Algebraic Multigrid Methods for Large Scale Finite Element Equations

DISSERTATION

zur Erlangung des akademischen Grades
“Doktor der Technischen Wissenschaften”

eingereicht von
Dipl.-Ing. Stefan Reitzinger

angefertigt am Institut für Analysis und Numerik,
Abteilung für Numerische Mathematik und Optimierung,
der Technisch–Naturwissenschaftlichen Fakultät
der Johannes Kepler Universität Linz

o.Univ.–Prof. Dr. U. Langer	1. Gutachter
o.Univ.–Prof. Dr. R. H. W. Hoppe	2. Gutachter

Linz, im Jänner 2001

Zusammenfassung

Die vorliegende Dissertation beschäftigt sich mit der Lösung von linearen Gleichungssystemen, welche von der Finite-Elemente-Diskretisierung von partiellen Differentialgleichungen stammen. Im Speziellen werden selbstadjungierte, elliptische partielle Differentialgleichungen der Ordnung 2 betrachtet.

Die entstehenden linearen Gleichungssysteme sind typischerweise symmetrisch und positiv definit und sollen mit einen Verfahren gelöst werden, welches effizient, flexibel und robust ist. Eine Möglichkeit sind algebraische Mehrgittermethoden, welche diesen Anforderungen gerecht werden. Diese Arbeit beschreibt einen allgemeinen Zugang zu algebraischen Mehrgitterverfahren, mit welchen die benötigten Vergröberungs-, Interpolations- und Glättungsoperatoren konstruiert werden können. Eine dafür eingeführte Hilfsmatrix repräsentiert eine 'virtuelle' Gitterhierarchie, in der Gitter- und Operatoranisotropien widergespiegelt werden, sodaß eine geometrische Mehrgittermethode bestmöglich nachgeahmt werden kann. Die Hauptpunkte dabei bestehen in der Erzeugung der benötigten virtuellen Gitterhierarchie und in der richtigen Repräsentation der Freiheitsgrade des zu lösenden Gleichungssystems in der Hilfsmatrix. Basierend auf diesem allgemeinen Konzept werden algebraische Mehrgittermethoden für Matrizen konstruiert, welche von einer Finite-Elemente-Diskretisierung mit Lagrange oder Nédélec Elementen stammen. Zusätzlich zu diesem Konzept stellen wir eine notwendige Bedingung an den Interpolationsoperator, welche garantiert, daß die Eigenschaften der Matrix (ohne wesentliche Randbedingungen) auf gröberen Gittern sich nicht ändert. Die Grobgittermatrix wird mit Hilfe der Galerkin Methode berechnet. Zur Glättung verwenden wir Punkt- oder Block Gauss-Seidel Methoden. Für skalare, selbstadjungierte, elliptische Gleichungen bietet die sogenannte 'Elementvorkonditionierungs-Technik' eine Möglichkeit, die häufig für algebraische Mehrgitterverfahren benötigte M-Matrix Eigenschaft der Systemmatrix zu umgehen. Diese Methode ist wiederum ein Spezialfall des in dieser Arbeit präsentierten allgemeinen Zugangs zu algebraischen Mehrgittermethoden.

Neben den diskutierten algebraischen Mehrgitterverfahren wird auch das effiziente Aufstellen der Matrixhierarchie näher beleuchtet. Insbesondere zeigen wir Möglichkeiten auf, wie die CPU-Zeit für nichtlineare und zeitabhängige Probleme möglichst gering gehalten werden kann. Desweiteren wird eine parallele Version von algebraischen Mehrgittermethoden vorgestellt.

Die vorgestellten algebraischen Mehrgitterverfahren wurden in dem Programmpacket PEBBLES implementiert. Einige numerische Beispiele aus Naturwissenschaft und Technik zeigen das Potential von algebraischen Mehrgitterverfahren.

Diese Arbeit wurde vom Österreichischen 'Fonds zur Förderung der Wissenschaftlichen Forschung' (FWF) im Teilprojekt F1306 des Spezialforschungsbereichs SFB F013 'Numerical and Symbolic Scientific Computing' unterstützt.

Abstract

In these theses the algebraic multigrid method is considered for several different classes of matrices which arise from a finite element discretization of partial differential equations. In particular we consider system matrices that originate from a finite element discretization of some self-adjoint, elliptic partial differential equations of second order. Such matrices are typically symmetric and positive definite.

We are looking for an efficient, flexible and robust solution of the arising system of equations. The algebraic multigrid method is adequate to fulfill these requirements. A general concept to the construction of such multigrid methods is proposed. This concept provides an auxiliary matrix that allows us to construct optimal coarsening strategies, appropriate prolongation and smoothing operators. The auxiliary matrix mimics a geometric hierarchy of grids and comprehend operator- and mesh anisotropies. The key point consists in constructing a 'virtual' finite element mesh and representing the degrees of freedom of the original system matrix on the virtual mesh appropriately. Based on this concept we derive algebraic multigrid methods for matrices stemming from Lagrange and Nédélec finite element discretizations. In addition, we present a necessary condition for the prolongation operator that ensures that the kernel of the system matrix (without essential boundary conditions) is resolved on a coarser level. The coarse grid operator is computed by the Galerkin method as usual. For the smoothing operator standard methods are taken, i.e., point-, or block Gauss-Seidel methods. One remedy for the often required M-matrix property of the system matrix is the so called 'element preconditioning technique' for scalar self-adjoint equations. It turns out that this method can be seen as a special case of the general approach.

Apart from the discussed algebraic multigrid method approach a brief note on the setup phase is given along with the optimization for the setup phase for nonlinear, time-dependent and moving body problems. Furthermore, a parallel algebraic multigrid algorithm is presented, which is applicable to problems originating from Lagrange and Nédélec finite element discretizations.

All algebraic multigrid methods proposed in these theses are implemented in the software package PEBBLES. Finally, numerical studies are given for problems in science and engineering, which show the great potential of the proposed algebraic multigrid techniques.

This work has been supported by the Austrian Science Found - 'Fonds zur Förderung der wissenschaftlichen Forschung' (FWF) - within the subproject F 1306 of the Special Research Program SFB F013 'Numerical and Symbolic Scientific Computing'

Acknowledgements

First of all I wish to express my gratitude to my supervisor Professor U. Langer for the interesting and stimulating discussions we had and for the support and encouragement he has given me in the completion of these theses. At the same time, I am grateful to Professor R.H.W. Hoppe for co-refereeing this dissertation.

Furthermore, I am greatly indebted to my colleagues of the Special Research Program SFB F013 'Numerical and Symbolic Scientific Computing' – I got support and help whenever I needed some in the scientific as well in the social environment. My special thanks goes to G. Haase, M. Kaltenbacher, M. Kuhn, W. Mühlhuber and J. Schöberl for all the fruitful discussions and for proofreading the manuscript.

Last but not least I would like to thank my parents and Karin for the support throughout the years.

Contents

Basic Notations

$\mathbb{R}$, $\mathbb{R}^d$	–	Set of real numbers and set of vectors $\mathbf{x} = (x_i)^T_{i=1,..,d}$, $x_i \in \mathbb{R}$, $i = 1, .., d$.
u, $\mathbf{u}$	–	Scalar valued function, vector valued function.
u_h, $\underline{u}_h$	–	Finite element function and its vector representation.
K_h	–	System matrix $K_h \in \mathbb{R}^{N_h \times N_h}$.
$(.,.)$	–	Euclidean inner product.
$(.,.)_0$	–	L^2-inner product.
$\langle .,. \rangle$	–	Duality product.
Ω, $\Gamma = \partial\Omega$	–	Bounded domain (open and connected subset of $\mathbb{R}^d$, $d = 1, 2, 3$) with sufficiently smooth boundary $\Gamma = \partial\Omega$.
$\mathbf{n}$	–	Normal unit (outward) direction with respect to the boundary $\Gamma = \partial\Omega$ of some domain Ω.
grad	–	Gradient, $\operatorname{grad} u(x) = \left(\frac{\partial u(x)}{\partial x_i}\right)^T_{i=1,..,d}$ for $x \in \mathbb{R}^d$.
Δ	–	Laplace operator, $\Delta u(x) = \sum_{i=1}^d \frac{\partial^2 u(x)}{\partial x_i^2}$ for $x \in \mathbb{R}^d$.
curl	–	curl–operator (also: rot), $\operatorname{curl} u = \left(\frac{\partial u}{\partial x_2}, -\frac{\partial u}{\partial x_1}\right)^T$ for a scalar function $u = u(x_1, x_2)$, $\operatorname{curl} \mathbf{u} = \left(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}, \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}, \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}\right)^T$, for a vector-valued function $\mathbf{u} = (u_i(x_1, x_2, x_3))^T_{i=1,2,3}$.
div	–	Divergence operator, $\operatorname{div} \mathbf{u} = \sum_{i=1}^d \frac{\partial u_i}{\partial x_i}$ for a vector-valued function $\mathbf{u} = (u_i(x_1, x_2, x_3))^T_{i=1,..,d}$.
Id	–	Identity operator.
$\operatorname{rank}(A)$	–	Rank of a matrix $A \in \mathbb{R}^{m \times n}$.

$\operatorname{im} A$	–	Image of an operator A.
$\ker A$	–	Kernel of an operator A.
dim(.)	–	Dimension of the argument.
meas(.)	–	Measure of the argument.
δ_{ij}	–	Kronecker's delta, $\delta_{ij} = 1$ for $i = j$, $\delta_{ij} = 0$ for $i \neq j$.
$L^2(\Omega)$	–	Space of scalar square–integrable functions on Ω.
$(L^2(\Omega))^d$	–	Space of vector valued square–integrable functions on Ω.
$H^1(\Omega)$	–	$H^1(\Omega) = \{v \in L^2(\Omega) : \ \nabla v \in (L^2(\Omega))^d\}$.
$H^{1/2}(\partial\Omega)$	–	Trace space of $H^1(\Omega)$.
$H^1_0(\Omega)$	–	$H^1_0(\Omega) = \{v \in H^1(\Omega) : \ v = 0 \ \text{ on } \partial\Omega\}$.
$(H^1(\Omega))^d$	–	$(H^1(\Omega))^d = \{\mathbf{v} \in (L^2(\Omega))^d : \ \frac{\partial v_i}{\partial x_j} \in L^2(\Omega) \ \forall i, j = 1, \dots, d\}$.
$H(\operatorname{div}, \Omega)$	–	$H(\operatorname{div}, \Omega) = \{\mathbf{v} \in (L^2(\Omega))^3 : \ \operatorname{div} \mathbf{v} \in L^2(\Omega)\}$.
$H_0(\operatorname{div}, \Omega)$	–	$H_0(\operatorname{div}, \Omega) = \{\mathbf{v} \in H(\operatorname{div}, \Omega) : \mathbf{v} \cdot \mathbf{n} = 0 \ \text{ on } \partial\Omega\}$.
$H(\operatorname{curl}, \Omega)$	–	$H(\operatorname{curl}, \Omega) = \{\mathbf{v} \in (L^2(\Omega))^3 : \ \operatorname{curl} \mathbf{v} \in (L^2(\Omega))^3\}$.
$H_0(\operatorname{curl}, \Omega)$	–	$H_0(\operatorname{curl}, \Omega) = \{\mathbf{v} \in H(\operatorname{curl}, \Omega) : \mathbf{v} \times \mathbf{n} = 0 \ \text{ on } \partial\Omega\}$.
AMG	–	Algebraic Multi-Grid.
DOF	–	Degree Of Freedom.
FE	–	Finite Element.
IC	–	Incomplete Cholesky.
MG	–	Multi-Grid.
PCG	–	Preconditioned Conjugate Gradient.
PDE	–	Partial Differential Equation.
PMG	–	Parallel Multi-Grid.
PPCG	–	Parallel Preconditioned Conjugate Gradient.
SPD	–	Symmetric Positive Definite.
SPSD	–	Symmetric Positive Semi-Definite.

Chapter 1

Introduction

Many laws in physics are governed by a scalar partial differential equation (PDE) or a system of PDEs with appropriate boundary conditions and initial conditions in the case of time-dependent problems. Some important representatives of physical models are:

1. *Maxwell equations*, e.g. [44, 59], which describe electromagnetic fields,

2. *Cauchy-Navier equations*, e.g. [17, 43], which model continuum mechanics,

3. *Navier-Stokes equations*, e.g. [24], which are related to fluid dynamics.

The above equations can be treated as they are, or they can be coupled. In the latter case we speak about *Coupled Field Problems.*

The numerical simulation of physical models is of great interest. Instead of performing experiments with a real life experimental setup, such an experiment can be done virtually by computer simulation. For instance, for the development of electrodynamic loudspeakers the experimental setup and the necessary measurements for a new prototype take about 7 days. In contrast to that particular problem, the numerical simulation for one prototype takes a few hours of CPU-time and moreover all parameter settings can be arranged in principle. Consequently, the numerical simulation technique saves a lot of development time and therefore saves money and resources.

The solution of the general settings of the PDEs of item 1., 2., or 3. above is not efficiently possible yet, but a reasonable down-sizing of the above equations is admissible and sufficient for many applications. Down-sizing the problem complexity makes it manageable by numerical methods even for nowadays computer resources.

In most cases the solution of the reduced physical models is only possible if numerical discretization methods are used. The most common methods for elliptic PDEs are the *finite element* (FE) method, the *finite difference* (FD) method and the *finite volume* (FV) method, or alternatively for the Maxwell equations the *finite integration technique* (FIT). For an overview see [27, 89]. For almost the rest of the theses we are considering the FE-method for the discretization of elliptic, self-adjoint second order PDEs. In particular we will concentrate on the robust, and efficient solution of the arising linear system of equations

$$K_h \underline{u}_h = \underline{f}_h \,, \tag{1.1}$$

which is one key task in the overarching numerical simulation process for many applications appearing in science and engineering. For instance in the case of nonlinear, time-dependent, or optimization problems such linear systems have to be solved repeatedly as part of an outer iteration loop, e.g. a Newton iteration or a fixed point scheme for the case of nonlinear equations. Equation (1.1) is typical for numerical discretization schemes, with $K_h \in \mathbb{R}^{N_h \times N_h}$ is a symmetric positive definite (SPD) sparse system matrix, $\underline{f}_h \in \mathbb{R}^{N_h}$ is a given right-hand side and $\underline{u}_h \in \mathbb{R}^{N_h}$ is the solution vector. The number N_h of degrees of freedom (DOFs) in (1.1) behaves asymptotically as $N_h = O(h^{-d})$ (with $d = 1, 2, 3$ the spatial dimension) with the mesh size parameter h, and thus the linear system is usually very large. Moreover the condition number $\kappa(K_h)$ of the system matrix K_h is typically of order $O(h^{-2})$. This is the reason for exhibiting slow convergence in iterative solution methods (see e.g. [4, 40, 68]). Consequently, the efficient, robust (e.g. with respect to the anisotropy) and fast solution of (1.1) is an important aspect of the FE-method and therefore optimal solvers (CPU-time and memory requirement are proportional to N_h) are of interest. Krylov subspace methods [40] together with multigrid methods [39] fulfill these requirements (see [47, 48]).

The geometric multigrid (MG) methods (e.g. [39, 15]) can be used if we have a nested sequence of FE-spaces. These FE-spaces are usually constructed on a hierarchy of conform FE-meshes. By defining appropriate transfer operators the FE-spaces are linked together. For example, let $\mathbb{V}_h$, $\mathbb{V}_H$ ($\mathbb{V}_H \subset \mathbb{V}_h$) be fine and coarse FE-spaces, respectively, and K_h, K_H the corresponding system matrices. Further $P_h : \mathbb{V}_H \mapsto \mathbb{V}_h$ is an appropriate prolongation operator. The two grid process performs ν_F pre-smoothing steps on the fine system, then restricts the residuum on the coarse space. On the coarse level the system is solved exactly and the defect is prolongated on the fine level. Finally, ν_B post-smoothing steps are performed. A recursive application leads to a general MG-cycle (see [39]). The effective interplay of smoother and coarse grid correction results in an optimal solver (preconditioner), see [39].

Algebraic multigrid (AMG) methods are of special interest if geometric multigrid can not be applied, or if standard preconditioners (e.g. incomplete Cholesky (IC), Jacobi) break down because of their non-optimality (see [40]). There are at least two further reasons for using AMG:

1. The discretization provides no hierarchy of FE-meshes, so geometric MG can not be applied.

2. The coarsest grid of a geometric multigrid method is too large to be solved efficiently by a direct or classical iterative solver.

In contrast to geometric multigrid methods, where a grid hierarchy is required explicitly, AMG is able to construct the matrix hierarchy and the prolongation operators by knowing at least the stiffness matrix. The first serious approach to AMG was made in 1982 by A. Brandt, S. McCormick, and J. W. Ruge in [12] and an improved version of [12] can be found in [13] . This method is mainly concerned with SPD matrices K_h, which are additionally M-matrices (or 'small perturbations' of them). In this approach the smoother is usually fixed (e.g. Gauss-Seidel point relaxation) and the prolongation operator is constructed such that the error which is not affected by the smoother is in the range of the prolongation operator. This objective can be realized well for M-matrices [11, 77, 78, 81] but it turned out that it is hard to fulfill for general SPD matrices. A review on this AMG method is made in [82, 83] including also a proposal for the remedy for the M-matrix property. In spite of the fact that this method works robust for M-matrices, the setup time (i.e., the construction of the matrix hierarchy with corresponding transfer operators) and the application as a preconditioner requires plenty of time for many practical problems.

To overcome these drawbacks D. Braess [9] suggested a quite simple AMG method, where the preconditioner is constructed and applied very fast. The method benefits from the piecewise constant interpolation and consequently there is less fill-in on the coarser levels which in turn implies a fast application. On the other hand this method fails at hand if anisotropic structures are considered due to the poor prolongation operator. A related work is given by F. Kickinger in [57], where an improved prolongation was proposed. In a subsequent work the agglomeration technique was combined with a patch smoother [58].

A new technique was developed and analyzed by P. Vanek, J. Mandel, and M. Brezina [84, 66] and a related work is found in [88]. The idea is to construct a 'tentative' prolongation operator which already takes care of the kernel of K_h (without essential boundary conditions), and improve it by some smoothing steps (usually one damped Jacobi step is used). The smoothing step provides to pull energy out of the basis function. This approach is called 'smoothed aggregation'.

A completely new idea was realized by the workers at Lawrence Livermore National Laboratory, which is called AMGe . This method basically assumes access to the element stiffness matrices and is able to compute a measure for 'algebraically smooth error'. In addition an improved prolongation operator can be constructed, which does not rely on the M-matrix property. For the construction and analysis of AMGe see [14, 45]. An element stiffness matrix free version, i.e., working with the assembled system matrix, is given in [41].

Recently, C. Wagner [87] has developed an AMG method that is also applicable for non-symmetric problems. The key point in this approach is to find the best possible two neighbors, which produce the best possible prolongation. The prolongation weights are computed by local minimization problems and the coarsening is done in a similar way. This approach provides a parallel AMG method in a natural way and it can be used additionally for a scalar as well as for a block system of equations.

The above methods are partly able to deal with matrices K_h stemming from an FE-discretization of a system of PDEs (see [84, 41, 57, 82]) and to handle non-symmetric matrices K_h, e.g. arising from convection diffusion problems (see [57, 82, 21, 76, 26, 87, 70]). Other interesting AMG approaches can be found in [23, 20, 68, 86] and references therein. Applications of AMG methods in various practical (engineering) areas are given in [25, 52, 53, 56, 26, 92, 82, 83, 55, 34, 51].

All AMG methods inherently need the efficient interplay of smoothing and coarse grid correction, which are the key ingredients for multigrid methods. The crucial point in the construction of AMG methods is the numerical effort of the coarsening process and the construction of appropriate transfer operators. The challenge is to construct an AMG method with a good convergence rate but rather low costs in the setup and the application. An AMG method always consists of four ingredients, namely

1. the coarsening process,
2. the transfer operators,
3. the smoothing operators, and
4. the coarse grid operators.

In these theses we concentrate on the first two items, i.e., coarsening process and transfer operators. The remaining ingredients, i.e., smoothing and coarse grid operators, are covered by standard methods. For example we use the point-, patch-, and block Gauss-Seidel methods for smoothing operators and the Galerkin method for the coarse grid operator. If it is not stated explicitly we describe the AMG method by a two grid technique in these theses. Therefore we use the subscripts h and H for fine and coarse grid quantities, respectively.

A common feature of all presented AMG methods is that they construct a preconditioner out of the system (or the element matrices) exclusively. We will generalize these AMG approaches in some directions in order to get them more flexible, robust and efficient for certain classes of problems. Since we will apply the AMG method to matrices arising from FE-discretizations our motivation is to construct a hierarchy of 'virtual FE-meshes' comparable to the geometric counterpart. We need more information on the finest grid besides the system matrix, which is usually available in standard FE-codes. This can be achieved by introducing an auxiliary matrix B_h which represents a virtual FE-mesh. For instance, B_h is a nodal distance matrix which also reflects the underlying PDE and therewith B_h reflects anisotropies of the FE-mesh and the PDE. In addition the auxiliary matrix B_h is constructed such that it is an M-matrix. The idea is as follows: First of all, the diagonal entries of B_h are interpreted as nodes and the off-diagonal entries as edges in the virtual FE-mesh. Considering either a Lagrange FE-discretization or a Nédélec FE-discretization the DOFs of K_h are related to the entries of B_h with a disambiguate mapping. Instead of performing a coarsening on the system matrix K_h we perform it on the auxiliary matrix B_h. Because B_h is an M-matrix this is a robust calculation and additionally we are able to define a prolongation operator for the auxiliary system. After the definition of an appropriate transfer operator for B_h a coarse matrix B_H is constructed by Galerkin's method. The resulting coarse grid auxiliary matrix is again interpreted as a virtual FE-mesh and the entries of B_H represents the DOFs of the coarse system matrix. Consequently, we use the coarsening of B_h also for the system matrix K_h and therewith construct a transfer operator for the system matrix. It is worth mentioning that the transfer operators for the auxiliary and the system matrix can not be chosen independently, since the DOFs of the system matrix and the entries of the auxiliary matrix must fit together. Once the matrices are defined on a coarser level, the setup process is applied recursively.

The second key point we interest in is the prolongation operator for the system matrix K_h. It is obvious that the prolongation operators have to be chosen problem depended in order to deal with the properties of the underlying PDE. We assume throughout these theses the knowledge from which variational form and FE-discretization the system matrix K_h stems from. Therewith we know the kernel of the considered variational form (without essential boundary conditions). By this knowledge, a necessary condition on the prolongation operator is posed. This knowledge of the kernel is essential, because by the construction of a coarse grid operator K_H with Galerkin's method it is necessary for the AMG method, that K_H has the 'same' properties as K_h. Especially, the kernel of both operators have to be 'equal' in a certain sense.

First, the general AMG approach is applied to systems of equations that stem from a Lagrange FE-discretization. We will consider three representatives, namely the potential equation, standard solid mechanics and magnetostatic equations. While for the first two examples the FE-discretization with nodal elements is standard, it is more delicate for the magnetostatic problems. In this problem class the function space $H_0(\mathrm{curl}, \Omega)$ is appropriate and it is known that a function $\mathbf{u} \in H_0(\mathrm{curl}, \Omega)$ can be split in the direct sum of a function (in case of convex Ω) $\mathbf{v} \in (H_0^1(\Omega))^3$ plus a function $\mathrm{grad}\,\phi$, $\phi \in H_0^1(\Omega)$, i.e., $(H_0^1(\Omega))^3 \oplus \mathrm{grad}\,H_0^1(\Omega)$. Loosely speaking, the FE-discretization can be applied for the $(H_0^1(\Omega))^3$ and the $H_0^1(\Omega)$ part, separately. As a consequence we arrive at a block system of linear equations. The auxiliary matrix B_h is constructed such that the matrix pattern of K_h and B_h are the same, i.e., $|(B_h)_{ij}| \neq 0 \Leftrightarrow \|(K_h)_{ij}\| \neq 0$. Notice, that an entry $(K_h)_{ij} \in \mathbb{R}^{p \times p}$, $p \geq 1$ is matrix valued in general. The auxiliary matrix is constructed such that the distance of two neighboring grid points as well as the operator type is reflected (see [37]). We suggest the following matrix entry for $(B_h)_{ij}$, $i \neq j$: Either there is no geometrical edge connecting node i and j then $(B_h)_{ij} = 0$, or there is a geometrical edge then the entry is $(B_h)_{ij} = -\frac{1}{\|\mathbf{a}_{ij}\|}$, with $\mathbf{a}_{ij}$ is the distance vector connecting the nodes with the indices i and j and $\|\cdot\|$ is an appropriate norm. The diagonal entry of B_h is defined by $(B_h)_{ii} = \sum_j (B_h)_{ij}$. Then the coarsening process is performed on B_h and the resulting partitioning in coarse and fine nodes is carried over to the system matrix. With this knowledge we are able to construct transfer operators and a block-smoother. The prolongation operator for block systems is computed as for scalar systems, but the entries of the prolongation operator are matrix valued. The prolongation operators are constructed by simple averaging, or more sophisticated, by discrete harmonic extension. Moreover we are able to show in the considered scalar case that the constructed prolongation operators fulfill the general assumptions which are necessary for the kernel of K_h. So far it is not possible to show these preliminaries for the block case. Let us additionally mention that for $B_h \equiv K_h$ in the scalar case and $(B_h)_{ij} = -\|(K_h)_{ij}\|$, $i \neq j$ the technique turns out to be the classical AMG method.

Numerical simulations based on the 3D Maxwell equations are important tools for many applications in science and engineering. While the methods above are closely related to $H^1(\Omega)$-based problems or problems with additional properties (e.g. M-matrix), the matrices stemming from an FE-discretizations of the Maxwell equations with Nédélec FE-functions causes additional difficulties. Hence a new AMG method for this problem class is proposed. For the design of multilevel methods for the linear system (1.1), the Helmholtz decomposition of the vector field into a solenoidal and a gradient field is the key point. Especially the rotation free functions need special treatment in the smoother. A geometric MG method was set up by R. Hiptmair [42] the first time, a different multigrid approach is due to D. Arnold, R. Falk, R. Winther [3]. Many references are found in [7]. For numerical results using geometric MG methods we refer to [63, 79]. An algebraic multigrid (AMG) approach for the solution of (1.1) requires, in addition to the geometric MG components, a proper coarsening strategy. The transfer operators have to be designed with pure algebraic knowledge. In spite of the fact that the FE-matrix K_h is SPD, the classical approaches of [9, 11, 12, 13, 57, 77, 78, 81, 84] and variants of them fail for problem (1.1) at hand. All these methods are designed for SPD problems which either stem from FE-discretizations of H^1-based problems, or need beside the SPD property special characteristics of the system matrix (e.g. M-matrix property). A first AMG approach to solve (1.1) arising from an edge element discretization of the Maxwell equations was made by R. Beck in [6]. The key idea there is to split an $H_0(\text{curl}, \Omega)$ function into an $(H_0^1(\Omega))^3$ function and a gradient function, and apply standard AMG for all components. This differs from our approach, since we apply the coarsening directly for the $H_0(\text{curl}, \Omega)$-matrix. The general AMG approach provides a possibility to cope with such a system. Again an auxiliary matrix B_h is constructed which rather presents the FE-mesh than the matrix pattern of the system matrix. Now, every off-diagonal entry of B_h can be related to an edge (or degree of freedom) in the system matrix K_h. The challenge for the construction of an AMG method is to cope with the kernel of the curl-operator. Basically, we suggest the following strategy:

1. Identify connected pairs of nodes with the connecting edge, i.e., set up a 'node to edge' map.

2. Perform a coarsening technique such that the 'node to edge' map hands over to the coarse level.

3. Define a prolongation operator compatible with the Helmholtz decomposition.

A pivotal point consists in the construction of the 'node to edge' map, to be able to construct the prolongation operator, and the smoother for K_h. Since we are concerned with an FE-discretization, a feasible 'node to edge' map is given by an auxiliary matrix B_h, which provides the opportunity to be interpreted as a virtual FE-mesh as before. Consequently, a 'node to edge' map of this virtual FE-mesh is given in a natural way. If B_h is assumed to be an M-matrix and the prolongation operators for the auxiliary matrix are appropriate, then a 'node to edge' map on the coarse level is given by B_H. The resulting coarse edges on the virtual coarse grid are degrees of freedom on that. An appropriate prolongation operator for the edge FE-space and a suitable smoothing iteration will be defined with the benefit of the 'node to edge' map. By recursion, the setup process and a multigrid method is defined as usual.

Another method which fits in the general AMG concept is called *element preconditioning*. The method was constructed in order to get rid of the M-matrix property of the system matrix which is needed for the classical AMG method [12, 13, 81, 77, 78]. The basic ideas are depicted in [71, 73, 38] and read as follows: Sometimes it is desirable to derive an AMG preconditioner for K_h from a nearby, spectrally equivalent matrix B_h that is sometimes called regularizator [48]. Especially in our case we need an M-matrix for applying the standard AMG efficiently. Thus, if we are able to construct such an SPD regularizator B_h in the class of the M-matrices, then we can derive a good preconditioner C_h for K_h by applying a symmetric AMG cycle to B_h instead of K_h. We propose the element preconditioning method for the construction of a spectrally equivalent M-matrix B_h and therefore we need access to the element stiffness matrices. Additionally, the kernel of all element matrices should be known, or must be calculated in advance. The numerical experiments show that the standard AMG applied to B_h and used as a preconditioning step in the conjugate gradient method results in a robust and efficient solution strategy.

If additional speedup is required, the use of parallel computers is the only way to overcome the limitations. Hereby, distributed memory computers, e.g. PC clusters, are of special practical interest because of the availability and low costs of such machines see [60, 87, 35]. The parallelization based on domain decomposition ideas has been proved to be very efficient. Efficient domain decomposition methods and parallel geometric multigrid methods, both requiring a hierarchy of meshes, have been investigated in [32, 33, 46]. In these theses we apply similar ideas for the design of a general parallel AMG method. There are several sequential versions of AMG methods like [9, 57, 78, 84, 87]. The methods mainly differ in the setup phase, i.e., construction of the matrix hierarchy and prolongation operators. The multilevel cycle is then realized in the classical way and performs well if all components are properly chosen. Parallel versions of AMG are rather rare. Some of them are [22, 60, 87]. In contrast to the parallelization approaches in [22], we base our parallelization strategy for AMG on the non-overlapping domain decomposition. Special attention is paid to a modular implementation of our parallel code, such that other AMG codes fit in the parallelization concept as well. The parallelization was done for scalar and block problems with Lagrange FE-discretization [35] and for the Nédélec FE-discretization [34, 33].

The implementation was done in the algebraic multigrid package PEBBLES (**P**arallel **E**lement **B**ased grey **B**ox **L**inear **E**quation **S**olver). The name originates from the subsequent items.

1. *Parallel:* The package should provide a parallel AMG preconditioner in order to be able to solve really large-scale problems arising from real life applications.

2. *Element Based:* Using PEBBLES in a standard FE-code, the system matrix has to be moved to PEBBLES. In order to do this efficiently, an interface

based on element matrices is appropriate and thus the matrix is stored just once. Moreover the knowledge of element stiffness matrices can be used to construct an efficient and robust preconditioner (see Chapter 5).

3. *grey Box:* We do not claim to construct an optimal and robust solver for general SPD matrices. But we are able to construct preconditioners for certain well defined classes of problems stemming from an FE-, FD-, FV-, or FIT-discretization.

4. *Linear Equation Solver:* The task of solving a linear equation system arises in non-linear, time-dependent or optimization problems. In all these applications the solution of a linear system is an important ingredient.

All these objectives are realized in PEBBLES and up to now the interface is used for the FE-codes FEPP [63], CAPA [65], CAUCHY [91] and MAFIA [19].

Using AMG as a preconditioner we have to expect a relative high setup time (i.e., for the construction of the preconditioner) . This is of special interest if a sequence of linear equations has to be solved, arising from non-linear iteration, time-dependent problems or moving body problems. According to [48] it is sufficient to use a spectrally equivalent matrix (for a definition of spectrally equivalent see Chapter 2) B_h with respect to the original system matrix K_h for the preconditioned conjugate gradient (PCG) method. Thus for solving the current equation, we apply the preconditioner constructed from B_h for several steps. The convergence rate of the PCG method depends on the spectral equivalence constants between K_h and B_h, which can be controlled, e.g. in the case of nonlinear problems via the nonlinearity $\nu(\underline{u}_h)$. In other words, the management of a new setup is done automatically. This can be used in order to minimize the overall computation time. Similar strategies are presented for time-dependent and moving body problems.

The remainder of the work is organized as follows: In Chapter 2 we present a brief overview on the considered problem classes (PDEs), appropriate FE-discretizations and some basics on iterative solvers. In addition the properties of the resulting FE-system matrix are repeated. Our general approach to AMG is given in Chapter 3. Chapter 4 applies the general results to three particular problem classes. Chapter 5 is concerned with the element preconditioning technique. Chapter 6 is related to the parallel version of AMG. In Chapter 7 the AMG program package PEBBLES is presented. Moreover the setup phase is discussed. In particular we show the possibilities for the reduction of the CPU-time for time-dependent, non-linear and moving body problems. Finally, numerical studies are given in Chapter 8.

Chapter 2

Preliminaries

2.1 Sobolev Spaces

Sobolev spaces [1] play an important role in the modern treatment of PDEs. This concerns the right variational formulation as well as the numerical analysis of its FE-discretization. Before we present a brief overview on the required Sobolev spaces, we describe the computational domain Ω. We always assume $\Omega \subset \mathbb{R}^d$ (with $d = 1, 2, 3$ the spatial dimension) be a bounded domain with Lipschitz-boundary Γ, see Fig. 2.1. We assume the boundary Γ to be split in Γ_1 and Γ_2, with $\Gamma_1 \cap \Gamma_2 = \emptyset$ and $\text{meas}(\Gamma_1) \neq 0$, which correspond to a Dirichlet and Neumann boundary condition for further discussion. Additionally, the domain Ω is split into disjoint subdomains $\overline{\Omega} = \overline{\Omega}_1 \cup \overline{\Omega}_2$, corresponding to different material parameters. The interface boundary Γ_I is given by $\Gamma_I = \overline{\Omega}_1 \cap \overline{\Omega}_2$. For reasons of simplicity only two subdomains are illustrated, but the theory can readily be extended to an arbitrary finite number of subdomains. Finally, $\mathbf{n}$ denotes the unit outward normal vector.

Figure 2.1: Principle structure of the computational domain.

A scalar, measurable function $u(\cdot) : \Omega \mapsto \mathbb{R}$ is called square integrable on Ω, if

$$\int_\Omega |u(x)|^2 \, dx < \infty$$

holds. The space of all square integrable functions on Ω is denoted by $L^2(\Omega)$, which is a Hilbert space with inner product and norm

$$(u,v)_0 = \int_\Omega u \cdot v \, dx \quad \text{and} \quad \|u\|_0^2 = (u,u)_0 \,,$$

respectively. The derivatives of functions are defined in the generalized sense of distributions (see [1]). The gradient of a scalar function $u(\cdot)$ is a vector denoted by

$$\operatorname{grad} u = \left(\frac{\partial u}{\partial x_i} \right)^T_{i=1,\dots,d} \,, \tag{2.1}$$

the divergence of a vector valued function $\mathbf{u}(\cdot) = (u_1(\cdot), \dots, u_d(\cdot))^T$ is given by

$$\operatorname{div} \mathbf{u} = \sum_{i=1}^d \frac{\partial u_i}{\partial x_i} \tag{2.2}$$

and the curl of a vector valued function for $d = 3$ reads as

$$\operatorname{curl} \mathbf{u} = \left(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}, \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}, \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \right)^T \,. \tag{2.3}$$

Finally, the product space X^p (for X being a Hilbert space and $p \in \mathbb{N}$) is defined by

$$X^p = X \times \ldots \times X$$

with the norm

$$\| \cdot \|_{X^p}^2 = \| \cdot \|_X^2 + \ldots + \| \cdot \|_X^2 \,.$$

For the further discussion we need the following function spaces:

$$H^1(\Omega) = \{ u \in L^2(\Omega) \, : \, \operatorname{grad} u \in L^2(\Omega) \} \,, \tag{2.4}$$

$$H(\operatorname{curl}, \Omega) = \{ \mathbf{u} \in \left(L^2(\Omega)\right)^d \, : \, \operatorname{curl} \mathbf{u} \in \left(L^2(\Omega)\right)^d \} \,, \tag{2.5}$$

$$H(\operatorname{div}, \Omega) = \{ \mathbf{u} \in \left(L^2(\Omega)\right)^d \, : \, \operatorname{div} \mathbf{u} \in \left(L^2(\Omega)\right)^d \} \,. \tag{2.6}$$

These function spaces are Hilbert spaces equipped with the inner products and norms

$$(u,v)_1 = (u,v)_0 + (\operatorname{grad} u, \operatorname{grad} v)_0 \quad \text{and} \quad \|u\|_1^2 = (u,u)_1 \,, \tag{2.7}$$

$$(\mathbf{u},\mathbf{v})_{\text{curl}} = (\mathbf{u},\mathbf{v})_0 + (\operatorname{curl} \mathbf{u}, \operatorname{curl} \mathbf{v})_0 \quad \text{and} \quad \|\mathbf{u}\|_{\text{curl}}^2 = (\mathbf{u},\mathbf{u})_{\text{curl}} \,, \tag{2.8}$$

$$(\mathbf{u},\mathbf{v})_{\text{div}} = (\mathbf{u},\mathbf{v})_0 + (\operatorname{div} \mathbf{u}, \operatorname{div} \mathbf{v})_0 \quad \text{and} \quad \|\mathbf{u}\|_{\text{div}}^2 = (\mathbf{u},\mathbf{u})_{\text{div}} \,, \tag{2.9}$$

respectively. Boundary conditions, which are essential for our problem class, are introduced in the sense of the trace operator. The following lemma is valid.

Lemma 2.1.1. *The trace operator*

$$T : H^1(\Omega) \mapsto H^{\frac{1}{2}}(\partial\Omega)$$

is a continuous linear mapping from $H^1(\Omega)$ onto $H^{\frac{1}{2}}(\partial\Omega)$.

Proof. A proof can be found in [1]. □

Similar to the space $H^1(\Omega)$ a trace operator can be defined for the spaces $H(\mathrm{curl},\Omega)$ and $H(\mathrm{div},\Omega)$ with traces of $\mathbf{u}\times\mathbf{n}$ and $\mathbf{u}\cdot\mathbf{n}$ in $H^{-\frac{1}{2}}(\partial\Omega)$, respectively (for more details see [24]). A Sobolev space $H^s(\partial\Omega)$ of fractional order $s\in\mathbb{R}$ is defined as in [1]. In this context the spaces

$$\begin{aligned}
H_0^1(\Omega) &= \{u\in H^1(\Omega)\,:\,u_{|\Gamma}\equiv 0\}\,, && (2.10)\\
H_{\Gamma_1}^1(\Omega) &= \{u\in H^1(\Omega)\,:\,u_{|\Gamma_1}\equiv 0\}\,, && (2.11)\\
H_0(\mathrm{curl},\Omega) &= \{u\in H(\mathrm{curl},\Omega)\,:\,(\mathbf{u}\times\mathbf{n})_{|\Gamma}\equiv 0\}\,, && (2.12)\\
H_0(\mathrm{div},\Omega) &= \{u\in H(\mathrm{div},\Omega)\,:\,(\mathbf{u}\cdot\mathbf{n})_{|\Gamma}\equiv 0\} && (2.13)
\end{aligned}$$

are introduced with the same norms defined before for the corresponding spaces without boundary conditions.

The following results can be found in [24].

Theorem 2.1.2. *Every function $\mathbf{v}\in(L^2(\Omega))^3$ has the orthogonal decomposition (Helmholtz Decomposition)*

$$\mathbf{v}=\mathrm{curl}\,\mathbf{u}+\mathrm{grad}\,\phi\,, \qquad (2.14)$$

with respect to the L^2-inner product, where $\phi\in H^1(\Omega)_{|\mathbb{R}}$ is the unique solution of

$$(\mathrm{grad}\,\phi,\mathrm{grad}\,\psi)_0=(\mathbf{v},\mathrm{grad}\,\psi)_0$$

and $\mathbf{u}\in(H^1(\Omega))^3$ satisfies $\mathrm{div}\,\mathbf{u}=0$ in Ω, $\mathrm{curl}\,\mathbf{u}\cdot\mathbf{n}=0$ on Γ.

Proof. A proof is given in [24], Chapter 1, Corollary 3.4. □

Lemma 2.1.3. *For any $\mathbf{v}\in H_0(\mathrm{curl},\Omega)$, there exists a function $\mathbf{w}\in H_0(\mathrm{curl},\Omega)$ and $\phi\in H_0^1(\Omega)$ such that*

$$\mathbf{v}=\mathbf{w}+\mathrm{grad}\,\phi$$

and

$$(\mathbf{w},\mathrm{grad}\,\psi)_0=0\quad\forall\psi\in H_0^1(\Omega)\,.$$

Remark 2.1.4. *In the case of convex Ω, the splitting*

$$H_0(\mathrm{curl},\Omega)=(H_0^1(\Omega))^3\,\oplus\,\mathrm{grad}\,H_0^1(\Omega)$$

of the space $H_0(\mathrm{curl},\Omega)$ is valid. Here the space $\mathrm{grad}\,H_0^1(\Omega)$ is defined by

$$\mathrm{grad}\,H_0^1(\Omega)=\left\{\mathrm{grad}\,\psi\,:\,\psi\in H_0^1(\Omega)\right\}\,.$$

2.2 Classical Formulations

2.2.1 Maxwell Equations

The Maxwell equations [44, 59], which are given by differential equations

$$\operatorname{curl}\mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}\,, \tag{2.15}$$

$$\operatorname{curl}\mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}\,, \tag{2.16}$$

$$\operatorname{div}\mathbf{D} = q\,, \tag{2.17}$$

$$\operatorname{div}\mathbf{B} = 0\,, \tag{2.18}$$

are the mathematical model of magnetic and electric fields in a continuum. Therein, $\mathbf{H}$ denotes the magnetic field strength, $\mathbf{E}$ the electric field strength, $\mathbf{D}$ the electric field density, $\mathbf{B}$ the magnetic induction, $\mathbf{J}$ the current density and q the charge carrier density. In addition the boundary and interface conditions are defined on $\partial\Omega = \Gamma_1 \cup \Gamma_2$ and Γ_I, respectively (see Fig. 2.1), i.e.,

$$\mathbf{X}\cdot\mathbf{n} = 0\,, \tag{2.19}$$

$$\mathbf{Y}\times\mathbf{n} = 0\,, \tag{2.20}$$

$$\big[\mathbf{X}\cdot\mathbf{n}\big]_{\Gamma_I} = \mathbf{X}_2\cdot\mathbf{n} - \mathbf{X}_1\cdot\mathbf{n} = 0\,, \tag{2.21}$$

$$\big[\mathbf{Y}\times\mathbf{n}\big]_{\Gamma_I} = \mathbf{Y}_2\times\mathbf{n} - \mathbf{Y}_1\times\mathbf{n} = 0\,. \tag{2.22}$$

In the last formulae $(\mathbf{X},\mathbf{Y})$ stands either for the pair $(\mathbf{B},\mathbf{H})$ or $(\mathbf{D},\mathbf{E})$. Further the material relations

$$\mathbf{J} = \mathbf{J}_I + \sigma\cdot\mathbf{E} = \mathbf{J}_I + \mathbf{J}_E\,, \tag{2.23}$$

$$\mathbf{D} = \epsilon\cdot\mathbf{E}\,, \tag{2.24}$$

$$\mathbf{B} = \mu\cdot\mathbf{H} \tag{2.25}$$

must hold, with $\mathbf{J}_I$ and $\mathbf{J}_E$ are called the impressed and eddy currents. The non-linear, time-dependent rank two tensors σ, ϵ and μ are assumed to be piecewise constant functions with $\sigma \geq 0$, $\epsilon > 0$ and $\mu > 0$ ($\nu = \mu^{-1}$) if it is not stated differently. In practice the whole set of the Maxwell equations can often be reduced. For the rest we always assume the displacement current density $\frac{\partial \mathbf{D}}{\partial t}$ equal to zero, where $\frac{\partial}{\partial t}$ denotes the time derivative.

Electric Field Problem: In this case we assume a pure electric field problem and consequently $\mathbf{H} \equiv \mathbf{B} \equiv 0$. Because of (2.16) a scalar potential ϕ can be introduced for $\mathbf{E}$, i.e.,

$$\mathbf{E} = -\operatorname{grad}\phi\,,$$

and therefore we obtain with (2.17)

$$-\operatorname{div}(\epsilon\operatorname{grad}\phi) = q \quad \text{in } \Omega \tag{2.26}$$

$$\frac{\partial\phi}{\partial\mathbf{n}} = 0 \quad \text{on } \Gamma \tag{2.27}$$

$$\big[\epsilon\cdot\operatorname{grad}\phi\cdot\mathbf{n}\big]_{\Gamma_I} = 0 \quad \text{on } \Gamma_I\,. \tag{2.28}$$

The general interface condition (2.21) and boundary condition (2.19) results in (2.28) and the Neumann boundary condition (2.27). This boundary value problem will also be called *'potential equation'* further.

For instance, in the case of a capacitor we assume the following conditions on a inner surface $\Gamma_e = \Gamma_{e,0} \cup \Gamma_{e,1}$ (see Fig. 2.2) : $\Gamma_{e,1}$ is the surface of given potential ϕ_0

Figure 2.2: Principle structure of a capacitor.

(e.g. potential of voltage loaded capacitor) and $\Gamma_{e,0}$ is the surface of zero potential (grounded electrode). Thus the potential ϕ on $\Gamma_{e,0}$ is equal zero and the potential on $\Gamma_{e,1}$ has a constant value ϕ_0.

Magnetic Field Problem: The second case is related to the so called quasistatic case (eddy current problem). Because of (2.18) a vector potential $\mathbf{A}$ can be introduced for $\mathbf{B}$, i.e.,

$$\mathbf{B} = \operatorname{curl} \mathbf{A}\,,$$

and consequently we arrive at

$$\begin{aligned} \sigma \cdot \frac{\partial \mathbf{A}}{\partial t} + \operatorname{curl} \mu^{-1} \operatorname{curl} \mathbf{A} &= \mathbf{J}_I \quad \text{in } \Omega\,, && (2.29)\\ \mathbf{A} \times \mathbf{n} &= 0 \quad \text{on } \Gamma\,, && (2.30)\\ \left[\mu^{-1} \cdot \mathbf{A} \times \mathbf{n}\right]_{\Gamma_I} &= 0 \quad \text{on } \Gamma_I && (2.31) \end{aligned}$$

by assuming $\mathbf{E} = -\frac{\partial \mathbf{A}}{\partial t}$. The boundary (2.30) and interface (2.31) conditions are consequences of the corresponding general conditions, i.e., (2.20) and (2.22). Appropriate initial conditions are assumed. Furthermore, the gauge condition

$$\operatorname{div} \mathbf{A} = 0 \tag{2.32}$$

can be assumed because $\mathbf{J}$ is conservative for our applications. This gauge condition is useful to gain uniqueness of the boundary value problem. The special case of a static magnetic field problem (2.29) reduces to

$$\operatorname{curl} \mu^{-1} \operatorname{curl} \mathbf{A} = \mathbf{J}_I\,. \tag{2.33}$$

2.2.2 Cauchy-Navier Equations

An other important set of physical equations model solid mechanical problems. We assume the dynamic, linearized elasticity equations for small deformations and Hook's law. This reads as

$$\begin{aligned} \rho \cdot \frac{\partial^2 \mathbf{u}}{\partial t^2} - \operatorname{div} \boldsymbol{\sigma} &= \mathbf{f} \quad \text{in } \Omega\,, & (2.34)\\ \boldsymbol{\sigma} &= D\boldsymbol{\epsilon}\,, & (2.35)\\ \epsilon_{ij} &= \frac{1}{2} \cdot \Big(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\Big)\,, & (2.36)\\ \mathbf{u} &= 0 \quad \text{on } \Gamma_1\,, & (2.37)\\ \boldsymbol{\sigma} \cdot \mathbf{n} &= \mathbf{g} \quad \text{on } \Gamma_2\,, & (2.38) \end{aligned}$$

with appropriate initial conditions. Therein

$$\mathbf{u} = (u_1, u_2, u_3)^T\,, \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_{11} & \epsilon_{12} & \epsilon_{13} \\ \epsilon_{21} & \epsilon_{22} & \epsilon_{23} \\ \epsilon_{31} & \epsilon_{32} & \epsilon_{33} \end{pmatrix}$$

denote the displacement vector, the stress tensor and the strain tensor, respectively. The material density is given by ρ and D is the tensor of elastic coefficients. Further the div-operator is defined component-wise, i.e.,

$$(\operatorname{div} \boldsymbol{\sigma})_i = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}\,.$$

A special case of the above equations are the static Lamé equations, which are valid for an isotropic material, i.e.,

$$-\mu\Delta\mathbf{u} - (\lambda + \mu) \operatorname{grad} \operatorname{div} \mathbf{u} = \mathbf{f} \qquad (2.39)$$

and $\mu\,, \lambda > 0$ are called the Lamé parameters.

2.2.3 Coupled Field Problems

Apart from the single field problems we have a focus on coupled field problems, which occur when at least two physical fields interact on a given domain Ω. At this point we concentrate on a simple coupled field problem that connects an electric field and mechanical field (see [50, 49, 2, 64, 85]). The essential part is the coupling which is modeled as a simple right hand side coupling, i.e.,

$$\begin{aligned} -\mu\Delta\mathbf{u} - (\lambda + \mu) \operatorname{grad} \operatorname{div} \mathbf{u} &= \mathbf{f}_E \quad \text{in } \Omega\,, & (2.40)\\ \mathbf{u} &= 0 \quad \text{on } \Gamma_1\,, & (2.41)\\ \boldsymbol{\sigma} \cdot \mathbf{n} &= \mathbf{g} \quad \text{on } \Gamma_2\,, & (2.42) \end{aligned}$$

and the electric part is given by

$$\begin{aligned} -\operatorname{div}(\epsilon \operatorname{grad} \phi(\mathbf{u})) &= q(\mathbf{u}) \quad \text{in } \Omega\,, & (2.43)\\ \frac{\partial \phi}{\partial \mathbf{n}} &= 0 \quad \text{on } \Gamma\,, & (2.44)\\ \big[\epsilon \operatorname{grad} \phi \cdot \mathbf{n}\big]_{\Gamma_I} &= 0\,. & (2.45) \end{aligned}$$

The electrostatic force $\mathbf{f}_E$ is calculated by

$$\mathbf{f}_E = \int_{\Sigma_I} \mathbf{T}_E \cdot \mathbf{n}\, d\Sigma\,,$$

with the electrostatic force tensor

$$\mathbf{T}_E = \begin{pmatrix} \epsilon E_x^2 - \frac{1}{2}\epsilon|\mathbf{E}|^2 & \epsilon E_x E_y & \epsilon E_x E_z \\ \epsilon E_y E_x & \epsilon E_y^2 - \frac{1}{2}\epsilon|\mathbf{E}|^2 & \epsilon E_y E_z \\ \epsilon E_z E_x & \epsilon E_z E_y & \epsilon E_z^2 - \frac{1}{2}\epsilon|\mathbf{E}|^2 \end{pmatrix}\,,$$

$\mathbf{E} = (E_x, E_y, E_z)$ and Σ_I the surface on which the force acts.

2.3 Primal Variational Formulation

The last section gave some examples of PDEs which have to be solved in an appropriate numerical way. It is hardly possible to solve such boundary value problems in a classical analytic way. Thus we are using the FE-method for the approximate solution of the boundary value problems discussed in Section 2.2. In order to be able to apply the FE-technique, the variational formulation is used, see e.g. [16]. We consider only the primal variational formulation for second order self-adjoint elliptic PDEs. The subsequent examples are obtained by partial integration and appropriate incorporation of the boundary conditions. Moreover appropriate Sobolev spaces are defined and a potential homogenization of the essential boundary conditions is made. Therefore, all variational problems to be considered can be written in the abstract setting:

$$\text{find } u \in \mathbb{V}_B : \; a(u,v) = \langle f, v\rangle \quad \forall v \in \mathbb{V}_B\,, \tag{2.46}$$

with the Hilbert space $\mathbb{V}$ and $\mathbb{V} \supset \mathbb{V}_B$, the bilinear form $a(\cdot,\cdot) : \mathbb{V} \times \mathbb{V} \mapsto \mathbb{R}$ and the duality product $\langle\cdot,\cdot\rangle : \mathbb{V}_B^* \times \mathbb{V}_B \mapsto \mathbb{R}$ with $\mathbb{V}_B^*$ the dual space of $\mathbb{V}_B$.

An important property is the kernel of the bilinear form $a(\cdot,\cdot)$, i.e.,

$$\mathbb{V}_0 = \{u \in \mathbb{V} \,|\, a(u,v) = 0 \;\forall v \in \mathbb{V}\}\,, \tag{2.47}$$

can be expressed as

$$\mathbb{V}_0 = \Lambda\mathbb{Q} = \{\Lambda u \,|\, u \in \mathbb{Q}\}\,,$$

by introducing a bounded, linear operator $\Lambda : \mathbb{Q} \subset \mathbb{V} \mapsto \mathbb{V}$. Note, the kernel of the bilinear form is of special interest for the construction of an AMG method.

Electrostatic Field Problem: In a first case we consider the potential equation, where we can usually use the spaces $\mathbb{V} = H^1(\Omega)$ and $\mathbb{V}_B = H^1_{\Gamma_1}(\Omega)$. The bilinear form is given by

$$a(u,v) = \int_\Omega \operatorname{grad} v^T \epsilon \operatorname{grad} u\, dx \tag{2.48}$$

and $f \in L^2(\Omega)$ is assumed. Furthermore with

$$\Lambda = \mathrm{Id} \quad \text{and} \quad \mathbb{Q} = \{u \in \mathbb{V} \,|\, \operatorname{grad} u = 0\} = \{b \,|\, b \in \mathbb{R}\}$$

the kernel is described. This result follows immediately by the mean value theorem. In the latter formula Id is the abbreviation for the identity operator.

Linearized Elasticity: In these theses we are neither considering geometric nor material locking effects and therefore we use the Sobolev spaces $\mathbb{V} = (H^1(\Omega))^d$ and $\mathbb{V}_B = (H^1_{\Gamma_1}(\Omega))^d$. Consequently the bilinear form for (2.49) reads as

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \epsilon(\mathbf{v})^T D \epsilon(\mathbf{u}) \, dx \,. \tag{2.49}$$

Moreover a right-hand side $\mathbf{f} \in (L^2(\Omega))^d$ is assumed. With the definition

$$\Lambda = \mathrm{Id} \quad \text{and} \quad \mathbb{Q} = \{\mathbf{u} \in \mathbb{V} \,|\, \epsilon(\mathbf{u}) = 0\} = \left\{\mathbf{a} \times \mathbf{x} + \mathbf{b} \,|\, \mathbf{a}, \, \mathbf{b} \in \mathbb{R}^d\right\} ,$$

the kernel is described (see e.g. [10], Chapter 6, Remark 3.2).

Magnetostatic Field Problem I: Let us consider the linear, static equation (2.33) with the function spaces $\mathbb{V} = H(\operatorname{curl}, \Omega)$ and $\mathbb{V}_B = H_0(\operatorname{curl}, \Omega)$. In Section 2.1 the splitting $\mathbb{V}_B = H_0(\operatorname{curl}, \Omega) = (H^1_0(\Omega))^3 \oplus \operatorname{grad} H^1_0(\Omega)$ was presented. This fact and the gauge condition (2.32) imply the bilinear form

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \operatorname{curl} \mathbf{v}^T \mu^{-1} \operatorname{curl} \mathbf{u} \, dx + \int_\Omega \operatorname{div} \mathbf{v} \mu^{-1} \operatorname{div} \mathbf{u} \, dx \,, \tag{2.50}$$

by adding the term $\int_\Omega \operatorname{div} \mathbf{v}^T \mu^{-1} \operatorname{div} \mathbf{u} \, dx$ (which vanishes in the continuous solution) to (2.33) (see [8, 54]). This can be seen as a penalty formulation where the gauge condition (2.32) is penalized by μ^{-1} and added to the original equation. Further we assume a right-hand side $\mathbf{f} \in (L^2(\Omega))^d$. If no essential boundary conditions are imposed then

$$\Lambda = \mathrm{Id} \quad \text{and} \quad \mathbb{Q} = \{\operatorname{grad} u \in \mathbb{V} \,|\, \operatorname{div} \operatorname{grad} u = 0\}$$

represent the kernel of (2.50). The relation $\Lambda\mathbb{Q} \subseteq \mathbb{V}_0$ is obvious. On the other hand the inclusion $\mathbb{V}_0 \subseteq \Lambda\mathbb{Q}$ can be seen as follows: Let the variational form (2.50) be equal zero and set especially $\mathbf{v} = \mathbf{u}$, i.e.,

$$\int_\Omega \operatorname{curl} \mathbf{u}^T \mu^{-1} \operatorname{curl} \mathbf{u} \, dx + \int_\Omega \operatorname{div} \mathbf{u} \mu^{-1} \operatorname{div} \mathbf{u} \, dx = 0 \,.$$

Since both terms of the above integral are positive, each of them must vanish. The kernel of the curl-operator is the space of all gradient fields for convex Ω (see below). Therefore the relation $\operatorname{div} \operatorname{grad} \phi = 0$ must hold, with $\operatorname{grad} \phi$ being in the kernel of the curl-operator. This ensures the second term to be equal zero. This variational formulation is especially suited for Lagrange FE-functions (see Section 2.4).

Magnetostatic Field Problem II: Let us again consider the linear, static Maxwell equation (2.33) with the function spaces $\mathbb{V} = H(\mathrm{curl}, \Omega)$ and $\mathbb{V}_B = H_0(\mathrm{curl}, \Omega)$. Because the resulting variational form has a non-trivial kernel we add the term $\int_\Omega \sigma \cdot \mathbf{uv}\, dx$ (with $\sigma > 0$ be an artificial conductivity) in order to ensure uniqueness of the boundary value problem, i.e., the bilinear form results in

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \mathrm{curl}\, \mathbf{v}^T \mu^{-1} \,\mathrm{curl}\, \mathbf{u}\, dx + \int_\Omega \sigma \cdot \mathbf{uv}\, dx \tag{2.51}$$

and the right-hand side $\mathbf{f} \in (L^2(\Omega))^d$ is assumed. For $\sigma = 0$ the kernel is non-trivial and consists of all gradient fields if the domain Ω is simply connected (see e.g. [24], Chapter 1, Theorem 2.9). By defining

$$\Lambda = \mathrm{grad} \quad \text{and} \quad \mathbb{Q} = H_0^1(\Omega)$$

the kernel fits into our general representation scheme. This formulation is suited for Nedéléc FE-discretization (Section 2.4).

The *Lax-Milgram Lemma* is the basic tool which provides the existence and uniqueness of the solution of the variational forms above under appropriate assumptions.

Lemma 2.3.1. *Let $\mathbb{V}_B$ be a real Hilbert space and $a(\cdot,\cdot) : \mathbb{V}_B \times \mathbb{V}_B \mapsto \mathbb{R}$ an elliptic, bounded bilinear form, i.e., there exist positive constants μ_1 and μ_2 such that*

$$|a(u, v)| \le \mu_2 \cdot \|u\|_{\mathbb{V}_B} \cdot \|v\|_{\mathbb{V}_B}$$

and

$$a(u, u) \ge \mu_1 \cdot \|u\|^2_{\mathbb{V}_B}\,,$$

respectively. Further let $f \in \mathbb{V}_B^*$. Then the equation

$$\textit{find } u \in \mathbb{V}_B \;: \quad a(u, v) = \langle f, v\rangle \quad \forall v \in \mathbb{V}_B$$

has an unique solution and

$$\|u\|_{\mathbb{V}_B} \le \frac{1}{\mu_1} \cdot \|f\|_{\mathbb{V}_B^*}\,.$$

Proof. The proof can be found in [10], Chapter 2, Theorem 2.5. □

The same result carries over to boundary value problems where $a(\cdot,\cdot)$ is semi-elliptic in $\mathbb{V}$, if $f \in \mathbb{V}_0^\perp$, i.e.,

$$\langle f, v\rangle = 0 \quad \forall v \in \mathbb{V}_0\,.$$

The latter condition follows from the *Fredholm Theory* (see e.g. [61], Chapter 8).

2.4 Finite Element Method

The variational forms (2.48), (2.49), (2.50), and (2.51) of Section 2.3 are the starting point for the FE-discretization. Let us consider the general variational problem (2.46). In order to get a finite substitution of the original equation, the space $\mathbb{V}$ is approximated by a sequence of subspaces $\{\mathbb{V}_h\}_{h>0} \subset \mathbb{V}$ with h being the mesh-size parameter. The FE-discretization is based on a regular partition τ_h of the domain Ω into, e.g. tetrahedral, finite elements T [16]. In addition we impose on the FE-discretization

$$h_1 < h_2 \Rightarrow \mathbb{V}_{h_2} \subset \mathbb{V}_{h_1}$$

and in the limit case $h \to 0$ we require $\mathbb{V}_h = \mathbb{V}$. The same properties are assumed on the subspace $\mathbb{V}_B$. Thus we require a sequence of finite dimensional subspaces $\{\mathbb{V}_{Bh}\}_{h>0} \subset \mathbb{V}_B$ in order to approximate the space $\mathbb{V}_B$. Then (2.46) changes to

$$\text{find } u_h \in \mathbb{V}_{Bh} \; : \; a(u_h, v_h) = \langle f, v_h \rangle \quad \forall v_h \in \mathbb{V}_{Bh} \tag{2.52}$$

which is equivalent to the linear system of equations

$$K_h \underline{u}_h = \underline{f}_h \,, \tag{2.53}$$

by the FE-isomorphism

$$G_h^{sys} : V_h \mapsto \mathbb{V}_h \,,$$

with $V_h = \mathbb{R}^{N_h}$. The FE-spaces specified for our particular model problems are pretty much standard.

1. For the scalar problem of the potential equation (2.48) the standard nodal FE-functions are used [16] which are $H^1(\Omega)$-conforming. For each element $T \in \tau_h$ we define the space

   $$\mathbb{V}_{h,T} = \{(\mathbf{a} \cdot \mathbf{x} + b)|_T \; : \; \mathbf{a} \in \mathbb{R}^d, b \in \mathbb{R}\},$$

   and set

   $$\mathbb{V}_h = \{v \in H^1(\Omega) \; : \; v|_T \in \mathbb{V}_{h,T}\} \,.$$

   In an analogous way we define a conforming $(H^1(\Omega))^3$ FE-space [10, 43], which is used for the variational form (2.49).

2. For the weak formulation (2.50), the space $H_0(\text{curl}, \Omega)$ is split into the $(H_0^1(\Omega))^3$ and the grad $H_0^1(\Omega)$ part due to Remark 2.1.4. Consequently we can use again the Lagrange FE-functions of item 1. for both parts.

3. The canonical finite elements for the approximation of $H(\text{curl}, \Omega)$ of (2.51) are Nedéléc FE-functions [69]. For each element $T \in \tau_h$ we define the space

   $$\mathbb{V}_{h,T} = \{(\mathbf{a} \times \mathbf{x} + \mathbf{b})|_T \; : \; \mathbf{a}, \mathbf{b} \in \mathbb{R}^d\},$$

and set

$$\mathbb{V}_h = \{\mathbf{v} \in H(\mathrm{curl}, \Omega) \,:\, \mathbf{v}|_T \in \mathbb{V}_{h,T}\}\,.$$

The integrals of the tangential components along edges give the proper degrees of freedom. The edge elements have the property that the tangential component is continuous while the normal component is free to jump, i.e., an edge element discretization is $H(\mathrm{curl}, \Omega)$-conforming. This is important for (2.51) in the case of non-convex domains Ω, or if the coefficient function ν has jumps to get a 'good approximation' of the continuous solution.

The 'kernel' space $\mathbb{Q} \subset \mathbb{V}$ is assumed to be approximated by a finite space $\mathbb{Q}_h$ such that

$$\|q - q_h\|_{\mathbb{V}} \leq O(h) \quad \forall q \in \mathbb{Q}\,,$$

if h tends to zero. In addition we define an FE-isomorphism

$$G_h^{ker} : Q_h \mapsto \mathbb{Q}_h\,,$$

with Q_h being an appropriate parameter space.

After the definition of FE-spaces the discrete mapping $\Lambda_h : Q_h \mapsto V_h$, which maps the discrete kernel into the discrete space, is defined by

$$\Lambda_h \underline{q}_h = (G_h^{sys})^{-1}\,\Lambda\, G_h^{ker}\,\underline{q}_h \quad \forall \underline{q}_h \in Q_h$$

and thus the discrete kernel reads as

$$\begin{aligned} V_{0h} &= \{\underline{u}_h \in V_h \,|\, a(G_h^{sys}\underline{u}_h, G_h^{sys}\underline{v}_h) = 0 \;\forall \underline{v}_h \in V_h\} \\ &= \{\underline{q}_h \in Q_h \,|\, a(\Lambda G_h^{ker}\underline{q}_h, G_h^{sys}\underline{v}_h) = 0 \;\forall \underline{v}_h \in V_h\} \;=\; \Lambda_h Q_h\,. \end{aligned}$$

By defining coarser spaces $\mathbb{V}_H \subset \mathbb{V}_h$ and $\mathbb{Q}_H \subset \mathbb{Q}_h$ (which is explained in more detail in Chapter 4) and the appropriate FE-isomorphisms G_H^{sys} and G_H^{ker}, the operator $\Lambda_H : Q_H \mapsto V_H$ is defined by

$$\Lambda_H \underline{q}_H = (G_H^{sys})^{-1}\,\Lambda\, G_H^{ker}\,\underline{q}_H \quad \underline{q}_H \in Q_H\,.$$

Next, we assume a transfer operator $P_h^{sys} : V_H \mapsto V_h$ to be given with full rank, then the kernel of the coarse space is given by

$$V_{0H} = \{\underline{v}_H \,|\, P_h^{sys}\underline{v}_H \in V_{0h}\}\,, \tag{2.54}$$

in the case of using Galerkin's method for the coarse grid operator, i.e.,

$$K_H = (P_h^{sys})^T\, K_h\, P_h^{sys}\,.$$

Remark 2.4.1. *In the subsequent items we presented the discrete kernels of the considered problem classes. We use the vector of coordinates* $(\mathbf{x}, \mathbf{y}, \mathbf{z})^T \in \mathbb{R}^{N_h}$ *and therewith* $\mathbf{x}$, $\mathbf{y}$ *and* $\mathbf{z} \in \mathbb{R}^{M_h}$ *are the* $x-$, $y-$ *and* z*-coordinates of the grid points, respectively. The vector* $\mathbf{1} \in \mathbb{R}^{M_h}$ *denotes the constant vector.*

1. *The discrete kernel arising from (2.48) is given by*
$$V_{0h} = \left\{\underline{v}_h \in V_h \mid \operatorname{grad} G_h^{sys} \underline{v}_h = 0\right\} = Q_h \tag{2.55}$$
and
$$Q_h = \operatorname{span}\{\mathbf{1}\}$$
holds.

2. *The discrete kernel stemming from (2.49) reads as follows*
$$V_{0h} = \left\{\underline{\mathbf{v}}_h \in V_h \,|\, \epsilon(G_h^{sys} \underline{\mathbf{v}}_h) = 0\right\} = Q_h \tag{2.56}$$
and we have
$$Q_h = \operatorname{span}\left\{ \begin{pmatrix}\mathbf{1}\\\mathbf{0}\\\mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\\mathbf{1}\\\mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\\mathbf{0}\\\mathbf{1}\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\\mathbf{z}\\-\mathbf{y}\end{pmatrix}, \begin{pmatrix}-\mathbf{z}\\\mathbf{0}\\\mathbf{x}\end{pmatrix}, \begin{pmatrix}\mathbf{y}\\-\mathbf{x}\\\mathbf{0}\end{pmatrix} \right\}.$$

3. *The discrete kernel of (2.50) is given by*
$$V_{0h} = \left\{(G_h^{sys})^{-1} \operatorname{grad} G_h^{sys} \underline{v}_h \in V_h \mid \operatorname{div} \operatorname{grad} G_h^{sys} \underline{v}_h = 0\right\} = Q_h\,. \tag{2.57}$$

4. *The discrete kernel of the* curl*-operator for (2.51) is defined by*
$$V_{0h} = \{\underline{v}_h \in V_h \mid \operatorname{curl} G_h^{sys} \underline{v}_h = 0\} = \operatorname{grad}_h Q_h\,, \tag{2.58}$$
with the 'discrete' gradient operator $\operatorname{grad}_h : Q_h \to V_{0h}$
$$\operatorname{grad}_h \underline{q}_h = (G_h^{sys})^{-1} \operatorname{grad} G_h^{ker} \underline{q}_h \quad \forall \underline{q}_h \in Q_h\,. \tag{2.59}$$

5. *The challenging task is to ensure* $\Lambda_H Q_H = V_{0H}$ *under the assumption that* $\Lambda_h Q_h = V_{0h}$*. This property is closely related to the transfer operators* $P_h^{sys} : V_H \mapsto V_h$ *and* $P_h^{ker} : Q_H \mapsto Q_h$*.*

2.5 Iterative Solvers

Before we recall the basic theory of iterative solvers (e.g. Richardson and Conjugate Gradient iteration), we give some notations and results. For a detailed discussion we refer to e.g. [4, 40, 68]. In spite of the fact that some results hold for more general matrices we assume the matrices $A, B \in \mathbb{R}^{N\times N}$ to be symmetric and positive (semi) definite (SP(S)D) . In addition the entries of a matrix and a vector are given by
$$A = \{a_{ij}\}_{i,j=1,\dots,N} \qquad \underline{u} = \{u_i\}_{i=1,\dots,N}\,.$$

Remark 2.5.1.

1. *The transposed of* A *is denoted by* A^T*. The dimension of* A *is given by* $\dim(A) = N$ *and the rank of* A *is equal to* $\operatorname{rank}(A) = N - \dim(\ker(A))$*, where* $\ker(A) = \{\underline{u} \in \mathbb{R}^N : A\underline{u} = 0\}$ *is the kernel of* A*. The matrix* A *is called definite iff* $\operatorname{rank}(A) = \dim(A) = N$*. The matrix* A *is called semi-definite iff* $\dim(\ker(A)) \neq 0$*.*

2. *The eigenvalues* $\{\mu_i(A)\}_{i=1,\dots,N}$ *with corresponding eigenvectors* $\{\phi_i(A)\}_{i=1,\dots,N}$ *of* A *are assumed to fulfill the following two properties*

$$0 \le \mu_1(A) \le \ldots \le \mu_N(A) \quad \textit{and} \quad (\phi_i(A), \phi_j(A)) = \delta_{ij}\,,$$

where $(\cdot,\cdot)$ *is the Euclidean inner product and* δ_{ij} *is the Kronecker symbol. The maximal and minimal non-zero eigenvalue (and the corresponding eigenvector) are denoted by* $\mu_{\max}(A) = \mu_N(A)$ *(*$\phi_{\max}(A) = \phi_N(A)$*) and* $\mu_{\min}(A) = \mu_k(A)$ *(*$\phi_{\min}(A) = \phi_k(A)$*), with* $k = \min\{i \,:\, \mu_i(A) \neq 0\}$.

3. *The generalized condition number of* A *is given by*

$$\kappa(A) = \frac{\mu_{\max}(A)}{\mu_{\min}(A)}\,.$$

If A *is regular we call it condition number .*

4. A *and* B *are called spectrally equivalent if*

$$\exists c_1, c_2 \in \mathbb{R}^+ : c_1 \cdot (B\underline{u}, \underline{u}) \le (A\underline{u}, \underline{u}) \le c_2 \cdot (B\underline{u}, \underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^N\,,$$

which is briefly denoted by

$$c_1 \cdot B \le A \le c_2 \cdot B$$

and we use the fact that

$$\kappa(B^\dagger A) \le \frac{c_2}{c_1}\,,$$

where $B^\dagger$ *is the pseudo-inverse of* B.

5. *The spectral radius* $\rho(A)$ *of* A *is given by*

$$\rho(A) = \mu_{\max}(A)\,.$$

6. *The* $N_h \times N_h$ *identity matrix is denoted by* I_h.

7. *Finally, we define the set of matrices* Z_N *by*

$$Z_N = \{A \in \mathbb{R}^{N\times N} \,:\, a_{kk} > 0,\, a_{ij} \le 0\, \forall i \neq j \quad i,j,k = 1,\dots,N\}\,.$$

In the case $A \in Z_N$ *is regular, the matrix* A *is called M-matrix.*

Remark 2.5.2. *For our applications the resulting system matrices* $K_h \in \mathbb{R}^{N_h\times N_h}$ *have the following properties:*

1. K_h *is SP(S)D.*

2. $\dim(K_h) = N_h = O(h^{-d})$, *where* h *denotes the typical average mesh size, i.e., the system matrix can be a large scale matrix.*

3. *$\kappa(K_h) = O(h^{-2})$, i.e., the generalized condition number is very large as h tends to zero.*

4. *The system matrix K_h has a sparse pattern, i.e., the number of non-zero entries (*NNE_h*) is of order $O(h^{-d})$. In addition the average number of non-zero entries per row is given by*

$$\mathrm{NME}_h = \frac{\mathrm{NNE}_h}{N_h}\,.$$

5. *The system matrix is stored in such a way, that the p unknowns related to a node (edge) are stored in a sub-matrix k_{ij}, i.e.,*

$$K_h = (k_{ij})_{i,j=1,\dots,M_h}\,, \quad k_{ij} \in \mathbb{R}^{p\times p}$$

with M_h the number of nodes (edges). Thus the number of unknowns is given by $N_h = M_h \cdot p$.

Before we proceed, a preconditioner C_h^{-1} which is spectrally equivalent to K_h (i.e. $c_1 \cdot C_h \leq K_h \leq c_2 \cdot C_h$) is assumed. We call

$$\|\underline{u}_h\|^2_{K_h} = (K_h\underline{u}_h, \underline{u}_h)$$

the K_h-*energy norm* of $\underline{u}_h$ and

$$\|\underline{u}_h\|^2_{K_hC_h^{-1}K_h} = (K_hC_h^{-1}K_h\underline{u}_h, \underline{u}_h)$$

the $K_hC_h^{-1}K_h$-*energy norm* of $\underline{u}_h$.

Now, let us consider the linear equation (2.53) with the properties given in Remark 2.5.2. Further we assume the right-hand side $\underline{f}_h \in \ker(K_h)^\perp$ in order to gain solvability. First the *Richardson iteration* scheme (Alg. 1) is considered with a preconditioner C_h^{-1}. The iteration matrix is given by

$$R_h = I_h - \tau \cdot C_h^{-1} \cdot K_h\,,$$

and the spectral radius of R_h is bounded by

$$\rho(R_h) \leq \max\{|1-\tau\cdot c_1|,\, |1-\tau\cdot c_2|\}\,.$$

It is well known that the optimal damping parameter is given by $\tau = \frac{1}{c_1+c_2}$ and therewith the convergence rate is bounded by

$$\rho(R_h) \leq \frac{\kappa(C_h^{-1}K_h) - 1}{\kappa(C_h^{-1}K_h) + 1}$$

The second case is the *preconditioned conjugate gradient (PCG) method* with preconditioner C_h^{-1}. In Alg. 2 the algorithm is presented. The PCG-method is a nonlinear iteration scheme. The advantage of this method consists in the fact that we do not need to compute the damping parameter, i.e., the spectral constants are not

Algorithm 1 Richardson Iteration (K_h,C_h,$\underline{u}_h$,$\underline{f}_h$)

```
define the damping parameter $0 < \tau < \frac{2}{c_2}$
define the relative error bound $\varepsilon > 0$
$\underline{u}_h^0 \leftarrow \underline{u}_h$                                    get the start vector
while $\|\underline{f}_h - K_h\underline{u}_h\| > \varepsilon \cdot \|\underline{f}_h - K_h\underline{u}_h^0\|$ do
    $\underline{u}_h \leftarrow \underline{u}_h + \tau \cdot C_h^{-1} \cdot (\underline{f}_h - K_h\underline{u}_h)$
end while
```

Algorithm 2 Preconditioned conjugate gradient method(K_h,C_h,$\underline{u}_h$,$\underline{f}_h$)

```
define the relative error bound $\varepsilon > 0$
$\underline{u}_h^0 \leftarrow \underline{u}_h$                                    get the start vector
$\underline{r}_h \leftarrow \underline{f}_h - K_h\underline{u}_h^0$
$\underline{d}_h \leftarrow C_h^{-1} \cdot \underline{r}_h$
$\underline{s}_h \leftarrow \underline{d}_h$
while $\|\underline{f}_h - K_h\underline{u}_h\| > \varepsilon \cdot \|\underline{f}_h - K_h\underline{u}_h^0\|$ do
    $\gamma \leftarrow (\underline{s}_h, \underline{r}_h)$
    $\alpha \leftarrow \frac{\gamma}{(K_h\underline{d}_h, \underline{d}_h)}$
    $\underline{u}_h \leftarrow \underline{u}_h + \alpha \cdot \underline{d}_h$
    $\underline{r}_h \leftarrow \underline{r}_h - \alpha \cdot K_h\underline{d}_h$
    $\underline{s}_h \leftarrow C_h^{-1} \cdot \underline{r}_h$
    $\beta \leftarrow \frac{\gamma}{(\underline{s}_h, \underline{r}_h)}$
    $\underline{d}_h \leftarrow \underline{s}_h + \beta \cdot \underline{d}_h$
end while
```

required in order to get an optimal convergence rate. The error in the l^{th} iteration step of the PCG method in the K_h-energy norm is bounded by (see e.g. [40])

$$\|\underline{u}_h^l - \underline{u}_h^*\|_{K_h} \leq q_l \cdot \|\underline{u}_h^0 - \underline{u}_h^*\| \tag{2.60}$$

with

$$q_l = \frac{2q^l}{1 + q^{2l}}, \quad q = \frac{\sqrt{\kappa(C_h^{-1}K_h)} - 1}{\sqrt{\kappa(C_h^{-1}K_h)} + 1}.$$

Further, $\underline{u}_h^l$ denotes the l^{th} iterate of the PCG iteration step, $\underline{u}_h^*$ is the exact solution of (2.53) and $\underline{u}_h^0$ is the given start vector. An acceleration of the convergence by the square root is given.

For both iteration schemes the stopping criterion is based on the Euclidean norm of the residual. Let us mention that especially for the PCG method the stopping criterion based on the $K_h C_h^{-1} K_h$-energy norm is favorable compared to other norms since this norm is automatically calculated by the PCG iteration.

The solution strategies motivate us to construct preconditioners C_h with the following properties:

1. The condition number $\kappa(C_h^{-1}K_h)$ is as small as possible (close to one), but at least independent of the mesh-size h, i.e., $\kappa(C_h^{-1}K_h) = O(1)$ if h tends to zero.
2. The condition number $\kappa(C_h^{-1}K_h)$ is independent of parameter jumps or other 'bad' parameters.
3. The preconditioner C_h^{-1} is easy to realize, i.e., the arithmetic costs of applying C_h^{-1} are about the same as applying K_h.
4. The application of C_h^{-1} is easy to parallelize.
5. The memory requirement for the preconditioning process is of optimal order, i.e., it is about the same as for K_h.

Example 2.5.3.

1. *The classical Richardson iteration is given by defining $C_h = I_h$, which results in a condition number $\kappa(C_h^{-1}K_h) = O(h^{-2})$, i.e., the condition number asymptotically behaves like $\kappa(K_h)$. By using the symmetric Gauss-Seidel or the Jacobi method as a preconditioner instead we get the same result.*
2. *If we define a preconditioner by $C_h = K_h$ then it results in a direct method with $\kappa(C_h^{-1}K_h) = 1$, but the application of the preconditioner and the memory requirement is not of optimal order.*
3. *Multigrid methods used as a preconditioner are of optimal order (see e.g. [39, 48, 47]).*

Chapter 3

General Approach to AMG Methods

3.1 Motivation for AMG Methods

In Section 2.5 a brief motivation was given for optimal iterative solvers and preconditioners, i.e., the arithmetic costs are of order $O(N_h)$. For further discussion we will use solver and preconditioner in a synonymous way. One advantage of iterative solvers compared to direct ones is the required memory, which is $O(N_h)$. But the convergence rate of a classical iterative solver depends on the condition number of the linear system. Therefore, optimal solvers are of interest, i.e., the convergence rate is independent of the mesh-size parameter h and other 'bad' parameters of the system. The solution of linear systems is an essential aspect of the FE-method. MG methods (geometric and algebraic versions) and Krylov subspace methods together with MG preconditioners fulfill the requirements of an optimal solver. Because of robustness and efficiency, the latter are prior to be used [48, 47].

The key point in MG methods is the efficient interplay of smoothing and coarse grid operators. The error which is not efficiently reduced by the smoothing operator has to be well approximated by a coarser system. The error on the coarse system, which is smooth on the fine level, turns out to contain again high frequency error with respect to the coarser system. That means that a certain spectrum of error frequencies is effectively reduced on each level by the smoother. On the coarsest grid a direct solver is often applied. While the construction is rather simple if a hierarchy of grids is available, the construction of a matrix hierarchy is not an easy task if either the matrix only or information on the finest grid are available. Thus AMG methods are of special interest if geometric multigrid can not be applied. There are at least two situations where AMG is needed:

1. The discretization provides no hierarchy of FE-meshes, which is essential for the geometric MG method. This is the case for many FE-codes, especially commercial ones.

2. The coarsest grid of a geometric multigrid method is too large to be solved efficiently by a direct or classical iterative solver.

The classical AMG approach assumes an SPD system matrix which is additionally an M-matrix (see e.g. [78]). For such matrix classes a matrix hierarchy can be constructed, mimicking the geometric counterpart well. It can easily be shown, that the information of an SPD system matrix is not enough in order to construct an efficient and robust AMG method. Therefore, we assume the knowledge of the underlying PDE, the FE-discretization scheme and additional information on the given FE-mesh. In this way the kernel and other useful properties can be extracted before an AMG method is assembled. By this knowledge the AMG technique is able to mimic a geometric MG method for certain classes of matrices. Subsequently, a general approach to AMG methods is discussed.

3.2 Components of an AMG Method

3.2.1 Auxiliary Matrix ('virtual' FE-mesh)

Let us assume the system matrix K_h stems from an FE-discretization on the FE-mesh $\omega_h = (\omega_h^n, \omega_h^e)$, with ω_h^n, $|\omega_h^n| = M_h$ being the set of nodes and ω_h^e being the set of edges (see Fig. 3.1). An edge is defined as a pair of indices for which the

Figure 3.1: Clipping of an FE-mesh in 2D.

connection of the two points is a geometric edge. For instance, let $i,\, j \in \omega_h^n$ be the indices of the nodes $x_i,\, x_j \in \mathbb{R}^d$ then the edge is given by

$$e_{ij} = (i, j) \in \omega_h^e$$

and the corresponding geometric edge vector can be expressed by

$$\mathbf{a}_{ij} = \mathbf{x}_i - \mathbf{x}_j \in \mathbb{R}^d\,.$$

The first task we are concerned with is the construction of an auxiliary matrix $B_h \in \mathbb{R}^{M_h \times M_h}$ with the following properties:

$$(B_h)_{ij} = \begin{cases} b_{ij} \le 0 & i \ne j \\ -\sum_{j \ne i} b_{ij} \ge 0 & i = j\,. \end{cases}$$

The entries of B_h should be defined in such a way that

1. the distance between two geometric grid points is reflected and
2. the operator $D(x)$ from the variational forms (2.48), (2.49), (2.50), or (2.51) is reflected.

Remark 3.2.1.

1. $B_h \in Z_{M_h}$ by construction.

2. The matrix pattern of B_h can be constructed via several objectives:

(a) B_h reflects the geometric FE-mesh, i.e., $|b_{ij}| \neq 0 \Leftrightarrow (i,j)$ is an edge in the FE-mesh.

(b) B_h has the same pattern as K_h, i.e., $\|k_{ij}\| \neq 0 \Leftrightarrow |b_{ij}| \neq 0$, where $\|\cdot\|$ is an appropriate matrix norm.

Example 3.2.2. *Let $D(x) \in \mathbb{R}^{d\times d}$ $(d = 1,2,3)$, the coefficient matrix of the variational form (2.48), (2.50), or (2.51) be SPD. Further let $\mathbf{a}_{ij} \in \mathbb{R}^d$ be the geometric vector that connects node i with node j for $i \neq j$, i.e., $\mathbf{a}_{ij}$ is a geometric edge in the FE-mesh (see Fig. 3.1). Note, that $\|\mathbf{a}_{ij}\|_D^2$ represents the length of $\mathbf{a}_{ij}$ with respect to the $\|\cdot\|_D$-norm. By defining $b_{ij} = -\frac{1}{\|\mathbf{a}_{ij}\|_D^2}$ for $i \neq j$, B_h results in an appropriate auxiliary matrix (see Example 3.2.3).*

Example 3.2.3. *Let us consider the variational form (2.48) with*

$$D(x) = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} ,$$

$\epsilon > 0$ and a quadratic finite element with side length $h = 1$ (see Fig. 3.2) with bilinear FE-functions.

Figure 3.2: Quadratic finite element.

The element stiffness matrix K_h^r is given by

$$K_h^r = \frac{1}{\epsilon} \cdot \begin{pmatrix} 2+2\cdot\epsilon^2 & 1-2\cdot\epsilon^2 & -2+\epsilon^2 & -1-\epsilon^2 \\ 1-2\cdot\epsilon^2 & 2+2\cdot\epsilon^2 & -1-\epsilon^2 & -2+\epsilon^2 \\ -2+\epsilon^2 & -1-\epsilon^2 & 2+2\cdot\epsilon^2 & 1-2\cdot\epsilon^2 \\ -1-\epsilon^2 & -2+\epsilon^2 & 1-2\cdot\epsilon^2 & 2+2\cdot\epsilon^2 \end{pmatrix}$$

and it can be seen that for $\epsilon \ll 1$ positive off diagonal entries appear which may cause difficulties in classical AMG methods. With an auxiliary matrix, presenting the same non-zero pattern as the system matrix, the problem is better reflected. The element matrix B_h^r (using the method of Example 3.2.2) becomes

$$B_h^r = \frac{1}{\epsilon^2+\epsilon} \cdot \begin{pmatrix} \epsilon^2+\epsilon+1 & -\epsilon^2-\epsilon & -\epsilon-1 & -\epsilon \\ -\epsilon^2-\epsilon & \epsilon^2+\epsilon+1 & -\epsilon & -\epsilon-1 \\ -\epsilon-1 & -\epsilon & \epsilon^2+\epsilon+1 & -\epsilon^2-\epsilon \\ -\epsilon & -\epsilon-1 & -\epsilon^2-\epsilon & \epsilon^2+\epsilon+1 \end{pmatrix} .$$

Now, the FE-mesh and the operator anisotropy are represented in the auxiliary matrix and a standard coarsening is appropriate.

Example 3.2.4. *Let us consider a stiffness matrix arising from linear elasticity (2.49), thus $(K_h)_{ij} = k_{ij} \in \mathbb{R}^{p\times p}$ ($p = 2, 3$). Then $b_{ij} = -\|k_{ij}\|$ for $i \neq j$ is an appropriate auxiliary matrix. In this particular case it is not possible to use the material parameter $D(x)$ of (2.49) in a straight forward way for the auxiliary matrix.*

Example 3.2.5. *The last example is related to the Maxwell equations (2.51) and an edge element discretization. In this case the FE-mesh has to be represented by the auxiliary matrix. Using the method of Example 3.2.2 and $D(x)$ of Example 3.2.3 we get the following element matrix*

$$B_h^r = \frac{1}{\epsilon} \cdot \begin{pmatrix} \epsilon+1 & -\epsilon & -1 & 0 \\ -\epsilon & \epsilon+1 & 0 & -1 \\ -1 & 0 & \epsilon+1 & -\epsilon \\ 0 & -1 & -\epsilon & \epsilon+1 \end{pmatrix} .$$

The entries $(B_h^r)_{14}$, $(B_h^r)_{23}$, $(B_h^r)_{41}$ and $(B_h^r)_{32}$ are zero, i.e., there is no diagonal edge in the virtual FE-mesh related to Fig. 3.2.

Remark 3.2.6.

1. *We can think of B_h as related to elements, such that an element agglomeration is performed (see [45]). This can be realized by defining a distance between two elements, e.g. one over the geometric distance of the barycenter for elements in a neighborhood (see Fig. 3.3).*

Figure 3.3: Element agglomeration technique.

2. The geometric information is required on the finest grid only. By Galerkin's method we get a coarse auxiliary matrix if the transfer operators for the auxiliary matrix are defined properly.

3.2.2 Coarsening Process

The auxiliary matrix may look artificial but a closer look shows that B_h represents a virtual FE-mesh in the following sense:

1. If a point j is 'far' away from i, then the auxiliary matrix has a small negative entry $(B_h)_{ij}$.

2. On the other hand if a point j is 'close' to i, then the auxiliary matrix has a large negative entry $(B_h)_{ij}$.

As it was mentioned above the matrix $B_h \in Z_{M_h}$ and therefore the coarsening process for B_h is straight forward. Let us remind the necessary steps for a matrix $B_h \in Z_{M_h}$. We know that such a matrix represents a virtual FE-mesh $\omega_h = (\omega_h^n, \omega_h^e)$. Such a virtual FE-mesh can be split into two disjoint sets of nodes, i.e.,

$$\omega_h^n = \omega_C^n \cup \omega_F^n, \;\; \omega_C^n \cap \omega_F^n = \emptyset$$

with sets of coarse grid nodes ω_C^n and fine grid nodes ω_F^n. The splitting is usually performed such that

1. No coarse grid nodes are connected directly.

2. The coarse grid nodes should be as many as possible.

Let us introduce the following sets:

$$\begin{aligned} N_h^i &= \{j \in \omega_h^n \,:\, |b_{ij}| \neq 0\,, i \neq j\}\,, \\ S_h^i &= \{j \in N_h^i \,:\, |b_{ij}| > \mathrm{coarse}(B_h, i, j)\,, i \neq j\}\,, \\ S_h^{i,T} &= \{j \in N_h^i \,:\, i \in S^j\}\,, \end{aligned}$$

where N_h^i is the set of neighbors, S_h^i denotes the set of 'strong connections' and $S_h^{i,T}$ is related to the set of nodes with a strong connection to node i, respectively. The cut-off (coarsening) function is taken by, e.g.

$$\mathrm{coarse}(B_h, i, j) = \begin{cases} \theta \cdot \sqrt{|b_{ii}||b_{jj}|} & \text{see [84]} \\ \theta \cdot \max_{l \neq i} |b_{il}| & \text{see [78]} \\ \theta \quad \text{see [57]} \end{cases} \tag{3.1}$$

with an appropriate $\theta \in [0,1]$ (see also Section 3.3). In addition, we define the local sets

$$\omega_C^i = \omega_C^n \cap N_h^i\,, \quad \omega_F^i = \omega_F^n \cap N_h^i$$

and

$$E_h^i = \{(i,j) \in \omega_h^e \,:\, j \in N_h^i\}\,.$$

Further we call $(I_h^i)_{i=1}^{M_H}$ ($|\omega_C^n| = M_H < M_h$) a 'disjoint' splitting for the agglomeration method if

$$I_h^i \cap I_h^j = \emptyset, \qquad \bigcup_{i=1}^{M_H} I_h^i = \omega_h^n \,,$$

is valid. If an appropriate prolongation P_h^B for B_h is defined then a coarse auxiliary matrix is computed by

$$B_H = (P_h^B)^T B_h P_h^B$$

and B_H represents a virtual FE-mesh $\omega_H = (\omega_H^n, \omega_H^e)$, with $\omega_H^n = \omega_C^n$. It can be shown that $B_H \in Z_{M_H}$ if the prolongation operator P_h^B fulfill certain criteria (see [78]). In addition it is assumed that the degrees of freedom on the coarse grid are numbered first. For instance the nodes are reordered like

$$\omega_h^n = (\omega_C^n, \omega_F^n)$$

and as a consequence the system matrix can be written as

$$K_h = \begin{pmatrix} K_C & K_{CF} \\ K_{CF}^T & K_F \end{pmatrix} .$$

Remark 3.2.7.

1. *The auxiliary matrix B_h represents a virtual FE-mesh and therefore it can be related to the degrees of freedom in the original matrix. Consequently it is very important that the prolongation operators P_h^B (see above), P_h^{sys} and P_h^{ker} (see Section 2.4) are consistent, i.e.,*
   (a) *if Lagrange FE-functions are used, then $\|k_{ij}\| \neq 0 \Leftrightarrow |b_{ij}| \neq 0$ for $i \neq j$ on all levels (see Section 4.1 and 4.2).*
   (b) *if Nédélec FE-functions are used then b_{ij}, with $i > j$ (or $i < j$), represents an edge in a virtual FE-mesh (see Section 4.3).*
2. *If $K_h \in Z_{N_h}$ stems from a scalar problem then we can take $B_h \equiv K_h$ which results in a classical AMG method, e.g. [78] (small positive off-diagonal entries of K_h are admissible).*
3. *If K_h stems from a scalar case and we construct a preconditioner B_h such that $\gamma_1 \cdot B_h \leq K_h \leq \gamma_2 \cdot B_h$, $0 < \gamma_1 \leq \gamma_2$, based on the element stiffness matrices, then the technique is equivalent to the element preconditioning technique (see Chapter 5).*

3.2.3 Prolongation Operators

In most AMG approaches the kernel of the underlying operator (see Section 2.3) is disregarded, or it is only implicitly considered. Many AMG approaches preserve the constant functions, which is closely related to the variational form (2.48). But this prerequisite is not sufficient for (2.49), (2.50), or (2.51). It is of great importance for multigrid methods that the characteristics of the discretized operator are the same on all levels, e.g. the kernel has to be preserved. Consequently, AMG-methods have to meet this requirement too. The following theorem provides a necessary condition for the prolongation operator.

Theorem 3.2.8. *Let V_H, V_h, Q_H, Q_h, Λ_H and Λ_h be defined as in Section 2.4. Moreover, $P_h^{sys} : V_H \mapsto V_h$ and $P_h^{ker} : Q_H \mapsto Q_h$ are matrices with full rank. If the equation*

$$P_h^{sys} \Lambda_H \underline{q}_H = \Lambda_h P_h^{ker} \underline{q}_H \quad \forall \underline{q}_H \in Q_H \tag{3.2}$$

holds and additionally

$$\forall \underline{q}_h \in Q_h \quad \exists \underline{q}_H \in Q_H : \ \underline{q}_h = P_h^{ker} \underline{q}_H \tag{3.3}$$

is fulfilled, then

$$V_{0H} = \Lambda_H Q_H$$

is valid.

Proof. The proof splits into two parts:

1. We show that $\Lambda_H Q_H \subseteq V_{0H}$. Let us take an arbitrary but fixed $\underline{q}_H \in Q_H$ and recall the definition of V_{0H} (2.54). Thus we get
$$P_h^{sys} \Lambda_H \underline{q}_H = \Lambda_h P_h^{ker} \underline{q}_H \, ,$$
which is true because of (3.2). Consequently we obtain $\Lambda_H \underline{q}_H \in V_{0H}$.

2. We show that $V_{0H} \subseteq \Lambda_H Q_H$. We take a $\underline{v}_H \in V_{0H}$ and perform the following calculation
$$P_h^{sys} \underline{v}_H = \Lambda_h \underline{q}_h = \Lambda_h P_h^{ker} \underline{q}_H = P_h^{sys} \Lambda_H \underline{q}_H \, .$$
Because P_h^{sys} is assumed to have full rank we conclude that $\underline{v}_H = \Lambda_H q_H$, which is the desired result.

□

Finally we want to mention that for a given splitting $\omega = \omega_C \cup \omega_F$ the optimal prolongation operator is given by the Schur-complement approach, i.e.,

$$K_H = K_C - K_{FC} K_F^{-1} K_{CF} = \tilde{P}_h^T K_h \tilde{P}_h$$

with

$$\tilde{P}_h = (I_H, -K_{CF} K_F^{-1})^T \, .$$

The prolongation operator $\tilde{P}_h$ can hardly be realized in practice since $-K_{CF} K_F^{-1}$ involves the inverse of K_F, which in turn implies a global transport of information. In addition the coarse grid operator K_H becomes dense. The goal of an AMG method is to approximate $\tilde{P}_h$ by some prolongation operator P_h which acts only locally and therewith produces a sparse coarse grid matrix.

3.2.4 Smoother and Coarse Grid Operator

Smoothing Operator: An essential point in MG methods is the smoothing operator $S_h \in \mathbb{R}^{N_h \times N_h}$ which reduces the high frequency error components. Typically, a particular smoother works for certain classes of matrices. It can easily be shown that the smoother is closely related to the underlying system matrix. It is shown in [11] that a point Gauss-Seidel or point Jacobi smoother are appropriate for FE-discretizations with Lagrange FE-functions for scalar elliptic PDEs of second order. Analogously, the block Gauss-Seidel and block Jacobi smoother hold for the block counterpart, e.g. for moderate linear elasticity problems. A patch smoother is discussed in [58] and other smoothers are represented by the incomplete Cholesky, conjugate gradient, and related methods. A general and more rigorous discussion is given in Section 3.4.

Coarse Grid Operator: Subsequently the superscript 'sys' is suppressed. The coarse grid operator K_H is usually constructed by Galerkin's method , i.e.,

$$K_H = P_h^T K_h P_h \,. \tag{3.4}$$

This can be motivated in two ways. We note that the coarse grid correction is given by

$$\underline{u}_h \leftarrow \underline{u}_h + P_h K_H^{-1} P_h^T (\underline{f}_h - K_h \underline{u}_h) \,. \tag{3.5}$$

Let $\underline{e}_h = \underline{u}_h - \underline{u}_h^*$ be the error vector with $\underline{u}_h^*$ being the exact solution of (2.53) then (3.5) can be rewritten as

$$\underline{e}_h \leftarrow T_h \underline{e}_h \quad \text{with} \quad T_h = I_h - P_h K_H^{-1} P_h^T K_h \,, \tag{3.6}$$

where T_h is called the *two grid correction operator* .

1. The task of the correction step is roughly to eliminate the error in $\text{im}(P_h)$. Thus if $\underline{e}_h = P_h \underline{v}_H$ for some $\underline{v}_H \in V_H$ it is desirable that

$$T_h \underline{e}_h = T_h P_h \underline{v}_H = (I_h - P_h K_H^{-1} P_h^T K_h) P_h \underline{v}_H = 0 \,,$$

   or, since P_h has full rank, that (3.4) is valid. Note, that with this setting T_h becomes a projection (see Section 3.4).

2. Equation (3.4) can also be derived by considering the variational principle on

$$\|\underline{e}_h - P_h \underline{v}_H\|_{K_h} \mapsto \min \,.$$

   Again we find that the restriction operator has to be defined by $R_h = P_h^T$ and the coarse grid operator should fulfill (3.4). This is possible if the underlying system matrix K_h is SPD (see Corollary 3.4.1).

The consequences of Galerkin's method are well known. Because we are considering SPD problems and prolongation operators with full rank the coarse grid operator is again SPD.

3.3 Routines and Algorithms

For the implementation of AMG-methods a concept is necessary which is general and flexible with respect to the following items:

1. sequential and parallel versions,
2. coarsening strategies,
3. prolongation operators,
4. smoothing operators.

We will see later (Chapter 4) that for all particular AMG-methods only a small amount of implementational work has to be done, since the general coarsening algorithm is the same for all problems. In addition this approach saves time during the development and keeps the AMG-code as compact as possible. The routine

$$(\{N_h^i\}, \{\, S_h^{i,T}\}) \leftarrow \textsc{GetStrong}(B_h)\,,$$

provides the local sets $\{N_h^i\}$ and $\{S_h^{i,T}\}$ of neighbors and strong connections for all $i \in \omega_h^n$ with an appropriate cut-off function, e.g. (3.1). A coarsening, i.e., a splitting in coarse and fine grid nodes, can be calculated with methods described in [9, 25, 57, 78, 84]. The corresponding general routine is

$$(\omega_C^n, \omega_F^n) \leftarrow \textsc{Coarse}(\{S^{i,T}\}, \omega_h^n)\,,$$

which is depicted in Alg. 3. Therein the function

$$i \leftarrow \textsc{Pick}(\omega_\ell^n \setminus (\omega_C^n \cup \omega_F^n))$$

returns a node i where the number $|S_h^{i,T}| + |S_h^{i,T} \cap \omega_F^n|$ is maximal. The prolongation

Algorithm 3 Coarsening phase $\textsc{Coarse}(\{S_h^{i,T}\}, \omega_h^n)$

```
$\omega_C^n \leftarrow \emptyset, \quad \omega_F^n \leftarrow \emptyset$

while $\omega_C^n \cup \omega_F^n \neq \omega_h^n$ do
  $i \leftarrow \textsc{Pick}(\omega_h^n \setminus (\omega_C^n \cup \omega_F^n))$
  if $|S_h^{i,T}| + |S_h^{i,T} \cap \omega_F^n| = 0$ then
    $\omega_F^n \leftarrow \omega_h^n \setminus \omega_C^n$
  else
    $\omega_C^n \leftarrow \omega_C^n \cup \{i\}$
    $\omega_F^n \leftarrow \omega_F^n \cup (S_h^{i,T} \setminus \omega_C^n)$
  end if
end while
```

operator is computed for the system and the auxiliary problem by the routines called

$$P_h^{sys} \leftarrow \textsc{Weights}(\{S_h^{i,T}\}, K_h, \omega_C^n, \omega_F^n) \quad \text{and} \quad P_h^B \leftarrow \textsc{Weights}(\{S_h^{i,T}\}, B_h, \omega_C^n, \omega_F^n)\,.$$

We refer to [9, 57, 78, 87] and to Chapter 4 for particular transfer operators and their implementations. The coarse grid matrices K_H and B_H are defined by the Galerkin method (3.4). The according routines are denoted by

$$K_H \leftarrow \textsc{Galerkin}(K_h, P_h^{sys}) \quad \text{and} \quad B_H \leftarrow \textsc{Galerkin}(B_h, P_h^B)\,.$$

Finally a pre- and post-smoothing operator is required for a MG-cycle. In spite of the fact that for standard point relaxation methods no preliminary work has to be done, there is some work for the block-variants. The routine is given by

$$S_h \leftarrow \text{SMOOTH}(K_h, \{S_h^{i,T}\})\,.$$

With the functions defined we are able to assemble the *Setup Algorithm* 4. Therein the parameter CoarseGrid is an appropriately chosen coarse-grid size for which a direct solver is applicable. The variable CoarseLevel stores the number of levels generated by the coarsening process until the size of the system is smaller than CoarseGrid. Initially, the setup function Setup(K, B, ω, ℓ) is called for $\ell = 1$ with the system matrix K_h, the auxiliary matrix B_h and the set ω_h^n. For an illustration

Algorithm 4 Sequential Setup Setup(K, B, ω, ℓ)

```
if $|\omega| >$ CoarseGrid then
    $K_\ell \leftarrow K, \quad B_\ell \leftarrow B\ , \quad \omega_\ell \leftarrow \omega$
    $(\{S_\ell^{i,T}\}, \{N_\ell^i\}) \leftarrow$ GetStrong$(B_\ell)$
    $(\omega_C, \omega_F) \leftarrow$ Coarse$(\{S_\ell^{i,T}\}, \omega_\ell)$
    $S_h \leftarrow$ Smooth$(K_\ell, \{S_\ell^{i,T}\})$

    $P_\ell^B \leftarrow$ Weights$(\left\{S_\ell^{i,T}\right\}, B_\ell, \omega_C, \omega_F)$
    $B_{\ell+1} \leftarrow 0$
    $B_{\ell+1} \leftarrow$ Galerkin$(B_\ell, P_\ell^B)$

    $P_\ell^{sys} \leftarrow$ Weights$(\left\{S_\ell^{i,T}\right\}, K_\ell, \omega_C, \omega_F)$
    $K_{\ell+1} \leftarrow 0$
    $K_{\ell+1} \leftarrow$ Galerkin$(K_\ell, P_\ell^{sys})$

    $\omega_{\ell+1} \leftarrow \omega_C$
    Setup$(K_{\ell+1}, B_{\ell+1}, \omega_{\ell+1}, \ell + 1)$
else
    CoarseLevel $\leftarrow \ell$
    $L_\ell L_{\ell T} \leftarrow$Factorization$(K_\ell)$
end if
```

of the two grid method see Fig. 3.4.

Remark 3.3.1. *In contrast to the geometric MG method, the levels in the AMG matrix hierarchy are numbered in the opposite direction. Hence the finest level is related to $\ell = 1$ and the coarsest level is given by $\ell =$ CoarseLevel.*

After a successful setup a MG-cycle can be performed as usual (e.g. see [39]). For instance in Alg. 5 a $V(\nu_F, \nu_B)$-cycle with variable pre- and post-smoothing steps is depicted.

Figure 3.4: Illustration of the general AMG setup.

Algorithm 5 Sequential V-cycle MGSTEP$(K, \underline{u}, \underline{f}, \ell)$

```
  $K_\ell \leftarrow K, \quad \underline{f}_\ell \leftarrow \underline{f}, \quad \underline{u}_\ell \leftarrow \underline{u}$

  **if** $\ell ==$ COARSELEVEL **then**
    $\underline{u}_\ell \leftarrow$ COARSEGRIDSOLVER $(L_\ell L_\ell^T, \underline{f}_\ell)$
    RETURN
  **else**
    $\underline{d}_\ell \leftarrow 0$, $\underline{w}_{\ell+1} \leftarrow 0$
    $\underline{u}_\ell \leftarrow S_\ell^{\nu_F}(\underline{u}_\ell, \underline{f}_\ell)$
    $\underline{d}_\ell \leftarrow \underline{f}_\ell - K_\ell \underline{u}_\ell$
    $\underline{d}_{\ell+1} \leftarrow (P_\ell^{sys})^T \underline{d}_\ell$
    MGSTEP$(K_{\ell+1}, \underline{w}_{\ell+1}, \underline{d}_{\ell+1}, \ell + 1)$
    $\underline{w}_\ell \leftarrow P_\ell^{sys} \underline{w}_{\ell+1}$
    $\underline{u}_\ell \leftarrow \underline{u}_\ell + \underline{w}_\ell$
    $\underline{u}_\ell \leftarrow S_\ell^{\nu_B}(\underline{u}_\ell, \underline{f}_\ell)$
  **end if**
```

3.4 Convergence Theory and Limitations

In the following section we give a brief overview on the convergence theory of AMG methods and their limitation. Since our general approach is mainly concerned with the prolongation operator and the smoother the subsequent theory proposed by J. Ruge and K. Stüben in [78] can be applied. Thus the main results are recalled. Until now it is not clear whether there exists a satisfying proof for the V-cycle or not. Before we discuss this in detail we present the preliminaries for the convergence theory. Let us consider the following problem setting of an AMG method.

1. The system matrix K_h stems from an FE-, or FD-discretization of an elliptic self-adjoint linear PDE, i.e., K_h is SPD.
2. The transfer operators P_h are assumed to have full rank.

In literature many different AMG approaches are discussed for this task e.g. [9, 14, 57, 78, 84, 87]. All these methods can be seen in one of the 3 items. In addition we have to mention that these AMG techniques are well suited for $H^1(\Omega)$-elliptic problems and some of them also for $(H^1(\Omega))^p$-elliptic problems. Besides, R. Beck proposed in [6] a preconditioner for $H(\text{curl}, \Omega)$-elliptic problems.

1. The AMG techniques in [78, 14, 87] fix a point relaxation method (e.g. Gauss-Seidel smoother) and construct the best possible prolongation operator in order to reduce both error components effectively. In this case the so called *algebraically smooth error* is introduced, which is the error that can not be efficiently reduced by the smoother.
2. A completely different approach to AMG is to use a simple prolongation and improve the convergence rate by a patch smoother [58] or by scaling the residual [9].
3. A method related to both approaches is to use a standard Gauss-Seidel smoother and to construct a tentative prolongation operator. In a second step the prolongation operator is improved by some smoothing steps [84].

Our attempt is to use an appropriate smoother and prolongation operator for an AMG method such that the *convergence rate* is as good as possible and the application of the AMG method is still cheap. It gets quite clear that a universal AMG method can not be constructed, but necessary conditions for classes of problems can be analyzed and also realized in algorithms (see Chapter 4).

Before we propose special AMG methods for different classes of matrices we prepare some general theory. Most of the theory can be found in [67] by S. McCormick and in [78, 82, 83] by J.W. Ruge and K. Stüben. For the rest of this thesis we assume to work with FE-discretization and we recall the definition of the coarse grid correction operator

$$T_h = I_h - P_h K_H^{-1} P_h^T K_h \,.$$

Further, we use the inner products and corresponding norms

$$(\underline{u}_h, \underline{v}_h)_1 = (K_h \underline{u}_h, \underline{v}_h) \quad \text{and} \quad (\underline{u}_h, \underline{v}_h)_2 = (K_h \underline{u}_h, K_h \underline{v}_h)\,,$$

where the diagonal entries of the system matrix K_h are assumed to be equal to one throughout this section. In addition the superscript 'sys' is suppressed.

Corollary 3.4.1. *Let K_h be SPD and a full rank prolongation P_h be given. Then the coarse grid correction operator T_h is an orthogonal projector with respect to the energy inner product, which in particular results in:*

1. $(T_h\underline{v}_h, P_h\underline{u}_H)_1 = 0$ *for all* $\underline{v}_h \in V_h$ *and for all* $\underline{u}_H \in V_H$.
2. *For all* $\underline{u}_h \in \operatorname{im}(T_h)$ *and* $\underline{v}_h \in \operatorname{im}(P_h)$ *we have* $\|\underline{u}_h + \underline{v}_h\|_1^2 = \|\underline{u}_h\|_1^2 + \|\underline{v}_h\|_1^2$.
3. *The energy operator norm is equal one, i.e.,* $\|T_h\|_1 = 1$.
4. *For all* $\underline{e}_h$ *we have* $\|T_h\underline{e}_h\|_1 = \min_{\underline{e}_H} \|\underline{e}_h - P_h\underline{e}_H\|_1$.

Proof. The proof can be found in [82]. □

Item 4. of the corollary expresses the variational principle (see Subsection 3.2.4), i.e., Galerkin based coarse grid correction minimizes the energy norm of the error with respect to all variations in $\operatorname{im}(P_h)$. Further we pinpoint that

$$V_h = \operatorname{im}(P_h) \oplus \ker(P_h^T)\,,$$

since P_h has full rank and V_h has finite dimension. In addition we notice the useful relations

$$\operatorname{im}(P_h) \equiv (\ker P_h^T)^\perp \quad \text{and} \quad \ker(P_h^T) \equiv \ker((I_h - T_h)K_h^{-1})\,,$$

and as a consequence

1. the error components in $\ker(P_h^T)$ have to be efficiently reduced by the smoother and
2. the error components in $\operatorname{im}(P_h)$ have to be treated by the coarse grid correction.

This already implies two important consequences.

1. The coarse grid correction terms are in $\operatorname{im}(P_h)$ and such error components are orthogonal to $\ker(P_h^T)$. Consequently, the coarse grid correction term does not influence relaxation.
2. On the other hand high frequency error components are amplified by the coarse grid correction. This can be seen as follows: Let $\underline{e}_h^n$ be the error in the n^{th} iteration and in addition assume that $\underline{e}_h^n = \underline{e}_R^n + \underline{e}_S^n$, with $\underline{e}_R^n \in \ker(P_h^T)$ and $\underline{e}_S^n \in \operatorname{im}(P_h)$. The new iterate $\underline{e}_h^{n+1}$ is given by

$$\underline{e}_h^{n+1} = (I_h - P_h K_H^{-1} P_h^T K_h)(\underline{e}_R^n + \underline{e}_S^n)\,.$$

Because of $\underline{e}_S^n = P_h\underline{v}_H$ for some $\underline{v}_H \in V_H$ and the Galerkin method, we are faced with

$$\underline{e}_h^{n+1} = (I_h - P_h K_H^{-1} P_h^T K_h)\underline{e}_R^n\,,$$

which means that if the high frequency component $\underline{e}_R^n$ is made sufficiently small by the smoother, then these components are magnified by a factor bounded by

$$\max \|P_h K_H^{-1} P_h^T K_h \underline{z}_h\| \quad \forall \underline{z}_h \in \ker(P_h^T),\ \|\underline{z}_h\| = 1\,.$$

Up to now, we were not considering the convergence and the convergence rate of an AMG method. To get some insight we follow [78] and prepare the main results. At the end we give some comments and useful consequences. First of all we give a general result on a quantitative estimate of the V-cycle convergence rate.

Theorem 3.4.2. *Let K_h be SPD and assume the prolongation operator P_h to have full rank. Further suppose for all $\underline{e}_h \in V_h$*

$$\|S_h\underline{e}_h\|_1^2 \leq \|\underline{e}_h\|_1^2 - \delta_1 \cdot \|T_h\underline{e}_h\|_1^2 \tag{3.7}$$
$$\|S_h\underline{e}_h\|_1^2 \leq \|\underline{e}_h\|_1^2 - \delta_2 \cdot \|T_hS_h\underline{e}_h\|_1^2 \tag{3.8}$$

hold for some δ_1, $\delta_2 > 0$ independent of $\underline{e}_h$ and h (where h denotes the discretization parameter of the considered level). Then the following is valid.

1. *If at least one smoothing step is performed after each coarse grid correction step, then $\delta_1 < 1$ and the V-cycle convergence factor is bounded from above by $\sqrt{(1-\delta_1)}$.*

2. *If at least one smoothing step is performed before each coarse grid correction step, then the V-cycle convergence factor is bounded from above by $\sqrt{1/(1+\delta_2)}$.*

3. *If at least one smoothing step is performed before and after each coarse grid correction step, then $\delta_1 < 1$ and the V-cycle convergence factor is bounded from above by $\sqrt{(1-\delta_1)/(1+\delta_2)}$.*

Proof. The proof can be found in [78]. □

Theorem 3.4.3. *Let $\delta_1 = \frac{\alpha_1}{\beta_1}$ and $\delta_2 = \frac{\alpha_2}{\beta_2}$. Then the conditions*

$$\|S_h\underline{e}_h\|_1^2 \leq \|\underline{e}_h\|_1^2 - \alpha_1 \cdot \|\underline{e}_h\|_2^2 \tag{3.9}$$
$$\|T_h\underline{e}_h\|_1^2 \leq \beta_1 \cdot \|\underline{e}_h\|_2^2 \tag{3.10}$$

imply (3.7). In a similar way the conditions

$$\|S_h\underline{e}_h\|_1^2 \leq \|\underline{e}_h\|_1^2 - \alpha_2 \cdot \|S_h\underline{e}_h\|_2^2 \tag{3.11}$$
$$\|T_h\underline{e}_h\|_1^2 \leq \beta_2 \cdot \|\underline{e}_h\|_2^2 \tag{3.12}$$

imply (3.8).

Proof. The proof can be found in [78]. □

Remark 3.4.4.

1. *The two separated inequalities for S_h and T_h are called smoothing property and approximation property .*

2. *We say that the relaxation operator S_h satisfies the smoothing property with respect to a matrix K_h being SPD if*

$$\|S_h\underline{e}_h\|_1^2 \leq \|\underline{e}_h\|_1^2 - \sigma \cdot \|\underline{e}_h\|_2^2$$

holds for some $\sigma > 0$ independent of $\underline{e}_h$. This implies that S_h is efficient in reducing the error $\underline{e}_h$ as long as $\|\underline{e}_h\|_2$ is relatively large compared to $\|\underline{e}_h\|_1$.

3. *The approximation property can hardly be used because of the unhandy characterization of a prolongation operator. Theorem 3.4.5 gives a more practical characterization.*

Theorem 3.4.5. *Let $K_h \in \mathbb{R}^{N_h \times N_h}$ be SPD and S_h satisfies the smoothing property. Further suppose the prolongation operator P_h has full rank and for every $\underline{e}_h \in V_h$*

$$\min_{\underline{e}_H} \|\underline{e}_h - P_h \underline{e}_H\|_0^2 \leq \beta \cdot \|\underline{e}_h\|_1^2 \tag{3.13}$$

holds for some $\beta > 0$ independent of $\underline{e}_h$. Then $\beta > \alpha_1$ and the two level convergence factor satisfies

$$\|S_h T_h\|_1 \leq \sqrt{1 - \frac{\alpha_1}{\beta}}\,.$$

Proof. The proof can be found in [78]. □

Unfortunately, we can not prove a level independent V-cycle convergence with Theorem 3.4.5. Clearly this dependency could be overcome by a W-, or F-cycle but these cycles suffer from the high costs of an application. The problem we are faced with is the lack of the approximation property. A better prolongation operator could be constructed by the following lemma. Unfortunately in this case the prolongation operator can hardly be realized with pure algebraic information.

Lemma 3.4.6. *If*

$$\|T_h \underline{e}_h\|_1^2 \leq \beta \cdot \|\underline{e}_h\|_2^2$$

holds, then we also have

$$\min_{\underline{e}_H} \|\underline{e}_h - P_h \underline{e}_H\|_0^2 \leq \beta^2 \cdot \|\underline{e}_h\|_2^2\,.$$

Proof. The proof can be found in [78]. □

Equation (3.13) is a necessary condition for the approximation property. This condition implies a useful property for the prolongation operator to be constructed. Suppose K_h has a non-trivial kernel V_{0h}. Then the whole theory is valid if the iteration is performed orthogonal to the kernel, i.e., in the space $V_{0h}^{\perp}$. Consequently, equation (3.13) must be exact on the kernel, i.e., the prolongation operator has to preserve the nullspace. Let us suppose a prolongation operator of the form

$$P_h = P_h^{opt} - P_h^{\epsilon}\,.$$

The first part of P_h is an optimal prolongation operator $P_h^{opt} : V_{0H}^{\perp} \mapsto V_{0h}^{\perp}$, which is exact on the kernel. The second part is some perturbation of the null operator, i.e., $P_h^{\epsilon} : V_H \mapsto V_h$. In general P_h does not preserve the kernel of K_h. Consequently we get

$$\min_{\underline{e}_H} \|\underline{e}_h - P_h \underline{e}_H\|_0^2 = \min_{\underline{e}_H} \|\underline{e}_h - (P_h^{opt} - P_h^{\epsilon})\underline{e}_H\|_0^2 \leq \beta \cdot \|\underline{e}_h\|_1^2 = 0\,,$$

and thus $P_h^{\epsilon} \equiv 0$ must hold. On the other hand the coarse grid operator K_H has to have the same properties as the fine grid operator K_h. For instance the kernel has to be preserved. This is already depicted in Theorem 3.2.8.

Chapter 4

Special Algebraic Multigrid Methods

4.1 Symmetric $H^1(\Omega)$-based Equations

In this subsection we discuss a problem setting arising from an FE-discretization of a second order elliptic self-adjoint scalar PDE. An FE-discretization with linear Lagrange FE-functions of the variational form (2.48) is taken as a representative. Actually the main results carry over to more general $H^1(\Omega)$-elliptic equations (e.g. [82]).

Construction of 'virtual' FE-meshes: First of all we mention that the classical AMG methods fit perfectly into our scheme by defining the auxiliary matrix

$$B_h \equiv K_h \, .$$

In the case of $K_h \in Z_{N_h}$, i.e., K_h is an M-matrix, the classical AMG methods result in a robust and efficient solver. However, it can easily be seen that the drawback of the M-matrix property might result in a 'bad' splitting (coarsening).

Example 4.1.1. *Let us consider (2.48) with linear Lagrange FE-functions on a rectangle (see Fig. 4.1). Then node* 1 *is strongly connected to nodes* 3 *and* 4*. From*

Figure 4.1: (a) Standard coarsening using the system matrix. (b) Standard coarsening using the auxiliary matrix.

geometric MG we know that this is 'wrong' in the sense that node 1 *should only be strongly connected to node* 3*.*

To overcome this problem we suggest to construct an auxiliary matrix B_h as depicted in Example 3.2.2. In this way the matrix pattern of K_h and B_h are the same, i.e.,

$$|(K_h)_{ij}| \neq 0 \Leftrightarrow |(B_h)_{ij}| \neq 0 \,.$$

Since the i^{th}-row of K_h has only entries which stem from neighboring FE-functions, the construction for B_h is valid. By considering the same coarsening strategy for B_h as in the example above, we obtain that only 3 is strongly connected to 1, which is the correct coarsening (see Fig. 4.1). Note, in order to construct such an auxiliary matrix we need access to the coordinates on the finest grid, which should be available in all FE-codes. Another possibility to construct an auxiliary matrix B_h is the *element preconditioning method* discussed in Chapter 5.

Construction of coarse FE-spaces: In Subsection 3.2.3 we discussed the best possible prolongation operator, which is not realizable. In spite of this fact we can approximate it by local calculations. The simplest prolongation operator is given by the piecewise constant interpolation (e.g. see [9]). A better one is discussed in [57], which is given by

$$(P_h^{sys})_{ij} = \begin{cases} 1 \quad i = j \in \omega_C^n \\ \frac{1}{|S_h^i \cap \omega_C^n|} \qquad i \in \omega_F^n,\, j \in S_h^{i,T} \cap \omega_C^n \\ 0 \quad \text{else}\,. \end{cases} \tag{4.1}$$

For instance, Ruge and Stüben proposed in [78] the discrete harmonic extension for a prolongation operator. This prolongation operator is the best of those proposed for many applications and it reads as

$$(P_h^{sys})_{ij} = \begin{cases} 1 \quad i = j \in \omega_C^n \\ -\frac{k_{ij}+c_{ij}}{k_{ii}+c_{ii}} \quad i \in \omega_F^n,\, j \in \omega_C^i \\ 0 \quad \text{else} \end{cases} \tag{4.2}$$

with

$$c_{ij} = \sum_{p \in \omega_F^i} \frac{k_{ip} \cdot k_{pj}}{\sum_{q \in \omega_C^i} k_{pq} + k_{pi}} \,.$$

As it was mentioned in Subsection 3.2.3 we have to show that the proposed prolongation operators verifies the preliminaries of Theorem 3.2.8.

Corollary 4.1.2. *Let us consider the scalar variational problem (2.48) and further assume the prolongation operators P_h^{sys}, P_h^{ker} and P_h^B to be equal, i.e., $P_h^{sys} \equiv P_h^{ker} \equiv P_h^B = P_h \in \mathbb{R}^{N_h \times N_H}$, with full rank and P_h satisfies*

$$\sum_{j=1}^{N_H} (P_h)_{ij} = 1 \quad \forall i = 1, \ldots, N_h \,. \tag{4.3}$$

Then the preliminaries of Theorem 3.2.8 are valid. The piecewise constant prolongation operator, and the operators given in (4.1) and (4.2) fulfill the requirements above.

Proof. Since $\Lambda_h = \mathrm{Id}_h$, $\Lambda_H = \mathrm{Id}_H$ and the prolongation operators P_h^{sys}, P_h^{ker} and P_h^B are equal with full rank, the first condition of Theorem 3.2.8 is valid. The second condition (3.3) is true because of

$$Q_h = \mathrm{span}\{\mathbf{1}_h\} \quad \text{and} \quad Q_H = \mathrm{span}\{\mathbf{1}_H\}$$

with $\mathbf{1}_h = (1, \dots, 1)^T \in \mathbb{R}^{N_h}$, and (4.3). $\square$

Smoothing operator: For the smoothing operator we usually use a point Gauss-Seidel method, e.g. [11], or a patch Gauss-Seidel method, e.g. [58]. The latter one is taken for (strong) anisotropic problems.

4.2 Symmetric $(H^1(\Omega))^p$-based Equations

$(H^1(\Omega))^p$-elliptic equations are more delicate because of the block structure of the PDEs. Such problems arise from an FE-discretization of the bilinear forms (2.49) or (2.50), which are taken as representatives for this subsection. The main problem in the construction of an AMG method for such PDEs is the kernel of the underlying operator. For instance in the linear elasticity case (2.49) we are concerned with the rigid body modes, that form a six dimensional subspace. Even more difficult is the linear magnetic case (2.50) where we are faced with a kernel containing the harmonic functions, i.e., the kernel is an infinite dimensional subspace.

Construction of 'virtual' FE-meshes: The definition of the auxiliary matrix B_h plays an important role for this problem class. The classical approach uses

$$(B_h)_{ij} = -\|k_{ij}\| \quad \text{for} \quad i \neq j$$

with an appropriate matrix norm $\|\cdot\|$, e.g. $\|\cdot\| = \|\cdot\|_\infty$. But with this setting we get less information on the underlying operator and FE-mesh and this might cause problems similar to the scalar case. Instead we can define an auxiliary matrix in the same way as for the scalar case. Now the degrees of freedom per node of the system matrix have to be related to an entry in the auxiliary matrix, which in turn implies that the matrix pattern of K_h and B_h has to be equal, i.e.,

$$\|(K_h)_{ij}\| \neq 0 \Leftrightarrow |(B_h)_{ij}| \neq 0\,.$$

Construction of coarse FE-spaces: The prolongation operators of the system matrix are defined in the same way as for the scalar problem. Thus the simplest one is given by interpolating all components piecewise constant. An improved version reads as (analoguely to prolongation (4.1))

$$(P_h^{sys})_{ij} = \begin{cases} I_p \quad i = j \in \omega_C^n \\ \frac{1}{|S_h^{i,T} \cap \omega_C^n|} \cdot I_p \qquad i \in \omega_F^n,\ j \in S_h^{i,T} \cap \omega_C^n \\ 0 \quad \text{else}\,. \end{cases} \tag{4.4}$$

The AMG method shows a better convergence behavior with the subsequent discrete harmonic extension (in analog to prolongation (4.2)) compared to (4.4), i.e.,

$$(P_h^{sys})_{ij} = \begin{cases} I_p & i = j \in \omega_C^n \\ -k_{ii}^{-1}\big(k_{ij} + c_{ij}\big) & i \in \omega_F^n,\ j \in \omega_C^i \\ 0 & \text{else} \end{cases} \tag{4.5}$$

with

$$c_{ij} = \sum_{p\in\omega_F^i} \Big(\sum_{q\in\omega_C^i} k_{pq}\Big)^{-1} k_{ip} k_{pj} \,.$$

Note, the entries of the prolongation operators are matrix valued, e.g. $(P_h^{sys})_{ij} \in \mathbb{R}^{p\times p}$, as the entries of the system matrix K_h. We assume the prolongation operators P_h^{sys} and P_h^{ker} to be equal, i.e., $P_h^{sys} \equiv P_h^{ker} = P_h \in \mathbb{R}^{N_h\times N_H}$ and the suggested prolongation operator for the auxiliary problem $P_h^B \in \mathbb{R}^{M_h\times M_H}$. In order to get the same matrix pattern for the system matrix and the auxiliary matrix we choose a piecewise constant prolongation operator for both, or the combination (4.4) and (4.1), or (4.5) and (4.2) for the prolongation operators P_h^{sys} and P_h^B, respectively. The preliminaries for Theorem 3.2.8 can neither be shown for linear elasticity nor for linear magnetics.

Smoothing operator: Similar to the scalar equations we use a block Gauss-Seidel method as smoothing operator, e.g. [11], or a patch-block Gauss-Seidel method, e.g. [58]. Again, the latter one is taken for anisotropic problems.

4.3 Symmetric $H(\mathrm{curl}, \Omega)$-based Equations

The third class originates from an FE-discretization with Nédélec FE-functions of the variational form (2.51). As it was depicted in Section 2.3 the kernel is given by the gradient fields, i.e., it is infinite dimensional.

Construction of 'virtual' FE-meshes: It was motivated by R. Hiptmair in [42] for geometric MG methods that for such class of problems a refinement of the FE-mesh can be performed on the nodes of an FE-mesh as it is usually done for Lagrange FE-functions. We use this fact and base our coarsening on an auxiliary matrix B_h which is constructed for instance by the method discussed in Example 3.2.2 for general FE-meshes or from an FE-discretization of (2.48) on a tetrahedra FE-mesh. Let us recall that an FE-mesh is represented by

$$\omega_h = (\omega_h^n, \omega_h^e)\,,$$

i.e., the set of nodes ω_h^n and the set of edges ω_h^e (see Subsection 3.2.1) and a coarsening is performed as usual (see Subsection 3.2.2). The coarse grid is defined by identifying each coarse grid node $j \in \omega_C^n$ with an index $k \in \omega_H^n$. This is expressed by the index map ind(.) as

$$\omega_H^n = \mathrm{ind}(\omega_C^n)\,.$$

A 'useful' set of coarse grid edges ω_H^e can be constructed if we invest in a special prolongation operator $P_h^B \equiv P_h^{ker} : Q_H \mapsto Q_h$ for the auxiliary matrix B_h. The prolongation operator P_h^B is constructed such that each fine grid variable prolongates exactly from one coarse grid variable. We extend the index map ind : $\omega_C^n \mapsto \omega_H^n$ defined above onto the whole fine set ω_h^n by assigning the coarse grid index of the representative of the cluster

$$\text{ind} : \omega_h^n \to \omega_H^n \,.$$

A consequence is that $\text{ind}(i) = \text{ind}(j)$ iff $i, j \in \omega_h^n$ prolongate from the same coarse grid variable. We define an agglomerate (cluster) I_h^i of a grid point $i \in \omega_h^n$ (see Fig. 4.2 and Subsection 3.2.2) by

$$I_h^i = \{j \in \omega_h^n \,|\, \text{ind}(j) = \text{ind}(i)\} \subset N_h^i$$

and hence the set of coarse grid nodes can be written as

$$\omega_H^n = \{\text{ind}(i) \,|\, i \in \omega_h^n\} \,.$$

The prolongation operator P_h^B has only 0 and 1 entries by construction, i.e.,

$$(P_h^B)_{ij} = p_{ij}^n = \begin{cases} 1 & i \in \omega_h^n,\ j = \text{ind}(i) \\ 0 & \text{otherwise}\,. \end{cases} \tag{4.6}$$

The coarse grid matrix B_H is calculated by Galerkin's method (3.4), which is equivalent to the formula

$$(B_H)_{kl} = \sum_{i \in I_h^{\tilde{k}}} \sum_{j \in I_h^{\tilde{l}}} p_{ik}^n \cdot (B_h)_{ij} \cdot p_{il}^n \tag{4.7}$$

with $k = \text{ind}(\tilde{k})$, $l = \text{ind}(\tilde{l})$, and $\tilde{k}, \tilde{l} \in \omega_H^n$. B_H has useful properties, which originate in the prolongation operator defined in (4.6). This is the content of the next lemma.

Lemma 4.3.1. *Let $\tilde{k}, \tilde{l} \in \omega_C^n$, $\tilde{k} \neq \tilde{l}$ and $k = \text{ind}(\tilde{k}) \in \omega_H^n$, $l = \text{ind}(\tilde{l}) \in \omega_H^n$. Furthermore, B_h stems from an FE-discretization of (2.48) with linear nodal FE-functions and P_h^B is defined by (4.6). $B_H = (P_h^B)^T B_h P_h^B$. If for all $i \in I_h^{\tilde{k}}$ and for all $j \in I_h^{\tilde{l}}$*

$$(B_h)_{ij} = 0$$

then

$$(B_H)_{kl} = 0 \,.$$

Proof. The proof follows immediately by using (4.7). □

Remark 4.3.2.

1. *The essence of Lemma 4.3.1 is, that a coarse grid edge only exists if there is at least one fine edge connecting the agglomerates I_h^i and I_h^j ($i \neq j$), i.e.,*

$$\exists r \in I_h^i, \exists s \in I_h^j \text{ such that } (r, s) \in \omega_h^e$$

(see Fig. 4.2).

Figure 4.2: 'Virtual' FE-mesh with a feasible agglomeration.

2. *A decrease of the number of edges in the coarsening process is not proofed in general, but a decrease is heuristically given, if the average number of nonzero entries of B_h does not grow too fast.*

Construction of coarse FE-spaces: The construction of the prolongation operator $P_h^{sys} : V_H \mapsto V_h$, is delicate because of the kernel of the curl-operator. P_h^{sys} is defined for $i = (i_1, i_2) \in \omega_h^e, j = (j_1, j_2) \in \omega_H^e$ as

$$(P_h^{sys})_{ij} = \begin{cases} 1 & \text{if } j = (\text{ind}(i_1), \text{ind}(i_2)), \\ -1 & \text{if } j = (\text{ind}(i_2), \text{ind}(i_1)), \\ 0 & \text{otherwise}, \end{cases} \tag{4.8}$$

by assuming a positive orientation of an edge $j = (j_1, j_2)$ from j_1 to j_2 if $j_1 < j_2$ holds. The constructed prolongation operator P_h^{sys} has full rank, because the coarse grid edges prolongate to N_H distinct fine grid edges by construction.

Next we note that the operator $\text{grad}_h : Q_h \mapsto V_h$ defined in (2.59) has the representation (with $i = (i_1, i_2) \in \omega_h^e$ and $\underline{q}_h \in Q_h$)

$$(\text{grad}_h \, \underline{q}_h)_i = q_{h,i_2} - q_{h,i_1}. \tag{4.9}$$

This can be seen by evaluating (2.59) for a $\underline{q}_h \in Q_h$ and using the fact that $\mathbb{Q}_h$ is a piecewise linear FE-space and the degree of freedom of the edge element discretization is the path integral on the edge (i_1, i_2). In analogy, we define $\text{grad}_H : Q_H \mapsto V_H$ on the coarse level. Since we use a Galerkin approach, the coarse grid kernel V_{0H} defined in (2.54) is a subspace of the fine grid kernel V_{0h} given by (2.58). The crux is that P_h^{sys} prolongates discrete gradients of the coarse space to discrete gradients of the fine space, which is generally shown in Theorem 3.2.8. The following two lemmas provide the preliminaries for that theorem.

Lemma 4.3.3. *For $\underline{q}_H \in Q_H$ there holds*

$$P_h^{sys} \operatorname{grad}_H \underline{q}_H = \operatorname{grad}_h P_h^B \underline{q}_H . \tag{4.10}$$

This means, the commuting diagram

$$\begin{tikzcd} Q_H \arrow[r, "\operatorname{grad}_H"] \arrow[d, "P_h^B"] & V_H \arrow[d, "P_h^{sys}"] \\ Q_h \arrow[r, "\operatorname{grad}_h"] & V_h \end{tikzcd}$$

is valid.

Proof. We consider the edge $i = (i_1, i_2) \in \omega_h^e$. We have to distinguish two cases. First, let us assume the edge is inside one agglomerate, i.e., $\operatorname{ind}(i_1) = \operatorname{ind}(i_2)$. Then both sides of (4.10) vanish. The left hand side vanishes by definition of the prolongation operator P_h^{sys}, the right hand side vanishes since $(P_h^B \underline{q}_H)_{i_1} = (P_h^B \underline{q}_H)_{i_2}$.

Now, we assume that $\operatorname{ind}(i_1) \neq \operatorname{ind}(i_2)$. Thus, there exists a coarse grid edge $j = (j_1, j_2)$ such that either $j_1 = \operatorname{ind}(i_1), j_2 = \operatorname{ind}(i_2)$ or $j_1 = \operatorname{ind}(i_2), j_2 = \operatorname{ind}(i_1)$. In both cases there holds $(\operatorname{grad}_H \underline{q}_H)_j = \pm(q_{H,j_1} - q_{H,j_2})$. The sign in the prolongation compensates, such that $(P_h^{sys} \operatorname{grad}_H \underline{q}_H)_i = q_{H,\operatorname{ind}(i_1)} - q_{H,\operatorname{ind}(i_2)}$. Evaluating $(\operatorname{grad}_h P_h^B \underline{q}_H)_i$ gives the same result. □

Lemma 4.3.4. *For every $\underline{q}_h \in Q_h$ there exists a $\underline{q}_H \in Q_H$ such that*

$$\underline{q}_h = P_h^B \underline{q}_H .$$

Proof. Let $\underline{v}_H \in V_{0H}$. Since the kernels are nested we get that

$$\underline{v}_h = P_h^B \underline{v}_H \in V_{0h}$$

and thus there exists a $\underline{q}_h \in Q_h$ such that

$$\underline{v}_h = \operatorname{grad}_h \underline{q}_h .$$

By the definition of P_h^{sys} the values inside an agglomerate vanish, i.e., $(\underline{v}_h)_i = 0$ for $i = (i_1, i_2)$ and $\operatorname{ind}(i_1) = \operatorname{ind}(i_2)$. Because

$$(\underline{v}_h)_i = q_{h,i_1} - q_{h,i_2}$$

the potential $\underline{q}_h$ is constant inside an agglomerate. Thus there exists a $\underline{q}_H \in Q_H$ such that $\underline{q}_h = P_h^B \underline{q}_H$, that concludes the proof. □

Corollary 4.3.5. *Under the preliminaries of Theorem 3.2.8*

$$V_{0H} = \operatorname{grad}_H Q_H ,$$

holds.

Proof. The proof follows immediately by combining Lemma 4.3.3 and 4.3.4. □

Figure 4.3: Detail view of a virtual FE-mesh.

Smoothing operator: To complete the ingredients for an AMG method for edge element FE-discretizations, we need an appropriate smoother. We consider two different types of smoothers for K_h. The first one was suggested by D. Arnold, R. Falk and R. Winther in [3]. This is a block Gauss-Seidel smoother where all edges are smoothed simultaneously which belong to E_h^i for all $i \in \omega_h^n$ (see Fig. 4.3).

Another kind of smoother was suggested by R. Hiptmair in [42]. A mathematical equivalent formulation is outlined in Alg. 6. Therein the vector $\underline{g}_h^{e,i} \in V_h$ is defined

Algorithm 6 Hybrid smoother of Hiptmair: Smooth(K_h,$\underline{f}_h$,$\underline{u}_h$)

```
$\underline{u}_h \leftarrow$ GaussSeidel($K_h$, $\underline{f}_h$, $\underline{u}_h$)

for all $i \in \omega_h^n$ do
    $\underline{u}_h \leftarrow \underline{u}_h + \frac{((\underline{f}_h - K_h \underline{u}_h), \underline{g}_h^{e,i})}{(K_h \underline{g}_h^{e,i}, \underline{g}_h^{e,i})} \cdot \underline{g}_h^{e,i}$
end for
```

by

$$\underline{g}_h^{e,i} = \operatorname{grad}_h \underline{g}_h^{n,i} = \begin{cases} 1 & j < i \qquad (i,j) \in E_h^i \\ -1 & j > i \qquad (i,j) \in E_h^i \\ 0 & \text{otherwise} \end{cases}$$

with a vector $\underline{g}_h^{n,i} \in Q_h$, $(\underline{g}_h^{n,i})_j = \delta_{ij}$.

Chapter 5

Element Preconditioning Technique

5.1 Motivation

It is well known that the AMG method of J. W. Ruge and K. Stüben [78] is robust and efficient for M-matrices and 'small perturbations' of that class (i.e., small positive off-diagonal entries are admissible). For many applications in science and engineering the M-matrix property is not given, e.g. in anisotropic problems or by using higher order FE-functions. To overcome this remedy we can use the *element preconditioning method*, which can be embedded in the general approach of Chapter 3. This can be seen in the following way. Let us assume that K_h is SPD and that K_h has no M-matrix property. Further, let B_h be an M-matrix that is spectrally equivalent to K_h. B_h can be seen as an auxiliary problem which represents the grid but could also be taken as a preconditioner for K_h. Furthermore, the classical AMG applied to B_h (which is robust and efficient) turns out to be a good preconditioner for K_h if B_h is close to K_h in the spectral sense. Consequently, the aim is to construct an SPD M-matrix B_h, that has a small condition number with respect to K_h, i.e., $\kappa(B_h^{-1}K_h)$ is small. It might be clear that the construction of a spectrally equivalent M-matrix B_h out of K_h is numerically too expensive. The element preconditioning technique is able to construct B_h based on the element matrices of K_h, which are usually available in FE-codes before the assembling process.

Subsequently, we present the theoretical background and propose an algorithm which can be used in any FE-code. The original works are given in [71, 73, 38] and applications can be found in [52, 53, 56].

5.2 Basic Concept

To cope with the kernel in the generalized eigenvalue problem and with the classification of element matrices introduced later, we need some well-known results from linear algebra which are summarized in the following lemma.

Lemma 5.2.1. *Let $A \in \mathbb{R}^{n\times n}$ be SPSD,* $\operatorname{rank}(A) = n-m$, *$n > m$ and*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = Q^{-1} \cdot \begin{pmatrix} A_{11} & 0 \\ 0 & 0 \end{pmatrix} \cdot Q^{-T}\,, \tag{5.1}$$

where $A_{11} \in \mathbb{R}^{(n-m)\times(n-m)}$, $A_{12} = A_{21}^T \in \mathbb{R}^{(n-m)\times m}$ and $A_{22} \in \mathbb{R}^{m\times m}$. Further, $Q^T \in \mathbb{R}^{n\times n}$ is a matrix of the form

$$Q^T = \begin{pmatrix} I_{n-m} & \phi_1 \\ 0 & \phi_2 \end{pmatrix} \quad \text{with} \quad \phi = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} \in \mathbb{R}^{n\times m}$$

a basis of $\ker(A)$. *Then for every SPSD matrix $B \in \mathbb{R}^{n\times n}$ with* $\ker(B) = \ker(A)$ *the generalized eigenvalue problem*

$$A\underline{u} = \lambda B\underline{u}$$

can be written as

$$A_{11}\underline{v}_1 = \lambda B_{11}\underline{v}_1, \qquad \underline{v}_2 = 0$$

with the notation

$$\begin{pmatrix} \underline{v}_1 \\ \underline{v}_2 \end{pmatrix} = Q^{-T}\underline{u}$$

and regular sub-matrices A_{11}, B_{11}.

Proof. The proof of these elementary results is left to the reader. □

In order to get a robust AMG method we propose a technique to produce a spectrally equivalent M-matrix B_h from the stiffness matrix K_h. This is important because if the non-negative values in the stiffness matrix K_h get too large, then AMG will lose robustness and efficiency. For instance, the M-matrix property of K_h can be lost if higher order (e.g. quadratic) FE-functions are used, or if long thin rectangles or flat prisms are taken for an FE-discretization. One method to get a spectrally equivalent M-matrix is to exploit information from the element stiffness matrices. A part of the following lemma can be found already in [5].

Lemma 5.2.2. *Let the stiffness matrix $K_h \in \mathbb{R}^{N_h\times N_h}$ be SPD, and K_h be assembled 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*

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

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

$$c_1^{(r)} \cdot B^{(r)} \le K^{(r)} \le c_2^{(r)} \cdot B^{(r)} \qquad \forall r \in \tau_h \tag{5.2}$$

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

$$B_h = \sum_{r\in\tau_h} A_r^T B^{(r)} A_r \tag{5.3}$$

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

$$c_1 \cdot B_h \le K_h \le c_2 \cdot B_h, \tag{5.4}$$

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 and an element of Z_{N_h}. If K_h is only SPSD, then the spectral equivalence inequalities (5.4) remain valid, $B_h \in Z_{N_h}$ and SPSD, and $\ker(B_h) = \ker(K_h)$.

Proof. Let us suppose that K_h is SPSD (Note, the SPD case is included.). It is easy to see that

$$\left(\sum_{r\in\tau_h} A_r^T K^{(r)} A_r \underline{u}, \underline{u}\right) = \sum_{r\in\tau_h} (K^{(r)} A_r \underline{u}, A_r \underline{u})\,.$$

It follows from (5.2) that

$$\sum_{r\in\tau_h} c_1^{(r)} \cdot (B^{(r)} A_r \underline{u}, A_r \underline{u}) \le \sum_{r\in\tau_h} (K^{(r)} A_r \underline{u}, A_r \underline{u}) \le \sum_{r\in\tau_h} c_2^{(r)} \cdot (B^{(r)} A_r \underline{u}, A_r \underline{u})\,.$$

By taking the minimum $c_1 = \min_{r\in\tau_h}\{c_1^{(r)}\}$ on the left side and the maximum $c_2 = \max_{r\in\tau_h}\{c_2^{(r)}\}$ on the right side, the result follows immediately, i.e.,

$$c_1 \cdot B_h \le K_h \le c_2 \cdot B_h\,.$$

Finally, every spectrally equivalent element matrix $B^{(r)} \in Z_{n_r}$ and thus $B_h \in Z_{N_h}$. □

We note that in the 2D case $n_r = 3$ and $n_r = 4$ represent linear and bilinear elements, respectively. Similarly, linear and trilinear elements for the 3D case are represented by $n_r = 4$ and $n_r = 8$, respectively. Lemma 5.2.2 is the theoretical background for Alg. 7. This algorithm returns the best SPSD $B^{(r)} \in Z_{n_r}$ from the element matrix $K^{(r)}$ with respect to the ratio $\frac{c_2^{(r)}}{c_1^{(r)}}$ of the spectral equivalence constants.

5.3 Optimization Procedure

In Alg. 7 a restricted optimization problem has to be solved. The optimization routine currently used is based on an SQP (sequential quadratic programming) algorithm using a Quasi-Newton update formula for estimating the Hessian of the

Algorithm 7 GeneralSpectralMatrix ($K^{(r)}$)

```
$n_r \leftarrow \dim(K^{(r)})$
if $K^{(r)} \notin Z_{n_r}$ then
  if $K^{(r)}$ is SPSD then
    Transform $K^{(r)}$ by Remark 5.2.1 to $K$
    Update $n_r$
  else
    $K \leftarrow K^{(r)}$
  end if

  Calculate $B$ from the restricted minimization problem
  $X \leftarrow K^{-1/2} B K^{-1/2}$

  $\frac{\mu_{max}(X)}{\mu_{min}(X)} \rightarrow$ min subject to $B \in Z_{n_r}$ and $B$ is SPD

  if $K^{(r)}$ is SPSD then
    Transform $B$ back by Remark 5.2.1 to $\tilde{B}$
  else
    $\tilde{B} \leftarrow B$
  end if
else
  $\tilde{B} \leftarrow K^{(r)}$
end if

RETURN $\tilde{B}$
```

objective (see [80]). Therefore we should find a good initial guess for the solution of the optimization problem and determine the gradient of the objective function. For simplicity we restrict ourself to the important cases $\ker(K^{(r)}) = \{0\}$ and $\ker(K^{(r)}) = \text{span}\{\mathbf{1}\}$ (i.e., $\sum_{j=1}^{n_r} (K^{(r)})_{ij} = 0$ for all $i = 1, \dots, n_r$). Additionally we omit the index $r \in \tau_h$ for further discussion (i.e., $K = K^{(r)}$, $B = B^{(r)}$, and $n = n_r$). First, we calculate the gradient of the objective function for the case of SPD element matrices K.

Lemma 5.3.1. *Let us consider the generalized eigenvalue problem*

$$K\underline{u} = \lambda B\underline{u} \tag{5.5}$$

with some given SPD matrix $K \in \mathbb{R}^{n\times n}$ and an SPD matrix $B \in \mathbb{R}^{n\times n}$. Equation (5.5) is equivalent to the standard eigenvalue problem (see Section 2.5)

$$X\phi = \mu\phi \tag{5.6}$$

with $X = K^{-1/2} B K^{-1/2}$, $\mu = \mu(X) = \frac{1}{\lambda}$ and $\phi = \phi(X) = K^{-1/2}\underline{u}$ denote the eigenvalues and the normalized eigenvectors of X, respectively. The partial derivative *of the condition number*

$$\kappa(X) = \frac{\mu_{\max}(X)}{\mu_{\min}(X)} \tag{5.7}$$

with respect to b_{ij} $(i, j \in \{1, \dots, n\})$ *is given by*

$$\frac{\partial \kappa(X)}{\partial b_{ij}} = \kappa(X) \cdot \left(\zeta_{ij}^{\max}(X) - \zeta_{ij}^{\min}(X)\right) \, ,$$

with

$$\zeta_{ij}^{*}(X) = (\phi_*(X))_j \cdot (\phi_*(X))_i \cdot \frac{1}{\mu_*(X)} \quad \textit{and} \quad * \in \{\min, \max\} \, . \tag{5.8}$$

Proof. First, (5.6) is multiplied with ϕ^T and we get

$$\mu = \frac{(X\phi, \phi)}{(\phi, \phi)}. \tag{5.9}$$

By using the chain rule and (5.6), the derivative of (5.9) with respect to b_{ij}, i.e., $\dot{\mu} = \frac{\partial \mu}{\partial b_{ij}}$, reads as

$$\dot{\mu} = \frac{((\dot{X}\phi, \phi) + 2 \cdot (X\phi, \dot{\phi})) \cdot (\phi, \phi) - 2 \cdot (\phi, \dot{\phi}) \cdot (X\phi, \phi)}{(\phi, \phi)^2} = (\dot{X}\phi, \phi) = \phi_i \cdot \phi_j \, . \tag{5.10}$$

In the last formula we used that $(\phi, \phi) = 1$. With a straight forward calculation involving (5.10) and (5.6) we get with (5.8)

$$\frac{\partial \kappa(X)}{\partial b_{ij}} = \frac{\mu_{\max}(X)}{\mu_{\min}(X)} \cdot \left(\zeta_{ij}^{\max}(X) - \zeta_{ij}^{\min}(X)\right) \, ,$$

which is the desired result. □

Remark 5.3.2. *Lemma 5.3.1 gives a result for the gradient of the objective function if the underlying matrices K and B are SPD. If K and B are SPSD with* $\ker(K) = \ker(B)$ *then Remark 5.2.1 has to be applied first.*

A good choice for the initial guess is very important for the efficiency of the optimization problem. As it was mentioned above, two important cases of element stiffness matrices are considered. First, the case $K \in \mathbb{R}^{n \times n}$, $\ker(A) = \{0\}$. Alg. 8 constructs an appropriate initial guess and Lemma 5.3.3 gives the spectral equivalence constants.

Lemma 5.3.3. *Let $K \in \mathbb{R}^{n \times n}$ be SPD, $K \notin Z$ and let $B \in \mathbb{R}^{n \times n}$ be the matrix produced by Alg. 8. Then $B \in Z$, B is SPD and $B = K + M$ with*

$$M = \begin{cases} m_{ii} \geq 0 & i = 1, \dots, n \\ m_{ij} \leq 0 & i \neq j \, . \end{cases} \tag{5.11}$$

M is weakly diagonally dominant, i.e., $\sum_{j=1,\dots,n} m_{ij} \geq 0$ for all $i = 1, \dots, n$. Moreover $c_1 \cdot B \leq K \leq c_2 \cdot B$ with spectral equivalence constants $c_1 = \frac{\mu_{\min}(K)}{\mu_{\min}(K) + \mu_{\max}(M)}$ and $c_2 = 1$, respectively.

Algorithm 8 StartMatrix_SPD(K)

```
$n \leftarrow \dim(K)$
$B \leftarrow 0$
**for all** i=1,n **do**
  **for all** j=i,n **do**
    **if** $k_{ij} > 0$ and $i \neq j$ **then**
      $b_{ii} \leftarrow b_{ii} + \frac{k_{ij}}{2}$
      $b_{jj} \leftarrow b_{jj} + \frac{k_{ij}}{2}$
    **else**
      $b_{ij} \leftarrow k_{ij}$
    **end if**
  **end for**
**end for**
RETURN $B$
```

Proof. It is obvious that $B \in Z$ and B is symmetric. Next, we investigate the spectral equivalence constants. Because $K \notin Z$ there exists at least one positive off diagonal element. Therefore $M \neq 0$, i.e., M can not be the zero matrix. Additionally, the matrix B produced by Alg. 8 can be written as $B = K + M$. The matrix M is weakly diagonally dominant by construction. This is easy to see because for every positive off diagonal element k_{ij} Alg. 8 updates the diagonal elements m_{ii} and m_{jj} with $\frac{k_{ij}}{2}$, and the off diagonal entries m_{ij} and m_{ji} with the value $-\frac{k_{ij}}{2}$. We have by construction $\sum_{j=1,\dots,n} m_{ij} = 0$ for all $i = 1, \dots, n$. Therefore

$$(M\underline{u}, \underline{u}) \geq 0 \qquad \forall \underline{u} \in \mathbb{R}^n.$$

Thus, on one hand we get

$$(K\underline{u}, \underline{u}) \leq (K\underline{u}, \underline{u}) + (M\underline{u}, \underline{u}) = (B\underline{u}, \underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^n\,.$$

On the other hand we can write

$$(B\underline{u}, \underline{u}) = (K\underline{u}, \underline{u}) + (M\underline{u}, \underline{u}) = (K\underline{u}, \underline{u}) \cdot \left(1 + \frac{(M\underline{u}, \underline{u})}{(K\underline{u}, \underline{u})}\right) \qquad \forall \underline{u} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$$

which results in

$$(B\underline{u}, \underline{u}) \leq (K\underline{u}, \underline{u}) \cdot \left(1 + \frac{\mu_{\max}(M)}{\mu_{\min}(K)}\right) \qquad \forall \underline{u} \in \mathbb{R}^n \setminus \{\mathbf{0}\}\,.$$

The result follows by defining $c_1 = \frac{\mu_{\min}(K)}{\mu_{\min}(K)+\mu_{\max}(M)}$ and $c_2 = 1$. We have shown that B is SPD and thus B is an admissible initial guess for the optimization problem. $\square$

Let us mention that the lower bound c_1 of Lemma 5.3.3 can be supposed to be uniformly bounded away from zero as h tends to zero.

The second case, i.e., $K \in \mathbb{R}^{n\times n}$ SPSD, $\ker(K) = \operatorname{span}\{\mathbf{1}\}$, requires some extra work. We define $\psi^{ij} \in \mathbb{R}^n$ by

$$(\psi^{ij})_l = \begin{cases} 1 & \text{if l = i} \\ -1 & \text{if l = j} \\ 0 & \text{else} \end{cases} \qquad l = 1, \dots, n\,.$$

Actually ψ^{ij} and $\psi^{ij} \cdot \psi^{ij^T}$ look like

$$\psi^{ij} = \begin{array}{c|l} \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \end{pmatrix} & \begin{array}{l} \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{array} \end{array} \in \mathbb{R}^n \qquad \psi^{ij} \cdot \psi^{ij^T} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n} .$$

Further, we define $K_{ij} \in \mathbb{R}^{2\times 2}$ by

$$K_{ij} = \begin{pmatrix} k_{ii} & k_{ij} \\ k_{ji} & k_{jj} \end{pmatrix} \qquad \forall i, j = 1, \dots , n \quad i \neq j \tag{5.12}$$

and a permutation matrix $P_{ij} \in \mathbb{R}^{n\times n}$ such that

$$\tilde{K} = P_{ij}^T K P_{ij} = \begin{pmatrix} K_{ij} & C_{ij}^T \\ C_{ij} & D_{ij} \end{pmatrix} \tag{5.13}$$

holds, where $C_{ij} \in \mathbb{R}^{(n-2)\times 2}$ and $D_{ij} \in \mathbb{R}^{(n-2)\times(n-2)}$. In addition a Schur-complement $S_{ij} \in \mathbb{R}^{2\times 2}$ of $\tilde{K}$ is calculated which is given by

$$S_{ij} = K_{ij} - C_{ij}^T D_{ij}^{-1} C_{ij} = \begin{pmatrix} \beta_{ij} & -\beta_{ij} \\ -\beta_{ij} & \beta_{ij} \end{pmatrix} \tag{5.14}$$

with a value $\beta_{ij} \in \mathbb{R}^+$.

Remark 5.3.4.

1. *The regularity of D_{ij} can be seen in the following way: Let us assume that D_{ij} is not regular, i.e., $\exists \underline{v} \in \mathbb{R}^{n-2}$ such that $(D_{ij}\underline{v}, \underline{v}) = 0$. Further define*

$$\tilde{\underline{v}} = \begin{pmatrix} 0 \\ \underline{v} \end{pmatrix} \in \mathbb{R}^n .$$

   Applying (5.13) to $\tilde{\underline{v}}$ results in

$$(\tilde{K}\tilde{\underline{v}}, \tilde{\underline{v}}) = (D_{ij}\underline{v}, \underline{v}) = 0 .$$

   The last result implies $\tilde{\underline{v}} \in \ker(\tilde{K})$, which is a contradiction to $\ker(\tilde{K}) =$ span$\{\mathbf{1}\}$ and consequently D_{ij} has to be regular.

2. *The special structure of the Schur-complement (5.14) is given, because the null-space of K implies that*

$$\sum_{j=1}^{n} K_{ij} = 0 \qquad \forall i = 1, \dots , n .$$

   The Schur-complement preserves the weak diagonal dominance and the SPSD property. Therefore β_{ij} has to be positive.

Algorithm 9 StartMatrix_SPSD(K)

```
$n \leftarrow \dim(K)$
$B \leftarrow 0$
for all i=1,n do
  for all j=1,n do
    if $i \neq j$ then
      Calculate the Schur-complement $S_{ij}$ and get parameter $\beta_{ij}$
      $B \leftarrow B + \beta_{ij} \cdot \psi^{ij} \cdot \psi^{ij^T}$
    end if
  end for
end for
RETURN $B$
```

With the definitions above, Alg. 9 produces a good initial guess for Alg. 7 can be written in the following form:
The following Lemma provides the corresponding spectral constants.

Lemma 5.3.5. *Let $K \in \mathbb{R}^{n\times n}$ be SPSD with $\ker(K) = span\{\mathbf{1}\}$, $K \notin Z$ and let $B \in \mathbb{R}^{n\times n}$ be the matrix produced by Alg. 9. Then $B \in Z$, B is SPSD and $c_1 \cdot B \leq K \leq c_2 \cdot B$ holds with spectral equivalence constants $c_1 = \frac{1}{n\cdot(n-1)}$ and $c_2 = \frac{1}{\min_{i,j;i\neq j}\{\frac{\beta_{ij}}{\alpha_{ij}},1\}}$, where $\alpha_{ij} = \frac{k_{ij}}{2}$ and β_{ij} is defined by (5.14).*

Proof. Alg. 9 results in

$$B = \sum_i \sum_{j\neq i} \beta_{ij} \cdot \psi_{ij} \cdot \psi_{ij}^T \tag{5.15}$$

and therefore

$$(B\underline{u}, \underline{u}) = \sum_i \sum_{j\neq i} \beta_{ij} \cdot (u_i - u_j)^2 \geq 0 \qquad \forall \underline{u} \in \mathbb{R}^n\,,$$

because of $\beta_{ij} > 0$. As a consequence we get

$$\ker(B) = \text{span}\{\mathbf{1}\} = \ker(K)\,.$$

Thus $B \in Z$ and SPSD. Next, the spectral equivalence constants are determined. We observe, that

$$(\beta_{ij} \cdot \psi_{ij} \cdot \psi_{ij}^T \underline{u}, \underline{u}) \leq (K\underline{u}, \underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^n\,. \tag{5.16}$$

This is valid because the Schur-complement produces an orthogonal splitting of $\mathbb{R}^n$ with respect to the energy inner product, i.e., $(K\underline{u}, \underline{u})$. The necessary transformation is given by

$$T_{ij} = \begin{pmatrix} I_2 \\ -D_{ij}^{-1} C_{ij} \end{pmatrix} \in \mathbb{R}^{n\times 2}\,.$$

The Schur-complement (5.14) can be written as

$$S_{ij} = T_{ij}^T \tilde{K} T_{ij} \, .$$

Each vector $\underline{u} \in \mathbb{R}^n$ can be expressed as $\underline{u} = T_{ij}\underline{w} + \underline{v}$ with $\underline{w} \in \mathbb{R}^2$ and $\underline{v} \in \mathbb{R}^n$, such that $(KT_{ij}\underline{w}, \underline{v}) = 0$. By using $(\tilde{K}\underline{u}, \underline{u}) = (K\underline{u}, \underline{u})$ we get

$$(\tilde{K}\underline{u}, \underline{u}) = (\tilde{K}(T_{ij}\underline{w} + \underline{v}), T_{ij}\underline{w} + \underline{v}) = (T_{ij}^T \tilde{K} T_{ij}\underline{w}, \underline{w}) + (\tilde{K}\underline{v}, \underline{v}) + 2 \cdot (\tilde{K}T_{ij}\underline{w}, \underline{v}) \, . \tag{5.17}$$

The orthogonality $(KT_{ij}\underline{w}, \underline{v}) = 0$ and the semi-definiteness of K (5.17) yields

$$(K\underline{u}, \underline{u}) = (\tilde{K}\underline{u}, \underline{u}) \geq (T_{ij}^T \tilde{K} T_{ij}\underline{w}, \underline{w}) = (S_{ij}\underline{w}, \underline{w}) \, , \tag{5.18}$$

which is the desired result. The lower bound follows with (5.15), (5.16), and (5.18), i.e.,

$$(B\underline{u}, \underline{u}) \leq n \cdot (n-1) \cdot (K\underline{u}, \underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^n \, .$$

Further, we can express K as $K = \sum_i \sum_{j \neq i} \alpha_{ij} \cdot \psi_{ij} \cdot \psi_{ij}^T$, with $\alpha_{ij} = \frac{k_{ij}}{2}$. By comparing components we get

$$(\beta_{ij} \cdot \psi_{ij} \cdot \psi_{ij}^T \underline{u}, \underline{u}) \geq c_h \cdot (\alpha_{ij} \cdot \psi_{ij} \cdot \psi_{ij}^T \underline{u}, \underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^n \, ,$$

with $c_h = \min_{i,j;i \neq j}\{\frac{\beta_{ij}}{\alpha_{ij}}, 1\}$. Define $c_1 = \frac{1}{n \cdot (n-1)}$ and $c_2 = \frac{1}{c_h}$ to complete the proof. □

Again we mention that the upper bound c_2 of Lemma 5.3.5 can be supposed to be uniformly bounded from above as h tends to zero. Therefore our spectral equivalence constants for the stiffness matrix K_h with respect to the constructed M-matrix B_h are independent of the discretization parameter h.

Remark 5.3.6.

1. *In practice the generalized condition number $\kappa(B^\dagger K)$ of the original matrix K and the initial guess B produced by Alg. 8 and Alg. 9 is approximately between 10 and 100. Applying a few optimization steps (i.e., approximately 10 steps) we achieved $\kappa \leq 10$ for many examples (see Chapter 8).*

2. *Let us consider the element stiffness matrices K arising from the bilinear and the linear FE-discretization of the Laplace operator on an anisotropic rectangular element and on an anisotropic triangular element, respectively (see Fig. 5.1). In these two cases, we can establish the explicit dependence of the generalized condition number $\kappa(K^\dagger B)$ on the anisotropy parameter q.*

   Indeed, the element stiffness matrix of the rectangle has the form

$$K = \frac{1}{6q} \begin{pmatrix} 2+2q^2 & 1-2q^2 & -2+q^2 & -1-q^2 \\ 1-2q^2 & 2+2q^2 & -1-q^2 & -2+q^2 \\ -2+q^2 & -1-q^2 & 2+2q^2 & 1-2q^2 \\ -1-q^2 & -2+q^2 & 1-2q^2 & 2+2q^2 \end{pmatrix} \, . \tag{5.19}$$

For $0 < q < 1$*,* $K \notin Z$*. Already Alg. 9 produces a matrix* B*, for which the estimate*

$$\kappa(K^{\dagger}B) \leq 8$$

holds, i.e., the generalized condition number is independent of q*. We recall that the matrix* B *produced by Alg. 9 is the initial guess for Alg. 7 that minimizes the general condition number.*

In the case of an anisotropic triangle, the element stiffness matrix K *has the form*

$$K = \frac{1}{4q} \begin{pmatrix} 1+q^2 & 1-q^2 & -2 \\ 1-q^2 & 1+q^2 & -2 \\ -2 & -2 & 4 \end{pmatrix} . \tag{5.20}$$

Again, for $0 < q < 1$*,* $K \notin Z$*. The best matrix* $B \in Z$ *in the sense of Alg. 7 is*

$$B = \frac{1}{4q} \begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & -2 \\ -2 & -2 & 4 \end{pmatrix} . \tag{5.21}$$

Solving the generalized eigenvalue problem, we get

$$\kappa(K^{\dagger}B) = \frac{1}{q^2}.$$

Thus, if q *tends to zero then the generalized condition number grows very fast. Anyway, in case of an isotropic differential operator it is not advisable to use such elements for an FE-discretization because of their poor approximation property. Nevertheless, this technique is successful for many examples (see Chapter 8).*

Figure 5.1: Thin FE-structures.

5.4 Classification Strategy

In many cases the element stiffness matrices $K^{(r)} \in \mathbb{R}^{n_r \times n_r}$, $r \in \tau_h$, are 'similar' up to a factor in the sense of spectral equivalence. In other words only some of the element matrices have to be optimized. We use the following procedure in practice. First, we classify which element stiffness matrices are 'similar' in the sense of spectral equivalence. Then one matrix of each class is optimized and the remaining matrices of this class have the same spectrally equivalent matrix. This method saves a huge amount of computational work. The next lemma shows that this technique works. We re-scale every element matrix $K^{(r)}$ as follows

$$\tilde{K}^{(r)} = \frac{1}{\max_{i=1,\dots,n_r}\{k_{ii}^{(r)}\}} \cdot K^{(r)} \qquad \forall r \in \tau_h. \tag{5.22}$$

Lemma 5.4.1. *Let $K^{(r)}, K^{(s)} \in \mathbb{R}^{n\times n}$ ($n_r = n_s = n$) be SPD. Assume that $K^{(r)}, K^{(s)}$ are scaled as in (5.22) and that*

$$K^{(s)} = K^{(r)} + \Delta \tag{5.23}$$

where $\Delta \in \mathbb{R}^{n\times n}$ is symmetric. Moreover let $B^{(r)} \in \mathbb{R}^{n\times n}$, $B^{(r)} \in Z_n$ be SPD and $c_1 \cdot B^{(r)} \le K^{(r)} \le c_2 \cdot B^{(r)}$. If there exist $\delta_1, \delta_2, \lambda_1 \in \mathbb{R}^+$ such that

$$-\delta_1 \le \frac{(\Delta\underline{u},\underline{u})}{(\underline{u},\underline{u})} \le \delta_2 \qquad \forall \underline{u} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \tag{5.24}$$

$$\lambda_1 \le \frac{(B^{(r)}\underline{u},\underline{u})}{(\underline{u},\underline{u})} \qquad \forall \underline{u} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \tag{5.25}$$

and $c_1 > \frac{\delta_1}{\lambda_1}$ hold, then $K^{(s)}$ is spectrally equivalent to $B^{(r)}$, i.e.,

$$\left(c_1 - \frac{\delta_1}{\lambda_1}\right) \cdot B^{(r)} \le K^{(s)} \le \left(c_2 + \frac{\delta_2}{\lambda_1}\right) \cdot B^{(r)} .$$

Proof. Starting with

$$c_1 \cdot (B^{(r)}\underline{u},\underline{u}) \le (K^{(r)}\underline{u},\underline{u}) \le c_2 \cdot (B^{(r)}\underline{u},\underline{u}) \qquad \forall \underline{u} \in \mathbb{R}^n \setminus \{0\}$$

we get for all $\underline{u} \in \mathbb{R}^n \setminus \{0\}$

$$c_1 \cdot (B^{(r)}\underline{u},\underline{u}) + (\Delta\underline{u},\underline{u}) \le (K^{(s)}\underline{u},\underline{u}) \le c_2 \cdot (B^{(r)}\underline{u},\underline{u}) + (\Delta\underline{u},\underline{u}) ,$$

or equivalently,

$$(B^{(r)}\underline{u},\underline{u}) \cdot \left(c_1 + \frac{(\Delta\underline{u},\underline{u})}{(B^{(r)}\underline{u},\underline{u})}\right) \le (K^{(s)}\underline{u},\underline{u}) \le (B^{(r)}\underline{u},\underline{u}) \cdot \left(c_2 + \frac{(\Delta\underline{u},\underline{u})}{(B^{(r)}\underline{u},\underline{u})}\right).$$

With assumptions (5.24), (5.25) we conclude the proof. □

Remark 5.4.2.

1. *Lemma 5.4.1 gives a result for the classification of SPD matrices. SPSD matrices $K^{(r)}$ and $K^{(s)}$ ($r \ne s$) can only be in the same class if $\ker(K^{(r)}) = \ker(K^{(s)})$. If this is the case, $K^{(r)}$ and $K^{(s)}$ are transformed into (5.1) and we can work with the regular parts of $K^{(r)}$ and $K^{(s)}$. Note that in view of the conditions $c_1 > \frac{\delta_1}{\lambda_1}$ and (5.25), relation (5.24) turns into a necessary condition which $\Delta \in \mathbb{R}^{n\times n}$ is admissible.*

2. *Practically, the matrix norm $\|\Delta\|_\infty = \max_i \sum_{j=1}^n \Delta_{ij}$ is calculated and δ_1 is set sufficiently small (e.g. $\delta_1 = 10^{-1}$).*

Chapter 6

A Parallel Version of AMG

6.1 A Unified Data Distribution Concept

The parallelization of AMG is essentially done by using ideas of domain decomposition methods and apply them carefully to the AMG method. The method presented here relies on ideas presented by G. Haase in [29, 30].

6.1.1 Notations

We use a non-overlapping domain decomposition for our approach, i.e., we decompose $\overline{\Omega}$ into P subdomains $\overline{\Omega}_s$ such that

$$\overline{\Omega} = \bigcup_{s=1}^{P} \overline{\Omega}_s \text{ with } \Omega_s \cap \Omega_q = \emptyset,\ \forall q \neq s,\ s, q = 1, \ldots, P$$

holds. Each subdomain Ω_s is discretized by a mesh $\mathcal{T}_{h,s}$, such that the whole triangulation $\mathcal{T}_h = \bigcup\limits_{s=1}^{P} \mathcal{T}_{h,s}$ of Ω is conform. A global FE-space $\mathbb{V}_h$ is defined with respect to $\mathcal{T}_h$ and the local spaces $\mathbb{V}_{h,s}$ are restrictions of $\mathbb{V}_h$ onto $\mathcal{T}_{h,s}$. The index set of nodes in $\overline{\Omega}_s$ is denoted by ω_s^n. Note, that

$$\omega_h^n = \bigcup_{s=1}^{P} \omega_s^n \quad \text{but} \quad N_h = |\omega_h^n| \leq \sum_{s=1}^{P} |\omega_s^n| =: \sum_{s=1}^{P} N_s$$

holds, i.e., different subdomains may share unknowns although elements from different subdomains do not overlap.

The mapping of a vector $\underline{u}_h \in \mathbb{R}^{N_h}$ in global numbering onto a local vector $\underline{u}_s \in \mathbb{R}^{N_s}$ in subdomain $\overline{\Omega}_s$ $(s = 1, \ldots, P)$ is represented symbolically by subdomain connectivity matrices $A_s : \mathbb{R}^{N_h} \mapsto \mathbb{R}^{N_s}$ with entries

$$(A_s)_{ij} := \begin{cases} 1 & \text{if} \quad j \widehat{=} \text{global number of } i \\ 0 & \text{else} \end{cases} \qquad \forall i \in \omega_s^n\,,\ \forall j \in \omega_h^n\ . \tag{6.1}$$

The transpose A_s^T of these binary matrices A_s maps a local vector back to the global one. The index set of all those subdomains containing a grid point x_j, $j \in \omega_h^n$ is denoted by

$$\sigma^{[j]} := \{s \,|\, x_j \in \overline{\Omega}_s\} = \{s \,|\, \exists i \in \omega_s^n : (A_s)_{ij} \neq 0\} . \tag{6.2}$$

6.1.2 Vector and Matrix Types

We store the data of a vector component u_i on processor s if $s \in \sigma^{[i]}$. There are (at least) two opportunities to store those components and consequently that vector (see also Fig. 6.1).

1. A vector $\underline{\mathfrak{u}}$ is called an *accumulated* vector if each vector component $\mathfrak{u}_i$ is stored in all subdomains Ω_s, $s \in \sigma^{[i]}$ with its full value. The local vectors $\underline{\mathfrak{u}}_s$ can be represented as

$$\underline{\mathfrak{u}}_s := A_s \cdot \underline{\mathfrak{u}} . \tag{6.3}$$

2. A vector $\underline{\mathsf{r}}$ is called a *distributed* vector if it is decomposed into local vectors $\underline{\mathsf{r}}_s$ such that

$$\underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \cdot \underline{\mathsf{r}}_s \tag{6.4}$$

holds, i.e., all subdomains Ω_s, $s \in \sigma^{[i]}$ store only $\underline{\mathsf{r}}_s$ and possess a portion of the full vector value r_i which can be determined only by taking the sum in (6.4).

The conversion of a distributed vector $\underline{\mathsf{v}}$ into an accumulated vector $\underline{\mathfrak{w}}$ can be done by evaluating the sum in (6.4) followed by the restriction (6.3), i.e.,

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathsf{v}} \quad : \quad \underline{\mathfrak{w}}_s := A_s \cdot \underline{\mathfrak{w}} = A_s \cdot \sum_{s=1}^{P} A_s^T \cdot \underline{\mathsf{v}}_s . \tag{6.5}$$

The conversion in the other direction is not unique - we prefer an equally weighted distribution of the accumulated vector. A matrix weighted distribution is also feasible in case of varying coefficients. The weights are chosen such that the re-conversion (6.5) will result in the original vector :

$$\underline{\mathsf{v}} \leftarrow \underline{\mathfrak{w}} \quad : \quad \underline{\mathsf{v}}_s := (R_s)^{-1} \cdot \underline{\mathfrak{w}}_s , \tag{6.6}$$

with

$$R_s = \operatorname*{diag}_{i \in \omega_s^n} \{|\sigma^{[i]}|\} ,$$

i.e., $(R_s)_{ii}$ stores the number of subdomains the component i belongs to. The system matrix defined in (2.53) can also be stored in two ways. With respect to an element-wise domain decomposition, we can store the FE-matrix accumulated or distributed.

1. A matrix $\mathfrak{M}$ is called *accumulated* if its local restrictions $\mathfrak{M}_s$ possess the full entries of it, and we can write

$$\mathfrak{M}_s := A_s \cdot \mathfrak{M} \cdot A_s^T . \tag{6.7}$$

Figure 6.1: Illustration for accumulated (a) and distributed (b) vectors and matrices.

2. We call a matrix K *distributed* if we have locally stored matrices K_s such that

$$\mathsf{K} \;:=\; \sum_{s=1}^{P} A_s^T \cdot \mathsf{K}_s \cdot A_s \tag{6.8}$$

   holds, i.e., each subdomain $\overline{\Omega}_s$ stores only a part of its full entries.

We obtain distributed matrices K_s automatically after the local FE-accumulation with respect to $\mathbb{V}_{h,s}$, due to our non-overlapping construction principle. The sum in (6.8) is equivalent to a global matrix accumulation which will not be carried out in the parallel case.

6.1.3 Basic Operations

This section collects some results from the papers [28, 29, 30, 36]. It is trivial that additive combinations of vectors from the same type do not require any communication. The inner product of different type vectors requires one global reduce operation of the local inner products:

$$(\underline{\mathfrak{w}}, \underline{\mathsf{r}}) \;=\; \underline{\mathfrak{w}}^T \underline{\mathsf{r}} \;=\; \underline{\mathfrak{w}}^T \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s \;=\; \sum_{s=1}^{P} (A_s \underline{\mathfrak{w}})^T \underline{\mathsf{r}}_s \;=\; \sum_{s=1}^{P} (\underline{\mathfrak{w}}_s, \underline{\mathsf{r}}_s) \;. \tag{6.9}$$

Any other combination of vector types requires more effort. The multiplication of a distributed matrix K with an accumulated vector $\underline{\mathfrak{w}}$

$$\mathsf{K}\underline{\mathfrak{w}} \;=\; \sum_{s=1}^{P} A_s^T \mathsf{K}_s A_s \cdot \underline{\mathfrak{w}} \;=\; \sum_{s=1}^{P} A_s^T \left(\mathsf{K}_s \underline{\mathfrak{w}}_s\right) \;=\; \sum_{s=1}^{P} A_s^T \underline{\mathsf{v}}_s \;=\; \underline{\mathsf{v}} \;, \tag{6.10}$$

results in a distributed vector. The realization requires no communication at all because we only have to compute locally $\underline{\mathsf{v}}_s = \mathsf{K}_s \underline{\mathfrak{w}}_s$.

The situation changes if we use an accumulated matrix $\mathfrak{M}$. Here we have to ensure that the pattern of $\mathfrak{M}$ fulfills the condition

$$\forall i, j \in \omega_h^n : \qquad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}_{ij} = 0\,. \tag{6.11}$$

If the pattern condition (6.11) holds, then the operations

$$\underline{\mathfrak{w}} = \mathfrak{M}\underline{\mathfrak{u}} \qquad \text{and} \qquad \underline{\mathsf{d}} = \mathfrak{M}^T \underline{\mathsf{r}} \tag{6.12}$$

can be performed locally without any communication, i.e.,

$$\underline{\mathfrak{w}}_s = \mathfrak{M}_s\underline{\mathfrak{u}}_s \qquad \text{and} \qquad \underline{\mathsf{d}}_s = \mathfrak{M}_s^T \underline{\mathsf{r}}_s \quad \forall s = 1, \dots, P\,. \tag{6.13}$$

Under the assumption (6.11), a special product of three matrices can be performed locally without communication (see also [30]),

$$\tilde{\mathsf{K}} = \mathfrak{M}^T\, \mathsf{K}\, \mathfrak{M}\,, \tag{6.14}$$

i.e.,

$$\tilde{\mathsf{K}}_s = \mathfrak{M}_s^T\, \mathsf{K}_s\, \mathfrak{M}_s\,.$$

6.1.4 Basic Algorithms

The operations (6.9) and (6.10) already allow to formulate a parallel preconditioned conjugate gradient (PPCG) algorithm for solving the matrix equation (2.53), i.e.,

$$K_h \underline{u}_h = \underline{f}_h\,,$$

with a preconditioner C_h.

Algorithm 10 Parallel PCG algorithm $\textsc{ppcg}(\mathsf{K}_h, C_h, \underline{\mathfrak{u}}_h, \underline{\mathsf{f}}_h)$

```
while no convergence do
    $\underline{\mathsf{v}}_h \leftarrow \mathsf{K}_h \cdot \underline{\mathfrak{s}}_h$
    $\alpha \Leftarrow \sigma / (\underline{\mathfrak{s}}_h, \underline{\mathsf{v}}_h)$
    $\underline{\mathfrak{u}}_h \leftarrow \underline{\mathfrak{u}}_h + \alpha \cdot \underline{\mathfrak{s}}_h$
    $\underline{\mathsf{r}}_h \leftarrow \underline{\mathsf{r}}_h - \alpha \cdot \underline{\mathsf{v}}_h$
    $\underline{\mathfrak{w}}_h \Leftarrow C_h^{-1} \cdot \underline{\mathsf{r}}_h$
    $\sigma \Leftarrow (\underline{\mathfrak{w}}_h, \underline{\mathsf{r}}_h)$
    $\beta \leftarrow \sigma / \sigma_{\text{old}}$   ,   $\sigma_{\text{old}} \leftarrow \sigma$
    $\underline{\mathfrak{s}}_h \leftarrow \underline{\mathfrak{w}}_h + \beta \cdot \underline{\mathfrak{s}}_h$
end while
```

Besides the inner products, only the preconditioning step

$$\underline{\mathfrak{w}}_h \Leftarrow C_h^{-1} \cdot \underline{\mathsf{r}}_h$$

involves communication indicated by using $\Leftarrow$ instead of $\leftarrow$. In the case of $C_h = I_h$, i.e., no preconditioning, this step reduces to a type conversion (6.5) involving communication. We require that the communication costs for applying any other preconditioner C_h^{-1} are of the same order.

One possible choice for the preconditioner is $C_h^{-1} = (I_h - MG_h)K_h^{-1}$, with MG_h being the multigrid iteration operator for K_h. The parallel multigrid iteration is presented in Alg. 11, where ℓ denotes the level such that $\ell = 1$ stands for the finest grid. The algorithm applies the parallel version of the smoother $S_h(\mathsf{K}_h, \underline{\mathfrak{u}}_h, \underline{\mathsf{f}}_h)$ and its transpose, e.g. a block Jacobi smoother with Gauss-Seidel smoothing in the blocks containing interior unknowns of the subdomains, see also [46]. Furthermore, the interpolation $\mathfrak{P}_h$ has to fulfill the pattern condition (6.11). The coarse grid system can be solved directly (either in parallel or sequentially on one processor) or again by some iterative method similar to the PPCG in Alg. 10. Besides the coarse grid

Algorithm 11 Parallel multigrid PMG$(\mathsf{K}_h, \underline{\mathfrak{u}}_h, \underline{\mathsf{f}}_h, \ell)$

```
if $\ell ==$ CoarseLevel then
    $\underline{\mathfrak{u}}_h \Leftarrow$ Solve$\left( \sum_{s=1}^{P} A_s^T \mathsf{K}_s A_s \, \underline{\mathfrak{u}}_h = \underline{\mathsf{f}}_h \right)$
else
    $\widetilde{\underline{\mathfrak{u}}}_h \Leftarrow S_h^{pre}(\mathsf{K}_h, \underline{\mathfrak{u}}_h, \underline{\mathsf{f}}_h)$
    $\underline{\mathsf{d}}_h \leftarrow \underline{\mathsf{f}}_h - \mathsf{K}_h \widetilde{\underline{\mathfrak{u}}}_h$
    $\underline{\mathsf{d}}_H \leftarrow \mathfrak{P}_h^T \underline{\mathsf{d}}_h$
    $\underline{\mathfrak{w}}_H \leftarrow 0$
    $\underline{\mathfrak{w}}_H \Leftarrow$ PMG$(\mathsf{K}_H, \underline{\mathfrak{w}}_H, \underline{\mathsf{d}}_H, \ell + 1)$
    $\underline{\mathfrak{w}}_h \leftarrow \mathfrak{P}_h \underline{\mathfrak{w}}_H$
    $\widehat{\underline{\mathfrak{u}}}_h \leftarrow \widetilde{\underline{\mathfrak{u}}}_h + \underline{\mathfrak{w}}_h$
    $\underline{\mathfrak{u}}_h \Leftarrow S_h^{post}(\mathsf{K}_h, \widehat{\underline{\mathfrak{u}}}_h, \underline{\mathsf{f}}_h)$
end if
```

solver, only the smoothing sweeps require communication.

6.2 The Parallel AMG Algorithm and Its Implementation

It follows from Alg. 11, the Galerkin approach (3.4) and rule (6.14) that the crucial point for a recursive parallel AMG algorithm consists in guaranteeing the pattern condition (6.11) for the interpolation matrix $\mathfrak{P}_h$. This pattern is controlled by the strong connections $\{S_h^{i,T}\}$ as input parameters in WEIGHTS (see Subsection 3.3, page 33). Therefore, we have to control the sets of strong connections in the coarsening step to ensure the required interpolation pattern (see also [30]).

6.2.1 Communication Groups and Node Sets

Besides the global inner products in PPCG (Alg. 10) any other communication is only next neighbor communication involving a subset of all processors. The model example in Fig. 6.2 possesses by definition (6.2) the communication groups $\{1,2,3,4\}$, $\{1,2\}$, $\{1,3\}$, $\{2,4\}$, $\{3,4\}$ and trivially $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$. These communication groups σ_k are uniquely ordered with the rules

Figure 6.2: Non-overlapping subdomains with triangulation and row-wise numbering.

1. $|\sigma_k| > |\sigma_\ell| \quad \Rightarrow \quad k > \ell$,
2. $|\sigma_k| = |\sigma_\ell|$ and σ_k lexicographically larger than $\sigma_\ell \quad \Rightarrow \quad k > \ell$.

This ordering guarantees the consistency of local and global ordering and prevents communication deadlocks in the algorithm. Furthermore, let mc_s denote the number of communication groups of subdomain Ω_s.

Remark 6.2.1. *We get for subdomain 1 (i.e., Ω_1) in our example $mc_1 = 4$ groups where $\sigma_1 = \{1,2,3,4\}$, $\sigma_2 = \{1,2\}$, $\sigma_3 = \{1,3\}$ and $\sigma_4 = \{1\}$. The last one does not require any communication but simplifies the algorithms. It is obvious from Fig. 6.2, that $\sigma_1 = \sigma^{[25]}$, $\sigma_2 = \sigma^{[18]}$, $\sigma_3 = \sigma^{[24]}$ and $\sigma_4 = \sigma^{[1]}$.*

The definition of communication groups lead directly to the idea to group locally the nodes ω_s^n of subdomain Ω_s into four classes with respect to a certain communication group σ_k :

1. active nodes: $\omega_a^n := \{i \in \omega_s^n \,:\, \sigma^{[i]} \equiv \sigma_k\}$,
2. visible nodes: $\omega_v^n := \{i \in \omega_s^n \,:\, \sigma^{[i]} \subset \sigma_k\}$,
3. grouped nodes: $\omega_g^n := \{i \in \omega_s^n \,:\, \sigma^{[i]} \supset \sigma_k\}$,
4. invisible nodes : $\omega_s^n \setminus \left(\omega_a^n \cap \omega_v^n \cap \omega_g^n\right)$.

This results, e.g. for subdomain Ω_1 and $\sigma_3 = \{1,3\}$ in $\omega_a^n = \{22,23,24\}$, $\omega_g^n = \{25\}$ and $\omega_v^n = \{1,2,3,8,9,10,15,16,17\}$.

6.2.2 Parallel Coarsening

For the determination of strong connections as well as for the calculation of interpolation weights we need the accumulated system matrix instead of the given distributed one.

1. A first step to control the admissible interpolation pattern consists in a special accumulated matrix
$$\widetilde{\mathfrak{K}} = \sum_{s=1}^{P} A_s^T \widetilde{\mathsf{K}}_s A_s$$
with
$$(\widetilde{\mathsf{K}}_s)_{ij} = \begin{cases} (\mathsf{K}_s)_{ij} & \text{iff} \quad \sigma^{[i]} \subseteq \sigma^{[j]} \vee \sigma^{[i]} \supseteq \sigma^{[j]} \\ 0 & \text{else} \end{cases} , \tag{6.15}$$
which is locally stored as $\widetilde{\mathfrak{K}}_s = A_s \widetilde{\mathfrak{K}} A_s^T$. This means for Fig. 6.2, that the resulting matrix $\widetilde{\mathfrak{K}}$ does not contain entries $K_{18,26}$, $K_{24,32}$, $K_{26,18}$ and $K_{32,24}$.

2. The second step takes advantage of the fact that our ordering of the communication groups from Subsection 6.2.1 already provides the correct relation between the nodes due to ω_a^n, ω_v^n and ω_g^n. The local coarsening will be done subsequently on the active sets of all communication groups σ_k. We have to guarantee coherency of the resulting coarse and fine nodes for active sets shared by more than one subdomain. The easiest way consists in a coarsening only on the root process of communication group σ_k, then broadcasting the result to the group and finally doing the coarsening with the marked coarse nodes ω_m^n on the remaining processes. The two coarsening parts before and after the communication can be handled by the same routine
$$(\{S_h^{i,T}\}, \omega_C, \omega_F) \leftarrow \text{CoarseP}(\{S_h^{i,T}\}, \omega_m^n, \omega_a^n, \omega_v^n)\,,$$
which can be derived from the sequential routine Coarse (see Section 3.3, page 33). The sequential routine is contained in the parallel one via the call
$$(\{S_h^{i,T}\}, \omega_C, \omega_F) \leftarrow \text{CoarseP}(\{S_h^{i,T}\}, \emptyset, \omega_h^n, \omega_h^n)\,,$$
with $\omega_C = \omega_F = \emptyset$ as initial parameters.

3. A third step has to restrict the strong connections such that no fine node from the active set ω_a^n has connections to the visible nodes from ω_v^n.

These three steps guarantee the admissible pattern (6.13) for interpolation and restriction. Algorithm 12 presents the routine for parallel coarsening on each subdomain Ω_s. The modified strong connections $\{S_h^{i,T}\}$ ensure together with $\widetilde{\mathfrak{K}}$ the admissible matrix pattern for the interpolation. Therefore, we call
$$\mathfrak{P}_s \leftarrow \text{Weights}(\{S_h^{i,T}\}, \widetilde{\mathfrak{K}}, \omega_C, \omega_F)$$
on each processor s without any communication.

Algorithm 12 Parallel coarsening ParCoarse($\{S_h^{i,T}\}, \omega_s^n$)

```
Determine list $[\sigma_k]_{k=1,\dots,\mathrm{mc}_s}$ (= communicator groups).
$\omega_C \leftarrow \emptyset$, $\omega_F \leftarrow \emptyset$
for all $k = 1,\dots,\mathrm{mc}_s$ do
  $\omega_a^n \leftarrow \{i \in \omega_s^n : \sigma^{[i]} \equiv \sigma_k\}$,    $\omega_v^n \leftarrow \{i \in \omega_s^n : \sigma^{[i]} \subset \sigma_k\}$
  $\omega_g^n \leftarrow \{i \in \omega_s^n : \sigma^{[i]} \supset \sigma_k\}$,   $\omega_m^n \leftarrow \emptyset$
  if $s$ ==Root($\sigma_k$) then
    $(\omega_C, \omega_F) \leftarrow$ CoarseP ($\{S_h^{i,T}\}, \omega_m^n, \omega_a^n, \omega_v^n, \omega_C, \Omega_F$)
  end if
  $\omega_m^n \leftarrow$ Broadcast ($\sigma_k, \omega_C \cap \omega_a^n, \omega_C, \Omega_F$)
  if $s \neq$Root($\sigma_k$) then
    $(\omega_C, \omega_F) \leftarrow$ CoarseP ($\{S_h^{i,T}\}, \omega_m^n, \omega_a^n, \omega_v^n$)
  end if
  for all $i \in \omega_a^n$ do
    if $i \in \omega_F$ then
      $S_h^{i,T} \leftarrow S_h^{i,T} \cap \left(\omega_a^n \cup \omega_g^n\right)$
    else
      $S_h^{i,T} \leftarrow S_h^{i,T} \cap (\omega_a^n \cup \omega_v^n)$
    end if
  end for
end for
```

6.2.3 Coarse Grid Matrix and Parallel Setup

The pattern condition (6.13) for the interpolation matrix $\mathfrak{P}_h$ is fulfilled by construction in Alg. 12. Therefore, we can apply (6.14) with the distributed fine grid matrix K_h so that the coarse matrix K_H is again distributially stored and it can be calculated completely in parallel without any communication, i.e., we call

$$\mathsf{K}_{H,s} \leftarrow \textsc{Galerkin}(\mathsf{K}_s, \mathfrak{P}_s)$$

locally on each processor s.

The parallel setup (Alg. 13) collects the subroutines of Section 3.3. The coarsening part in Alg. 13 terminates if the global number of coarse nodes becomes smaller than CoarseGrid, cf. the sequential part.

6.2.4 Object Oriented Approach

The implementation of the parallel code exploits the C++ principles of overloading and inheritance. Given a class hierarchy with several types (e.g. SparseMatrix, BlockMatrix) derived from the base class (e.g. BaseMatrix), we introduce an additional 'parallel' class (e.g. ParallelMatrix) derived from the base class. The parallel class has a reference to some object of the base class and has access to additional 'parallel' data structures (see Fig. 6.3). The method GetObject() (e.g. GetMatrix()) of the base class returns the reference of the object itself. For the parallel class, this method returns the reference of the 'sequential' object. This con-

Algorithm 13 Parallel Setup ParSetup($\mathsf{K}_h, \omega, \ell$)

```
if $|\omega| >$ CoarseGrid then
  for all $s = 1, \dots, P$ do
    $\widetilde{\mathfrak{K}} \leftarrow \left(\sum_{i=1}^{P} A_i^T \widetilde{\mathsf{K}}_i A_i\right)_{\text{pattern}}$, $\widetilde{\mathfrak{K}}_s \leftarrow A_s \widetilde{\mathfrak{K}} A_s^T$
    $(\{N_h^i\}, \{S_h^{i,T}\}) \leftarrow$ GetStrong$(\widetilde{\mathfrak{K}}_s)$
    $\omega_C \leftarrow \emptyset$ , $\omega_F \leftarrow \emptyset$
    $(\{S_h^{i,T}\}, \omega_{C,s}, \omega_{F,s}) \leftarrow$ ParCoarse$(\{S_h^{i,T}\}, \omega_s^n)$
    $\mathfrak{P}_{h,s} \leftarrow$ Weights$(\left\{S_h^{i,T}\right\}, \widetilde{\mathfrak{K}}_s, \omega_{C,s}, \omega_{F,s})$
    $\mathsf{K}_{H,s} \leftarrow 0$
    $\mathsf{K}_{H,s} \leftarrow$ Galerkin$(\mathsf{K}_{h,s}, \mathfrak{P}_{h,s})$
    $\omega_{H,s}^n \leftarrow \omega_{C,s}$
    ParSetup$(\mathsf{K}_H, \omega_H^n, \ell + 1)$
  end for
else
  CoarseLevel $\leftarrow \ell$
end if
```

cept allows to switch easily between the sequential and parallel methods depending on the requirements of the algorithm. For example, the function

$$\text{GetDiag}(i)$$

returns the accumulated diagonal element of row i as required for the Jacobi iteration within the smoother. On the other hand,

$$\text{GetMatrix}().\text{GetDiag}(i)$$

accesses the diagonal element of the distributed matrix.

Using this approach, the changes in the sequential code are minimal as long as the existing code is well structured and global methods, including input and output, have already been implemented as methods of appropriate class variables. Of course, additional parallel-specific code sequences have to be added. This includes the interfaces to the parallel library DDComm [62] by M. Kuhn for the setup of the communication. In particular, access to the topology of the mesh has to be provided. Furthermore, the methods which are to be overloaded have to be supplied. In the example given above, this includes the initialization of the accumulated diagonal:

```
for(i=1; i<=n; i++)
     accumulated_diag.Elem(i) = matrix->Get(i,i);
accumulator.MultInline(accumulated_diag);
```

The diagonal is being accumulated using the methods of the Distributed Data Communication library DDComm [62].

```
// Class hierarchy:  BaseMatrix, SparseMatrix, ParallelMatrix
class BaseMatrix
      virtual double GetDiag(int i) const = 0;
      virtual BaseMatrix & GetMatrix() { return *this; };

class SparseMatrix ::  BaseMatrix
      virtual double GetDiag(int i) const {return Get(i,i); };

class ParallelMatrix ::  BaseMatrix
      BaseMatrix * matrix;
      Vector accumulated_diag;
      Accumulator * accumulator;
      ParallelMatrix(BaseMatrix * am) { matrix = am; };
      virtual double GetDiag(int i) const
           { return accumulated_diag.Get(i); };
      virtual BaseMatrix & GetMatrix() { return *matrix;};
```

Figure 6.3: Example for a class hierarchy of matrices in C++.

Chapter 7

AMG Software Package PEBBLES

7.1 Requirements and Implementation

The goal of the implementation of the AMG software package PEBBLES is to provide an efficient, robust and flexible solver kernel which additionally fulfills certain requirements. The main application areas are:

1. PEBBLES can be used as a coarse grid solver within a geometric MG code.
2. PEBBLES can be used as a solver in standard FE-codes, which do not support a hierarchical grid structure.

Since different FE-codes support different data structures for the system matrix, we need an interface such that the matrix is not stored twice. For that reason we decided to construct an interface which is based on element matrices and thus the matrix management is exclusively done in PEBBLES. This is illustrated in Fig. 7.1. Such an interface structure is possible for almost all FE-codes, because the assembling process simply has to be redirected. Finally, the solution vector is returned to the FE-code (see Fig. 7.1 for an illustration). Analogously an interface is defined for the right-hand side. In addition, the element based interface is essential for the element preconditioning technique (see Chapter 5). So far we have an interface structure for the matrix and right-hand data. Furthermore, PEBBLES provides the following routines:

1. *Essential boundary conditions*, i.e., Dirichlet conditions, can be easily defined. In addition, the updating of the values of the Dirichlet nodes is possible for time-dependent problems.
2. The definition of constraints is possible, i.e., one node is the *master node* and predefined *slave nodes* share the same unknown solution value. This is of special importance for electrostatic problems, e.g. Section 2.2.
3. The preconditioner can also be kept constant by PEBBLES for several outer iteration steps. This is the case for many different right-hand sides, time-dependent problems, or nonlinear problems. Furthermore PEBBLES can be

Figure 7.1: Information flow for an interface to PEBBLES.

instructed to construct a new preconditioner from the calling FE-code, in order to apply the strategy of Section 7.2.

4. In case of coupled field problems PEBBLES is able to support different matrices stemming from distinct FE-discretizations.
5. Finally, PEBBLES provides an interface for matrices stored in a given file format.

A more rigorous explanation of the interface routines and detailed examples are found in the 'PEBBLES - User's Guide' [74].

The implementation of PEBBLES has been done in C++ in order to use the concepts of overloading and inheritance. In addition to that we use sophisticated data structures and memory management to obtain a good performance. There are at least two objectives which have to be achieved:

1. The memory requirement is of optimal order, i.e., $O(N_h)$ as h tends to zero.
2. The number of necessary operations is of optimal order, i.e., $O(N_h \cdot \mathrm{NME}_h^p)$ with $1 < p \leq 2$. NME_h denotes the average number of nonzero entries per row of the system matrix K_h.

The first item can be reached in an easy way: We use a compact row storage of the required matrices, i.e., system and auxiliary matrix, and appropriate prolongation operators. By additionally defining appropriate data structures the memory requirement is of order $O(N_h \cdot \mathrm{NME}_h)$.

The second item can be achieved by a careful implementation of the setup phase, i.e., coarsening and construction of the coarse grid and prolongation operators. In addition the prolongation operators have to be constructed such that the number of nonzero entries per row NME_H on the coarse level does not grow too fast. Otherwise the numerical effort on the coarse level grows and the optimal complexity of the algorithm gets lost. For example, if

$$\text{NME}_H \sim N_H$$

then the complexity of the algorithm exhibits a complexity $O(N_H^p)$ with $p \geq 2$. Furthermore, the memory consumption would be non optimal, i.e., $O(N_H^2)$. The *operator complexity* , i.e.,

$$OC(K_h) = \frac{\sum_{k=1}^{\ell} \text{NME}_k \cdot N_k}{\text{NME}_1 \cdot N_1}\,, \tag{7.1}$$

with N_k, NME_k and ℓ are the number of unknowns, the average number of non zero entries per row on level k and the number of constructed levels, respectively, which gives a good idea of arithmetic costs and memory storage relative to the finest grid for the coarser grids. Another measure is the *grid complexity* , i.e.,

$$GC(K_h) = \frac{\sum_{k=1}^{\ell} M_k}{M_1}\,, \tag{7.2}$$

that presents how quick 'virtual' FE-meshes are reduced in size. M_k denotes the number of nodes on level k.

7.2 Setup Phase

7.2.1 General Relations

The aim of this section is to provide general tools for the construction of an (almost) optimal solution strategy for nonlinear, time-dependent and moving body problems (with respect to CPU-time). The three problem classes have in common that a sequence of SPD problems has to be solved in order to get the final solution, e.g. according to a fixed point iteration scheme of the linearized systems within nonlinear equations. Furthermore we can assume that the arising system matrices change little in the spectral sense during several outer iterations. A particular linear equation (2.53), i.e.,

$$K_h \underline{u}_h = \underline{f}_h\,,$$

can be solved with the PCG method. According to [40] it is enough to use an SPD preconditioner C_h in the PCG method in order to solve (2.53). In addition to this it is sufficient to assemble C_h from a spectrally equivalent matrix B_h with respect to K_h.

Let us make the theory more rigorous. Therefore let K_h be the current matrix of an outer iteration step and B_h be the matrix of a previous step. Furthermore let C_h be the constructed preconditioner for B_h (e.g. C_h is realized by k steps of a $V(\nu_F, \nu_B)$-cycle, $k = 1$ or $k = 2$). We can assume that K_h and B_h are SPD, and therewith spectrally equivalent with lower and upper constants $\underline{\alpha}$ and $\overline{\alpha}$, respectively. The first lemma shows that the preconditioner C_h is also spectrally equivalent to K_h, with a condition number bounded by $\alpha \cdot \kappa(C_h^{-1} B_h)$, $\alpha \geq 1$. This condition number is the basic convergence measure for the iterative methods given by Alg. 1 or Alg. 2, see Section 2.5.

Lemma 7.2.1. *Let K_h, $B_h \in \mathbb{R}^{N_h \times N_h}$ be SPD and let K_h be spectrally equivalent to B_h with spectral constants $\underline{\alpha}$ and $\overline{\alpha}$. Further let $C_h \in \mathbb{R}^{N_h \times N_h}$ be an SPD preconditioner for B_h with lower and upper spectral constants $\underline{\gamma}$ and $\overline{\gamma}$, respectively. Then the relations*

$$\underline{\alpha} \cdot \underline{\gamma} \cdot C_h \leq K_h \leq \overline{\gamma} \cdot \overline{\alpha} \cdot C_h$$

are valid and consequently

$$\kappa(C_h^{-1} K_h) \leq \kappa(C_h^{-1} B_h) \cdot \frac{\overline{\alpha}}{\underline{\alpha}} \, .$$

Proof. By assumption we know that

$$\underline{\alpha} \cdot B_h \leq K_h \leq \overline{\alpha} \cdot B_h$$

and by using the spectral equivalence inequality

$$\underline{\gamma} \cdot C_h \leq B_h \leq \overline{\gamma} \cdot C_h$$

we consequently obtain the desired result, i.e.,

$$\underline{\alpha} \cdot \underline{\gamma} \cdot C_h \leq K_h \leq \overline{\gamma} \cdot \overline{\alpha} \cdot C_h \, .$$

□

Remark 7.2.2.

1. *The underlying strategy is based on the PCG method (Alg. 2), where the error in the K_h-energy norm is given by (2.60). For an outer solution strategy we can assume that the iterative solution $\underline{u}_{i-1}^{l}$ of step $i-1$ is a good initial guess for the i^{th} outer iteration step (for an initial guess $\underline{u}_1^0 \equiv 0$ is usually taken).*

2. *The objective is to keep the overall numerical costs as small as possible. Therefore we try to keep the preconditioner C_h constant as long as possible (in order to keep the setup costs low), while the condition number $\kappa(C_h^{-1} K_h)$ is bounded by a given constant $\alpha_{\max} \cdot \kappa(C_h^{-1} B_h)$.*

3. *Equation (2.60) reflects the fact that the convergence rate of the PCG method deteriorates if $\kappa(C_h^{-1} K_h)$ becomes too large, i.e., $\kappa(C_h^{-1} K_h)$ has to be kept small. In practice we calculate the spectral constants $\overline{\alpha}$ and $\underline{\alpha}$ and we require*

$$\alpha = \frac{\overline{\alpha}}{\underline{\alpha}} \leq \alpha_{\max} \, ,$$

with $\alpha_{\max} \in (1, 10)$.

For further discussion we denote by i the outer iteration index, e.g. K_h^i is the system matrix in the i^{th} nonlinear step. The same is assumed for the other variables. In addition, we define the actual system matrix of the i^{th} step and a previous system matrix of the j^{th} step by

$$K_h = K_h^i \quad \text{and} \quad B_h = K_h^j \,,$$

respectively, with $j \leq i$. The general outer iteration algorithm Alg. 14 is used for further discussion. For instance, Alg. 14 can be seen as a fixed point iteration in the nonlinear case.

Algorithm 14 General outer iteration $\textsc{outer}(K_h^0, u_h^0, f_h^0)$

```
$\underline{f}_h \leftarrow \underline{f}_h^0$                                  get the right-hand side
$\underline{w}_h \leftarrow \underline{u}_h^0$                                  get the start solution
$K_h \leftarrow K_h^0(\underline{w}_h)$                               get the system matrix
$\underline{d}_h \leftarrow \underline{d}_h(\underline{f}_h, \underline{w}_h)$                          calculate the defect
$C_h \leftarrow C_h(K_h)$                               calculate a new preconditioner
while no convergence do
  $K_h \underline{w}_h = \underline{d}_h$                                 solve with PCG
  $\underline{u}_h \leftarrow \underline{u}_h(\underline{w}_h)$                             update solution
  $\underline{f}_h \leftarrow \underline{f}_h(\underline{u}_h)$                             update right-hand side
  $K_h \leftarrow K_h(\underline{u}_h)$                             update system matrix
  $\underline{d}_h \leftarrow \underline{d}_h(\underline{f}_h, \underline{u}_h)$                        calculate the defect
  $\alpha \leftarrow \frac{\overline{\alpha}}{\underline{\alpha}}$                                 calculate the spectral constants
  if $\alpha > \alpha_{\max}$ then
    $C_h \leftarrow C_h(K_h)$                           calculate a new preconditioner
  end if
end while
```

7.2.2 Special Examples

Nonlinear equation: Let us consider the nonlinear variational form

$$\text{find } u \in \mathbb{V} : \ a(\nu(w); u, v) = \langle f, v \rangle \quad \forall v \in \mathbb{V}$$

with $\nu(w)$ being a scalar, continuous nonlinearity, e.g.

$$w = \| \operatorname{grad} u \| \quad \text{and} \quad a(\nu(w); u, v) = \int_\Omega \nu(\| \operatorname{grad} u \|) \cdot \operatorname{grad} u \operatorname{grad} v \, dx \,.$$

The FE-discretization is done as usual and the subsequent assumptions are made for the discretization of $\nu(\cdot)$:

1. The discrete nonlinearity is abbreviated by ν_h^i in the i^{th} nonlinear step.

2. ν_h^i is piecewise constant, i.e., $\nu_h^{i,r} \equiv \mathit{const}$ on each element $r \in \tau_h$.

We know that the stiffness matrix can be expressed in terms of element stiffness matrices, i.e.,

$$K_h^i = \sum_{r \in \tau_h} A_r^T K_h^{i,r} A_r \,, \tag{7.3}$$

where $K_h^{i,r} \in \mathbb{R}^{n_r \times n_r}$ is the element stiffness matrix, and $A_r \in \mathbb{R}^{N_h \times n_r}$ is the element connectivity matrix of the finite element $r \in \tau_h$. Since ν_h^i has the special structure $\nu_h^{i,r} \equiv const$ on each $r \in \tau_h$ the element stiffness matrices have the special form

$$K_h^{i,r} = \nu_h^{i,r} \cdot K_h^r \quad \forall r \in \tau_h.$$

The spectral equivalence constants $\overline{\alpha}$ and $\underline{\alpha}$ are given by

$$\underline{\alpha} = \min\left\{ \min_{r \in \tau_h} \frac{\nu_h^{j,r}}{\nu_h^{i,r}}, 1 \right\} \quad \text{and} \quad \overline{\alpha} = \max\left\{ \max_{r \in \tau_h} \frac{\nu_h^{j,r}}{\nu_h^{i,r}}, 1 \right\}, \tag{7.4}$$

for $i \geq j$.

Remark 7.2.3. *A generalization to the case of a tensor valued $\nu(\cdot)$ is straightforward. For the discrete case we assume the matrices*

$$\nu_h^{i,r} = \begin{pmatrix} \nu_{11} & \nu_{12} & \nu_{13} \\ \nu_{21} & \nu_{22} & \nu_{23} \\ \nu_{31} & \nu_{32} & \nu_{33} \end{pmatrix} \qquad \forall r \in \tau_h$$

to be SPD. Consequently, the constants of (7.4) change to

$$\underline{\alpha} = \min\left\{ \min_{r \in \tau_h} \frac{\mu_{\min}(\nu_h^{j,r})}{\mu_{\min}(\nu_h^{i,r})}, 1 \right\} \quad \textit{and} \quad \overline{\alpha} = \max\left\{ \max_{r \in \tau_h} \frac{\mu_{\max}(\nu_h^{j,r})}{\mu_{\max}(\nu_h^{i,r})}, 1 \right\},$$

where $\mu_{\min}(\nu_h^{i,r})$ and $\mu_{\max}(\nu_h^{i,r})$ denote the minimal and the maximal eigenvalue of $\nu_h^{i,r}$, respectively.

Moving body problem: Next we look at a special moving body problem. Let us consider the variational form (2.48) on the domain given by Fig. 7.2. Further we assume inside the domain a unit which is driven by a rigid body mode for instance. This is a coupled field problem, see Subsection 2.2.3. Since only a few element system matrices change a little bit during one outer iteration step, we can assume that the global system matrix does not change essentially in the spectral sense. For instance, this can be seen as follows: Let us assume that

$$K_h = K_h^i = \sum_{r \in \tau_h} A_r^T K_h^{i,r} A_r \quad \text{and} \quad B_h = K_h^j = \sum_{r \in \tau_h} A_r^T K_h^{j,r} A_r$$

are the system matrices for the i^{th} an the j^{th} step, respectively. Since $K_h^{i,r}$ and $K_h^{j,r}$ are spectrally equivalent, the spectral constants are given by

$$\underline{\alpha} = \min_{r \in \tau_h} \{\mu_{\min}(J_r)\} \quad \text{and} \quad \overline{\alpha} = \max_{r \in \tau_h} \{\mu_{\max}(J_r)\} \,,$$

Figure 7.2: Sequence of a moving body problem.

with

$$J_r = \left(K_h^{j,r}\right)^{-1/2} K_h^{i,r} \left(K_h^{j,r}\right)^{-1/2} .$$

Time-dependent equation: This class of problems can often be written as

$$K_h^i = M_h + \nu(t, \Delta t^i) \cdot K_h$$

where M_h is a mass matrix, t describes the preceding time and Δt^i the increment of the i^{th} time-step. For instance

$$\nu(t, \Delta t^i) = \Delta t^i$$

for standard FE-discretizations of a parabolic PDE. With a straight forward calculation we obtain

$$\underline{\alpha} = \min\left\{1, \frac{\nu_j}{\nu_i}\right\} \quad \text{and} \quad \overline{\alpha} = \max\left\{1, \frac{\nu_j}{\nu_i}\right\}$$

for the spectral constants, with $\nu_l = \nu(t, \Delta t^l)$.

7.3 Applications

In order to test the proposed interface and the AMG implementation PEBBLES itself, our attempt was to use PEBBLES as a solver in various FE-codes. Before we present the FE-codes we mention that the current version is able to deal with the FE-discretizations proposed in Chapter 4. Since other discretization schemes yield system matrices with the same properties as for FE-discretizations, it is possible to apply PEBBLES even to more general SPD systems (e.g. SPD matrices stemming from a finite difference discretization). In the subsequent discussion we present four codes which are completely different from the task of applications and their implementations.

1. FEPP[1] [32] is an FE-code with a geometric MG solver. This FE-code is able to deal with Lagrange as well as Nédélec FE-discretizations. We use PEBBLES as a preconditioner for the solution of the arising system matrices. This code is mainly well suited for elliptic problems. Some results can be found in [37, 38, 75].

[1]Courtesy of J. Schöberl et al., University of Linz

2. CAPA[2] [65] is an FE-code written in Fortran 77 which is especially suited for coupled field problems. In this case we apply PEBBLES as a solver for the arising system matrices stemming from nonlinear, time-dependent or moving body problems and test the proposed 'setup-strategy' of Section 7.2. According to the coupled field problems we use the fact that PEBBLES can handle various system matrices. In addition, a lot of calculations can be found in [52, 53, 54, 56, 55].

3. CAUCHY[3] [91] is a tool-box which deals with the inverse source reconstruction of the current distribution in a human head. The kernel of CAUCHY is written in Fortran 77, whereas the memory management is done by a C++ implementation. We use PEBBLES as a solver tool for the solution of the required 10000 different right-hand sides for a linear static electric field equation. The obtained solutions are used for the inverse calculation. First calculations and comparisons with other solvers are presented in [92].

4. MAFIA[4] [19] is a code for solving the Maxwell equations based on the finite integration technique (FIT). Since the arising matrices are similar to an FE-discretization based on Nédélec FE-functions we are able to apply the method of Section 4.3 to these matrices. MAFIA is written in Fortran 77 and is able to deal with nonlinear and time-dependent problems. First calculations with PEBBLES are found in [18].

Several other matrices were tested, which stem from different fields of application. Moreover a parallelization of the sequential code PEBBLES was done using the domain decomposition communication library DDComm [62], see Chapter 6.

[2]Courtesy of M. Kaltenbacher et al., University of Erlangen
[3]Courtesy of C. Wolters et al., MPI Leipzig
[4]Courtesy of M. Clemens et al., University of Darmstadt

Chapter 8

Numerical Studies

In order to demonstrate the numerical efficiency of the proposed AMG methods we present several examples from natural and engineering sciences. Before the examples, corresponding to the variational forms of Chapter 2, are presented some abbreviations and notations are provided. The components of the AMG method are chosen as follows: The coarsening is done by Alg. 3 and the prolongation operator is defined for each particular example. Further we use a $V(1,1)$-cycle. The AMG method is applied as a preconditioner in the PCG method and the PCG iteration is stopped if the error in the $K_hC_h^{-1}K_h$-energy norm was reduced by a factor of ε (see Alg. 2). All calculations were done with the AMG software package PEBBLES described in Chapter 7. In order to assess the robustness and efficiency of PEBBLES, we compare our AMG with the geometric MG (represented by FEPP) and with the the IC preconditioned PCG for some examples.

All sequential calculations were done on an SGI Octane, 300 MHz, R12000 workstation, and the parallel computations were either made on an SGI ORIGIN 2000, 300 MHz, R12000 or on a PC LINUX cluster. For the rest of this chapter we use the following notations and abbreviations.

1. The preconditioners are denoted by
   (a) 'AMG': standard algebraic multigrid.
   (b) 'AMG with AUX': algebraic multgrid with auxiliary matrix.
   (c) 'AMG with EP': algebraic multigrid with element preconditioning.
   (d) 'IC': incomplete Cholesky without fill in and with zero threshold.
   (e) 'MG': geometric multigrid.
2. 'N_h' denotes the number of unknowns, i.e., DOFs.
3. 'solver' gives the CPU-time for the solution process in seconds.
4. 'setup' gives the CPU-time for the setup process in seconds.
5. 'solution' is the sum of 'solver' and 'setup' in seconds.
6. 'it' number of PCG iterations.

7. '$\kappa = \kappa(C_h^{-1} K_h)$' denotes the condition number of the preconditioned system.
8. '$OC = OC(K_h)$' is related to the operator complexity defined by (7.1).
9. '$GC = GC(K_h)$' is related to the grid complexity defined by (7.2).
10. For the nonlinear studies (realized by the fixed point iteration) we use additionally
    (a) 'nit': number of fixed point iterations.
    (b) 'sit': sum of required PCG iterations in the fixed point iteration.
    (c) 'precond': number of generated preconditioners during the fixed point iteration.
11. For the parallel computations we use
    (a) 'matrix': CPU-time for the parallel system matrix generation in seconds.
    (b) 'proc': number of used processors.
    (c) '1 it': speedup with respect to one iteration.

8.1 Symmetric $H^1(\Omega)$-based Equations

We begin with the simplest problem class for AMG, namely Poisson like problems related to the variational form (2.48). The success of AMG is well documented in [9, 13, 12, 14, 25, 57, 78, 81, 82, 83] for this problem class, however, we will show that our AMG methods are the favorite choice if anisotropic equations are considered.

8.1.1 Element Preconditioning Studies

The first and the second example (shielding and boundary layer problem) were calculated within FEPP up to a relative accuracy $\varepsilon = 10^{-8}$. Our purpose for these examples is to assess the condition number of the overall preconditioner using the element preconditioning technique discussed in Chapter 5. The third example (piezoelectric transducer) was calculated within CAPA up to a relative accuracy $\varepsilon = 10^{-5}$. We used the harmonic extension based prolongation (4.2) in all cases.

Shielding problem in 2D: Let us consider an isotropic, homogeneous magnetic field $\mathbf{B} = (B_x, B_y)$ in a given domain $\Omega \subset \mathbb{R}^2$ (e.g. bounded area of the earth) and no further source. In addition we place a shielding unit in Ω (see Fig. 8.1). This shielding object is assumed to be a hole square with thin wall thickness such that the diameter of the unit is large compared to the wall thickness. By introducing a scalar potential for the magnetic induction $\mathbf{B}$ in the Maxwell equations the problem is reduced to the variational form (2.48) with zero right-hand side and Dirichlet boundary conditions are prescribed (see e.g. [72]). The subsequent calculations are done with parameters $\mu_1 = 1$, $\mu_2 = 1000$, $B_x = 21$, $B_y = 45$ and domain $\Omega = (-6,6)^2 \subset \mathbb{R}^2$, $\Omega_2 = (-0.5, 0.5)^2 \setminus (-0.5 + \epsilon, 0.5 - \epsilon)^2$, as shown in Fig. 8.1. The shielding unit is anisotropic and hence we use long thin quadrilaterals for an FE-discretization with bilinear FE-functions. If we would discretize the shielding unit with a graded mesh, e.g. with triangles, then the number of unknowns would be much larger.

Table 8.1 displays the results for 3 different FE-discretizations and 3 different pre-

Figure 8.1: Sketch of the shielding problem in 2D with two materials.

conditioners. It can be seen that 'AMG with EP' improves the condition number κ considerably and κ appears to be unaffected by the problem size N_h and the wall thickness ϵ. Due to the anisotropy ϵ the condition numbers of the other proposed preconditioners grow with the number of unknowns N_h and the anisotropy ϵ. Let us further mention that the operator complexity OC is slightly worse for 'AMG with EP' compared to the standard AMG, however, the entire matrix hierarchy can be stored in just over three times the storage for the fine grid matrix. The grid complexity GC hovers between 1.47 and 1.52 for both methods, which indicates a reduction of unknowns of approximately 2 of two consecutive levels. Because of the classifica-

		AMG with EP			AMG			MG
ϵ	N_h	κ	OC	GC	κ	OC	GC	κ
10^{-1}		10.51	2.11	1.47	59.26	2.54	1.51	25.42
10^{-3}	9345	10.23	2.14	1.47	1009.13	2.59	1.52	56258.9
10^{-5}		7.28	2.14	1.47	202095	2.59	1.52	1.9e+7
10^{-1}		11.91	2.27	1.49	170.83	2.60	1.51	24.14
10^{-3}	37121	12.21	2.84	1.51	834.23	2.78	1.52	54174.4
10^{-5}		8.88	2.83	1.51	-	2.75	1.51	479076
10^{-1}		12.19	2.34	1.50	230.63	2.69	1.51	26.33
10^{-3}	147969	13.84	3.18	1.52	123491	2.84	1.52	53051
10^{-5}		11.57	3.16	1.52	-	2.84	1.52	874875

Table 8.1: Shielding problem in 2D for different thickness parameters and DOFs.

tion strategy the numerical work for the element preconditioning part is negligible for the shielding problem. A spectrally equivalent matrix $B^{(r)}$ in the set Z_{n_r} had to be only calculated for about 16 element stiffness matrices $K^{(r)}$.

Boundary layer problem in 2D: A problem related to the variational form (2.48) is the bilinear form

$$\int_\Omega \epsilon^2 \operatorname{grad} u \operatorname{grad} v \, dx + \int_\Omega uv \, dx = \int_\Omega 1v \, dx \,,$$

where a mass term is added to (2.48) and homogeneous Dirichlet boundary conditions are assumed. We consider a domain $\Omega = (0,1)^2$ and a discretization as depicted in Fig. 8.2, where the discretization is known to be optimal for this kind of problem. Again, if we use a graded mesh for resolving the boundary layer then the number of unknowns is very large. The FE-discretization is done with bilinear FE-functions.
In Tab. 8.2 the numerical robustness of the element preconditioning technique

Figure 8.2: Sketch of the boundary layer problem in 2D.

is presented. The condition number κ for 'AMG with EP' does neither depend significantly on the number of unknowns N_h nor on the anisotropy ϵ, whereas if we use standard AMG or geometric MG, then κ grows as ϵ gets small. For this example the operator complexity OC is not satisfactory small. Indeed, for larger problem sizes the operator complexity gets worse if the anisotropy ϵ tends to zero. In addition the grid complexity GC rises, which indicates a non sufficient decrease of the unknowns. The examination of these numbers reveals a high application time and a large amount of memory requirement. In the case of the boundary layer problem the numerical work for the element preconditioning method consists in constructing 3 different spectrally equivalent matrices and is therefore negligible.

Piezoelectric Transducer: Finally, the numerical calculation of the electric field within a piezoelectric transducer (material PZT 5A) with partial electrodes is studied. This is a 3D problem where the element preconditioning technique is applied. The electrodes of the transducer are squares with length 10 mm and have a distance of 0.5 mm (for a detailed discussion see e.g. [53]). The FE-discretization has been performed by brick elements (Fig. 8.3) and the maximal ratio of the

		AMG with EP			AMG			MG
ϵ	N_h	κ	OC	GC	κ	OC	GC	κ
10^{-1}		7.83	1.64	1.36	11.21	1.66	1.37	13.51
10^{-3}	2401	7.96	2.51	1.64	41.67	2.40	1.67	47.94
10^{-5}		4.68	2.50	1.64	28.12	2.38	1.66	42.86
10^{-1}		8.90	2.42	1.52	17.91	2.82	1.57	18.69
10^{-3}	9409	11.38	4.41	1.88	178.49	3.66	1.92	203.89
10^{-5}		7.16	4.48	1.89	99.96	3.54	1.88	164.66
10^{-1}		8.95	2.54	1.52	24.71	3.30	1.59	20.84
10^{-3}	37249	11.20	8.54	2.68	632.86	7.73	2.80	829.17
10^{-5}		10.78	8.38	2.67	145.52	6.83	2.75	573.99

Table 8.2: Boundary layer problem in 2D for different anisotropic parameters and DOFs.

longest to the shortest side of a brick was approximately 50. The top electrode

Figure 8.3: FE-discretization of the piezoelectric transducer.

of the piezoelectric actuator has been loaded by a voltage. Table 8.3 presents the CPU-time of the overall solution time and the number of PCG iterations for both, the 'AMG with EP' and the AMG preconditioner as a function of unknowns for a relative accuracy of $\varepsilon = 10^{-5}$. It can be seen that 'AMG with EP' has a much better performance than the standard AMG method. In addition this examples requires approximately 100 spectrally equivalent matrices to be calculated which is again negligible with respect to the overall computation time.

	AMG with EP		AMG	
N_h	it	solution	it	solution
18965	18	5.75	44	12.07
67088	21	23.95	42	43.59
147930	28	69.42	44	102.49

Table 8.3: Total CPU-times (in seconds) and number of iterations of the piezoelectric transducer.

We note that the element preconditioning technique appears to work well in 2D as well as for 3D. This conclusion is fairly predictable since the element preconditioning method is based on pure algebraic information of the element matrices.

8.1.2 Anisotropic Studies with an Auxiliary Matrix

So far the element preconditioning technique was used in order to get a robust preconditioner with respect to the considered anisotropy. Subsequently we study the properties of an AMG method which is explicitly constructed with an auxiliary matrix. We consider the variational form (2.48) in 2D and 3D and apply the average based prolongation (4.1). The numerical studies were done within PEBBLES up to a relative accuracy $\varepsilon = 10^{-8}$.

We start with the 2D problem (2.48) with $\Omega = (0,1)^2 \subset \mathbb{R}^2$ and the material tensor

$$D = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} ,$$

where ϵ will be varied with three different values. We assume homogeneous Dirichlet and Neumann boundary conditions on $[0,1] \times \{0\}$ and $\partial\Omega \setminus [0,1] \times \{0\}$, respectively. The FE-discretization was done with bilinear FE-functions.

The arising linear equation is solved on one hand by the new method based on an auxiliary matrix constructed via the method given in Example 3.2.2, and on the other hand by the standard AMG method. The results for both are displayed in Tab. 8.4. It can be seen that the performance of the auxiliary matrix based method is much better than the standard AMG method. In addition, the new method is relatively insensitive with respect to the anisotropic parameter ϵ, whereas the standard method fails in the sense of the required PCG iterations for all cases $\epsilon \leq 10^{-2}$. Both AMG methods works well if the anisotropy is moderate ($\epsilon = 10^{-1}$). We detect minor variations in the operator and grid complexity for both methods, but the operator complexity is significantly less for the auxiliary based method than for the standard approach. This in turn implies a fast application of the 'AMG with AUX' preconditioner. The amount of work and memory requirement entailed for the auxiliary matrix is of optimal order, but has to be regarded. However, the auxiliary matrix based method pays off for anisotropic equations which are aligned with the grid and

		AMG with AUX					AMG				
N_h	ϵ	it	setup	solver	OC	GC	it	setup	solver	OC	GC
10201		20	1.08	1.12	1.90	1.91	30	1.26	2.01	3.11	1.79
40401	10^{-1}	21	4.32	5.63	1.91	1.92	38	5.55	12.36	3.14	1.79
90601		33	9.78	20.83	1.91	1.92	47	12.20	34.75	3.16	1.80
10201		20	1.14	1.16	1.96	1.97	206	1.26	13.79	3.11	1.79
40401	10^{-2}	19	4.41	5.21	1.98	1.99	248	5.57	80.05	3.13	1.79
90601		19	10.20	12.25	1.99	2.00	269	12.22	198.56	3.16	1.80
10201		25	1.13	1.45	1.96	1.99	442	1.26	29.81	3.11	1.79
40401	10^{-3}	26	4.41	7.18	1.97	1.99	-	5.57	-	3.14	1.79
90601		25	10.22	16.19	1.98	2.00	-	12.22	-	3.16	1.80

Table 8.4: Number of iterations, CPU-times (in seconds), grid- and operator complexity for the anisotropic problem in 2D.

seems to be quit effective.

Turning our attention to the 3D case with $\Omega \subset \mathbb{R}^3$ be an L-shaped domain and the material tensor given by

$$D = \begin{pmatrix} 2+\epsilon & -\epsilon & 2-\epsilon \\ -\epsilon & 2+\epsilon & -2+\epsilon \\ 2-\epsilon & -2+\epsilon & 4+\epsilon \end{pmatrix} .$$

We assume homogeneous Dirichlet boundary conditions on $\partial\Omega$. Since we use a tetrahedra FE-mesh with linear FE-functions we are faced with a problem featuring a non-aligned anisotropic operator. For the smoother we use an overlapping patch Gauss-Seidel smoother with maximal patch size 10 instead the standard Gauss-Seidel method. The smoother is applied in a $V(2,2)$-cycle. The auxiliary matrix is constructed as suggested in Example 3.2.2.
The results for different ϵ are given in Tab. 8.5. The performance of both methods are similar compared to each other. The reason therefore is that they detect approximately the same strong connections, i.e., the nodes which are potential candidates for prolongation. In this case, where the anisotropy is not aligned with the grid, an optimal solver is hard to realize.

8.1.3 Nonlinear Studies

The next series of problems deals with nonlinear version of the variational form (2.48) where we assume a nonlinear, scalar material parameter, i.e.,

$$\nu = \nu(\|\operatorname{grad} u\|) .$$

The nonlinear (electrostatic and magnetostatic) equations are solved via a fixed point method, see e.g. [93] within CAPA. The outer nonlinear iteration stopped if

		AMG			AMG with AUX		
N_h	ϵ	it	setup	solver	it	setup	solver
2025		5	0.37	0.70	5	0.53	0.72
14161	10^{-3}	9	3.45	15.28	9	4.09	14.55
105633		16	32.65	242.19	14	35.53	195.67
2025		5	0.35	0.69	5	0.50	0.75
14161	10^{0}	8	3.32	12.49	8	4.05	12.47
105633		11	28.92	156.09	10	34.52	137.66
2025		6	0.41	0.81	7	0.63	1.29
14161	10^{+3}	11	3.58	18.77	13	5.15	24.68
105633		19	32.99	291.36	23	43.18	372.02

Table 8.5: Number of iterations and CPU-times (in seconds) for the anisotropic problem in 3D.

the relative error of the residuum

$$r(\underline{u}_h^i) = f_h - K_h(\underline{u}_h^i)$$

is smaller than 10^{-4} and the PCG iteration in each nonlinear step terminated if the error was reduced by a factor of $\varepsilon = 10^{-5}$. All calculations have been performed by using the element preconditioning method (see Chapter 5). Comparisons are made for the required setup and solver CPU-times in the cases of the AMG and IC preconditioner. In addition the suggested 'setup control' of Section 7.2 is studied.

Electrostatic: The first numerical study is due to the nonlinear version of the piezoelectric transducer (Fig. 8.3) with the nonlinearity given in Fig. 8.4. In the

Figure 8.4: Nonlinear function of the relative permittivity.

nonlinear case the top electrode is loaded by an electric charge. To define an equipotential, surface constraints are applied to the FE-nodes within the area of the top electrode (cf. Fig. 2.2).

The performance of AMG for the linear case was already reported in Subsection 8.1.1. The nonlinear case reflects the results of the linear case, which is illustrated in Tab. 8.6 and Tab. 8.7.

			AMG with EP		IC	
N_h	nit	precond	sit	solution	sit	solution
18964	5	5	111	33.57	271	35.28
67088	5	5	125	136.39	418	197.89
147930	5	5	142	348.11	536	565.42

Table 8.6: Electrostatic: nonlinear case with $\alpha_{\max} = 1$ and CPU-times in seconds.

Regarding the performance of the preconditioners with the 'setup control' of Section 7.2 we observe that the best results are obtained for $\alpha_{\max} = 2$, but it does not save much CPU-time. This is also displayed in Tab. 8.6 and Tab. 8.7. Thus the 'setup control' does not significantly gain speedup, which can be explained by the moderate nonlinearity.

			AMG with EP		IC	
N_h	nit	precond	sit	solution	sit	solution
18964	5	1	119	30.77	324	37.38
67088	5	2	125	124.28	419	187.42
147930	5	2	151	338.55	541	546.34

Table 8.7: Electrostatic: nonlinear case with $\alpha_{\max} = 2$ and CPU-times in seconds.

Magnetostatic: In the magnetostatic case a magnetic assembly with an air-gap, driven by a permanent magnet, has been considered (Fig. 8.5). As for the piezoelectric transducer, brick elements are used for the FE-discretization. The considered nonlinearity is given in Fig. 8.6. Compared to the nonlinearity of the piezoelectric transducer this one is more difficult to handle. As we will see later, the number of nonlinear iterations is larger than for the electrostatic case. In the linear case the performance of the AMG preconditioner and the IC preconditioner is shown in Tab. 8.8. Up to about 50000 unknowns the IC-PCG is faster than the AMG-PCG. However, practical important discretizations consists of up to 10^6 number of unknowns. The results of the nonlinear case are given in Tab 8.9, Tab 8.10 and Tab 8.11. It is important to notice that the fastest strategy for the nonlinear case is given for $\alpha_{\max} = 2$. Thus by applying the 'setup control' we definitely save CPU-time in the overall computation.

Figure 8.5: FE-discretization of the magnetic circuit

	AMG with EP		IC	
N_h	it	solution	it	solution
9102	29	3.86	42	2.64
65846	35	37.03	82	37.74
149625	40	97.00	112	118.85

Table 8.8: Magnetostatic: Linear case and CPU-times in seconds.

			AMG with EP		IC	
N_h	nit	precond	sit	solution	sit	solution
9102	14	14	309	43.05	575	35.01
65846	15	15	503	525.15	1194	544.31
149625	18	18	705	1683.18	1944	2021.15

Table 8.9: Magnetostatic: nonlinear case with $\alpha_{\max} = 1$ and CPU-times in seconds.

			AMG with EP		IC	
N_h	nit	precond	sit	solution	sit	solution
9102	14	5	315	39.73	598	32.96
65846	15	5	498	478.98	1237	532.32
149625	18	5	706	1570.17	2053	2046.85

Table 8.10: Magnetostatic: nonlinear case with $\alpha_{\max} = 2$ and CPU-times in seconds.

Figure 8.6: Nonlinear function of the permeability.

			AMG with EP		IC	
N_h	nit	precond	sit	solution	sit	solution
9102	14	4	336	41.58	610	34.10
65846	15	4	524	500.49	1294	558.90
149625	18	4	747	1638.70	2264	2249.87

Table 8.11: Magnetostatic: nonlinear case with $\alpha_{\max} = 4$ and CPU-times in seconds.

8.1.4 Coupled Field Problems

Concerning coupled field problems of Subsection 2.2.3 we apply AMG exclusively as a solver for the magnetic or electric field equation. Since the magnetic system matrices changes permanently during the outer iteration, while the mechanical (acoustic) system matrix remains constant, it is advisable to solve the magnetic (electric) part by AMG with an appropriate 'setup control' and the mechanical (acoustic) part via a direct method. The subsequent computations were carried out with CAPA up to a relative accuracy of $\varepsilon = 10^{-4}$ for the loudspeaker, and $\varepsilon = 10^{-6}$ for the electrostatic driven bar. For both examples the element preconditioning technique was applied.

The loudspeaker: We start with the electrodynamic loudspeaker (see Fig. 8.7). The computational domain can be reduced to an axisymmetric problem. The discretization of the computational domain was done by rectangles, with maximal ratio of the longest to the shortest side of about 20 (see Fig. 8.8). The number of unknowns for the magnetic, mechanical and acoustic parts are 14738, 31071 and 154191, respectively. Quadratic FE functions are used. We have to perform 5000 time steps

Figure 8.7: The principle structure of the electrodynamic loudspeaker.

Figure 8.8: The FE-discretization of the electrodynamic loudspeaker.

for the transient analysis of the electrodynamic loudspeaker (one time step 20 ns) in order to get the desired accuracy. The AMG preconditioner of the first time step is kept constant throughout all 5000 time steps and applied in the PCG iteration. For a detailed discussion on the solution process see [52]. The amount of work entailed for the element preconditioning method is negligible since we only have to produce about 190 spectrally equivalent element stiffness matrices. The AMG preconditioner appears to work well for this problem, although quadratic FE-functions are used.

Table 8.12 presents the CPU-time for the electrodynamic loudspeaker for one time step for computations with 'AMG with EP' and the standard AMG. It can be seen that the CPU-time for the setup phase as well for the solution phase is essentially less for 'AMG with EP' than for the standard AMG.

Electrostatic driven bar: Next, we consider the coupled field problem of a typical voltage driven micromechanical bar (see e.g. [50]) with a length of 1 mm, thickness of 1 μm and an air-gap of 3 μm (distance between the two electrodes). The devices are operated near the snap in point in order to gain a maximum on

	it	setup	solver
AMG with EP	68	2.38	6.46
AMG	77	13.42	11.40

Table 8.12: CPU-time (seconds) for the electrodynamic loudspeaker.

efficiency. The electrostatic transducer has been loaded by 0.1 V and the mechanical deformation of the bar has been calculated. The FE-discretization was done with hexahedra elements. Again, the mechanical part is solved via a direct method, whereas the electric field is solved by the AMG method. Table 8.13 displays the results for the electric field computation performed by the AMG solver of one single nonlinear iteration. The results of the coupled field computation required 6 nonlinear

N_h	setup	solver
25505	1.8	3.8
50355	3.5	7.2
127368	10.7	25.3

Table 8.13: Electrostatic driven bar: CPU-times of the AMG-PCG for setup and solution phase in seconds.

iterations. The total CPU-time for different discretizations is shown in Tab. 8.14.

N_h	solution
25505	42.1
50355	103.9
127368	240.7

Table 8.14: Electrostatic driven bar: Total CPU-time in seconds.

8.1.5 Inverse Source Reconstruction

One application in inverse source reconstruction is used in medical applications in order to reconstruct current distributions in the human brain by knowing magnetic and electric field measurements around the head [90]. This is actually an inverse problem. However, in order to recover the current distribution, we have to solve several forward problems of (2.48) with different right-hand sides. The engineers assume Neumann boundary conditions and they fix one node with zero potential (reference electrode). A realistic head model has about $N_h \sim 300000$ DOFs and approximately 10000 different right-hand sides which are in general linear independent. Additionally, the material tensor of the brain is anisotropic. The challenging task is to solve this system for all given right-hand sides within 6 hours in order to make it applicable to clinical use. The calculations were done within the inverse source reconstruction program CAUCHY.

The first tests[1] with a single processor machine are displayed in Fig. 8.9. Therein

Figure 8.9: Comparison of different solvers.

the AMG method is compared with standard preconditioners like IC. Up to a relative accuracy of $\varepsilon = 10^{-3}$ all preconditioners are quite similar with respect to CPU-time. For a relative accuracy $\varepsilon < 10^{-3}$ the AMG method appears to work much better than the other preconditioners. The experiments were repeated for several different FE-meshes, number of unknowns and brain anisotropies. AMG maintains robust and efficient for all test cases and is the favorite choice in that application. However, in order to solve the whole problem setting in the prescribed CPU-time of 6 hours, the problem with about 10000 right-hand sides has to be solved by a parallel version. In Fig. 8.10 a typical solution of a given dipole in the brain is shown.

Figure 8.10: Typical solution of a dipole in the brain.

[1]Courtesy of A. Basermann (NEC), C. Wolters (MPI Leipzig) and the Simbio Project.

8.2 Symmetric $(H^1(\Omega))^p$-based Equations

8.2.1 Linear Elasticity Problems

The robust and efficient solution of SPD matrices arising from an FE-discretization of the variational form (2.49) is a challenging task for AMG. The reason therefore is the nontrivial kernel of the corresponding operator which can not be approximated well enough yet. However, we present some numerical studies for the new AMG method based on an auxiliary matrix and compare it to standard AMG. We restrict our studies to the cantilever beam in 2D and to the crank shaft in 3D. For both examples we assume a Poisson ratio 0.3 and a standard FE-discretization. The related calculations were done within FEPP up to a relative accuracy of $\varepsilon = 10^{-8}$. A $V(2,2)$-cycle is used in the AMG preconditioners with a block Gauss-Seidel smoother.

For the cantilever beam we assume an FE-discretization with bilinear FE-functions on a uniform rectangular grid with ratio $\epsilon : 1$. Additionally, homogeneous Dirichlet boundary conditions are assumed on one side of the beam and free boundary conditions on the rest of the boundary, see Fig 8.11. The parameter ϵ is employed with

Figure 8.11: Cantilever Beam in 2D.

two different values. First, we consider the isotropic case for $\epsilon = 1$. The results are presented in Tab. 8.15. The performance of the standard AMG with harmonic extension based prolongation (4.5) performs best, whereas the average based prolongation (4.4) performs comparably well. There are only minor variations in the number of iterations and CPU-times in Tab. 8.15. The results for the cantilever

	AMG with AUX[1]			AMG[1]			AMG[2]		
N_h	it	setup	solver	it	setup	solver	it	setup	solver
20402	15	1.79	3.67	12	1.11	3.00	11	1.28	2.77
80802	21	7.21	21.57	14	4.48	14.61	11	5.11	11.64
181202	22	16.29	51.29	15	9.96	35.40	12	11.44	28.70

Table 8.15: CPU-times (seconds) and number of iterations for the cantilever beam in 2D with $\epsilon = 1$. [1]prolongation (4.4), [2]prolongation (4.4).

beam with $\epsilon = 10^{-1}$ are displayed in Tab. 8.16. In this case the standard AMG method with harmonic extension based prolongation operator is much better than

	AMG with AUX[1]			AMG[2]		
N_h	it	setup	solver	it	setup	solver
20402	113	7.00	54.74	50	1.58	15.68
80802	109	30.80	224.27	65	6.26	87.71
181202	140	71.57	654.36	81	13.57	244.43

Table 8.16: CPU-times (seconds) and number of iterations for the cantilever beam in 2D with $\epsilon = 10^{-1}$. [1]prolongation (4.4), [2] prolongation (4.5).

the new AMG method. Heuristically, this can be explained as follows: While the average based prolongation only preserves constant functions, the harmonic extension based prolongation is able to approximate more than constant functions. Since the constant functions are only a subspace of the current kernel we can neither expect an optimal preconditioner for the harmonic extension prolongation nor for the average based prolongation. This is the reason that the efficiency generally degrades for this problem class.

We continue this subsection with a 3D crank shaft with geometry given in Fig. 8.12 and the input data is the same as for the cantilever beam, except the Dirichlet boundary data that is prescribed on the hole surface for simplicity. This experiment is run for an uniform tetrahedra FE-mesh. The results are listed in

Figure 8.12: Crank Shaft in 3D.

Tab. 8.17 and show that both methods perform comparably well. As for the cantilever beam the standard AMG method with the harmonic extension based prolongation is the best method in that comparison. The above considerations on the prolongation operators apply also in the 3D case.

	AMG with AUX[1]			AMG[1]			AMG[2]		
N_h	it	setup	solver	it	setup	solver	it	setup	solver
3039	8	2.45	0.89	8	0.63	0.82	7	0.67	0.73
17769	16	12.98	10.63	16	6.18	10.63	11	2.18	7.38
118359	23	110.0	118.86	23	13.52	118.75	19	17.53	96.83

Table 8.17: CPU-times (seconds) and number of iterations for the crank shaft in 3D. [1]prolongation (4.4), [2]prolongation (4.5).

8.2.2 Magnetic Field Equations in 3D with Nodal Elements

Let us turn our attention to the FE-discretization of the variational form (2.50) with Lagrange FE-functions. In particular we consider TEAM Problem 20 and 27 and perform a full numerical simulation of them. Therefore we solve the nonlinear equation up to a relative error of the residual $r(\underline{u}_h^i)$ of 10^{-3} and the PCG iteration stopped if the error was reduced by a factor of $\varepsilon = 10^{-5}$. For comparison the IC preconditioner is presented. The following computations are done within CAPA.

TEAM Problem 20: TEAM 20 defines a 3D nonlinear magnetostatic field problem consisting of a yoke with center pole and a coil for excitation (see Fig. 8.13). In a first step we performed a linear analysis using different meshes consisting of

Figure 8.13: TEAM 20: FE-mesh of yoke, center pole and coil.

hexahedra and tetrahedra finite elements. In Tab. 8.18 the CPU-times for the setup and solver as well as the number of PCG-iterations for the AMG-PCG solver are displayed.

N_h	it	setup	solver
91260	37	34.3	78.2
136620	53	46.3	163.8
175329	59	55.6	229.9
241368	49	81.9	269.4

Table 8.18: CPU-times (seconds) and number of iterations of the AMG-PCG solver in the linear case for TEAM 20.

The comparison of the AMG- and the IC-PCG solver is given in Tab. 8.19 and shows the strong dependences of the IC-PCG method on the number of unknowns.

	AMG		IC	
N_h	it	solution	it	solution
91260	37	112.5	81	184.8
136620	53	210.1	112	353.6
175329	59	285.5	122	487.1
241368	49	351.3	150	795.9

Table 8.19: Comparison of AMG- and IC-PCG method in the linear case for TEAM 20. CPU-times in seconds.

By using the 'setup control' proposed in Section 7.2 the total CPU-time for the nonlinear computation can be reduced. It is important to notice, that the fastest strategy for the nonlinear case is given for $\alpha_{\max} = 4$.

$\alpha_{\max}$	nit	setups	solution	sit
1	18	18	6736	913
2	18	13	6279	959
4	18	4	5832	1009
8	18	3	6193	1213

Table 8.20: Nonlinear case on the finest mesh ($N_h = 241368$) and 4000 ampere-turns excitation for TEAM 20. CPU-times in seconds.

The comparison of measured and simulated magnetic induction is given in Tab. 8.21.

	B_z (P_2)	
ampere-turns	simulated	measured
1000 A	23.7 mT	24.0 mT
3000 A	65.8 mT	63.0 mT
4500 A	73.6 mT	72.0 mT
5000 A	75.3 mT	74.0 mT

Table 8.21: Measured and simulated magnetic induction of TEAM 20 at location P_2 (12.5 mm,5 mm,25.75 mm).

TEAM Problem 27: TEAM 27 defines a 3D linear eddy current problem, where a deep flaw in an aluminum cylinder leads to an asymmetry in the magnetic field. This asymmetry is detected by the difference signal of the two Hall Effect Sensors (HES1 and HES2), which measure the horizontal flux density. The principle setup is shown in Fig. 8.14. The begin of the measurement is the current turn-off in the coil. For the simulations a time step of $10\,\mu$s has been used.

Figure 8.14: TEAM 27: a) Coil with Hall effect sensors (HEF1 and HEF2) and aluminum cylinder b) Detail of aluminum cylinder to see the flaw.

In Tab. 8.22 the CPU-times as well as the number of PCG iterations of the AMG-PCG solver are displayed for different fine grids. Since the performance (i.e., the CPU-time) is quite constant for all time steps, the results of just one time step are shown.

N_h	it	setup	solver
110432	56	37.0	180.3
174824	46	62.2	240.1
311360	60	110.9	559.5

Table 8.22: Performance of the AMG-PCG solver for one time step. CPU-times in seconds.

In Tab. 8.23 the comparison between the AMG- and IC-PCG solver is displayed.

	AMG		IC	
N_h	it	solution	it	solution
110432	56	217.3	60	291.3
174824	46	302.3	66	490.5
311360	60	670.4	75	1233.8

Table 8.23: Comparison of AMG- and IC-PCG method for one time step. CPU-times in seconds.

As in the 3D nonlinear magnetostatic case, the efficiency of the AMG-PCG solver is demonstrated. In Fig. 8.15 the measured and the simulated data are compared. It

Figure 8.15: Measured and simulated data of the HDFD.

has to be noticed, that this problem is very sensitive to the quality of the FE-mesh, since the measured quantity (flux density) is very small. Any asymmetry in the mesh leads to additional differences between the flux densities of the HEFs, which is of course a pure numerical effect.

The results of TEAM 20 in Tab. 8.18 and TEAM 27 in Tab. 8.22 indicate that the domain configuration generally has again little affect on AMG behavior while the structure of the mesh is more important. This explains the minor variations in the number of iterations.

8.3 Symmetric $H(\text{curl}, \Omega)$-based Equations

8.3.1 Magnetic Field Equations in 3D with Edge Elements

Let us consider the variational form (2.51) and an appropriate edge FE-discretization. Our purpose here is to assess the method proposed in Section 4.3 for that particular problem class. Regarding the AMG performance we can not expect an optimal solver since the used prolongation is non optimal. The subsequent calculations were done within FEPP.

Unit cube: We start with a fairly simply example, namely the unit cube with an uniform tetrahedra FE-mesh. The relative error is given by $\varepsilon = 10^{-6}$. Every row of Tab. 8.24 and Tab. 8.25 consists of two sub-rows. The first one is directed to a $V(2,2)$-cycle and the second one to a generalized V-cycle with 2^s smoothing steps on level s ($s = 1, \dots, \ell$ and $s = 1$ is the finest level). On the boundary $\partial\Omega$ of Ω Dirichlet boundary conditions are prescribed. We use the parameter setting $\nu = 1$ and $\sigma = 10^{-4}$ in Ω.

Although the system matrix is SPD we show that all suggested ingredients of the new AMG method are necessary. Therefore we solve this problem with $N_h = 4184$ with different preconditioners. The results depicted in Fig. 8.16 reveal that only the new AMG method appears to be an appropriate preconditioner. It can be seen that

Figure 8.16: Comparison of different solvers.

the standard AMG, which was constructed for $H^1(\Omega)$-based problems fails as well as the IC preconditioner in the sense of robustness. An interesting observation is that an AMG method with the correct prolongation operator (defined in Chapter 4), but with a point Gauss-Seidel smoother fails as well.

Let us continue this study by testing the two suggested smoothers. Results are displayed in Tab. 8.24 for the smoothing iteration proposed by D. Arnold, R. Falk, R. Winther in [3] and in Tab. 8.25 for the smoother proposed by R. Hiptmair in [42]. In both cases a slight increase can be detected in the number of iterations with respect to the number of unknowns. This might be an effect of the designed prolongation operator. As it can be expected the generalized V-cycle performs better for both smoothers compared to the $V(2,2)$-cycle. Actually there are only minor variations with respect to CPU-time of the used smoothers.

Magnetic valve: Let us turn our attention to an example from engineering. We consider the magnetic valve as it is depicted in Fig. 8.17. The simulation of the magnetic valve requires a nonlinear, time dependent study, but for test reasons we

N_h	it	setup	solver
4184	9	0.24	0.23
	9		0.27
31024	14	1.98	4.76
	12		4.56
238688	20	16.83	58.15
	16		53.59

Table 8.24: CPU-times (seconds) and number of iterations for the unit cube with the smoother of Arnold, Falk, Winther.

N_h	it	setup	solver
4184	12	0.15	0.30
	11		0.31
31024	17	1.32	6.29
	15		6.21
238688	21	11.39	67.98
	17		63.93

Table 8.25: CPU-times (seconds) and number of iterations for the unit cube with the smoother of Hiptmair.

are simply considering one linear step to show the performance of AMG. The given FE-mesh contains a lot of flat tetrahedra due to the small airgaps of the magnetic valve, which in turn explains the relative large iteration numbers in Tab. 8.26. The results are computed for the relative accuracy of $\varepsilon = 10^{-8}$ and the $V(2,2)$-cycle with the Arnold, Falk, Winther smoother.

N_h	it	setup	solver
8714	14	0.53	2.49
65219	28	4.17	44.01
504246	63	34.49	792.49

Table 8.26: CPU-times (seconds) and number of iterations for the magnetic valve with the smoother of Arnold, Falk, Winther.

The key observation is that the suggested AMG method proposed here can deal with system matrices stemming from an edge FE-discretization without too much degradation in efficiency. This carries over to the next discretization scheme.

8.3.2 FIT Discretization

Another discretization method for the Maxwell equations is the so called FIT (finite integration technique) scheme (see [89, 18]). We don't want to go into detail for

Figure 8.17: The magnetic valve without surrounding air.

this discretization scheme, but we test the AMG preconditioner for the arising SPD matrices. Therefore we choose the smoother of D. Arnold, R. Falk, R. Winther and a $V(1,1)$-cycle. The calculations were done within MAFIA on a SUN Enterprise 3500, 336MHz. Further the AMG preconditioner is compared to a highly tuned SSOR preconditioner which takes care of the special tensor product mesh. The relative accuracy is given by $\varepsilon = 10^{-8}$.

The C-magnet (see Fig. 8.18) is taken as an example. The whole simulation requires a time dependent calculation, but for test reasons only one linear time step is regarded. The results[2] are depicted in Fig. 8.19 and it can be seen that AMG

Figure 8.18: The C-magnet without surrounding air.

shows an almost optimal CPU-time behavior. The AMG preconditioner is not the fastest method, but this might be due to a non optimal C++ compiler. Nevertheless, the cross-over can be extrapolated from Fig. 8.19.

[2]Courtesy of T. Weiland and M. Clemens (University of Darmstadt).

Figure 8.19: Results for different solvers for the C-magnet.

8.4 Parallel Results

The following examples were computed within FEPP and make extensive use of the Distributed Data Communication Library DDComm [62] by M. Kuhn, i.e., PEBBLES and FEPP contain parallel versions which are realized by DDComm. A partition of a given FE-mesh is computed by the modified recursive spectral bisection method with interface smoothing [31]. This algorithm produces an almost uniform distribution of the elements to the subdomains for any number of subdomains. We run the experiments on an SGI ORIGIN 2000 with R12000, 300MHz processors and overall 20 GB of main memory. We investigate the speedup for 1 up to 32 processors. MPI is used as message passing library. In a second run we repeat some experiments on a PC cluster with 4 double processor machines with INTEL Pentium 500 MHz processors and at least 256 KB of main memory. The machines are running under SUSE LINUX 6.3 and are connected via fast Ethernet. The LAM-distribution of MPI is used for the PC cluster.

8.4.1 Symmetric $H^1(\Omega)$-based Equations

Let us consider the variational form (2.48) with Dirichlet boundary conditions on the domain of a simplified crank shaft (see Fig. 8.12). We solve the resulting matrix equation by the AMG-PCG solver (Alg. 10 together with Alg. 11). In particular, we apply AMG with element preconditioning (see Chapter 5). The CPU-time required for generating spectrally equivalent M-(element-)matrices is contained in the matrix generation CPU-time given below. The prolongation operator is given by the average based prolongation (4.1) and we apply a $V(2,2)$-cycle in one preconditioning step. The smoother is a block Jacobi smoother with Gauss-Seidel smoothing in blocks containing interior unknowns of the subdomains. We choose a relative accuracy of $\varepsilon = 10^{-4}$ as stopping criterion.

Table 8.27 shows the number of iterations, the wall clock-time (measured by MPI_WTIME) required for the AMG setup, matrix generation and solver for a problem size of 3227648 elements and 576833 global unknowns. The CPU-time for partitioning as well as for the communication setup is not included. Since both,

proc	it	matrix	setup	solver	solution
1	31	211.9	184.2	383.4	779.3
2	33	106.5	94.5	223.7	424.7
4	33	51.0	50.4	111.5	219.9
8	24	24.6	22.5	29.3	76.4
16	25	12.6	12.2	13.3	38.1
24	25	9.0	10.2	9.5	28.7
32	26	7.3	8.4	7.0	22.7

Table 8.27: Number of iterations and CPU-times (seconds) for the crank shaft on the SGI 2000.

coarsening and smoother depend on the decomposition into subdomains, we observe that the number of iterations depends on the number of processors. That is the reason for considering the speedup with respect to 1 iteration. The corresponding speedup results are found in Tab. 8.28. The matrix generation is purely local and

proc	1it	matrix	setup	solver	solution
1	1.0	1.0	1.0	1.0	1.0
2	1.8	2.0	1.9	1.7	1.8
4	3.7	4.1	3.6	3.4	3.5
8	10.1	8.6	8.2	13.1	10.2
16	23.2	16.8	15.1	28.8	20.4
24	32.5	23.5	18.0	40.3	27.2
32	45.9	29.0	21.9	54.8	34.3

Table 8.28: Speedups for one iteration of the considered crank shaft.

gives the reference value for the quasi optimal speedup. Looking at the speedup of the setup phase, we observe some kind of saturation effects which are due to the increasing communication effort. In contrast to the solver, these communication steps have to be synchronized. Hence, increasing the number of processors for a fixed problem size increases the sequential overhead. However, in many applications, e.g. transient or nonlinear problems, a fixed setup can be used for repeated calls of the solver. Then the speedups for the solver alone are of interest. Here we observe optimal speedups. Moreover we profit from cache effects since the local problem size decreases.

Now we repeat our experiments on a PC cluster. Because of memory limitations we reduce the problem size down to 403456 elements and 77153 nodes. The results are presented in the same way as above in Tab. 8.29. We consider speedup for 1 up to 8 processors in Tab. 8.30. As expected we observe lower efficiency compared to the computations on the ORIGIN 2000. However, the PC cluster caused almost

proc	it	matrix	setup	solver	solution
1	23	20.0	10.4	24.0	54.4
2	23	9.9	5.4	12.0	27.3
4	23	5.0	3.4	6.4	14.8
6	22	3.4	2.5	4.9	10.8
8	18	2.5	2.3	3.4	8.2

Table 8.29: Number of iterations and CPU-times (seconds) for the crank shaft on a PC cluster.

proc	1it	matrix	setup	solver	solution
1	1.0	1.0	1.0	1.0	1.0
2	2.0	2.0	1.9	2.0	2.0
4	3.7	4.0	3.0	3.7	3.7
6	4.7	5.9	4.1	4.9	5.0
8	5.5	8.0	4.5	7.1	6.6

Table 8.30: Speedups for one iteration for the crank shaft on a PC cluster.

no additional costs for the network since the standard fast Ethernet has been used. Table 8.31 shows the overall CPU-times on the ORIGIN 2000 and the PC cluster

proc	ORIGIN 2000	PC Cluster
1	59.6	54.4
2	31.1	27.3
4	15.6	14.8
8	7.1	8.2

Table 8.31: Total CPU-times (seconds) on the ORIGIN 2000 and the PC Cluster.

for the same (smaller) problem size. For very few processors the faster processors of the PC cluster give lower computation time in comparison with the ORIGIN 2000. If the effort for communications grows with the number of processors, the ORIGIN 2000 outperforms the PC Cluster due to the superior network.

8.4.2 Symmetric $H(\mathrm{curl}, \Omega)$-based Equations

In principle the parallelization of edge FE-discretizations can be done as for the nodal FE-discretization, but special attention has been taken to the smoother. Again a $V(2,2)$-cycle is used and we use a relative accuracy of $\varepsilon = 10^{-6}$ as stopping criterion. The artificial conductivity is $\sigma = 10^{-6}$ in all cases.

Figure 8.20 shows the geometry and the mesh of a C-magnet being considered now. The problem has 180426 unknowns and a material parameter $\nu \equiv 1$. The field is generated by a current in the coil. In order to analyze the parallel performance we again consider the following components of the algorithm: generation of the

Figure 8.20: C-Magnet: geometry and mesh with 153408 elements and 206549 nodes.

system matrix, setup and solver, see Tab. 8.32 and Tab. 8.33 for the absolute CPU-time and for the speedup, respectively. In addition the parallel versions of AMG and geometrical MG are compared. In the case of geometric MG the setup phase involves the setup of the smoother and the LU-decomposition of the coarse-grid system matrix with 2907 unknowns. Since the setup is dominated by the sequential LU-decomposition, only low speedups can be observed. In the case of AMG the setup involves the coarsening itself and the LU-decomposition of the coarse-grid system which has less than 500 unknowns. Hence, the setup shows reasonable speedups since it is dominated by the coarsening of the inner nodes. In both cases (AMG and geometric MG) the solver shows a similar speedup behavior. Although most of the gain is due to the additional CPU capacity, the additional cache which comes with each processor also contributes to the acceleration. A closer look at Tab. 8.32 shows that whereas the solver CPU-time is only 2 times larger for AMG, the number of iterations is approximately 4 times larger for AMG compared with geometric MG. There might be several reasons for this effect (e.g. different implementation of the AMG and the geometric MG code, different complexities of the coarse grid systems). However, the main reason is that the prolongation operator is very simple in the case of AMG, which explains the fast application. But this prolongation suffer from the non-optimality. On the other hand, we have an optimal prolongation operator for the geometric MG, which in turn is more expensive.

		AMG			MG		
proc	matrix	it	setup	solver	it	setup	solver
1	40.3	60	44.4	247.3	13	13.1	99.4
2	19.6	52	24.4	127.1	13	10.8	69.2
4	9.7	52	12.6	48.9	13	9.1	31.8
8	4.7	58	7.6	28.2	13	8.8	16.2

Table 8.32: Comparing AMG and MG for edge FE-discretization in the parallel case. CPU-times in seconds.

		AMG			MG	
proc	matrix	1it	setup	solver	setup	solver
1	1	1	1.0	1.0	1.0	1.0
2	2	1.7	1.8	1.9	1.2	1.4
4	4.1	4.3	3.5	5.0	1.4	3.1
8	8.5	8.4	5.8	8.8	1.5	6.1

Table 8.33: Speedups for the C-magnet: AMG and MG for edge FE-discretization in the parallel case.

8.5 Other Comparisons

Since our attempts are closely related to the Maxwell equations we present two further studies. Therefore we assume the geometry of TEAM 20 (see Subsection 8.2.2).

In a first test, a pure tetrahedra FE-mesh and edge FE-functions are used for the bilinear form (2.51). The geometric MG as well as the AMG preconditioner has been applied for the solution of the algebraic system. We use a $V(2,2)$-cycle for the AMG method and a $V(1,1)$-cycle for the geometric MG method. The iteration was stopped if an error reduction has been achieved by a factor $\varepsilon = 10^{-6}$. The comparison is shown in Tab. 8.34 for two different FE-meshes. Because of the optimal prolongation

	AMG		MG	
N_h	it	solution	it	solution
108552	26	74.82	8	37.4
237893	32	188.55	10	89.6

Table 8.34: Comparison of geometric MG and AMG solver for one nonlinear iteration with edge FE-functions of TEAM 20. CPU-times in seconds.

operator for the MG method the number of iterations stays essentially constant while for the AMG method the number of iterations grows again. It is worth to mention that in both cases (nodal and edge FE-discretizations) good agreement between measured and simulated data has been achieved as shown in Tab. 8.35. In addition AMG appears as an efficient preconditioner for both FE-discretizations. This leads us to our final studies. We compare the AMG method itself with the two proposed FE-discretizations of variational forms (2.50) and (2.51) on the same FE-mesh, i.e., we use nodal (Lagrange) and edge (Nédélec) FE-functions for the discretization, respectively. In Tab. 8.36 the results for 3 different FE-meshes and one nonlinear iteration step are presented. As expected, we observe that the number of iterations are slightly growing in both cases. This is due to the non-optimal prolongation operators, mentioned several times before. Nevertheless the AMG preconditioner performs well for both FE-discretizations.

	B_z (P_2)		
ampere-turns	nodal	edge	measured
1000 A	23 mT	24 mT	24 mT
3000 A	65 mT	58 mT	63 mT
4500 A	73 mT	66 mT	72 mT
5000 A	75 mT	70 mT	74 mT

Table 8.35: TEAM 20: Measured and simulated results of the magnetic induction at location P_2 (12.5 mm,5 mm,25.75 mm).

nodal				edge			
N_h	it	setup	solver	N_h	it	setup	solver
1114	8	0.10	0.18	2018	11	0.10	0.30
7203	15	0.80	3.30	14408	17	0.87	5.72
50427	27	7.97	48.34	108552	26	7.02	67.80

Table 8.36: TEAM 20: Number of iterations and CPU-times (seconds) for AMG-PCG with node and edge FE-discretization.

Bibliography

[1] R. A. Adams, *Sobolev Spaces*, Pure and Applied Mathematics, Academic Press, New York, 1975.

[2] N. R. Aluru and J. White, *Direct Newton finite-element/boundary-element technique for micro-electro-mechanical-analysis*, Solid-State Sensor and Actuator Workshop (Hilton Head, South Carolina), 1996, pp. 54 – 57.

[3] D. Arnold, R. Falk, and R. Winther, *Multigrid in H(div) and H(curl)*, Numer. Math. **85** (2000), 197–218.

[4] O. Axelsson, *Iterative solution methods*, second ed., Cambridge University Press, 1996.

[5] R. E. Bank and T. F. Dupont, *Analysis of a two-level scheme for solving finite element equations*, Report CNA-159, Center for Numerical Analysis, University of Texas at Austin, May 1980.

[6] R. Beck, *Algebraic multigrid by component splitting for edge elements on simplicial triangulations*, Preprint SC 99-40, Konrad Zuse Zentrum für Informationstechnik Berlin, December 1999.

[7] R. Beck, P. Deuflhard, R. Hiptmair, R. Hoppe, and B. Wohlmuth, *Adaptive multilevel methods for edge element discretizations of Maxwell's equations*, Surveys Math. Indust. **8** (1999), 271–312.

[8] O. Biro, *Numerische Aspekte von Potentialformulierungen in der Elektrodynamik*, Habilitation, 1992.

[9] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing **55** (1995), 379–393.

[10] ______, *Finite Elemente - Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*, second ed., Springer Verlag, Berlin, Heidelberg, New York, 1997.

[11] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput. **19** (1986), 23–56.

[12] A. Brandt, S. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations*, Report, Inst. Comp. Studies Colorado State Univ., 1982.

[13] ______, *Algebraic multigrid (AMG) for sparse matrix equations*, Sparsity and its Application (D.J. Evans, ed.), 1984, pp. 257–284.

[14] M. Brezina, A.J. Cleary, R.B. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge, *Algebraic multigrid based on element interpolation AMGe*, SIAM J. Sci. Comput. **22** (2000), no. 5, 1570–1592.

[15] W. L. Briggs, V. E. Henson, and S. McCormick, *A multigrid tutorial*, second ed., SIAM, 2000.

[16] P. G. Ciarlet, *The finite element method for elliptic problems*, North-Holland Publishing Company, Amsterdam, New York, Oxford, 1987.

[17] ______, *Mathematical Elasticity: Three-Dimensional Elasticity*, Studies in Mathematics and Its Applications, vol. 20, North-Holland Publishing Company, Amsterdam-New York-Oxford-Tokyo, 1994.

[18] M. Clemens, S. Drobny, M. Wilke, and T. Weiland, *Numerical algorithms for the calculation of magneto-quasistatic fields using the finite integration technique*, Proceedings of SCEE 2000, Lecture Notes in Computational Science and Engineering, accepted for publication.

[19] M. Clemens, P. Thoma, U. van Rienen, and T. Weiland, *A survey on the computational electromagnetic field compuation with the FI-method*, Surveys on Mathematics for Industry **8** (1999), no. 3–4, 213–232.

[20] P. M. de Zeeuw, *Matrix dependent prolongations and restrictions in a black box multigrid*, J. Comput. Appl. Math. **33** (1990), 1–27.

[21] J. Dendy, *Blackbox multigrid for nonsymmetric problems*, Appl. Math. Comput. **48** (1983), 366–386.

[22] R. Falgout, Van E. Henson, J. E. Jones, and U. Meier Yang, *Boomer AMG: A parallel implementation of algebraic multigrid*, Techn. Report UCRL-MI-133583, Lawrence Livermore National Laboratory, March 1999.

[23] J. Fuhrmann, *A modular algebraic multilevel method*, WIAS-Preprint 203, Weierstraß Institute for Applied Analysis and Stochastics, Berlin, 1996.

[24] V. Girault and P. A. Raviart, *Finite Element Methods for the Navier–Stokes Equations - Theory and Algorithms*, Springer Verlag, Berlin, Heidelberg, New York, 1986.

[25] T. Grauschopf, M. Griebel, and H. Regler, *Additive multilevel preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs*, SFB Bericht 342/02/96 A, Technische Universität München, Institut für Informatik, 1996.

[26] M. Griebel, T. Neunhöffer, and H. Regler, *Algebraic multigrid methods for the solution of the Navier-Stokes equations in complicated geometries*, Internat. J. Numer. Methods Fluids **26** (1998), no. 3, 281–301.

[27] C. Großmann and H.-G. Roos, *Numerik partieller Differentialgleichungen*, Teubner Studienbücher Mathematik, 1994.

[28] G. Haase, *Parallel incomplete Cholesky preconditioners based on the non-overlapping data distribution*, Parallel Computing **24** (1998), no. 11, 1685–1703.

[29] ______, *Parallelisierung numerischer Algorithmen für partielle Differentialgleichungen*, BG Teubner, Stuttgart, 1999.

[30] ______, *A parallel AMG for overlapping and non-overlapping domain decomposition*, Electronic Transactions on Numerical Analysis (ETNA) **10** (2000), 41–55.

[31] G. Haase and M. Kuhn, *Preprocessing for 2D FE-BE domain decomposition methods*, Comp. Vis. Sci. **2** (1999), 25–35.

[32] G. Haase, M. Kuhn, and U. Langer, *Parallel multigrid 3D Maxwell solvers*, Proceedings of the 12^{th} International Conference on Domain Decomposition in Chiba, October 1999 (H. Kawarada T. Chan, T. Kako and O. Pironneau, eds.), DDM.ORG, accepted for publication.

[33] ______, *Parallel multigrid 3D Maxwell solvers*, Parallel Comput. (2000), accepted for publication.

[34] G. Haase, M. Kuhn, U. Langer, S. Reitzinger, and J. Schöberl, *Parallel Maxwell solvers*, Proceedings of SCEE 2000, Lecture Notes in Computational Science and Engineering, accepted for publication.

[35] G. Haase, M. Kuhn, and S. Reitzinger, *Parallel AMG on distributed memory computers*, Tech. Report 00-16, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing", 2000.

[36] G. Haase, U. Langer, and A. Meyer, *The approximate Dirichlet decomposition method. part I,II*, Computing **47** (1991), 137–167.

[37] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl, *A general approach to algebraic multigrid*, Tech. Report 00-33, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing", 2000.

[38] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl, *Algebraic multigrid methods based on element preconditioning*, International Journal of Computer Mathematics **80** (2001), no. 3–4, accepted for publication.

[39] W. Hackbusch, *Multigrid methods and application*, Springer Verlag, Berlin, Heidelberg, New York, 1985.

[40] ______, *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, BG Teubner, Stuttgart, 1991.

[41] V.E. Henson and P.S. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, Tech. report, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2000, submitted.

[42] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal. **36** (1999), no. 1, 204–225.

[43] T. J. R. Hughes, *The finite element method*, New Jersey, Prentice-Hall, 1987.

[44] N. Ida and P.A. Bastos, *Electromagnetics and calculation of fields*, Springer, 1997.

[45] J.E. Jones and P.S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comp. (2000), accepted for publication.

[46] M. Jung, *On the parallelization of multi-grid methods using a non-overlapping domain decomposition data structure*, Appl. Numer. Math. **23** (1997), no. 1, 119–137.

[47] M. Jung and U. Langer, *Applications of multilevel methods to practical problems*, Surveys Math. Indust. **1** (1991), 217–257.

[48] M. Jung, U. Langer, A. Meyer, W. Queck, and M. Schneider, *Multigrid preconditioners and their application*, Proceedings of the 3rd GDR Multigrid Seminar held at Biesenthal, Karl-Weierstraß-Institut für Mathematik, May 1989, pp. 11–52.

[49] M. Kaltenbacher, H. Landes, R. Lerch, and F. Lindinger, *A finite-element/boundary-element method for the simulation of coupled electrostatic-mechanical systems*, J. Phys. III France **7** (1997), 1975–1982.

[50] M. Kaltenbacher, H. Landes, K. Niederer, and R. Lerch, *3D simulation of controlled micromachined capacitive ultrasound transducers*, Proceedings of the IEEE Ultrasonic Symposium 1999 (Lake Tahoe), accepted for publication.

[51] M. Kaltenbacher, H. Landes, S. Reitzinger, and M. Schinnerl, *Multigrid methods for the efficient computation of electromagnetic fields*, Proceedings CADFEM Meeting (Friedrichshafen), 2000.

[52] M. Kaltenbacher and S. Reitzinger, *Algebraic multigrid for solving electromechanical problems*, Multigrid Methods VI (E. Dick, K. Rienslagh, and J. Vierendeels, eds.), Lecture Notes in Computational Science and Engineering, vol. 14, 2000, pp. 129–135.

[53] ______, *Algebraic multigrid for static nonlinear 3D electromagnetic field computations*, Tech. Report 00-07, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing", 2000.

[54] ______, *Nonlinear 3D magnetic field computations using Lagrange FE-functions and algebraic multigrid*, IEEE Trans. on Magn. (2000), submitted.

[55] M. Kaltenbacher, S. Reitzinger, M. Schinnerl, J. Schöberl, and H. Landes, *Multigrid methods for the computation of 3D electromagnetic field problems*, COMPEL 20, 2001, accepted for publication.

[56] M. Kaltenbacher, S. Reitzinger, and J. Schöberl, *Algebraic multigrid for solving 3D nonlinear electrostatic and magnetostatic field problems*, IEEE Trans. on Magnetics **36** (2000), no. 4, 1561–1564.

[57] F. Kickinger, *Algebraic multigrid for discrete elliptic second-order problems*, Multigrid Methods V. Proceedings of the 5th European Multigrid conference (W. Hackbusch, ed.), Springer Lecture Notes in Computational Science and Engineering, vol. 3, 1998, pp. 157–172.

[58] F. Kickinger and U.Langer, *A note on the global extraction element-by-element method*, ZAMM (1998), no. 78, 965–966.

[59] A. Kost, *Numerische Methoden in der Berechnung elektromagnetischer Felder*, Springer Verlag, Berlin, Heidelberg, New York, 1995.

[60] A. Krechel and K. Stüben, *Parallel algebraic multigrid based on subdomain blocking*, Parallel Comput. (2000), submitted.

[61] E. Kreyszig, *Introductory functional analysis with applications*, John Wiley & Sons, 1978.

[62] M. Kuhn, *DDCom - Distributed Data Communication library*, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 2000, www.sfb013.uni-linz.ac.at.

[63] M. Kuhn, U. Langer, and J. Schöberl, *Scientific computing tools for 3D magnetic field problems*, The Mathematics of Finite Elements and Applications X (J. R. Whiteman, ed.), Elsevier, 2000, pp. 239 – 258.

[64] R. Lerch, M. Kaltenbacher, H. Landes, and F. Lindinger, *Computerunterstützte Entwicklung elektromechanischer Transducer*, e&i (1996), no. 7/8, 532 –545.

[65] R. Lerch, H. Landes, and M. Kaltenbacher, *CAPA User's Guide*, Lehrstuhl für Sensorik, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999, www.lse.e-technik.uni-erlangen.de/CAPA/index.htm.

[66] J. Mandel, M. Brezina, and P. Vanek, *Energy optimization of algebraic multigrid bases*, Computing (2000), accepted for publication.

[67] S. F. McCormick, *An algebraic interpretation of multigrid methods*, SIAM J. Numer. Anal. **19** (1982), no. 3, 548–560.

[68] G. Meurant, *Computer solution of large linear systems*, Studies in Mathematics and its Applications, vol. 28, Elsevier, 1999.

[69] J. Nédélec, *A new family of mixed finite elements in R^3*, Numer. Math. **50** (1986), 57–81.

[70] Y. Notay, *A robust algebraic preconditioner for finite difference approximations of convection-diffusion equations*, Report GANMN 99-01, Service de Métrologie Nucléaire, Université Libre de Bruxelles, 1999.

[71] S. Reitzinger, *Algebraic mutigrid and element preconditioning I*, Tech. Report 98-15, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 1998.

[72] ______, *Robust algebraic multigrid methods in magnetic shielding problems*, Master's thesis, Johannes Kepler University Linz, 1998.

[73] ______, *Algebraic mutigrid and element preconditioning II*, Tech. Report 99-18, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 1999.

[74] ______, *PEBBLES - User's Guide*, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 1999, www.sfb013.uni-linz.ac.at.

[75] S. Reitzinger and J. Schöberl, *Algebraic multigrid for edge elements*, Tech. Report 00-15, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 2000, submitted.

[76] A. Reusken, *On the approximate cyclic reduction preconditioner*, Bericht 144, Rheinisch-Westfälische Technische Hochschule Aachen, Institut für Geometrie und Praktische Mathematik, 1997.

[77] J. W. Ruge and K. Stüben, *Efficient solution of finite difference and finite element equations*, Multigrid Methods for integral and differential equations (D. Paddon and H. Holstein, eds.), 3, Clarendon Press, Oxford, 1985, pp. 169–212.

[78] ______, *Algebraic multigrid (AMG)*, Multigrid Methods (S. McCormick, ed.), Frontiers in Applied Mathematics, vol. 5, SIAM, Philadelphia, 1986, pp. 73–130.

[79] M. Schinnerl, J. Schöberl, and M. Kaltenbacher, *Nested multigrid methods for the fast numerical computation of 3D magnetic fields*, IEEE Trans. on Magnetics, 2000, submitted.

[80] K. Schittkowski, *The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function. Part 1: Convergence analysis*, Numer. Math. **38** (1981), 83–114.

[81] K. Stüben, *Algebraic multigrid (AMG): Experiences and comparisions*, Appl. Math. Comput. **13** (1983), 419–452.

[82] ______, *Algebraic multigrid: An introduction with applications*, Report 53, GMD, 1999.

[83] ______, *A review of algebraic multigrid*, Report 69, GMD, 1999.

[84] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing **56** (1996), 179–196.

[85] G. Wachutka, *Tailored modeling of miniaturized electrothermomechanical systems using thermodynamic methods*, Micromechanical Systems **40** (1992), 183 – 198.

[86] C. Wagner, *Introduction to algebraic multigrid*, Course Notes of an Algebraic Multigrid course at the University Heidelberg, 1999, http://www.iwr.uni-heidelberg.de/∼Christian.Wagner.

[87] ______, *On the algebraic construction of multilevel transfer operators*, Computing **65** (2000), 73–95.

[88] W.L. Wan, T.F. Chan, and B.Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput. (2000), to appear.

[89] T. Weiland, *A discretization method for the solution of Maxwell's equations for six-component fields*, Electronics and Communications AEÜ **31** (1977), no. 3, 116–120.

[90] C. Wolters, *Direkte Methoden zur Berechnung dipolinduzierter elektrischer und magnetischer Felder und inverse Strategien zur Quellokalisaton im Gehirn*, Master's thesis, Technische Hochschule Aachen, 1997.

[91] ______, *Cauchy*, Max-Planck-Institute Leipzig, 1999.

[92] C. Wolters, S. Reitzinger, A. Basermann, S. Burkhardt, F. Kruggl, and A.Anwander, *Improved tissue modelling and fast solver methods for high resolution FE-modelling in EEG/MEG-source localization*, Biomag 2000, Proceedings of the 12th International Conference on Biomagnetism (J. Nenonen, R.J. Ilmoniemi, and T. Katila, eds.), Helsinki University of Technology, Espoo, 2000, accepted for publication.

[93] O. C. Zienkiewicz and R.L. Taylor, *The finite element method*, 4 ed., vol. 2, McGraw-Hill, 1991.

Index

Ich, Dipl.-Ing. Stefan Reitzinger, geboren am 6. Dezember 1971 in Steyr, versichere, daß ich die vorliegende Dissertation selbständig verfaßt habe und keine anderen Quellen als die im Literaturverzeichnis angegebenen verwendet habe. Weiters erkläre ich, daß ich die vorliegende Dissertation bisher an keiner inländischen oder ausländischen Universität vorgelegt habe.

Linz, im Jänner 2001

(Dipl.-Ing. Stefan Reitzinger)

Curriculum Vitae

Name: Stefan Reitzinger

Date and Place of Birth: December 6, 1971, Steyr

Nationality: Austria

Home Address: Schudutz 38, 3350 Haag, Austria

Affiliation: Institute of Analysis and Computational Mathematics
Johannes Kepler University Linz
Altenbergerstr. 69, A-4040 Linz, Austria
www-Page: http://www.sfb013.uni-linz.ac.at/∼reitz

Title: Dipl.-Ing.

Education:

1978 - 1986:	Primary school in Haag
1986 - 1991:	High School for Agricultural Engineering in Wieselburg
1991:	High School Diploma
1992 - 1998:	Studies in Mathematics (Industrial Mathematics) at the Johannes Kepler University Linz
1997:	Foreign Semester at the Technical University of Denmark, Lyngby
1998:	Graduated to Dipl.-Ing., Johannes Kepler University Linz

Employment:

1991 - 1992:	Military service
Since 1998:	Research Assistant at the Institute of Analysis and Computational Mathematics, Johannes Kepler University Linz, currently funded via FWF Project SFB F013

Selected Project Experience:

1998 - :	Multigrid algorithms, especially algebraic multigrid methods and applications (SFB subproject F1306)