1]$ be the unit interval. Find a function u solving Poisson's equation on I with homogeneous Dirichlet boundary conditions

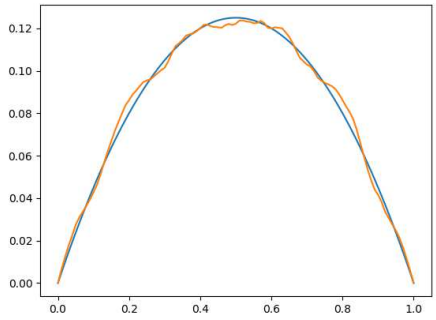
$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= 1 \qquad \text{on } I, \\ u(0) = u(1) &= 0. \end{aligned}$$

In Figures 4.1a to 4.1f we show the behavior of applying a Jacobi smoother on different initial values. The analytical solution $u(x) = \frac{1}{2}(x^2 - x)$ to Problem 3 is plotted in blue. Figure 4.1a shows the solution of the Poisson equation with an added random uniformly distributed noise of amplitude 0.01. We see that the smoother acts well in suppressing errors with frequencies similar to the mesh grid (Figure 4.1b). For an overlap of low and high frequency errors we take a look at an initial guess of a sinus function with the same uniformly distributed noise as before (Figure 4.1c). The Jacobi smoother is not able to quickly correct the low frequency error as seen in Figure 4.1d). On the other hand Figure 4.1e and Figure 4.1f show that the low frequency error is effectively corrected by the Jacobi smoother on the coarser grid.

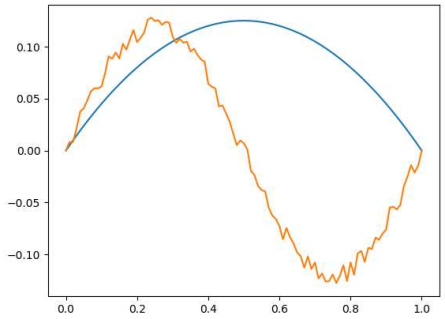
Generally smoothers are well understood and cheap to implement, but they are only able to quickly reduce errors which are in the frequency corresponding to the discretization. As demonstrated this is due to the nature of such smoothers to average across neighboring degrees of freedom.



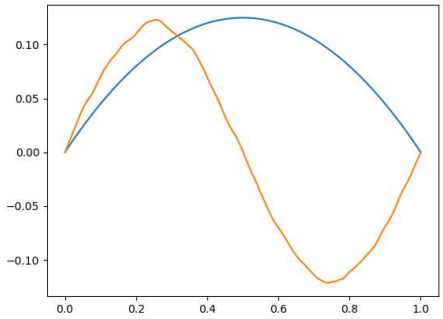
(a) Solution with added random noise.



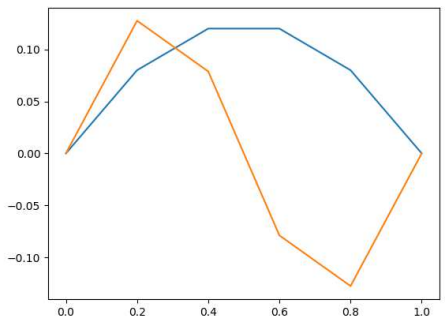
(b) Solution with added random noise after 3 steps of Jacobi smoothing.



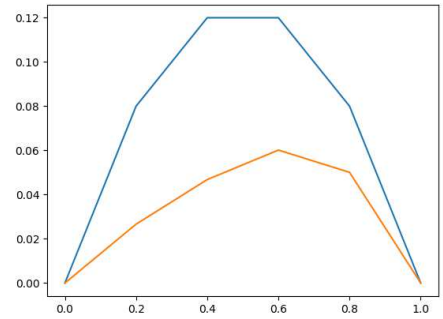
(c) Low frequency initial guess with added high frequency random noise.



(d) Low frequency initial guess with added high frequency random noise after 3 steps of Jacobi smoothing.



(e) Low frequency initial guess on a coarse grid.



(f) Low frequency initial guess on a coarse grid after 3 steps of Jacobi smoothing.

Figure 4.1.: Smoothing property of a Jacobi preconditioner on different grid resolutions.

4.3.2. General Multigrid Topics

There are two major classes of multigrid methods:

- One is the so called *geometric multigrid (GMG)*, which relies on a hierarchy of discretization meshes of the geometry. For each mesh a finite element space is constructed. The prolongation operator is directly given by this hierarchy of meshes. The hard part here is to choose an appropriate smoother for each level.
- In contrast, the *algebraic multigrid (AMG)* method doesn't rely on a hierarchy of geometrical meshes, but only uses the matrix arising from the FEM discretization of a fixed mesh size h. The hard part here is to choose appropriate prolongations which then provide a suitable hierarchy of coarse matrices where the smooth errors can be represented. On the other hand it is easier to smooth each level, since standard smoothers can be used once a matrix for the level is given.

Given a hierarchy of finite element spaces

$$V_0 \subset V_1 \subset \cdots \subset V_{l_{\max}},$$

we need two essential ingredients for any multigrid algorithm on each level l such that $0 \leq l < l_{\max}$:

- A prolongation operator $P_l : V_l \to V_{l+1}$ to map coarse grid functions to fine grid functions. The transpose $P_l^T : V_{l+1} \to V_l$ is called the restriction operator and maps fine grid functions to a coarser grid.
- A smoothing operator $S_l : V_l \to V_l$.

The setup for such multigrid methods could be as presented in Algorithm 1.

Algorithm 1 General setup for multigrid

```
1: procedure SETUP(A_l)
2:     if level l < l_max then
3:         Build prolongation P_l
4:         Restrict the system matrix A_{l-1} = (P_l)^T A_l P_l
5:         SETUP(A_{l-1})
6:     else
7:         Perform a factorization of A_l
```

For both the GMG and AMG methods, the setup can be viewed as returning a matrix C_l^{-1} to be applied as a substitution for the inverse matrix A_l^{-1}. Depending on the actual cycle used for the multigrid method we get different schemes for applying this matrix C^{-1} to a vector. For a multiplicative V-cycle multigrid loop see Algorithm 2. The inputs of the procedure are the current level l, the right hand side $\boldsymbol{f}_l$ and the return value as the solution $\boldsymbol{u}_l$ to

$$\boldsymbol{u}_l = C_l^{-1} \boldsymbol{f}_l.$$

To use this multigrid scheme as a preconditioner for the conjugate gradient method it is essential to obtain a symmetric method. So in case of using Gauß-Seidel methods for the smoothing, one needs to reverse the order of unknowns in the second smoothing step. That is why this step is sometimes called *backward smoothing*.

There are several alternatives to this specific multigrid loop. First one could use an additive instead of a multiplicative approach. In practice multiplicative approaches seem to work better.

Also instead of the V-cycle, one can employ a different cycle method. Another popular method is the W-cycle.

Algorithm 2 Multiplicative Multigrid $V(\nu_F, \nu_B)$-cycle.

procedure MULTIGRID($\boldsymbol{u}_l$, $\boldsymbol{f}_l$, l)
  if $l = l_{\max}$ **then**
    $\boldsymbol{u}_l = (A_l)^{-1}\boldsymbol{f}_l$ with a direct solver
  else
    Smooth ν_f times on $A_l\boldsymbol{u}_l = \boldsymbol{f}_l$
    Calculate the defect $\boldsymbol{d}_l = \boldsymbol{f}_l - A_l\boldsymbol{u}_l$
    Restrict the defect to the next coarser level $l+1$: $\boldsymbol{d}_{l+1} = P_l^T\boldsymbol{d}_l$
    Set $\boldsymbol{u}_{l+1} = 0$
    MULTIGRID($\boldsymbol{u}_{l+1}$, $\boldsymbol{d}_{l+1}$, $l+1$)
    Prolongate the correction $\boldsymbol{s}_l = P_l\boldsymbol{u}_{l+1}$
    Update the solution $\boldsymbol{u}_l = \boldsymbol{u}_l + \boldsymbol{s}_l$
    Smooth ν_B times on $A_l\boldsymbol{u}_l = \boldsymbol{f}_l$

4.4. Algebraic Multigrid

The AMG methods are relevant for problems where there is no hierarchy of geometries possible or available. One of the original treatments for elliptic equations can be found in e.g. [RS87].

Since there is no hierarchy of meshes for AMG one of the essential questions is how to obtain a suitable coarser representation of the system. Remembering again the typical behavior of smoothers shown in Section 4.3.1, we note that it would be beneficial to construct the coarse representations and the associated prolongation operators such that they are capable of representing so called smooth errors. In a more mathematical sense, an error is called *smooth* if it has slow convergence with respect to a smoother S [RS87]:

$$\|\boldsymbol{e}\|_1 \approx \|S\boldsymbol{e}\|_1$$

Although this concept is derived from the geometric perspective it doesn't necessarily need to correspond to a low frequency error.

The concept of strong connections was derived from this notion of smooth errors. The main idea of it is that a smooth error varies slowly in the direction of a strong connection. This concept can then be used to construct a coarsening algorithm by clustering vertices with strong connections between them into one group and then assigning this group to

one coarse vertex. A similar concept adapted to the $H(\mathrm{curl}, \Omega)$ setting is discussed in Section 5.4.

4.4.1. Reitzinger/Schöberl Prolongation

To use the AMG approach for Maxwell's equations, Reitzinger and Schöberl developed the first prolongation to treat the kernel of the curl operator correctly [RS02]. We follow their presentation for this chapter.

Let ω_l^e and ω_{l+1}^e be the set of edges on the fine and coarse level respectively and ω_l^n and ω_{l+1}^n the set of nodes on the fine and coarse level. The number of nodes on level l is N_l^n and the number of edges is N_l^e. Then the index map is defined by a coarsening algorithm as the map of fine nodes to their corresponding coarse nodes:

$$\mathrm{ind} : \omega_l^n \to \omega_{l+1}^n.$$

This implies that nodes on the fine level $i, j \in \omega_l^n$ prolongate from the same coarse node if and only if $\mathrm{ind}(i) = \mathrm{ind}(j)$. The nodal prolongation operator is thus defined as

$$(P^n)_{ij} = \begin{cases} 1 & i \in \omega_l^n, j = \mathrm{ind}(i), \\ 0 & otherwise. \end{cases} \tag{4.6}$$

It has full rank, since every fine node has one corresponding coarse node.

Definition 4 (Conforming edge prolongation [RS02])**.** The edge prolongation is defined for $i = (i_1, i_2) \in \omega_l^e, j = (j_1, j_2) \in \omega_{l+1}^e$

$$(P^e)_{ij} = \begin{cases} 1 & \text{if } j = (\mathrm{ind}(i_1), \mathrm{ind}(i_2)), \\ -1 & \text{if } j = (\mathrm{ind}(i_2), \mathrm{ind}(i_1)), \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

From the definition we see that taking the orientation of the edges into account is a key aspect. This prolongation also has full rank, since every coarse edge maps to at least one fine grid edge.

To show the usefulness of the prolongation, we first need a few concepts. The *Galerkin isomorphisms* map assignments of degrees of freedoms to actual functions in our finite element spaces:

$$G^e : V_l \to \boldsymbol{V_l}$$
$$G^n : Q_l \to \boldsymbol{Q_l}$$

with $V_l = \mathbb{R}^{N_l^e}$ and $Q_l = \mathbb{R}^{N_l^n}$. We need the Galerkin isomorphisms to describe the *discrete gradient operator* $\mathrm{grad}_l : Q_l \to V_{l,0}$, where $V_{l,0} = \{v_h \in V_h, \mathrm{curl}\, G^e v_h = 0\}$. It is given for $q_l \in Q_l$ by

$$\mathrm{grad}_l\, q_l = (G^e)^{-1}\, \mathrm{grad}\, G^n q_l.$$

With this notions, we can state the following two theorems, which show, that the exact sequence property of the de Rham complex is retained on the coarse level.

Lemma 1. *For $q_{l+1} \in Q_{l+1}$ there holds*

$$P_e \operatorname{grad}_{l+1} q_{l+1} = \operatorname{grad}_l P_n q_{l+1} \quad \forall q_{l+1} \in Q_{l+1}. \tag{4.8}$$

This implies that the we obtain the commuting diagram:

$$\begin{array}{ccc} Q_{l+1} & \xrightarrow{\operatorname{grad}_{l+1}} & V_{l+1} \\ \downarrow P_n & & \downarrow P_e \\ Q_l & \xrightarrow{\operatorname{grad}_l} & V_l \end{array}$$

Proof. See [RS02] p. 12. □

Lemma 2. *The coarse grid kernel functions are exactly gradient functions, i.e., there holds*

$$V_{l+1,0} = \operatorname{grad}_{l+1} Q_{l+1}. \tag{4.9}$$

Proof. See [RS02] p. 12. □

4.4.2. Auxiliary-space Maxwell Solver

Another related algebraic multigrid approach to tackle Maxwell's equation, which is worth mentioning here, is the so called *Auxiliary-space Maxwell Solver (AMS)* introduced by Hiptmair and Xu in [HX07].

The idea is to use the already well developed theories of algebraic multigrid methods for H^1 problems and apply one of them for each vector component of the $H(\operatorname{curl}, \Omega)$ and also one extra H^1 AMG to the kernel of the curl operator, similarly to the Hiptmair preconditioner.

5. Algorithm

In this chapter, we describe our proposed algorithm. Parts of it are based on previous work [Sch16]. For recent similar developments see [NP19].

5.1. Overview

To construct the hierarchy of matrices we choose to first make an abstract mesh representation which includes just the connectivity information from the discretization mesh. No actual geometric information is preserved there. It can be argued that the proposed method is not a pure AMG method, since it relies on the connectivity information of edges to vertices of the available mesh. For cases where the mesh is not available for some reason, the connectivity information could be also obtained by setting up a lowest order nodal finite element problem (e.g. Poisson problem) and obtain the connectivity information from the resulting system matrix. This relies on the fact, that the off-diagonal entries only occur for vertices connected by an edge.

To obtain a coarser representation we use weights to decide how to build a coarser version of this mesh. With these weights we decide how to proceed with so called collapsing, where edges are chosen and collapsed, meaning their end vertices are considered as one coarse vertex. We remember that map of fine vertices to coarse vertices and can then calculate a prolongation from it. (See Section 4.4.1). With the resulting prolongations we can restrict the system matrix to a coarser level with fewer degrees of freedom. We stop this coarsening process, when the size of the current coarse system matrix is small enough to quickly calculate its inverse as shown in Algorithm 2. To compute the inverse we use the sparse Cholesky factorization provided by NGSolve. During this coarsening process we also construct a Block Jacobi smoothers and H1 AMG for the nodal problem by means already existing in NGSolve. This concludes the setup of the preconditioner.

5.2. Weight Calculation

In the proposed algorithm we use the concept of weights in several places. Here a weight is a scalar value assigned to an edge or a face. We compute these weights during the assembly phase of the finite element system matrix. We recall from Definition 2, that Nédélec elements of the first kind of order 0 have one degree of freedom (DOF) per edge. So the element matrices are matrices of size 6×6 for the tetrahedral elements. The presented concept of substitution matrices is similar to the concept of edge matrices introduced in [Kra06].

Edge weights are calculated as the Schur complement of the element matrix with respect to the single index corresponding to the edge. And for faces, we add up the diagonal entries

of the Schur complement of the three edges corresponding to the face. This properties can be viewed as a minimal extension principle. For more details on the relation of Schur complements to the minimal extension principle see Appendix A.

Definition 5 (Spectral equivalence)**.** Two symmetric positive semi-definite (SPSD) matrices A and B are called spectrally equivalent with bounds γ_1, γ_2 iff:

$$\gamma_1 \boldsymbol{u}^T B \boldsymbol{u} \leq \boldsymbol{u}^T A \boldsymbol{u} \leq \gamma_2 \boldsymbol{u}^T B \boldsymbol{u} \qquad \forall \boldsymbol{u} \in \mathbb{R}^n \tag{5.1}$$

We can view these weights as entries in a spectrally equivalent matrix similar as in [HLRS01]. An element matrix can thus be approximated by a diagonal matrix for the edge weights, and a diagonal matrix for the face weights together with the coupling of local edges to faces.

$$\int_K \operatorname{curl} \boldsymbol{u} \cdot \operatorname{curl} \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \approx \mathrm{CURL} \cdot \begin{bmatrix} w_{F_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{F_4} \end{bmatrix} \cdot \mathrm{CURL}^T + \begin{bmatrix} w_{E_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{E_6} \end{bmatrix}, \tag{5.2}$$

where CURL is the local discrete curl operator defined as the mapping of faces to its contained edges while respecting the edge orientations and w_E and w_F are edge and face weights respectively.

For further discussion it is important to note, that the system matrix discretized by the edge elements is not an M-matrix but SPSD. With this local approximation matrices, we can also construct a global approximation matrix which is spectrally equivalent to the assembled system matrix. This global spectral equivalence is guaranteed by the following theorem:

Theorem 7. *Let the stiffnes matrix $K_h \in \mathbb{R}^{N_h \times N_h}$ be SPD, and K_h be composed from SPSD element matrices $K^{(r)} \in \mathbb{R}^{n_r \times n_r}$, $r \in \tau_h$ i.e., K_h can be represented in the form*

$$K_h = \sum_{r \in \tau_h} C_r^T K^{(r)} C_r, \tag{5.3}$$

where τ_h denotes the index set of all finite elements, and $C_r \in \mathbb{R}^{n_r \times N_h}$ are the element connectivity matrices. Further, let us suppose that, for all $r \in \tau_h$, there are SPSD matrices $B^{(r)}$, such that the spectral equivalence inequalities

$$c_1^{(r)} B^{(r)} \leq K^{(r)} \leq c_2^{(r)} B^{(r)}, \quad \forall r \in \tau_h \tag{5.4}$$

hold, with h-independent, positive spectral equivalence constants $c_1^{(r)}$ and $c_2^{(r)}$. Then the matrix

$$B_h = \sum_{r \in \tau_h} C_r^T B^{(r)} C_r \tag{5.5}$$

is spectrally equivalent to the stiffnes matrix K_h, i.e.,

$$c_1 B_h \leq K_h \leq c_2 B_h \tag{5.6}$$

with the spectral equivalence constants

$$c_1 = \min_{r \in \tau_h} c_1^{(r)} \quad \textit{and} \quad c_2 = \max_{r \in \tau_h} c_2^{(r)}.$$

Additionally, the matrix B_h is SPD. If K_h is only SPSD, the spectral equivalence inequalities Equation (5.6) remain valid, B_h is SPSD, and $\ker(B_h) = \ker(K_h)$.

Proof. See Lemma 2.2 in [HLRS01]. □

5.3. Coarsening Algorithm

To construct the hierarchy of abstract meshes we start with a given abstract mesh on a level and apply the coarsening algorithm as described here in more detail. First an edge is chosen based on some priority and then its two vertices are collapsed, which means we consider them as the same vertex on the coarser level. To obtain the priority for edges we introduce so called *collapse weights*. The exact calculation for the collapse weights is described in the next section. Figure 5.1 shows a representation of a coarse abstract mesh (red) obtained by collapsing some fine edges (black) and Algorithm 3 summarizes the coarsening algorithm.

Algorithm 3 Coarsening

```
1: procedure COARSENING(AbstractMesh)
2:     Compute collapse weights for all edges in the Mesh
3:     Sort edges by their collapse weight
4:     for all edges e in priority sorted edges do
5:         if no connected edge is already collapsed then
6:             Collapse edge (Identify vertices of the edge with a single coarse vertex)
7:     Build the coarse abstract mesh from the collapse information
```

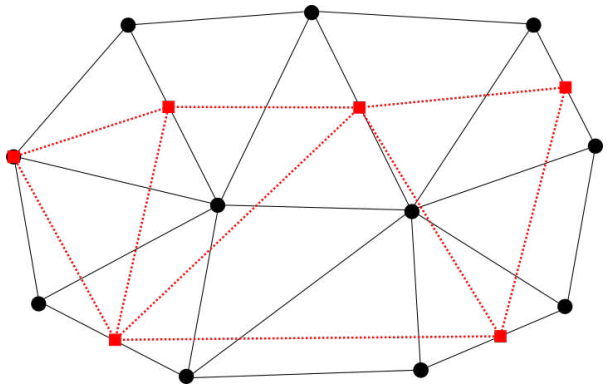


Figure 5.1.: Schematic of the coarsening process for a mesh

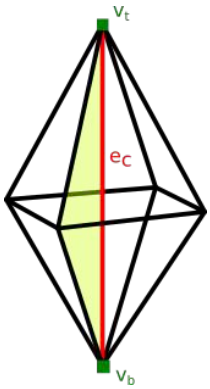


Figure 5.2.: Cell patch with the red highlighted center edge e_c and a highlighted center face.

5.4. Collapse Weights

In this section, we describe the algorithm and underlying heuristic to obtain the collapse weights. It is based on the notion that after collapsing an edge a flux through collapsed faces needs to be diverted via adjacent faces.

First we need to get some notation out of the way. Figure 5.2 shows a red highlighted center edge e_c with the corresponding *cell patch* around the edge. The cell patch is described by all cells which contain e_c. We call these *center cells* in the following discussion.

Center faces (denoted as F_c) are all faces which contain e_c. The set of center faces is $\mathcal{F}_c$. One such face is highlighted in a transparent yellow color in Figure 5.2. If we were to collapse the center edge, also all center faces would collapse and no flux could pass through any of these center faces.

All edges in the cell patch which are connected to the *top vertex* v_t, but excluding the center edge, are called *top edges* and notated as $e_t \in \mathcal{E}_t$. Also $\mathcal{F}_t$ denotes the set of faces formed by two top edges. Respectively, all edges connected to the *bottom vertex* v_b excluding the center edge are called bottom edges and similarly notated as $e_b \in \mathcal{E}_b$ and $\mathcal{F}_b$ is the set of faces formed by two bottom edges.

Every center face contains exactly one top edge, one bottom edge, and the center edge. Since the number of center cells, center faces, top edges, and bottom edges are thus all identical we designate their number by N_c.

We start by looking at a global discrete flux $\boldsymbol{B}_h \in \boldsymbol{W}_h$ and its corresponding discrete global vector potential $\boldsymbol{u}_h \in \boldsymbol{V}_h$. As before it holds that $\boldsymbol{B}_h = \operatorname{curl} \boldsymbol{u}_h$. First we want to express this vector potential locally on the patch by its degrees of freedom. Since we use lowest order edge elements, each degree of freedom corresponds exactly to one edge. Thus $\boldsymbol{u}_h$ can be described on the edge path via the following degrees of freedom:

- one DOF corresponding to the center edge $u_c \in \mathbb{R}$
- one DOF u_{e_t} per top edge $e_t \in \mathcal{E}_t$, therefor all DOFs corresponding to the top edges are collected in the vector $\boldsymbol{u}_t \in \mathbb{R}^{N_c}$

- one DOF u_{e_b} per bottom edge $e_b \in \mathcal{E}_b$, therefor all DOFs corresponding to the bottom edges are collected in the vector $\boldsymbol{u}_b \in \mathbb{R}^{N_c}$

The degree of freedom of the flux $\boldsymbol{B}_h$ corresponding to the center face F_c is called B_{F_c}. These DOFs are collected in the vector $\boldsymbol{B}_c \in \mathbb{R}^{N_c}$. Now to judge how "easy" it would be to divert $\boldsymbol{B}_c$ via the other faces connected to top and bottom edges, we will look at the following two norms.

$$\|\boldsymbol{B}_h\|_1^2 = \min_{\substack{\boldsymbol{u}_t,\boldsymbol{u}_b,\boldsymbol{u}_c \\ \boldsymbol{B}_c(\boldsymbol{u})=\boldsymbol{B}_c}} \sum_{F_c\in\mathcal{F}_c} w_{F_c}|B_{F_c}|^2 + \sum_{e_t\in\mathcal{E}_t} w_{e_t}|u_{e_t}|^2 + \sum_{e_b\in\mathcal{E}_b} w_{e_b}\left|u_{e_b}\right|^2 \tag{5.7}$$

$$\begin{aligned}\|\boldsymbol{B}_h\|_2^2 = \min_{\substack{\boldsymbol{u}_t,\boldsymbol{u}_b,\boldsymbol{u}_c \\ \boldsymbol{B}_c(\boldsymbol{u})=\boldsymbol{B}_c}} & \sum_{F_c\in\mathcal{F}_c} w_{F_c}|B_{F_c}|^2 + \sum_{e_t\in\mathcal{E}_t} w_{e_t}|u_{e_t}|^2 + \sum_{e_b\in\mathcal{E}_b} w_{e_b}\left|u_{e_b}\right|^2 \\ & + \left[\sum_{e_t\in\mathcal{E}_t}\sum_{\substack{e_t\subset F \\ F\notin\mathcal{F}_t \\ F\notin\mathcal{F}_c}} w_F|u_{e_t}|^2\right] + \sum_{\substack{F_t\in\mathcal{F}_t \\ e_{t_1},e_{t_2}\in F_t}} w_{F_t}\left|u_{e_{t_1}} - u_{e_{t_2}}\right|^2 \\ & + \left[\sum_{e_b\in\mathcal{E}_b}\sum_{\substack{e_b\subset F \\ F\notin\mathcal{F}_b \\ F\notin\mathcal{F}_c}} w_F\left|u_{e_b}\right|^2\right] + \sum_{\substack{F_b\in\mathcal{F}_b \\ e_{b_1},e_{b_2}\in F_b}} w_{F_b}\left|u_{e_{b_1}} - u_{e_{b_2}}\right|^2\end{aligned} \tag{5.8}$$

We see that the second norm is the same as the first norm with additional terms for the non-center faces connected to the top and bottom edges. Similar to Section 5.2, we are looking for spectral equivalence of the norms, such that for any given flux $\boldsymbol{B}_h$ it holds that

$$\|\boldsymbol{B}_h\|_2\,\gamma_1 \le \|\boldsymbol{B}_h\|_1 \le \gamma_2\|\boldsymbol{B}_h\|_2\,. \tag{5.9}$$

Heuristically, a flux is easy to divert through the non-center faces if the norms are very similar. This would mean that the additional terms in the second norm don't have an disproportional high contribution. Since all weights are positive and therefor the additional terms in the second norm can only have positive contributions we know that $\gamma_2 \le 1$. Thus the value of interest is the lower bound γ_1 of the equivalence Equation (5.9). Assuming we can represent the two norms by matrices N_1 and N_2 we can determine the lower bound as the smallest eigenvalue of the generalized eigenvalue problem (EVP):

$$N_1\boldsymbol{x} = \gamma N_2\boldsymbol{x} \tag{5.10}$$

For each edge e_c we can then setup this generalized EVP corresponding to the cell patch around e_c and assign the biggest eigenvalue γ_1 of this problem as the collapse weight to the edge. How to obtain such matrices for the EVP from the norms described as minimization problems will be the topic of the next subsection.

5.4.1. Solving the Minimization Problem

Both norms, Equation (5.7) and Equation (5.8), can be generalized to a common structure as shown in Equation (5.11). D_{F_c} is the diagonal matrix built by the face weights of the central faces. M_t and M_b are the non-center parts contributing to the norm which differ between the two norms.

$$\|\boldsymbol{B}_h\| = \min_{\substack{\boldsymbol{u}_t,\boldsymbol{u}_b,u_c \\ \boldsymbol{u}_t-\boldsymbol{u}_b-u_c\mathbb{1}=\boldsymbol{B}_c}} \|\boldsymbol{B}_c\|^2_{D_{F_c}} + \|\boldsymbol{u}_t\|^2_{M_t} + \|\boldsymbol{u}_b\|^2_{M_b} + w_{e_c}u_c^2 \tag{5.11}$$

We can solve this minimization problem via the method of Lagrange multipliers. In the following derivation I is the identity matrix, $\mathbb{1}$ is the one vector and $\boldsymbol{\lambda}$ is the vector of Lagrange parameters. The Lagrange system for Equation (5.11) is given by

$$\begin{bmatrix} M_t & & & I \\ & M_b & & -I \\ & & w_c & -\mathbb{1}^T \\ I & -I & -\mathbb{1} & \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_t \\ \boldsymbol{u}_b \\ u_c \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boldsymbol{B}_c \end{bmatrix}. \tag{5.12}$$

Now from the first and second equation of the system we get

$$\boldsymbol{u}_t = -M_t^{-1}\boldsymbol{\lambda}, \qquad \boldsymbol{u}_b = M_b^{-1}\boldsymbol{\lambda}.$$

We plug these identities into the fourth equation and solve for the Lagrange parameters:

$$\begin{aligned} \boldsymbol{B}_c &= \boldsymbol{u}_t - \boldsymbol{u}_b - u_c\mathbb{1} \\ \boldsymbol{B}_c &= \left(-M_t^{-1} - M_b^{-1}\right)\boldsymbol{\lambda} - u_c\mathbb{1} \\ \boldsymbol{\lambda} &= -\underbrace{\left(M_t^{-1} + M_b^{-1}\right)^{-1}}_{M}(u_c\mathbb{1} + \boldsymbol{B}_c) = -M(u_c\mathbb{1} + \boldsymbol{B}_c) \end{aligned} \tag{5.13}$$

A short side note on inverting the matrices M_t and M_b here: They are not necessarily regular, but this can be fixed by adding a small ε regularization. Since they are inverted twice to obtain the matrix M, the error for M is again in the order of ε.

Also $\boldsymbol{u}_t$ and $\boldsymbol{u}_b$ can be expressed in terms of u_c by plugging this result back into the first and second equation of the Lagrange system:

$$\boldsymbol{u}_t = -M_t^{-1}M(u_c\mathbb{1} + \boldsymbol{B}_c), \qquad \boldsymbol{u}_b = M_b^{-1}M(u_c\mathbb{1} + \boldsymbol{B}_c)$$

With Equation (5.13) for the Lagrange multipliers together with the third equation of the Lagrange system we can solve for u_c:

$$\begin{aligned} 0 &= w_c u_c - \mathbb{1}^T\boldsymbol{\lambda} \\ 0 &= w_c u_c + \mathbb{1}^T M(u_c\mathbb{1} + \boldsymbol{B}_c) \\ u_c &= \frac{-\mathbb{1}^T M\boldsymbol{B}_c}{w_c + \mathbb{1}^T M\mathbb{1}} \end{aligned}$$

Now we look to the original norm Equation (5.11) again and represent the norms with their respective matrices to simplify the equation with the obtained minimizers. The next step is to eliminate $\boldsymbol{u}_b$ and $\boldsymbol{u}_t$:

$$\begin{aligned}
\|\boldsymbol{B}_c\| &= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c + \min_{\substack{\boldsymbol{u}_t,\boldsymbol{u}_b,u_c \\ \boldsymbol{u}_t-\boldsymbol{u}_b-u_c\mathbb{1}=\boldsymbol{B}_c}} \boldsymbol{u}_t^T M_t \boldsymbol{u}_t + \boldsymbol{u}_b^T M_b \boldsymbol{u}_b + w_c u_c^2 \\
&= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c \\
&\quad + \min_{\substack{\boldsymbol{u}_t,\boldsymbol{u}_b,u_c \\ \boldsymbol{u}_t-\boldsymbol{u}_b-u_c\mathbb{1}=\boldsymbol{B}_c}} w_c u_c^2 \\
&\quad + (u_c\mathbb{1}^T + \boldsymbol{B}_c^T) \cancel{M^T M_t^{-T} M_t M_t^{-1}} M (u_c\mathbb{1} + \boldsymbol{B}_c) \\
&\quad + (u_c\mathbb{1}^T + \boldsymbol{B}_c^T) \cancel{M^T M_b^{-T} M_b M_b^{-1}} M (u_c\mathbb{1} + \boldsymbol{B}_c) \\
&= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c + \min_{u_c} (u_c\mathbb{1}^T + \boldsymbol{B}_c^T) M (u_c\mathbb{1} + \boldsymbol{B}_c) + w_c u_c^2
\end{aligned}$$

Finally we can plug in the minimizing u_c to arrive at the desired matrix representation of the norms:

$$\begin{aligned}
\|\boldsymbol{B}_c\| &= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c + \left(\frac{-\mathbb{1}^T M \boldsymbol{B}_c}{w_c + \mathbb{1}^T M \mathbb{1}} \mathbb{1}^T + \boldsymbol{B}_c^T \right) M \left(\frac{-\mathbb{1}^T M \boldsymbol{B}_c}{w_c + \mathbb{1}^T M \mathbb{1}} \mathbb{1} + \boldsymbol{B}_c \right) + w_c \left(\frac{-\mathbb{1}^T M \boldsymbol{B}_c}{w_c + \mathbb{1}^T M \mathbb{1}} \right)^2 \\
&= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c + \frac{(\mathbb{1}^T M \boldsymbol{B}_c)^2}{(w_c + \mathbb{1}^T M \mathbb{1})^2} \mathbb{1}^T M \mathbb{1} - 2 \frac{\mathbb{1}^T M \boldsymbol{B}_c}{w_c + \mathbb{1}^T M \mathbb{1}} \mathbb{1}^T M \boldsymbol{B}_c + \boldsymbol{B}_c^T M \boldsymbol{B}_c + w_c \frac{(\mathbb{1}^T M \boldsymbol{B}_c)^2}{(w_c + \mathbb{1}^T M \mathbb{1})^2} \\
&= \boldsymbol{B}_c^T D_{F_c} \boldsymbol{B}_c + \boldsymbol{B}_c^T M \boldsymbol{B}_c - \frac{(\mathbb{1}^T M \boldsymbol{B}_c)^2}{w_c + \mathbb{1}^T M \mathbb{1}} \\
&= \underbrace{\boldsymbol{B}_c^T \left(D_{F_c} + \underbrace{M - \frac{1}{w_c + 1^T M \mathbb{1}} M^T \mathbb{1} \mathbb{1}^T M}_{\widetilde{M}} \right)}_{N} \boldsymbol{B}_c
\end{aligned}$$

The algorithm based on this heuristic is summarized in Algorithm 4.

5.5. Smoothed Prolongations

Another technique to improve the condition number of our solver is to smooth the prolongation by means of a damped Jacobi method as first introduced in [Van95]. The sections follows the presentation in [BGH+03], where the concept was adapted to the $H(\mathrm{curl}, \Omega)$ setting.

The intuition behind the technique comes from the fact, that smooth error components are typically characterized by a small energy. With A_l scaled such that $\|A_l\| = 1$ this can be written as: $\boldsymbol{e}^T A_l \boldsymbol{e} \ll \boldsymbol{e}^T \boldsymbol{e}$. To reduce the energy of the interpolated coarse grid

Algorithm 4 ComputeCollapseWeights

procedure ComputeCollapseWeights(AbstractMesh,EdgeWeights,FaceWeights)
  for all edges $e_c \in \mathcal{E}$ **do**
    for $i = 1, 2$ **do**
      Build M_t and M_b matrices of $\|\cdot\|_i$ with respect to e_c
      Compute $M \coloneqq \left(M_t^{-1} + M_b^{-1}\right)^{-1}$
      Compute $\widetilde{M} \coloneqq M - \frac{1}{w_c + \mathbb{1}^T M \mathbb{1}} M^T \mathbb{1}\mathbb{1}^T M$
      $N_i \coloneqq D_{F_c} + \widetilde{M}$
    Set the edge collapse weight w_{e_c}, to the minimal eigenvalue of the generalized EVP $N_1 \boldsymbol{x} = \gamma N_2 \boldsymbol{x}$

correction $\hat{P}_l u_{l+1}$ on level l, with a given tentative edge prolongation $\hat{P}_l$, we can instead use an associated smoothed prolongation

$$P_l = (I - \alpha D_l^{-1} A_l)\hat{P}_l, \tag{5.14}$$

where I is the identity matrix, α is the damping parameter, A_l is the system matrix on level l and D_l is its diagonal. The essential argument for the application of this method to the context of $H(\mathrm{curl}, \Omega)$ multigrid is the following lemma, where instead of A_l we use the part of the system matrix arising from the curl part of Equation (3.2). Here denoted as $K^e_{\mathrm{curl},l}$.

Lemma 3 (Lemma 1 in [BGH+03])**.** *Assume that an unsmoothed edge interpolation operator, $\hat{P}^e_l$, satisfies the commutative relation* (4.8) *and that* (5.14) *is used for $K^e_{\mathrm{curl},l}$ to produce a smoothed interpolation operator, P^e_l. Then P^e_l also satisfies* (4.8)*.*

Proof. We have

$$\begin{aligned} P^e_{l+1} \operatorname{grad}_{l+1} &= (I - \alpha D_l^{-1} K^e_{\mathrm{curl},l}) \hat{P}^e_l \operatorname{grad}_{l+1} \\ &= (I - \alpha D_l^{-1} K^e_{\mathrm{curl},l}) \operatorname{grad}_l P^n_l \\ &= \operatorname{grad}_l P^n_l, \end{aligned}$$

where we use the fact that

$$K^e_{\mathrm{curl},l} \operatorname{grad}_l = \Theta$$

and Θ denotes the zero matrix. □

Empirically we found that it is benificial possible to use the full system matrix instead of only the curl part since the mass term has a very low coefficient anyway. Also empirically we determined a damping parameter of $\alpha = \frac{1}{2}$.

One problem we encountered with the smoothed prolongation is, that it drastically increases the operator complexity (see Definition 6), which is a measure for the overhead due to the multigrid levels.

	no smoothing	smooth $\hat{P}$ on every level	smooth three levels
AMG complexity	2.2	41.8	2.0

Table 5.1.: Comparison of AMG complexities for several smoothed prolongation schemes.

Definition 6 (Operator complexity)**.**

$$\text{AMG complexity} = \frac{\sum_{l=1}^{N} \text{nnz}(A_l)}{\text{nnz}(A_1)}, \tag{5.15}$$

where $\text{nnz}(A)$ are the number of non-zero entries of a matrix A.

In [BGH+03] this problem is resolved by putting more than two fine vertices into an agglomerate which then corresponds to a coarse vertex on the next level. Since our coarsening is based on collapsing edges, at most two fine vertices are mapped to one coarse vertex on the next level. To handle the increased operator complexity, we skip building system matrices on some levels by first multiplying three Reitzinger/Schöberl-prolongations and then smooth the resulting prolongation:

$$P_l = (I - \alpha D_l^{-1} A_l)(\hat{P}_l \hat{P}_{l+1} \hat{P}_{l+2}) : V_l \to V_{l+3}. \tag{5.16}$$

Table 5.1 summarizes the impact of smoothing on the operator complexity for an example setup as in Chapter 6. The third column shows that the technique from Equation (5.16) is comparable to no smoothing in terms of operator complexity.

6. Numerical Experiments

To show the robustness and performance of the proposed algorithm we apply it to two magnetostatic problems. Both problems consist of a coil placed in a box of air. The first problem contains an iron rod as a core and the second problem has a closed iron ring as a core. The driving force is a current impressed in the coil for both problems.

6.1. Example Setup

A picture for the discretized geometry of the first problem is depicted in Figure 6.1. Figure 6.3 shows the same for the second problem. Figure 6.2 and Figure 6.4 show the field lines of the $\boldsymbol{B}$-field.

Homogeneous Dirichlet boundary conditions are used on the box boundary. These model a perfect magnetic conductor as presented in Section 2.1.2. We can clearly see the orthogonal field lines at the box boundaries for both problems in the Figures 6.2 to 6.4 as discussed in Section 2.2.

The main fixed parameters of the examples are given as follows:

- Impressed current $\boldsymbol{j}_i = 1000\,\mathrm{A/m^2}$
- Vacuum permeability $\mu_0 = 1.257 \times 10^{-6}\,\mathrm{H\,m^{-1}}$

To see the effect of different regularization parameters, κ is varied for the timings presented in Section 6.2. Typically permeabilities are given as relative permeabilites μ_r with respect to the vacuum permeability such that $\mu = \mu_r \mu_0$. In the region of the iron core the relative permeability $\mu_{r,Fe}$ is varied to examine the solvers behavior for parameter jumps. The relative permeability of the coil as well as the air are set to 1 which is a good approximation for the permeabilitys of copper and air.

The `Python` setup is the same for both examples and it is attached in Appendix B. Only the geometries are different.

The preconditioner was used together with the conjugate gradient solver provided by NGSolve. It was set up to iterate until a residual reduction of 10^{-8} is achieved. Our AMG coarsening setup was configured to reduce down to an abstract meshes with 1000 vertices. On the coarsest level we used the sparse Cholesky inverse provided by NGSolve.

6.2. Timings and Results

Table 6.1 and Table 6.2 show the condition number, iterations, and timings for several parameter combinations. The condition number is determined via an Lanczos eigenvalue solver available in NGSolve. Overall the algorithm is quite stable with respect to the

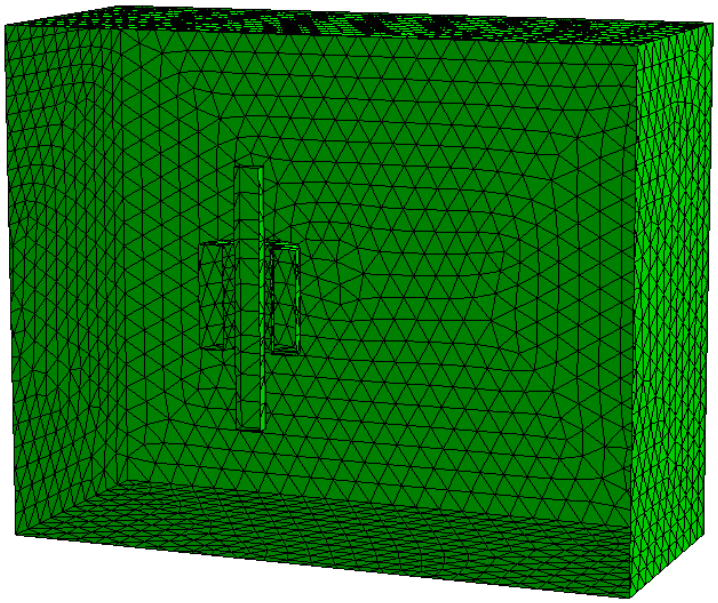


Figure 6.1.: Mesh of a coil with an iron rod as a core.

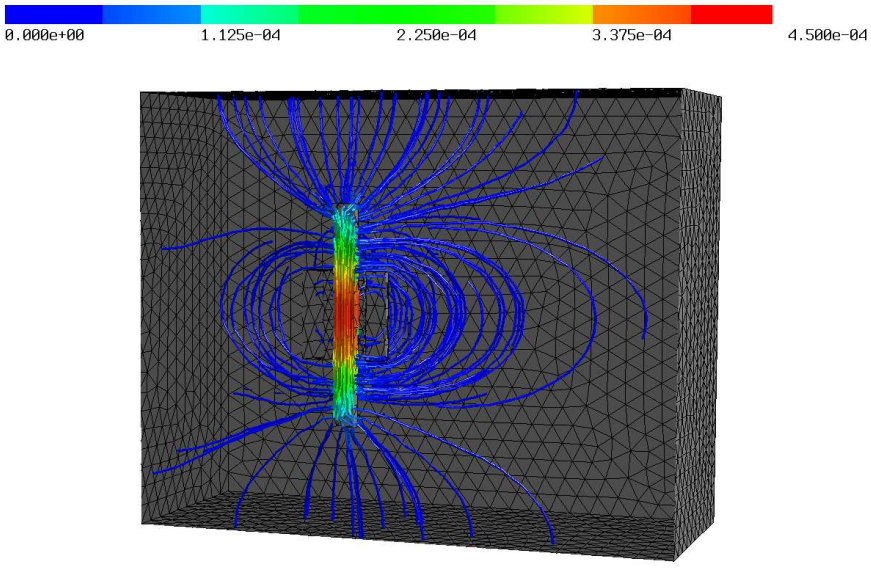


Figure 6.2.: Field lines of the B-field of the first model problem.

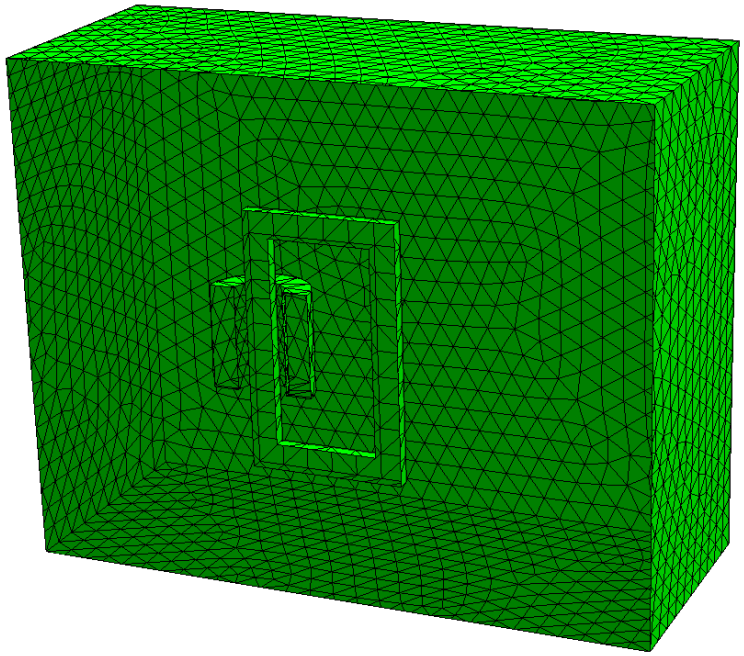


Figure 6.3.: Mesh of a coil with a closed iron ring as a core.

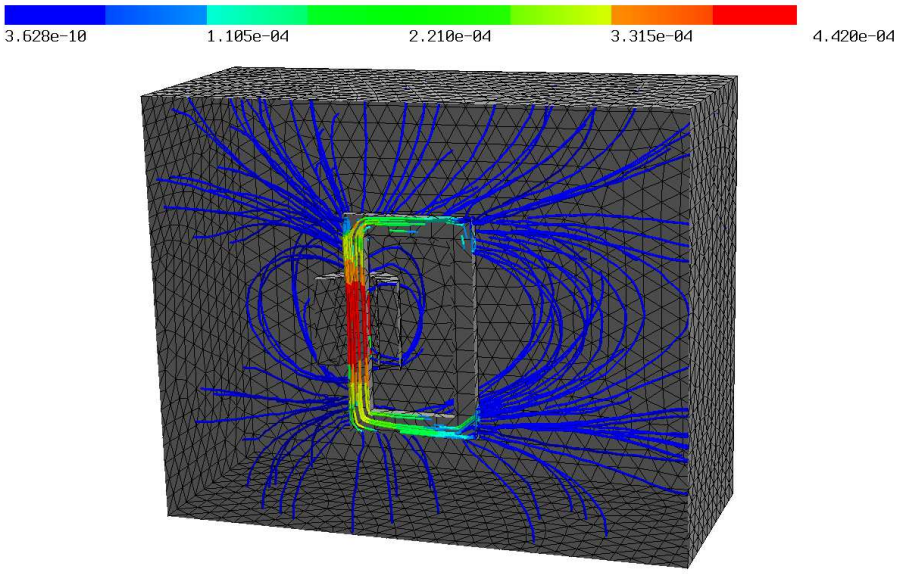


Figure 6.4.: Field lines of the B-field of the second model problem.

#DOFs	κ	$\mu_{r,Fe}$	levels	cond$(C^{-1}A)$	iterations	setup t	solver t	total t
8k	1	1	4	5.2	21	1.6s	0.2s	1.8s
8k	1	1e6	4	6.1	22	1.5s	0.2s	1.7s
8k	1e-8	1	4	5.1	17	1.3s	0.2s	1.5s
8k	1e-8	1e6	4	6.0	18	1.3s	0.1s	1.4s
680k	1	1	13	17.2	37	79.5s	12.3s	91.8s
680k	1	1e6	13	18.9	40	84.3s	12.8s	97.1s
680k	1e-8	1	13	16.6	33	88.9s	11.0s	99.9s
680k	1e-8	1e6	13	27.3	38	77.0s	12.1s	89.1s
5M390k	1	1	16	39.9	57	525.7s	155.2s	680.9s
5M390k	1	1e6	16	230k	95	603.2s	255.5s	858.7s
5M390k	1e-8	1	16	37.4	50	513.1s	150.6s	663.7s
5M390k	1e-8	1e6	16	66.0	94	533.9s	251.1s	785.0s

Table 6.1.: Numerical results for the rod core example.

#DOFs	κ	$\mu_{r,Fe}$	levels	cond$(C^{-1}A)$	iterations	setup t	solver t	total t
8k	1	1	4	5.9	22	1.7s	0.2s	1.9s
8k	1	1e6	4	11.3	26	1.7s	0.2s	1.9s
8k	1e-8	1	4	6.0	17	1.8s	0.2s	2.0s
8k	1e-8	1e6	4	50k	33	1.9s	0.3s	2.2s
680k	1	1	13	18.1	38	67.0s	11.2s	78.2s
680k	1	1e6	13	44.0	45	81.5s	15.7s	97.2s
680k	1e-8	1	13	17.6	34	75.9s	11.5s	87.4s
680k	1e-8	1e6	13	35.6	65	75.9s	21.6s	97.5s
5M390k	1	1	16	40.6	57	507.1s	142.3s	649.4s
5M390k	1	1e6	16	18k	101	575.0s	326.3s	901.3s
5M390k	1e-8	1	16	35.0	49	576.9s	129.3s	706.2s
5M390k	1e-8	1e6	16	159.9	135	529.9s	407.0s	936.9s

Table 6.2.: Numerical results for the closed core example.

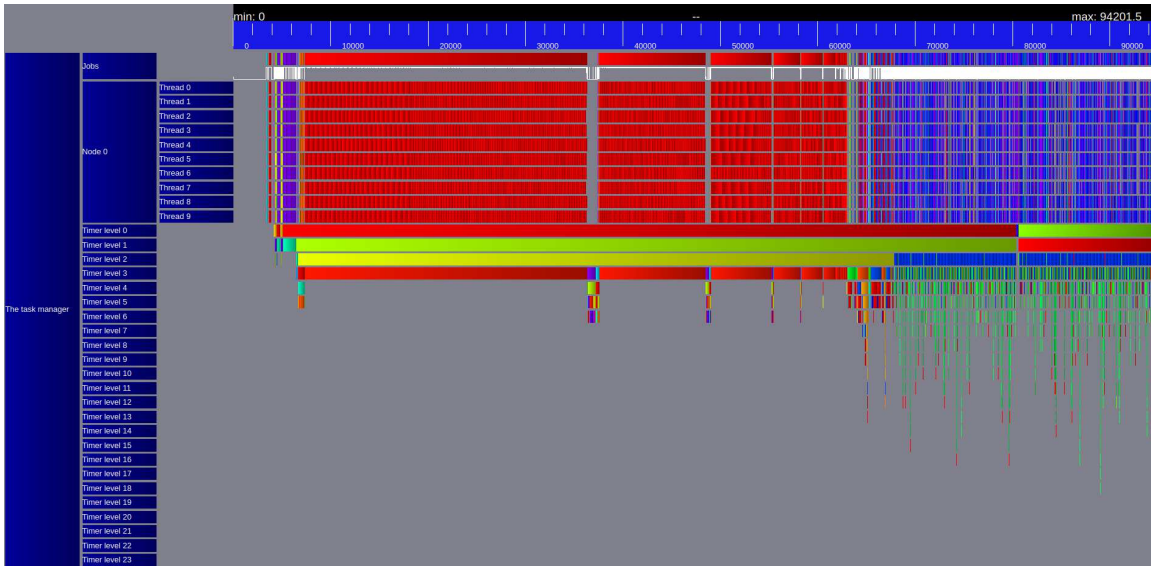


Figure 6.5.: Full trace plot of an example with 680k degrees of freedom.

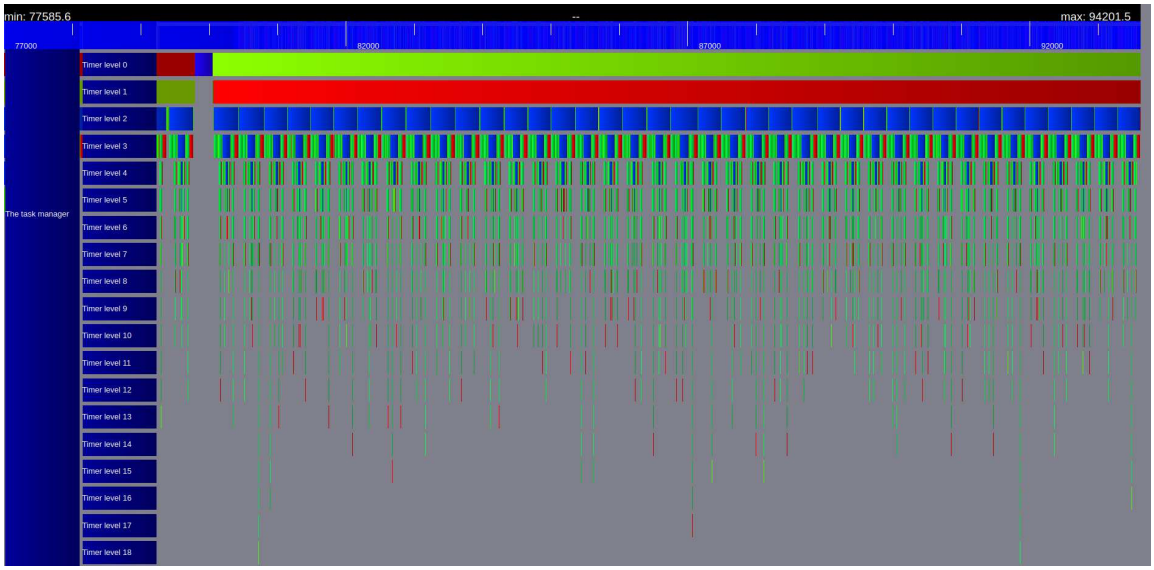


Figure 6.6.: Trace plot of the solving stage of an example with 680k degrees of freedom.

#DOFs	RAM	iterations	total time
980k	2 GB	37	22sec
5M380k	10 GB	53	2.5min
42M217k	88 GB	77	25min

Table 6.3.: Additionial numerical results

regularization κ in both examples. Certain combinations have very high conditions numbers but a reasonable number of iterations. This behavior is due to single spurious eigenvalues.

The solution time seems roughly linear with respect to the number of degrees of freedom. The biggest part is in the setup time which is almost exclusively to the bad performance of the collapse weight calculation algorithm. It should be noted that also the time spend on determining the eigenvalues for the condition number is included in the setup time. This extra overhead amounts to approximately the time used for solving. In concrete applications the calculation of the condition number is not necessary.

Details of the timings are shown in the ViTE[1] trace plots Figure 6.5 and Figure 6.6. We see a good utilization of all 10 threads in the first plot. The red bars are the dominating edge collapse weights computations. In the second plot we see blue bars recursively stacked. They represent the V-cycle part of our multigrid and the higher levels are surrounded by the smoothers shown in green and red.

All the previous timings were done with 10 Threads on an Intel Xeon CPU E7-4850 machine. Even more promising are the timings shown in Table 6.3 produced with 8 threads on an Intel Core i7-9800X CPU by Matthias Hochsteger at CERBSim GmbH.

[1] `http://vite.gforge.inria.fr/index.php`

7. Conclusion and Outlook

We combined several techniques to establish a preconditioner for magnetostatic problems, which is robust in the regularization parameter. Improved robustness with respect to the mesh-size and jumps in the permeability parameter were demonstrated.

Also the performance of the algorithm looks very promising. The most time consuming part is currently the setup of the preconditioner where a lot of time is spent in calculating the collapse weights. This is the prime candidate for further optimizations.

One could extend the work to properly handle complex coefficients, which arise in the case of eddy-current problems. Most parts of the algorithm can already handle complex coefficients and entries and also NGSolve is capable of using complex numbers. The main problem here seems to be the proper combination with an iterative solver, since the prerequisites for conjugate gradient method are not fulfilled for this regime. One would need to resort to GMRES or related algorithms.

A. Schur Complement

The material of this chapter is from [Axe94] chapter 3.2. We consider a matrix A split in the following way

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where $A_{ii}, i = 1, 2$ are square matrices.

Definition 7 (Definition 3.4 in [Axe94])**.** If A_{11} is non-singular, we define

$$S := A/A_{11} := A_{22} - A_{21}A_{11}^{-1}A_{12}$$

which is called the *Schur complement* of A with respect to A_{11}.

It is the matrix resulting from a block Gauß elimination with A_{11} as the pivot block. The triangular factorization of A can be stated as:

$$A = \begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix} \tag{A.1}$$

Theorem 8 (Theorem 3.8 in [Axe94])**.** *Let A be a Hermitian positive definite, A_{11} be regular, and $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$ be a partitioning of $\boldsymbol{x}$ consistent with the partitioning of A. Then*

(a) A_{11} and A_{22} are Hermitian and positive definite.

(b) $\boldsymbol{x}^* A \boldsymbol{x} \geq \boldsymbol{x}_2^* S \boldsymbol{x}_2$

(c) For any $\boldsymbol{x}_2$, $\min_{\boldsymbol{x}_1} \boldsymbol{x}^* A \boldsymbol{x} = \boldsymbol{x}_2^* S \boldsymbol{x}_2$

(d) $\min_{\boldsymbol{x}, \boldsymbol{x}_2 \neq \boldsymbol{0}} \boldsymbol{x}^* A \boldsymbol{x} = \min_{\boldsymbol{x}_2 \neq \boldsymbol{0}} \boldsymbol{x}_2^* S \boldsymbol{x}_2$

Proof. To prove (a) note that $\boldsymbol{x}^* A \boldsymbol{x} = \boldsymbol{x}_1^* A_{11} \boldsymbol{x}_1$ if $\boldsymbol{x}_2 = \boldsymbol{0}$ and if $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ is partitioned consistently with the partitioning of A. Hence, it follows that $\boldsymbol{x}_1^* A_{11} \boldsymbol{x}_1 > 0 \forall \boldsymbol{x}_1 \neq \boldsymbol{0}$, so A_{11} is positive definite. Similarly A_{22} is positive definite. For (b) consider the factorization Equation (A.1) and $A_{12}^* = A_{21}$:

$$\boldsymbol{x}^* A \boldsymbol{x} = (\boldsymbol{x}_1 + A_{11}^{-1} A_{12} \boldsymbol{x}_2)^* A_{11} (\boldsymbol{x}_1 + A_{11}^{-1} A_{12} \boldsymbol{x}_2) + \boldsymbol{x}_2^* S \boldsymbol{x}_2 \geq \boldsymbol{x}_2^* S \boldsymbol{x}_2$$

. Then (c) and (d) follow by choosing $\boldsymbol{x}_1 = -A_{11}^{-1} A_{12} \boldsymbol{x}_2$ for the above equation. □

B. Code Listing of Example

```
import ngsolve as ngs
from ngsolve import x, y, curl

from hcurl_amg import HCurlAMG  # import the Preconditioner

ngs.SetNumThreads(10)

with ngs.TaskManager():
    mesh = ngs.Mesh('coil_core0.2.vol')
    fes = ngs.HCurl(mesh, order=0, dirichlet='outer')

    u = fes.TrialFunction()
    v = fes.TestFunction()

    mu0 = 1.257e-6

    regularization_factor = 1e-6
    mu_r_iron = 200000

    nu_dict = {'air': 1/mu0, 'coil_mat': 1/mu0,
               'iron' : 1/(mu_r_iron * mu0)}
    nu = ngs.CoefficientFunction([nu_dict[mat] for mat in
        mesh.GetMaterials()])

    a = ngs.BilinearForm(fes, symmetric=False)
    a += ngs.SymbolicBFI(nu * curl(u) * curl(v))
    a += ngs.SymbolicBFI(regularization_factor * nu * u * v)

    c = HCurlAMG(a, {'levels': 30,
                     'vertex_factor': 0.9,
                     'min_edges': 1000,
                     'test': True,
                     'node_amg': True})
    a.Assemble()

    # Setup the impressed current in the coil
    cur = 1000/16
    r = ngs.sqrt(x*x+y*y)
```

```
current_dict = {'coil_mat': (cur * y/r, cur * -x/r, 0),
                'air': (0, 0, 0),
                'iron': (0, 0, 0)}
current = ngs.CoefficientFunction([current_dict[mat] for mat
    in mesh.GetMaterials()])

f = ngs.LinearForm(fes)
f += ngs.SymbolicLFI(current * v)
f.Assemble()

u = ngs.GridFunction(fes)
bvp = ngs.BVP(bf=a, lf=f, gf=gfu, pre=c)
bvp.Do()

ngs.Draw(curl(u), mesh, 'B-field', draw_surf=False)
```

Bibliography

[AFW00] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$. *Numer. Math.*, 85(2):197–217, 2000.

[AFW06] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numer.*, 15:1–155, 2006.

[Axe94] Owe Axelsson. *Iterative solution methods*. Cambridge University Press, Cambridge, 1994.

[BGH+03] Pavel B. Bochev, Christopher J. Garasi, Jonathan J. Hu, Allen C. Robinson, and Raymond S. Tuminaro. An improved algebraic multigrid method for solving maxwell's equations. *SIAM J. Sci. Comput*, 25(2):623–642, 2003.

[Bre] Scott L. Ridgway Brenner, Susanne C. *The Mathematical Theory of Finite Element Methods*. Springer.

[GR86] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986. Theory and algorithms.

[Hac16] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95 of *Applied Mathematical Sciences*. Springer, [Cham], second edition, 2016.

[Hip99] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1999.

[HLRS01] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl. Algebraic multigrid methods based on element preconditioning. *Int. J. Comput. Math.*, 78(4):575–598, 2001.

[HX07] Ralf Hiptmair and Jinchao Xu. Nodal auxiliary space preconditioning in $H(\mathrm{curl})$ and $H(\mathrm{div})$ spaces. *SIAM J. Numer. Anal.*, 45(6):2483–2509, 2007.

[Kra06] J. K. Kraus. On the utilization of edge matrices in algebraic multigrid. In *Large-scale scientific computing*, volume 3743 of *Lecture Notes in Comput. Sci.*, pages 121–129. Springer, Berlin, 2006.

[Mon03] Peter Monk. *Finite element methods for Maxwell's equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2003.

[NP19] Artem Napov and Ronan Perrussel. Revisiting aggregation-based multigrid for edge elements. *Electron. Trans. Numer. Anal.*, 51:118–134, 2019.

[Né86] J.-C. Nédélec. A new family of mixed finite elements in $\mathbb{R}^3$. *Numer. Math.*, 50(1):57–81, 1986.

[RS87] J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, volume 3 of *Frontiers Appl. Math.*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[RS02] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9(3):223–238, 2002.

[Sch97] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, Jul 1997.

[Sch14] J. Schöberl. C++11 implementation of finite elements in ngsolve. Technical report, Institute for Analysis and Scientific Computing, Technische Universität Wien, 2014.

[Sch16] Bernd Schwarzenbacher. Algebraic Multigrid methods for Maxwell's equations, Bachelor thesis., 2016.

[Van95] Petr Vaněk. Fast multigrid solver. *Appl. Math.*, 40(1):1–20, 1995.

[vdV03] Henk A. van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2003.

[Zag06] Sabine Zaglmayr. High order finite element methods for electromagnetic field computation, PhD thesis, 2006.