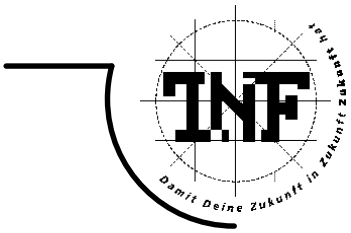
JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis




# Efficient Algebraic Multigrid Preconditioners for Boundary Element Matrices

## Diplomarbeit

zur Erlangung des akademischen Grades

## Diplomingenieur

in der Studienrichtung

## Technische Mathematik

Angefertigt am *Institut für Numerische Mathematik*

Betreuung:

*O.Univ.Prof. Dipl.-Ing. Dr. U. Langer*

Eingereicht von:

*David Pusch*

Linz, Jänner 2003

# Zusammenfassung

Multigrid Algorithmen sind zur Behandlung von großen Gleichungssytemen, die aus der Diskretisierung von partiellen Differentialgleichungen zweiter Ordnung stammen, nicht mehr wegzudenken. Eingesetzt als Vorkonditionierer für iterative Gleichungslöser, wie zum Beispiel das konjugierte Gradientenverfahren (PCG), gehören sie zu den effizientesten Lösungsalgorithmen. Die zugrunde liegende Idee ist die seperate Behandlung von hoch- bzw. niedrigfrequenten Eigenfunktionen im Iterationsfehler. Realisiert wird die Idee im geometrischen Multigridverfahren (GMG) durch das Vorhandensein von mehreren Gitterebenen und im algebraischen Multigridverfahren (AMG) durch das rein algebraische Aufstellen einer passenden Matrixhierarchie.

Die Verwendung von Multigrid Methoden wurde hauptsächlich zur Auflösung von Differenzenschemata und zur Lösung von Finite-Elemente-Gleichungen entwickelt. Darum sind die wesentlichen Bestandteile wie Glätter oder Prolongationsoperatoren für diese Problemklasse ausgelegt. Eine alternative Diskretisierungsmethode zur Finiten-Elemente-Methode (FEM) stellt die Randelemente Methode (Boundary Element Method, BEM) dar, bei der wir zwar vollbesetzte Matrizen erhalten, aber nur der Gebietsrand diskretisiert werden muss. Wendet man nun AMG auf BEM-Matrizen an müssen einige Adaptionen vorgenommen werden. Genauer gesagt arbeiten im Fall des sogenannten Einfachschichtpotentials die herkömmlichen Glätter nicht mehr optimal, da für diesen Operator Eigenwerte und Eigenfunktionen konträres Verhalten zu denen einer FEM-Matrix zeigen.

Um überhaupt einen effizienten iterativen Löser für BEM-Gleichungen zu erhalten, muss man sich darüber Gedanken machen, wie man eine schnelle Matrix-Vektor Multiplikation für unsere vorerst vollbesetzten Matrizen sicherstellt. In den letzten Jahren wurden einige Verfahren zur Matrix Approximation entwickelt. Wir verwenden die 'Adaptive Cross Approximation' Methode (ACA), bei der die BEM-Matrix durch Niedrigrang Matrizen angenähert wird. Damit kann eine schnelle Matrix Multiplikation gewährleistet und damit der Einsatz des AMG vorkonditionierten iterativen Löser mit fast optimaler Komplexität durchgeführt werden.

Einige erste Versuche bestätigen die guten Vorkonditionierungeigenschaften mittels des AMG Verfahrens. Sowohl für symmetrische Galerkin-Matrizen als auch für nichtsymmetrische Kollokations-Matrizen erhält man schnelle Konvergenzraten bzw. kleine Iterationszahlen.

Für echte BEM-FEM Kopplungsaufgaben wurde der AMG Löser vorerst nur im FEM Teil eines kommerziellen Gleichungssystemlösers eingebaut. Auch hier zeigen erste Versuche sehr gute Resultate. In weiterer Folge könnte ein Multigrid basierter Löser konstruiert werden, der auf das gesamte gekoppelte Gleichungssystem angewendet werden kann. Die Voraussetzungen für beide Arten von Matrizen sind nunmehr im Wesentlichen gegeben.

# Abstract

Multigrid methods are a widely used solving strategy for large-scale equation systems arising from the finite difference and finite element discretization of partial differential equations of second order. Most efficiency will be reached, if they are applied as preconditioners, e.g. for the conjugate gradient method (PCG). The fundamental idea of the multigrid approach is the separate treatment of high frequency and low frequency eigenfunctions of the iteration error. In the case of geometric multigrid methods (GMG) this will be realized by a grid hierarchy, in the case of algebraic multigrid methods (AMG) we are concerned with a single grid only and therefore we have to construct a matrix hierarchy by pure algebraic operations.

Since multigrid methods were basically developed for the application to the solution of finite difference or finite element equations, their most essential components such as smoothing and prolongation operator are primarily designed for these problem classes. An alternative solving approach is the boundary element method (BEM), which on the one hand yields fully populated matrices, but on the other hand we only have to discretize the boundary of the computational domain. If we want to apply AMG to BEM-matrices we will be forced to make some adaptions. In particular, common smoother types do not work properly in the case of the single layer potential because eigenvalues and eigenfunctions act conversely.

In order to obtain an efficient iterative solver for boundary element equations, we have to consider a fast matrix-by-vector multiplication for dense matrices. Thus, several different methods for a matrix approximation have been proposed in the last years. In our examples we are using the 'adaptive cross approximation' method (ACA) which provides a low-rank approximation. Now we are able to perform a fast matrix-by-vector multiplication and therefore it is feasible to apply an AMG preconditioned solver reasonably.

First numerical experiments show almost optimal complexity of the suggested AMG approach. We obtain fast convergence and small numbers of iterations, in the case of symmetric Galerkin-matrices as well as in the case of non-symmetric collocation-matrices.

In addition, for real-life BEM-FEM coupled systems we implemented the AMG solver for the finite element method (FEM) part only at the moment. As we will see, first tests provides proper results again. Furthermore, one might suggest a multigrid based solver that could be applied on the overall coupled equation system. Both appearing types of matrices, the FEM and BEM matrices, can be treated well and therefore all necessary presuppositions can be provided for constructing an appropriate solver.

# Acknowledgements

First of all I would like to express my thanks to Prof. Dr. Ulrich Langer for giving me the chance to work on this exciting topic and to write my diploma thesis on that. Further I would like to thank him for answering my questions and for suggesting new ideas during interesting discussions.

I would like to express my special thanks to Dr. Stefan Reitzinger for supervising me through the development of this thesis, especially for giving me vital hints and support concerning the implementation work and presentation experiences. I would like to thank him for his patience to answer all my arising questions and for giving me advice on overcoming particular difficulties.

I also would like to thank Dr. Stefan Kurz (Robert Bosch GesmbH) and his co-workers of the department FV/FLO-MOD who gave me the chance to stay six months at Stuttgart-Schillerhöhe. I have gathered a lots of experience in implementation and treatment of real life problems.

In addition, I would like to thank Dr. Volker Rischmüller for supervising and supporting me during my stay at the Robert Bosch GesmbH.

Finally, I would like to express my sincere gratitude to my parents Leopoldine and Heinrich for making me possible to study and for supporting and sustaining me all the years.

# Contents

# Chapter 1

# Introduction

In this diploma thesis we are concerned with the numerical solution of boundary element equations. Most algebraic multigrid applications are designed for sparse matrices arising from the finite element (FE) discretizing of second-order elliptic partial differential equations. Thus, we have to solve

$$K_h \underline{u}_h = \underline{f}_h \quad \text{in } \mathbb{R}^{N_h} \tag{1.1}$$

and, because we are faced with a large number of unknowns, the solution process will be a tricky task.

One of the most efficient methods to solve such system of equations is the multigrid approach [18]. The simplest version of the multigrid iteration will be the twogrid (TG) algorithm which consists of several ingredients. In fact, the efficiency depends on a clever interaction of smoothing on the fine level and coarse grid correction on the coarse level, respectively. In the case of finite element applications the multigrid methods are well understood, for geometric multigrid (GMG) as well as for algebraic multigrid (AMG). In the further chapters we will suggest algebraic multigrid methods, which have essential differences compared to the geometric version of multigrid methods. Since only a single grid information is available we have to ensure an appropriate construction of transfer operators and the according matrix hierarchy. In many cases, we will obtain better results (i.e. number of iterations, setup time, etc.) if we are using additional geometric information. Thus, the setup procedure is performed by using an auxiliary matrix, which contains some of the geometric informations in a certain sense. Preferably, we are using M-matrices which allow the construction of a prolongation operator with some important properties [8, 10, 36, 37]. The implementation of a numerical algorithm corresponding to this approach allows a proper treatment of anisotropies, polynomial trial functions of higher order and other applications (see Chapter 3). In fact, most coarsening strategies are designed for sparse FE-matrices and, as we will see later, in the case of boundary element (BE) matrices we have to realize the auxiliary matrix approach as well.

Anyway, for many applications it is suitable to use boundary element methods. Let us consider field problems that take effect on an unbounded domain. Since finite element methods assume a bounded computational domain, the modeling of such a problem class always demands some restrictions and that implies less accuracy. By means of boundary element methods we are able to obtain exact results on arbitrary points in the unbounded domain. Moreover, in this case only the surface of the geometric body needs to be discretized and therefore the dimension of the computational domain is of order one less than the dimension of an equivalent FE-discretization. Another advantage of the BE-method occurs in their advantageous handling considering the treatment of moving parts. If we will apply FE-discretization in this case, we would be concerned with disturbing problems during the whole discretization performance. We obviously not able to use the same grid throughout the entire calculation, thus the moving parts need to be discretized new after each time step. Due to that disadvantage the boundary element method will be the more convenient technique in this case.

Nevertheless, we have to deal with at least two essential drawbacks. Independent of classical boundary element discretization techniques, we are concerned with fully populated BE-matrices. Hence, every iterative solving algorithm will result in a complexity of $\mathcal{O}(N_h^2)$ for arithmetical operations and for memory consumption, respectively. Furthermore, if we apply Galerkin's method to construct the BE-matrices, we will take into account the problem of evaluating each matrix entry. In order to avoid the calculation of a double-integral, most commercial program packages kindly perform discretization via collocation methods, which unfortunately yields a non-symmetric system matrix. For this reason it is not possible to apply the conjugate gradient algorithm. Thus, we need some appropriate Krylov subspace method. Widely used methods are the GMRES and BiCGSTAB algorithms, see [29].

The fastest versions of iterative solvers considered in the last paragraph will be the preconditioned variations. Since the number of iterations to gain a certain accuracy depends on the condition number $\kappa(K_h)$ of the system matrix $K_h$ it would be preferable to have $\kappa(K_h) = \mathcal{O}(1)$. Let $h$ be the typical mesh size, then for standard FE-matrices we have $\kappa(K_h) = \mathcal{O}(h^{-2})$. For BE-matrices arising from discretizing the single layer potential we obtain $\kappa(K_h) = \mathcal{O}(h^{-1})$, the same result is true for the hypersingular operator. For the symmetric, positive definite system matrix $K_h$, the spectral inequalities

$$\gamma_l(\underline{u}, \underline{u}) \leq (K_h \underline{u}, \underline{u}) \leq \gamma_u(\underline{u}, \underline{u}) \quad \forall \underline{u} \in \mathbb{R}^{N_h}$$

yield the spectral condition number bound

$$\kappa(K_h) := \frac{\lambda_{max}}{\lambda_{min}} \leq \frac{\gamma_u}{\gamma_l}.$$

The parameters $\lambda_{min}$ and $\lambda_{max}$ denote the upper and lower eigenvalue of $K_h$, respectively. In addition, we introduce the term of spectrally equivalent ma-

trices. Thus, we look for a matrix $C_h$ called preconditioner, which fulfills the spectral equivalent inequality

$$\underline{\gamma}(C_h\underline{u},\underline{u}) \leq (K_h\underline{u},\underline{u}) \leq \overline{\gamma}(C_h\underline{u},\underline{u}) \quad \forall \underline{u} \in \mathbb{R}^{N_h}$$

such that the following properties are fulfilled:

1. $\kappa(C_h^{-1}K_h) \leq \frac{\overline{\gamma}}{\underline{\gamma}} = \mathcal{O}(1)$
2. 'fast' realization of $C_h^{-1} * \underline{d}$.

Obviously, $C_h \equiv K_h$ would be the best choice to achieve the first property and the choice $C_h \equiv I$ (or any diagonal matrix $D$) would satisfy the second item best. Of course, we have to find a reasonable trade-off between these two possibilities. Fast realization of $C_h^{-1} * \underline{d}$ means here, that $\mathcal{O}(C_h^{-1} * \underline{d}) = \mathcal{O}(K_h * \underline{d})$ up to a polylogarithmic factor. If we use multigrid methods to realize a preconditioner for our iterative solving algorithm, we will obtain a very efficient option. Depending on the regularity suppositions we have constant condition number or at most increasing condition number corresponding to the level of the multigrid algorithm, respectively.

In the case of boundary element methods the overall multigrid approach only will make sense, if the cost of a single matrix-by-vector multiplication can be reduced essentially (especially in 3D). Several different methods to perform a boundary element matrix compression has been developed in the last years [2, 3, 15, 19, 20, 35]. The application of a sparse representation algorithm allows us to realize the matrix-by-vector multiplication in almost $\mathcal{O}(N_h)$ operations.

Furthermore, the arising single layer potential operator shows different behavior compared to finite element matrices. Due to the reverse action of eigenvalues and eigenvectors (because the single layer operator is a pseudo differential operator of order minus one) the smoothing component of the multigrid algorithm has to adapted properly. Moreover, the construction of the coarse grid system, especially setting up the coarse system matrix via Galerkin's method, requires some considerations. In the following chapters we will focus on the appropriate construction of an algebraic multigrid algorithm for boundary element equations. In particular, we will use the AMG method as a preconditioner in the conjugate gradient algorithm (CG) and according variants for non-symmetric matrices.

A combination of FEM and BEM methods leads to a coupled system. This class of problems often arises by dealing with Maxwell's equations. The solid contributions of a considered electromagnetic device could be described properly via finite elements, the exterior field components could be evaluated best by using the boundary element technique. In order to show the efficiency of the AMG approach, we only consider the FEM-part. The overall calculation strategy is given by some clever domain decomposition and according preconditioning. This process yields the Schur-complement of the FE-matrix. At

this point the AMG package will be installed to realize $K_{\Omega\Omega}^{-1} * \underline{d}$, with $K_{\Omega\Omega}$ the part of the FE-matrix, which corresponds only to non-boundary nodes. Another idea for solving a BEM-FEM-coupled system is to apply a multigrid algorithm to the overall equation system [25]. This approach combined with multigrid methods and sparse boundary element matrix representations could be an interesting field of research in upcoming applications.

The rest of this thesis is organized as follows. In Chapter 2 we introduce the basics of boundary element methods. Moreover, we point out a method to approximate the fully populated boundary element matrices in a sparse manner. Therefore, we give a brief overview over the adaptive cross approximation method (ACA) [2, 3]. The AMG technique is treated in Chapter 3. Therein the most essential ingredients of a twogrid-algorithm are presented, as well as some convergence estimates. Consequently, the adaption of AMG parts with regards to boundary element matrices is shown in Chapter 4. First experiments and examples in 2D can be found in Chapter 5. In this chapter we treat dense BE-matrices as well as ACA-matrices. Finally, in Chapter 7 we give a conclusion and a prospective outlook to the future.

# Chapter 2

# Boundary Element Method

In this chapter we want to introduce the basics of the boundary element method. In particular we first of all consider the theory of the Sobolev spaces and their expansion to arbitrary manifolds. A motivation for using the boundary element method is given by considering the interior Dirichlet problem for Laplace's equation. Within this example we point out the properties of the arising boundary element operators and analyze appropriate discretization methods. In addition, we suggest an approximation method for BE-matrices in order to avoid any further calculations with dense matrices. Therefore, we apply an ACA-algorithm [2, 3] to get rid of this essential drawback and obtain the desired results, which are a matrix-by-vector multiplication of almost $\mathcal{O}(N_h)$.

## 2.1 Basic Theory

Let $\Omega \subset \mathbb{R}^d$ (d=2,3) be a single connected, bounded domain with sufficient smooth boundary $\Gamma = \partial\Omega$. We consider the boundary element technique by means of the interior Dirichlet problem for Laplace's equation:

$$\begin{aligned} -\Delta u(x) &= 0 \qquad && x \in \Omega \\ u(x) &= g(x) && x \in \Gamma \end{aligned} \tag{2.1}$$

Inhomogeneous problems or problems with other differencial operators (e.g. elasticity, etc.) are treated in [12, 42, 43]. In order to find the solution $u$ of (2.1) let us introduce the well-known *Sobolev spaces*, see [1]. In addition we give some common definitions and notations. Let us begin with the Sobolev space $H^k(\Omega),\ k \in \mathbb{N}$

$$H^k(\Omega) := \{u \in L_2(\Omega) : \partial^\alpha u \in L_2(\Omega),\ 0 \leq |\alpha| \leq k\}$$

$$(u, v)_{H^k(\Omega)} := \sum_{|\alpha| \leq k} \int_\Omega |\partial^\alpha u(x)||\partial^\alpha v(x)| dx$$

with the corresponding norm

$$\|u\|^2_{H^k(\Omega)} := (u,u)_{H^k(\Omega)}.$$

In the definition above $\partial^\alpha u$ denotes the generalized derivative, with the common mullti-index $\alpha = (\alpha_1, \ldots, \alpha_d)$, $|\alpha| = \alpha_1 + \cdots + \alpha_d$. For real $s \in \mathbb{R}^+$ with $s = k + \sigma$ and $k \in \mathbb{N}_0$, $\sigma \in (0,1)$ we are concerned with the so-called *Sobolev-Slobodeckij spaces*

$$H^s(\Omega) := \{u \in H^k(\Omega) : |u|_{k+\sigma} < \infty\}$$

with the inner product

$$\begin{aligned} (u,v)_{H^s(\Omega)} &:= (u,v)_k + (u,v)_{k+\sigma} \\ (u,v)_{k+\sigma} &:= \sum_{|\alpha|=k} \int_\Omega \int_\Omega \frac{(\partial^\alpha u(x) - \partial^\alpha u(y))(\partial^\alpha v(x) - \partial^\alpha v(y))}{|x-y|^{d+2\sigma}} dxdy \end{aligned}$$

As shown above we get the corresponding norm and semi-norm, respectively

$$\begin{aligned} |u|^2_{k+\sigma} &:= (u,u)_{k+\sigma} \\ \|u\|^2_{H^s(\Omega)} &:= \|u\|^2_{H^k(\Omega)} + |u|^2_{k+\sigma}. \end{aligned}$$

Sobolev spaces $H^{-s}(\Omega)$, $s > 0$ with negative index are defined as the dual spaces of $H^s(\Omega)$ with corresponding norm

$$\|u\|_{H^{-s}(\Omega)} := \sup_{0 \neq v \in H^s(\Omega)} \frac{|(u,v)_{H^s(\Omega)}|}{\|v\|_{H^s(\Omega)}}.$$

Furthermore let us mention the possibility to define Sobolev spaces on manifolds of the space $\mathbb{R}^d$, especially on the boundary $\Gamma = \partial\Omega$. This generalization of the Sobolev spaces gives us the fundamental of existence and uniqueness assertions in the whole boundary element approach. At the end of this short overview over Sobolev spaces let us note that $H^0(\Omega) \equiv L^2(\Omega)$.

Now we consider the Green's Formula and their behavior when using special functions. In particular the fundamental solution of the Laplace operator $-\Delta_x$ will be such a function.

The following Green's Formula hold for the Laplace operator $-\Delta_x$:

- I. Green's Formula
  $\forall u \in H^1(\Omega)$ and $\forall v \in H^2(\Omega)$

$$\int_\Omega u(x)\Delta v(x)dx = \int_\Gamma u(x)\frac{\partial v}{\partial n_x}(x)ds_x - \int_\Omega \nabla_x^\top u(x)\nabla_x v(x)dx \qquad (2.2)$$

- II. Green's Formula
  $\forall u \in H^2(\Omega)$ and $\forall v \in H^2(\Omega)$

$$\int_\Omega u(x)\Delta v(x)dx - \int_\Omega \Delta u(x)v(x) = \int_\Gamma u(x)\frac{\partial v}{\partial n_x}(x)ds_x - \int_\Gamma \frac{\partial u}{\partial n_x}(x)v(x)ds_x \tag{2.3}$$

Whereby $n_x$ denotes the unit outward normal vector to $\Gamma$ at some point $x \in \Gamma$. The II. Green's Formula is the starting point of the direct boundary integral method. If we set $v(x)$ the fundamental solution $E(x,y)$ we will be able to formulate a boundary integral equation with certain properties. Assume a scalar elliptic differential operator $L_x$ that is applied to a fundamental solution $E(x,y)$ the following equations hold for arbitrary fixed $y \in \mathbb{R}^d$

$$\begin{aligned} L_x E(.,y) &= \delta(.-y) \qquad in\ \mathcal{D}' \\ \langle L_x E(.,y), \phi\rangle_{\mathcal{D}'\times\mathcal{D}} &= \phi(y) \qquad \forall \phi \in \mathcal{D}. \end{aligned}$$

with $\mathcal{D}$ denotes the space of basic functions and the dual space $\mathcal{D}'$ the space of distributions. Moreover $\langle .,. \rangle_{\mathcal{D}'\times\mathcal{D}}$ denotes the duality product with respect to $\mathcal{D}$ [26]. Finally, $\delta(.-y)$ is the usual notation for the delta-distribution.
In this chapter we are considering the Laplace operator $-\Delta_x$ for which the fundamental solutions can be found immediately. Hence, in $\mathbb{R}^d$ $(d = 1, 2, 3)$ we have

$$E(x,y) = \begin{cases} \frac{1}{2}(1-|x-y|) & x,y \in \mathbb{R} \\ -\frac{1}{2\pi}\ \ln|x-y| & x,y \in \mathbb{R}^2 \\ -\frac{1}{4\pi}\frac{1}{|x-y|} & x,y \in \mathbb{R}^3 \end{cases}$$

that can be checked via simple recalculating. Take into account the singularity in the fundamental solution $E(x,y)$ for $x$ tends to $y$ we get a representation formula for the solution $u$ of (2.1)

- III. Green's Formula
  $\forall u \in H^2(\Omega)$

$$\sigma(y)u(y) = -\int_\Gamma u(x)\frac{\partial E}{\partial n_x}(x,y)ds_x + \int_\Gamma \frac{\partial u}{\partial n_x}(x)E(x,y)ds_x. \tag{2.4}$$

Furthermore for $\sigma(y)$ the following values hold

$$\sigma(y) = \begin{cases} 0 & \forall y \in \mathbb{R}^d \backslash \bar{\Omega} \\ \frac{\Theta}{2\pi} & \forall y \in \Gamma = \partial\Omega \\ 1 & \forall y \in \Omega \end{cases}$$

with $\Theta$ denotes the angle enclosed by the domain $\Omega$ at the point $y$. In the case of a smooth boundary it follows that $\sigma(y) = \frac{1}{2}$.

It is obvious that the task of finding a solution $u$ of (2.1) reduces to the task of solving the missing Cauchy-data $\frac{\partial u}{\partial n}$. Additionally in the case of Neumann boundary condition we have to evaluate the Dirichlet value $u$ at the boundary $\Gamma$. Therefore we make a closer look at the arising operators and their properties in the next section.

## 2.2 Boundary Integral Operators and Properties

The representation formula (2.4) can be formulated in operator form, hence we have

$$(\frac{1}{2}I + K)u(y) = Vv(y) \tag{2.5}$$

with the boundary integral operators

$$(Vv)(y) = \int_\Gamma E(x,y)v(x)ds_x \tag{2.6}$$

$$(Ku)(y) = \int_\Gamma u(x)\frac{\partial E}{\partial n_x}(x,y)ds_x \tag{2.7}$$

Moreover we can define a second boundary integral equation which comes from the normal derivative of (2.5)

$$(\frac{1}{2}I - K')v(y) = Du(y) \tag{2.8}$$

whereby the appearing operators are defined by

$$(Du)(y) = -\frac{\partial}{\partial n_y}\int_\Gamma \frac{\partial}{\partial n_x}E(x,y)u(x)ds_x \tag{2.9}$$

$$(K'v)(y) = \int_\Gamma v(x)\frac{\partial E}{\partial n_y}(x,y)ds_x. \tag{2.10}$$

If we assume the boundary $\Gamma$ to be sufficient smooth those boundary integral operators will define pseudo differential operators of order 0 and $\pm 1$, respectively

$$\begin{array}{rcll} K : H^s(\Gamma) & \mapsto & H^s(\Gamma) & \text{Double Layer Potential} \\ K' : H^s(\Gamma) & \mapsto & H^s(\Gamma) & \text{Adjoint Double Layer Potential} \\ V : H^s(\Gamma) & \mapsto & H^{s+1}(\Gamma) & \text{Single Layer Potential} \\ D : H^s(\Gamma) & \mapsto & H^{s-1}(\Gamma) & \text{Hypersingular Operator} \end{array}$$

for $s \in \mathbb{R}$, cf. [34].
The operator equations (2.5) and (2.8) can be written in a compact form

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}I - K & V \\ D & \frac{1}{2}I + K' \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}. \tag{2.11}$$

Thus, equation (2.11) defines a projection with

$$C = \begin{pmatrix} \frac{1}{2}I - K & V \\ D & \frac{1}{2}I + K' \end{pmatrix}$$

is called *Calderon* projector. Moreover we are able to define an explicit description of the Neumann data with respect to the Dirichlet data. Proceeding from equation (2.5) we obtain

$$v = V^{-1}(\frac{1}{2}I + K)u =: Su.$$

The mapping $S$ defines the *Steklov-Poincaré* operator (Dirichlet-Neumann-map) which can also be formulated in a symmetric version

$$S = (D + (\frac{1}{2}I + K'))V^{-1}(\frac{1}{2}I + K).$$

Hence in this case we are working with the *symmetric Steklov-Poincaré* operator.

Let us suggest a boundary $\Gamma \in C^{0,1}$ and $\langle \cdot, \cdot \rangle : H^{-\alpha}(\Gamma) \times H^{\alpha}(\Gamma) \mapsto \mathbb{R}$ the respective duality product ($\alpha = \pm\frac{1}{2}$), then several properties for boundary integral operators can be observed:

1. The operators $V$ and $D$ are self-adjoint

$$\begin{aligned} \langle u, Vv \rangle &= \langle Vu, v \rangle \quad \forall u, v \in H^{-1/2}(\Gamma) \\ \langle u, Dv \rangle &= \langle Du, v \rangle \quad \forall u, v \in H^{1/2}(\Gamma). \end{aligned}$$

2. Operator $K'$ is adjoint to operator $K$

$$\langle Ku, v \rangle = \langle u, K'v \rangle \quad \forall u \in H^{1/2}(\Gamma) \quad \forall v \in H^{-1/2}(\Gamma).$$

3. The hypersingular operator $D$ is positive semidefinite, i.e.

$$\begin{aligned} & \langle Du, u \rangle \geq 0 && \forall u \in H^{1/2}(\Gamma) \\ \exists \underline{\mu}_D > 0 : \quad & \langle Du, u \rangle \geq \underline{\mu}_D \|u\|^2_{H^{1/2}(\Gamma)} && \forall u \in H^{1/2}(\Gamma)|_{\ker D} \\ & \ker D = \operatorname{span}\{1\}. \end{aligned}$$

4. In the case of $\Omega \subset \mathbb{R}^3$ the single layer potential is positive definite

$$\exists \underline{\mu}_V > 0 \quad \langle v, Vv \rangle \geq \underline{\mu}_V \|v\|^2_{H^{-1/2}(\Gamma)} \quad \forall v \in H^{-1/2}(\Gamma).$$

   If we consider the case of $\Omega \subset \mathbb{R}^2$ then in general $V$ will not be positive definite. In order to get ellipticity we have to satisfy the condition $\operatorname{diam} \Omega < 1$ but that can be immediately fulfilled by an appropriate scaling of the domain $\Omega$, see [13, 21].

The last property of the pseudo differential operators $K, K', V$ and $D$ is expressed in Proposition 2.2.1 which points out the conditions for continuity

**Proposition 2.2.1.** *Let $\Gamma \in C^{0,1}$ and $s \in (-\frac{1}{2}, \frac{1}{2})$ then the pseudo differential operators defined in (2.6), (2.7),(2.9) and (2.10) are continuous*

$$\begin{array}{rcccc} K & : & H^{1/2+s}(\Gamma) & \mapsto & H^{1/2+s}(\Gamma) \\ K' & : & H^{-1/2+s}(\Gamma) & \mapsto & H^{-1/2+s}(\Gamma) \\ V & : & H^{-1/2+s}(\Gamma) & \mapsto & H^{1/2+s}(\Gamma) \\ D & : & H^{1/2+s}(\Gamma) & \mapsto & H^{-1/2+s}(\Gamma) \end{array} .$$

*Furthermore the single layer potential $V$ is continuous for $s = \pm\frac{1}{2}$.*

*Proof.* A proof can be found in [13]. □

## 2.3 Discretization

In this section we study two procedures of discretization. On the one hand we analyze a projection method, in particular the well-known Galerkin method. On the other hand we consider the collocation method that is mostly used in commercial boundary element method software packages. Moreover we use the collocation method in our numerical examples in Chapter 5.

### 2.3.1 Galerkin Method

Let $\Omega \subset \mathbb{R}^2$, $\Gamma = \partial\Omega \in C^{0,1}$ and $V : H^{-1/2}(\Gamma) \mapsto H^{1/2}(\Gamma)$ be the single layer potential. Thus, we have a problem formulation for the boundary integral equation in operator form: Find $v \in H^{-1/2}(\Gamma)$

$$Vv = f \qquad \text{in } H^{1/2}(\Gamma). \tag{2.12}$$

In addition the equivalent variational formulation is: Find $v \in H^{-1/2}(\Gamma)$

$$\langle w, Vv\rangle = \langle w, f\rangle \qquad \forall w \in H^{-1/2}(\Gamma). \tag{2.13}$$

Let us mention the previous section and assume diam $\Omega < 1$ , then we know that

- $V$ is self-adjoint
- $V$ is bounded and elliptic in $H^{-1/2}(\Gamma)$, i.e. there exists constants $\underline{\mu}_V, \overline{\mu}_V$ and $\forall v, w \in H^{-1/2}(\Gamma)$ such that the inequalities

$$\begin{array}{rcl} \langle v, Vv\rangle & \geq & \underline{\mu}_V \|v\|^2_{H^{-1/2}(\Gamma)} \\ \langle w, Vv\rangle & \leq & \overline{\mu}_V \|w\|_{H^{-1/2}(\Gamma)} \|v\|_{H^{-1/2}(\Gamma)}. \end{array}$$

hold. Therefore all conditions for the *Lax-Milgram* Lemma are complied and we obtain the result

$$\exists!\, v \in H^{-1/2}(\Gamma) \text{ that fulfills (2.13)}.$$

The proof of the Lax-Milgram Lemma and according estimates of the iteration error can be looked up in [26] and references therein. Now we define a finite subspace of $H^{-1/2}(\Gamma)$ where the trial and test functions are choosed from:

$$X_h := \text{span } \Phi_h = \text{span } [\phi_1, \ldots, \phi_{N_h}] \subset H^{-1/2}(\Gamma).$$

Consequently an approximation $v_h$ of the solution $v$ of (2.13) can be found by

$$\begin{aligned} v_h(x) &= \sum_{k=1}^{N_h} v_k \phi_k(x) \\ \underline{v}_h &= (v_1, \ldots, v_{N_h})^\top. \end{aligned}$$

Furthermore, the variational formulation (2.13) has to be evaluated for each test function $\phi_k$. Hence we get for all $k = 1, \ldots, N_h$

$$\begin{aligned} \langle \phi_k, V v_h \rangle &= \langle \phi_k, V \sum_{j=1}^{N_h} v_j \phi_j \rangle \\ &= \sum_{j=1}^{N_h} v_j \langle \phi_k, V \phi_j \rangle \\ &= \langle \phi_k, f \rangle \end{aligned}$$

and in compact matrix form

$$V_h \underline{v}_h = \underline{f}.$$

The system matrix $V_h$ is dense, symmetric and positive definite that can be immediately derived from the properties of the single layer potential

$$\langle \phi_k, V \phi_j \rangle_0 = \int_\Gamma \int_\Gamma E(x,y) \phi_j(x) \phi_k(y) ds_x ds_y.$$

More precisely, if we have the discretization nodes $\{x_1, \ldots, x_{N_h}\}$ with $x_i \in \Gamma$ and $x_i \neq x_j$, we will be able to define a corresponding partitioning of $\Gamma$. Thus, we get a discretized boundary $\Gamma_h = \bigcup_{j=1}^{N_h} \Gamma_j$ with $\Gamma_j$ are generalized polygons in the case of $\Omega \subset \mathbb{R}^3$ and $\Gamma_j = \{x_j + t(x_{j+1} - x_j) | t \in [0,1)\}$ in the case of $\Omega \subset \mathbb{R}^2$, respectively. A widely used basis functions of our trial space are piecewise polynomes. In particular we are going to use *B-splines* which are splines with the almost smallest support. A rigorous treatment of approximation and convergence properties for splines is given in [39].

### 2.3.2 Collocation Method

If we apply the collocation method in order to solve equation (2.12) we first of all will define a set of collocation points $\{y_i | i = 1, \ldots, N_h\}$. In the case of $\Omega \subset \mathbb{R}^2$ an appropriate choice could be $y_i = x_i + \frac{1}{2}(x_{i+1} - x_i)$ with $x_i$ the discretization points. We again write the approximated solution $v_h$ as

$$v_h(x) = \sum_{i=1}^{N_h} v_i \phi_i(x)$$

that leads to the collocation equation

$$Vv(y_j) = f(y_j) \qquad j = 1, \ldots, N_h.$$

If we assume piecewise constant functions $\phi_j$ for the Neumann-data the corresponding equation will hold for $j = 1, \ldots, N_h$

$$\begin{aligned}(Vv_h)(y_j) &= \int_{\Gamma_h} E(x, y_j) v_h(x) ds_x \\ &= \int_{\Gamma_h} E(x, y_j) \sum_{i=1}^{N_h} v_i \ \phi_i(x) ds_x \\ &= \sum_{i=1}^{N_h} v_i \int_{\Gamma_i} E(x, y_j) ds_x.\end{aligned}$$

In addition we have the compact matrix equation

$$V_h \underline{v}_h = \underline{f}_h \tag{2.14}$$

with $(V_h)_{ij} = \int_{\Gamma_j} E(x, y_i) ds_x$ is fully populated and of course non-symmetric in general. Hence, it is a quite difficult task to solve (2.14) efficient. Most software packages use some direct solvers or Krylov subspace methods like GMRES and BiCGStab, see [29, 38].

## 2.4 Sparse Representation of BEM-Matrices

In order to get an optimal solver for boundary integral equations, we want to apply iterative methods. Thus, it is necessary to perform a fast matrix-by-vector routine. Naive realization of such a procedure would cost $\mathcal{O}(N_h^2)$ arithmetical operations and the required memory would increase like $\mathcal{O}(N_h^2)$ too.
To avoid those drawbacks we consider an approximation of the dense boundary element system matrices. Our aim is to find a sparse representation $\tilde{K}_{BEM}$ of

the system matrix $K_{BEM} = V_h$ which approximates the dense boundary element system matrix in a certain sense.
The construction of the sparse matrix can be realized by several different methods. A general approach of the popular multipole method can be found in [35, 11], panel-clustering methods in [20] and a $\mathcal{H}$-matrix approach is given in [19].
In the following, we will consider the adaptive cross approximation method suggested by M. Bebendorf and S. Rjasanow. A much rigorous explanation and detailed proofs of this method can be found in [2, 3].

### 2.4.1 Adaptive Cross Approximation

The adaptive cross approximation is a very simple method to approximate matrices originated from collocation boundary element discretization. In fact it is not necessary to know an explicite description of the kernel. The approximation is based on pure algebraic transformations, therefore only a procedure for evaluating the collocation integral has to be applied.

The basic idea is to decompose the computational domain into smaller clusters and classify them to a near-field domain and a far-field domain, respectively. Furthermore the matrix blocks from the interaction of the far-field clusters can be approximated by low-rank matrices, i.e. for a matrix block $A \in \mathbb{R}^n_m$ the cost of memory and matrix-by-vector multiplication will decrease from $\mathcal{O}(m*n)$ down to $\mathcal{O}(r*(m+n))$, with $r$ denotes the rank of the low-rank matrix.

Thus, we first take a closer look to the domain decomposition and the partitioning of the resulting system matrix. In order to avoid any confusions let us mention, that in the remaining of this chapter most appearing notations are directly adopted from [2], e.g. $D_h$ means a subset of $\mathbb{R}^d$, $d = 2, 3$ (and not the hypersingular integral operator) with

$$D_h = \bigcup_{i=1}^{N_h} X_i$$

whereby $X_i$ denotes the support of the trial functions $\phi_i$, i.e.

1. Collocation-Method

$$(A)_{ij} = \int_{X_j} \kappa(x, y_i)\phi_j(x) ds_x, \tag{2.15}$$

   $y_i$ the collocation point of subset $X_i$.

2. Galerkin Method

$$(A)_{ij} = \int_{X_i} \int_{X_j} \kappa(x, y)\phi_i(x)\phi_j(y) ds_x ds_y. \tag{2.16}$$

Moreover, we want to decompose the index-set $I = \{1, ..., N_h\}$ into index-clusters $t \subset I$ in order to get clusters

$$D_h^t = \bigcup_{i \in t} X_i$$

with

$$I = \bigcup_{k=1}^{N_C} t_k, \qquad D_h = \bigcup_{k=1}^{N_C} D_h^{t_k}$$

with $N_C$ denotes the number of index-clusters.
Now we give a prescription that classifies clusters into a near-field and a far-field. If we need not take into account the singularity of the boundary integral kernel (i.e. both clusters have a certain distance) we will call them $\eta$ - *admissible.*

**Definition 2.4.1.** *Let $(D_1, D_2)$ be a cluster pair with $D_1, D_2 \subset \mathbb{R}^d$, then $(D_1, D_2)$ is called $\eta$ - admissible if*

$$diam\, D_2 \leq \eta\, dist(D_1, D_2). \tag{2.17}$$

In addition we define $\eta$-admissibility for index cluster pairs according to Definition 2.4.1.

**Definition 2.4.2.** *An index cluster pair $(t_1, t_2)$ with $t_1, t_2 \subset I$ is called $\eta$ - admissible, if $\min(\sharp t_1, \sharp t_2) > 1$ and the cluster pair $(D_h^{t_1}, D_h^{t_2})$ is $\eta$ - admissible:*

$$diam\, D_h^{t_1} \leq \eta\, dist(D_h^{t_1}, D_h^{t_2}). \tag{2.18}$$

The desired decomposition of the index-set $I = \{1, \ldots, N_h\}$ is done by using the recursive Algorithm 1. The resulting set of pairwise disjoint index-clusters is hierarchical structured and will be called *cluster-tree*. The function $S(t)$

**Algorithm 1** Cluster-Tree

**set** T = ClusterTree(I)
**function** ClusterTree(t)
**return** $t \cup \bigcup_{s \in S(t)}$ ClusterTree$(s)$

specifies the splitting of the index-set $t$ into several subsets $t_i$ with $t = \bigcup_i t_i$ and $t_i \cap t_j = \emptyset$, $i \neq j$. Thus, we give a suitable procedure $S(t)$ to build a decomposition of an index set $I$ and therefore of the entire domain $D_h$. First we will characterize a cluster $D_h^t = \bigcup_{i \in t} X_i$ by a certain point $m_t$, for instance the center point

$$m_t = \frac{1}{k} \sum_{i=1}^{k} x_i.$$

In order to split the cluster we have to determine the main alignment $w_t$ which fulfills

$$\frac{1}{\|w_t\|_0}\sum_{i=1}^{k}|w_t^\top(x_k-m_t)|^2=\max_{v\in\mathbb{R}^d}\frac{1}{\|v\|_0}\sum_{i=1}^{k}|v^\top(x_k-m_t)|^2.$$

Once we evaluated the main alignment we can formulate a condition to divide the index cluster $t$ into $t_1$ and $t_2=t\backslash t_1$, see Figure 2.1:

$$t_1=\{i\in t: w_t^\top(m_t-m_i)>0\}.$$

The task of calculating $w_t$ is rather simple due to the fact that $w_t$ is the eigen-

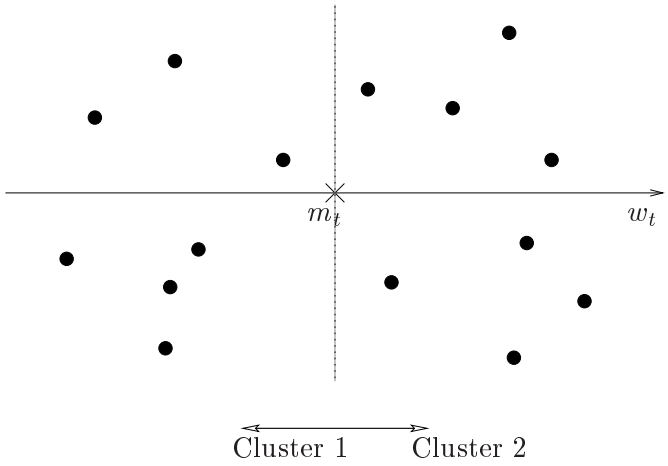



Figure 2.1: Domain Decomposition

vector which corresponds to the greatest eigenvalue of the covariance matrix $C\in\mathbb{R}_d^d$

$$C=\sum_{i=1}^{N_t}(x_i-m_t)(x_i-m_t)^\top\quad N_t=\dim\ t$$

Hence, we can get $w_t$ by using von Mises' vector iteration

$$x_{k+1}=\frac{Cx_k}{\|Cx_k\|_2},\quad k=0,1,2,\ldots$$

with an initial guess $x_0\neq 0$. This procedure can be stopped after only few iteration steps.
The matrix partitioning is done by constructing a tree according to Algorithm 2. Here, the recursive method is applied to an index-pair $(t_1,t_2)\subset I\times I$. An appropriate function $SS(b)$ is for instance

$$SS(b)=\begin{pmatrix}(s_1,s_2), & \text{if}\, S(t_1)\neq\emptyset\neq S(t_2)\\ \emptyset & \text{if}\, S(t_1)=\emptyset\vee S(t_2)=\emptyset\end{pmatrix}.$$

**Algorithm 2** Clusterpair Tree

**set** TT = ClusterTree(I × I)
**function** $ClusterTree(b)$
**return** $b \cup \bigcup_{s \in SS(b)} ClusterTree(s)$

In comparison to the cost of building the approximating matrix, the effort of constructing the cluster-tree can be neglected. But the construction of the clusterpair tree would be inefficient, therefore we modify Algorithm 2 in such a way, that clusters which fulfill (2.18) need not to be decomposed any longer.

Now we want to give a brief overview over the algebraic matrix approximation. Hence, it is our aim to find a low-rank description of the system matrix $K_{BEM}$. In particular for every block matrix arising from the interaction of two clusters $(D_1, D_2)$ we have to decide whether it is $\eta$-admissible or not. Assuming that a block matrix satisfies the condition (2.17), only few matrix entries has to be calculated and thus the cost of storage and CPU time for matrix-by-vector multiplication will decrease essentially. If a clusterpair does not fulfill $\eta$-admissibility the according matrix will be calculated directly.
Let us consider the continuous problem

$$(\mathcal{L}_j\kappa)(y) = \int_{D_1} \kappa(x,y)\varphi_j(x)ds_x \qquad j = 1,\dots,n$$
$$y \in D_2$$

and generate a sequence of functions $(r_k)$, $(s_k)$ It can be observed that $\mathcal{L}_j r_k$

**Algorithm 3**

**set** $i = 0,\ r_0(x,y) = \kappa(x,y),\ s_0 = 0$
**for all** k=1,... **do**
  **let** $i_k$ be the first index in $\{i_{k-1}+1,\dots,m\}$
  with $\exists j_k \in \{1,...,n\} : \mathcal{L}_{j_k} r_{k-1}(y_{i_k}) \neq 0$
  **if** $i_k$ doesn't exist $\rightarrow$ **stop**
  **set** $\gamma_k^{-1} = \mathcal{L}_{j_k} r_{k-1}(y_{i_k})$
  $r_k(x,y) = r_{k-1}(x,y) - \gamma_k(\mathcal{L}_{j_k} r_{k-1})(y) r_{k-1}(x,y_{i_k})$
  $s_k(x,y) = s_{k-1}(x,y) + \gamma_k(\mathcal{L}_{j_k} s_{k-1})(y) r_{k-1}(x,y_{i_k})$
**end for**

accumulates zeros step by step. Take into account $\mathcal{L}_j\kappa = \mathcal{L}_j s_k + \mathcal{L}_j r_k$ we obviously can conclude that $\mathcal{L}_j s_k$ interpolates $\mathcal{L}_j\kappa$. This leads us to the task of finding a low-rank description of a given block matrix.

Let us consider an admissible matrix

$$A \in \mathbb{R}^{m\times n} \qquad \text{with} \qquad (A)_{ij} = \mathcal{L}_j\kappa(y_i)$$

and a $\eta$-admissible cluster-pair $(D_1, D_2)$. The evaluation of the boundary element integrals gives the matrix entries

$$(\mathcal{L}_j\kappa)(y_i) = \int_{D_1} \kappa(x, y_i)\varphi_j(x)ds_x \qquad j = 1, \ldots, n$$
$$y_i \in D_2 \qquad i = 1, \ldots, m.$$

Thus we use the following Algorithm 4 with $\|\cdot\|_F$ denotes the Frobenius norm.

**Algorithm 4** Fully Pivoted ACA

**let** $A \in \mathbb{R}^{n\times m}$ be a given matrix
**set** $R_0 = A$
**for all** $k = 1, \ldots$ **do**
$(R_k)_{i_{k+1},j_{k+1}} = max_{i,j}|(R_k)_{i,j}|$
$u_{k+1} = R_k e_{j_{k+1}}$
$v_{k+1} = R_k^\top e_{i_{k+1}}$
$\gamma_{k+1} = (R_k)^{-1}_{i_{k+1},j_{k+1}}$
$R_{k+1} = R_k - \gamma_{k+1}u_{k+1}v_{k+1}^\top$
**if** $\|R_k\|_F \leq \epsilon\|A\|_F \rightarrow$ **stop**
**end for**

The approximation matrix $S_k$ can be described by a sum of dyadic products of the vectors $u_i, v_i$

$$S_r = \sum_{i=1}^{r} u_i v_i^\top.$$

This variant of the ACA algorithm is called fully pivoted ACA. It is obvious that each entry of the given matrix $A$ is necessary for the construction of $S_r$. Hence we need $\mathcal{O}(n*m)$ arithmetical operations and that is clearly not efficient. To avoid this drawback we are going to modify the fully pivoted ACA-algorithm and obtain Algorithm 5, this leads us to the so-called partially pivoted ACA.

Finally, we give short results of complexity analysis for the considered adaptive cross approximation technique. For more detailed proofs we refer to [2, 3]. As far as the error estimate is concerned, we are not able to give conclusions in a direct way. Thus, we introduce a system of basis functions which are easy to treat. The simplest choice might be the monom basis

$$\psi_i(x) = x^{\mathbf{i}} = x_1^{i_1} \cdots x_d^{i_d}$$

with $\|\mathbf{i}\|_\infty \leq p-1$, where $d$ is the spatial dimension and $p$ denotes the accuracy of our estimation, see [2]. Moreover we have to assume the tensor-product

$$\zeta_i = (\xi_{i_1}, \ldots, \xi_{i_d})$$

with $\xi_i$ are zeros of Tschebyscheff polynomials.
According to [3] we obtain the following complexity estimates. It is possible

**Algorithm 5** Partially Pivoted ACA

**set** $\hat{i}_0 = 1$
**for all** $k = 1, \ldots$ **do**
  **set** $i_k = \hat{i}_{k-1}$ (or take $i_k \in \{1, \ldots, m\}$)
  with $a_{i_k j} - \sum_{l=1}^{k-1} (u_l)_{i_k} (v_l)_j \neq 0 \quad j = 1, \ldots, n$
  $(\tilde{v}_k)_j = a_{i_k, j} - \sum_{l=1}^{k-1} (u_l)_{i_k} (v_l)_j \quad j = 1, \ldots, n$
  **set** $j_k = argmax_{j=1,\ldots,n} |(\tilde{v}_k)_j|$
  $v_k = (\tilde{v}_k)_{j_k}^{-1} \tilde{v}_k$
  $(u_k)_i = a_{i, j_k} - \sum_{l=1}^{k-1} (u_l)_i (v_l)_{j_k} \quad j = 1, \ldots, n$
  **set** $\hat{i}_k = argmax_{i \neq i_k} |(u_k)_i|$
**end for**
**if** $\|u_r\|_F \|v_r\|_F \leq \epsilon \|S_r\|_F \rightarrow$ **stop**

to show the effort of partitioning the BE-matrix is of order $\mathcal{O}(\eta^{-dim} N_h \log N_h)$ where $dim$ denotes the dimension of the boundary $\Gamma$.
Furthermore, it turns out that the approximation complexity and in fact the overall algorithm effort for a desired accuracy $\|K_{BEM} - \tilde{K}_{BEM}\|_F \leq \epsilon$ is of order $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$ with an arbitrary small $\alpha > 0$.
In conclusion, the memory demand and the cost of one matrix-by-vector multiplication is of the same order as the arithmetical operations to construct a low-rank approximation of the boundary element matrix $K_{BEM}$.

# Chapter 3

# Algebraic Multigrid Method

## 3.1 Motivation

In modern applications multigrid methods are widely used to solve large linear equation systems. We consider the case of solving

$$K_h \underline{u}_h = \underline{f}_h \tag{3.1}$$

with $K_h$ is a symmetric positive definite system matrix. Moreover we usually assume $K_h$ to be sparse in a certain sense. We have to recognize that multigrid methods are not able to solve (3.1) efficient for any symmetric positive definite matrix $K_h$. In fact, the components has to be adapted properly according to the underlying physical problem and variational formulation. Our main objective is to construct an optimal solver, that means linear increase in memory consumption and linear increase in CPU time for a single matrix-by-vector multiplication with respect to the number of unknowns.
In contrast to classical iterative equation solvers (such as Jacobi, Gauss-Seidel and variants) which are performing multiplications with a single matrix only, multigrid methods demands a grid hierarchy or a matrix hierarchy (Figure 3.1). Contrary to geometric multigrid methods, the algebraic multigrid approach assembles this hierarchy only by exploiting the system matrix information corresponding to the finest grid. In order to obtain a 'virtual' grid hierarchy there exists several different coarsening strategies [24, 37, 44, 32]. With corresponding prolongation operators the properties of the fine grid system hands over to a coarse level system.
Multigrid methods are motivated by considering the eigenfrequencies and the eigenvalues of the underlying partial differential equation system. While highly oscillating contributions of the iteration error are treated well on a fine grid, it seems to be clear that for lower eigenfrequencies a coarser grid is sufficient, see Figure 3.2. In general, this separation of the eigenfrequencies provides a fast reduction of the iteration error and in conclusion we obtain fast convergence. Let us remark that geometric multigrid methods are based on an existing grid

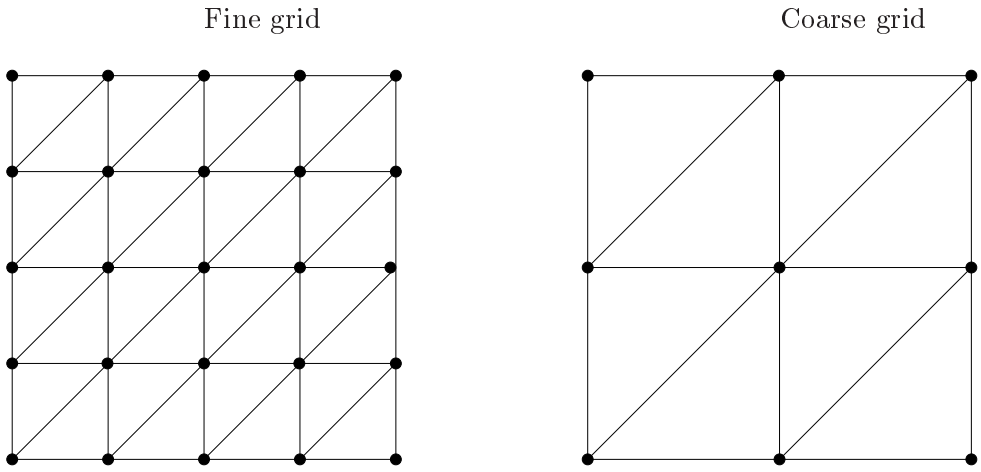



Figure 3.1: Fine - Coarse

hierarchy. Therefore, coarsening procedures need not to be performed and more detailed proofs could be given. More rigorous approaches in geometric multigrid can be found in e.g. [17, 23, 22, 40].

## 3.2 Components of an AMG-Step

In this section we will analyze a single AMG step based on a twogrid algorithm. Fine grid and coarse grid quantities are denoted by indices $h$ and $H$, respectively. The efficiency of AMG depends on a clever interaction of the smoothing process and the coarse grid correction. The twogrid-cycle (see Figure 3.3) describes the essentialist multigrid components which will be discussed in the following. Moreover, in the case of boundary element methods, which provides matrices with essential different properties, a closer consideration of this specific AMG design is refered in Chapter 4.

1. *Smoothing*
   The first and last step of the twogrid-cycle consists in reducing highly oscillating eigenfrequencies of the defect on the finest level. In general, only a few smoothing steps are required (typically 1-3 iterations). In the case of matrices arised from nodal finite element discretization, damped Jacobi or Gauss-Seidel methods (and variations) are appropriate smoothers. Take into account the different properties of the boundary integral operator (single layer potential) and its corresponding matrix operator (in particular the negative order of the pseudo differential operator) another approach will be necessary. In this case we consider the proposal given by Bramble-Leyk-Pasciak (BLP) in [5]. An overview over the BLP-smoother is given in the next chapter. The abbreviation $\mathcal{SM}$ denotes the smoothing

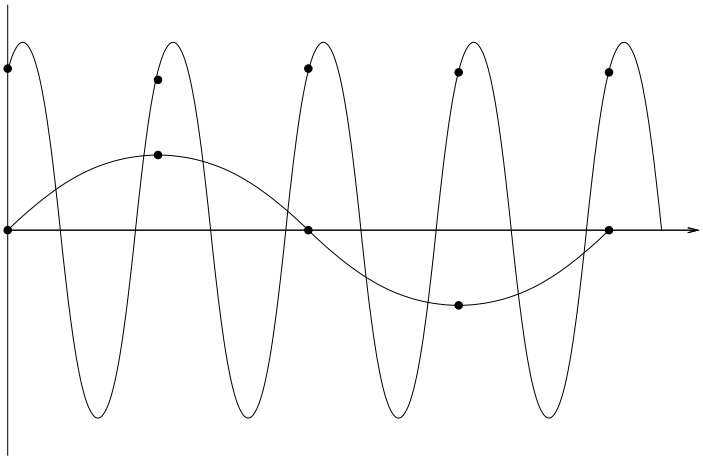


Figure 3.2: Motivation

process at this point of the algorithm.

$$\underline{u}_h = \mathcal{SM}(K_h, \underline{u}_h, \underline{f}_h), \quad \underline{d}_h = \underline{f}_h - K_h \underline{u}_h.$$

2. *Restriction - Prolongation*
A crucial point of AMG is the coarsening strategy in order to get a matrix hierarchy. Most coarsening techniques are based on the matrix graph, see [4, 9, 37, 44]. Moreover, these approaches are developed for sparse matrices arising from finite element discretization. Nevertheless in many applications it will be reasonable to specify an auxiliary matrix $B_h$ in order to perform the coarsing process. It is obvious that if we are faced with fully populated BE-matrices, traditional coarsening strategies will fail in any case . Hence, we are using the general approach proposed in [16, 31] and specify an auxiliary matrix $B_h \in \mathbb{R}^{N_h \times N_h}$ which represents the discretized geometry. Preferable, we are looking for matrices $B_h$ that fulfills M-matrix properties. Since such a choice is possible, we are able to construct prolongation operators with certain features [8, 10, 37, 36]. In order to obtain a coarse system matrix $K_H$ we have to define an appropriate prolongation operator $P_h^K : \mathbb{R}^{N_H} \mapsto \mathbb{R}^{N_h}$, where $N_h$ and $N_H$ denote the number of unknowns on the fine and coarse level, respectively. Because we are using the auxiliary matrix $B_h$, a corresponding prolongation operator $P_h^B$ has to be defined. In addition, a prolongation operator $P_h^K$ is built according to the system matrix $K_h$. Finally, for both matrices adequate restriction operators has to be formulated. Once the transfer operators are constructed, the coarse matrices $K_H$ and $B_H$ become

$$\begin{aligned} K_H &= (P_h^K)^\top K_h P_h^K \\ B_H &= (P_h^B)^\top B_h P_h^B. \end{aligned}$$

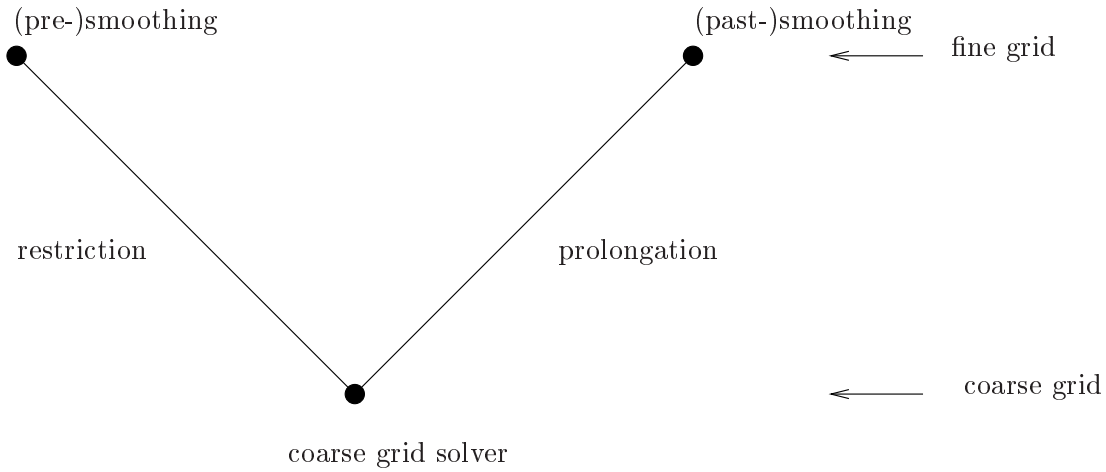



Figure 3.3: Twogrid-Cycle

Based on the general approach proposed in [31, 16] we assume that $B_h$ is a sparse M-matrix which represents the underlying nodal mesh. In particular, each diagonal entry can be related to a grid node and each off-diagonal entry represents a local connection to the neighbors. Due to the local definition of the auxiliary matrix it is possible to introduce the following index sets on a pure algebraic level.
Let $\omega_h$ be the set of unknowns on level $h$, then

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

where $N_h^i$ be the set of neighbors around a node $i \in \omega_h$, further $S_h^i$ the set of strong connections and finally $S_h^{i,T}$ the set of nodes with a strong connection to node $i$. The decision whether a node $i$ represents a coarse node or not will be done by the cut-off function $coarse(B_h, i, j)$, for details see [31] and references therein. Now we are able to apply a standard coarsening process on the auxiliary matrix $B_h$. This matrix will be interpreted as a description of a 'virtual' mesh (Figure 3.4) which can be split into two disjoint sets of grid nodes, i.e.

$$\omega_h = \omega_C \cup \omega_F, \qquad \omega_C \cap \omega_F = \emptyset$$

where $\omega_C$ denotes the set of coarse grid nodes and $\omega_F$ the set of fine grid nodes. Usually the set of coarse grid nodes fulfills two properties,

(a) no coarse grid nodes are connected directly,

(b) the number of coarse grid nodes is as large as possible.

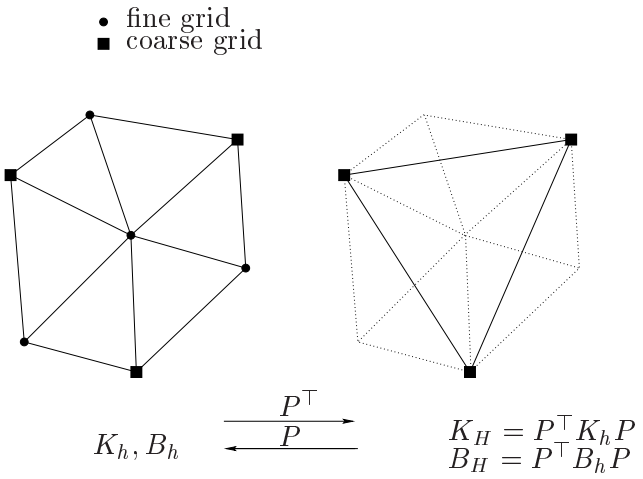



Figure 3.4: Coarsening

Once the set $\omega_C$ was built up, the grid on the coarse level $H$ is defined by

$$\omega_H = \omega_C.$$

In order to construct the coarse auxiliary matrix we use the Galerkin projection method. Therefore, we need a prolongation operator $P_h^B : \mathbb{R}^{N_H} \mapsto \mathbb{R}^{N_h}$ which is constructed by give several rules on the sets $\omega_h$ and $\omega_H$. Let us apply this prolongation operator on the system matrix $K_h$ (i.e. $P_h^K \equiv P_h^B \equiv P_h$). Furthermore, let us mention that Galerkin's method assumes the restriction operator to be the transposed of the prolongation operator $R_h \equiv P_h^\top$, then we have

$$\begin{aligned} K_H &= P_h^\top K_h P_h \\ B_H &= P_h^\top B_h P_h. \end{aligned}$$

Let us remark that the application of an auxiliary matrix will be useful if the system matrix contains no suitable geometric information, for instance in the case of

- application of edge-elements,
- trial functions of higher order,
- anisotropy behavior of the differential operator.

We now consider a concrete realization of the prolongation operator $P_h \in \mathbb{R}_{N_H}^{N_h}$. Therefore we take into account the index sets mentioned above and give the following definition

$$(P_h)_{ij} = \begin{cases} 1 & i = j \in \omega_C \\ \frac{1}{|\omega_C \cap S_h^{i,T}|} & i \in \omega_F, j \in \omega_C \\ 0 & else \end{cases}$$

Hence, we have all ingredients to perform a setup phase in order to build up the matrix hierarchy and the corresponding transfer operators. In the case of BE-matrices several adaptions have to be executed for each multigrid step. More details are given in the next Chapter 4.

The last step to complete the formulation of the equation system on the coarse level $H$ consists in restricting the resiudal $\underline{d}_h$ by using the prolongation operator above, i.e.

$$\underline{d}_H = P_h^\top \underline{d}_h$$

3. *Coarse Grid Correction*
On the coarse level the low frequency contributions of the defect $\underline{d}_H$ will be detected properly. If the number of unknowns on the coarse grid is small enough, it will be reasonable to use a direct solver to get a solution of the equation system on the level $H$. Nevertheless in some cases classical iterative methods (e.g. conjugate gradient algorithm, etc.) may be a suitable routine for solving the coarse grid correction

$$K_H \underline{s}_H = \underline{d}_H \quad \rightarrow \underline{s}_H$$

A recursive application of the smoothing process and the coarse grid correction automatically leads to a hierarchy of matrices and corresponding transfer operators. Thus, a complete V-cycle can be assembled as shown in Algorithm 6. Here $COARSELEVEL$ denotes the coarsest 'virtual' grid level, represented

**Algorithm 6** MG($\underline{u}_l, \underline{f}_l, l$)

**if** l = COARSELEVEL **then**
  define $\underline{u}_l = (K_l)^{-1}\underline{f}_l$ by some direct solver
**else**
  smooth $\nu_F$ times on $K_l\underline{u}_l = \underline{f}_l$
  calculate the defect $\underline{d}_l = \underline{f}_l - K_l\underline{u}_l$
  restrict the defect to the next coarser level $l+1 : \underline{d}_{l+1} = P_l^\top \underline{d}_l$
  set $\underline{u}_{l+1} \equiv 0$
  call MG($\underline{u}_{l+1}, \underline{d}_{l+1}, l+1$)
  prolongate the correction $\underline{s}_l = P_l\underline{u}_{l+1}$
  update the solution $\underline{u}_l = \underline{u}_l + \underline{s}_l$
  smooth $\nu_B$ times on $K_l\underline{u}_l = \underline{f}_l$
**end if**

by the restricted system matrix on the coarsest level. according to a presumed number of unknowns with respect to the restricted system.

At the end of this chapter we provide a brief overview over the advantages of common multigrid methods. Since the multigrid algorithm is an iterative

solving method the main contribution of the memory demand is given by the system matrix $K_h$. Therefore, we asymptotically have to provide $\mathcal{O}(N_h)$ storage units for FE-matrices. Moreover, the overall multigrid process passes through with a complexity of $\mathcal{O}(N_h)$ arithmetical operations.

The CPU time for setting up the matrix hierarchy and for constructing the prolongation operator cannot be neglected for a lower number of unknowns. Thus, we have to take into account that multigrid methods are only efficient for large equation systems. Furthermore, we need to know the underlying variational formulation to adapt an appropriate multigrid algorithm. It is not possible to use them as 'black-box' solver for any regular matrix equation systems.

The advantages and disadvantages discussed in the paragraph above hold for geometric multigrid as well as for algebraic multigrid. Additionally the application of AMG could be necessary in the case of:

- only the finest grid level is available,
- the coarsest grid of a geometric multigrid application is to large for direct solving.

As shown in the last section the coarse grid equation system is built up by using information from the fine system matrix $K_h$ and of course by exploiting an auxiliary matrix $B_h$. The task to perform an efficient coarsening procedure often depends on heuristic assumptions and not at least on plentiful experience.

## 3.3 Algebraic multigrid methods as preconditioner

In this section we are going to present preconditioning techniques for iterative solver. As we know for classical linear iterative methods (i.e. Richardson-algorithm, etc.) the number of iterations $I(\epsilon)$ to reach a certain accuracy $\epsilon$ directly depends on the condition number $\kappa(K_h)$

$$I(\epsilon) = \mathcal{O}(\kappa(K_h) \ln \epsilon^{-1}).$$

In addition we can give some estimate for the condition number $\kappa(K_h)$ exploiting the spectral equivalence constants $\gamma_1, \gamma_2$ from the spectral equivalent inequality which is read as

$$\gamma_1(\underline{v}, \underline{v}) \leq (K_h \underline{v}, \underline{v}) \leq \gamma_2(\underline{v}, \underline{v}) \quad \forall \underline{v} \in \mathbb{R}^{N_h}.$$

It is well known that these constants fulfill the estimates

$$\gamma_1 \leq \lambda_{min}(K_h) \quad \text{and} \quad \gamma_2 \geq \lambda_{max}(K_h)$$

with $\lambda_{min}, \lambda_{max}$ eigenvalues of $K_h$. Thus, we are now able to conclude

$$\kappa(K_h) := \frac{\lambda_{max}(K_h)}{\lambda_{min}(K_h)} \leq \frac{\gamma_2}{\gamma_1}.$$

Provided that $\gamma_1, \gamma_2$ are sharp in their asymptotic behavior, the condition number in fact behaves like $\gamma_2/\gamma_1$, i.e. in the case of a FE-matrix we have $\kappa(K_h) = \mathcal{O}(h^{-2})$ and in the case of a single layer potential operator (and hypersingular operator) we get $\kappa(K_h) = \mathcal{O}(h^{-1})$ with $h$ the typical mesh size. We note that for finer discretization the number of iterations would increase too much. For that reason it is desirable to keep the condition number constant and of course close to one. In order to get a smaller condition number we consider the generalized eigenvalue problem

$$K_h \underline{v} = \lambda C_h \underline{v}$$

with a symmetric positive definite matrix $C_h$. Moreover the spectral equivalent inequality holds

$$\begin{array}{rcccll} \underline{\gamma}(C_h\underline{v},\underline{v}) & \leq & (K\underline{v},\underline{v}) & \leq & \overline{\gamma}(C_h\underline{v},\underline{v}) & \forall \underline{v} \in \mathbb{R}^{N_h} \\ \underline{\gamma}(\underline{v},\underline{v}) & \leq & (C_h^{-1}K_h\underline{v},\underline{v}) & \leq & \overline{\gamma}(\underline{v},\underline{v}) & \forall \underline{v} \in \mathbb{R}^{N_h} \end{array}$$

and thus we obtain the estimate

$$\kappa(C_h^{-1}K_h) \leq \frac{\overline{\gamma}}{\underline{\gamma}}.$$

Now it is simple to find out appropriate properties for the precondition matrix $C_h$ to gain a condition number close to one

- $C_h$ has to approximate the system matrix $K_h$,
- the multiplication $C_h^{-1} * \underline{d}_h$ should happen in almost $\mathcal{O}(N_h)$ operations.

The choice $C_h \equiv K_h$ will satisfy the first condition, whereas a diagonal matrix $C_h$ fulfills the fast matrix-by-vector operation. There exists several different approaches for constructing a preconditioner that approximates $K_h$ and additionally provides a fast matrix-by-vector multiplication, see [27].

In the case of symmetric positive definite matrices the algebraic multigrid operator are mainly implemented as a preconditioner in the conjugated gradient algorithm in order to speedup the convergence rate (Algorithm 7). In every iteration step we have to solve $C_h^{-1} * r_{k+1}$ which is realized by one multigrid cycle.

If the relative error exceeds the stopping criterion we will cancel the iteration procedure. During the FOR-loop we need to calculate the residual in any case, therefore we get the comparative quantity 'for free'. Of course we can

**Algorithm 7** PCG($K_h, u_0, f$)

$r_0 = K_h u_0 - f$
$u_0 = 0$
$\overline{r}_0 = C_h^{-1} r_0$
$p_0 = \overline{r}_0$
$\gamma_0 = (\overline{r}_0, r_0)$
**for all** i=0,1,... **do**
  $z_k = K_h p_k$
  $\alpha_k = -\frac{\gamma_k}{(p_k, z_k)}$
  $u_{k+1} = u_k + \alpha_k p_k$
  $r_{k+1} = r_k + \alpha_k z_k$
  $\overline{r}_{k+1} = C_h^{-1} r_{k+1}$
  $\gamma_{k+1} = (\overline{r}_{k+1}, r_{k+1})$
  if $\gamma_{k+1}$ small enough $\rightarrow$ stop
  $\beta_k = \frac{\gamma_{k+1}}{\gamma_k}$
  $p_{k+1} = \overline{r}_{k+1} + \beta_k p_k$
**end for**

measure the iteration error in the norm $\|.\|_{K_h C_h^{-1} K_h}$. This norm is close to the energy norm provided that the preconditioner $C_h$ is good enough.

The preconditioned CG-algorithm provides the following estimate considering the number of iterations needed in order to obtain a relative accuracy $\epsilon \in (0, 1)$

$$I(\epsilon) = \mathcal{O}(\sqrt{\kappa(C_h^{-1} K_h)} \ln \epsilon^{-1}).$$

It is obvious that the best results would be reached if the condition number will be fixed constant. But a typical behavior of algebraic multigrid methods is a dependence corresponding to the number of multigrid levels [6, 7].

As long as we are working with symmetric positive definite matrices, the PCG-algorithm acts well. But if we are using collocation method, we generally will be faced with non-symmetric matrices. In this case the PCG-algorithm will fail and therefore other Krylov subspace methods [29], such as BiCGStab (BiConjugate Gradients Stabilized method) or GMRES (Generalized Minimal RESidual method), have to be implemented. In our AMG software package we are using the BiCGStab method (Algorithm 8), in particular the preconditioned variant of them.

**Algorithm 8** BiCGStab($K_h, u_0, f$)

$r_0 = f - K_h u_0$
$\overline{r}_0 = r_0$
$\rho_0 = \alpha = \omega_0 = 1$
$v_0 = p_0 = 0$
**for all** i=1,... **do**
  $\rho_i = (\overline{r}_0, r_{i-1})$
  $\beta = \frac{\rho_i}{\rho_{i-1}} \cdot \frac{\alpha}{\omega_{i-1}}$
  $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$
  $\overline{p}_i = C_h^{-1} p_i$
  $v_i = K_h \overline{p}_i$
  $\alpha = \frac{\rho_i}{(\overline{p}, v_i)}$
  $s = r_{i-1} - \alpha v_i$
  $\overline{s} = C_h^{-1} s$
  $t = K_h \overline{s}$
  $\omega_i = \frac{(t,s)}{(t,t)}$
  $u_i = u_{i-1} + \omega_i s + \alpha p_i$
  $r_i = s - \omega_i t$
  if $r_i$ small enough $\rightarrow$ stop
**end for**

# Chapter 4

# AMG designed for BE-Matrices

## 4.1 Smoothing Methods

### 4.1.1 Hypersingular Operator

The smoothing process depends strongly on the type of the boundary integral operator. Since the behavior of eigenvectors and eigenvalues of the hypersingular operator $D : H^{1/2}(\Gamma) \mapsto H^{-1/2}(\Gamma)$

$$(Du)(y) = -\frac{\partial}{\partial n_y}\partial n_y \int_\Gamma \frac{\partial}{\partial n_x} E(x,y)u(x)ds_x$$

is well known it can be observed that $D$ acts as an pseudo differential operator of order plus one. Therefore, traditional FE-smoothers are appropriate choices to perform a smoothing step. Take into account this remark, standard Gauss-Seidel will satisfy this problem class [45].

### 4.1.2 Single Layer Potential Operator

In order to construct a smoothing procedure for the single layer potential operator we have to consider that high frequency eigenfunctions correspond to a small eigenvalue. In particular, we have to introduce a smoother for pseudo differential operators of order minus one. The following idea was suggested by Bramble, Leyk and Pasciak, for details see [5].

Let us consider a symmetric bilinear form

$$\mathcal{V}(.,.) \qquad \text{on } H^{-1/2}(\Gamma)$$

which satisfies the following coercivity and boundness inequalities:

$$c_0\|v\|^2_{H^{-1/2}(\Gamma)} \leq \mathcal{V}(v,v) \leq c_1\|v\|^2_{H^{-1/2}(\Gamma)} \qquad \forall\ v \in H^{-1/2}(\Gamma)$$

with constant $c_0, c_1 > 0$. To overcome the conversely behavior of pseudo differential operators of negative order we assume a certain inner product. This

base inner product should correspond to a weaker norm than that induced by $\mathcal{V}(.,.)$. Hence, it turns out to use the norm in $H^{-1}(\Gamma)$.

Setting up a multigrid cycle will lead us to the following well known ingredients. Assume a nested sequence of finite dimensional inner product spaces

$$\mathcal{M}_0 \subset \mathcal{M}_1 \subset \cdots \subset \mathcal{M}_l \subset H^{-1/2}(\Gamma)$$

and formulate the problem:
Find $u_l \in \mathcal{M}_l$ satisfying

$$\mathcal{V}(u_l, \phi) = f(\phi) \qquad \forall \phi \in \mathcal{M}_l. \tag{4.1}$$

The smoothing operator $R_k : \mathcal{M}_k \mapsto \mathcal{M}_k$ will be defined properly by

$$[R_k\omega, \Theta] = \frac{1}{\overline{\lambda}_k}\langle \omega, \Theta \rangle_{H^{-1}(\Gamma)} \qquad \forall \Theta \in \mathcal{M}_k$$

where $\overline{\lambda}$ denotes a upper boundary for the largest eigenvalue of the operator $\mathcal{V}$ and $\langle .,. \rangle_{H^{-1}(\Gamma)}$ is the inner product of $H^{-1}(\Gamma)$. A practical and rather simple method of calculating the upper boundary $\overline{\lambda}_k$ is given in the further of this subsection. In order to continue the multigrid process we have to define the prolongation operator $P_k : H^{-1}(\Gamma) \mapsto \mathcal{M}_k$. Thus, the projection

$$\langle P_k\omega, \phi \rangle_{H^{-1}(\Gamma)} = \langle \omega, \phi \rangle_{H^{-1}(\Gamma)} \qquad \forall \phi \in \mathcal{M}_k$$

gives us the fundamental of the required transfer operators. On each level $k$ the mapping $C_k$ should represent an approximation of $\mathcal{V}_k^{-1}$. Moreover on the coarsest level an explicite inversion of $\mathcal{V}_0$ is provided

$$C_0 \equiv \mathcal{V}_0^{-1}.$$

Performing the steps mentioned above we get a multigrid algorithm of common knowledge.

**Algorithm 9** V-Cycle

**if** $k = 0$ **then**
  set $C_0 = \mathcal{V}_0^{-1}$
**else**
  $C_k$ is defined in terms of $C_{k-1}$
  let $g \in \mathcal{M}_k$
  $x^1 = R_k g$
  coarse grid correction step:
  $x^2 = x^1 + C_{k-1}P_{k-1}(g - \mathcal{V}_k x^1)$
  finally $C_k g = x^2 + R_k(g - \mathcal{V}_k x^2)$
**end if**

For a further analysis let us consider the multigrid process applied to the problem

$$\mathcal{V}_l v = f.$$

Moreover let $v_k$ be the solution after the k-th multigrid step and in addition let $z_k = v - v_k$ be the error after the k-th multigrid step, respectively. Thus, there exists a simple relationship between the errors on each level

$$z_k = (I - C_l \mathcal{V}_l) z_{k-1}.$$

In fact, the multigrid reduction process is equivalent to the linear preconditioned iterative scheme

$$v_{k+1} = v_k + C_l(f - \mathcal{V}_l v_k) \qquad k = 0, 1, \ldots$$

and therefore the symmetric V-cycle algorithm is obviously described in a notation which is convenient for a more rigorous analysis, see [6].

Now we define the discrete inner products $[.,.]_k$ used in the definition of the smoothing operators $R_k$ above. In particular, it will be described in terms of a difference operator $A_k : \mathcal{M}_k \mapsto \mathcal{M}_k$. We next consider the case of $\mathcal{M}_k$ consists of discontinuous functions. For the sake of simplicity $\Gamma$ is a rectangle in $\mathbb{R}^2$ which is divided in a rectangular mesh $\{\tau_0^i\}$. In order to obtain a grid hierarchy the finer meshes $\{\tau_k^i\}$ immediately can be constructed by breaking each given rectangle $\{\tau_{k-1}^i\}$ into four congruent smaller rectangles. In addition, trial functions in $\mathcal{M}_k$ are defined to be piecewise constant with respect to $\{\tau_k^i\}$. Assume that $c_{ij}$ is the value on the (i,j)-th rectangle we accordingly define $A_k$ to be the symmetric difference operator with

$$(A_k c, c) = h_k^2 \sum_{(i,j)\in\mathcal{N}_k} c_{ij}^2 + \sum (c_{ij} - c_{i,j+1})^2 + \sum (c_{ij} - c_{i+1,j})^2 \tag{4.2}$$

where $\mathcal{N}_k$ contains the indices labeling the rectangles $\{\tau_k^i\}$. The terms in the last two sums only will be included if $(i, j+1)$ and $(i+1)$ are element of the set $\mathcal{N}_k$. Let us note that $A_k$ is the five point difference operator (with appropriate modifications near the boundary).
Based on this considerations we define the discrete inner product $[.,.]_k$ by

$$[v, w]_k = (A_k^{-1} v, w) \qquad \forall v, w \in \mathcal{M}_k. \tag{4.3}$$

Hence, we are able to formulate the following lemma.

**Lemma 4.1.1.** *Let $A_k$ be defined by (4.2) and $[.,.]_k$ be defined by (4.3). Then, there exists positive constant $c_0, c_1$ which are independent of the level $k$ and satisfy*

$$c_0 \|v\|^2_{H^{-1}(\Gamma)} \leq [v, v]_k \leq c_1 \|v\|^2_{H^{-1}(\Gamma)}$$

*Proof.* The proof can be found in [5]. □

Before we are going to present the matrix form of the multigrid cycle (Algorithm 9) let us suggest a common notation for matrices and vectors. Assume, that $N_k = \dim(\mathcal{M}_k)$ and $\phi_k^i$ $i = 1, \ldots, N_k$ denotes the basis functions for $\mathcal{M}_k$, we then have

$$\begin{aligned}(\bar{A}_k)_{ij} &= (A_k\phi_k^i, \phi_k^j) \quad i,j = 1,\ldots,N_k \\ (\bar{\mathcal{V}}_k)_{ij} &= \mathcal{V}(\phi_k^i, \phi_k^j) \quad i,j = 1,\ldots,N_k\end{aligned}$$

Let us remark that $\phi_k^i$ fulfills the scaling condition $(\phi_k^i, \phi_k^j)_0 = \delta_{ij}$ with $\delta_{ij}$ is the Kronecker delta. Finally, let $\bar{P}_k \in \mathbb{R}_{N_k}^{N_{k+1}}$ be a matrix with coefficients $p_k^{ij}$ that satisfy

$$\phi_k^i = \sum_{j=1}^{N_{k+1}} p_k^{ij}\phi_{k+1}^i$$

In matrix terms problem (4.1) will lead to the discrete problem:
Find $\underline{v}$ that satisfies

$$\bar{\mathcal{V}}_l\underline{v} = \underline{f} \tag{4.4}$$

where $\underline{f}$ denotes the vector $f(\phi_l^i)$ $i = 1, \ldots, N_l$. Moreover, $\underline{v}$ is the vector of coefficients $v_1, \ldots, v_{N_l}$ that represent the expansion of the function $v_l$ in the basis $\{\phi_l^i\}$

$$v_l = \sum_{i=1}^{N_l} v_i\phi_l^i.$$

The following Algorithm 10 provides a suitable handling of the multigrid operator $C_l$:

**Algorithm 10** V-Cycle Matrix form

**if** $k = 0$ **then**
  set $\bar{C}_0 = \bar{\mathcal{V}}_0^{-1}$
**else**
  $\bar{C}_k$ is defined in terms of $C_{k-1}$
  let $\underline{g} \in \mathbb{R}^{N_k}$
  $x^1 = \bar{\lambda}_k^{-1}\bar{A}_k\underline{g}$
  $x^2 = x^1 + \bar{P}_{k-1}^{\top}\bar{C}_{k-1}\bar{P}_{k-1}(\underline{g} - \bar{\mathcal{V}}_k x^1)$
  $\bar{C}_k\underline{g} = x^2 + \bar{\lambda}_k^{-1}\bar{A}_k(\underline{g} - \bar{\mathcal{V}}_k x^2)$
**end if**

It is obvious, that in this concrete realization the smoothing process is only performed by a simple matrix-by-vector operation. As we shall see later, the difference operator $\overline{A}_k$ in Algorithm 10 can be exploited to be the auxiliary matrix proposed in Chapter 3. Since this approach is used to realize the coarsening process, it turns out that the auxiliary matrix $B_k$ is some discretization

corresponding to the Laplace-Beltrami operator living on the boundary $\Gamma$ and therefore we are able to set $B_k = \overline{A}_k$ with $B_k$ be the auxiliary matrix on each according level $k$. The Algorithm 10 with the operator $\bar{C}_l$ defines a feasible approximation of the multigrid operator $C_l$, which can be formulated in the following proposition.

**Proposition 4.1.1.** *Let $v \in \mathcal{M}_k$ and $\bar{v}$ denote the vector of coefficients for the expansion of the function $v$ in the basis $\phi_k^i$. Then, $\bar{C}_k \bar{\mathcal{V}}_k \bar{v}$ is the vector of coefficients for the expansion of the function $C_k \mathcal{V}_k v$.*

*Proof.* The proof of this proposition can also be found in [5]. □

In addition we are able to note further relations:

**Corollary 4.1.2.** *For all $v \in \mathcal{M}_j$*

$$\begin{aligned} (\bar{\mathcal{V}}_l(I - \bar{C}_l\bar{\mathcal{V}}_l)\underline{v}) \cdot \underline{v} &= \mathcal{V}((I - C_l\mathcal{V}_l)v, v) \\ (\bar{\mathcal{V}}_l\underline{v}) \cdot \underline{v} &= \mathcal{V}(v, v). \end{aligned}$$

*Proof.* Follows from Proposition 4.1.1. □

As a consequence of Proposition 4.1.1 we are able to take over results instantly from analytic considerations to their discrete realization. In particular, it implies that contraction estimates for $I - C_l\mathcal{V}_l$ and estimates on the condition number $\kappa(C_l\mathcal{V}_l)$ immediately lead to the same results for their matrix counterparts.

1. No-regularity analysis:
   The condition number $\kappa(\bar{C}_l\bar{\mathcal{V}}_l)$ depends on the level $j$ and thus is bounded by $c \cdot l$. Note that a bounded condition number provides a fast convergence of the preconditioned conjugate gradient algorithm.

2. $\mathcal{V}$ is defined by the single layer potential
   In this case the condition number $\kappa(\bar{C}_l\bar{\mathcal{V}}_l)$ is independed by the multigrid level $l$. Therefore the condition number fulfills $\kappa(\bar{C}_l\bar{\mathcal{V}}_l) \leq c$ with a constant $c > 0$.

Moreover, some convergence estimates and further details can be found in [5, 6, 7] and references therein.
Finally let us mention the numerical realization to determine the eigenvector $\bar{\lambda}_k$ in Algorithm 10. Since it represents a damping parameter in the iteration

$$\underline{u}_h = \underline{u}_h + \tau\bar{A}_h(\underline{f} - \bar{\mathcal{V}}\underline{u}_h)$$

we have to ensure the condition $0 < \tau < 1/\bar{\lambda}$. Therefore, we have to solve the corresponding generalized eigenvalue problem

$$K_h\underline{\phi} = \lambda\bar{A}_h^{-1}\underline{\phi} \tag{4.5}$$

and find out the largest eigenvalue $\bar{\lambda}$ on each multigrid level $k$. Here, $\bar{A}_h$ denotes the Laplace-Beltrami operator considered above. In order to get an approximation of $\bar{\lambda}$ we use the following Algorithm 11 on every level.

**Algorithm 11** Preconditioned Gradient Method $(K_h, \underline{v}_h^0)$

$\underline{r}_h = -K_h \underline{v}_h^0$
**for all** $i = 0, \ldots, k$ **do**
  $\underline{s}_h = \bar{A}_h \underline{r}_h^k$
  $\zeta_i = \frac{(\underline{s}_h, \underline{r}_h)}{(K_h \underline{s}_h, \underline{s}_h)}$
  $\underline{v}_h^{k+1} = \underline{v}_h^k + \zeta_i \cdot \underline{s}_h$
  $\underline{r}_h^{k+1} = \underline{r}_h^k - \zeta_i \cdot \underline{s}_h$
**end for**

If we apply this $A_h^{-1}$-preconditioned gradient method to the homogeneous system (see also [28])

$$K_h \underline{v}_h = 0$$

we will be able to give an approximation for the smallest and the largest eigenvalue, respectively. Thus, we have to determine the roots of the quadratic polynomial

$$p_k(\lambda) = (1 - \zeta_k \cdot \lambda) \cdot (1 - \zeta_{k-1} \cdot \lambda) - \delta \tag{4.6}$$

where $\delta^2 = (K_h \underline{v}_h^k, \underline{v}_h^k)/(K_h \underline{v}_h^{k-2}, \underline{v}_h^{k-2})$ and $\underline{v}_h^k$, $\underline{v}_h^{k-2}$ are the iterates provided from Algorithm 11. This procedure is very fast and after starting with a non-trivial initial guess $\underline{v}_h^0$, only few iterations (usually less then 10 steps) are necessary to get a sufficient accuracy. Hence, we can choose an appropriate upper boundary of the largest eigenvalue (i.e. a damping parameter) according to each multigrid level $l$.

## 4.2 Restriction and Prolongation Operators

In this section we want to give an overview over the construction of the transfer operators, which is a very important but also a tricky task to perform reasonable prolongation and restriction operations. As shown in the previous Chapter 3, most coarsening techniques were developed for sparse matrices arising from finite element discretization. Therefore, we have to specify an auxiliary matrix $B_h \in \mathbb{R}^{N_h \times N_h}$ in the case of boundary element matrices.

An appropriate definition of the auxiliary matrix $B_h$ is the so-called nodal distance matrix, which also has the important property to be a M-matrix. This matrix reflects the underlying mesh of the boundary element discretization. And since we are using piecewise constant trial functions in our examples, the

number of unknowns in $B_h$ is related to the number of grid nodes.

$$(B_h)_{ij} = \begin{cases} -\frac{1}{\|e_{ij}\|_0} & \text{if } j \in N_h^i \\ \sum\limits_{k \neq i} \frac{1}{\|e_{ik}\|_0} & \text{if } i = j \\ 0 & else \end{cases}$$

The vector $e_{ij}$ stand for the distance vector between two nodes $x_i$ and $x_j$, i.e. $e_{ij} = x_i - x_j$. In addition, the length of the vector $e_{ij}$ is measured in the Euclidian norm $\|\cdot\|_0$.
Once an auxiliary matrix $B_h$ is available, the construction of the sets

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

according Chapter 3 is possible. Furthermore we know, that prolongation and restriction operators would be realized as follows:

$$(P_h)_{ij} = \begin{cases} 1 & i = j \in \omega_C \\ \frac{1}{|\omega_C \cap S_h^{i,T}|} & i \in \omega_F, j \in \omega_C \\ 0 & else \end{cases}$$

The technique presented in the previous Section 4.1 are independent whether the boundary matrices are described in a sparse manner (e.g. adaptive cross approximation) or not, because the treated system matrix is only needed for single matrix-by-vector multiplications. Because of the specific representation of a boundary matrix by using the ACA method, we are able to implement the prolongation process in a rather simple way. An ACA compressed boundary element matrix $K_h$ can be split into a near-field matrix and a far-field matrix, respectively.

$$K_h = K_h^{far} + K_h^{near},$$

where $K_h^{near}$ is a sparse matrix and $K_h^{far}$ represents the approximation part consisting of $N_B$ low-rank submatrices

$$K_h^{far} = \sum_{i=1}^{N_B} \sum_{j=1}^{r_i} u_j^i \, (v_j^i)^\top \tag{4.7}$$

$r_i$ the rank of the $i-th$ approximated block.

Using the well-known Galerkin method

$$K_H = P_h^\top K_h P_h$$

immediately leads us to

$$K_H = P_h^\top (K_h^{near} + K_h^{far}) P_h.$$

Now we are going to discuss the near-field and far-field contributions separately. Since the near-field matrix is composed of sparse pattern we are able to treat it in a same way as sparse FE-matrices. Consequently, we are applying a sparse matrix-by-matrix multiplication and will obtain the coarse near-field matrix

$$K_H^{near} = P_h^\top K_h^{near} P_h.$$

As we will see, the second part of the approximated system matrix is split up in its skeletons and thus, for every vector $u_j^i$ and $v_j^i$ a matrix multiplication has to be realized

$$\begin{aligned} K_H^{far} &= P_h^\top \sum_{i=1}^{N_B} \sum_{j=1}^{r_i} u_j^i \, (v_j^i)^\top P_h \\ &= \sum_{i=1}^{N_B} \sum_{j=1}^{r_i} P_h^\top u_j^i \, (P_h^\top v_j^i)^\top. \end{aligned}$$

After the substitution

$$s_j^i = P^\top u_j^i \qquad t_j^i = P^\top v_j^i$$

we finally get

$$K_H^{far} = \sum_{i=1}^{N_B} \sum_{j=1}^{r_i} s_j^i \, (t_j^i)^\top.$$

Thus, the approximated coarse system matrix $K_H$ has also a near-field contribution $K_H^{near}$ and a far-field matrix $K_H^{far}$. Due to the preserving of representation (4.7) for the coarse counterpart $K_H^{far}$ we can exploit some advantages in our numerical realization of the coarsening process. In particular, in our implementation we are using the same ACA datastructures for the fine grid boundary element matrix $K_h$ and the corresponding matrix $K_H$ on the coarser level.

## 4.3 Coarse Grid Solver

In this last section, we briefly treat the possibilities to solve

$$K_H \underline{u}_H = \underline{f}_H$$

on the coarsest multigrid level. In order to get best coarse grid correction, the application of a direct solver would be the simplest method for realization. Standard algorithms like $LU$-decomposition, $LL^\top$-decomposition and similar procedures are convenient methods. Nevertheless, in some cases iterative methods are indispensable. Since we have no explicite description of our sparse ACA boundary element matrix, we are not able to built the inverse matrix $K_H^{-1}$ directly. Therefore, we apply the conjugate gradient method, which provides a sufficient accuracy after only a few iteration steps. It is clear, that for iterative solvers only a simple matrix-by-vector multiplication has to be provided, and

**Algorithm 12** CG-Solver

$d_H = f_H - K_H u_H$
**for** j=0,...,k **do**
  $\alpha_j = \frac{w_H, d_H)}{(K_H s_H, s_H)}$
  $u_H = u_H + \alpha_j s_H$
  $d_H^{j+1} = d_H^j - \alpha_j s_H$
  $w_H^{j+1} = d_H^{j+1}$
  $\beta_{j+1} = \frac{w_H^{j+1},)}{w_H^j, d_H^j)}$
  $s_H = w_H^{j+1} + \beta_{j+1} s_H$
  **if** $(w_H^j, d_H^j) < \epsilon^2 (w_H^0, d_H^0)$ **then**
    stop
  **end if**
**end for**

that is fortunately an available operation for our ACA matrices. It is worth to mention, that for a rigorous consideration of our AMG method we have to assume a linear iterative solver (e.g. Richardson or Tschebyscheff algorithm), and the CG-method is obviously non-linear. Another possibility to overcome the drawback of non-linearity is to suppose $N_H$ iteration steps of the CG-algorithm. In this case we know, that the CG-method will act as a kind of a direct solver. It is also possible to convert the coarse ACA matrix into a common non-compressed representation, that would allow us to apply a direct solver again.

## 4.4 Complexity Analysis

The complexity of an iterative solver for linear systems of algebraic equations is usually determined by analyzing the cost of one matrix-by-vector multiplication. For boundary element matrices without any compression method we are faced with dense matrices. This leads to an effort of $\mathcal{O}(N_h^2)$ arithmetical operations for one single matrix multiplication. Our aim of this section is to show that the overall cost of constructing the proposed AMG preconditioner is also of this order. Furthermore, it turns out that in the case of low-rank matrices the effort to build up the preconditioner is bounded by the cost of the matrix-by-vector multiplication too. Let us first consider the case of classical BEM, which provides fully populated matrices.

1. Smoother: The smoothing process consists essentially of one multiplication by $K_h$ and one multiplication by the auxiliary matrix $B_h$. Since $B_h$ is sparse it only requires $\mathcal{O}(N_h)$ arithmetical operations. The algorithm for calculating the damping parameters also contains one multiplication by $K_h$ and a bounded number of multiplications by $B_h$. There-

fore, the effort for the smoothing procedure is mainly determinated by $\mathcal{O}(K_h * \underline{d}_h) = \mathcal{O}(N_h^2)$. Let us remark that the damping parameter has to be evaluated only once in the overall solving procedure.

2. Prolongation (Restriction): The transfer operators are defined locally and therefore the application of them is of order $\mathcal{O}(N_h)$.

3. Galerkin: If we are performing the Galerkin method, we will pass through the most critical part with respect to the number of arithmetical operations. But because of the sparse prolongation operator the needed matrix-by-matrix multiplication can be done in $\mathcal{O}(N_h \cdot N_H)$ and that is clearly less then $\mathcal{O}(N_h^2)$.

If we now consider the case of adaptive cross approximation, the cost of a matrix-by-vector multiplication will decrease to $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$, see Section 2.4.1. Thus, the complexity of the AMG-preconditioner becomes:

1. Smoother: The essential operation is the multiplication by $K_h$ as considered above, here $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$.

2. Prolongation (Restriction): There is no difference compared to the case of dense boundary element matrices.

3. Galerkin: Since we are able to perform a matrix-by-vector multiplication in $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$ operations and moreover the prolongation operator is a sparse matrix, the computation of the coarse grid matrix can be done at least in $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$. Let us remark, that the Galerkin prolongation method performed for ACA matrices contains some potential for speeding-up the construction of the coarse system matrix. Several considerations in reducing the number of blocks on coarser levels could be thought. But in this thesis we only implemented a straight forward coarsening strategy, which preserves the number of admissible block on each multigrid level, which probably not necessary.

We note that the proposed AMG method is of order $\mathcal{O}(N_h^2)$ in the first case of dense matrices and of order $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$ in the second case of sparse representation as $h$ tends to zero. Furthermore the estimates considered in the last paragraphs holds for coarser levels, because in each row of the auxiliary matrix the number of non-zero entries is bounded from above for a lot of practical examples.
Finally, we will analyze the memory demand considering the appearing matrices in the AMG algorithm. Based on the assumption that the number of unknowns will be sufficiently reduced during the coarsening process, the same arguments holds as for the cost of arithmetical operations. Thus, we have essentially:

1. $K_h$: Depending on whether we are treating sparse or fully populated boundary element matrices, we have to accept an effort of $\mathcal{O}(N_h^2)$ or $\mathcal{O}(\epsilon^{-\alpha} N_h^{1+\alpha} \log N_h)$, respectively.

2. $B_h$: Since $B_h$ is a sparse matrix the memory demand is of order $\mathcal{O}(N_h)$.

What we have shown in this chapter was an appropriate design for an algebraic multigrid preconditioner for boundary element matrices. Independent of the concrete description of $K_h$ we could construct an almost optimal AMG preconditioner, i.e. the number of arithmetical operations is of the same order as the cost of one matrix-by-vector multiplication. Based on this results, we are now going to present some numerical experiments.

# Chapter 5

# Numerical Studies

In order to show the efficiency of the suggested AMG approach we present some results in 2D. The boundary element matrices are generated by the collocation method with piecewise constant trial functions, as well for Dirichlet data $u_h$ as for Neumann data $v_h = \frac{\partial u_h}{\partial n}$. Moreover, in the case of sparse BE-matrices a simple ACA-algorithm was applied. The AMG preconditioner is realized within the software package PEBBLES [30]. The proposed technique is used as a preconditioner $C_h$ in the PCG algorithm and in the preconditioned BiCGStab algorithm, respectively (see [22]). In particular, we use the symmetric $V(1,1)$-cycle, i.e. one pre-smoothing and one post-smoothing step per iterative cycle. The iteration error is measured in the $K_h C_h^{-1} K_h$ energy norm for applications with the PCG solver and the defect-test is used in the BiCGStab solver.

Unless otherwise declared, all examples are accomplished with a stopping criterion of $10^{-6}$ with respect to the relative error, the iterative solution process starts with a random initial guess $\underline{u}_h^0$ and the right-hand-side $\underline{f}_h = 0$. Furthermore, all calculations are done on a PC with 1800MHz AMD Athlon(tm) processor.

## 5.1 Interior Laplace Equation in 2D

Let us consider the interior Laplace's equation.

$$\begin{aligned} -\Delta u(x) &= 0 \qquad && x \in \Omega \\ u(x) &= g(x) && x \in \Gamma \end{aligned}$$

As shown in Chapter 2 this problem reduces to the task of solving

$$V_h \underline{v}_h = \underline{f}_h$$

with

$$(V_h)_{ij} = \int_{\Gamma_j} E(x, y_i) ds_x.$$

### 5.1.1 Disk 2D

First we consider $\Omega \subset \mathbb{R}^2$ to be a disk with radius 0.1 and thus, the boundary $\Gamma = \partial\Omega$ is the circle with the same radius (Figure 5.1). Assuming such a diameter (actually diam $\Omega < 1$) the single layer potential operator is positive definite. This simple geometry combined with an uniform discretization leads to a symmetric BE-matrix, which is presumed for using the fast PCG-method. The collocation points are choosed as the middle of the direct connection of two single grid nodes.

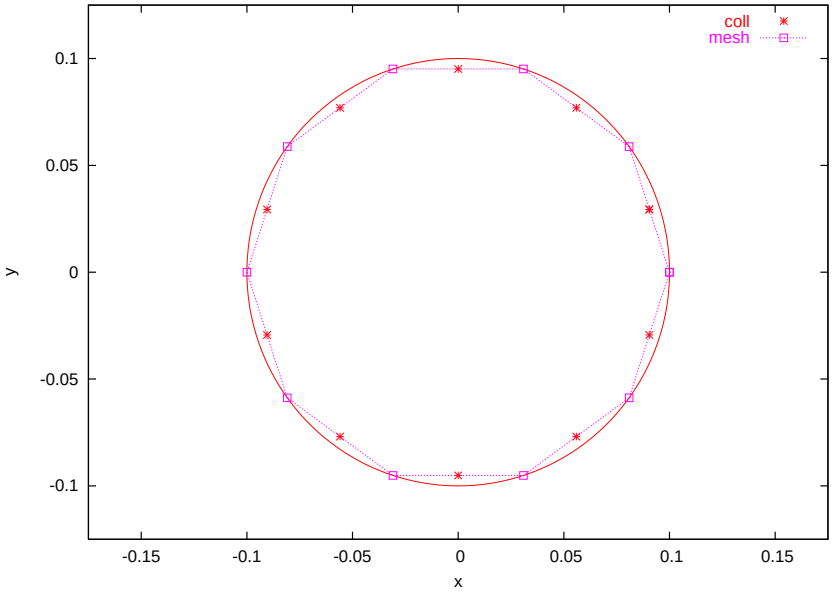



Figure 5.1: Circle

The traditional application of boundary element methods yields a fully populated matrix. The first experiment deals with this dense matrices. As we know, the theory predicts an increase for the arithmetical cost of order $\mathcal{O}(N_h^2)$ for a single matrix-by-vector multiplication. In order to show the efficiency of our AMG preconditioner we should notice the same increase of effort in the overall solving algorithm. As expected we can observe in the first table (Table 5.1) the quadratic increase as well for the setup time as for one single CG-step.

With this result we may conclude, that our solver keeps the effort $\mathcal{O}(N_h^2)$ of one multiplication with a dense matrix. In the next table (Table 5.2) we are comparing the condition number $\kappa(C_h^{-1}K_h)$ obtained either by our BLP-smoother or by the standard Gauss-Seidel smoother. One can notice, that in the case of the BLP-smoother we are concerned with condition numbers close to one. On the other hand the Gauss-Seidel smoother is clearly not optimal for

| Number of Unknowns | setup (sec) | CG-cycle (sec) |
|---|---|---|
| 1024 | 0.60 | 0.09 |
| 2048 | 2.46 | 0.36 |
| 4096 | 10.04 | 1.42 |
| 8192 | 45.98 | 5.78 |

Table 5.1: CPU times for the AMG preconditioned CG-algorithm

the single layer potential. We point out a larger condition number and even a slight increase of them with respect to the number of unknowns.

| Number of Unknowns | Gauss-Seidel | Bramble-Leyk-Pasciak |
|---|---|---|
| 1024 | 3.01 | 1.03 |
| 2048 | 3.27 | 1.03 |
| 4096 | 3.53 | 1.03 |
| 8192 | 3.79 | 1.03 |

Table 5.2: Condition number $\kappa(C_h^{-1}K_h)$

The larger condition number is also reflected in the convergence rate. In Figure 5.2 we consider both smoothers for the single layer potential with 4096 discretization nodes. We will notice, that the Gauss-Seidel smoother causes at least two times more iterations to reach the same accuracy as needed in the case of the Bramble-Leyk-Pasciak smoother.

In most BEM-software packages the inversion of boundary element matrices will be executed by using some direct solver. Therefore, we would like a comparison between a direct solver and the AMG preconditioned iterative solver. In Table 5.3 we are considering a direct solver and for the sake of simplicity we are choosing the implemented solver, which is also used for coarse grid correction in our AMG code. In our case we are using the SuperLU solver [14], knowing that for this purpose several more efficient direct solvers are existing.

Although we still note the quadratic increase of CPU time for the AMG based CG-algorithm, it is much faster than a direct inversion of the BE-matrix. As we can see in the first column, the typical rate of $\mathcal{O}(N_h^3)$ occurs during the increment of the unknowns.

### 5.1.2 L-shape 2D (Comparision)

Of course, the disk is a very simple geometry, therefore in the next example we are faced with $\Omega \subset \mathbb{R}^2$ a kind of a L-shape domain, see Figure 5.3. Once again

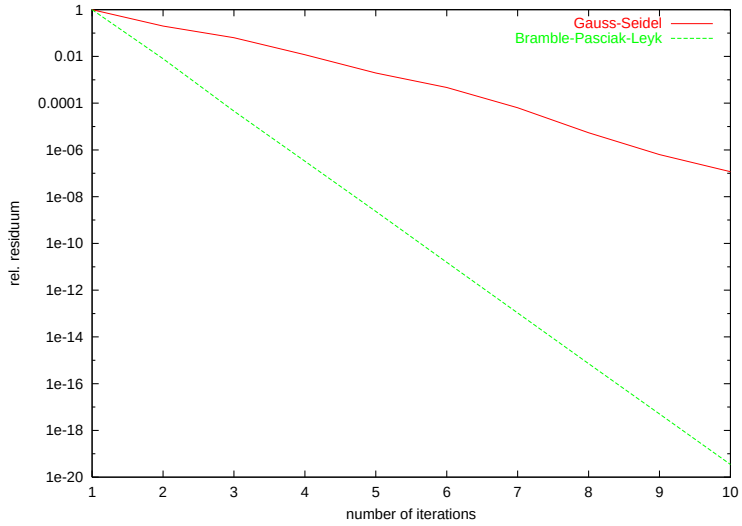



Figure 5.2: Convergence

| Number of Unknowns | direct (sec) | AMG-PCG (sec) |
|---|---|---|
| 1024 | 29.2 | 0.96 |
| 2048 | 279 | 3.88 |
| 4096 | 3342 | 15.71 |
| 8192 | ∼10h | 69.07 |

Table 5.3: Overall solving time

we assume diam $\Omega < 1$ in order to obtain a positive definite BE-matrix.

Due to this specific geometry and the collocation discretization method we are concerned with a non-symmetric BE-matrix. Thus, the conjugate gradient method cannot be applied and we consequently we have to perform some Krylov subspace methods. In particular, we implemented the BiCGStab algorithm for our following examples.

In order to show the efficiency of our AMG based preconditioner, it would be advantageously to compare our method with other preconditioners by means of the benchmark example Figure 5.3. In [46] we can found some results for the Laplace equation with Dirichlet boundary conditions. The solver used therein is a conjugate gradient algorithm, preconditioned with the hypersingular operator. Thus, we are going to direct our attention to the comparision between these two preconditioners.

We have to remark that the comparision can be done only to a certain extent, since there are some different approaches for the discretization and the according trial functions. In particular, the system matrix of [46] arised

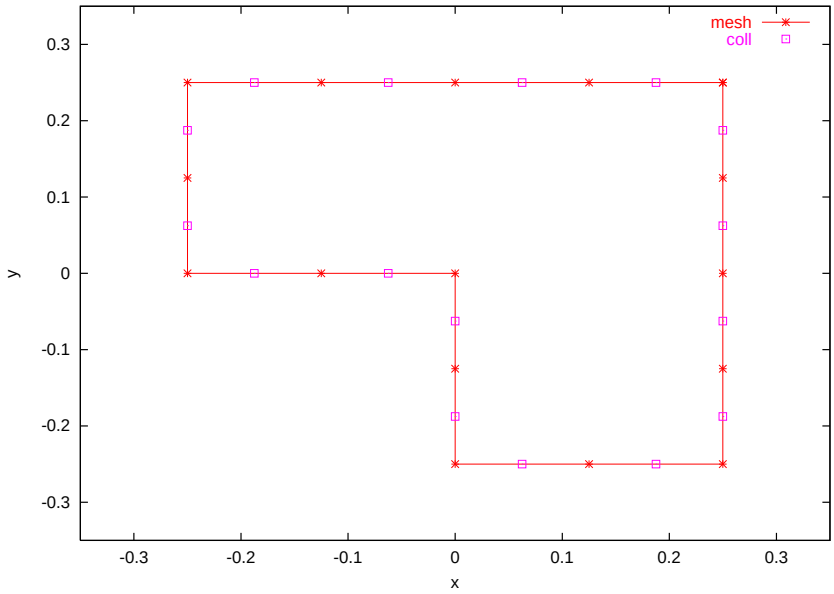



Figure 5.3: L-Shape

from the Galerkin method, that provides a symmetric, positive definite matrix. Moreover, the trial functions for the Dirichlet data $u$ are assumed to be piecewise linear and for the Neumann data $\frac{\partial u}{\partial n}$ piecewise linear, non-conform. On the contrary, for our AMG preconditioned solver we are using boundary element matrices originated from the collocation method with according piecewise constant trial functions for Dirichlet data $u$ and Neumann data $\frac{\partial u}{\partial n}$. In the following calculations the relative accuracy is set to $\epsilon_{rel} = 10^{-8}$ and the evaluation is done at the coordinate $x^* = (0.01, 0.01)$.

The first instance discusses the case of a linear Dirichlet boundary condition $g(x) = 4(x_1 - x_2)$. We can note immediately the resulting piecewise constant Neumann data with the assigned values 4 or $-4$, see Figure 5.4 and Figure 5.5. Since the resulting Neumann data are included in the trial space, we may expect results independent from the characteristic mesh size $h$, and as we can see in Table 5.4 and Table 5.5 the result in $x^* = (0.01, 0.01)$ remains almost constant.

Taking a look on the number of iterations, we clearly observe a smaller number of iterations in the case of the AMG preconditioner. Compared with the hypersingular operator preconditioner, the number of iterations was reduced essentially. An interesting observation can find in the behavior of the $L_2$-error of the Neumann data. Considering the Galerkin boundary matrices we notice constant $L_2$-error as expected. But in the case of collocation matrices, the error decreases with $\mathcal{O}(h^{1/2})$.

| Number of Unknowns | Iterations | Setup (sec) | Cycle (sec) | Solution (sec) | $L_2$-error of $\frac{\partial u}{\partial n}$ on $\Gamma$ | $\lvert u(x^*) - u_h(x^*)\rvert$ |
|---|---|---|---|---|---|---|
| 128 | 3 | 0 | 0.007 | 0.02 | 1.98e-1 | 1.28e-11 |
| 256 | 3 | 0 | 0.06 | 0.18 | 1.40e-1 | 3.39e-11 |
| 512 | 4 | 0.61 | 0.12 | 1.09 | 9.92e-2 | 4.41e-10 |
| 1024 | 4 | 5.12 | 0.23 | 6.05 | 7.02e-2 | 3.32e-10 |

Table 5.4: Benchmark 1: Solver ($C_h$ : AMG)

| Number of Unknowns | Iterations | Matrix (sec) | Solution (sec) | $L_2$-error of $\frac{\partial u}{\partial n}$ on $\Gamma$ | $\lvert u(x^*) - u_h(x^*)\rvert$ |
|---|---|---|---|---|---|
| 128 | 13 | 2.00 | 0.15 | 2.15e-5 | 2.44e10 |
| 256 | 14 | 8.12 | 0.71 | 3.80e-6 | 2.57e-11 |
| 512 | 14 | 32.95 | 2.74 | 1.41e-6 | 4.52e-10 |
| 1024 | 15 | 131.94 | 11.32 | 2.88e-5 | 2.19e-10 |

Table 5.5: Benchmark 1: Solver ($C_h$ : Hypersingular Operator)

The second numerical example deals with the Dirichlet condition $g(x) = \log|x-y|$, $y \notin \overline{\Omega}$. Obviously, the corresponding solution for the Neumann data is $g_1(x) = \frac{(x-y,n)}{|x-y|^2}$ with $n$ denotes the outward normal vector. Once again, the number of iterations for the AMG based solver is clearly smaller than in the hypersingular case. Considering the $L_2$-norm of the Neumann data, we can observe a decrease of almost order $\mathcal{O}(h^{3/2})$ for the Galerkin approach. In the case of our collocation method the $L_2$-error is reduced in $\mathcal{O}(h^{1/2})$ again. More convergence estimates and their theoretical background can be found in e.g. [12, 42].

| Number of Unknowns | Iterations | Setup (sec) | Cycle (sec) | Solution (sec) | L2-error of $\frac{\partial u}{\partial n}$ on $\Gamma$ | $\lvert u(x^*) - u_h(x^*)\rvert$ |
|---|---|---|---|---|---|---|
| 128 | 3 | 0 | 0.003 | 0.01 | 1.82e-2 | 2.85e-4 |
| 256 | 3 | 0 | 0.06 | 0.18 | 1.29e-2 | 9.09e-5 |
| 512 | 4 | 0.61 | 0.128 | 1.08 | 9.08e-3 | 2.88e-5 |
| 1024 | 4 | 4.69 | 0.22 | 5.58 | 6.41e-3 | 9.07e-6 |

Table 5.6: Benchmark 2: Solver ($C_h$ : AMG)

| Number of Unknowns | Iterations | Matrix (sec) | Solution (sec) | L2-error of $\frac{\partial u}{\partial n}$ on $\Gamma$ | $\lvert u(x^*) - u_h(x^*)\rvert$ |
|---|---|---|---|---|---|
| 128 | 13 | 2.00 | 0.15 | 2.39e-2 | 3.09e-4 |
| 256 | 14 | 8.12 | 0.71 | 7.82e-3 | 7.49e-5 |
| 512 | 14 | 32.95 | 2.74 | 2.65e-3 | 1.85e-5 |
| 1024 | 15 | 131.94 | 11.32 | 9.15e-4 | 4.62e-6 |

Table 5.7: Benchmark 2: Solver ($C_h$ : Hypersingular Operator)

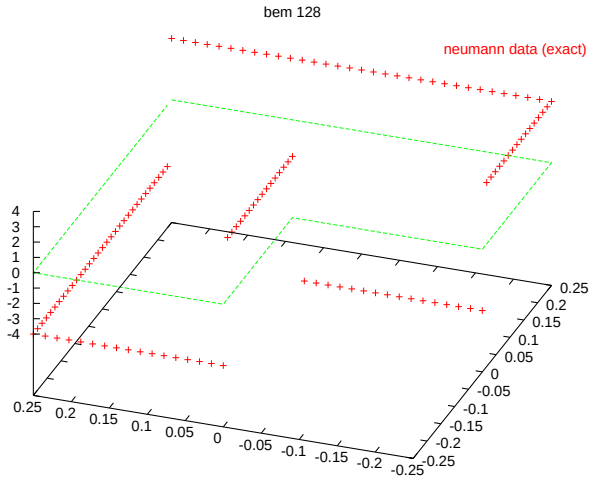



Figure 5.4: Benchmark 1: Neumann Data (Exact)

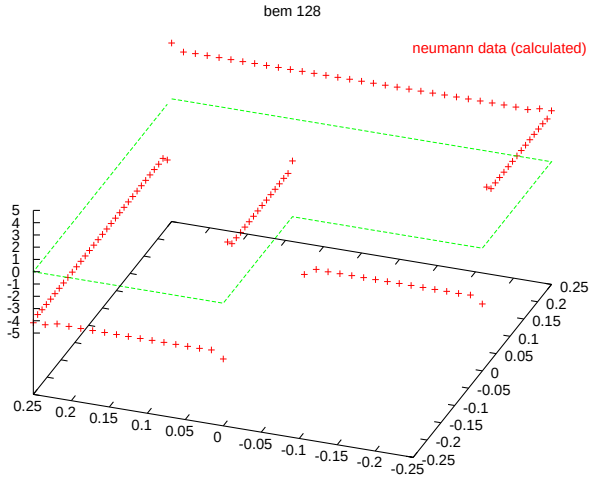



Figure 5.5: Benchmark 1: Neumann Data (Calculation)

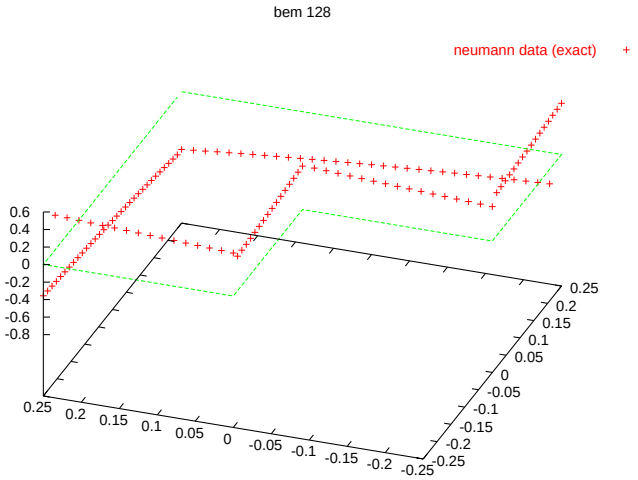



Figure 5.6: Benchmark 2: Neumann Data (Exact)

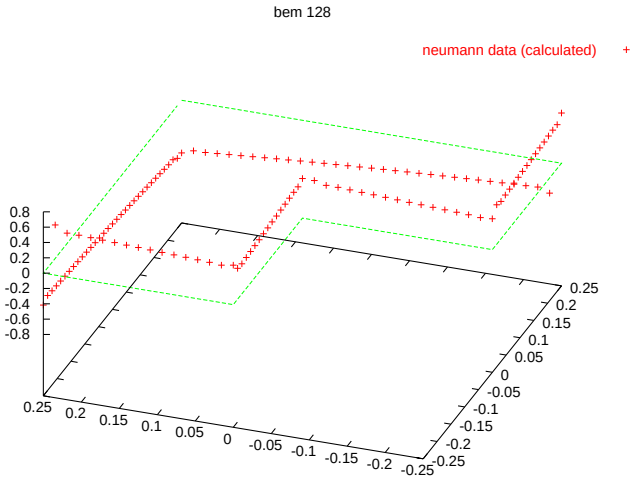



Figure 5.7: Benchmark 2: Neumann Data (Calculation)

## 5.2 Experiments with ACA - matrices

### 5.2.1 Disk 2D

As shown for fully populated matrices, the CPU time for solving an equation system increases by $\mathcal{O}(N_h^2)$. This dramatically increase of CPU time (and of course memory demand) makes clear, that practical applications essentially need some sparse approximation methods for boundary element matrices. In the rest of this chapter all further calculations will be performed with approximated matrices.
Now we are going to demonstrate some first experiments with low-rank matrices provided by the ACA-algorithm. Therefore, we are again starting with the disk example (see Figure 5.1). With this rather simple geometry we are able to present some impressive results.
First of all, we shall do a comparison with the results gained from the fully populated boundary element matrices in the previous section. In Table 5.8 the quantities for setup and a single PCG-cycle will reflect the efficiency of sparse approximation methods.

| Number of Unknowns | setup (sec) full BEM | setup (sec) ACA BEM | PCG-cycle (sec) full BEM | PCG-cycle ACA BEM |
|---|---|---|---|---|
| 1024 | 0.60 | 0.56 | 0.09 | 0.05 |
| 2048 | 2.46 | 1.19 | 0.36 | 0.12 |
| 4096 | 10.04 | 2.40 | 1.42 | 0.26 |
| 8192 | 45.98 | 5.54 | 5.78 | 0.55 |

Table 5.8: Comparison: sparse approximated BE-matrix

In contrast to resulting issues of full BEM examples, we are able to observe the almost linear increase of time for the setup process and for one single iteration step. In the further we only focus on analyzing the algebraic multigrid method for sparse BE-matrices. In fact, we can increment the number of unknowns essentially, even up to the factor ten.
In the first Figure 5.8 we show the length of time to construct the matrix hierarchy and the prolongation operators. This step includes the most critical point of building the AMG preconditioner with respect to CPU time. There is a slight discrepancy to linear behavior, which could be explained by some inefficient numerical implementation. The procedure of multiplying the BE-matrix with the sparse prolongation operators yields in a multiplication of a sparse matrix times a sparse vector. It could be possible to speed-up this step in the algorithm. Nevertheless the setup of the preconditioner is of a better order then $\mathcal{O}(N_h^2)$.
One single AMG preconditioned CG-iteration is demonstrated in Figure 5.9.

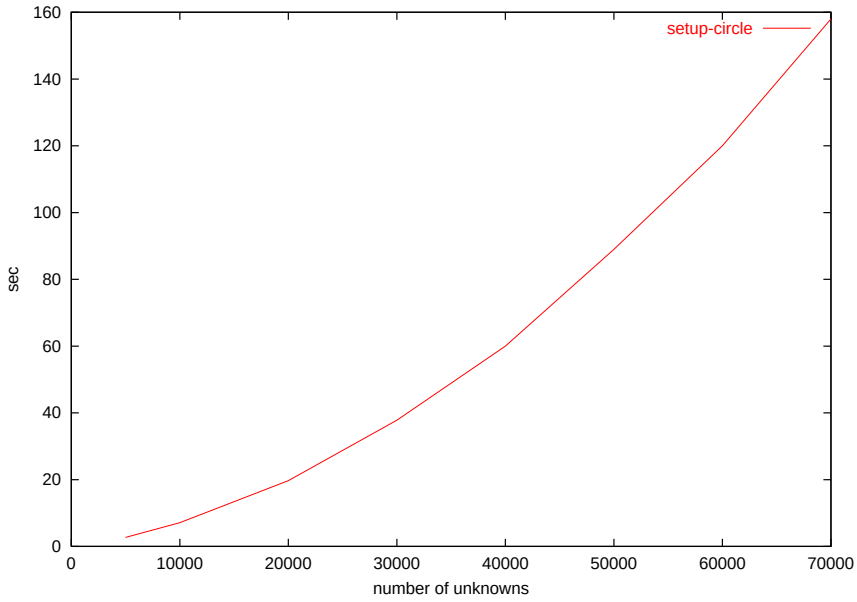



Figure 5.8: Circle: Setup

As we would expect we can observe linear increase of time for one preconditioned CG-iteration.

Finally, we present the condition numbers for the AMG preconditioned single layer potential operator using a BLP-smoother. As almost shown in the case of full BEM (Table 5.2), we again can demonstrate the efficiency of our preconditioner, see Table 5.9.

| Number of Unknowns | BPL-smoother |
|---|---|
| 20000 | 1.05 |
| 60000 | 1.08 |
| 100000 | 1.13 |

Table 5.9: Condition number $\kappa(C_h^{-1}K_h)$, Disk, ACA

### 5.2.2 L-shape 2D

Similar to the organization of the resulting graphs of the disk example above, let us now consider Figure 5.10 and Figure 5.11. Hence, in the graph showing the setup time we can notice the same discrepancy to a linear function as in the case of the disk. But once more a single iteration step of the BiCGStab algorithm shows the optimal linear increase of calculation time with respect to

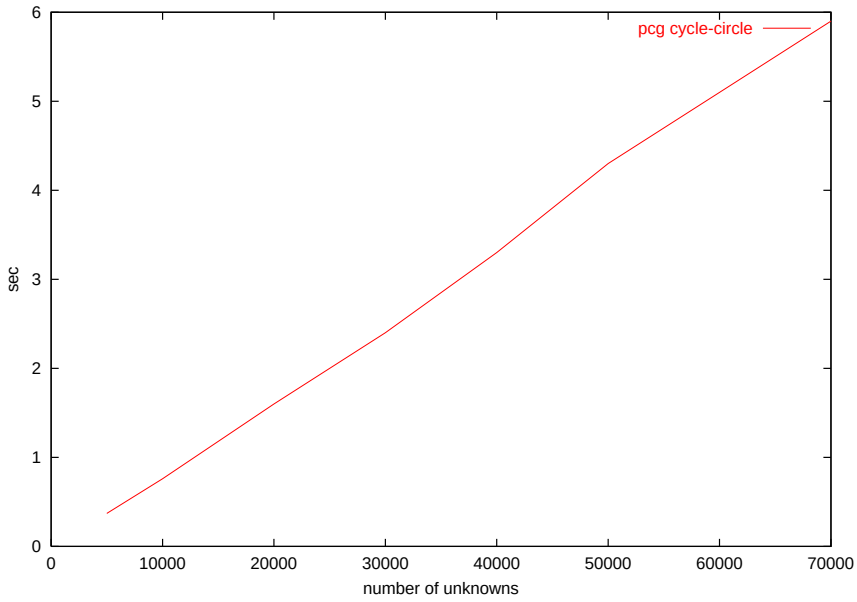



Figure 5.9: Circle: PCG-Cycle

the number of unknowns.

It is well known that the number of iterations which is necessary to exceed a certain accuracy depends on the condition number $\kappa(C_h^{-1}K_h)$. Since we are faced with a non-symmetric matrix $K_h$, a direct calculation of the condition number is not possible any longer. Thus, we consider the number of iterations of the BiCGStab algorithm to reach a given accuracy. If the iterations will keep constant, the preconditioner would be optimal. Therefore, we take a closer look to Table 5.10, which in fact points out almost constant iteration numbers for the L-shape.

| Number of Unknowns | Iterations |
|---|---|
| 10000 | 3 |
| 30000 | 4 |
| 50000 | 4 |
| 70000 | 6 |

Table 5.10: Number of iterations, L-shape, ACA

These experiments, both the disk and the L-shape, give a first impression about the efficiency of a well adapted algebraic multigrid preconditioner for boundary element matrices. Additionally to the resulting solving time we actually have to add the time for calculating the boundary element matrix, the

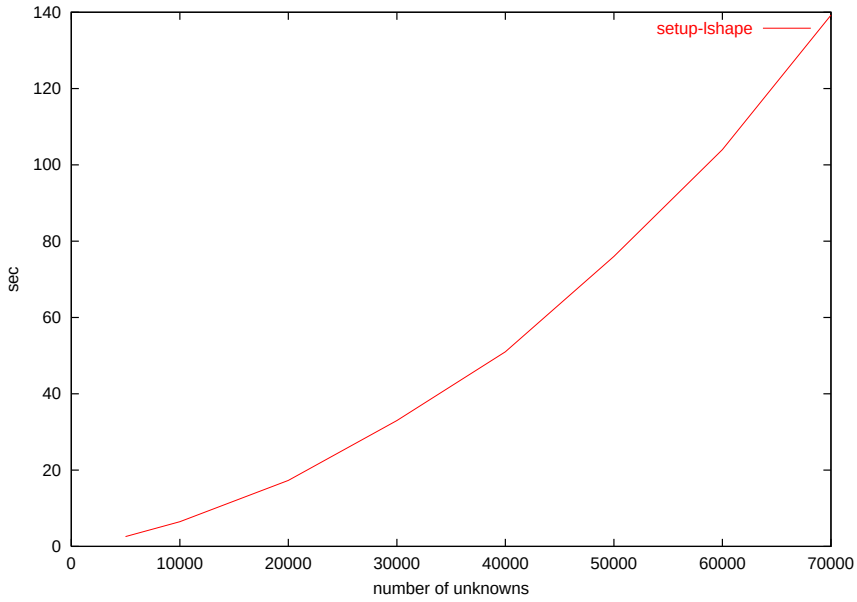



Figure 5.10: L-shape: Setup

dense matrix as well as the approximated low rank matrix. But this times are not so important, because our examples were performed for the purpose of reflecting the efficiency of the solving strategy only.

## 5.3 Sparse/Dense BE-Matrices (Comparison)

In this last section we would like to show some general results concerning the approximation of the dense boundary element matrices. In order to show the big advantages of using the sparse representation of the BE-matrices provided by the adaptive cross approximation algorithm, we consider one single matrix-by-vector multiplication. In Figure 5.12 we give a comparison between fully populated BE-matrices and ACA-matrices.

One might notice the quadratic increase in the length of CPU time using dense BE-matrices. In contrary we observe the linear increase in the case of sparse ACA-matrices. Moreover it turns out that there is a certain moment where the rate of time efficiency alternates. A lower number of unknowns let the dense matrix by vector multiplication be reasonable, a higher number makes the ACA-method suitable.

The reason for the linear behavior of a matrix-by-vector multiplication of an approximated matrix is directly connected with the linear increase of generated blocks by the ACA algorithm. Considering the ACA-algorithm we take a look on the number of generated admissible and non-admissible blocks for both

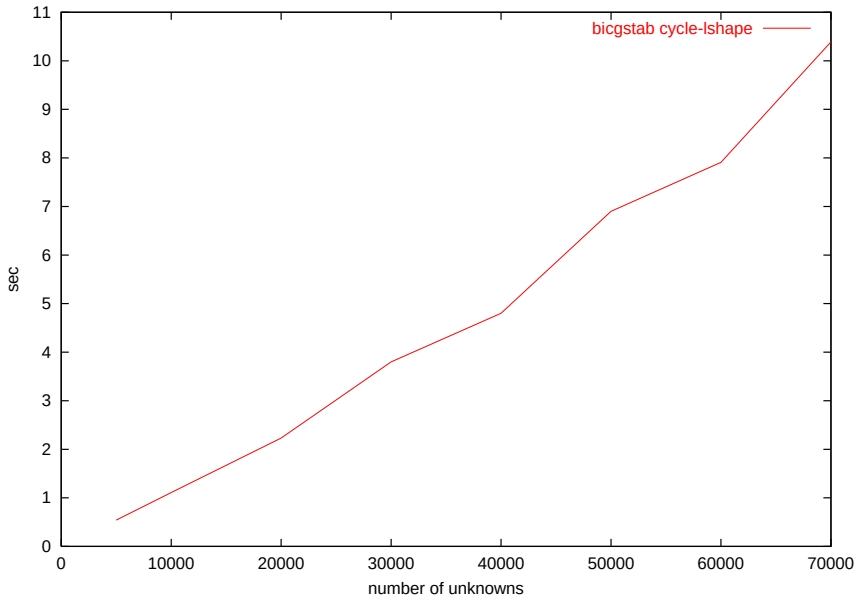



Figure 5.11: L-shape: BiCGStab-Cycle

examples (Figure 5.13 and Figure 5.14).

This number strongly depends on some adjustments in the clustering subroutines of the ACA-algorithm and the underlying geometry. As mentioned in Section 2.4.1 the parameter $\eta$ controls the admissibility of an arbitrary clusterpair. A smaller parameter $\eta$ leads to a larger spatial distance (see Definition 2.4.1) and therefore to a more accurate approximation. Of course the efficiency of a matrix-by-vector operation increases by choosing a larger parameter, because more admissible blocks are generated. Thus, it has to find out a appropriate 'trade-off' between those two properties. As discussed in Section 2.4.1 an improved ACA-algorithm obviously shows an almost linear behavior as $h$ tends to zero, see [2].

In order to analyze the memory management of an ACA based solution strategy we will consider the memory demand for the near field matrix $K_h^{near}$ and for the far field matrix $K_h^{far}$, respectively. In Figure 5.15 and Figure 5.16 we can observe the dramatically break down of the memory requirements, therefore we are able to increment the number of unknowns essentially. Once again a proportional connection according to the generated admissible and non-admissible blocks can be noticed.

Let us conclude with another impressive graph given by Figure 5.17, which plots the memory demand of the BE-matrices arising by treating the disk example. A comparison of the required memory shows, that approximation methods for matrices arised from boundary element discretization are a must. Due to the quadratic increase of memory demand, the threshold value (typical

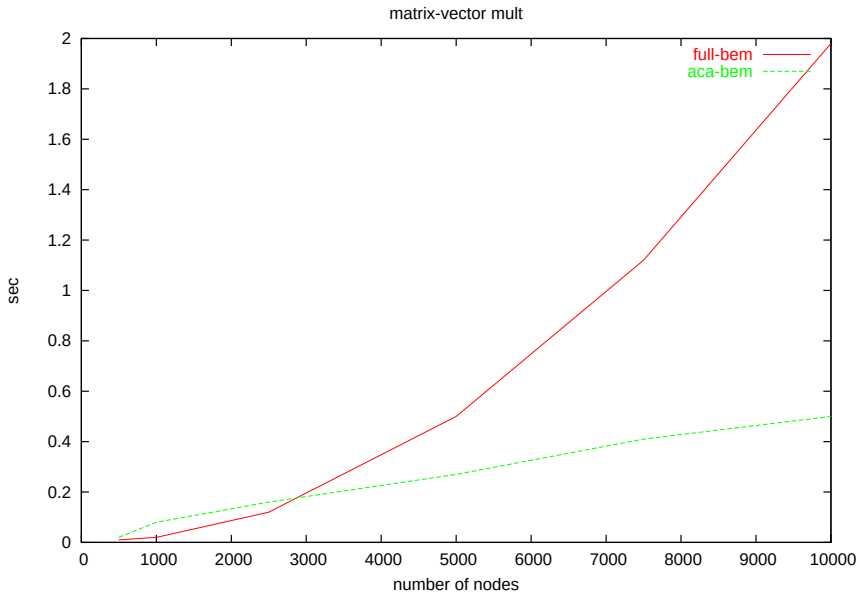



Figure 5.12: Single matrix-by-vector multiplication

2 GByte) of the calculable degrees of freedom is reached very soon on common workstations.

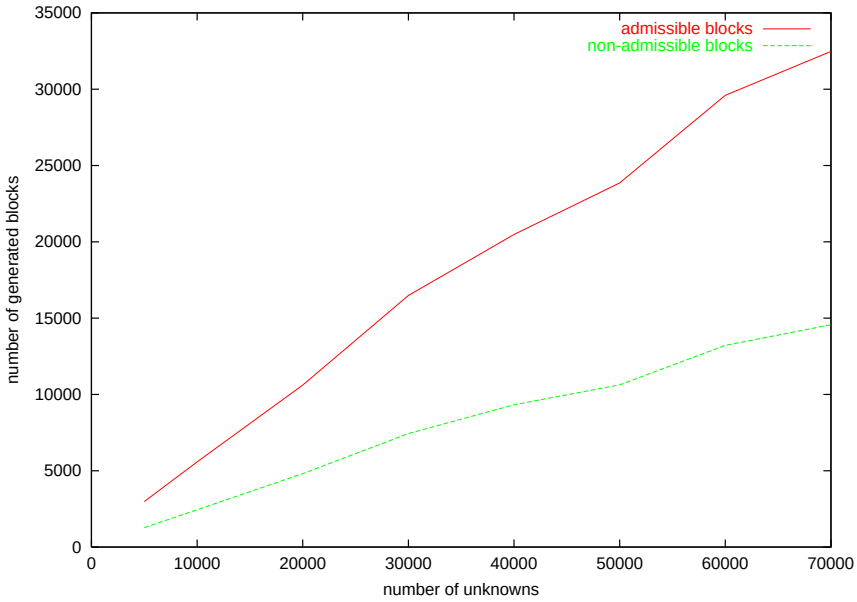



Figure 5.13: Circle: number of generated blocks

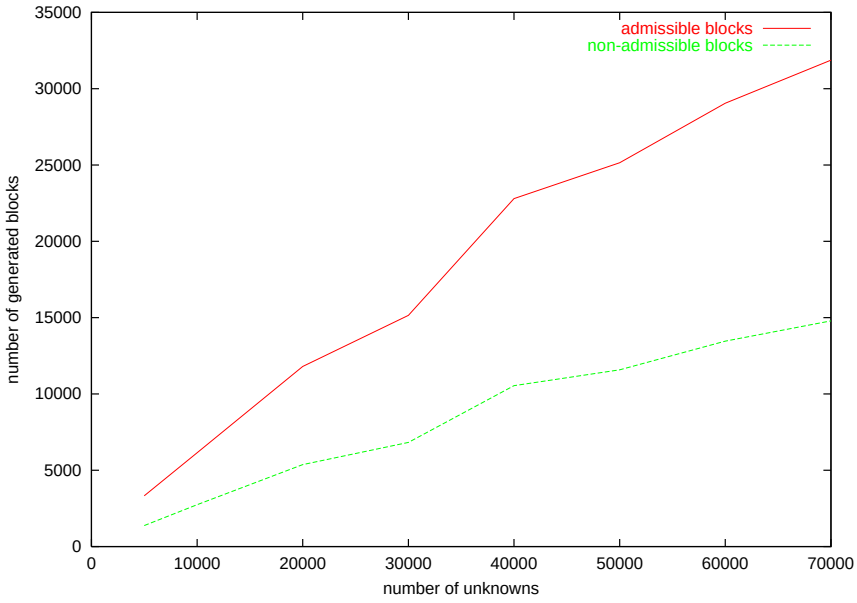



Figure 5.14: L-shape: number of generated blocks

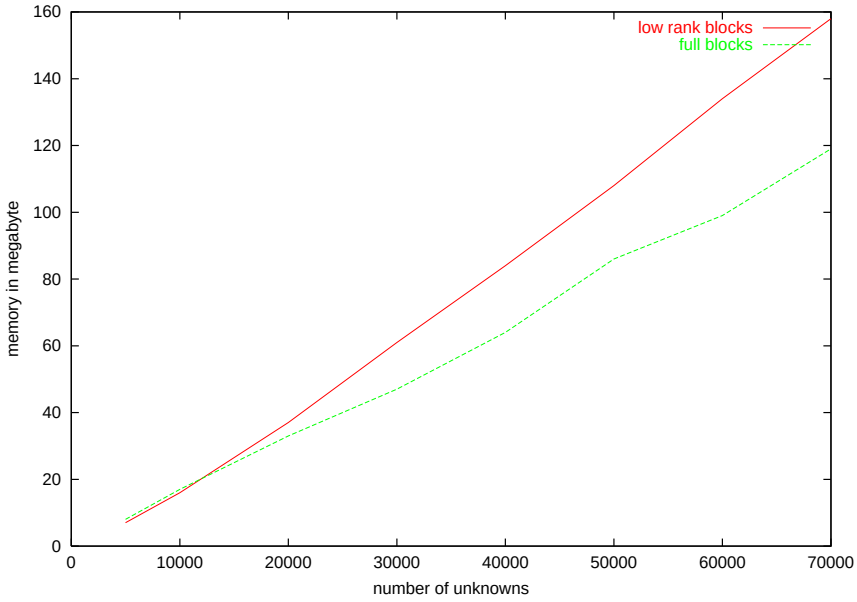



Figure 5.15: Circle: memory demand (ACA)

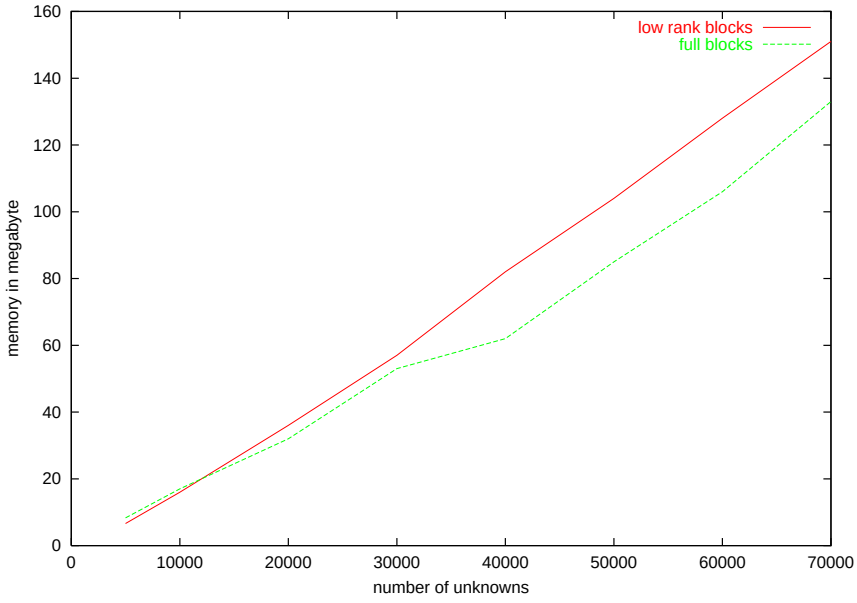



Figure 5.16: L-shape: memory demand (ACA)

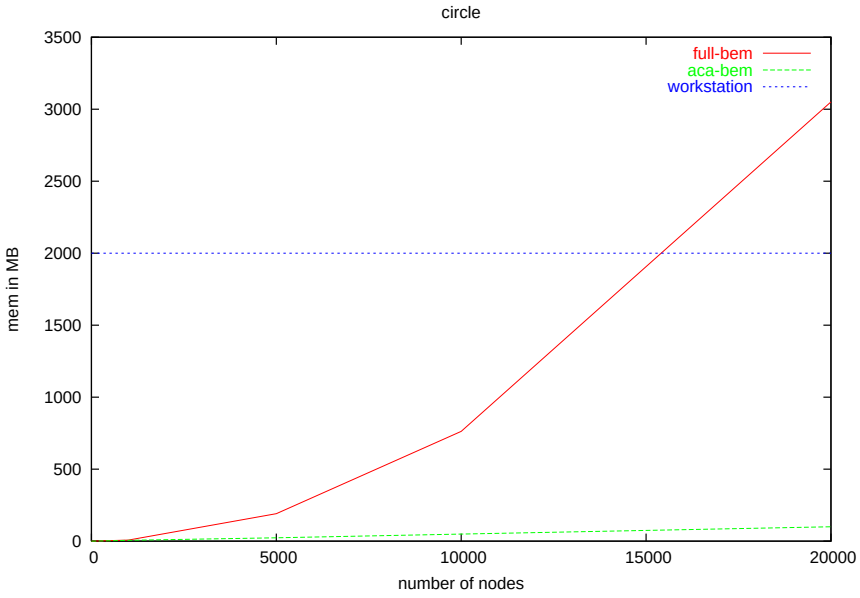



Figure 5.17: Memory Requirements

# Chapter 6

# BEM-FEM Coupling

In this chapter we are concerned with the method of coupling the boundary element and the finite element method. In fact, for the modeling of magnetic field problems this approach has several advantages. If we consider computational domains which include some moving parts it will be the favorable method. Furthermore, problems on unbounded domains are easy to handle within the BE-method. The finite element method is the proper choice to treat non-linearities as well as included densities (e.g. impressed current). A combination of both methods will provide the advantages in each sub-domain.

In the following sections we would like to give an overview over the BEM-FEM coupling approach. In particular we are going to discuss the Maxwell equations and their numerical treatment by the finite element method and boundary element method, respectively. Starting with a partitioning of the computational domain in a FEM-part $\Omega_{FEM}$ and a BEM-part $\Omega_{BEM} = \mathbb{R}^d \backslash \overline{\Omega}_{FEM}$, we will describe the variational formulation of the Maxwell equations in $\Omega_{FEM}$. In addition, we will show the connection to the BEM approach and conclude with the resulting equation system. In order to solve the coupled matrix system we are exploiting the Schur-complement within some kind of domain decomposition according [33]. Finally, some numerical experiments will substantiate the efficiency of multigrid methods, particularly algebraic multigrid in our case.

## 6.1 Problem Formulation

In this section we consider an exterior magnetic field problem and present additionally the variational formulation for the arising coupled system. Starting point of all considerations according electromagnetic field problems are the Maxwell equations. For the sake of simplicity in the rest of the chapter we only suppose the stationary case of Maxwell's equations. An more rigorous analysis of eddy current problems can be found in [33]. Hence, Maxwell's equations in

the stationary case reads as

$$\begin{aligned} \operatorname{curl} H &= J && (6.1)\\ \operatorname{div} B &= 0 && (6.2) \end{aligned}$$

with $H$ denotes the magnetic field strength and $B$ the magnetic induction. Moreover we assume the impressed current density $J = 0$ in $\mathbb{R}^d \backslash \overline{\Omega}_{FEM}$ and divergence-free in $\Omega_{FEM}$

$$\operatorname{div} J = 0 \qquad \text{in } \Omega_{FEM}.$$

For an illustration confer Figure 6.1. The according boundary conditions for Maxwell's equations (6.1) and (6.2) are

$$\begin{aligned} B \cdot n &= 0 && (6.3)\\ H \times n &= 0 && (6.4) \end{aligned}$$

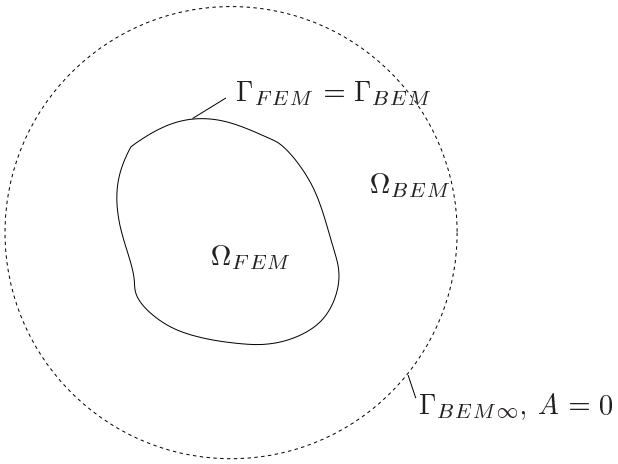



Figure 6.1: BEM-FEM Domains

The properties of the used material are given by the constitutive relationship

$$B = \mu H \tag{6.5}$$

where the non-linear $\mu$ defines the permeability. Since the identity (6.2) holds, we may introduce a vector potential $A$ with

$$B = \operatorname{curl} A.$$

Due to this approach equation (6.1) consequently can be written as

$$\operatorname{curl}(\nu \operatorname{curl} A) = J \tag{6.6}$$

where $\nu$ is defined as the reluctivity $\nu := 1/\mu$. It is worth to mention that $A$ is not unique, for this reason some gauge condition needs to be formulated. In our case it is convenient to choose the gauge condition

$$\operatorname{div} A = 0. \tag{6.7}$$

In order to get a variational formulation that fulfills ellipticity, we are going to add a certain term to the classical formulation. Exploiting the gauge condition (6.7) we obtain

$$\operatorname{curl}(\nu \operatorname{curl} A) - \operatorname{grad}(\nu_0 \operatorname{div} A) = J \tag{6.8}$$

with $\nu_0$ the reluctivity in the free space ($1/\nu_0 = 4\pi 10^{-7}$).

Moreover we are assuming the Sommerfeld's radiation condition $|A| = \mathcal{O}(1/R)$ with $R$ the diameter of a suggested sphere with the computational domain $\Omega_{FEM}$ in the center. As sketched in Figure 6.1 the vector potential $A$ tends to zero at the boundary $\Gamma_{BEM\infty}$.

Now we are able to present the regularized weak formulation of Maxwell's equations (6.1) and (6.2). Take into account the formulation (6.8) and the corresponding boundary conditions (6.3) and (6.4) the weak variational formulation reads as

$$\begin{aligned} \int_\Omega (\nu \operatorname{curl} A \operatorname{curl} v + \nu_0 \operatorname{div} A \operatorname{div} v) dx \quad &+ \\ \int_{\Gamma_{FEM}} (\nu n \times \operatorname{curl} A - \nu_0 n \operatorname{div} A) \cdot v ds_x \quad &= \quad \int_\Omega J \cdot v dx \end{aligned}$$

with $v$ the appropriate test functions. Assuming the interface condition (6.4) on the common boundary $\Gamma_{BEM} = \Gamma_{FEM}$ we have

$$\begin{aligned} H \times n &= 0 \\ \nu \operatorname{curl} A \times n &= 0. \end{aligned}$$

Let us note, that the reluctivity $\nu$ will shift on the boundary interface from $\nu$ in $\Omega_{FEM}$ to $\nu_0$ in $\Omega_{BEM}$. Take into account $n$ denotes the unit outward normal vector of the domain $\Omega_{BEM}$ our weak formulation results in

$$\begin{aligned} \int_\Omega (\nu \operatorname{curl} A \operatorname{curl} v + \nu_0 \operatorname{div} A \operatorname{div} v) dx \quad &- \\ \nu_0 \int_{\Gamma_{BEM}} (n \times \operatorname{curl} A - n \operatorname{div} A) \cdot v ds_x \quad &= \quad \int_\Omega J \cdot v dx \end{aligned}$$

and can be rewritten by using the identity

$$n \times \operatorname{curl} A - n \operatorname{div} A = (n \times \operatorname{grad}) \times A - \frac{\partial A}{\partial n}$$

which provides a splitting of the derivatives into tangential and normal contributions. For the rest of the chapter we are introducing the abbreviation

$$Q = \frac{\partial A}{\partial n}$$

and thus, we are finally obtain (for $J = 0$ in $\Omega_{FEM}$)

$$\begin{aligned} \int_{\Omega_{FEM}} (\nu \operatorname{curl} A \operatorname{curl} v + \nu_0 \operatorname{div} A \operatorname{div} v) dx &- \nu_0 \int_{\Gamma_{BEM}} Q v ds_x - \\ \nu_0 \int_{\Gamma_{BEM}} ((n \times grad) \times A) v ds_x &= 0 \end{aligned}$$

where $Q$ is the corresponding derivative of the normal vector $n$. In order to obtain uniqueness of the solution $A$ we have to ensure a relationship between $A$ and $Q$. By means of such a connection we will be able to solve the equation system properly.

As we will see, a relationship between $A$ and $Q$ can be derived very simple from the boundary element method. Let us considering the classical Maxwell equation formulation including the gauge condition (6.8) with the reluctivity $\nu_0$ in $\Omega_{BEM}$. Thus, the formula can be transformed as follows

$$\operatorname{curl}(\nu_0 \operatorname{curl} A) - \operatorname{grad}(\nu_0 \operatorname{div} A) = -\nu_0 \Delta A = J$$

and leads to the 3D scalar Laplace equation. For this class of problem the fundamental solution is well-known and reads as

$$E(x, y) = \frac{1}{4\pi \|x - y\|}$$

and in addition its normal derivative becomes

$$\frac{\partial E}{\partial n_x}(x, y) = -\frac{(x - y) \cdot n_x}{4\pi \|x - y\|^3}.$$

According to Chapter 2 we can formulate the boundary integral equation

$$\sigma(y) A(y) + \int_{\Gamma} A(x) \frac{\partial E}{\partial n_x}(x, y) ds_x - \int_{\Gamma} Q(x) E(x, y) ds_x = A_J(y)$$

with the right-hand-side

$$A_J(y) = \frac{1}{\nu_0} \int_{\Omega_{BEM}} J(x) E(x, y) dx.$$

Once more $n$ denotes the unit outward vector, $y \in \Gamma_{BEM}$ and $\sigma(y) = 1/2$ for smooth boundary. The vector potential $A_J$ can be calculated by some kind of numerical integration method. With the considered boundary integral equation we are now in a position to formulate the overall matrix equation system.

## 6.2 Discretization

In this section we want to give a brief summary concerning the discretization methods in the case of the FEM method as well as in the case of the BEM method. In order to gain a matrix equation system we discretize the weak Maxwell equations formulation (6.9) by nodal finite elements. The finite element technique is performed either with linear trial functions or, for more accuracy, with quadratic trial functions. In addition we get the corresponding discrete linear equation system. For a closer consideration see [33].
The arising FEM matrices will satisfy the following equations

$$(K_h A_h, v_h) = \int_{\Omega_{FEM}} (\nu \operatorname{curl} A_h \operatorname{curl} v_h + \nu_0 \operatorname{div} A_h \operatorname{div} v_h) dx \tag{6.9}$$

$$(B_h A_h, v_h) = \nu_0 \int_{\Gamma_{BEM}} ((n \times grad) \times A_h) v_h ds_x \tag{6.10}$$

$$(T_h Q_h, v_h) = \nu_0 \int_{\Gamma_{BEM}} Q_h v_h ds_x. \tag{6.11}$$

It is obvious that these three matrices are sparse and symmetric, therefore the handling of them is rather simple.

On the contrary the boundary integral operators leads to fully populated matrices, which additionally become non-symmetric in the case of using the collocation discretization method

$$(G)_{ij} = \int_{\Gamma_{BEM_j}} E(x, y_i) ds_x \tag{6.12}$$

$$(H)_{ij} = \int_{\Gamma_{BEM_j}} \frac{\partial E}{\partial n_x}(x, y_i) ds_x. \tag{6.13}$$

The boundary integrals has to be evaluated at the collocation point $y_i$ using the partitioning of the boundary $\Gamma_{BEM}$

$$\Gamma_{BEM} = \bigcup_j \Gamma_{BEM_j}.$$

If the boundary element matrices will be evaluated by the common Galerkin discretization technique, we would be concerned with symmetric matrices one the one hand, but on the other hand we have to calculate twice as much integrals numerically. Moreover, the matrices are still dense and thus, the treatment of them is expensive as well. Hence, most BEM software packages are still realizing the construction of the BEM-matrices by implementing the collocation method in order to restrict the arithmetical cost of their calculation. Nevertheless, with this considerations we are now able to describe the overall linear

equation system in a compact matrix formulation

$$\begin{pmatrix} K_{\Omega\Omega} & K_{\Omega\Gamma} & \mathbf{0} \\ K_{\Gamma\Omega} & K_{\Gamma\Gamma}+B & -T \\ \mathbf{0} & H & G \end{pmatrix} \begin{pmatrix} A_\Omega \\ A_\Gamma \\ Q_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ A_{J\Gamma} \end{pmatrix} \tag{6.14}$$

where the indices $\Omega$ and $\Gamma$ label the corresponding degree of freedom, that belong to the domain $\Omega_{FEM}$ or to the boundary $\Gamma_{FEM}$. Furthermore, the right-hand-side will be represented by $A_{J\Gamma}$ that is given by the numerical evaluation of the integral

$$A_{J\Gamma}(y_i) = \frac{1}{\nu_0} \int\limits_{\Omega_{BEM}} J(x)E(x,y_i)dx$$

at every discretization point $y_i \in \Gamma_{BEM}$.

In the following we present a brief overview over a solving approach based on an appropriate domain decomposition method considered in [33]. Neglecting the non-linearity due to the permeability $\mu$ we are able to formulate a reduced equation system. Hence, only the unknowns $A_\Gamma$ corresponding to the boundary $\Gamma_{FEM}$ have to be calculated. In addition, the remaining unknowns $A_\Omega$ and $Q_\Gamma$ can be expressed in terms of $A_\Gamma$. The reduced equation system reads in matrix form as follows

$$\left(\mathrm{Schur}K + B + TG^{-1}H\right) A_\Gamma = TG^{-1}A_{J\Gamma} \tag{6.15}$$

with

$$\mathrm{Schur}K = K_{\Gamma\Gamma} - K_{\Gamma\Omega}K_{\Omega\Omega}^{-1}K_{\Omega\Gamma}.$$

In order to get reasonable convergence of the iterative solver it is advisable to use some kind of preconditioning. In our case it is convenient to choose the preconditioning matrix

$$C = TG^{-1}H$$

which provides several advantages. Now, the reduced $C^{-1}$ preconditioned matrix equation reads

$$(H^{-1}GT^{-1}(\mathrm{Schur}K + B) - I_{\Gamma\Gamma})A_\Gamma = H^{-1}A_{j\Gamma}. \tag{6.16}$$

The actual choice of the preconditioning matrix $C$ can be motivated by some geometry and material considerations. Let us assume some magnetic material that provides a high permeability ($\nu \ll \nu_0$), then we are allowed to approximate

$$C^{-1}\mathrm{Schur}K + I_{\Gamma\Gamma} \approx I_{\Gamma\Gamma}.$$

Moreover the matrix $B$ vanishes for 2D problems due to certain symmetry properties. In conclusion the whole system matrix roughly behaves like $I_{\Gamma\Gamma}$ and that will suppose fast convergence.

Now, it is reasonable to solve (6.16) by some iterative methods. Due to the non-symmetric BE-matrices it is not possible to use the CG-algorithm, but there are several other Krylov subspace methods. In the following numerical examples we are using the GMRES-algorithm [38], that turned out as the most efficient one in the present BEM-FEM software package. In order to improve the construction of the Schur-complement we are going to apply the AMG-method. Therefore, the inversion of the interior non-boundary contributions $K_{\Omega\Omega}$ of the FE system matrix $K_h$ is performed by an AMG preconditioned CG-algorithm. The setup of the matrix hierarchy and the transfer operators for the algebraic multigrid algorithm has to be evaluated only once which is advantageously, since the preconditioner can be used in each GMRES-step to perform $K_{\Omega\Omega}^{-1} * \underline{d}_h$ iteratively.

## 6.3 Numerical examples

In this section we show the efficiency of the AMG-preconditioned solving method compared with a direct solver. Moreover, we show the advantage of the auxiliary matrix approach in order to process polynomial trial functions of quadratic order. All BEM-FEM coupled calculations were done by EDYSON (internal software package of the Robert Bosch GesmbH) combined with the AMG program package PEBBLES [30]. Since we are not using any BE-matrix compression method (ACA) in these examples, the boundary element part would need most effort in the overall calculation process with respect to the calculation time. In order to minimize the CPU running time of the solving procedure we are using rather simple geometries. Moreover, the discretization is designed in such way, that the number of boundary elements are limited, but the number of finite elements are even high. Actually, we are treating the unit sphere with rough discretization near the boundary and fine discretization in the center (Figure 6.2).

This example yields the desired properties with the result that the overall computation time is mostly reflected by the numerical treatment of the FEM part. In the rest of this section we consider the case of non-linear magnetostatic problems. The non-linearity is treated by a Newton-Raphson algorithm [41], which is already implemented in the EDYSON software package.

### 6.3.1 Linear trial function

In our first example we consider a quarter sphere (Figure 6.3), where the effects of symmetry will be exploited.

In order to evaluate the Schur-complement $K_{\Gamma\Gamma} - K_{\Gamma\Omega}K_{\Omega\Omega}^{-1}K_{\Omega\Gamma}$ we are using a default direct solver on the one hand and the iterative AMG preconditioned CG-algorithm on the other hand. In particular, we would like to compare the

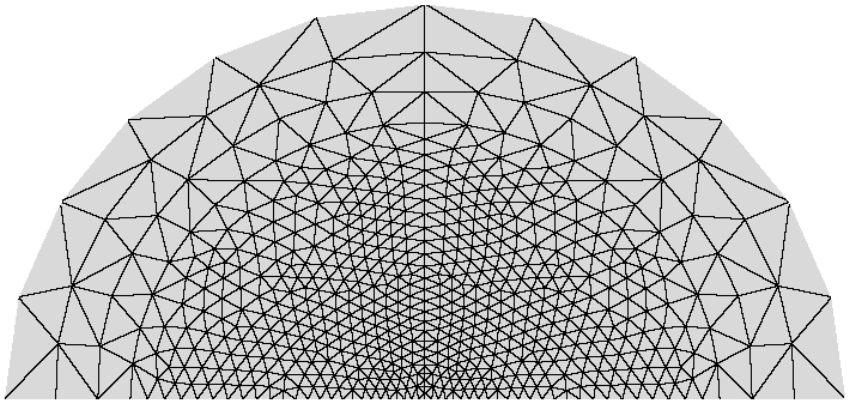


Figure 6.2: Discretization Sphere

efficiency of both solvers to perform $K_{\Omega\Omega}^{-1} * \underline{d}_h$. Due to the internal design of the software packages (EDYSON, PEBBLES) the actual number of unknowns resolved by the direct solver differs from the number of unknowns handled by the AMG method. In the case of utilizing symmetry properties, for each single boundary node several degrees of freedom could be canceled. Thus, for this kind of geometries the internal number of unknowns in EDYSON decreases, on the contrary the AMG method will demand each degree of freedom of a single node.

In Table 6.1 we compare the time to perform an AMG based solver with a direct solver. In the third column the time is split up into the overall setup time and in particular the time of the coarsening procedure. The last column contains the solving time for the whole BEM-FEM coupled system.

| Nodes | $N_h$ | | Setup (Coarsening) (sec) | Inversion (sec) | Time (sec) |
|---|---|---|---|---|---|
| 2562 | 9776 | AMG | 4,5 (3,5) | - | 162 |
| | 6732 | Direct | - | 9 | 98 |
| 26386 | 102800 | AMG | 119 (102) | - | 2810 |
| | 77486 | Direct | - | 1933 | 4914 |

Table 6.1: Calculation time for quarter sphere

As we would expect, the direct solver will not be clearly slower until the number of discretization nodes increases. Moreover, we notice that the major part of the setup time is going to the effort of the coarsening process.

In addition, we consider the full sphere (Figure 6.4), where the corresponding number of unknowns are almost the same in both classes of solver type due

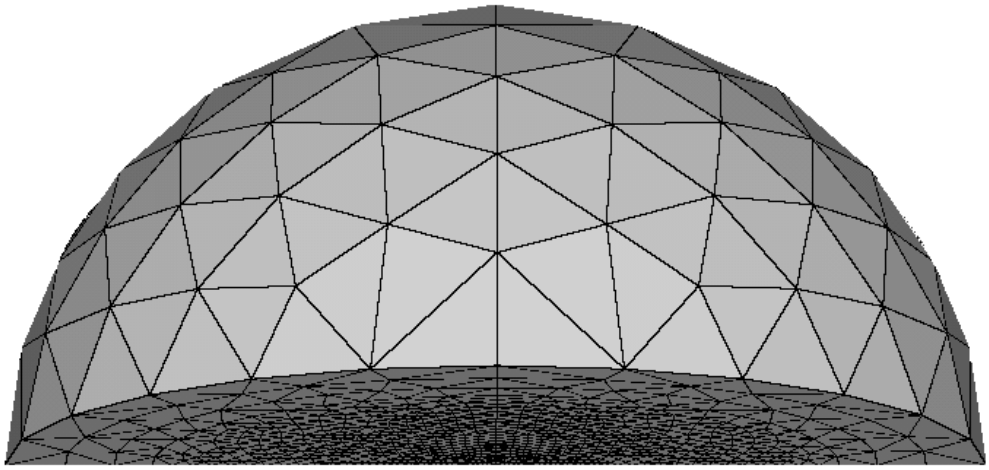


Figure 6.3: Quarter Sphere

to the non-presence of symmetric plains.

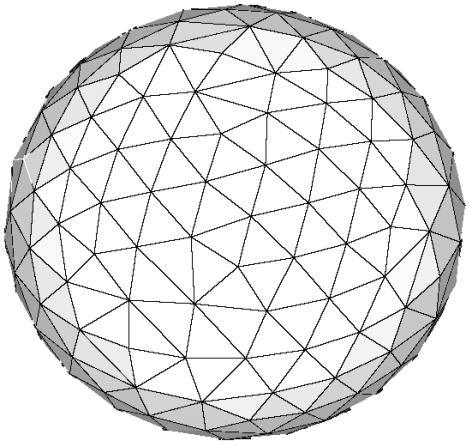


Figure 6.4: Full sphere

In Table 6.2 the same quantities are listed as in the previous Table 6.1. Again, it can be observed the efficiency of the algebraic multigrid solver for this kind of full geometries.

Moreover, one can emphasize that the direct inversion of the FE-matrix $K_{\Omega\Omega}$ takes nearly the same time than the overall calculation process in the case of the AMG application. In fact, the number of iterations for the conjugate gradient algorithm only increases slightly (see Table 6.3) and therefore we might conclude the distinguished quality of the AMG preconditioner.

| Nodes | $N_h$ | | Setup (Coarsening) (sec) | Inversion (sec) | Time (sec) |
|---|---|---|---|---|---|
| 7885 | 30732 | AMG | 40 (34) | - | 808 |
| | 31540 | Direct | - | 805 | 2009 |

Table 6.2: Calculation time for sphere

| $N_h$ | it |
|---|---|
| 9776 | 8 |
| 30732 | 9 |
| 102800 | 11 |

Table 6.3: Number of PCG iterations

In this subsection the calculation of our examples will be performed for matrices which arise from discretizing the computational domain with linear trial functions. If we are using polynomial trial functions of higher order, it will be reasonable to use the introduced auxiliary matrix within the AMG package. The following experiments will point out the great convenience and universality of the auxiliary matrix approach.

### 6.3.2 Quadratic polynomial trial function

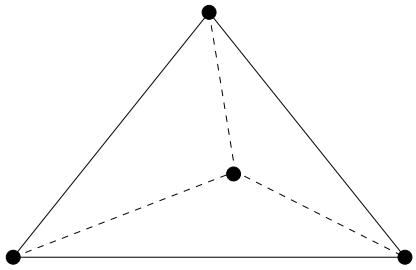
This section figures out the differences for some numerical examples, whether we are using quadratic polynomial trial functions or not. As shown in Figure 6.5 we obtain additional evaluation nodes in the case of the quadratic approach.

Since most coarsening strategies are designed for sparse matrices arising from linear finite element discretization, we may consider the auxiliary matrix approach. Take into account the following possible construction of the auxiliary matrix suggested in Section 4.2

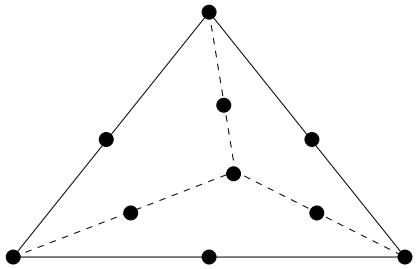
$$(B_h)_{ij} = \begin{cases} -\frac{1}{\|e_{ij}\|_0} & \text{if } j \in N_h^i \\ \sum\limits_{k \neq i} \frac{1}{\|e_{ik}\|_0} & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

with $\|e_{ik}\|_0$ measures the distance between two nodes $i$ and $j$ in the common Euklidian norm. Furthermore, the set $N_h^i$ contains all nodes $j$, that have a direct connection to the node $i$. The complete setup process of the transfer operators is executed only by utilizing the information entered in the auxiliary matrix. Finally, the matrix hierarchy corresponding to the system matrix $K_h$ is constructed once more by Galerkin's method

$$K_l = P^\top K_{l-1} P$$



Linear



Quadratic

Figure 6.5: Tetraeder: linear, quadratic trial functions

with $l$ labels the current multigrid level and $K_0 \equiv K_h$ the given system matrix on the finest grid. The construction of the prolongation operator $P$ has been described in Section 4.2. As considered in the linear trial function examples in the previous section, we again treat a quarter sphere (Figure 6.3). Due to the high number of unknowns the direct solver is the slowest variant in any case. In order to compare the different coarsening strategies, the quantity of interest will be the calculation time, which is necessary to setup the AMG preconditioner. Thus, in Table 6.4 we can notice a significant speed-up in the case of using the auxiliary matrix.

| Nodes | $N_h$ | | Setup (Coarsening) (sec) | Inversion (sec) | Time (sec) |
|---|---|---|---|---|---|
| 17802 | 57340 | Direct | - | 1471 | 1887 |
| | 69864 | AMG | 657 (420) | - | 1570 |
| | 69864 | AMG (AUX) | 49 (24) | - | 910 |

Table 6.4: Calculation time for quarter sphere, quadratic trial functions

Here, we are faced with very simple geometries, but in order to point out the efficiency of algebraic multigrid methods applied on spares finite element matrices these examples were sufficient in this chapter. For numerical examples with a higher amount of boundary elements, the main effort will be invested in the BEM part. In order to get rid of this drawback it is an absolutely must

to perform sparse boundary representations.

Since the overall solving process includes GMRES cycles as well as Newton iterations (and additionally Euler steps in the case of time depended examples) it is not necessary to take a high accuracy for our AMG methods. Therefore the number of iterations for the PCG algorithm could be reduced further.

# Chapter 7

# Conclusions

In these thesises we presented an algebraic multigrid approach that is adapted to matrices arising from the boundary element method. On the one hand, we developed powerful and efficient algebraic multigrid methods. The analysis allows us to construct optimal preconditioners for boundary element matrices resulting in a fast convergence of iterative solvers, such as the CG-algorithm for symmetric, positive definite matrices or similar Krylov-subspace methods for collocation matrices, which are usually non-symmetric. Our numerical experiments confirm this fast convergence in the case of the Galkerin method as well as the collocation method.

One drawback of standard boundary element methods is the fact that the corresponding system matrices are fully populated. Therefore, the complexity for storing the system matrix and for one single matrix-by-vector multiplication is $\mathcal{O}(N_h^2)$, where $N_h$ denotes the number of unknowns. This complexity can be reduced to almost $\mathcal{O}(N_h)$ (up to some polylogarithmic factor) by using sparse approximation techniques such as the ACA technique that was exploited in our thesises.

The construction of the algebraic multigrid method on the bases of the single grid information provided by an ACA matrix requires some important modifications of the AMG for fully populated boundary element matrices. Such AMG algorithms were developed in our theses. As was shown by our numerical experiment the ACA-AMG preconditioned iterative solvers are almost optimal in complexity, i.e. the memory demand and the arithmetical cost are almost proportional to the number of boundary unknowns. That allows us to solve really large-scale problems.
We also used and tested algebraic multigrid methods for finite element equations arising from the discretization of Maxwell equations in connection with 3D industrial application.

As a next step we will upgrade the algebraic multigrid technique in order to

solve boundary element equations in 3D. In this case only an appropriate realization of the Laplace-Beltrami operator on the corresponding surface of a 3D domain will ensure a proper smoothing process and therefore fast convergence. If we consider the case of using geometric multigrid methods, we hopefully will be able to give a rigorous analysis exploiting the present theoretical background.

Once the algebraic multigrid technique works well in the case of boundary element matrices, we can think of further applications in a BEM-FEM coupled system. Instead of building a Schur-complement and splitting up the whole matrix equation system into a finite element part and a boundary element part we can apply the adapted algebraic multigrid design to the overall matrix equation. Take into account an optimistical point of view, we should obtain a faster convergence and an increase of possible number of unknowns. This leads to an iterative solver with almost optimal convergence and memory consumption, which can be applied to large-scale industrial applications.

# Bibliography


[1] R. A. Adams, *Sobolev Spaces*, Pure and Applied Mathematics, Academic Press, New York, 1975.

[2] M. Bebendorf, *Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang-Matrizen*, Ph.D. thesis, Unviversität Saarbrücken, 2000.

[3] M. Bebendorf and S. Rjasanov, *Adaptive Low-Rank Approximation of Collocation Matrices*, Preprint No. 39, Universität des Saarlandes, Fachrichtung 6.1 - Mathematik, 2001.

[4] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing **55** (1995), 379–393.

[5] J. H. Bramble, Z. Leyk, and J. E. Pasciak, *The Analysis of Multigrid Algorithms for Pseudo-Differential Operators of Order Minus One*, Math. Comp. **63** (1994), no. 208, 461 – 478.

[6] J. H. Bramble and J. E. Pasciak, *New Estimates for Mulitlevel Algorithms Including the V-cycle*, Brookhaven Nat. Lab. Rep. (1995), no. # BNL-46730.

[7] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp. (1995), no. 57, 23 – 45.

[8] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput. **19** (1986), 23–56.

[9] ———, *Generally highly accurate algebraic coarsening*, Elec. Trans. Num. Anal. **10** (2000), 1–20.

[10] 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.

[11] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations.*, SIAM J. Sci. Statist. Comput. **9(4)** (1988), 669–686.

[12] G. Chen and J. Zhou, *Boundary element methods*, Computational Mathematics and Applications, Academic Press, 1992.

[13] M. Costabel, *Boundary integral operators on lipschitz domains: Elementary results.*, SIAM J. Math. Anal. **19** (1988), no. 3, 613–626.

[14] J. W. Demmel, J. R. Gilbert, and X. S. Lie, *SuperLU - User's Guide*, 1999, www.nersc.gov/∼xiaoye/SuperLU.

[15] S. A. Goreinov, E. E. Tyrtyshnikov, and A. Y. Yeremin, *Matrix-free iterative solution strategies for large dense linear systems*, Numer. Linear Algebra Appl. **4** (1997), no. 4, 273–294.

[16] 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.

[17] W. Hackbusch, *Multigrid methods and application*, Springer Verlag, Berlin, Heidelberg, New York, 1985.

[18] ———, *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, BG Teubner, Stuttgart, 1991.

[19] ———, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices*, Computing **62** (1999), no. 2, 89–108.

[20] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math. **54** (1989), no. 4, 463–491.

[21] G. C. Hsiao and W. L. Wendland, *A finite element method for some integral equations of the first kind.*, J. Math. Anal. Appl. **58** (1977), 449–481.

[22] M. Jung and U. Langer, *Applications of multilevel methods to practical problems*, Surveys Math. Indust. **1** (1991), 217–257.

[23] 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.

[24] 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.

[25] M. Kuhn and O. Steinbach, *Symmetric coupling of finite and boundary elements for exterior magnetic field problems*, Preprint 5, Univerität Stuttgart, Sonderforschungsbereich 404, 2001.

[26] U. Langer, *Skriptum zur Vorlesung NUMERIK I (Operatorgleichungen)*, Johannes Kepler University Linz, Institut für Analysis und Numerik, 1996, www.numa.uni-linz.ac.at/Teaching/lectures.

[27] _____, *Skriptum zur Vorlesung NUMERIK II (Numerische Verfahren für Randwertaufgaben)*, Johannes Kepler University Linz, Institut für Analysis und Numerik, 1996, www.numa.uni-linz.ac.at/Teaching/lectures.

[28] U. Langer and W. Queck, *Preconditioned Uzawa-type iterative methods for solving mixed finite element equations*, 3/1987, Wissenschaftliche Schriftenreihe der TU Karl-Marx-Stadt, Karl-Marx-Stadt, 1987.

[29] G. Meurant, *Computer solution of large linear systems*, Studies in Mathematics and its Applications, vol. 28, Elsevier, 1999.

[30] S. Reitzinger, *PEBBLES - User's Guide*, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing ", 1999, www.sfb013.uni-linz.ac.at.

[31] _____, *Algebraic Multigrid Methods for Large Scale Finite Element Equations*, Schriften der Johannes-Kepler-Universität Linz, Reihe C - Technik und Naturwissenschaften, no. 36, Universitätsverlag Rudolf Trauner, 2001.

[32] S. Reitzinger and J. Schöberl, *An Algebraic Multigrid Method for Finite Element Discretizations with Edge Elements*, Numer. Linear Algebra Appl. **31** (2002), no. 3, 223 – 238.

[33] V. Rischmüller, J. Fetzer, M. Haas, S. Kurz, and M. Rucker, *Computational Efficient BEM-FEM Coupled Analysis of 3D Nonlinear Eddy Current Problems Using Domain Decompostition.*, 2 3, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 1998.

[34] S. Rjasanow, *Vorkonditionierte iterative Auflösung von Randelementgleichungen für die Dirichlet-Aufgabe*, 7/1990, Wissenschaftliche Schriftenreihe der TU Chemnitz, Chemnitz, 1990.

[35] V. Rokhlin, *Rapid solution of integral equations of classical potential theory.*, J. Comput. Phys. **60(2)** (1985), 187–207.

[36] 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.

[37] ———, *Algebraic multigrid (AMG)*, Multigrid Methods (S. McCormick, ed.), Frontiers in Applied Mathematics, vol. 5, SIAM, Philadelphia, 1986, pp. 73–130.

[38] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.*, SIAM J. Sci. Stat. Comput. **7** (1986), no. 3, 856–869.

[39] A. H. Schatz, V. Thomée, and W. L. Wendland, *Mathematical theorie of finite and boundary element methods*, DMV Seminar, no. 15, Birkhäuser, 1990.

[40] J. Schöberl, *Robust multigrid preconditioning for parameter-dependent problems I: The stokes-type case*, Technical Report 97-2, Johannes Kepler Universität Linz, Institut für Mathematik, 1997, www.nathan.numa.uni-linz.ac.at/pub/technical_reports/.

[41] B. B. Shyamkamar and Z. J. Cendes, *Convergence of iterative methods for nonlinear magnetic field problems.*, IEEE Transactions on Magnetics **24** (1988), no. 6, 2585–2587.

[42] O. Steinbach, *Gebietszerlegungsmethoden mit Randintegralgleichungen und effiziente numerische Lösungsverfahren für gemischte Randwertprobleme*, Ph.D. thesis, University of Stuttgart, 1996.

[43] ———, *Stability Estimates for Hybrid Coupled Domain Decomposition Methods*, no. 1809, Springer Verlag, Berlin Heidelberg, 2003.

[44] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing **56** (1996), 179–196.

[45] T. von Petersdorf and E.P. Stephan, *On the convergence of the multigrid method for a hypersingular integral equation of the first kind*, Numer. Math. **57** (1990), 379–391.

[46] W. L. Wendland ed., *Boundary Element Topics*, Springer Verlag, 1997.

# Eidesstattliche Erklärung

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

# Curriculum Vitae

**Name:** David Pusch

**Date of Birth:** 27. November 1975

**Place of Birth:** Linz, Austria

**Citizenship:** Austria

**Education:**

| | | |
|---|---|---|
| 1982 - 1986 | VS Kirchberg/Donau |
| 1986 - 1990 | HS Neufelden |
| 1990 - 1995 | HTBLA Leonding |
| 1996 - 2003 | Johannes Kepler University Linz |

**Alternative Service:** 1995 - 1996

**External Practical:** Sept. 2001 - Feb. 2002 Robert Bosch GesmbH, Stuttgart, Germany