A Hierarchically Semiseparable Preconditioner for the Numerical Solution of 3D Electromagnetic Scattering Problems

von Ignacio Gutiérrez-Cañas Pazos

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften
- Dr.rer.nat. -

Vorgelegt im Fachbereich 3 (Mathematik & Informatik)
der Universität Bremen
im Dezember 2005

Datum des Promotionskolloquiums: 21. Februar 2006

Gutachter: Prof. Dr. Angelika Bunse-Gerstner (Universität Bremen)
Prof. Dr. Wolfgang Hiller (Alfred-Wegener Institut, Bremerhaven)

Para Edda, Elke, Iasone, Beñat y Jonathan

Acknowledgements

First of all, I would like to thank my supervisor Prof. Angelika Bunse-Gerstner for her guidance and support. Her intuition and her valuable advice has provided me with a deep insight into the exciting field of preconditioning. Also the numerous occasions she gave me to present my work around the world have been very helpful for the completion of this thesis. Especially the Matrix-theory conference in Haifa where the core ideas of this work were presented, providing a powerful impulse that has accompanied me since I returned from Israel in January of this year. Likewise, I thank her for introducing me to the people such as Prof. Gene Golub or Prof. Charles Van Loan, whom I would have otherwise never met.
In addition to this, the working atmosphere was simply the best I have enjoyed so far, not to mention the *non-working* atmosphere of the famous cheese parties. Moreover, the freedom she gave me has made the time spent in her department a pleasure. I will surely miss it.
I thank Dr. Rainer Bunger from EADS Bremen for all the two-minute phone calls that turned into involved, half-hour (or longer) conversations. Also his support and help when it came to finding information in his MLFMM code was really invaluable.
Prof. Stephan Frickenhaus has been for me during the last months an example of unlimited generosity. He has supported me continuously during the implementation of the preconditioner developed in the thesis. During this process he has helped me in situations which I otherwise would have not been able to solve alone.
I thank Prof. Wolfgang Hiller for the access to the supercomputer which I was allowed to use for some of the numerical results in this work and I express my gratitude to Dr. Michael Schröter for his help and support to overcome the difficulties related to the usage of the supercomputer.
Last but not least, I thank Prof. Shiv Chandrasekharan for the time he spent here in Bremen to work with me this year, and for all his support since the first time we met last year in Leuven.

Contents

List of Figures

Chapter 1

Introduction

The numerical solution of three dimensional scattering problems related to the Electric Field Integral Equation (EFIE) is known to be an extremely challenging task, both from the numerical and the computational point of view. In particular, this integral equation leads, after discretization, to a linear system of equations that, in general, is badly conditioned. This fact, coupled with the dimension of the resulting coefficient matrix which for industrial objects can quickly lead to millions of unknowns, contributes to exacerbate the situation. Among the most important industrial applications of the electromagnetic scattering analysis is the computation of the Radar Cross Section (RCS) of arbitrarily shaped 3D scatterers [57] as well as the analysis of electromagnetic compatibility [54]. As a result, the arising linear systems of equations in the electromagnetic scattering context are, in general, not only very large but also difficult to solve.

The prohibitive cost of storing the complete coefficient matrix explicitly makes the choice of an iterative method mandatory. Among them, the most popular and well-known are the Krylov methods. This family of solvers are known to reference the coefficient matrix only via a matrix-vector product. However, the computational cost of the standard matrix-vector product poses a significant restriction on the dimension of the linear system feasible to be solved. As an alternative to the standard matrix-vector product for the kind of matrices arising in electromagnetic applications, V. Rokhlin and L. Greengard proposed the famous Fast Multipole Method (FMM) in the 1980s [73, 44, 43]. Almost twenty years later E. Darve made the computation of large structures feasible at the expense of partial storage of the coefficient matrix. As a result, only the so-called near-field matrix is explicitly available [26].

Unfortunately, the FMM algorithm for the matrix-vector product represents only a partial solution to another obstacle arising in the electromagnetic context: the fact that without a suitable preconditioner standard solvers for the arising linear systems are prone to fail either because they do not converge in a reasonable amount of iterations or because they fail to converge at all.

This widely known situation has led to intense research in this field and consequently to somewhat sophisticated preconditioning techniques. So far, in the preconditioning context related to the previously mentioned problem, an extremely important feature of the previously mentioned coefficient matrices has not been fully exploited,

namely, the special matrix structure of the coefficient matrix. Such a matrix structure has been the object of intense research in recent years and has essentially lead to two main streams of research in mathematics, namely, the $\mathcal{H}$-matrix theory of Hackbusch et al. and the semiseparable theory, in all its denominations, initiated by Gohberg et al.. The matrix structure we will concentrate on in this thesis is denominated hierarchically semiseparable. This matrix structure comprises matrices which show, roughly speaking, a multilevel low-rank structure in the off-diagonal blocks.

The aim of this thesis is to develop a preconditioner that captures this special feature of the discretized scattering operator. In particular, we will present here a preconditioner that shares some features of the ILU preconditioning but overcomes the inherent storage problem associated to the latter in the electromagnetic context. For this purpose we will create a multilevel low-rank approximation to the coefficient matrix from the near-field matrix, the only part of it we can explicitly access.

The structure of the thesis is the following. In the next chapter the physical problem and the integral equation for perfectly conducting targets are reviewed. Furthermore, some properties of the EFIE as integral operator are acknowledged.

The third chapter comprises a brief introduction to the fundamental pillar of the work, namely the fast multipole methods and an attempt to bridge the gap between the resulting coefficient matrix, the impedance matrix, and the matrix structure previously mentioned. This chapter starts with a simple example that introduces the basic features of the FMM algorithm. Likewise, a simple example with a one dimensional integral operator will serve as motivation for the matrix structure pervading the whole work, the semiseparable structure. After outlining the multilevel version of the FMM, we provide a matrix form interpretation for this algorithm. This will allow us to note the fundamental feature that characterizes the impedance matrices arising in the electromagnetic context mentioned above.

In the fourth chapter the Hierarchically SemiSeparable matrix structure (HSS) is introduced with special emphasis on the electromagnetic applications. The fifth chapter reviews briefly the most successful preconditioning techniques we find currently in the literature. In addition, it serves as motivation for a different approach that the these techniques do not exploit, namely the special matrix structure.

The sixth chapter constitutes the core of the work. At the beginning of this chapter the existence of a suitable matrix permutation based on the MLFMM algorithm is presented. This permutation allows us to enforce the HSS structure on a part of the impedance matrix that constitutes the only un-approximated part of it, the previously mentioned near-field matrix. The resulting hierarchically semiseparable matrix leads to a preconditioner which shares the same matrix structure as the impedance matrix. The novelty of the work is thus the introduction of a new type

of structured preconditioning based on a multilevel low-rank approximation to the near-field matrix.

We present initial numerical results with this HSS preconditioner which include only partial information about the rank-structure of the impedance matrix. In a further step the information provided by the MLFMM algorithm inherent to the impedance matrix is exploited as much as possible so as to improve storage requirements and performance of the HSS preconditioner. Moreover, a preconditioned GMRES is proposed which applies the HSS preconditioner together with an estimation of the operational complexity involved in the construction as well as in the application of the aforementioned preconditioner.

Concluding this chapter, some numerical results are presented that support the claims above concerning performance and storage. In particular, these numerical experiments show that the HSS preconditioner to be developed in the thesis is equivalent, in terms of storage, to a matrix with a similar degree of sparsity to that of the near-field matrix itself. Consequently, this new preconditioner seems to overcome the storage problems which beset the popular ILU in this context without forfeiting performance.

The appendix provides a brief insight into the SHSS (Symmetric Hierarchically SemiSeparable Package). This package was written by the author with the purpose of integrating the HSS preconditioner into an existing MLFMM code, courtesy of EADS Bremen. It comprises routines that implement the algorithms in [20] together with the preconditioner object of this thesis.

The thesis concludes with a summary of our contributions and a declaration of intent for further work.

Chapter 2

Physical problem

2.1 Introduction

Electromagnetic scattering is concerned with the effect an obstacle or an inhomogeneous medium has on the propagation of electromagnetic waves. Even though the topic as such has a long history, the theory of electromagnetic scattering and more generally the simulation of electromagnetic wave propagation plays still a fundamental role in science and engineering. Among numerous areas of application are biomedicine, electromagnetic compatibility, radar imaging, antenna design and, last but not least, stealth technology. For a comprehensive introduction to this topic see [70, 67].

2.2 Electromagnetic Waves

Consider electromagnetic wave propagation in free-space. Due to the negligible polarization and magnetization of the air we can reasonably assume that our medium is the vacuum. For the latter we denote the constant electric permittivity by ε_0 and the magnetic permeability by μ_0. Electromagnetic waves are described by the electric field $\mathbf{E}$ and the magnetic field $\mathbf{H}$ in what we know as the **Maxwell's Equations**

$$\begin{aligned} \nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} && (2.1)\\ \nabla \cdot \mathbf{H} &= 0 && (2.2)\\ \mu_0 \frac{\partial \mathbf{H}}{\partial t} + rot\, \mathbf{E} &= 0 && (2.3)\\ -\varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + rot\, \mathbf{H} &= \mathbf{J}. && (2.4) \end{aligned}$$

In the equations above $\mathbf{E}(x,t)$ is the electric field, $\mathbf{H}(x,t)$ represents the magnetic field and $\mathbf{J}(x,t)$ is the electric current density. The charge density is represented by $\rho(x,t)$. The assumption of conservation of charge is expressed by means of the continuity equation and this relates $\mathbf{J}$ and ρ in the following way

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = 0.$$

If there is no current or electric charge present in the vacuum, the Maxwell's equations in free-space reduce to

$$\begin{aligned} \nabla \cdot \mathbf{E} &= 0 && (2.5)\\ \nabla \cdot \mathbf{H} &= 0 && (2.6)\\ \mu_0 \frac{\partial \mathbf{H}}{\partial t} + rot\, \mathbf{E} &= 0 && (2.7)\\ -\varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + rot\, \mathbf{H} &= 0. && (2.8) \end{aligned}$$

Due to the fact that the superposition principle applies, we can further simplify the equations taking Fourier transforms in order to remove the time-dependence. The consequence is that instead of solving one time dependent system we will have to solve several time independent systems for a range of different frequencies.
To this end let us assume that $\mathbf{E}$ and $\mathbf{H}$ satisfy the following relation

$$\begin{aligned} \mathbf{E}(x,t) &= Re\left(\mathbf{E}(x)e^{-i\omega t}\right) && (2.9)\\ \mathbf{H}(x,t) &= Re\left(\mathbf{H}(x)e^{-i\omega t}\right), && (2.10) \end{aligned}$$

where $\mathbf{E}(x)$ and $\mathbf{H}(x)$ represent a space dependent complex vector field and the parameter $\omega > 0$ represents the angular frequency. We will work from here on with the time independent $\mathbf{E}(x)$ and $\mathbf{H}(x)$. Note that under this assumption $\mathbf{E}(x)$ an ω completely determine the time-dependent behavior of $\mathbf{E}(x,t)$. The same applies obviously for $\mathbf{H}(x,t)$.
Under the assumption of harmonic time dependence described above, we obtain a set of equations independent of time

$$\begin{aligned} rot\mathbf{E} - i\omega\mu_0\mathbf{H} &= 0 && (2.11)\\ rot\mathbf{H} + i\omega\varepsilon_0\mathbf{E} &= 0 && (2.12)\\ \nabla \cdot \mathbf{E} &= 0 && (2.13)\\ \nabla \cdot \mathbf{H} &= 0. && (2.14) \end{aligned}$$

It can be easily seen that the time harmonic real fields satisfy the time dependent Maxwell's equations if and only if the equivalent complex space dependent fields satisfy the harmonic equations (2.11) - (2.14). $(\mathbf{E}, \mathbf{H})$ is now a solution to the harmonic Maxwell's equations in free-space with constant electric permittivity ε_0 and magnetic permeability μ_0.
Furthermore, we define the wave number $k > 0$ by $k = \sqrt{\varepsilon_0\mu_0}\omega$. Another useful relation concerning k is the following: $k = \frac{2\pi}{\lambda}$, where λ is the wavelength of the electromagnetic wave. It is also convenient to introduce the free-space impedance

Z_0 defined by $Z_0 = (\frac{\mu_0}{\varepsilon_0})^{1/2}$. It follows that $i\omega\varepsilon_0 = \frac{ik}{Z_0}$ and $i\omega\mu_0 = ikZ_0$. The continuity equation changes accordingly and reads

$$\nabla \cdot J - i\omega\rho = 0.$$

2.2.1 Direct Scattering Problem

The basic boundary-value problem for the Maxwell's equations is the following *scattering problem*: suppose we have an object S (S stands for scatterer) and let $\mathbf{n}$ denote the unit outward normal to S. The classical problem is that of a PEC (Perfect Electric Conductor) object in free-space lit by an incident plane wave. For a perfect conductor the electric field will not penetrate the scatterer and we will concentrate only on its surface Γ.
When an electromagnetic wave propagating in free-space impinges on a PEC, characterized by ε_1 and μ_1 and an *infinite* conductivity, a surface current J is induced on its surface.

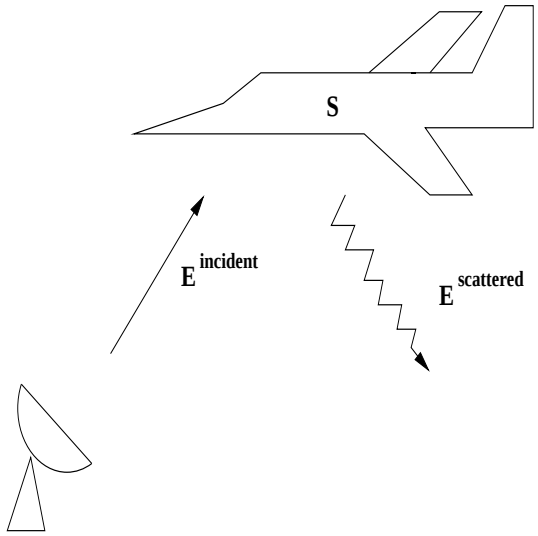


Figure 2.1: *Direct scattering problem.*

Due to the aforementioned infinite conductivity and the skin-effect, the electrical field on its surface must be zero or in other words, we have a boundary condition that imposes a zero tangential electric field on Γ, the surface of S

$$\mathbf{n} \times \mathbf{E} = 0 \ \ on \ \ \Gamma. \tag{2.15}$$

A convenient description for the fields $\mathbf{E}$ and $\mathbf{H}$ is to decompose the total field into an incident part due to sources that are far away and a scattered part due to the charges and currents induced on the scatterer

$$\mathbf{E}^{total} = \mathbf{E}^{incident} + \mathbf{E}^{scattered} \tag{2.16}$$

$$\mathbf{H}^{total} = \mathbf{H}^{incident} + \mathbf{H}^{scattered}. \tag{2.17}$$

To the condition on the conductor surface we have to add the radiation condition [67]. This condition describes the behavior of the scattered fields at infinity, providing a bound for the fields as x tends to infinity.
The formulation of the scattering problem is therefore the following: given $\mathbf{E}^{incident}$, $\mathbf{H}^{incident}$ as the solution to the Maxwell's equations (2.11)-(2.14) representing an incident electromagnetic field, find a solution $\mathbf{E}^{total}, \mathbf{H}^{total}$ satisfying (2.16) and (2.17) such that $\mathbf{E}^{scattered}, \mathbf{H}^{scattered}$ satisfy the **Silver-Müller radiation condition**

$$\lim_{r\to\infty}(\mathbf{H}^{scattered} \times x - r\mathbf{E}^{scattered}) = 0 \tag{2.18}$$

where $r = |x|$ and the limits hold uniformly in all directions $x/|x|$.

2.2.2 Integral Formulation of the Scattering Problem for PEC Objects

There are two main techniques to solve the problem numerically, namely, the differential approach and the integral approach. Whereas the differential methods solve for the fields, the integral methods solve for the induced currents.
In the sequel we limit ourselves to the latter. The integral formulation has several advantages. The primary advantage has been the availability of higher-order methods and a reduction in the number of spatial dimensions. In particular, the integral approach to our direct scattering problem allows us to cast a problem in 3D with a unbounded domain into a bounded one in 2D. Unfortunately, this approach has the disadvantage that the resulting systems of equations are not sparse due to the fact that the interactions are not local. This leads to expensive memory requirements and computation time. The goal of this section is to cast the previously mentioned scattering problem into a boundary integral equation. A thorough analysis of this may be found in [67].
In order to obtain the *Electric Field Integral Equation* (EFIE) the PEC surface is substituted by equivalence surface currents J_Γ that radiate in free-space. For this purpose we use the equivalence theorem (Huygens principle)[70]. The surface current J_Γ represents thus the unknown of the problem. Note also that $J_\Gamma = \mathbf{n} \times \mathbf{H}^{total}$.
For the discussion that follows, we will specialize the scattering formulation to that

from a **perfect conductor**, so that the total tangential electric field is zero at its surface Γ

$$\mathbf{n} \times (\mathbf{E}^{incident} + \mathbf{E}^{scattered}) = 0 \Leftrightarrow -\mathbf{n} \times \mathbf{E}^{incident} = \mathbf{n} \times \mathbf{E}^{scattered}. \tag{2.19}$$

By virtue of a representation theorem for the fields we can compute the scattered fields in every point exterior to the scatterer [67]

$$\mathbf{E}^{scattered}(y) = i\omega\mu_0 \int_\Gamma G_k(x-y) J_\Gamma(x) ds(x) + \frac{i}{\omega\varepsilon_0} \nabla_y \int_\Gamma G(x-y) div_\Gamma J_\Gamma(x) ds(x),$$

where $y \in \mathbb{R}^3 - S$ and

$$\mathbf{H}^{scattered}(y) \;\; = \;\; rot_y \int_\Gamma G_k(x-y) J_\Gamma(x) ds(x), y \in \mathbb{R}^3 - S,$$

where

$$G_k(x-y) = \frac{e^{ik|x-y|}}{4\pi|x-y|}, \tag{2.20}$$

represents the free-space Green's function fundamental solution to the Helmholtz operator $-\Delta - k^2$ satisfying the Sommerfeld radiation condition

$$\partial_r u - iku = o(1/r) \text{ with } r = |x|.$$

Putting both sides of (2.18) together and restricting the observation point to the

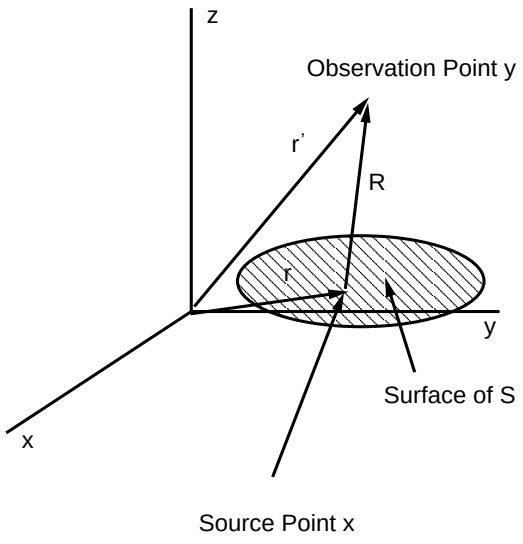


Figure 2.2: *Source and observation points.*

surface of the conducting object Γ yields the EFIE

$$-\mathbf{n}\times\mathbf{E}^{incident} = \\ = \mathbf{n}\times\left[i\omega\mu_0\int_\Gamma G_k(x-y)J_\Gamma(x)ds(x)+\frac{i}{\omega\varepsilon_0}\nabla_y\int_\Gamma G(x-y)div_\Gamma J_\Gamma(x)ds(x)\right],$$

where ∇_y and div_Γ are the gradient with respect to the observation point y and the surface divergence of a vector field tangent to Γ with respect to the source point x, respectively. The EFIE expresses the relation between the incident electric field outside the scatterer and the unknown induced surface current J_Γ.
This form of the EFIE is called the mixed-potential form (MPIE). This equation involves an integral operator with a Cauchy singularity which we will overcome by means of a vector integral identity. As it is well known, the EFIE holds true for both open and closed structures. For open surfaces or when the surface becomes very thin, such as wires or antennas, only the EFIE can be used.
The magnetic counterpart of the EFIE is the *Magnetic Field Integral Equation* (MFIE). Both integral equations suffer from interior resonances, or in terms of their numerical solution, the solutions to the EFIE and MFIE are not unique when analyzing scattering at frequencies that are close to the interior resonances of the body. This phenomena has in general a negative impact on the convergence in the numerical computation to be presented along the sequel. In the case of closed surfaces the *Combined Field Integral Equation* (CFIE) may be used. This integral equation consists of a linear combination of the EFIE and the MFIE and does not suffer from internal resonances [70]. Along the sequel we will concentrate on the EFIE since it allows us to consider structures with antennas and open surfaces.

2.2.3 Numerical Solution of the EFIE

In order to solve the EFIE we introduce a variational formulation where only first derivatives of the unknown J_Γ appear. Multiplying by a test surface current J'_Γ and using the Stokes formula on Γ we obtain

$$\int_\Gamma\int_\Gamma G_k(x-y)\left(J_\Gamma(x)\cdot J'_\Gamma(y)-\frac{1}{k^2}div_\Gamma J_\Gamma(x)div_\Gamma J'_\Gamma(y)\right)ds(x)ds(y) = \\ = \frac{i}{kZ_0}\int_\Gamma \mathbf{E}^{incident}(x)\cdot J'_\Gamma(x)ds(x). \quad (2.21)$$

Note again that in the weak formulation only the Green function itself is involved, therefore we encounter only a weak singular kernel unlike what occurred in the strong formulation presented above.
In this work we will not address the question of solution spaces for this weak formulation nor will we discuss the existence or uniqueness of solutions. For this question

we point to [67].
Following the Galerkin approach (sometimes called method of moments in the engineering community) we discretize the equation by expanding the unknown surface current $J_\Gamma(x)$ by means of the basis functions $\Psi_i(x), i = 1, 2, \ldots N$.

$$J_\Gamma(x) = \sum_{i=1}^{N} x_i \Psi_i(x). \tag{2.22}$$

The expansion coefficients $\{x_i\}_{i=1,2\ldots N}$, referred to as edge currents, are defined for each non-boundary edge of the discretized surface. These edges can be interpreted to be the normal component of the current flowing through an edge. For boundary edges, the edge currents are zero.
If we substitute the above expansion in (2.22) and take the inner product of the resulting expressions with each test function Ψ_i, we obtain a linear system of equations, where the expansion coefficients $\{x_i\}_{i=1,2\ldots N}$ are the unknowns.
The entries in the coefficient matrix are surface integrals and have the form

$$A_{ij} = \int_\Gamma \int_\Gamma G_k(x-y) \left(\Psi_i(x) \cdot \Psi_j(y) - \frac{1}{k^2} div_\Gamma \Psi_i(x) div_\Gamma \Psi_j(y) \right) ds(x) ds(y).$$

The right hand side reads

$$b_j = \frac{i}{kZ_0} \int_\Gamma E^{incident}(x) \cdot \Psi_j(x) ds(x). \tag{2.23}$$

In this fashion we obtain the following linear system of equations

$$Ax = b, \tag{2.24}$$

where A is the **impedance matrix**, b is the **voltage vector** and x represents the unknown **current vector**.
The resulting linear system for the EFIE is complex symmetric (note that we have taken the set of basis functions also as test functions) but non-hermitian. Due to the global coupling of the induced currents in the problem the resulting matrix is dense.
In the MoM (Method of Moments) code, courtesy of EADS Bremen, used for this thesis the basis functions are the Rao-Wilton-Glisson (RWG)[72]. The structure to be analyzed is approximated by planar triangular patches which have the ability to conform to any geometrical surface of boundary.

2.2.4 Some Aspects of the EFIE as Operator

As we are interested in solving the EFIE numerically, it is necessary to point out some initial features of the underlying continuous operator that the discrete counterpart will also reflect.
We can write the EFIE in operator form as

$$\mathcal{L}(J) = -n \times E^{incident},$$

where the right hand side represents the excitation vector (incident electric field). The operator $\mathcal{L}$ is a linear integro-differential operator and in particular a Fredholm operator of the first kind. This kind of operators are related to so-called ill-posed problems. Solving Fredholm integrals of the first kind with smooth kernels is an ill-conditioned problem since the singular values quickly decay to zero [50]. Unlike the usual preconditioning techniques used in the electromagnetic context which try to move the whole spectrum of the discretized operator A, in the context of ill-posed problems one seeks a preconditioner that improves only a part of the singular value spectrum of A and leaves the remaining singular values unchanged [50].
It is also well-known that a non-singular or even weakly-singular integrable kernel corresponds to a compact operator. In this sense the fact that the EFIE involves a singular kernel may be considered as a nice feature of the equation at least as far as its numerical solution is concerned. Unfortunately, this singular kernel accounts also for an increasing condition number as the number of unknowns in the discretized problem grows, even away from resonances.
Nevertheless, the first part of the EFIE represents a compact operator and consequently the eigenvalues always cluster in the vicinity of the origin. As a result, when this part of the operator dominates, we are faced with the clustering of the eigenvalues in an extremely inconvenient region for the convergence of iterative methods.
It is also important to note that within the numerical solution of the EFIE two forms of the equation are considered. For the MoM, the version already presented is used, whereas for the MLFMM (Multilevel Fast Multipole Method), to be presented in the next chapter, an alternative form is used with a strongly singular kernel, namely

$$-\mathbf{n} \times E^{incident} = \mathbf{n} \times \left[i\omega\mu_0 \int_\Gamma \left(I + \frac{\nabla\nabla\cdot}{k^2} \right) G_k(x-y) J_\Gamma(x) ds(x) \right]. \qquad (2.25)$$

The reason for this situation will become clear along the sequel.

2.2.5 The Electrical Size of a Scatterer

In the context of electromagnetic scattering one of the most relevant features of the scatterer is its electrical size. It relates the frequency of the incoming plane wave and its physical size. In particular, the electrical size of the object of interest is expressed in terms of the wavelength of the incoming field λ or alternatively, invoking the relation between the former and the wave number k, as follows: the electrical size ka of a test object is defined by the wave number k and the radius a of the minimum sphere that fully encloses the test object. A scatterer is considered electrically large if $ka > 1$ and electrically small if $ka \leq 1$. In the sequel we will stick to the former version and therefore we will provide, when relevant, the electrical size in wavelengths. Moreover, whenever we refer to the size of a scatterer we will mean the electrical size of it. Taking into account the previous lines, we could make a target larger simply by increasing the frequency (and thus reducing the wavelength) of the illuminating wave.

Chapter 3

Fast Multipole Methods

3.1 Introduction

As we have seen in the last chapter, the method of moments leads to a linear system of equations (2.24) with a complex dense matrix A. It is well known that in the context of standard iterative methods the solution to the linear system of equations obtained in the first chapter scales as $\mathcal{O}(N^2)$. Direct methods like Gaussian elimination are even worse requiring $\mathcal{O}(N^3)$ time, making it absolutely prohibitive for large N. In the case of iterative methods, the most expensive operation in terms of CPU is the matrix-vector product. The method we will present below will considerably reduce both the computational and storage cost of this operation.
The Fast Multipole Method (FMM) was developed by V. Rokhlin [73] in the context of the Poisson equation and gained fame with its application to the gravitational N-body problem, made by L. Greengard [44, 43].
In order to introduce this method, we present a simplified case of application of the FMM in one dimension. This example allows to explain the main features of the FMM in a fashion we will need later on to develop its matrix form.
The FMM algorithm requires for the matrix-vector product $\mathcal{O}(N^{3/2})$ (both memory and CPU complexity), whereas the classical algorithm demands $\mathcal{O}(N^2)$. E. Darve presented the MultiLevel Fast Multipole Method (MLFMM) in the context of the Maxwell's equations [26], further reducing the complexity to $\mathcal{O}(N\log(N))$. The basic idea of this method is to reduce the interactions due to the Green's function in the integral equations of the last chapter by developing the aforementioned function and considering interaction between groups of basis functions rather than between individual basis functions. The physical basis for the FMM is the fact that although short-range interactions can be arbitrarily complex, long-range interactions are relatively smooth, leading, in terms of matrices, to off-diagonal blocks which can be well approximated by low-rank matrices. This is the point where the underlying physics and mathematics meet each other. The first example below provides a good explanation of the ideas underlying the FMM [26, 25].

3.2 Simple Examples in 1-D

3.2.1 Introductory Example

Let us consider N points $x_i \in \mathbb{R}$ distributed quasi-uniformly on $[0, 1]$.
Suppose that the function $f(x_i, x_j)$ ($f : \mathbb{R}^2 \to \mathbb{R}$) can be approximated by means of a product, i.e., $f(x_i, x_j) \approx g(x_i)h(x_j)$. Let us analyze the cost of the following products:

$$\sum_{j=1}^{N} f(x_i, x_j)v_j, i = 1, \dots N,\ v \in \mathbb{R}^N. \tag{3.1}$$

For each of the N points x_i, we have to sum over j, that is, we have an $\mathcal{O}(N)$ operation. The complexity of the product is thus $\mathcal{O}(N^2)$. In the second case, namely for the *decoupled* product, we can write

$$\sum_{j=1}^{N} f(x_i, x_j)v_j = \sum_{j=1}^{N} g(x_i)h(x_j)v_j = g(x_i)\sum_{j=1}^{N} h(x_j)v_j, i = 1, \dots N,\ v_j \in \mathbb{R}. \tag{3.2}$$

According to the last expression, we can compute the product in two steps. First we evaluate

$$S = \sum_{j=1}^{N} h(x_j)v_j \tag{3.3}$$

in $\mathcal{O}(N)$ operations and then, for each i, we calculate the product $g(x_i)S$ in $O(N)$ operations. Thus, we end up with an $\mathcal{O}(N)$ algorithm.
Let us consider the following matrix:

$$\mathcal{M}_{ij} = \begin{cases} \frac{1}{x_i - x_j} & i \neq j \\ 1 & i = j. \end{cases} \tag{3.4}$$

We intend to accelerate the matrix-vector product $\mathcal{M}v$, where v is a given vector. Consider a point x_i and the following expression

$$\sum_{j=1, j\neq i}^{N} \frac{1}{x_i - x_j} v_j. \tag{3.5}$$

The point x_i is called the **observation point** and the x_j are the **source points**. The principal idea of the method is the separation of variables i and j, that is, the

decoupling of the points x_i and x_j.
The decoupling follows from a grouping of the points and the following formula

$$\sum_{n=0}^{\infty} \rho^n = \frac{1}{1-\rho},\ \rho \in (-1,1). \tag{3.6}$$

In order to regroup the points, we consider the following partition of the interval $[0,1]$: $G_1 = \left[0, \frac{1}{4}\right), G_2 = \left[\frac{1}{4}, \frac{1}{2}\right), G_3 = \left[\frac{1}{2}, \frac{3}{4}\right), G_4 = \left[\frac{3}{4}, 1\right]$. Let C_k be the center of the group G_k, with $k \in \{1, \ldots, 4\}$.
If we take, for example, $x_i \in G_1$ and $x_j \in G_3 \cup G_4$, the formula (3.6) allows us to rewrite the entries of the matrix $\mathcal{M}$ as

$$\frac{1}{x_i - x_j} = \frac{1}{C_1 - x_j - (C_1 - x_i)} = \frac{1}{C_1 - x_j} \sum_{n=0}^{\infty} \left(\frac{C_1 - x_i}{C_1 - x_j} \right)^n. \tag{3.7}$$

Assuming that $|C_1 - x_i| \leq \frac{|C_1 - x_j|}{2}$, it follows that for a tolerance $\varepsilon > 0$ there exists a number L_ε, of order $\frac{ln(\varepsilon^{-1})}{ln(2)}$, so that

$$\left| \frac{1}{x_i - x_j} - \sum_{n=0}^{L_\varepsilon} \frac{(C_1 - x_i)^n}{(C_1 - x_j)^{n+1}} \right| \leq \varepsilon \frac{1}{|x_i - x_j|}. \tag{3.8}$$

The matrix-vector product reads now

$$\begin{aligned} \sum_{j/x_j \in G_3 \cup G_4} \mathcal{M}_{ij} v_j &= \sum_{j/x_j \in G_3 \cup G_4} \frac{1}{x_i - x_j} v_j \\ &\approx \sum_{j/x_j \in G_3 \cup G_4} \sum_{n=0}^{L_\varepsilon} \frac{(C_1 - x_i)^n}{(C_1 - x_j)^{n+1}} v_j, \\ &= \sum_{n=0}^{L_\varepsilon} (C_1 - x_i)^n \sum_{j/x_j \in G_3 \cup G_4} \frac{v_j}{(C_1 - x_j)^{n+1}}. \end{aligned}$$

Remark 1

Observe that the relation (3.8) does not hold for $x_j \in G_1 \cup G_2$ since the condition for convergence, $|C_1 - x_i| < |C_1 - x_j|$ is not guaranteed. We need thus to distinguish between **near interactions** (x_i and x_j belong to neighbor groups) and **far interactions** (x_i and x_j belong to groups which are not neighbors).

So far the FMM consists in accelerating the computations of far interactions, whereas the near interactions remain unchanged. Summing up, we approximate the expression $\sum_{j=1, j\neq i}^{N} \frac{1}{x_i - x_j} v_j$ by means of the following terms

- Near interactions

$$\sum_{j/x_j \in G_1 \cup G_2 \, j \neq i}^{N} \frac{1}{x_i - x_j} v_j.$$

- Far interactions

$$\sum_{n=0}^{L_\varepsilon} (C_1 - x_i)^n \sum_{j/x_j \in G_3 \cup G_4} \frac{v_j}{(C_1 - x_j)^{n+1}}.$$

Consider now the gain in applying the FMM for the calculation of the far interactions. We do it in two steps. For each group G_k, for each $n \in \{1, \ldots, L_\varepsilon\}$ we compute the expression

$$S_k^n = \sum_{j/x_j \in G_{far}(k)} \frac{v_j}{(C_k - x_j)^{n+1}}.$$

where $G_{far}(k)$ is the union of the groups which are not neighbors of G_k. Observe that the expression above does not depend on i and that it represents the far contribution common to all observation points x_i in G_k. Further, it is enough to compute for each group G_k and for each point x_i of G_k the following expression

$$\sum_{n=0}^{L_\varepsilon} (C_k - x_i)^n S_k^n. \tag{3.9}$$

The order of complexity of this method is $\mathcal{O}(N^{3/2}\log(N))$ [26, 25].
When the size of the groups is uniform, we call the method a one-level FMM. The same procedure may be applied in each of the groups, leading to a multilevel method. In order to illustrate the algorithm, we can interpret the first step as a transfer of information from the source points x_j to the center C_k of the observation group considered. The second step would be a transfer from the center C_k to the observation point x_k considered. See Figure 3.1.

For the decoupling we have considered above the center of the observation group C_k. In other cases we may accomplish the decoupling between C_k and x_j by introducing the center of the source groups. This approach allows further improvement in the complexity. See Figure 3.2.

A different point of view consists in regarding the method as a compression algorithm for the matrix $\mathcal{M}$. The basic idea in this approach is to understand the algorithm as a low-rank approximation of the far interactions. Indeed, if we consider

$$\sum_{n=0}^{L_\varepsilon} \frac{(C - x_i)^n}{(C - x_j)^{n+1}}$$

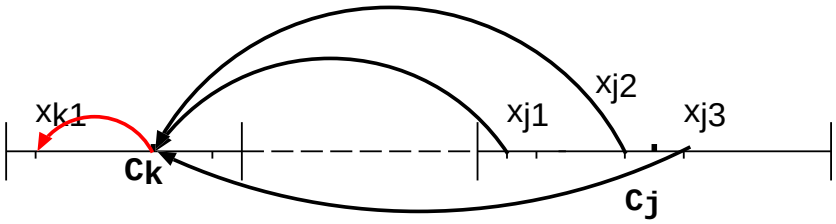


Figure 3.1: *Interpretation of the FMM.*

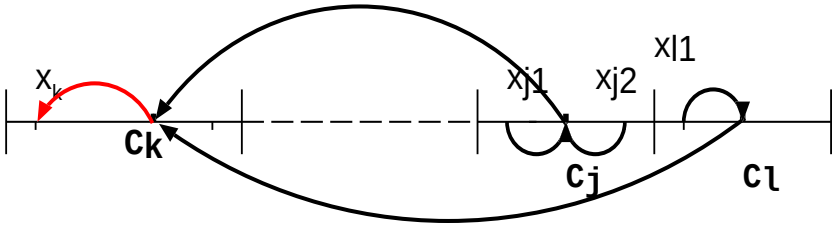


Figure 3.2: *Interpretation of the FMM considering the centers of source groups.*

for $x_i \in C_1$ and $x_j \in C_4$, the rank of the block (C_1, C_4) has the order of $1 + L_\varepsilon$ and not of N (we supposed the x_i to be uniformly distributed on $[0, 1]$, namely in each group the number of elements has the order $\mathcal{O}(N)$). This is the point of view we will pursue in this work. In particular, we will try to compress the far-field of $\mathcal{M}$ so as to obtain an approximation to this matrix which can be efficiently stored.
Unfortunately, the possibility of approximating the matrix by means of low-rank matrices has been possible due to properties of the function $1/x$ for $x \gg 1$. If we consider the Taylor series more in detail, we arrive at the conclusion that the performance of this algorithm depends on the following property: the derivatives of the kernel function decrease more and more rapidly as the degree of the derivative grows.
This kind of approximation is not applicable in the context of oscillating kernels as is the case of the Maxwell's Equations. For these equations we will use an approximation in terms of plane waves as we will see in Section 3.3.
The next section explains how the low-rank approximation to the matrix resulting from the Galerkin process is constructed. There we will consider again a simple 1D case but this time getting nearer to the problem we are interested in, namely the numerical solution of the EFIE. It is a representative example for the techniques we want to present in this thesis in that it justifies the concept of approximation mentioned above. The next section follows [7].

3.2.2 Model Integral Equation in 1D

We consider the following one-dimensional Fredholm equation of the first kind

$$f(x) = \int_0^1 log|x-y|u(y)dy,\ x \in [0,1], \tag{3.10}$$

for a suitable $f : [0,1] \to \mathbb{R}$. We look for a function $u : [0,1] \to \mathbb{R}$ that satisfies the equation on $[0,1]$.
The kernel $K(x,y) = log|x-y|$ in the domain $[0,1]\times[0,1]$ is singular on the diagonal and following the ideas of the introductory example to the fast multipole method, we try a Taylor expansion of the kernel.

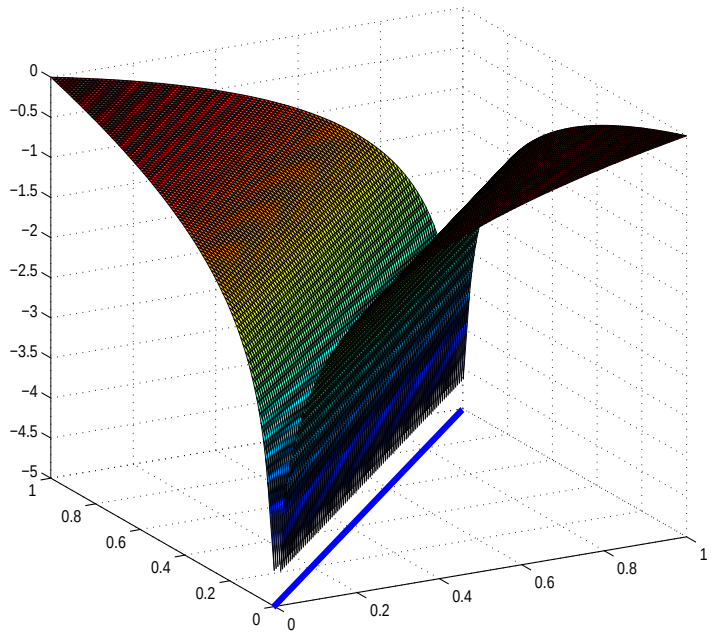


Figure 3.3: *The function $K(\cdot,\cdot)$ on $[0,1]\times[0,1]$.*

Let $I = [a,b]$ and $J = [c,d]$ be two disjoint intervals on $[0,1]$ with $c > b$ and consider $I \times J$ on $[0,1]\times[0,1]$. On this subdomain $K(\cdot,\cdot)$ is not singular. That is the very same situation as in the second form of the EFIE presented in the first chapter. As soon as we leave the diagonal the difficulties arising from the singularity in the kernel disappear. The following lemma provides us with the derivatives of $K(\cdot,\cdot)$. As in the previous example, we have again a very strong decay in the derivatives.

Lemma 2
The derivatives of $K(\cdot,\cdot)$ for $x \neq y$ and $p \in \mathbb{N}$ are

$$\begin{aligned}\frac{\partial^p K(x,y)}{\partial x^p} &= (-1)^{p-1}(p-1)!(x-y)^{-p}\\ \frac{\partial^p K(x,y)}{\partial y^p} &= (p-1)!(x-y)^{-p}\end{aligned}$$

We consider the Taylor series of $x \mapsto K(x,y)$ in $x_0 = \frac{a+b}{2}$. The next lemma establishes the convergence of the Taylor series.

Lemma 3
For each $k \in \mathbb{N}$ the truncated Taylor series

$$\tilde{K}(x,y) = \sum_{j=0}^{k-1} \frac{1}{j!} \frac{\partial^p K(x_0,y)}{\partial x^p}(x-x_0)^j \tag{3.11}$$

approximates the kernel $K(\cdot,\cdot)$ on the interval $[a,b]$ with the following error

$$|K(x,y) - \tilde{K}(x,y)| \leq \frac{|x_0-a|}{|c-b|}\left(1+\frac{|c-b|}{|x_0-a|}\right)^{-k}. \tag{3.12}$$

Proof
For $x \in [a,b]$ and $y \in [c,d]$ the remainder of the Taylor series can be estimated by

$$\begin{aligned}|K(x,y) - \tilde{K}(x,y)| &= \left|\sum_{j=k}^{\infty}(-1)^{j-1}\frac{(j-1)!}{j!}\left(\frac{x-x_0}{x_0-y}\right)^j\right|\\ &\leq \sum_{j=k}^{\infty}\left|\frac{x-x_0}{x_0-y}\right|^j\\ &\leq \sum_{j=k}^{\infty}\left(\frac{|x_0-a|}{|x_0-a|+|c-b|}\right)^j\\ &= \frac{|x_0-a|}{|c-b|}\left(1+\frac{|c-b|}{|x_0-a|}\right)^{-k}\end{aligned}$$

□

We point out that the Taylor expansion has in fact decoupled our integral kernel, decoupling the influence of x on y. Further note the inherent rank-deficiency indicated by (3.11), that will be also present in the resulting matrix.
From the error estimation we see that the radius of convergence covers the interval $[a,b]$ and that for $c \to b$ the estimate for the remainder worsens arbitrarily. Recall

the situation addressed by Remark 1. Even though this fact is what we naturally would expect, it admits another reading, namely we should get nearer to the singularity with decreasing steps if we want to control our estimation. This is the idea of the multilevel approach.
Following [41] we introduce a stronger condition for the disjointness of the intervals, namely the **admissibility condition**

$$diam(I) \leq dist(I, J). \tag{3.13}$$

It is easy to see that if the admissibility condition (3.13) is fulfilled, the approximation error (3.12) can be bounded uniformly.

Low-Rank Approximation of Matrix Blocks

We consider a Galerkin discretization scheme onto a n-dimensional space $V_n = span\{\phi_0, \ldots, \phi_{n-1}\}$. Let the set $I = \{0, \ldots, n-1\}$ contains the indices of the basis functions ϕ_i used above. As basis functions we consider piecewise constant functions uniformly distributed on $[0, 1]$. Dividing I in two sets t, s and defining

$$\tau := \bigcup_{i \in t} supp\, \phi_i, \sigma := \bigcup_{i \in s} supp\, \phi_i,$$

we can obtain a low-rank approximation of the matrix blocks by using the degenerate kernel $\tilde{K} = \sum_{\nu=0}^{k-1} g_\nu(x) h_\nu(y)$ for the indices $(i, j) \in t \times s$ provided $\tau \times \sigma$ satisfies the admissibility condition (3.13), i.e., $diam(\tau) \leq dist(\tau, \sigma)$.
In this case the kernel in

$$G_{ij} = \int_0^1 \int_0^1 \phi_i(x) K(x, y) \phi_j(y) dy dx \tag{3.14}$$

can be replaced by $\tilde{K}$. This yields the following expression for the matrix entries in $\tilde{G}$

$$\begin{aligned} \tilde{G}_{ij} &= \int_0^1 \int_0^1 \phi_i(x) \sum_{\nu=0}^{k-1} g_\nu(x) h_\nu(y) \phi_j(y) dy dx \\ &= \sum_{\nu=0}^{k-1} \left(\int_0^1 \phi_i(x) g_\nu(x) dx \right) \left(\int_0^1 \phi_j(y) h_\nu(y) dy \right). \end{aligned}$$

The submatrix $\tilde{G}|_{t \times s}$ can be interpreted in this manner as $\tilde{G}|_{t \times s} = AB^T, A \in \mathbb{R}^{t \times \{0, \ldots, k-1\}}, B \in \mathbb{R}^{s \times \{0, \ldots, k-1\}}$ with

$$A_{i\nu} = \int_0^1 \phi_i(x) g_\nu(x) dx, \; B_{j\nu} = \int_0^1 \phi_j(y) h_\nu(y) dy.$$

Note that the rank of the matrix AB^T is k independently of the cardinality of t and s. It is not difficult to see that the elementwise error is uniformly bounded, namely $|G_{ij} - \tilde{G}_{ij}| \leq n^{-2}3^{-k}$. We can also obtain an uniform bound for the error

$$||G - \tilde{G}||_F \leq n^{-1}3^{-k}.$$

We consider this particular example with $n = 800$ (basis functions), Taylor expansion of order 4 and a minimal block size of 5. Figures 3.4 and 3.5 show the matrix divided in near and far-field, respectively together with its rank-map, that is the rank distribution on a recursive partition of the basis functions. For the computation of the matrix we have used the library developed by the first two authors in [7], named HLib. Note the extremely strong decay in the rank of the off-diagonal

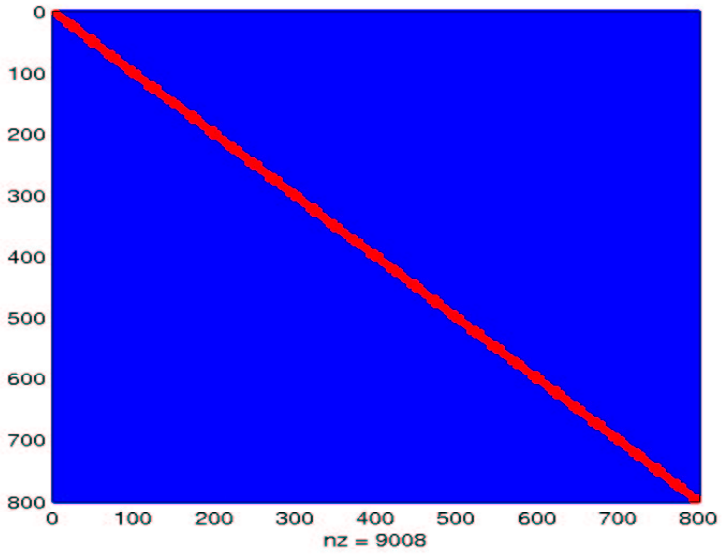


Figure 3.4: *Near- and far-field in the 1D integral equation.*

blocks that allows us to compress very efficiently the far-field part of the matrix. Also it deserves to be pointed out how an increasing number of levels in the recursion reduces the amount of near-field. As we implicitly mentioned above, the blocks in the near-field part of the matrix have full rank, making impossible any kind of compression without forfeiting the accuracy.

The difference in the 2-norm between the matrix A arising from the discretization of (3.10) and the corresponding HSS representation is 4.1484e-08.

Table 3.1 shows the ranks of the off-diagonal blocks in a 6 level bisection of the matrix resulting from the discretization of (3.10). Note the slow growth of the ranks with respect to the size of the block.

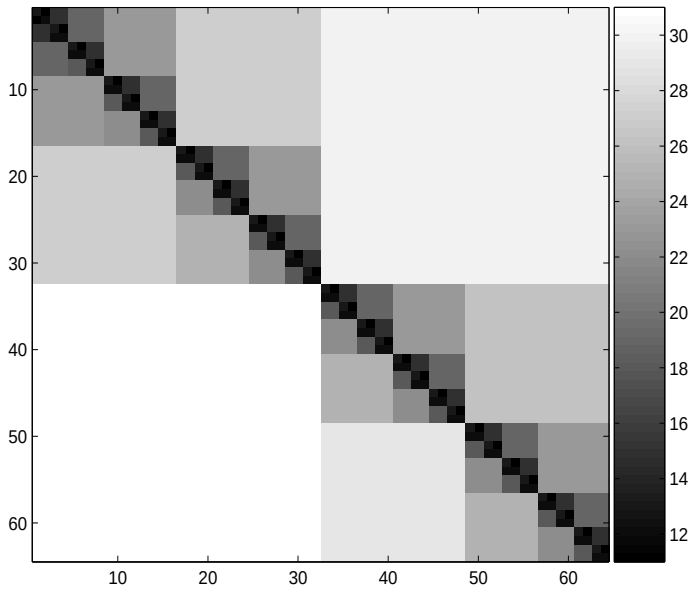


Figure 3.5: *Rank map for matrix arising from the discretization of (3.10).*

Block-size N	Rank of the $N \times N$ block	% of full rank
400	30	7.5
200	27	13.5
100	23	23
50	19	38
25	15	60
13	11	84.7

Table 3.1: *Rank of the blocks in a 800 × 800 matrix for 3.10.*

The information Table 3.1 conveys is the following. Let us consider in Figure 3.5 the submatrix on the right corner depicted in light gray. This submatrix has the size 400. If it were full rank, it would have rank equal to 400 but as the rank is only 30 we have thus 7.5 percent of the full rank. Figure 3.5 allows us to illustrate the previously mentioned strong decay of the rank in the off-diagonal blocks of the matrix.

The next section presents briefly the most important features of the FMM with especial emphasis on what we will need to express the algorithm in terms of matrices.

3.3 Approximation of the Integral Kernel in the FMM

In the previous example we carried out a Taylor expansion of the integral kernel to construct the low-rank approximation of the far field. In this section we will try again to decouple our kernel but this time by means of an expansion in plane waves. In particular, the next theorem is the fundamental pillar for the construction of the FMM. For the results below we note that

$$cos(u, v) = \langle \frac{u}{|u|}, \frac{v}{|v|} \rangle \tag{3.15}$$

where $< \cdot, \cdot >$ denotes the scalar product.

Theorem 4
The Green kernel $\frac{e^{ik|r_0|}}{|r_0|}$, where $r_0 \in \mathbb{R}^3$, admits an " expansion" in terms of plane waves

$$\frac{e^{ik|r_0+r|}}{|r_0+r|} = \lim_{l\to\infty} \int_{S^2} e^{ik<s,r>} T_{l,r_0}(s) ds, \tag{3.16}$$

where

$$T_{l,r_0}(s) = ik \sum_{m=0}^{l} \frac{(2m+1)i^m}{4\pi} h_m^{(1)}(k|r_0|) P_m(cos(s, r_0)) \tag{3.17}$$

for every $r \in \mathbb{R}^3$ such that $|r| < |r_0|$. P_m stands for a Legendre polynomial and $h_m^{(1)}$ is a spherical Hankel function of the first kind. S^2 is the unity sphere and the function $T_{l,r_0}(s)$ is called transfer function.

For an interpretation of the integration against T_{l,r_0} as a convolution with a function of type sinus cardinal and a proof of this theorem see [26].
With the expansion of the integral kernel and the introductory example in 1D in mind, we can now write the basic expression for the FMM algorithm.
We consider two circles S^1 and S^2 with void intersection. We will demote their centers by C_1 and C_2 and their radii by R_1, R_2, respectively. Further let x_1, x_2 be two arbitrary points of S^1 and S^2, respectively.
Let us define $r_0 = C_1 - C_2$ and $r = x_1 - C_1 + C_2 - x_2$. This yields for (3.16)

$$\frac{e^{ik|x_1-x_2|}}{|x_1-x_2|} = \lim_{l\to\infty} \int_{S^2} e^{ik<s,x_1-C_1>} T_{l,r_0}(s) e^{-ik<s,x_2-C_2>} ds. \tag{3.18}$$

Let us discretize the integral over S^2 for a finite l. To this end denote the directions of discretization by s_p and the weights for the quadrature by ω_p. This yields

$$\frac{e^{ik|x_1-x_2|}}{|x_1-x_2|} \sim \sum_p \omega_p e^{ik<s_p,x_1-C_1>} T_{l,r_0}(s) e^{-ik<s_p,x_2-C_2>} ds. \tag{3.19}$$

Observe that, as in the introductory example, we have achieved a separation of variables, thus we may develop a fast algorithm out of this formula. We omit the details concerning how to choose a reasonable l and how to integrate numerically (3.18). For the details we point again to [26].
The results above allows us to formulate a one level FMM.

3.4 One Level FMM

We consider a surface Γ of a PEC (Perfectly Electric Conductor) scatterer S. Let us assume further that on Γ a surface current has been induced by an incident electromagnetic field. Suppose we have N points x_i uniformly distributed on Γ and that we subdivide the surface in $\sqrt{N}$ boxes (sometimes in the sequel we will call them also groups). We denote these boxes by P_r. Consequently, each box P_r will contain $\sqrt{N}$ points.
Let us further denote by C_r the center and by R_r the radius of the box P_r, respectively. Moreover, we define the radius of P_r as follows

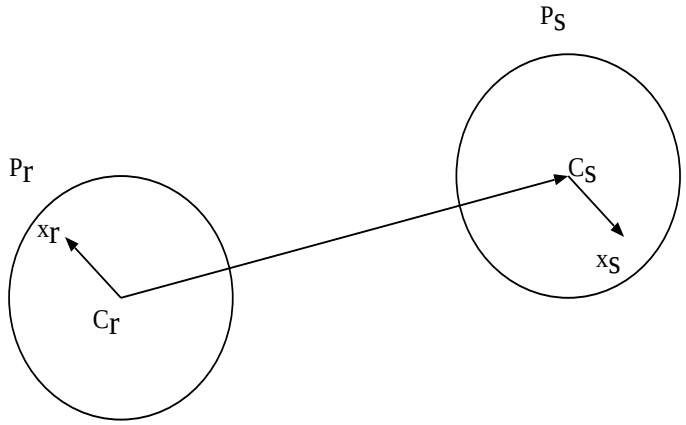


Figure 3.6: *Boxes P_r and P_s.*

$$R_r = \max_{x_i \in P_r} |x_i - C_r|. \tag{3.20}$$

The following parameter $\alpha > 1$ serves as criterion to define whether P_s and P_r are neighbors (compare with 3.13)

$$P_s \text{ and } P_r \text{ are neighbors } \Leftrightarrow |C_r - C_s| \leq \alpha(R_r + R_s). \tag{3.21}$$

Let us introduce the matrix counterpart M of matrix $\mathcal{M}$ presented in Chapter 1 for the electromagnetic context

$$M_{i\,j} = \begin{cases} \frac{e^{ik|x_i - x_j|}}{|x_i - x_j|} & i \neq j \\ 1 & i = j \end{cases} \tag{3.22}$$

The matrix M can be decomposed following the scheme above, namely $M = M^{far} + M^{near}$ where

$$M^{far}_{i\,j} = \begin{cases} 0 & x_i \text{ and } x_j \text{ belong to neighbor boxes} \\ M_{i\,j} & \text{otherwise} \end{cases} \tag{3.23}$$

and the obvious definition for M^{near}. M^{near} is formed within the method of moments, whereas the M^{far} is built by means of the FMM. The expression (3.19) allows us to compress the matrix as we indicated in the introductory example. Unlike in the example opening this chapter, we define below the main components of the FMM we will need in the sequel. Recall that we are interested in a fast matrix-vector product algorithm for $u = M^{far}v$. With this goal in mind and reviewing the steps already mentioned, we

- create the **radiation functions** $F_r(s_p)$ for each box P_r. These are defined as follows

$$F_r(s_p) = \sum_{x_j \in P_r} v_j e^{ik<s_p, C_r - x_j>} \tag{3.24}$$

- calculate the **transfer functions**. For this purpose we define

$$G_r(s_p) = \sum_{P_t \text{ far from } P_r} F_t(s_p) T_{l, C_r - C_t}(s_p), \tag{3.25}$$

  where "far from" means that the boxes P_s and P_r are not neighbors.

- obtain the value of u_i

$$u_i = \sum_{s_p} \omega_p G_r(s_p) e^{ik<s_p, x_i - C_r>}. \tag{3.26}$$

It is not difficult to check that the steps listed above lead to the desired matrix-vector product. For a proof see [26].
Supposing the points to be uniformly distributed over the surface, the memory and CPU complexity for the FMM scales as $\mathcal{O}(N^{3/2})$ [24, 26].

3.5 Multilevel Fast Multipole Method

The Multi Level Fast Multipole Method(MLFMM) allows to obtain a complexity of order $\mathcal{O}(N\log N)$, both for memory and CPU.
In this method the quadrature points on the surface of the scatterer are grouped into a hierarchy of boxes with a characteristic Al length that allows us to identify the level. The fact that our kernel becomes smoother with increasing distance suggests a grouping based on spatial proximity. In order to distinguish between far-field and near-field interactions we use the relative group sizes or levels and their separation. Hence the interaction between well separated groups can be computed using a far-field approximation. We point out that for well-separated groups we compute the interactions among them as groups, whereas for closely located groups we compute the interactions among them at the level of individual basis functions. We insist on this issue because in a forthcoming chapter we will try to see part of the near-field as a sort of *near* far-field.
In order to implement the MLFMM algorithm, the entire object is enclosed into a cube, which is partitioned into eight smaller cubes. Each sub-cube is then recursively subdivided into smaller cubes until the edge length of the smallest cube is about half the wavelength of the illuminating plane wave. Cubes at all levels are indexed and only nonempty cubes are recorded in the hierarchical data structure. This kind of recursion generates what is widely known as oct-tree [4]. In the sequel we will call the leaves, i.e., the leaf-boxes, of the resulting oct-tree also the MLFMM groups. Similarly, the previously mentioned boxes in the MLFMM algorithm will be often called groups.

3.6 A Matrix Representation of the FMM

In the literature several attempts can be found that express the MLFMM algorithm in terms of matrices. See [85, 11].
We consider the surface of a PEC scatterer, on which an incident electric field gives

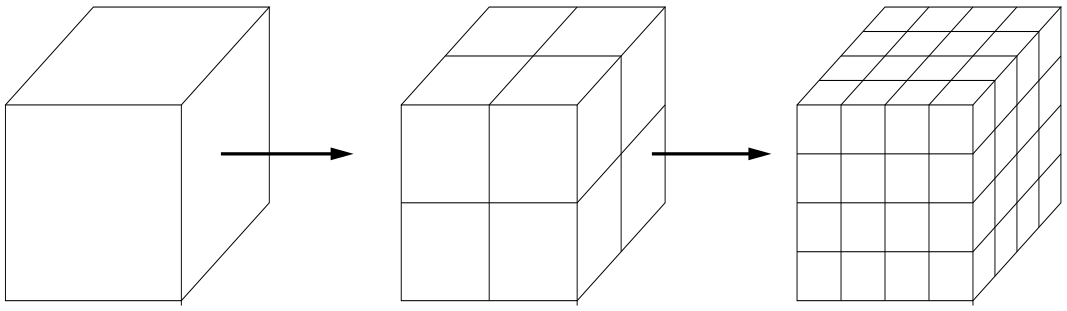


Figure 3.7: *Oct-tree for three levels.*

rise to a surface current. Let us suppose that we have N points $\{x_i\}_{i=1}^{N}$. The points on the surface of the scatterer are called Gauss quadrature points since we are dealing with an integral equation and the application of the method of moments demands numerical integration.
These N points are divided in M groups $B_1, \ldots, B_M$ with the corresponding center points $X_1, \ldots, X_M$.
The points living in the group B_i are $x_1^i, \ldots, x_{m_i}^i$ and $m_1 + \ldots + m_M = N$.
Adopting the same notation as for (3.18), for $|d| < |R|$ we obtain

$$\frac{e^{ik|R+d|}}{|R+d|} \sim ik \sum_{k=1}^{K} \omega_k e^{ik<s_k,d>} T_{l,R}(s_k), \tag{3.27}$$

where $\{s_k\}_{k=1}^{K}$ are discrete directions on the sphere S^2 and $\{\omega_k\}_{k=1}^{K}$ the corresponding quadrature weights.
The FMM thus approximates $\frac{e^{ik|R+d|}}{|R+d|}$ locally by superposing a finite number of plane waves with direction of propagation s_k and amplitude $T_{l,R}(s_k)$ [26].

$$T_{l,R}(s) = \frac{1}{4\pi} \sum_{m=0}^{l} i^m (2m+1) h_m^{(1)}(k|R|) P_m(cos(s,R)). \tag{3.28}$$

Taking $R = X - Y$ we have a kernel splitting:

$$\frac{e^{ik|x-y|}}{|x-y|} \approx \sum_{k=1}^{K} \omega_k e^{ik<s_k,(x-X)>} T_{l,R}(s_k) e^{ik<s_k,(Y-y)>}. \tag{3.29}$$

For two well separated points x_1^i and x_1^j in the groups i and j the formula, with $R_{ij} = X_i - X_j$, yields

$$\frac{e^{ik|x_1^i - x_1^j|}}{|x_1^i - x_1^j|} \approx \sum_{k=1}^{K} \omega_m e^{ik<s_k,(x_1^i - X_i)>} T_{l,R_{ij}}(s_k) e^{ik<s_k,(X_j - x_1^j)>}. \tag{3.30}$$

In order to obtain a matrix form of the FMM algorithm, we follow [26]. There we find how to carry out a product with the far-field matrix and a vector. The matrix M is decomposed as $M = M^{far} + M^{near}$ according to the distance of the boxes. We compress the M^{far} matrix by means of the FMM.
The author in [26] computes the matrix-vector product, $u = M^{far}v$, summing in the quadrature points. A matrix-vector product with the matrix

$$M^{\text{far}} = \left(\frac{e^{ik|x_i - x_j|}}{|x_i - x_j|}\right)_{i,j} - M^{\text{near}} \tag{3.31}$$

is carried out factorizing the M^{far} matrix.

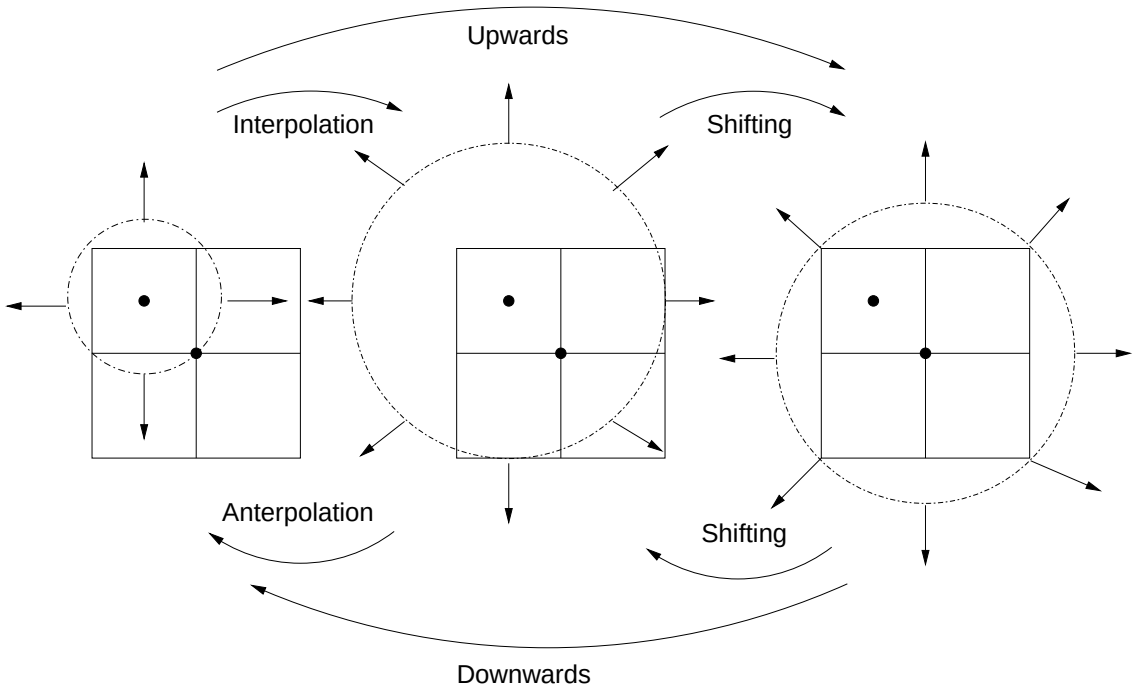


Figure 3.8: *Processes involved in the MLFMM [74].*

Figure 3.8 shows a graphical representation of the multilevel algorithm with the increasing number of directions in S^2 to perform the numerical integration as the size of the boxes increases. In order to represent the FMM algorithm in terms of matrices, we need to follow the steps mentioned above and find its matrix counterpart.
We need radiation functions for every box. For this purpose, we build a matrix V_i with K columns (as many as directions in the sphere) and m_i rows (the number of points in B_i.

$$V_i := \begin{pmatrix} w_1 e^{iks_1\cdot(X_i - x_1^i)} & w_2 e^{iks_2\cdot(X_i - x_1^i)} & \dots & w_K e^{iks_K\cdot(X_i - x_1^i)} \\ w_1 e^{iks_1\cdot(X_i - x_2^i)} & w_2 e^{iks_2\cdot(X_i - x_2^i)} & \dots & w_K e^{iks_k\cdot(X_i - x_2^i)} \\ \vdots & \vdots & & \vdots \\ w_1 e^{iks_1\cdot(X_i - x_{m_i}^i)} & w_2 e^{iks_2\cdot(X_i - x_{m_i}^i)} & \dots & w_K e^{iks_K\cdot(X_i - x_{m_i}^i)} \end{pmatrix} \tag{3.32}$$

The acceleration of the matrix-vector product is due to the fact that several basis functions close to each other can use the same translation operator to transfer their contributions to another area holding other basis functions. For this reason, we express the transfer function as a diagonal matrix that scales the whole matrix V_i. In other words, the transfer depends only on the distance between the centers of the boxes.
For $a_1, a_2, \dots, a_n \in \mathbb{C}$ we define the diagonal matrix

$$\operatorname{diag}_{1\le m\le n}(a_m) := \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{pmatrix}.$$

It yields

$$T_{i,j} := \operatorname{diag}_{1\le k\le K}\left(T_{l,R_{ij}}(s_k)\right).$$

For boxes i and j, that do not couple via the FMM we have $T_{i,j} = 0$. Coupling via FMM means here that the groups are not neighbors. Finally we define

$$W_j := \begin{pmatrix} e^{iks_1\cdot(x_1^j - X_j)} & e^{iks_1\cdot(x_2^j - X_j)} & \dots & e^{iks_1\cdot(x_{m_j}^j - X_j)} \\ e^{iks_2\cdot(x_1^j - X_j)} & e^{iks_2\cdot(x_2^j - X_j)} & \dots & e^{iks_2\cdot(x_{m_j}^j - X_j)} \\ \vdots & \vdots & & \vdots \\ e^{iks_K\cdot(x_1^j - X_j)} & e^{iks_K\cdot(x_2^j - X_j)} & \dots & e^{iks_K\cdot(x_{m_j}^j - X_j)} \end{pmatrix}.$$

The matrix form for the interaction between two groups i and j far enough from each other is the following

$$M_{ij} \approx V_i T_{ij} W_j. \tag{3.33}$$

Note that the size of M_{ij} is $m_i \times m_j$. As an example, we can factorize the block of the far-field matrix corresponding to the block G_1G_4 of the introductory example in

Section 2 of this chapter.

$$\begin{pmatrix} \frac{e^{ik|x_1^1-x_1^4|}}{|x_1^1-x_1^4|} & \dots & \dots & \frac{e^{ik|x_1^1-x_{m_4}^4|}}{|x_1^1-x_{m_4}^4|} \\ \frac{e^{ik|x_2^1-x_1^4|}}{|x_2^1-x_1^4|} & \dots & \dots & \frac{e^{ik|x_2^1-x_{m_4}^4|}}{|x_2^1-x_{m_4}^4|} \\ \vdots & \vdots & & \vdots \\ \frac{e^{ik|x_{m_1}^1-x_1^4|}}{|x_{m_1}^1-x_1^4|} & \dots & \dots & \frac{e^{ik|x_{m_1}^1-x_{m_4}^4|}}{|x_{m_1}^1-x_{m_4}^4|} \end{pmatrix} \approx V_1 T_{14} W_4.$$

The factorization of the whole far-field can be obtained by introducing previously the following block diagonal matrix for the matrices $V_1, V_2, \dots, V_p$

$$\mathrm{diag}_{1\le i\le p}(V_i) := \mathrm{diag}(V_1, V_2, \dots, V_p) := \begin{pmatrix} V_1 & 0 & \dots & 0 \\ 0 & V_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & V_p \end{pmatrix}. \tag{3.34}$$

The aforementioned factorization yields for an integer p (depending on the number of groups)

$$M^{\mathrm{far}} = \mathrm{diag}(V_1, V_2, \dots, V_p)\, T\, \mathrm{diag}(W_1, W_2, \dots, W_p) \tag{3.35}$$

with

$$T = \begin{pmatrix} T_{1,1} & T_{1,2} & \dots & T_{1,p} \\ T_{2,1} & T_{2,2} & \dots & T_{2,p} \\ \vdots & \vdots & & \vdots \\ T_{p,1} & T_{p,2} & \dots & T_{p,p} \end{pmatrix}. \tag{3.36}$$

Back to the introductory example at the beginning of this chapter we would have in one level in 1D the following far-field

$$M^{\mathrm{far}} = \mathrm{diag}(V_1, V_2, V_3, V_4) \begin{pmatrix} 0 & 0 & T_{1,3} & T_{1,4} \\ 0 & 0 & 0 & T_{2,4} \\ T_{3,1} & 0 & 0 & 0 \\ T_{4,1} & T_{4,2} & 0 & 0 \end{pmatrix} \mathrm{diag}(W_1, W_2, W_3, W_4). \tag{3.37}$$

The single level MLFMM can be considered as an approximate factorization

$$M \approx M^{near} + \mathrm{diag}(V_1, V_2, V_3, V_4) \begin{pmatrix} 0 & 0 & T_{1,3} & T_{1,4} \\ 0 & 0 & 0 & T_{2,4} \\ T_{3,1} & 0 & 0 & 0 \\ T_{4,1} & T_{4,2} & 0 & 0 \end{pmatrix} diag(W_1, W_2, W_3, W_4).$$

The computational advantages come from the fact that the far interactions are represented by a product of three sparse matrices. The matrices $\{V_i\}_{i=1,...,4}$ represent the far-field radiation pattern of the sources in the $j-th$ group. The T_{ij} matrices are the translation operators and map outgoing plane waves radiated away from the $j-th$ group to incoming plane waves received by the $i-th$ group. These operators are computed from a series that is related to the field radiated by multipoles giving the algorithm its name.
This formula has a simple interpretation in terms of Figure 3.9. This figure shows the

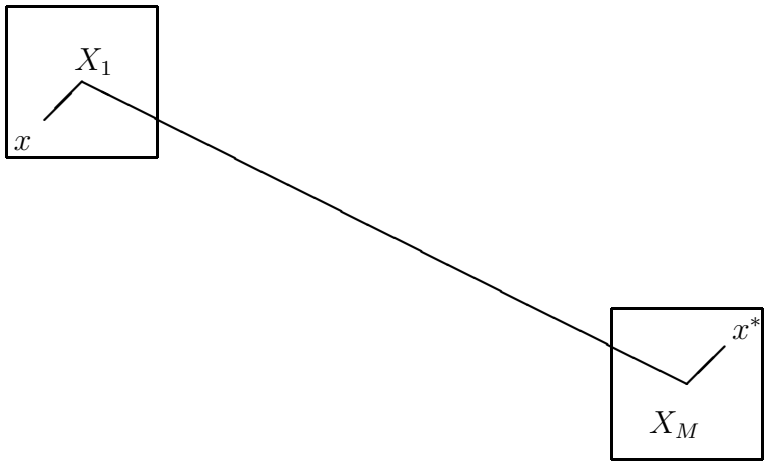


Figure 3.9: *Basic idea of the FMM*

fundamental idea underlying the FMM, namely to compute the interactions between groups instead of individuals based on the physically reasonable assumption that the interaction between x and $x*$ or in general every point x and x^* sufficiently near to the respective center of its box, X_1 and X_M, can be well approximated considering the interactions between the centers of the boxes.

3.6.1 Multilevel FMM

The introduction of more levels adds more complexity to describe the FMM in terms of matrices. To this end let L_{max} be the maximum number of levels in the oct-tree. Following again [26] it is possible to reconstruct the complete far-field of M according

to the following recursive rule

$$\begin{aligned} M &= M_1^{far} + M_1^{near} \\ M_1^{near} &= M_2^{far} + M_2^{near} \\ M_2^{near} &= M_3^{far} + M_3^{near} \\ &\vdots \\ M_{L_{max}-1}^{near} &= M_{L_{max}}^{far} + M_{L_{max}}^{near} \end{aligned}$$

Only at the level L_{max} do we have a factorization as in the previous section (no need

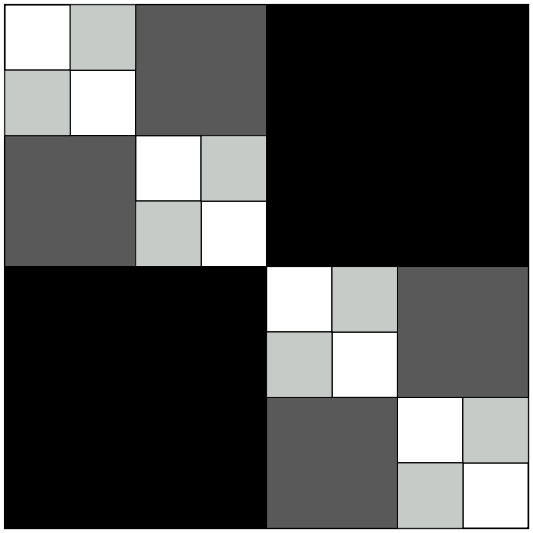


Figure 3.10: *Far-field component colored according to the different levels.*

of transfers). For the other levels we have to transport the information through the different levels until we come to the level where the groups involved couple via MLFMM. In the level where the coupling takes place, the previously mentioned transfer occurs. Unlike in the single level FMM, coupling here is a synonym of being in the interaction list. Following [26] we have that the *Interaction Set*(n) = {groups i such that i is a child of a neighbor of parent(n) but i is not itself a neighbor of n} We need to compute an interaction set for every box in every level. In other words, the far-field for a box is decomposed in the different interaction lists on the different levels. Recall that we consider the leaves level as the lowest level in the oct-tree.
Let us materialize these steps in matrices. Remind that in order to represent the interactions between a pair of groups in the matrix we view their descendants in the tree down to the lowest level and the interactions between the leaves boxes that build them are considered. From the leaves level we transfer the information up in

the tree via interpolation until the coupling takes place according to the pertaining interaction set described above. From there we perform the down-sweep until the leaves level again is reached.
Suppose we want to represent the submatrix pertaining the interaction between two groups i (evaluation group) and j (source group) on level $L_{max}-1$ interacting on this level. We look thus at the children of them (in 3D) on the lowest level with indices $i_1,\ldots,i_8$ and $j_1,\ldots,j_8$, respectively.
The corresponding submatrix looks like

$$M_{i,j}^{L_{max}-1}=\begin{pmatrix} M_{i_1,j_1} & M_{i_1,j_2} & \ldots & M_{i_1,j_8} \\ M_{i_2,j_1} & M_{i_2,j_2} & \ldots & M_{i_2,j_8} \\ \vdots & \vdots & & \vdots \\ M_{i_8,j_1} & M_{i_8,j_2} & \ldots & M_{i_8,j_8} \end{pmatrix}. \tag{3.38}$$

We have to compute the radiation functions at the lowest level ($L=L_{max}$) for the children of the group j. Let X_j, X_i be the centers of the groups j and i at level $L-1$, respectively. Instead of computing the radiation functions of the group j on level $L-1$, W_j, we compute them using the radiation functions of its children and then applying interpolation and translation. In this fashion we save computing time.
To this end let us arrange the matrix accordingly as $W_j=(W_j^1,W_j^2,\ldots,W_j^8)$, where

$$W_j^l=\begin{pmatrix} e^{iks_1^{L-1}\cdot(x_1^j-X_j)} & e^{iks_1^{L-1}\cdot(x_2^{j_l}-X_j)} & \ldots & e^{iks_1^{L-1}\cdot(x_{m_{j_l}}^{j_l}-X_j)} \\ e^{iks_2^{L-1}\cdot(x_1^{j_l}-X_j)} & e^{iks_2^{L-1}\cdot(x_2^{j_l}-X_j)} & \ldots & e^{iks_2^{L}-1\cdot(x_{m_{j_l}}^{j_l}-X_j)} \\ \vdots & \vdots & & \vdots \\ e^{iks_{k_{L-1}}^{L-1}\cdot(x_1^{j_l}-X_j)} & e^{iks_{k_{L-1}}^{L-1}\cdot(x_2^{j_l}-X_j)} & \ldots & e^{iks_{k_{L-1}}^{L-1}\cdot(x_{m_{j_l}}^{j_l}-X_j)} \end{pmatrix}. \tag{3.39}$$

After we compute the radiation functions of the children of group j at level L as above, we apply the interpolation operator and the translation to the center of the father, X_j. We denote the matrix holding the radiation functions for a child of group j by $\widetilde{W_j^l}$ and the interpolation operator by $\mathcal{I}_L^{L-1}$. We will call $E_{j_l,j}^{L-1}=\mathrm{diag}_{1\le m\le k_{L-1}}\left(e^{iks_m^{L-1}\cdot(X_{j_l}-X_j)}\right)$ the translation operator. Reminiscent of the notation for the center of group j, X_{j_l} will represent the center of the group j_l.
These formulas yield the representation of each W_j^l

$$W_j^l=E_{j_l,j}^{L-1}\mathcal{I}_L^{L-1}\widetilde{W}_j^l. \tag{3.40}$$

The assumption that the interaction between the groups i (evaluation group) and j (source group) takes place on level $L_{max}-1$ leads to a transfer on this very level by means of the corresponding transfer function

$$T_{i,j}=\mathrm{diag}_{1\le m\le k_{L-1}}\left(T_{l,R_{ij}}^{L-1}(s_m^{L-1})\right). \tag{3.41}$$

In case the interaction took place on a higher level λ, we would interpolate the radiation functions computed on the finest levels up to level λ and transfer on this level. In order to transfer the information to the finest level we use the transpose of the interpolation operator, namely, the anterpolation operator [26, 22] after we have translated to the center of every child.
The process of anterpolation is much more than a mere transpose of the interpolation [26, 22]. In the latter, an implicit numerical integration is carried out on the surface of the sphere on each level, as though we had uncoupled levels. As a matter of fact, we could imagine the MLFMM algorithm as an arc beginning at the sender group an ending at the receiving group. Remember that in the MLFMM only the leaves communicate. In particular, the steps from the top of the arc to the bottom carry out implicitly the integration on the sphere for each level and transfer the information downwards, preparing the last integration at the leaves level.
It remains the computation of the receiving pattern for the group i, i.e., the matrix

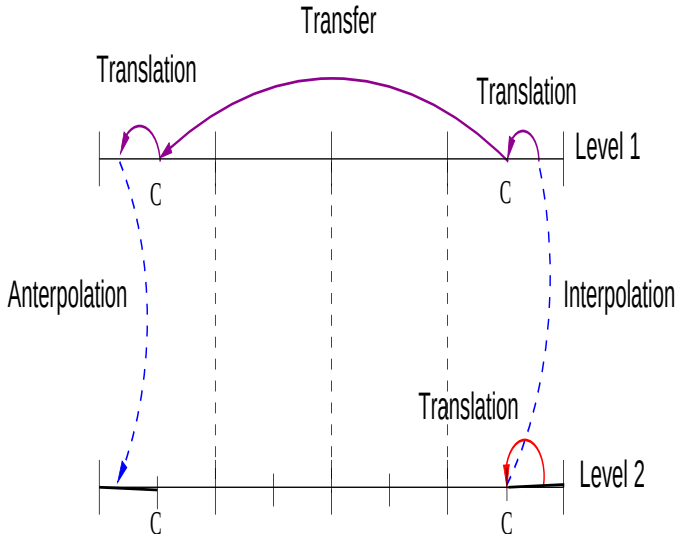


Figure 3.11: *Two-level MLFMM scheme.*

V_i. As in the previous case, we will compute this pattern from that of its children. For this purpose, we write the matrix V_i as follows

$$V_i = \begin{pmatrix} V_i^1 \\ V_i^2 \\ \vdots \\ V_i^8 \end{pmatrix}. \tag{3.42}$$

Each matrix V_i^ν, $\nu = 1, \ldots, 8$, looks like

$$V_i^\nu = \begin{pmatrix} w_1^{L-1} e^{iks_1^{L-1}\cdot(X_i - x_1^{i_\nu})} & w_2^{L-1} e^{iks_2^{L-1}\cdot(X_i - x_1^{i_\nu})} & \ldots & w_{k_{L-1}}^{L-1} e^{iks_{k_{L-1}}^{L-1}\cdot(X_i - x_1^{i_\nu})} \\ w_1^{L-1} e^{iks_1^{L-1}\cdot(X_i - x_2^{i_\nu})} & w_2^{L-1} e^{iks_2^{L-1}\cdot(X_i - x_2^{i_\nu})} & \ldots & w_{k_{L-1}}^{L-1} e^{iks_{k_{L-1}}^{L-1}\cdot(X_i - x_2^{i_\nu})} \\ \vdots & \vdots & & \vdots \\ w_1^{L-1} e^{iks_1^{L-1}\cdot(X_i - x_{m_{i_\nu}}^{i_l})} & w_2^{L-1} e^{iks_2^{L-1}\cdot(X_i - x_{m_{i_\nu}}^{i_l})} & \ldots & w_{k_{L-1}}^{L-1} e^{iks_{k_{L-1}}^{L}\cdot(X_i - x_{m_{i_\nu}}^{i_\nu})} \end{pmatrix} \tag{3.43}$$

Based on (3.43) we can create the corresponding matrices $\{\widetilde{V}_{i_\nu}\}_{\nu=1}^{\nu=8}$ for the children of group i at level L. In order to compute now the receiving pattern of group i at level $L-1$ let us define [22]

$$Q^{L-1} = \mathrm{diag}_{1\le m\le k_{L-1}}(w_m^{L-1}), \quad Q^L = \mathrm{diag}_{1\le m\le k_L}\left(\frac{1}{w_m^L}\right). \tag{3.44}$$

and so we may write

$$V_i^\nu = \widetilde{V}_{i_\nu} Q^L (\mathcal{I}^{L-1})^T Q^{L-1} (E_{i,i_\nu}^{L-1})^H. \tag{3.45}$$

Finally, using (3.40), (3.41) and (3.45) we obtain the matrix representation for the block M_{i_ν,j_l} in (3.38)

$$\widetilde{V}_{i_\nu} Q^L (\mathcal{I}^{L-1})^T Q^{L-1} (E_{i,i_\nu}^{L-1})^H T_{i,j} E_{j_l,j}^{L-1} \mathcal{I}_L^{L-1} \widetilde{W}_j^l. \tag{3.46}$$

In the next chapter we will exploit a column space (row space) relation that holds in the block-rows (block-columns) of the MLFMM matrix. In order to make it apparent, let us turn our attention back to the introductory example of this chapter. In particular, let us consider there the MLFMM matrix with 3 levels. As previously seen, the matrix block of M that represents the interaction between two groups G_{11} and G_{21} at level 2 (See Figure 3.12) looks like

$$M_{G_{11}G_{21}} \approx \begin{pmatrix} V_1^1 \\ V_1^2 \end{pmatrix} T_{1;2}^2 \begin{pmatrix} W_2^1 & W_2^2 \end{pmatrix} = \begin{pmatrix} V_1^1 T_{1;2}^2 W_2^1 & V_1^1 T_{1;2}^2 W_2^2 \\ V_1^2 T_{1;2}^2 W_2^1 & V_1^2 T_{1;2}^2 W_2^2 \end{pmatrix}. \tag{3.47}$$

For the sake of comparison we factorize the matrices in the block which approximates $M_{G_{11}G_{21}}$ as

$$\begin{pmatrix} \widetilde{V}_1^1 Q^2 (\mathcal{I}_2^1)^T Q^1 E_{1,1_1}^1 T_{1;2}^2 E_{2_1}^1 \mathcal{I}_2^1 \widetilde{W}_1^1 & \widetilde{V}_1^1 Q^2 (\mathcal{I}_2^1)^T Q^1 E_{1,1_1}^1 T_{1;2}^2 E_{2_2}^1 \mathcal{I}_2^1 \widetilde{W}_2^1 \\ \widetilde{V}_1^2 Q^2 (\mathcal{I}_2^1)^T Q^1 E_{1,1_2}^1 T_{1;2}^2 E_{2_1}^1 \mathcal{I}_2^1 \widetilde{W}_1^1 & \widetilde{V}_1^2 Q^2 (\mathcal{I}_2^1)^T Q^1 E_{1,1_2}^1 T_{1;2}^2 E_{2_2}^1 \mathcal{I}_2^1 \widetilde{W}_2^2 \end{pmatrix}. \tag{3.48}$$

Following the notation described in Figure 3.12 we can write (3.48) in the following fashion

$$M_{G_{11}G_{21}} \approx \begin{pmatrix} M_{G_{111}G_{211}} & M_{G_{111}G_{212}} \\ M_{G_{112}G_{211}} & M_{G_{112}G_{212}} \end{pmatrix}. \tag{3.49}$$

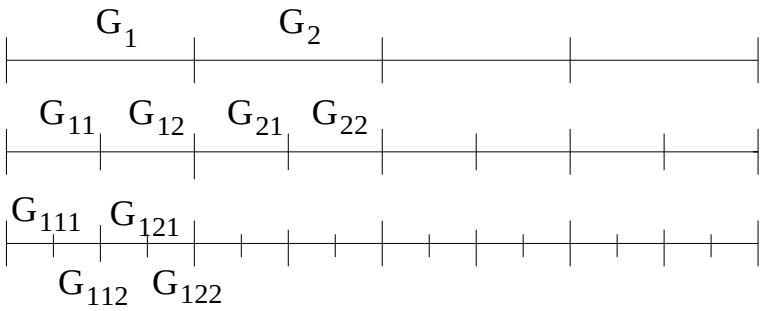


Figure 3.12: *Recursive subdivision of the interval [0,1].*

Chapter 4

Hierarchically Semiseparable Matrices

4.1 Introduction

As we have seen in the examples presented, it is often the case that blocks of matrices resulting from the discretization of integral equations, though dense and apparently full-rank, may be well approximated by low-rank matrices.
The advantages of such an approximation are extremely appealing in terms of operations and storage. A matrix $A \in \mathbb{C}^{m\times n}$ of rank r may be represented as

$$T = \sum_{i=1}^{r} u_i v_i^H,\ u_i \in \mathbb{C}^m,\ v_i \in \mathbb{C}^n.$$

The storage of A requires $r(n+m)$ units of memory and the matrix-vector product would need only $\mathcal{O}(r(n+m))$ operations against $\mathcal{O}(mn)$.
In many areas such as the numerical simulation of electromagnetic fields, potential theory, regularization methods for ill-posed problems, just to name a few, one encounters matrices $A \in \mathbb{C}^{m\times n}$ possessing large off-diagonal submatrices with the property mentioned above. It is therefore reasonable to pursue this direction and consider low-rank approximations to the off-diagonal blocks by means of algebraic or analytical methods, like in the FMM algorithm. The improvement in terms of operations and storage are in turn the basis for the widely-known as fast methods. In fact, fast methods are also referred to as matrix compression algorithms.
In particular, and as shown in Subsection 3.2.2, the low-rank structure in the off-diagonal blocks of the resulting matrix allowed us to approximate them very accurately and uniformly by means of low-rank approximations, obtaining even an exponential decay on the rank k.
This kind of phenomena had been previously observed in several areas and conforms the basis for well known methods like the already mentioned FMM methods, the $\mathcal{H}$-matrices [48, 49, 41] or the mosaic-skeleton technique [38, 39].
In the attempt to characterize the observed matrix structure and provide fast operations for this type of matrices several streams of research emerged in the last twenty years. Unfortunately these have been evolving scarcely aware of each other. As a result, we encounter a realm of names for this and closely related matrix structures and the operations related. In the sequel we will restrict our attention to a special

type of semiseparable matrices [35, 31, 32, 28].

4.1.1 Semiseparable Matrices

The concept of semiseparable matrices was first introduced in [35]. Following [32] we define

Definition 5
A matrix $S = \{s_{ij}\}_{i,j=1}^{N}$ is called **semiseparable of order n** if

$$s_{ij} = \begin{cases} u(i)v(j) & 1 \le i < j \le N, \\ p(i)q(j) & 1 \le j < i \le N, \\ 0 & i = j \end{cases} \tag{4.1}$$

where

$$p = \begin{bmatrix} p(1) \\ \vdots \\ p(N) \end{bmatrix},\ u = \begin{bmatrix} u(1) \\ \vdots \\ u(N) \end{bmatrix} \text{ and } q = [q(1), \ldots, q(N)],\ v = [v(1), \ldots, v(N)]$$

are, respectively, $N \times n$ and $n \times N$ matrices, $p(k), u(k), 1 \le k \le N$ are n-dimensional rows, $q(k), v(k), 1 \le k \le N$ are n-dimensional columns.

We consider an $N \times N$ matrix of the form $R = D + S$, where D is a diagonal matrix and $S = \{s_{ij}\}_{i,j=1}^{N}$ is a **semiseparable** matrix.
An alternative definition for semiseparability is the following [65].

Definition 6
A matrix S is called a lower (or upper) semiseparable matrix of semiseparability rank r if all submatrices that can be taken out of the lower (upper) triangular part of the matrix S have rank r and there exists at least one submatrix having rank r.

In [86] an comprehensive presentation not only of the origin of this matrix structure but also detailed historical accounts on results related to semiseparable matrices are given. The kind of matrices we are interested in are another variant of this structure, namely the Hierarchically SemiSeparable (HSS) matrices whose precursor, the Sequentially SemiSeparable matrices (SSS), we will not treat here. See [28, 19] and

the references therein for information about this class of matrices.
In the next section we will try to unravel the mystery around this structure and the connection to the MLFMM algorithm and especially how the MLFMM matrix may be understood as a special case of HSS matrix in which a particular expansion of the Green function has been chosen.
A classical interpretation of the matrix-vector product via the MLFMM depicts the algorithm as a communication network connecting the leaves in the oct-tree to each other by means of hubs [22]. In the next section we will try to identify the matrices representing the *communication* process and what the compression process in this case means.

4.2 Hierarchically Semiseparable Matrices

We begin the section with what we will call in the sequel the HSS representation of a matrix A [20].

Definition 7
The **HSS representation of a matrix A in K levels** consists of the following sequences of matrices

$$\begin{aligned}&\{D_i\}_{i=1}^{2^K}, \{U_{K;i}\}_{i=1}^{2^K}, \{V_{K;i}\}_{i=1}^{2^K}, \{\{R_{k;i}\}_{i=1}^{2^k}\}_{k=1}^{K},\\ &\{\{W_{k;i}\}_{i=1}^{2^k}\}_{k=1}^{K}, \{\{B_{k;2i-1,2i}\}_{i=1}^{2^{k-1}}\}_{k=1}^{K}, \{\{B_{k;2i,2i-1}\}\}_{i=1}^{2^{k-1}}\}_{k=1}^{K}.\end{aligned} \tag{4.2}$$

This set of matrices represents a data-sparse representation of A. In the sequel we will often refer to, for example, the $\{U_{K;i}\}_{i=1}^{2^K}$ matrices above simply as the U matrices.
Combining these matrices appropriately we could efficiently recover A. In particular, we could reconstruct each off-diagonal block of a hierarchical partition of A. In the matrices above the first subscript represents the level in the hierarchy, whereas the second one conveys information about the block-row (R matrices) and block-column (W matrices), respectively, they are associated to. A possible way to construct the HSS representation of A is the topic of the next section.

4.2.1 The Computation of the HSS representation outlined

The first step to construct the HSS representation in K levels is to define a partition of the matrix, i.e., a partition of the indices of the rows and columns of A. For

the sake of simplicity let us suppose the matrix has been partitioned in 2^K blocks at level K, that is, we have $m_1, m_2, \ldots, m_{2^K}$ different partition sizes. The rule to build the off-diagonal blocks comes as a certain path in a binary tree, the so-called merge-tree, constructed out of the partition sizes $\{m_i\}_{i=1}^{i=2^K}$ or more accurately, from the hierarchical grouping of the indices of A.

Following the notation of [20] we denote $\nu_i = \sum_{j=1}^{i} m_j$. With this notation the i-th leave in the tree contains the indices encompassed by $\nu_{i-1}+1$ to ν_i, with $\nu_0 = 0$. The hierarchical partition results from the association to non-leaf nodes of the indices of their children. This process is depicted in Figure 4.1. In the sequel we will adopt the following convention: the highest level or root will be level 0, whereas the leaf-nodes of the tree will represent its lowest level. At this stage it is necessary to point out

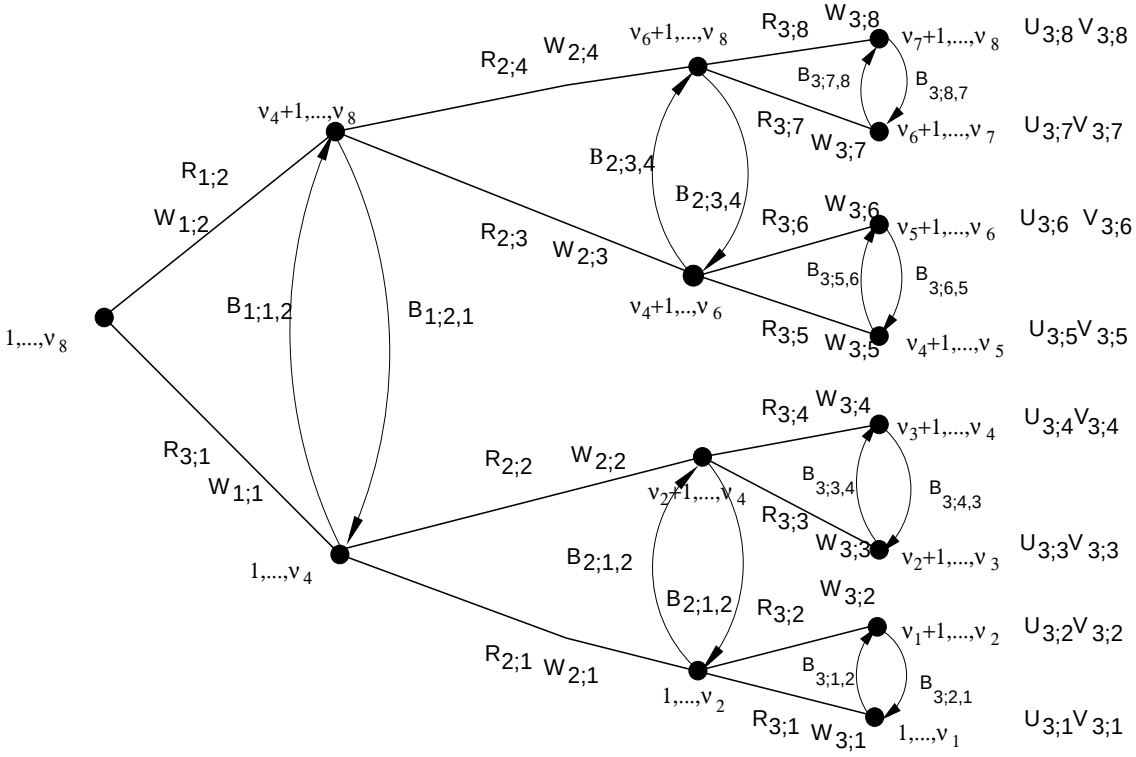


Figure 4.1: *Merge-tree for the HSS structure with K=3. The leftmost part of the merge-tree, i.e. the root, represents the highest level, whereas the lowest level of the merge-tree appears on its rightmost part.*

that the diagonal matrices, $\{D_i\}_{i=1}^{2^K}$ in the HSS representation of a matrix arising in the electromagnetic context previously presented, will be in charge of holding the strong diagonal. We insist on this point since, as we mentioned before, in order this HSS structure to be computable, strong interactions should be around the diagonal.

This is not a problem in one or two dimensions but in three dimensional problems the issue of (re)ordering the unknowns for this purpose is not that simple. Especially for our problem the next picture shows that such a location of the dominant entries in the matrix is far from being true. In particular, this small example corresponding to the rectangular waveguide example (for details see the next chapter) shows that without a suitable renumbering of the basis functions the HSS representation is of no use since we cannot identify the presumably low-rank blocks.
Fortunately, the question of partitioning the matrix can be solved in an elegant way if we consider in detail the MLFMM algorithm and especially the computation of the near-field matrix. The latter takes place at the lowest level, i.e., on the leaves-level of the associated oct-tree. Each leaf-node is thus a MLFMM group containing a set of basis functions that will interact with nearby located groups to build the near-field matrix. If we could order the unknowns in the impedance matrix accordingly, it would yield a matrix, where the strong interactions are in fact located around the diagonal. In other words, the numbering of the MLFMM groups can be adopted as a rule to renumber the basis functions according to the group they belong to. We will address this point later on.
The pictures in Figure 4.2 show the near-field matrix before and after reordering the matrix according to the ordering of the MLFMM groups. More details about the considered permutation will be provided in Chapter 6. Once we have reordered

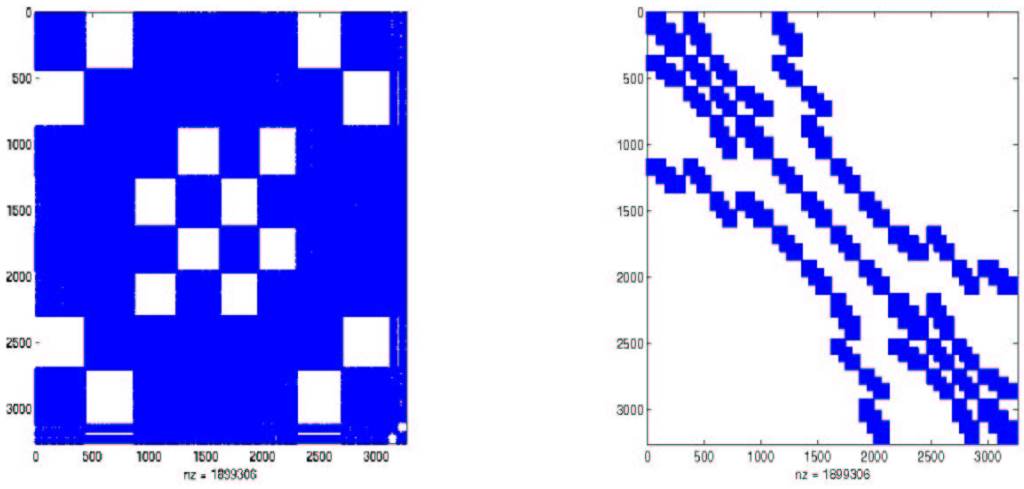


Figure 4.2: *Near-field matrix of the rectangular waveguide before and after reordering according to the MLFMM groups.*

the matrix according to the MLFMM groups, we know where the basis functions of each group are located. Even more important is the fact that the far-field appears in each block-row easily identifiable. At this stage we can establish the connection

of the HSS representation with the matrix form of the MLFMM. At the end of the last chapter we observed how the column-spaces of each block in a block-row were related in the MLFMM matrix for the 1D example. In particular, we identified in the blocks building the first block-row the matrix holding the column-space basis for the whole block-row. This is a fundamental point in the construction of the HSS representation and also in the MLFMM since it expresses algebraically something we took for granted in the communication network interpretation of the MLFMM algorithm. Note the fact that if we regard a block-row, the observation group is always the same. Consequently the information reaches in a block-row the same destination group.
Following the binary tree depicted in Figure 4.1 we would partition the matrix in block-rows and block-columns respectively and compute, for example, a singular value decomposition (SVD) of the matrices forming each block-row. More details about the construction of the HSS representation and some important results concerning the SVD will be provided in Chapter 6.

$$H_{k;i} = \left(A_{k;i,1}\ A_{k;i,2} \cdots A_{k;i,i-1}\ A_{k;i,i+1}\ A_{k;i,i+2} \cdots A_{k;i,2^k}\right). \tag{4.3}$$

Another point to mention is that unlike the HSS representation provided in [20], in the case of complex symmetric matrices, a significant part of the matrices conforming the HSS representation are no longer needed. That is the reason why we confine ourselves to the block-rows and omit the row-space relations mentioned in [20]. The authors in [20] propose an $\mathcal{O}(N^2)$ algorithm for the computation of the HSS representation based on SVDs. We will also follow [20] in this respect. Although the SVD is widely known to be a very expensive algorithm for dense matrices it is also known to provide the best rank-k approximation to a matrix if we compute only the first, say, k-largest singular values of A of the corresponding block.
The crucial point in the computation of the HSS representation is the fact that the U matrices are only computed on the lowest level K. In order to find the bases for the column-spaces in the levels above K, we create small matrices that *interpolate* the row-space and column-space bases, respectively. In particular, the aforementioned interpolation process is the task the R and W matrices, respectively, carry out. The symmetry of the impedance matrix accounts again for $R_{k;i} = \bar{W}_{k;i}$, where the bar stands for complex conjugate.
The interpolation matrices satisfy the following relations

$$U_{k;i} = \begin{pmatrix} U_{k+1;2i-1}R_{k+1;2i-1} \\ U_{k+1;2i}R_{k+1;2i} \end{pmatrix} \tag{4.4}$$

$$V_{k;i} = \begin{pmatrix} V_{k+1;2i-1}W_{k+1;2i-1} \\ V_{k+1;2i}W_{k+1;2i} \end{pmatrix}. \tag{4.5}$$

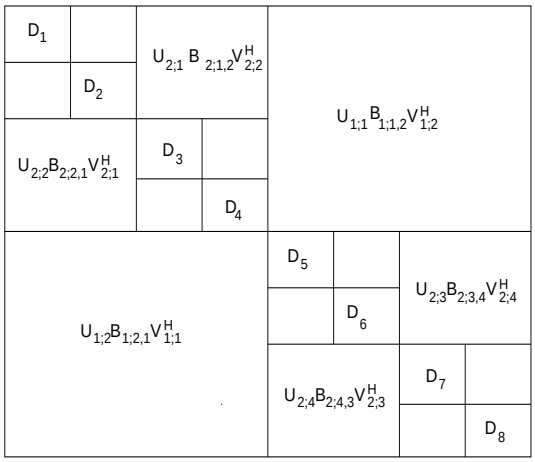


Figure 4.3: *Low-rank factorization of the off-diagonal blocks.*

The authors in [20] provide also a compact notation for the off-diagonal blocks that allows us to represent them as depicted in Figure 4.3. The B matrices are responsible for the transfer between siblings in the tree. They are easily recognizable as the transfer operator we encountered in the MLFMM algorithm. The computation of the B matrices will be addressed in Section 6.2.5.

4.2.2 Connection of the HSS Representation with the MLFMM Matrix of the Introductory Example

As an example we provide the 3×3 leading principal submatrix of a 3 level HSS matrix A. Note the subscripts of the U matrices respect to the block-row where they appear and also the factorization of low-rank blocks by means of products of matrices from the HSS representation

$$\begin{pmatrix} D_1 & U_{3;1}B_{3;1,2}V_{3;2}^H & U_{3;1}R_{3;1}B_{2;1,2}W_{3;3}^HV_{3;3}^H & \cdots \\ U_{3;2}B_{3;2,1}V_{3;1}^H & D_2 & U_{3;2}R_{3;2}B_{2;1,2}W_{3;3}^HV_{3;3}^H & \cdots \\ U_{3;3}R_{3;3}B_{2;2,1}W_{3;1}^HV_{3;1}^H & U_{3;3}R_{3;3}B_{2;2,1}W_{3;2}^HV_{3;2}^H & D_3 & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}. \tag{4.6}$$

Observe, for example, the first block-row of the matrix above. The matrix $U_{3;1}$ appears there on the left hand side of every entry throughout this block-row, thus confirming the previously mentioned relation between the column-spaces in each block-row. A similar relation can be observed, for example, in the first block-column concerning the row-spaces. The matrix $V_{3;1}^H$ appears on the right hand side of every

entry on this block-column.

To conclude this chapter, note the resemblance of the matrix depicted above to the matrix form of the MLFMM which was presented at the end of the third chapter. In particular, the submatrix depicted in (3.48) would correspond to the following submatrix whose first column appears on the right upper corner of (4.6)

$$\begin{pmatrix} \cdots & U_{3;1}R_{3;1}B_{2;1,2}W_{3;3}^{H}V_{3;3}^{H} & U_{3;1}R_{3;1}B_{2;1,2}W_{3;4}^{H}V_{3;4}^{H} & \cdots \\ \cdots & U_{3;2}R_{3;2}B_{2;1,2}W_{3;3}^{H}V_{3;3}^{H} & U_{3;2}R_{3;2}B_{2;1,2}W_{3;4}^{H}V_{3;4}^{H} & \cdots \\ & \vdots & \vdots & \end{pmatrix}. \tag{4.7}$$

Chapter 5

The Solution of the Linear System

5.1 Introduction

Preconditioners for electromagnetic scattering problems and further boundary integral equations (BIE) can be classified into two major categories: those which are applied prior to the discretization and those which we use on the discretized form of the problem. Usually the former are based on properties of the underlying physical problem and may lead to a reformulation of the problem and/or the corresponding BIE formulation while the latter come from matrix theory. Nevertheless, the boundary between these categories is evidently extremely blurred since the derivation of a suitable matrix preconditioner often, if not always, include considerations about the underlying operator and in the end about the physical problem.
In this work we will concentrate on the second class of preconditioners, namely those which play on the matrix theory level. This kind of preconditioner has the advantage that it can be applied on the top of an existing code. In the case of industrial applications where this code generally exists and only a preconditioner is required, these kind of preconditioners are clearly preferred.
To obtain the solution vectors of the discretized problem, it is necessary to solve, in general, a large dense indefinite system of complex valued linear equations. In the three dimensional case the dimension of this coefficient matrix normally requires the application of iterative methods in the solution process due to memory constraints and the fact that in the MLFMM context we only have the sparse A^{near} and never store A^{far}. Only for small targets ($N < 10.000$) a direct solution of the system would not present a major problem. As soon as we want to address electrically larger scatterers we are bound to use iterative techniques. An exception to this rule is the direct solver based in the HSS representation of the matrix and in general techniques that use a compressed storage of the system matrix. The former has been applied successfully in two dimensional scattering problems [20]. Among these methods we find the previously mentioned mosaic-skeleton approximation technique, $\mathcal{H}$-matrices and $\mathcal{H}^2$-matrices.
It is widely known that iterative methods generally have a $\mathcal{O}(N^2)$ complexity per iteration, assuming implicitly the standard matrix-vector product. The matrix-vector product operation represents the real bottleneck of the whole solution pro-

cess. Therefore, we should aim at solving the system in as few iterations as possible. Unfortunately for many realistic problems like the rectangular waveguide already presented, structures with open ended cavities or even more frequently, geometries with surface details like an antenna on an aircraft the resulting nonuniform mesh leads to an ill-conditioned matrix.

As a result, the application of a suitable preconditioner is essential to make the solution of large scale problems feasible. In this chapter we will not present the whole realm of preconditioning techniques used in electromagnetic simulation since it would be an impossible task but rather focus on the major and also most effective ones. In particular, we will describe only, so to speak, matrix-based techniques letting aside the analytic approach considered in [23, 21] and the references therein. The most important class of iterative methods are the **Krylov subspace methods** in which

$$x_n \in x_0 + \mathcal{K}_m \tag{5.1}$$

holds for an iterate approximation x_n to the true solution, where $\mathcal{K}_m$ stands for the Krylov subspace

$$\mathcal{K}_m = \mathcal{K}_m(A, r_0) = \left\{ v \in \mathbb{R}^n | v = \sum_{i=0}^{m-1} c_i A^i r_0 \right\} \tag{5.2}$$

with $r_0 = b - Ax_0$.

The different iterative methods in the Krylov-space context arise basically from the different ways to choose the basis for the $\mathcal{K}_m$. The Krylov subspace method has been extensively treated over the last 40 years, so we avoid an introduction to these type of methods and instead we refer to the following sources [36, 66].

Unfortunately, Krylov subspace methods for complex symmetric non-Hermitian systems are not sufficiently robust as to be applied as black-box solvers. There are examples of system matrices for which each of the usual methods fail to reach a prescribed accuracy or even worse, no convergence is attained.

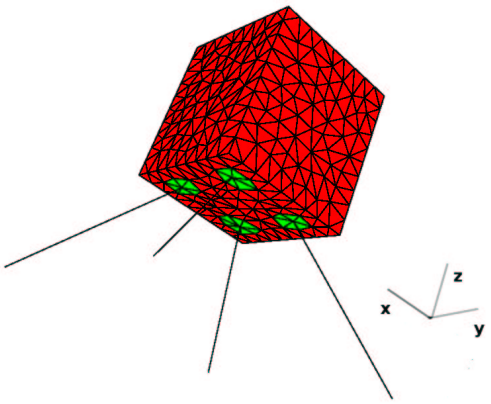
Still, when convergence is not the problem, the computing time is not admissible, since in some cases it may take several months to obtain a approximate solution. These facts make a careful analysis of the problem mandatory.

The convergence behavior of the Krylov methods depends basically on the distribution of the eigenvalues and the condition number of the matrix. Putting it roughly, we would like to cluster the spectrum of the preconditioned system around $(1, 0)$ and far from the origin. This graphical criterion suggests that the spectra represented in Figure 5.1 and Figure 5.2, respectively, will lead to many solver iterations to achieve convergence, a fact that the experiments confirm. The purpose of preconditioning is to make the preconditioned system matrix AM as close to the identity as possible,

i.e., M should approximate the inverse A^{-1}. In the context of the MLFMM it is essential to keep in mind that only the near-field part of the impedance matrix is explicitly stored, the far-field is stored in parametrized form in a more or less easily recoverable fashion. We access the complete system matrix via the matrix-vector operation. Further, a preconditioner on the top of an existing code may be preferable to a new recoding of a significant part of a MLFMM library. Unless we do something like the author in [16] proposes, the only information explicitly available concerning the impedance matrix is precisely this near-field matrix. This matrix holds the largest entries and constitutes the un-approximated portion of the system matrix.

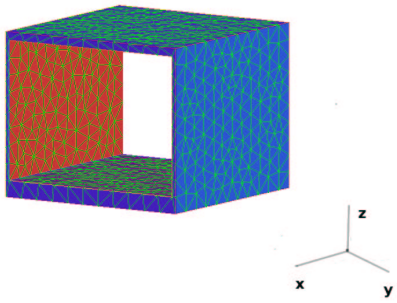
Idealized satellite ($N = 1254$)

Near-Field: 33.92%



Rectangular waveguide ($N = 3264$)

Near-Field: 17.83%



The pictures in Figure 5.1 and Figure 5.2 show the spectra of the two small scatterers (depicted above) and a zoom into the origin. Neither in the first nor in the second example is the desired clustering of the eigenvalues recognizable. The pictures on the right hand side of Figure 5.1 and Figure 5.2, respectively, show the presence of eigenvalues near the origin depicted as the cross-point of the dotted lines.

5.1.1 Performance of some Suitable Iterative Methods

The performance of some iterative methods suited for complex symmetric matrices is in very good agreement with the pictures in Figure 5.1 and Figure 5.2. Even though the examples are extremely small compared to real scale applications, they

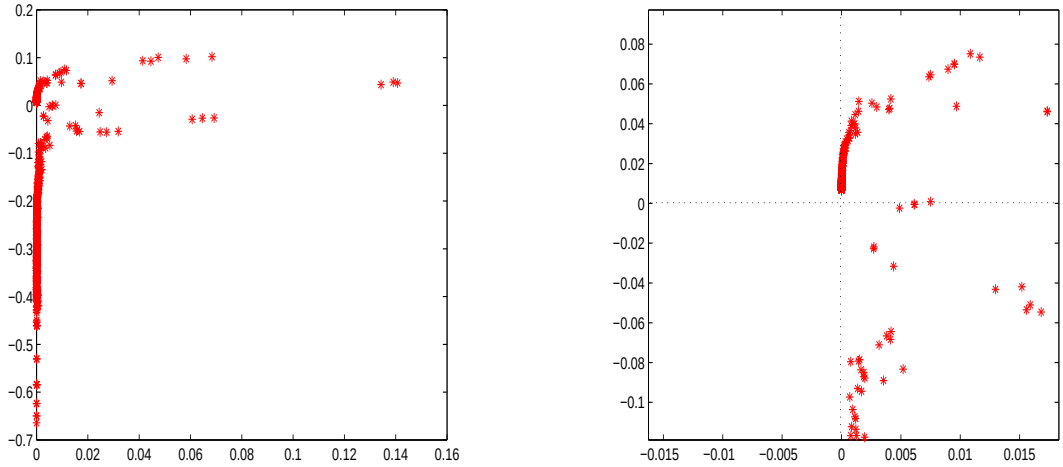


Figure 5.1: *Spectrum of the impedance matrix for the idealized satellite example.*

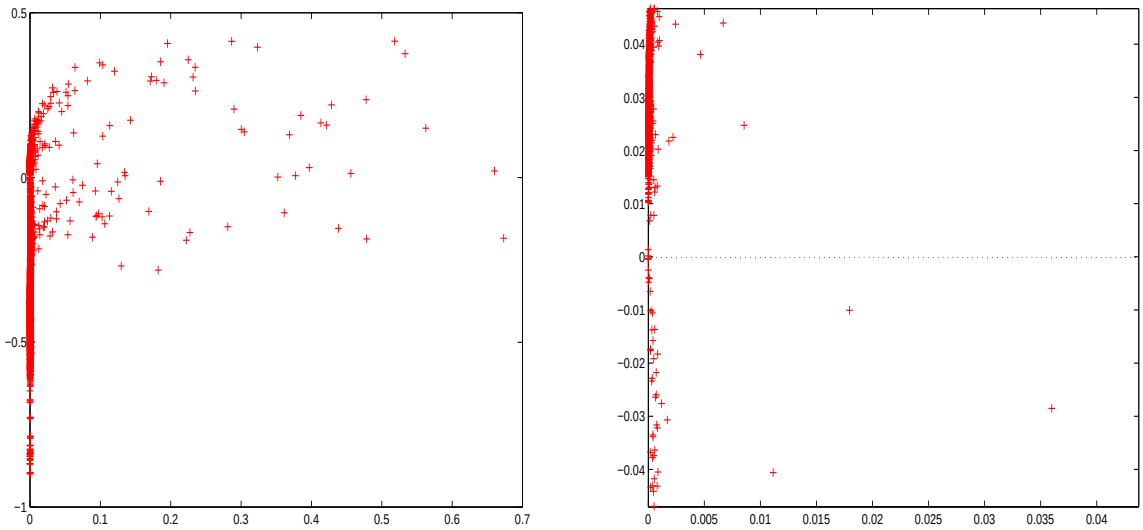


Figure 5.2: *Spectrum of the impedance matrix for the rectangular waveguide example.*

are representative regarding the number of iterations required to solve the respective unpreconditioned systems. Among them the COCG[87] and the CSYM [12], the former a version of the conjugate gradients and the latter a non-Krylov method stemming from a matrix factorization, deserve special attention since both are specially suited for the kind of matrices encountered here. Likewise, the QMR [34] is also ideally suited for complex symmetric indefinite matrices.

Examples	mv products **idealized satellite**	mv products **rectangular waveguide**
GMRES	180	908
CSYM	538	847
COCG	456	-
QMR	244	891

5.2 SPAI and SPAI-based Preconditioners

Among the most popular and effective preconditioning technique for electromagnetic simulation is the Sparse Approximate Inverse (SPAI). It was first described in [45, 58] and further developed for electromagnetic applications in [1, 16]. The SPAI preconditioner M is computed as the matrix which minimizes $||AM-I||_F^2$, satisfying a given sparsity pattern. In other words, a sparse matrix M that approximates A^{-1} is constructed.
For a matrix $A = (a_{ij}) = [a_1, \ldots, a_N]$ and $M = (m_{ij}) = [m_1, \ldots, m_N]$ we have

$$\min \|AM - I\|_F^2 = \min \sum_{j=1}^{N} \|Am_j - e_j\|_F^2 \tag{5.3}$$

$$= \sum_{j=1}^{N} \min \|Am_j - e_j\|_F^2. \tag{5.4}$$

This minimization problem leads to N least-squares problems for a few columns of A each, due to sparsity of M.
The efficiency of the SPAI preconditioner is closely related to the decay of the entries in A^{-1} and to the location of the largest entries in A^{-1} with respect to the largest entries of A. Therefore for certain problems it seems reasonable to use the sparsity pattern of the near-field matrix for the SPAI preconditioner in order to capture the most significant part of the true inverse.
If the sparsity pattern is chosen in advance this preconditioner can be set up independently of the solution process and only matrix-vector multiplication with the sparse M is required. For a thorough study of this preconditioning technique together with some interesting variations of it including underlying information of the problem we point to [16]. The work there has said somehow the last word on SPAI for computational electromagnetic applications.
Either scatterer, the idealized satellite and the rectangular waveguide, depict a relatively large near-field component. This fact has a direct repercussion in the quality of a SPAI preconditioner since we build the matrix M on the near field matrix. The

larger the near field is, the better the SPAI preconditioner results. With increasing dimension the portion of near-field diminishes and therefore the amount of information, though un-approximated, becomes very sparse. In particular, for the examples presented above, the SPAI requires 15 and 74 iterations respectively to achieve a norm of the relative residual less or equal 1e-4. In [16] the author addresses the selection of A in (5.3). As we mentioned before in [26] the only part of the matrix explicitly available is the near-field matrix, therefore first attempts to compute the SPAI preconditioner were based on a near-near approach, namely the computation of M with the sparsity pattern of the near-field as approximate inverse to the near-field matrix in the hope that in this fashion one would obtain a reasonable approximation to the inverse of the full matrix A. In other words, one regards the near-field matrix as an approximation to A in the sense that the strongest interactions are captured in the near field and hopes to approximate somehow the inverse of A.
In [16] the author suggests to enrich the available near-field matrix with underlying information related to the mesh. In particular, it yields $\tilde{A}_{near}$ and consequently the following problem is solved

$$\begin{aligned} \min \|\tilde{A}_{near}M - I\|_F^2 &= \min \sum_{j=1}^{N} \|\tilde{A}_{near}m_j - e_j\|_F^2 \qquad (5.5) \\ &= \sum_{j=1}^{N} \min \|\tilde{A}_{near}m_j - e_j\|_F^2. \qquad (5.6) \end{aligned}$$

In particular, this approach results in a geometrical and a topological version of SPAI whose performance is carefully studied on a wide variety of targets in [16]. It should be emphasized that the sparsity pattern of the near-field matrix is imposed on the matrix M of (5.5). The extra storage of $\tilde{A}_{near}$ with respect to A_{near} is required only in the set-up process not in the solving.
Numerical experiments connecting the density of M and the density of A are also presented in [16]. The bottom line says that the more information we put into the matrix A in (5.3) ((5.5), respectively), the better the performance is likely to be. The pictures in Figure 5.3 show this behavior for the SPAI in the case of the idealized satellite.
These pictures are representative for the behavior of the SPAI preconditioner and show that the case when the SPAI is built from the near-field matrix it is not capable of clustering all eigenvalues to the a region near (1,0) as desired. Consequently a significant number of eigenvalues remain in the vicinity of (0,0). The pictures in the second columns of Figure 5.3 and Figure 5.4 take as A in (5.3) the complete impedance matrix. Note that although the least squares problems to solve become dense we are faced again with a sort of residual of eigenvalues not willing to leave

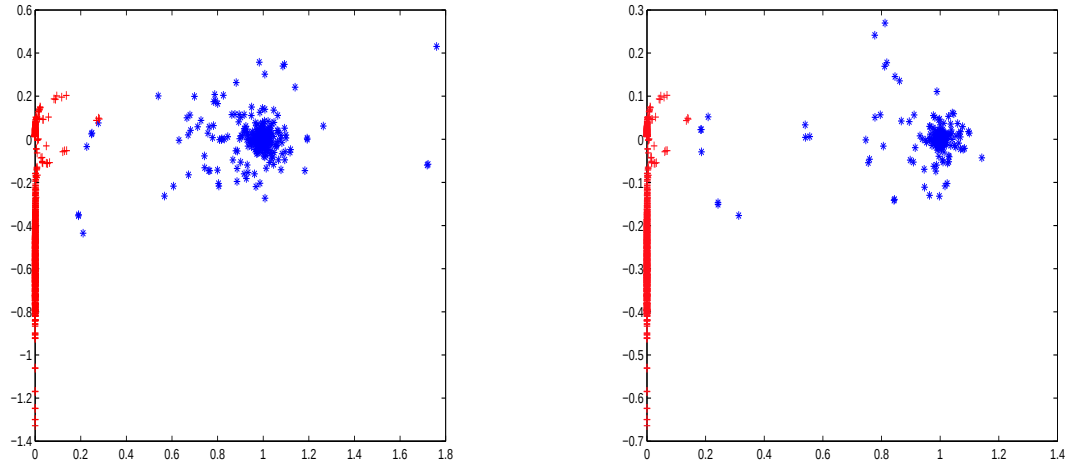


Figure 5.3: *Spectrum of the unpreconditioned impedance matrix (red) vs. SPAI preconditioned impedance matrix for the idealized satellite. SPAI constructed from the near-field and the impedance matrix respectively (both in blue).*

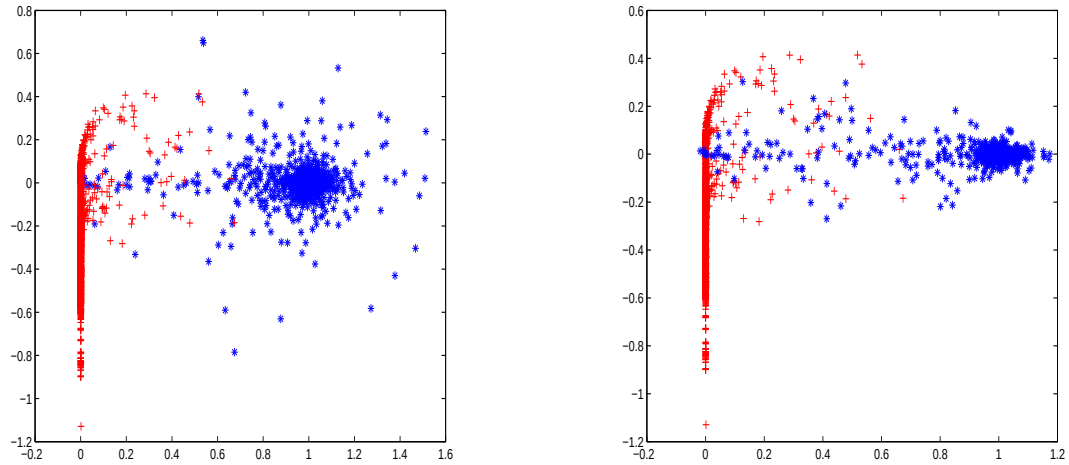


Figure 5.4: *Spectrum of the unpreconditioned impedance matrix (red) vs. SPAI preconditioned impedance matrix for the rectangular waveguide. SPAI constructed from the near-field and the impedance matrix respectively (both in blue).*

the $(0, 0)$ region leading to a poor performance of the iterative methods.
These pictures are feasible due to the size of the examples considered. Care must be exercised when considering them in the sense that it is extremely difficult, if not impossible, to extrapolate the depicted behavior to larger dimensions. The reduc-

tion in the amount of near-field entries has consequences that we cannot predict on the grounds of these examples.

5.3 Incomplete LU (ILU) Preconditioning

The near-field matrix admits an LU factorization $A^{near} = LU$. The matrices L and U can be rather full despite the sparsity of A^{near}. The idea behind ILU(p) is to keep the factors in the LU factorization sparse, and thus incomplete, as an approximation to the true L and U.
The parameter p controls the amount of fill-in in the approximants to L and U. The criterion to determine them is based usually on prescribing a threshold value τ, called the drop-tolerance, that select the entries entering the ILU factorization according to their magnitude, i.e., entries with value smaller than τ are omitted. It is clear that the smaller the τ, the more accurate the approximation will be but in electromagnetic applications it is necessary to find a trade-off between accuracy and storage since the lack of structure of the fill-in and the consequent storage cost places a strong constraint on τ.
The amount of fill-in can be set to zero, thus replicating the sparse format of the near-field. This is the ILU(0) variant. But as claimed in [16, 10] this preconditioning is not effective for the kind of systems we are interested in. It is widely known that the performance increases if more fill-in is allowed. Unfortunately, fill-in has a direct impact on storage. With large problems the size of the L and U factors grow fast and for N around 10^5 these factors may require several Gigabytes. For the case when the impedance matrix is complex symmetric we will store only one of the factors but still the storage of this matrix may be not feasible. If we want a good performance and the factors should dwell in core memory this is usually a problem. In the literature we find authors that perform the factorization on a partitioned matrix and swap the computed blocks between hard and core memory with apparent success [51] .
Another point we will address in the sequel is the fact that dropping criteria, in general, do not distinguish the role an entry plays in the rank structure of a matrix. Clearly, small entries could be crucial for the rank of a matrix but be rejected to enter the ILU factorization on the grounds of their magnitude. We will address this issue again as well as the performance of the ILU preconditioner for the small examples presented above in the next chapter. There we will briefly compare this preconditioner with the new preconditioner based on the HSS approximation to the near-field matrix for the two small scatterers previously mentioned.

5.4 Other Preconditioning Techniques

Among algebraic preconditioning techniques we should mention the two-level preconditioner proposed first for electromagnetic simulation in [17] and further developed in [30]. This technique exploits the core idea of multigrid improving the initial work of a preconditioner. In particular, it tackles situations like the one depicted in the pictures above, where after the SPAI preconditioner has been applied some eigenvalues still remain close to the origin deteriorating the performance of the iterative method.
In a sense the two-level preconditioner is a post-processing technique to a preconditioner since it handles the eigenvalues reluctant to leave their location near the origin after a preconditioner has already been applied.
A different two level scheme is proposed in [40] in the context of the integral form of the Laplace equation. In particular, in this work the interplay between accuracy and run-time in the matrix-vector products computed via the fast multipole is exploited. The two level scheme results from an outer solve with a high resolution matrix-vector product preconditioned by an inner solve based on a lower resolution matrix-vector product. The accuracy of the inner solve is controlled by a variant of the α criterion [4] or the multipole degree. This two level scheme suggested in [40] represents an interesting step towards the exploitation of the special matrix structure of the fast multipole matrix.
The previous idea was successfully applied in [16] in the electromagnetic scattering context. In particular, an inner-outer scheme is proposed consisting of a FGMRES [77] as outer solver and a preconditioned restarted GMRES as inner solver. The preconditioning of the inner iterations are crucial to achieve good performance [16]. This approach has obviously the advantage that unlike the previously mentioned methods that use only information concerning local interactions captured by the near-field matrix, here the preconditioner exploits global information related to the discrete Green's function. However, the utilization of embedded iterations requires the computation of two MLFMM oct-trees and consequently the storage and runtime necessary to set up the preconditioner may represent a problem. It is important to mention, though, that the computations in [16, 40] are carried out in a parallel context.

Chapter 6

A HSS preconditioned GMRES

6.1 Introduction

The preconditioning techniques briefly presented in the last chapter are the most representative and successful one can currently find in the literature. These techniques have in common that almost no information about the special structure of the matrix is exploited. In addition, the decreasing amount of near-field as the size of the scatterer increases, constitutes an obstacle very difficult to overcome.
With the advent of the techniques contributed by the $\mathcal{H}$-matrices, $\mathcal{H}^2$-matrices and also the semiseparable community, in all its variants, considerable progress has been made in designing fast algorithms for basic algebra operations like matrix-vector, matrix-matrix product for the kind of matrices in question. Unfortunately these techniques, although naturally optimal for preconditioning, have not yet produced significant results in this area at least in the electromagnetic scattering context.
As it is known, the basic idea of preconditioning is to transform the original linear system matrix A by an *approximation* to an easily invertible matrix. The first point is clearly satisfied by the previously mentioned techniques since they provide a multilevel low-rank approximation to the matrix. In particular, for implicit preconditioning like the previously mentioned ILU, we look for an approximant M to A, such that $M^{-1} \approx A^{-1}$. Preconditioning with M requires solving a linear system with coefficient matrix M. Of course, solving this linear system should be ideally easier and faster than solving the original linear system with A as coefficient matrix. As far as the inversion is concerned, it is clear that we do not want to invert explicitly but instead a linear system with the *approximant* to A is solved in each iteration and the resulting solution vector is multiplied by the coefficient matrix A. Therefore, for this purpose, namely the invertibility of the preconditioner, a suitable solver, fast if possible, for the kind of matrices involved is needed.
The first chapter showed that the MLFMM matrix is an HSS matrix. Therefore, if we seek for a preconditioner that fulfills the two requirements, namely good approximation properties and easy invertibility, it seems reasonable to look among preconditioners that share the same structure as the MLFMM matrix, thereby have a hierarchically semiseparable structure. Unfortunately, in the context of the MLFMM the only part of the matrix we have explicitly stored so as to build a preconditioner

is the near-field matrix. This matrix is sparse, after a proper permutation even banded, and therefore it makes no sense to talk about a low-rank approximation to its off-diagonal blocks but the question is whether a further splitting of this near-field A_{near} into two matrices A^1_{near} and A^2_{near} can be performed that allows us to enforce the HSS structure on one of the resulting matrices. This step would include a new component in the preconditioning, namely the structure, otherwise neglected. In addition, the latter step constitutes somehow a projection onto the manifold of the HSS matrices and consequently, a good HSS approximation to the A_{near}.

6.2 A Near-Field HSS Based Preconditioner

The solution to the resulting linear system of equations (2.24) is typically obtained by means of an iterative solver, in our case the GMRES [75]. The major reason for choosing GMRES in lieu of other solvers specially designed for complex symmetric matrices like CSYM [12] is the fact that among engineers this solver is by far the most popular as well as the most established due to ample numerical experience.
In order to compute the matrix-vector product required in each GMRES iteration we decompose the matrix into a near-field component and a far-field component. As previously mentioned, the near-field matrix represents the interactions between nearby basis functions, and conforms the un-approximated part of the matrix. For the matrix-vector product with the far-field component the MLFMM is used in each iteration.
First of all, let us define a possible permutation for the near-field matrix in order to concentrate the dominant entries in terms of their absolute value around the diagonal. As stated in the literature, [22, 79], this reordering is a rather difficult issue.

6.2.1 A Permutation Related to the MLFMM Algorithm

Let N be the number of basis functions in (2.22) and let $M_{\lambda_{max}}$ be the number of groups at the finest level λ_{max} in the previously mentioned oct-tree associated to the MLFMM. These groups have been referred to as the MLFMM groups in previous chapters and we will maintain this denomination in the sequel. At the level λ_{max} we will have the groups $\{B^{\lambda_{max}}_{m_\lambda}\}^{M_{\lambda_{max}}}_{m_\lambda=1}$. For each group $B^{\lambda_{max}}_{m\lambda}$ at the lowest level of the oct-tree the near interactions are computed to create the near-field matrix A^{near}. Let us recall the definition of a band matrix [27].

Definition 8

A matrix A is called *band matrix* with *lower bandwidth* b_L and *upper bandwidth* b_U if $a_{ij} = 0$ whenever $i > j + b_L$ or $i < j - b_U$.

Band matrices often arise from discretizing physical problems with nearest neighbor interactions on a mesh provided the unknowns have been ordered properly. The next lemma shows that the MLFMM itself provides such a permutation. Although it does not overcome the reordering problem stated above completely, it represents a good compromise for our purpose, namely that of building a preconditioner based on the HSS structure.

Proposition 9

Let us consider the basis functions in the groups $\{B_{m_\lambda}^{\lambda_{max}}\}_{m_\lambda=1}^{M_{\lambda_{max}}}$ *defined above and let* $\{f_j^{m_\lambda}\}_{j=1}^{s_{m_\lambda}}$ *be the basis functions enclosed in a group* B_{m_λ}*. If we assume, for the sake of simplicity, that the number of basis functions in each MLFMM group at the finest level of the oct-tree is constant and equal to* l*, then there exists a permutation matrix* $\mathcal{P}$ *of dimension* N *such that the symmetrically permuted near-field matrix, i.e.* $PA^{near}P^T$*, is a banded matrix with bandwidth at most* $27l$*.*

Proof
Each basis function has a local index related to the MLFMM group it belongs to. Likewise, a global index number is associated to each basis function. The mapping between the local and global numbering is clearly a bijection. Let us consider the MLFMM groups at level λ_{max} in successive order together with the global indices of their associated basis functions. Let us collect these global indices in successive order into an integer vector p of size N. We will denote the components of p as $\{p_i\}_{i=1}^N$.
This vector p allows the creation of the permutation matrix $\mathcal{P}$ defined as follows

$$\mathcal{P}_{ij} = \begin{cases} 1 & j = p_i \\ 0 & otherwise. \end{cases} \tag{6.1}$$

It is known that the strongest interactions with a basis function $f_j^{m_\lambda}$ belonging to the group m_λ take place within the group m_λ, namely with the other basis functions constituting this group, and with the basis functions in the adjacent groups. The number of adjacent groups in 3D is at most 26. Therefore, in the three dimensional case there can be at most $27l$ nonzero entries in each row of the symmetrically permuted near-field matrix. □

Remark 10
This estimation of the number of nonzero entries per row (or column, due to the

symmetry of the near-field) is extremely generous since neither the number of void groups has been considered nor the fact that in practice it is nearly impossible to assume a constant number of basis functions per group. However, the bottom line is that the reordered near-field matrix has a more convenient banded structure that conveys information concerning the MLFMM groups and their sparsity pattern essential to identify the HSS structure depicted in Figure 4.3.

A further consequence of the lemma is the following remark.

Remark 11
The symmetric permutation of the impedance matrix via $\mathcal{P}$, i.e. $PA^{near}P^T$, concentrates the dominant entries around the diagonal.

Clearly the successive ordering of the groups lacks further information about the relative location of one group to the rest. Nevertheless for the construction of a HSS preconditioner this permutation has very favorable properties as we will see. An important advantage of this permutation is the fact that it has its origin in the MLFMM itself, arising naturally as a successive ordering of the basis functions in the MLFMM groups. A blocking procedure also based on the MLFMM groups was proposed in [68] in the context of the SPAI preconditioning so as to reduce the complexity and further used in [8, 16].

6.2.2 The Near-Diagonal Component of the Near-Field Matrix

If we study the pictures given in Figure 6.1 corresponding to the reordered near-field in more detail an interesting block structure becomes apparent, namely the presence of a near-diagonal block that together with the block diagonal constitutes the near-field matrix. In these pictures this near-diagonal block is seen as two bands of nonzero entries that run almost parallel to the main diagonal. These pictures are reminiscent of the matrix arising in 2D when the Laplace equation with Dirichlet boundary conditions is discretized via finite differences using a lexicographic ordering.
As we pointed out in the chapter devoted to the HSS structure and Figure 6.1 confirms, the best candidates to become the D matrices in the HSS representation are clearly located in the block diagonal part of A^{near} since the strongest interactions

take place there. With this situation in mind the matrix can be written as

$$A = A^{near} + A^{far} = A^{diagonal} + A^{near\_diagonal} + \underbrace{V\Lambda W}_{A^{far}}, \tag{6.2}$$

where $V\Lambda, W$ are sparse matrices (see 3.37). Here, again, the distance between the groups of basis functions accounts for the terms *far* and *near* .
The matrix-vector product may be written consequently as:

$$Av = A^{near}v + A^{far}v = \underbrace{A^{diagonal}v + A^{near\_diagonal}v}_{Sparse\ matrix-vector\ product} + \underbrace{V\Lambda W v}_{MLFMMproduct} \tag{6.3}$$

The pictures in Figure 6.1 represent the near-field matrices after the reordering via the permutation matrix $\mathcal{P}$ defined above.
Although the examples considered are of small dimension, they are not easy to solve. In particular, the first one, the idealized satellite with $N = 1254$ can be considered as representative of real industrial simulations. The presence of antennas and sharp edges in the satellite makes the solution of this problem without preconditioning very expensive in terms of matrix-vector products. In particular, the unpreconditioned GMRES requires as much as 180 iterations to converge to a relative residual norm equal or less than 1e-4.
The difficulty in solving the second example, the rectangular waveguide with $N = 3264$, consists of the presence of resonances that lead to poor conditioning of the impedance matrix. For this example the unpreconditioned GMRES performs even worse requiring 909 iterations to converge to a relative residual norm equal or less than 1e-4.
In Figure 6.1 we observe the banded structure in the permuted near-field matrices. The pictures in Figure 6.2 depict the real part of the entries in the full matrix in either case. Stronger interactions correspond to white entries whereas weak interactions are gray. Again the diagonals still hold the strongest interactions. The percentage of near-field means the amount of near-field entries with respect to the total number of entries in the impedance matrix.

The pictures in Figure 6.3 show the absolute value of the entries in the full matrix for both examples along the anti-diagonal, that is, on the line going from the lower left corner of A to the upper right corner. We can see again confirmed the role of the diagonal block as to capture the largest entries and the near-diagonal holding still larger entries in comparison to the far field but smaller compared to the block diagonal.
As we mentioned before, the solver in [20] performs an implicit ULV^H decomposition that treats the far field and the block diagonal matrix by means of the unitary matrices U and V but as we see in the pictures, if we want the off-diagonal blocks to

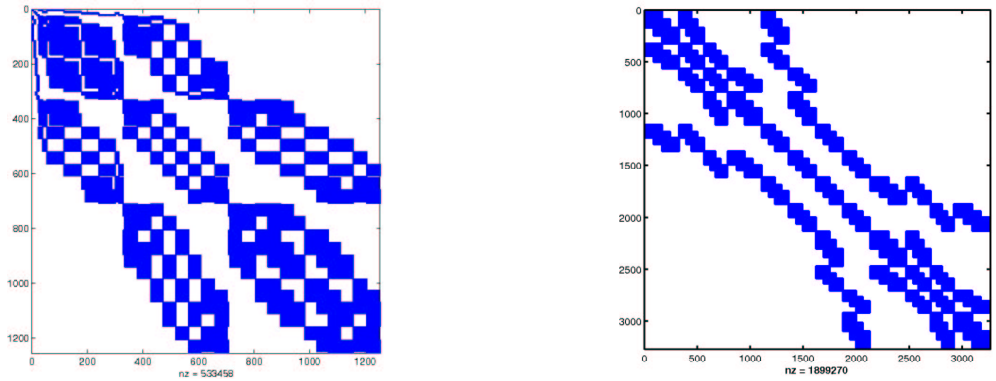


Figure 6.1: *Permuted near-field matrix of the idealized satellite (33.92% of the impedance matrix) on the left hand side and the rectangular waveguide (17.83% of the impedance matrix) on the right hand side.*

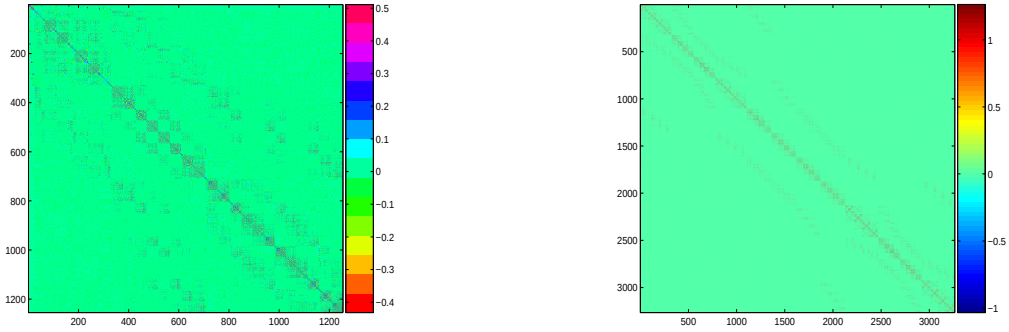


Figure 6.2: *Real part of the scaled impedance matrix for the idealized satellite (left hand side) and the rectangular waveguide (right hand side).*

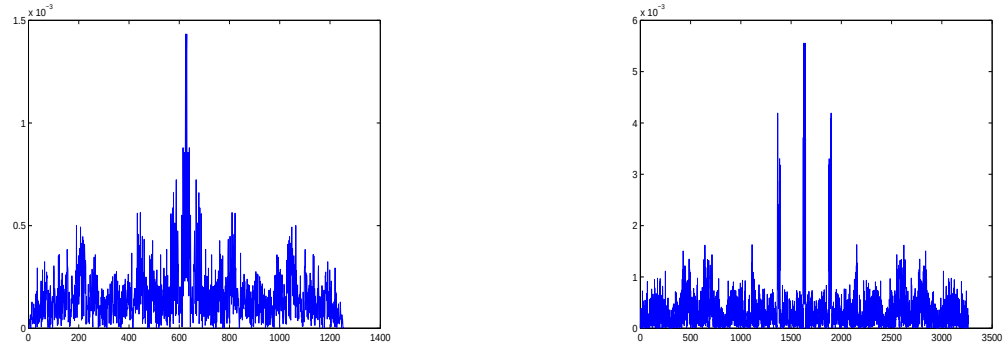


Figure 6.3: *Absolute value of the entries situated on the anti-diagonal of the near-field matrix for for the idealized satellite (left hand side) and the rectangular waveguide (right hand side).*

represent somehow the far field, it seems reasonable to permute the matrix according to the MLFMM groups. We may consider the near-field matrix from a different point of view, namely imposing a multilevel structure on the near-diagonal part of the near-field matrix so as to obtain a HSS matrix. Taking into account that the near-field matrix is computed within the method of moments, this approach constructs a hierarchical approximation to the absent impedance matrix from the only information available explicitly and without knowing the approximation to the impedance matrix provided by the MLFMM. The problem is therefore considered from the point of view of the underlying rank structure rather than at the entry level.
The fact that we should somehow approximate the rank structure of the impedance matrix is supported by our experiments that reflect this rank sensibility when preconditioning is applied to the system, but it has not yet been shown theoretically.

6.2.3 GMRES with Near-Field HSS-based Preconditioning

The basic idea behind preconditioning is to transform the resulting linear system of equations into an equivalent system

$$AMy = b, \tag{6.4}$$

where M is a nonsingular matrix of dimension N. If M is a good approximation to A^{-1} in some sense, then AM will be a good approximation to the identity matrix. As a result the eigenvalues are clustered around $(1, 0)$ leading in general to a faster convergence, as the speed of convergence depends very strongly on the spectrum of AM and the ratio between the smallest and largest eigenvalue or in other words, on how widely they are spread in the complex plane. There are, however, important exceptions to this rule where the right hand side has a decisive influence on the convergence of the method [3, 42, 64].
The strong limitation on the availability of the data in large scale electromagnetic simulation has led to many attempts to exploit to the greatest degree the information contained in the near-field matrix when it comes to building a preconditioner. Figure 6.4 shows the spectrum of the impedance matrix multiplied by the inverse of the near-field matrix for the *idealized satellite* example. Even though the spectrum of $A(A^{near})^{-1}$ appears clustered around $(1, 0)$, many eigenvalues lie still close to the origin, leading to slow convergence. In [16] the author addresses a very important issue in the development of preconditioning techniques in the electromagnetic simulation context, namely the degree of density of the matrix from which we build the preconditioner. Although this issue is addressed mostly in the sparse approximate

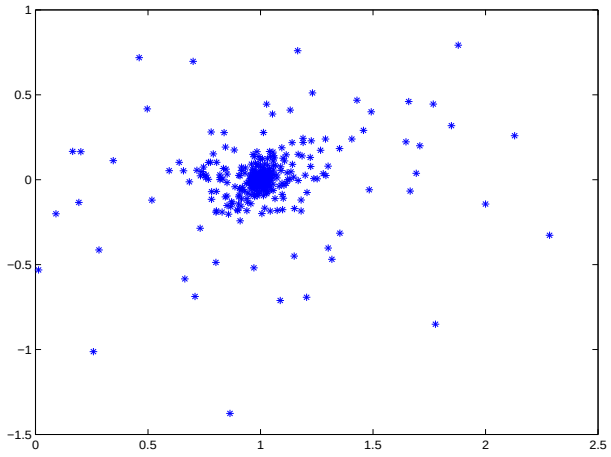


Figure 6.4: *Spectrum of* $A\,(A^{near})^{-1}$ *for the rectangular waveguide.*

inverse framework mentioned before (see Subsection 5.2), its relevance for other contexts can be easily understood if we note that with growing dimension of the problem, the number of entries in the near-field matrix decreases. As a result, the amount of information for building our preconditioner becomes extremely sparse for large dimensions. Further experiments with the near-field based ILU preconditioning show that the amount of fill-in in the preconditioner is a crucial issue for the performance of the latter [80, 16].

So far, people have tried to control the amount of fill-in on the entry level ignoring the underlying structure of the operator. In the dropping process there is no way to select the entries appearing in the preconditioner according to the importance they have with respect to the rank of the matrix-block of the MLFMM matrix they belong to. It is clear that small entries may have a significant influence on the rank of this submatrix. Therefore, it seems that in the dropping process, valuable information about the matrix we want to approximate may get lost and, as we have seen, the rank structure is precisely the most fundamental feature of the MLFMM matrix and the one which makes this algorithm so powerful.

In light of this fact, we propose a rank-fill-in that completes the near-field matrix not only on the entry level but also on the rank level leading to a preconditioner that approximates, in some sense still to be investigated, the rank-map of the impedance matrix.

6.2.4 Construction of the Preconditioner

Our preconditioner is based on an approximate HSS representation of the near-field matrix, denoted in the sequel as A_{HSS}^{near} or simply the HSS preconditioner. The approximation stems from the fact that, in the construction of the HSS representation of the near-field matrix, we will initially truncate the SVDs (Singular Value Decomposition) involved to a tolerance. Later on we will fix the number of singular values and singular vectors to be computed and denote it by nsv. As a result, we will obtain a multilevel low-rank approximation to the near-field matrix.
Now let us recall some fundamental results concerning the SVD that support our claim regarding the HSS preconditioner. For a $m \times n$ sparse matrix A with rank(A)=r, the SVD of A can be defined as

$$A = U\Sigma V^H, \tag{6.5}$$

where U and V are unitary matrices and $\Sigma = diag(\sigma_1, \ldots, \sigma_p)$, $\sigma_i > 0$ for $1 \leq i \leq r$, $\sigma_i = 0$ for $i \geq r+1$. The first r columns of the unitary matrices U and V represent the orthonormalized vectors associated with the r nonzero eigenvalues of AA^H and A^HA, respectively. The singular values of A are the diagonal entries of Σ. The set $\{u_i, \sigma_i, v_i\}$ is called the i-th singular triplet. The norm $||\cdot||$ refers to the Euclidean norm, whereas $||\cdot||_F$ is the Frobenius norm of a matrix.
The next well-known theorems concerning the SVD reveal the importance of this ubiquitous decomposition.

Theorem 12
Let the SVD of A be given by (6.5) and $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$, then the following holds:

i) Rank property: rank(A)=r. Kernel and range of A: Kern(A)$\equiv$ span$\{v_{r+1}, \ldots, v_n\}$, and R(A)$\equiv$ span$\{u_1, \ldots, u_r\}$, where $U = [u_1\, u_2\, \ldots\, u_m]$ and $V = [v_1\, v_2\, \ldots\, v_n]$.

ii) Dyadic decomposition: $A = \sum_{i=1}^{r} u_i \cdot \sigma_i \cdot v_i^H$.

iii) Norms: $||A||_F^2 = \sum_{j=1}^{r} \sigma_k^2$, and $||A||^2 = \sigma_1$.

The rank property allows us to use the singular values of A as a quantitative measure of the rank. The dyadic decomposition provides a representation of a matrix A as a sum of r rank-one matrices of decreasing importance, as measured by the singular values. The combination of the three results above yields the following theorem (See [37] for a proof):

Theorem 13 (Eckart and Young)
Let the SVD of A be given by (6.5) with $r = rank(A) \leq p = min(m, n)$ *and define:*

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^H \text{ with } k < r,$$

then

$$\min_{rank(B)=k} ||A - B|| = ||A - A_k|| = \sigma_{k+1} \tag{6.6}$$

$$\min_{rank(B)=k} ||A - B||_F^2 = ||A - A_k||_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_p^2. \tag{6.7}$$

This important theorem indicates that A_k is the best rank-k approximation (in a least squares sense) to the matrix A. This fact is the essential pillar in the construction of the HSS preconditioner since we will construct later on the best multilevel rank-k approximation to the near-field matrix. This approximant will provide us with a HSS matrix that will, in turn, approximate the impedance matrix.
Back to the construction of the HSS preconditioner, we note that the arising fill-in from the computation of the truncated SVDs is almost exclusively due to the near diagonal component of the near-field matrix as we compute the SVD of the block-rows without the diagonal block on each level. If we further select the number of levels in the HSS representation carefully with the information about the basis functions distribution on the MLFMM groups provided by the SHSS package before the near-field matrix is computed, it is fairly easy to separate the near-diagonal part of the near-field matrix from the block-diagonal part. It could be argued that the cost of the SVDs cannot lead to fast methods. This is only true as long as we have full matrices. In our case, the sparsity level of the block-rows (without the diagonal block) in the rectangular waveguide example is always between 80 and 99%. In the methods used for the computation of the sparse SVD the submatrices are only referenced via a matrix-vector product [60]. As a result, the time complexity for the computation of the HSS representation of the near-field matrix is, according to our experiments so far, less than $\mathcal{O}(N^2)$ although the implementation is at this stage certainly not optimal. This point will be addressed in the appendix devoted to the previously mentioned SHSS package and in the remarks to Proposition 14.
The fill-in takes place at the entry level but even though the matrix becomes a full matrix, we may still store it very cheaply because of the multilevel low-rank structure. In fact, for the examples we present below, the storage cost necessary to obtain a good preconditioner supposes in some cases a minor extra storage with respect to the cost of storing the near-field matrix. Therefore, it is more convenient

than an entry-based filled-in ILU preconditioner that performs similarly. As previously mentioned, the storage of the fill-in required in the ILU case to obtain a good performance represents an important drawback to this preconditioning technique [16]. The HSS preconditioner overcomes this storage problem enforcing a multilevel low-rank structure on the near-field matrix which clearly accounts for the effective storage of the arising fill-in.
In other words, we have as preconditioner a full matrix in terms of entries, but due to the multilevel low-rank structure, we can fulfill the storage constraints we already mentioned. From the practical point of view this preconditioner acts on top of the MLFMM code and therefore no recoding of an already rather cumbersome code is required. The preconditioner can be added to existing code with a moderate effort.

6.2.5 The Computation of A_{HSS}^{near}

Let us assume that the near-field matrix of dimension N has been partitioned according to the partition sizes at the lowest level K of the merge-tree in the HSS representation, $\{m_i\}_{i=1}^{2^K}$. For the levels above K we merge the indices via $\nu_i = \sum_{j=1}^{i} m_j$. The indices $\{\nu_j\}_{j=1}^{2^K}$ together with the merge-tree are depicted in Figure 6.5.
For the sake of simplicity, we assume that the number of basis functions per MLFFM group is constant and equal to l. In a first approach, we will utilize for the approximate HSS representation of the near-field the SVDs of the following sparse matrices

$$H_{k;i} = \left(A_{k;i,1}\ A_{k;i,2} \cdots A_{k;i,i-1}\ A_{k;i,i+1}\ \cdots A_{k;i,2^k}\right),\ k = 1, \ldots, K,\ i = 1, \ldots, 2^k.$$

As previously indicated, the matrix $H_{k;i}$ represents the block-row i on level k of matrix A, where the diagonal block has been removed. Due to the complex symmetry of the near-field matrix only the sequences of matrices U, R and B need to be computed. The near-field matrix, however, will be computed and stored as though it were unsymmetric since it occurs that symmetry can be assessed only up to a certain tolerance in the absolute value of its entries. The sequences V and W are just the complex conjugate of the sequences U and R, respectively. In order to point out the differences with respect to the algorithm provided in [20], let us recall how the computation of these sequences is performed.
Let

$$H_{k;i} = U_{k;i} C_{k;i} J_{k;i}^H \text{ for } k = 1, \ldots, K \text{ and } i = 1, \ldots, 2^k \tag{6.8}$$

be the SVD of $H_{k;i}$. As we observed in the section devoted to the matrix representation of the MLFMM, the column space is common to every off-diagonal block

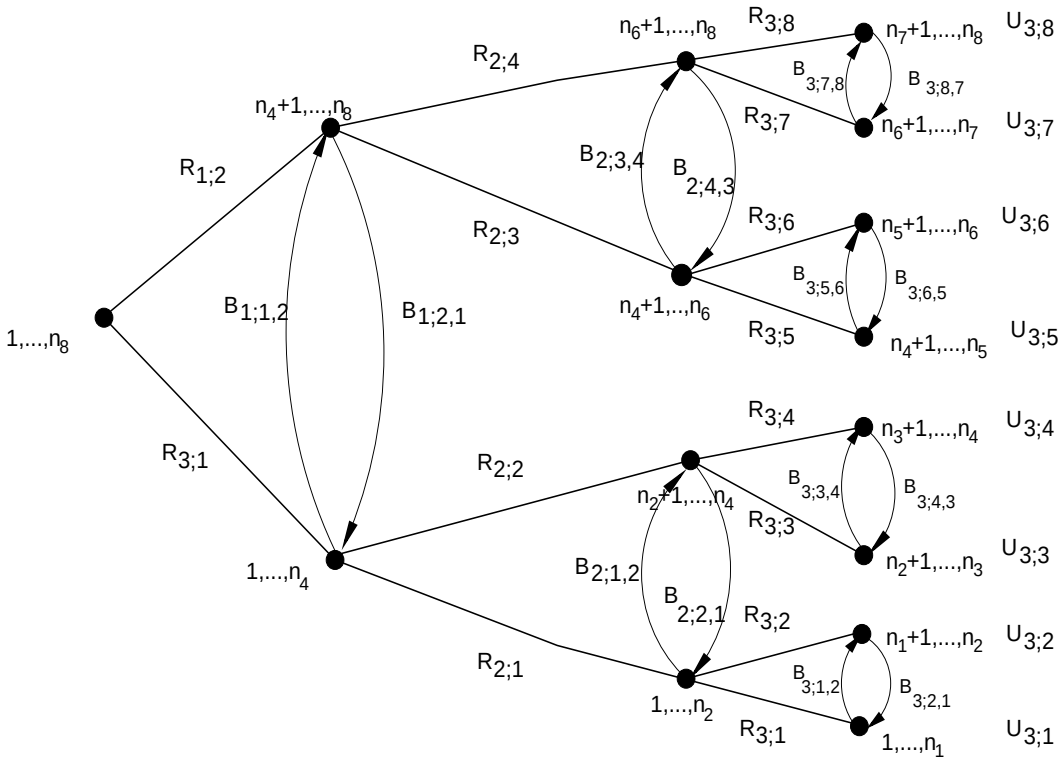


Figure 6.5: *Associated merge-tree to the HSS structure with 3 levels in the complex symmetric case with recursive bisection of the matrix indices.*

on each block-row and therefore the computation of the U matrices in the HSS representation is straightforward. Similarly, imposing the relation concerning the column spaces in the different levels described in Chapter 3 and partitioning $U_{k;i}$ conformingly yields the R matrices in the HSS representation of A_{near}.

$$U_{k;i} = \begin{pmatrix} (U_{k;i})_1 \\ (U_{k;i})_2 \end{pmatrix} = \begin{pmatrix} U_{k+1;2i-1}R_{k+1;2i-1} \\ U_{k+1;2i}R_{k+1;2i} \end{pmatrix}. \tag{6.9}$$

$$R_{k+1;2i-1} = U^H_{k+1;2i-1}(U_{k;i})_1 \tag{6.10}$$

$$R_{k+1;2i} = U^H_{k+1;2i}(U_{k;i})_2 \tag{6.11}$$

For the computation of the B matrices in the HSS representation of A_{near} we note that $A_{k;2i-1,2i}$ is the $2i-1$ submatrix of $H_{k;2i-1}$. Therefore, the following relation is obtained

$$A_{k;2i-1,2i} = U_{k;2i-1}B_{k;2i-1;2i}U^T_{k;2i} = U_{k;2i-1}C_{k;2i-1}(J_{k;2i-1})^H_{2i-1}, \tag{6.12}$$

where $(J_{k;2i-1})_{2i-1}$ represents the appropriate submatrix. The first relation in (6.12) was obtained by using (6.9) recursively.

The above relation, in turn, yields

$$B_{k;2i-1;2i} = C_{k;2i-1}(J_{k;2i-1})_{2i-1}^{H}\bar{U}_{k;2i}. \tag{6.13}$$

Similarly, the $B_{k;2i;2i-1}$ can be obtained as follows

$$B_{k;2i;2i-1} = C_{k;2i}(J_{k;2i})_{2i-1}^{H}\bar{U}_{k;2i-1}. \tag{6.14}$$

Summing up, we could write the HSS preconditioner in terms of the matrices entering its HSS representation as $A_{near}^{HSS} \equiv \{\mathbf{D}, \mathbf{U}, \mathbf{B}, \mathbf{R}\}$, where the bold letters represent the corresponding sequences of matrices as described in Definition 7. The latter relation gives the proposed preconditioner its name as a hierarchically semiseparable preconditioner. For our initial numerical results based on small examples we will resort to a computation of the SVDs based on a threshold for the size of the singular values. This has the disadvantage that no control on the number of singular values to be computed is possible. Consequently, estimating the computational complexity in advance is nearly impossible. However, for the purpose of testing small examples in Matlab this approach is quite convenient.
In realistic applications, however, we will contend ourselves with a fixed number nsv of singular triplets. It is not simple to estimate the computational complexity of the computation of the nsv-largest singular values and the corresponding singular vectors of the previously mentioned block-rows for the construction of the HSS preconditioner. Following the notation introduced above, we denote by $m_{k;i}$ the number of rows of block-row $H_{k;i}$. These block-rows of size $m_{k;i} \times (N - m_{k;i})$ of the near-field matrix with at most $27l$ nonzero entries will depict different singular value distributions depending, among other parameters, on the geometry of the scatterer. It is well known that the cost of the sparse SVDs depends strongly on this distribution. In particular, in the context of Lanczos-type algorithms used in the library PROPACK [60] (one of the main pillars of the SHSS package), the cost involved depends on how fast orthogonality is lost among the Lanczos vectors. For some matrices the orthogonality will be lost very quickly and the number of flops (floating point operations) spent orthogonalizing the Lanczos vectors will be very large. For others, there will be almost no loss of orthogonality and consequently the cost will be dominated by the number of flops necessary to perform the matrix-vector products in each operation.
The cost breakdown per iteration for a block-row B of size $m_{k;i} \times (N - m_{k;i})$ is:

i) Matrix-vector products (one with B and one with B^T) $\approx 4 * (27l * m_{k;i})$

ii) Vector operations $\approx 5 * (m_i + (N - m_{k;i}))$

iii) Gram-Schmidt orthogonalization requires between 0 and $2 * k * (m_{k;i} + (N - m_{k;i}))$, where k is the iteration number)

For a *perfect* block-row where no loss of orthogonality occurs, the total cost is approximately $k_{max} * (4 * (27 * l * m_{k;i}) + 5(m_{k;i} + (N - m_{k;i})))$ flops, where k_{max} is the number of iterations required for convergence. For a *worse* matrix where full orthogonalization has to be performed in every iteration the cost is completely dominated by iii) and the total cost amounts to $k_{max}^2 * (m_i + (N - m_{k;i}))$ flops.
There are theorems, known as Kaniel-Paige convergence theory [55, 69], giving bounds on k_{max} in terms of the singular value distribution of the matrix. However, for the scatterers considered in the sequel our numerical experience supports the assumption $k_{max} < m_{k;i}$.
The fundamental point in the computation of these SVDs is the fact that thanks to the permutation proposed above, the $H_{k;i}$ matrices we are interested in have their nonzero elements around the diagonal and therefore the $(N - m_{k;i})$ entering the aforementioned complexity is divided by a relative large number. Unfortunately, the number of columns entering this near-diagonal part of the near-field matrix is extremely geometry-dependent. As a result, it is nearly impossible to provide sharp estimations concerning the number of nonzero entries that constitute it. Therefore, the complexity estimation mentioned above represents an extremely generous one.

Proposition 14

The computational complexity in the computation of the HSS preconditioner is dominated by the computation of the nsv-largest singular triplets. In particular, under the assumption that $m_{k;i} = m_k$, for $k = 1, \ldots, K$ and that the recursive partition of the matrix indices has been obtained via bisection, the number of flops required to compute the sequences of matrices entering the HSS representation of the HSS preconditioner is of order N^2.

Proof
Let us estimate initially the number of flops required at the lowest level K in the merge-tree of the HSS representation of the near-field matrix to compute its HSS representation. Under the assumption that $m_{K;i} = m_K$ for $i = 1, \ldots, 2^K$ and the further assumption that $k_{max} < m_K$, the computation of the *nsv*-largest singular triplets for the 2^K block-rows requires at most $2^K\ (m_K * (4 * (27 * l * m_K) + 5(m_K + (N - m_K))))$ flops. We may simplify this expression by using the relation $2^K m_K = N$. This substitution yields

$$4 * (27 * l * m_K * N) + 5N^2 \text{ flops.}$$

The multilevel partition of the matrix indices in block-rows via bisection implies $m_k = 2^{K-k} m_K$, where m_k again is the fixed number of rows of each block-row at level $k < K$. As in the previous case, we assume again $k_{max} < m_k$.Therefore, the computation of the nsv-largest singular triplets for the 2^k block-rows at each level k requires

$$\sum_{k=1}^{K-1} \left[2^k \left(2^{K-k} m_K * (4 * 27 * l * 2^{K-k} * m_K + 5(2^{K-k} m_K + (N - 2^{K-k} m_K))) \right) \right] \text{ flops.}$$

Simplifying again it yields

$$\sum_{k=1}^{K-1} \left[2^{-k} * N^2 * 4 * 27 * l + 5N^2 \right] = \left((1 - 2^{1-K}) * 4 * 27 * l + 5(K-1) \right) N^2 \text{ flops.}$$

Collecting terms we obtain the claimed order for the number of flops associated to the construction of the HSS preconditioner. This order concerns only the computation of the nsv-largest singular triplets at each level and each block-row. Now we have to show that the number of flops required to form the HSS matrices is at most of order N^2. For this purpose let us reproduce the construction of the HSS matrices in Matlab-like notation. First let us analyze the computation of the B matrices.
For each block-row i on level k, the nsv-largest singular triplets are collected into the following matrices, $U_{k;i}, C_{k;i}, J_{k;i}$. In order to compute the B matrices at level k, i.e., $\{B_{k;i}\}_{i=1}^{2^k}$, of the HSS representation of the near-field matrix we need to perform the matrix-matrix products described below.

```
for i=1:2^k

    if mod(i,2)==1
```

$$\begin{aligned} B_{k;i} &= C_{k;i} (J_{k;i})_i^H; \\ B_{k;i+1} &= \bar{U}_{k;i}; \end{aligned}$$

```
        else
```

$$\begin{aligned} B_{k;i-1} &= B_{k;i-1} \bar{U}_{k;i}; \\ B_{k;i} &= C_{k;i} (J_{k;i})_{i-1}^H B_{k;i}; \end{aligned}$$

```
    end

end
```

Following the lines above, the number of flops required at level K is

$$2^{K-1}(2 * nsv^2 * m_K + 4 * (nsv^2) * m_K + nsv^3) = 2^{K-1}(6 * nsv^2 * m_K + nsv^3).$$

For level k we need $2^K(3*nsv^2*m_K+nsv^3)$. Therefore, the total cost of computing the B matrices amounts to

$$K(2^K(3*nsv^2*m_K+nsv^3)) = 3K*nsv^2*N+K2^K*nsv^3 \text{ flops.}$$

The matrix-matrix products needed for the computation of the R matrices at level k require $2^{K-(k-1)}*nsv^2*m_K$ flops (see 6.10). Consequently, the number of flops for the complete sequence of R matrices is

$$\sum_{k=1}^{K-1} 2^{K-(k-1)}*nsv^2*m_K = 2*nsv^2*N-4*nsv^2*m_K.$$

Putting the costs together it yields

$$\begin{aligned}\left(4*(27*l*m_K*N)+5N^2+(1-2^{1-K})*4*27*l+5\right)N^2+\\ 3Knsv^2*N+K2^Knsv^3+2*nsv^2*N-4*nsv^2*m_K=\\ =\left((1-2^{1-K})*4*27*l+10\right)N^2+\\ \left(4*(27*l*m_K)+(3K+2)nsv^2\right)N-4*nsv^2*m_K+K2^Knsv^3 \text{ flops.}\end{aligned}$$

As we see from the expression above, the number of flops required to compute the HSS preconditioner is, under the assumptions made above, of order N^2. □

Remark 15

The cost breakdown mentioned above tells us that assuming the upper bound on the number of iterations required for convergence in the computation of the nsv-largest singular triplets, the computation of the HSS preconditioner will be obviously cheaper than the order stated by the proposition. Not only did we assume a constant number of nonzero elements independently of the levels, but also the cost of vector operations listed above concerning the computation of the sparse SVDs will be lower than claimed. In particular, the permutation we perform on the near-field matrix concentrates the nonzero entries near the diagonal and as emphasized before, the HSS preconditioner is constructed from the near-diagonal part of the latter matrix. Consequently, in each block-row the nonzero entries can (and should) be localized more accurately around the diagonal. This improvement would lead to a much lower cost in vector operations and thus reduce the total cost of setting up the preconditioner. Figure 6.6 suggests that even for a small example, the localization of the nonzero part of the permuted near-field will reduce significantly the number of flops required for the vector operations of the cost breakdown for the sparse SVDs. Unfortunately, this fact has not yet been exploited in the implementation of the SHSS package. We only profit from the first point addressed above, namely, the fact that in the computation of the sparse SVDs the matrices (block-rows) are only referenced via matrix-vector products.

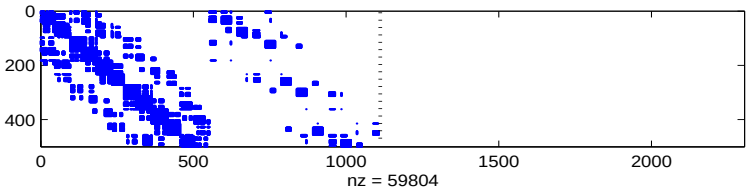


Figure 6.6: *Localization of the nonzero entries in a block-row for a PEC sphere with N=2304.*

Remark 16

Observe further that $H_{k;i}$ is closely related to $H_{k+1;2i-1}$ and $H_{k+1;2i}$. Using the algorithm for updating the SVD proposed in [47], it could be possible to reduce the cost stated above significantly, since only the SVDs at level K would need to be computed.

Remark 17

Due to the strong dependence of the resulting impedance matrix on the geometry of the scatterer, it is very difficult to obtain a criterion that allows us to select the parameter nsv analytically. Consequently, we use a simple criterion based on the storage cost of potential HSS preconditioners. The selection of the number of levels K is also based on the number of levels of the MLFMM as well as on the storage requirements. The SHSS package provides us, once we know the number of nonzero entries in the near-field matrix, with the cost in MByte a suitable set of values for nsv would lead to. It is important to compute a HSS preconditioner which leads to a storage cost similar to that of the near-field matrix. Keeping the aforementioned information in mind, it is usually not difficult to select nsv accordingly.
Moreover, note the role the parameter nsv plays in the estimation above. Although it only appears explicitly in the term multiplying N and lower order terms, large values of nsv will lead to a significant number of flops.

6.2.6 The Computation of the A_{HSS}^{near} Revisited

We begin this section with a remark resulting from Proposition 9

Remark 18
The reordering of the near-field matrix described in Proposition 9 leads to the following partition in block-rows:

$$A_{\mathcal{P}}^{near} = \begin{pmatrix} \mathbf{a}_{B_1^{\lambda_{max}}} \\ \vdots \\ \mathbf{a}_{B_{M_{\lambda_{max}}}^{\lambda_{max}}} \end{pmatrix}, \tag{6.15}$$

and consequently, the location in the impedance matrix of the far-field relative to a group i can be easily identified on the $i-th$ block-row of the reordered near-field matrix.

The remark above suggests that the partitioning of the matrix to compute the HSS representation and in particular, the resulting binary tree for the indices of the partitioned matrix, could be computed directly from the information concerning the distribution of the basis functions the MLFMM provides.
For this purpose, we create a new binary tree describing the partition of the MLFMM groups' indices, $i = 1, \ldots, M_{\lambda_{max}}$, which, in turn, will determine the previously mentioned merge-tree [20]. As we did in a previous section, let us denote the partition sizes at the finest level of the tree as $\{n_i\}_{i=1}^{2^K}$ and similarly $\mu_i = \sum_{j=1}^{i} n_j$ will represent the collected partition sizes (see Figure 6.7). Therefore, n_i will represent the number of MLFMM groups entering subset i. The μ_i accumulates, so to speak, the indices of the MLFMM groups allowing for an easy description of the process in terms of a binary tree similar to the one depicted in Figure 6.7.

6.2.7 The HSS Solver and its Properties

As we briefly mentioned before, the algorithm in [20] for solving a linear system with a HSS matrix as coefficient matrix computes an implicit ULV^H decomposition, where U and V are unitary matrices and L is a lower triangular matrix. The core of the algorithm is to carry out a compression of the block-rows followed by a recursive deflation of the linear system. It is important to note that for the HSS preconditioner we will try to choose the number of singular values as low as possible since this has an immediate repercussion in the computational complexity required for setting up and storing the HSS preconditioner. In particular, for large examples, the number of

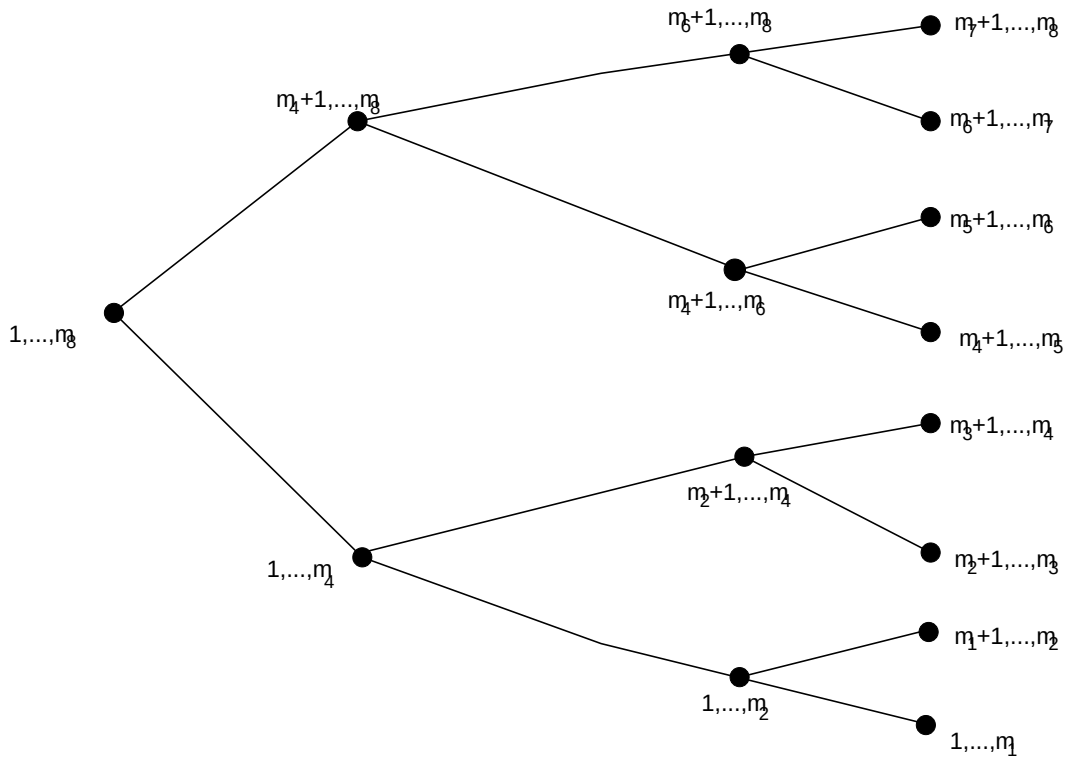


Figure 6.7: *Binary tree with 4 levels, where the zero level represents whole set of MLFMM groups, depicting the distribution of the MLFMM groups.*

rows of the block-rows at the finest level will be, in general, larger than the number of singular values to compute and thus the compression process will be possible. Here, again, K is the number of levels of the HSS representation. Moreover, the number of singular values is fixed to nsv for every block-row in every level. Recall that the linear system of equations to be solved is denoted as $Ax = b$.
The algorithm at level K comprises the following steps.
Step 1:
Under the assumption that $m_{K;i} = m_K$ for $i = 1, \ldots, 2^K$, there exists an unitary transformation $Q_{K;i}$ such that

$$\hat{U}_{K;i} = Q_{K;i}^H U_{K;i} = \begin{pmatrix} 0 \\ \hat{U}_{K;i}^{(2)} \end{pmatrix}. \tag{6.16}$$

Note that the compression has zeroed the first $m_K - nsv$ rows and only nsv nonzero rows remain. For this purpose we can use a QL factorization of $U_{K;i}$.
Step 2 (Apply the $Q_{K;i}$ to the i-th block of the right hand side. The latter has

been previously partitioned according to m_K)

$$Q_{K;i}^H b_{K;i} = \begin{pmatrix} \beta_{K;i} \\ \gamma_{K;i} \end{pmatrix}. \tag{6.17}$$

Step 3 (Lower triangularization of $Q_{K;i}^H D_i$):
There exists further an unitary transformation $S_{K;i}$ such that

$$\hat{D}_{K;i} = (Q_{K;i}^H D_i) S_{K;i}^H = \begin{pmatrix} \hat{D}_{i;1,1} & 0 \\ \hat{D}_{i;2,1} & \hat{D}_{i;2,2} \end{pmatrix}. \tag{6.18}$$

Again, note that the zero block is a $(m_K - nsv) \times nsv$ submatrix. This step is obtained by means of a LQ decomposition of $Q_{K;i}^H D_i$.
Step 4
At level K the block-columns and block-rows coincide due to the symmetry. This leads to $V_{K;i} = \bar{U}_{K;i}$. Therefore, only the sequence of U matrices need to be stored in the HSS representation of the near-field matrix. However, in the following levels up to the first one we need a new sequence of matrices which we will denote Vs, where the s stands for solver. These matrices Vs are only part of the solver and do not constitute part of the HSS representation.
if level=K then

$$\widehat{Vs}_{K;i} = (S_{K;i}\bar{U}_{K;i}) = \begin{pmatrix} \widehat{Vs}_{K;i}^{(1)} \\ \widehat{Vs}_{K;i}^{(2)} \end{pmatrix}$$

else (level=k < K)

$$\widehat{Vs}_{k;i} = (S_{k;i}Vs_{k;i}) = \begin{pmatrix} \widehat{Vs}_{k;i}^{(1)} \\ \widehat{Vs}_{k;i}^{(2)} \end{pmatrix}$$

end if
Step 5:
Since the block-column has been multiplied from the right by $S_{K;i}$, we need to change the unknowns accordingly, $S_{K;i}x_{K;i}$. Clearly, $x_{K;i}$ represents the conforming number of rows in the vector of unknowns, that is, so many as the number of rows in the i-th block-row.

$$S_{K;i}x_{K;i} = \begin{pmatrix} z_{K;i} \\ \hat{x}_{K;i} \end{pmatrix}. \tag{6.19}$$

Note that after carrying out the steps above we end up with a transformed block-row i where the first $m_K - nsv$ equations read as follows

$$\hat{D}_{i;1,1} z_{K;i} = \beta_{K;i}. \tag{6.20}$$

Step 6:
This linear system with $\hat{D}_{i;1,1}$ as coefficient matrix can be solved for $z_{K;i}$ directly by forward substitution.
Step 7:
Once we have the solution $z_{K;i}$, the first $m_K - nsv$ rows can be removed and similarly, the first $m_K - nsv$ columns can be subtracted from the modified right hand side. As a result, performing the previous steps on every block-row at level K the linear system shrinks in $2^K * (m_K - nsv)$ rows and $2^K * (m_K - nsv)$ columns at the expense of a further modification of the right hand side.
The resulting linear system has as coefficient matrix again a HSS matrix $\hat{A}$ represented by $\{\hat{D}_{i;2,2}\}_{i=1}^{2^K}$, $\{\hat{U}_{K;i}\}_{i=1}^{2^K}$, $\{\widehat{Vs}_{K;i}\}_{i=1}^{2^K}$, together with the sequences R and B of the HSS representation of the matrix A.

Remark 19
After these 7 steps have been performed on each block-row, we have seen that in each block-row i nsv equations are left. Clearly, there is no room for further compression here. The way around is to merge the remains of the system according to the following lemma in order to continue with the deflation of the system until we can solve directly.

Lemma 20

Let us assume that the HSS representation of a matrix A generates a matrix partitioned in $2^K \times 2^K$ blocks. Under this assumption, it is possible to create a $2^{K-1} \times 2^{K-1}$ partitioning matrix from the previous one as follows.

$$D_{K-1;i} = \begin{pmatrix} D_{K;2i-1} & U_{K;2i-1}B_{K;2i-1,2i}Vs_{K;2i}^H \\ U_{K;2i}B_{K;2i,2i-1}Vs_{K;2i-1}^H & D_{K;2i} \end{pmatrix}, \tag{6.21}$$

For $K > 1$ it follows

$$U_{K-1;i} = \begin{pmatrix} U_{K;2i-1}R_{K;2i-1} \\ U_{K;2i}R_{K;2i} \end{pmatrix}, \tag{6.22}$$

$$Vs_{K-1;i} = \begin{pmatrix} Vs_{K;2i-1}\bar{R}_{K;2i-1} \\ Vs_{K;2i}\bar{R}_{K;2i} \end{pmatrix}, \tag{6.23}$$

where $i = 1, \ldots, 2^{K-1}$. The bar on the R of (6.23) stands for the complex conjugate of R.
For the proof of the merging lemma see [65].

After the merging process we will have 2^{K-1} block-rows with $2 * nsv$ rows, therefore we can proceed again to Step 1 and solve recursively for x. This process continues until we reach the first level. For this case only the diagonal matrices are merged to a matrix of dimension $2 * nsv$ and the resulting linear system can be solved directly. Once we have the solution vector z, we can recover x by using the complex transpose of the $\{\{S_{k;i}\}_{i=1}^{2^k}\}_{k=1}^{K}$ transformations as follows

$$x_{k;i} = S_{k;i}^H \begin{pmatrix} z_{k;i} \\ \hat{x}_{k;i} \end{pmatrix}. \tag{6.24}$$

Remark 21
In [20] we find that the update of the right hand side in Step 7 can be performed very efficiently if one notes that the system of equations has been previously transformed as follows

$$(diag(q_{K;i}^H)\, A\, diag(S_{K;i}^H))\, diag(S_{K;i}x) = diag(q_{K;i}^H)b. \tag{6.25}$$

Therefore, the subtraction to update the right hand side can be written as a matrix-vector product with the matrix $(diag(q_{K;i}^H)\, A\, diag(S_{K;i}^H))$ and the vector $\begin{pmatrix} z_{K;i} \\ 0 \end{pmatrix}$. The latter matrix has, in turn, the following HSS representation $\{\hat{D}_{i;2,2}\}_{i=1}^{2^K}$, $\{\hat{U}_{K;i}\}_{i=1}^{2^K}$, $\{\widehat{Vs}_{i;2,2}\}_{i=1}^{2^K}$, together with the sequences R and B of the HSS representation of A. We will not present the fast HSS matrix-vector operation, although it represents an important pillar of the fast solver. The matrix-vector product algorithm represents an algebraic description of the already presented MLFMM and therefore we prefer to point to the already cited references [19, 20].

Remark 22
For the implementation of the solver that we will comment on in the Appendix, we see from the steps above that the bookkeeping work is significant as we need to know the level in which the different parts of the solution vector $\hat{x}$ were obtained so as to multiply with the right orthogonal transformation S.

Now we turn our attention to the operational complexity involved in the solver. Let m_K be again the number of rows of every block-row of the near-field matrix at level K.

Theorem 23
Let us assume that on every level and for every block-row in the computation of the SVDs in the HSS representation of the near-field matrix a fixed number of singular values nsv is computed and that for the same purpose a sufficient number of levels, $m_K = 2 * nsv$, is considered. Then the number of flops of the algorithm above, namely, the fast HSS solver, is of order $\mathcal{O}(N * nsv^2)$.

Proof

To estimate the complexity of the solver we proceed as in the last section, namely, considering each step and the pertaining cost associated to it. We will compute the number of flops for the real case since for the complex case in general only the multiplication with a constant is required as it is the case in the QL decomposition in Step 1, where the factor is four. Again, K is the number of levels in the HSS representation. We will further assume that the number of rows of each block-row $m_{K;i}$ is independent of i. We will denote this fixed size at level K as m_K.
In the first step we have to compute the QL factorization of 2^K $m_K \times nsv$ matrices. Following [36] this step requires $2 * nsv^2(m_K - \frac{nsv}{3})2^K$ flops.
In the second step the right hand side is multiplied by the resulting orthogonal matrices $Q_{k;i}$. This requires $2 * nsv * (2m_K - nsv)2^K$ flops. Next, in Step 3 the same operation is performed on the diagonal prior to the LQ factorization, therefore we need $2 * nsv * m_K(2m_K - nsv)2^K$ flops.
Next we compute the LQ factorization of the resulting matrices. This step requires $\frac{4}{3}m_K^3 2^K$ flops. The application of $S_{K;i}$ to $\bar{U}_{K;i}$ demands $2 * nsv * m_K^2 2^K$ flops.
In Step 6 the partial forward-substitution costs $(m_K - nsv)^2 2^K$ flops. The subtraction of the computed unknowns requires, provided we use Remark 21 , $2^K 2m_K(m_K + 2 * nsv) + 8(K-1)nsv^2(2^K - 2)$ flops.
To recover $x_{K;i}$ from $z_{K;i}$ we need $2m_K^2 2^K$ flops since we need only to multiply the orthogonal transformation $S_{K;i}$ with a vector of size m_K where the $z_{K;i}$ occupy the first $m_K - nsv$ rows.
After the steps above we proceed to the merging process which will require $8 * nsv^3 * 2^{K-1}$ flops. Likewise, merging the $\hat{U}$ and $\widehat{Vs}$ costs $8 * nsv^3 * 2^{K-1}$ flops.
The number of flops required at level K is therefore

$$2^K \left[\tfrac{4}{3}m_K^3 + 4m_K^2 * nsv - 2(nsv)^2 m_K - \tfrac{2}{3}(nsv)^3 + 2 * nsv * m_K^2 + m_K^2 + (nsv)^2\right] +$$
$$2^K \left[-2m_K * nsv + 2m_K^2 + 4m_K * nsv + 8K(nsv)^2(2^K - 2) - 8(nsv)^2(2^K - 2)\right] +$$
$$2^K \left[2m_K^2 + 8(nsv)^3\right].$$

The expression above can be simplified so as to obtain an upper bound. It yields

$$2^K \left[\frac{4}{3}m_K^3 + 6m_K^2 * nsv + 5m_K^2 + 4m_K * nsv + 8(nsv)^3 + (8K+1)(nsv)^2\right].$$

Under the assumption that $m_K = 2 * nsv$ which only requires the proper number of levels in the HSS representation along with $N = 2^K m_K$ we obtain

$$26N(nsv)^2 + (2^{K+3})(nsv)^3 + (6 + 2^K 8(K+1))(nsv)^2. \tag{6.26}$$

The expression above is the cost in flops of the solver at level K. For the levels k above K, i.e. $k < K$, the cost is only related to nsv. We noted in Step 5 concerning

the HSS solver that in each block-row we solve for $m_K - nsv$ and keep nsv equations. For the solver only the size of the matrices in the sequences of matrices of the HSS representation is important. Therefore, we note that after merging, the diagonals become $(2 * nsv \times 2 * nsv)$ matrices and the rest of the matrices in the new HSS representation (see Lemma 20) have size $nsv \times nsv$. Consequently, the order of operational complexity under the assumptions above will not change since only terms in nsv will enter expression (6.26). □

Remark 24
The assumption made above concerning the number of levels in the HSS representation, namely $m_K = 2 * nsv$, is clearly beneficial for the solver but will be detrimental for the cost of setting up the HSS preconditioner since more levels in the HSS representation require larger sequences of R and B matrices. The bottom line of the theorem above is that a trade-off should be found between the cost of setting up the preconditioner and the performance of the solver.

Remark 25
The last result is obviously a crude approach to the measuring of the solver efficiency [36] but it conveys valuable information that has a direct influence on the way we have implemented it. We will comment on this point in the appendix devoted to the SHSS package.

6.2.8 The Preconditioned GMRES

The following algorithm we roughly describe here is a preconditioned GMRES in which we solve exactly a linear system with A_{HSS}^{near} and the right hand side q_l by means of the HSS solver before we compute the matrix-vector product with the impedance matrix. The permutation matrix $\mathcal{P}$ in the permutations below is the one of Section 6.2.

A_{HSS}^{near} preconditioned GMRES

Permute w:

$w = r_0$

$w = \mathcal{P}w$

$h_{10} = \| r_0 \|_2$

$l = 0$

while $h_{l+1,l} > 0$

$q_{l+1} = w/h_{l+1,l}$

$l = l + 1$

Solve (HSS solver) : $A_{HSS}^{near} v_l = q_l$

Permute $\mathbf{v_l}$: $v_l = \mathcal{P}^T v_l$

$w = Av_l$

Permute w : $w = \mathcal{P}w$

for $j = 1 : l$

$h_{jl} = q_j^H w$

$w = w - h_{jl} q_j$

end

$h_{l+1,l} = \| w \|_2$

Compute z_l such that $\| H_{l+1,l}, z_l - h_{10} e_1 \|_2$ minimal

$x_l = x_0 + V_l z_l$

if $\varphi_l = \| b - Ax_l \|_2 < tol$ then STOP

end

6.3 Numerical Experiments

6.3.1 Initial Results

In this section we present initial results for the HSS preconditioner applied to the examples we have been referring to in previous sections. The tolerance for the relative residual norm will be set, unless otherwise stated, to 1e-4 for all the tests in the sequel. Likewise, the initial guess for all the numerical results that follow is the zero vector. The frequency of the incoming plane wave as well as its direction of incidence, the latter denoted by (φ_{inc}, θ_{inc}), for the initial results concerning the idealized satellite and the rectangular waveguide are given in the next subsection. The illuminating wave for all the numerical examples that follow is polarized in the θ direction, i.e., $E_{\varphi}^{incident}$=0. These initial results are provided only as motivation for Section 6.3.1.

The following results were obtained in Matlab by performing a recursive partition of the matrix such that in every level of the merge-tree the set of matrix indices is bisected. The numerical rank of the block-rows is computed to a certain tolerance in order to compress the far field as explained in [13]. For the idealized satellite, the following figures seem to confirm our conjecture. These examples have a very large near-field component and therefore SPAI and ILU also perform very well. However, the HSS preconditioner is fairly expensive in terms of storage. This point will be addressed in the next section.

The tolerances in the first column of the table below represent the truncation threshold for the computation of the SVD in the HSS representation of the near-field matrix and at the same time the drop tolerance for the ILU. We should point out that the comparisons below with the ILU are only intended as motivation for the HSS preconditioner. The problems and virtues concerning the ILU in the electromagnetic context and specially in 3D scattering for the EFIE are well known. For a thorough performance analysis see [16, 79, 63] and the references therein.

Concerning the figures related to storage, it is essential to note that the complex symmetry of the near-field matrix allows us to store only the U in the ILU decomposition since $L = U^T$. Likewise, for the HSS representation only the U, B and R matrices need to be stored. Consequently, the third column reflects the cost of storing this economic HSS representation and the fifth column gives the cost of storing only the U in the ILU decomposition.

Satellite	GMRES_HSS		GMRES ILU	
tol_{SVD}/drop tol.	# mv	MByte HSS rep.	# mv	MByte
1e-2	13	4.8530	28	1.3568
1e-3	12	10.5112	15	6.7220
1e-4	>180	16.2517	12	9.2634

In order to compare our preconditioner with a well established one, we note that solving the idealized satellite example with an inner-outer GMRES that performs as preconditioning some GMRES iterations (in this case 20) with the near-field matrix, i.e. $A^{near}v_l = q_l$, requires 24 matrix-vector products to converge, whereas the SPAI requires 15 iterations. The former preconditioner is also known as FGMRES [77] in the literature.

It is interesting to note how the number of iterations evolves. Even though it seems to decrease with decreasing tolerance, against expectations with tolerance set equal to 1e-4 the number of iterations grows again. Clearly, A^{near}_{HSS} approximates A^{near} with decreasing tolerance in the SVDs. The evolution of the number of iterations reflects a very strong rank-sensibility of the impedance matrix toward the preconditioner but still a somehow more interesting phenomena. As we may understand the preconditioner as a projection of the near-field matrix onto the manifold of HSS matrices, it may have to do with the geometry of this manifold. This phenomena occurs due to the change of a single parameter. In addition, decreasing the tolerance leads to a better approximation of the near-field matrix, as expected, but this bulge in the number of iterations occurs on the half-way point, loosely speaking, to the near-field matrix. The norm of the difference between A^{near}_{HSS} and A^{full} remains almost constant but the preconditioner performs differently depending on the rank structure.

We reach similar conclusions in the rectangular waveguide example:

Rectangular waveguide	GMRES_HSS		GMRES ILU	
tol_{SVD}/drop tol.	# mv	MByte HSS rep.	# mv	MByte
1e-2	41	15.2827	47	9.9805
1e-3	34	36.3393	31	30.7549
1e-4	27	60.8966	27	52.6991

Solving the rectangular waveguide example with the inner-outer GMRES described above requires 206 matrix-vector products to converge, and if the number of inner-iterations is increased to reduce the number of iterations to 60, then the computing time of a compiled code is comparable to the time of computation of Matlab for the case when the drop tolerance is set to 1e-3.

Figure 6.8 justifies graphically the good performance of the preconditioner, since it

succeeds in clustering the eigenvalues around $(1, 0)$. In particular, the picture shows the spectrum of the impedance matrix in the rectangular waveguide example multiplied by the inverse of A_{HSS}^{near} with tolerance in the SVDs set to 1e-2. The selection of this tolerance relies on the fact that we are interested in a cheap preconditioner and that this tolerance already leads to a substantial reduction of the number of iterations with respect to the inner-outer GMRES tested or the SPAI preconditioner which requires 74 iterations to converge. This picture should be compared with Figure 6.4, where the spectrum of $A(A^{near})^{-1}$ for the rectangular waveguide was depicted.
As mentioned before, for the tolerance set to 1e-2 the solution time is comparable to an ILU preconditioner. Figure 6.8 shows that the clustering of eigenvalues is

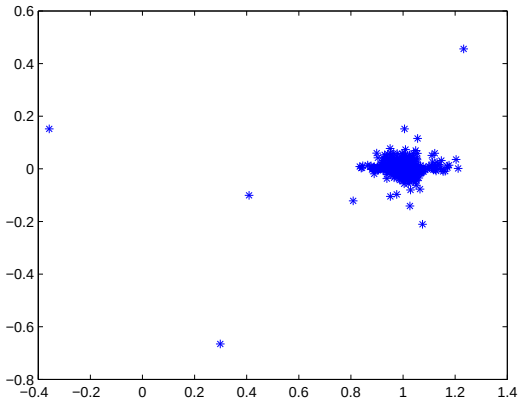


Figure 6.8: *Spectrum of* $A\,(A_{HSS}^{near})^{-1}$ *for the rectangular waveguide.*

much stronger than in the SPAI case and than in the case when we multiplied the impedance matrix from the right with the inverse of the near-field matrix.
In the last section we performed the partition of the near-field matrix following a bisection procedure on the set of indices of this matrix. According to the results showed above, it does indeed precondition the linear system of equations we want to solve but there is still a point that need to be addressed, namely, the fact that the MLFMM algorithm provides us with a better partitioning procedure. So, it seems wiser to compute the merge-tree with respect to the MLFMM groups and thus, divide the indices of the matrix accordingly. In this fashion interferences with the rank structure are at least partially avoided. Proceeding this way simply changes the location where we partition the matrix in block-rows.
The near-field matrix comes from a three dimensional data structure, the associated oct-tree, and therefore we cannot expect to capture it with a two dimensional

merge-tree. In the aforementioned partition we cut the near-field matrix at least at the end of a collection of groups and not in the middle of a MLFMM group.
In order to fully exploit the complexity estimations provided in [20] we confine ourselves to the case where the number of singular values is set to a constant nsv for every block-row and every level. In this fashion we obtain a quite compressed far field for large blocks, whereas for the lowest levels in the merge-tree we obtain a rather generous approximation, i.e., low compression.
The drawback to this approach could be that for some geometries the distribution of the number of basis functions per group may be very unbalanced. Fortunately for large examples we want to compute as few singular values as possible and the irregular distribution has a minor influence, as our experience so far has shown.
In the next section we will intensively compare the HSS preconditioner based on the approach proposed above with the previously mentioned inner-outer GMRES. The reasons for confining ourselves to these two are initially the fact that in the literature preconditioning techniques related to this solver are often encountered together with the good reputation the GMRES itself has in the engineering community. See [30, 62] and the references therein. Moreover, as we mentioned before, the work in [16] suggests to let aside the ILU in favor of other preconditioning techniques. For a thorough comparison of Krylov solvers and preconditioning techniques in the electromagnetic context, see for example [16, 11].

6.3.2 The Performance of the HSS Preconditioner Based on the MLFMM Groups

For the examples that follow, we will consider spherical coordinates where the angles φ and θ are defined as depicted in Figure 6.9. Unless stated otherwise, the near-field matrix is stored in complex double precision, i.e., `complex*16` in FORTRAN notation.

The Satellite Example

Figure 6.10 shows the convergence history for some HSS and FGMRES preconditioners. We will denote this form of FGMRES in the sequel by GMRES IO. In the legend not only the number of singular values computed in each block-row of the matrix partition for each HSS preconditioner is depicted but also the number of inner iterations for each GMRES IO. Recall that (φ_{inc}, θ_{inc}) denotes the vector of incidence in polar coordinates of the illuminating plane wave. In this example, (φ_{inc}, θ_{inc})=(0 deg., 180 deg.) and the frequency of the incident plane wave is 400

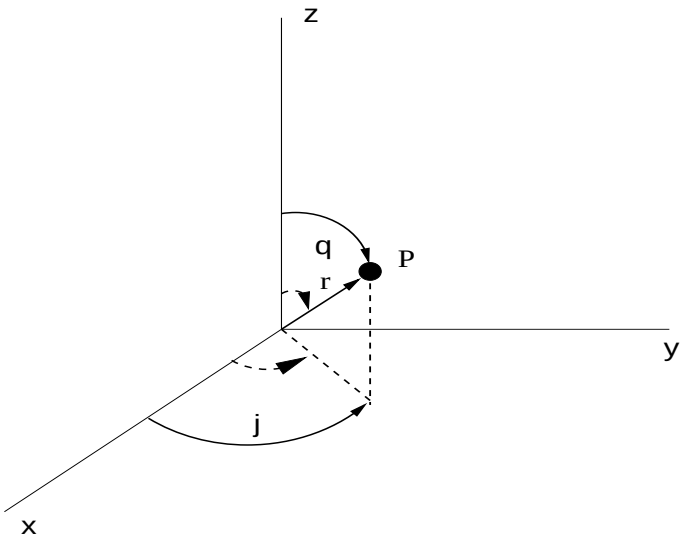


Figure 6.9: *Angles φ and θ in polar coordinates.*

MHz. The near-field entries represent 33.92% of the impedance matrix, requiring 8.14 MByte of storage. For this example there are no major differences in the performance of the methods as the percentage of near-field is very large. The presence of antennas makes the EFIE the only alternative for solving the scattering problem and, in particular, the challenge here is the fact that the MLFMM may get into trouble when the size of the MLFMM groups is too small, which is the case around the antennas [26]. Still, it is remarkable that in comparison with the un-preconditioned case the number of iterations has been reduced more than 90 percent, from 180 to 13 iterations to converge.

These results were obtained in Matlab with the HSS preconditioner. The number of singular values computed in each block-row B is chosen as the minimum of the number of rows in B and the corresponding number of singular values desired. In particular, it means that for the first block-row of the idealized satellite with 38 basis functions belonging to the first MLFMM group no compression at all is performed. The number of levels in the HSS representation of the near-field matrix is three.

It is also noticeable that the interesting phenomena concerning the evolution of the number of iterations observed in Section 6.3 has seemingly disappeared. In the results above we note a sort of barrier in the number of iterations required to achieve convergence so that no matter how many new singular values are additionally computed to the 120, the number of iterations does not improve any longer. Recall that the HSS preconditioner represents a structured approximation to the near-field matrix. Consequently, as nsv increases, we obtain clearly a better approximation to this matrix.

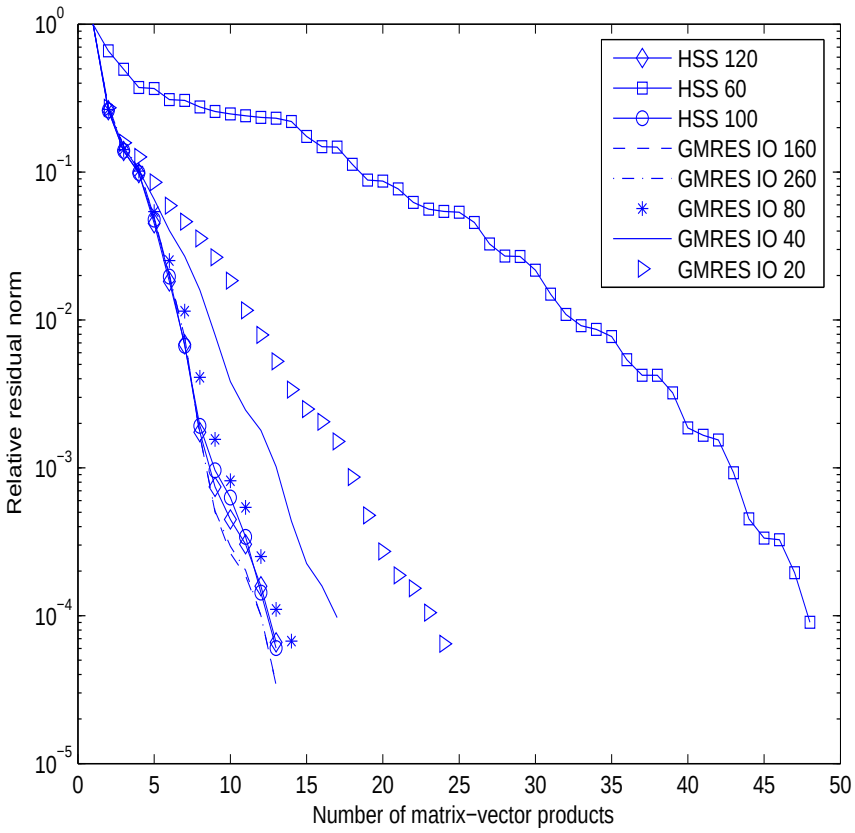


Figure 6.10: *Convergence history for the satellite.*

The examples below were no longer computed in Matlab but by means of the SHSS package we developed for this work. For details concerning the machines where these experiments were performed see the Appendix.

Rectangular Waveguide

Solving the linear system in this example is more demanding in terms of iterations and, at the same time, more interesting as far the behavior of the different preconditioners is concerned. The illuminating electromagnetic wave has a frequency of 9 GHz, impinging on the target with (φ_{inc}, θ_{inc})=(35 deg., 55 deg.). The figures for this example were obtained with the SHSS package which will be briefly presented in the Appendix. This package for Symmetric Hierarchically SemiSeparable (SHSS) matrices takes special advantage of the symmetry when it comes to the computation of the HSS representation of the near-field matrix. Moreover, the storage

figures given below ignore the cost of the diagonal matrices $\{D_i\}_{i=1}^{2^K}$ (K is again the maximum level in the HSS representation) since their cost is already included in the near-field storage and we only point to them when necessary. In particular, and for the sake of comparison, the near-field matrix for this example requires 36.24 MByte storage and the maximum level K is three.

Figure 6.11 shows an apparent sensitivity to resonances. In the inner-outer GMRES iterations a plateau arises in the convergence history as long as the number of inner iterations is relatively small and even for a number of iterations as large as 240 there are still signs of stagnation. In addition, we have to mention that the time required for each iteration in the solution of the linear system of equations with 240 inner iterations in Fortran takes almost as long as an iteration of the HSS preconditioned GMRES in Matlab. The reason is clearly the low sparsity of the near-field matrix.

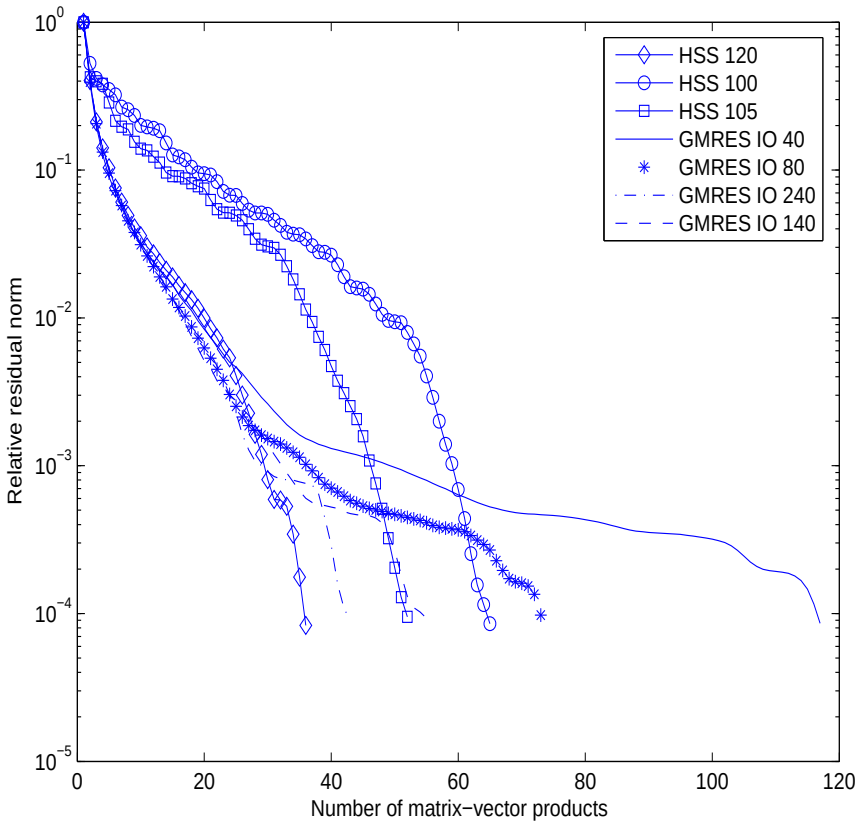


Figure 6.11: *Convergence history for the rectangular waveguide.*

# Sing. Values per block-row	GMRES_HSS			
	#mv	Min. time per iter.	Storage(Mb) HSS mat.	Set up time
100	65	0.3379 sec.	9.0847	31.205 sec.
105	52	0.3529 sec.	9.7404	33.801 sec.
120	35	0.3559 sec.	11.8264	42.441 sec.

Concerning the timings provided in the table above and the tables to come in the sequel, we should mention that unlike the implicit ULV^H decomposition we mentioned in the chapter devoted to the HSS representation and the corresponding fast solver, we can obtain such timings provided we perform some pseudo-explicit decomposition. In other words, we exploit what we have computed in the first solver iteration. It is not an explicit decomposition since we do not provide a lower triangular matrix, neither do we provide the orthogonal matrices used to deflate the original system of equations. It has, however, the speed of an explicit decomposition. Nevertheless, the price for the speed comes in storage, or, more accurately, the peak in storage. It is clear that if the computation of every possible reusable data takes place during the first iteration of GMRES_HSS, we will have more data residing in memory but the timings given above for the GMRES iterations support this approach and also open the door to an out-of-core storage of the data which the authors in [51] successfully implemented for the ILU. The orthogonal matrices which we need to reconstruct the solution vector in the solver are especially expensive to store unless we use elementary reflectors for this purpose. In our initial implementation we have preferred to form the matrices. More details about this will be provided in the Appendix.

PEC Sphere

In this example the illuminating electromagnetic wave has a frequency of 3 GHz, impinging the target with (φ_{inc}, θ_{inc})=(0 deg., 180 deg.). This is the first example that requires more than one level in the MLFMM. In particular, it requires two levels and also the percentage of near-field drops to 5.1835 %. Furthermore, N=2304 and the number of levels we consider for the HSS preconditioner is three. The behavior of the preconditioners for this example is extremely uniform since no matter how large the number of singular values or inner iterations in the GMRES inner-outer is, we end up at the barrier of 34 iterations. That is the best we can obtain from the near-field matrix with or without considering the underlying structure in the manner we do it, namely setting the number of singular values to be computed from the block-rows according to the distribution of the basis functions in the MLFMM

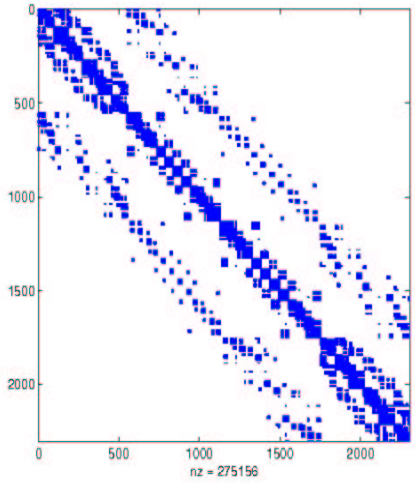


Figure 6.12: *Portrait of the permuted near-field matrix for the PEC sphere.*

groups at the leaves-level for all levels. In addition, the storage for the HSS preconditioners is larger than the storage of the near-field matrix itself which amounts to approximately 5.26 Mbyte. Our experience has shown that the more spherical the object is, the better the approximation of the MLFMM to the far-field component of the impedance matrix is.

However, for other geometries it is well known that the MLFMM tends to overestimate the ranks in the far-field component of the impedance matrix [5]. In this sense the fact that we build the HSS preconditioner from the near-field matrix by constructing another approximation to the impedance matrix from below, coupled with the numerical experiments performed, let us think that the HSS preconditioner may exploit this overestimation of the rank structure for the far-field. In fact, for the PEC sphere no improvement in the convergence history occurs as Figure 6.13 shows. The table below shows what we commented on above, namely, for the PEC sphere the maximum time per iteration decreases but the figures above make the usage of the HSS preconditioner for this target not competitive against this inner-outer GMRES, taking into account the time to set it up and the memory space for storing it.

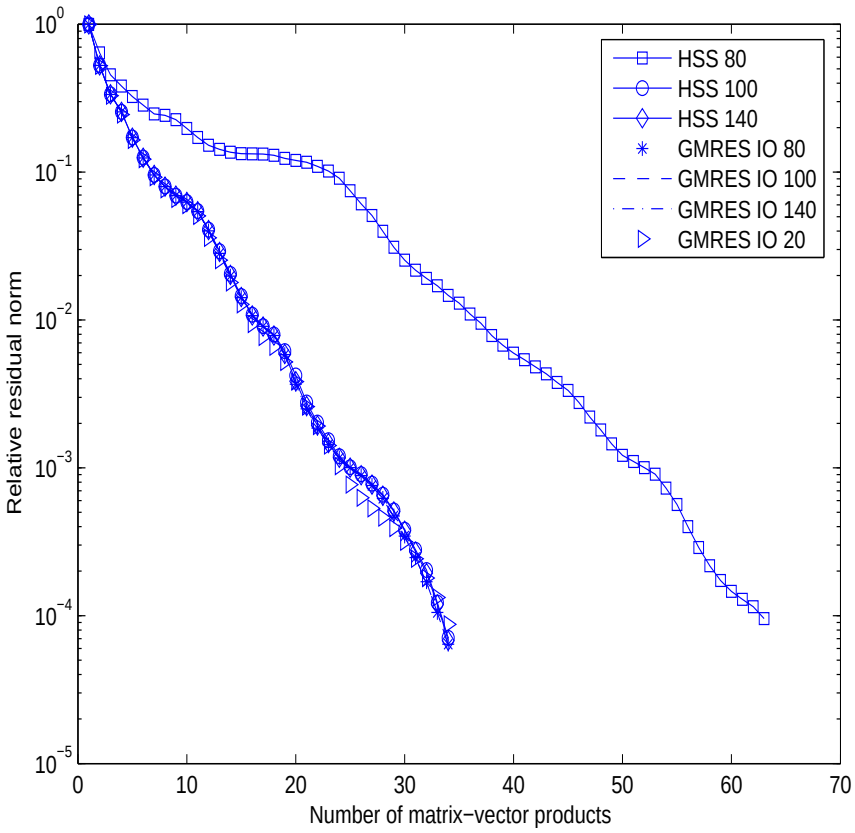


Figure 6.13: *Convergence history for the PEC sphere.*

# Sing. Values per block-row	GMRES_HSS			
	#mv	Max. time per iter.	Storage(Mb) HSS mat.	Set up time
80	63	0.180 sec.	5.4482	9.894 sec.
100	39	0.225 sec.	7.5796	17.022 sec.
120	34	0.281 sec.	10.0283	25.082 sec.
140	34	0.347 sec.	12.7944	33.660 sec.

Cylindrical Waveguide

This example constitutes a further variation of the rectangular waveguide already presented with a more *spherical* geometry. That is in essence the reason for testing it, since it couples the apparent resonant behavior and the, at least for the HSS preconditioner, less favorable uniform geometry (in terms of the curvature).

The number of unknowns N is 5901 and the near-field represents 9.1247% of the impedance matrix. The illuminating plane wave impinges on the target with (φ_{inc}, θ_{inc})=(0 deg., 35 deg.). The storage of the near-field matrix for this example requires approximately 60 Mbyte.
Figure 6.15 compares again the HSS preconditioner to the inner-outer GMRES.

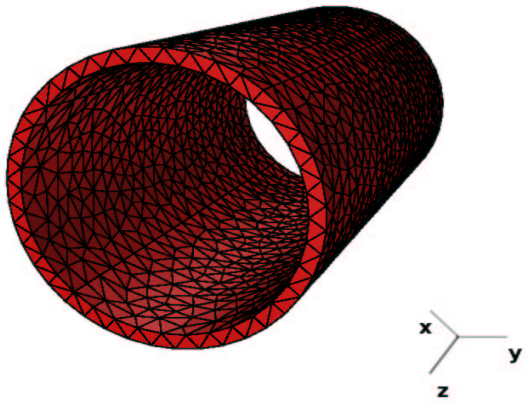


Figure 6.14: *The cylindrical waveguide where the principal axis of the target is parallel to the z-axis.*

Although the number of iterations of the latter improves with more inner iterations, the minimal time per iteration for the last and best case, 120 inner iterations, amounts to 4.7593 seconds. Even the second best, 60 inner iterations, demands at least 2.2357 seconds per iteration. A better solution time has a negative impact on the number of iterations required to solve the linear system to the selected accuracy. The number of HSS levels for this example is set to four, whereas the number of MLFMM levels is two. The storage of the near-field matrix requires 48.48 Mbyte.

# Sing. Values	GMRES_HSS			
per block-row	#mv	Max. time per iter.	Storage(Mb) HSS mat.	Set up time
150	67	1.281 sec.	33.6667	215.853 sec.
200	31	1.955 sec.	53.6564	365.933 sec.
250	31	2.761 sec.	78.0713	514.312 sec.

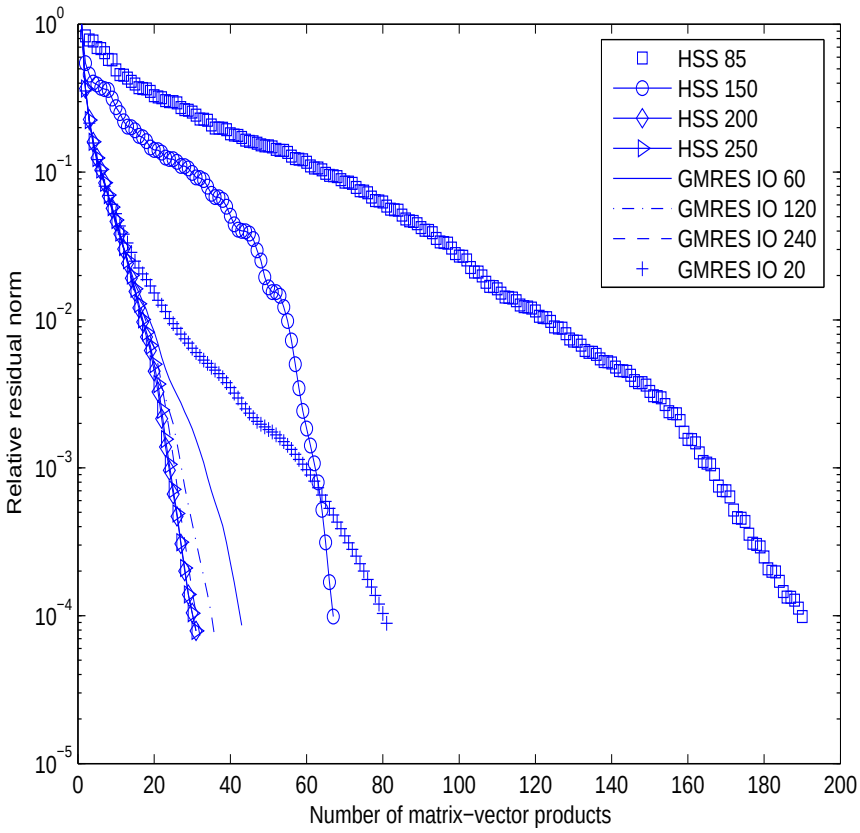


Figure 6.15: *Convergence history for the cylindrical waveguide.*

As in the previous example, we encounter again a barrier for the number of iterations. In particular, no matter how much we invest in the HSS preconditioner at the finest level and thereby for the other levels, the number of iterations does not decrease any more. Interesting to note is the fact that the rank structure of the preconditioner seems to make the difference and allows us to solve in less iterations than the inner-outer GMRES.

Small Airplane

In this example the HSS preconditioner is tested on a sparser near-field matrix than in the examples presented so far. The number of unknowns, though, is not larger than in the previous examples. In particular, the incident wave has the frequency 120 MHz with $(\varphi_{inc}, \theta_{inc})$=(0 deg., 70 deg.) and N=4619. The MLFMM uses four levels and the near-field entries represent 2.8528% of the impedance matrix requiring

11.63 Mbyte storage. The number of levels in the HSS representation of the near-field matrix is three. The permuted near-field matrix again has the structure we

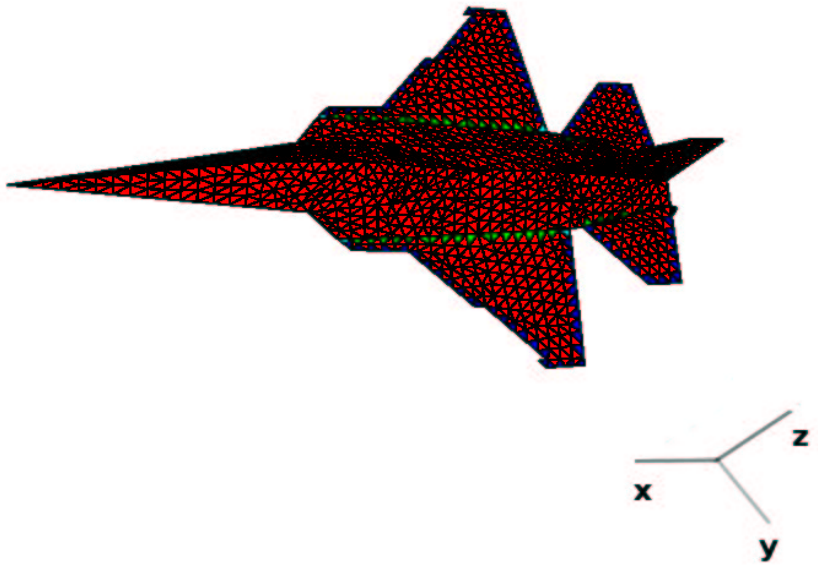


Figure 6.16: *Small airplane with approximate electrical size 3λ.*

mentioned at the beginning of the chapter, displaying the diagonal spine and two near-diagonal components on either side of the diagonal. This is the part of the near-field matrix the HSS preconditioner uses for its construction. The figures in the table below show an extremely good agreement in the number of iterations required by the HSS preconditioner and the inner-outer GMRES. The plane has no cavities which could lead to resonances and the number of iterations depicted in this table let us assume that the frequency of the illuminating wave does not lead to internal resonances.

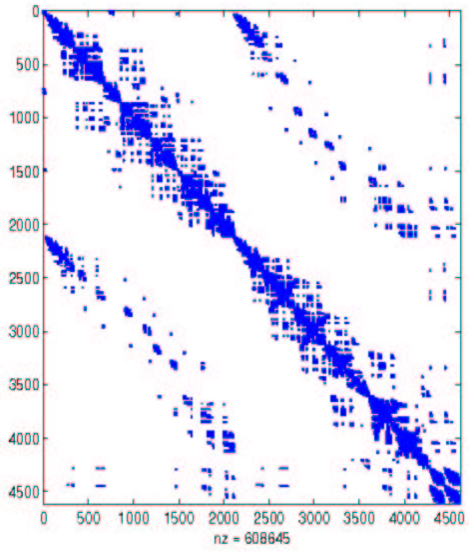


Figure 6.17: *Permuted near-field matrix of the small airplane.*

# Sing. Values per block-row	GMRES_HSS			
	#mv	Min. time per iter.	Storage(Mb) HSS mat.	Set up time
150	32	0.732 sec.	19.6922	71.274 sec.
160	32	0.779 sec.	21.6269	80.939 sec.
180	32	0.878 sec.	25.7343	102.520 sec.

Scatterer with Internal Resonances

This example is representative of a potential problem with surface integral equations which was previously mentioned. For a closed PEC target like Figure 6.19 shows, the EFIE is known to be beset by internal resonances. This phenomenon accounts for the lack of uniqueness in the solution of the EFIE for certain values of ω. These critical values, ω_{crit}, are known as internal resonance frequencies and represent the eigenvalues of the interior Maxwell problem [67].
As a result, the tangential components of one of the incident fields (either the electric or the magnetic field) at the scatterer surface lack enough information to uniquely determine the current density at resonant frequencies [70]. Under these circumstances, a preconditioner is absolutely mandatory to mitigate the effect these internal resonances have on the history of the relative residual norm at values of ω near to ω_{crit}. For these values, the impedance matrix is nearly singular and as a result, the

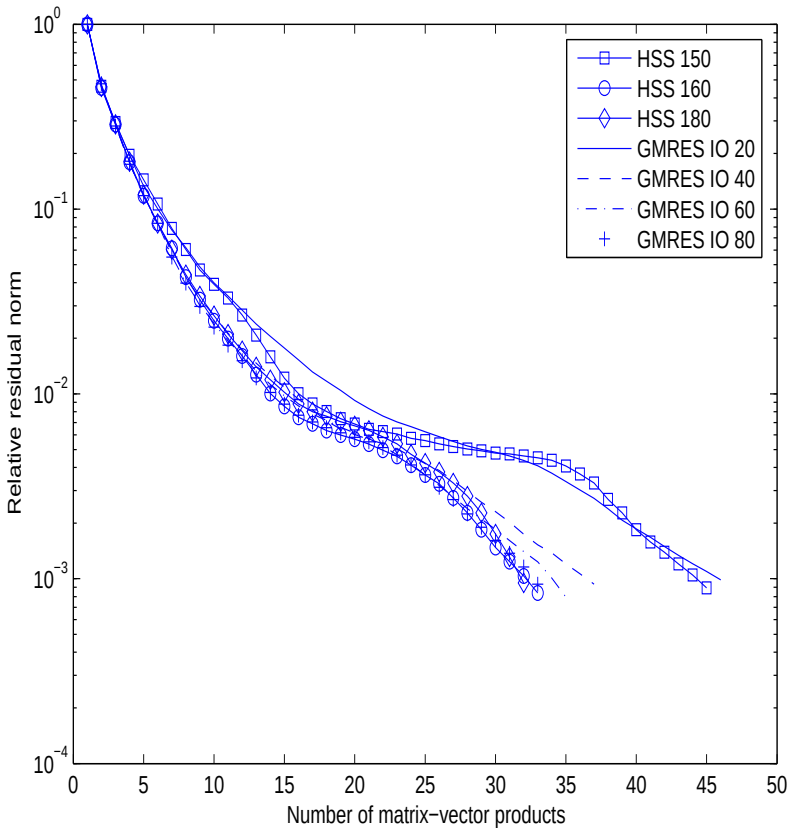


Figure 6.18: *Convergence history for the small airplane with tol=1e-3 in the relative residual norm.*

convergence of iterative methods may be painfully slow or they even fail to converge. The electrical size in wavelengths of this target is approximately 7λ and the number of unknowns $N = 48690$. The near-field entries represents 4.7554% of the impedance matrix requiring approximately 1.3 GByte of storage in complex single precision, i.e., `complex*8` in FORTRAN notation. The incident plane wave has the frequency 10 GHz with (φ_{inc}, θ_{inc})=(0 deg., 35 deg.). The MLFMM uses three levels, whereas the HSS representation of the near-field matrix is computed with five levels.
The following figures illustrate the difficulty in achieving convergence for this example. In particular, we applied the CGS (Conjugate Gradient Squared) [84] with a tolerance in the relative residual norm equal to 1e-3. Figure 6.20 depicts its performance against the evolution of the relative residual norm when preconditioning is applied. This picture shows also the extremely negative impact internal resonances have on the convergence history if no preconditioning is applied. Unlike previous

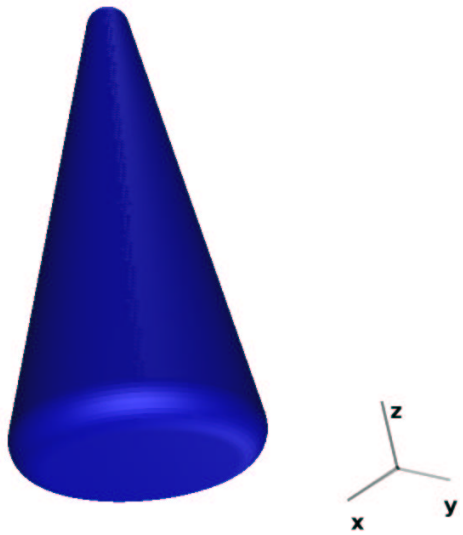


Figure 6.19: *Target with principal axis parallel to the z-axis. The base of the object stands on the xy-plane*

plots we have labeled the x-axis as number of iterations rather than as number of matrix-vector products. This change only affects the CGS that performs two matrix-vector products per iteration thus requiring 1990 matrix-vector products to converge, whereas the HSS preconditioner achieves the same relative residual norm in 159 matrix-vector products. The performance of the COCG [87], a Krylov solver especially suitable for complex symmetric matrices, is even worse since after 7000 matrix-vector products no reduction of the residual norm is recognizable.
Figure 6.21 depicts the convergence history only for the preconditioned system. Again we should note that the time required by the inner-outer GMRES with 5, 10 and 20 inner-iterations amounts to approximately 11, 22 and 42 seconds, respectively. In order to converge this preconditioner with 5, 10 and 20 inner-iterations requires 223, 171 and 159 iterations, respectively.

# Sing. Values	GMRES_HSS			
per block-row	#mv	Max. time per iter.	Storage(Mb) HSS mat.	Set up time
500	159	21.42 sec.	838.9113	9.09 hours
750	159	32.20 sec.	1606.3862	19.81 hours

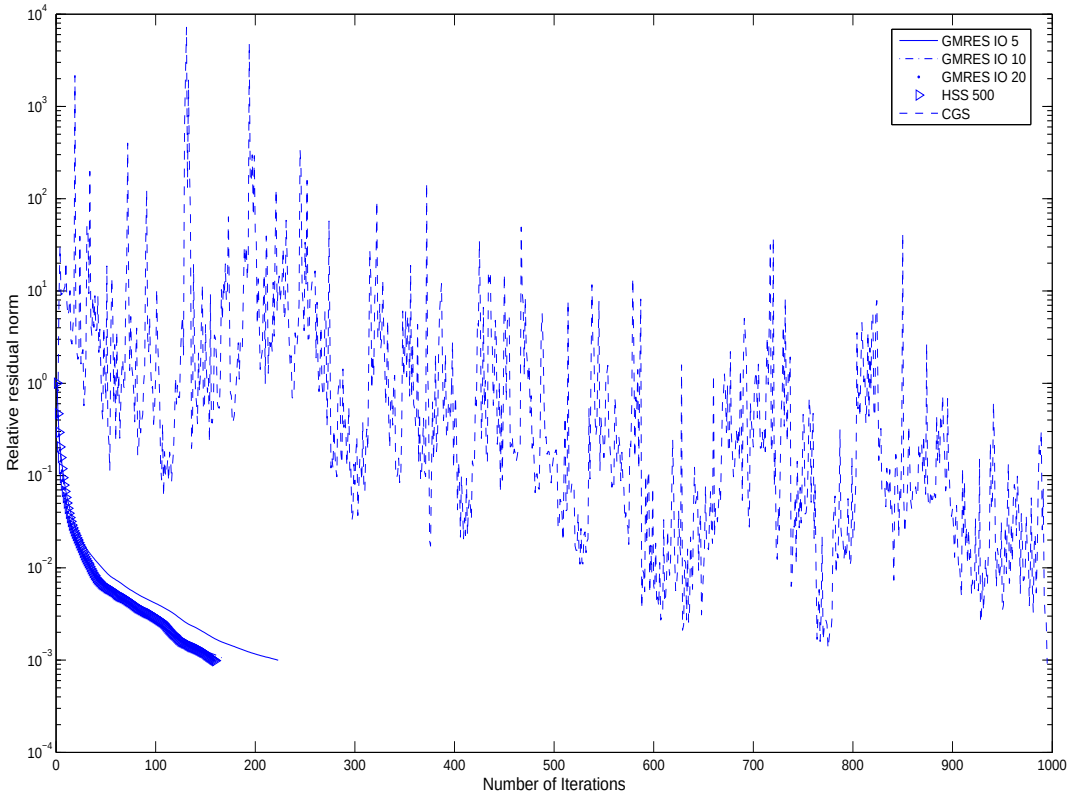


Figure 6.20: *Convergence history for the target with internal resonances with tol=1e-3 in the relative residual norm with 3 levels in the MLFMM.*

We have also computed the MLFMM with four levels instead of three as in the previous situation. Although with four levels MLFMM the size of the smallest group is very near to the constraint mentioned in [26], the numerical results agree with the previous situation. The purpose of this step is to decrease the amount of near-field entries to 1% of the impedance matrix. In particular, the percentage of near-field entries in the system matrix is 1.1097%. Figure 6.22 shows the performance of the HSS preconditioner with nsv=400 and the maximum level in the HSS representation of the near-field matrix equal to five as above. As a result of the reduction of near-field matrix entries, the number of iterations necessary to achieve convergence has logically increased but it is still competitive taking into account the unfavorable features of the example. A further consequence is that now the number of MBytes needed to store the near-field matrix is 501.98 in complex double precision.

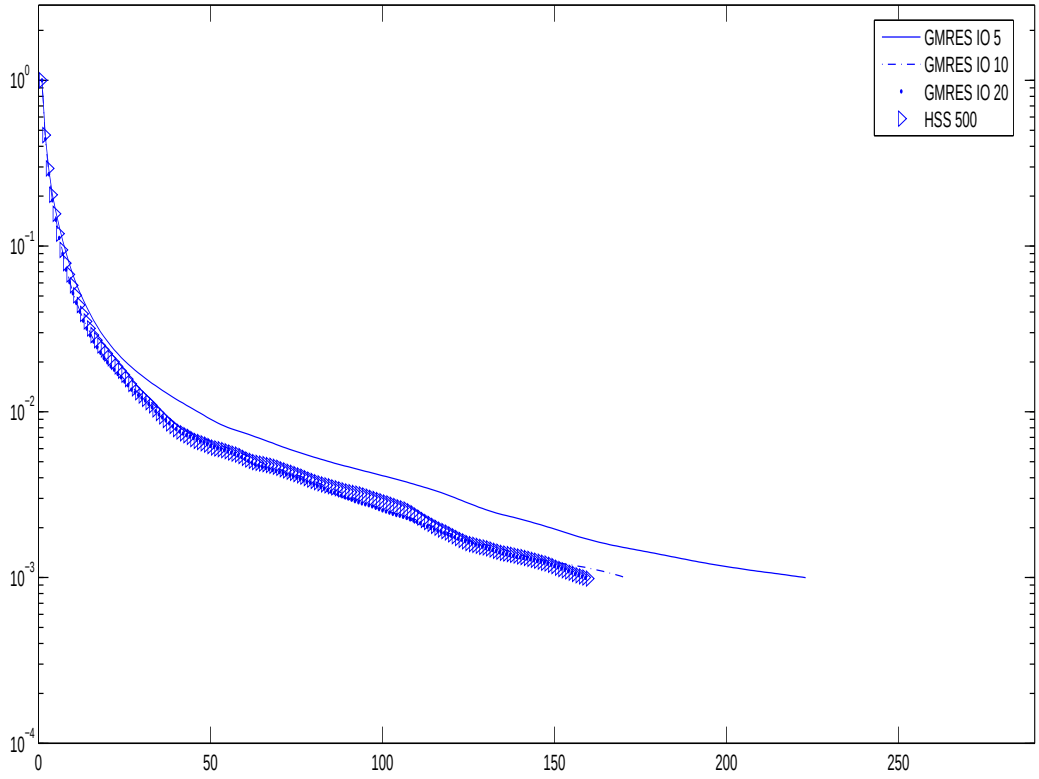


Figure 6.21: *Convergence history for the target with internal resonances with tol=1e-3 in the relative residual norm with 3 levels in the MLFMM.*

# Sing. Values per block-row	GMRES_HSS			
	#mv	Max. time per iter.	Storage(Mb) HSS mat.	Set up time
400	206	16.35 sec.	597.0748	4.26 hours
500	201	27.65 sec.	838.9113	8.87 hours

In order to conclude this section, the induced surface currents on the most interesting targets are depicted in Figures 6.24-6.27 applying the color scale shown in Figure 6.23.
These pictures, with the exception of Figure 6.27, show clearly the oscillating character of the surface current on the target's surface and especially in the case of the airplane they can help in designing low RCS targets [57]. In Figure 6.26 and Figure 6.27 note that the direction of the illuminating wave is easily recognisable.
The following pictures concerning the surface current were obtained by using a tool

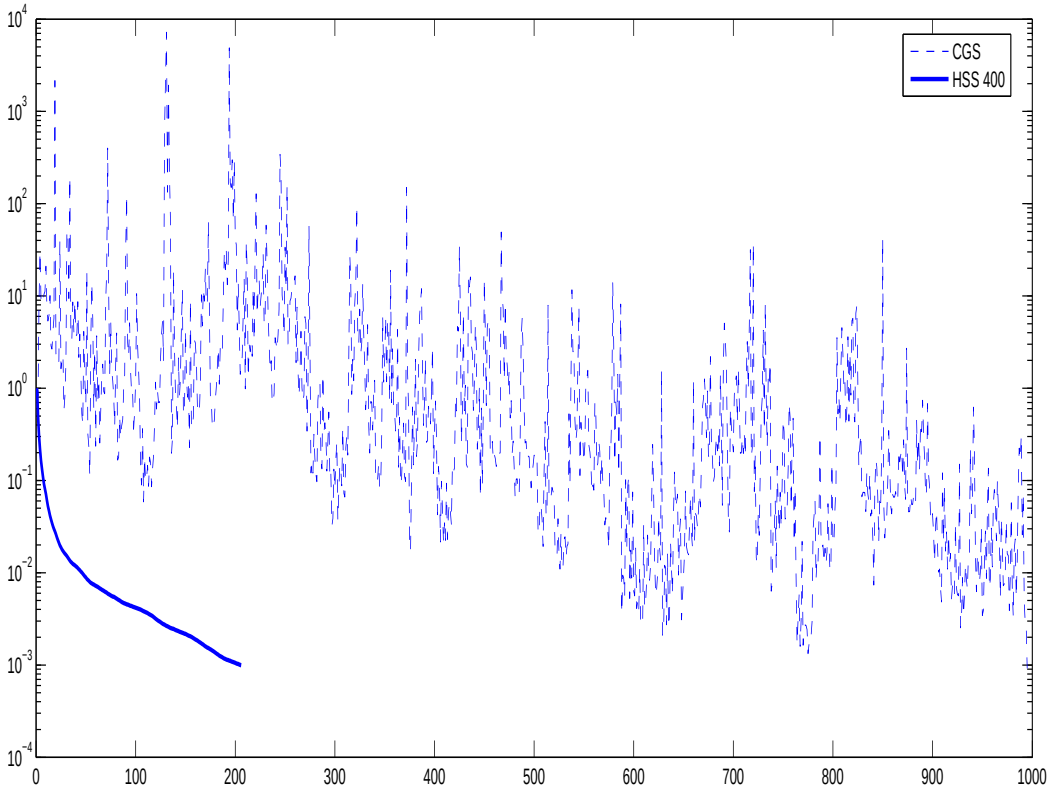


Figure 6.22: *Convergence history for the target with internal resonances with tol=1e-3 in the relative residual norm with 4 levels in the MLFMM.*

Figure 6.23: *Color scale for the Figures 6.24-6.27.*

courtesy of Jan Ritter at EADS Bremen. In particular, the depicted surface currents correspond to the solution of the linear system of equations (2.24) preconditioned by the HSS preconditioner presented in this chapter. To be more specific, for the cylindrical waveguide the HSS preconditioner is computed with nsv=200, whereas for the small airplane nsv=160. Finally, for the last scatterer nsv=400 with four levels MLFMM.

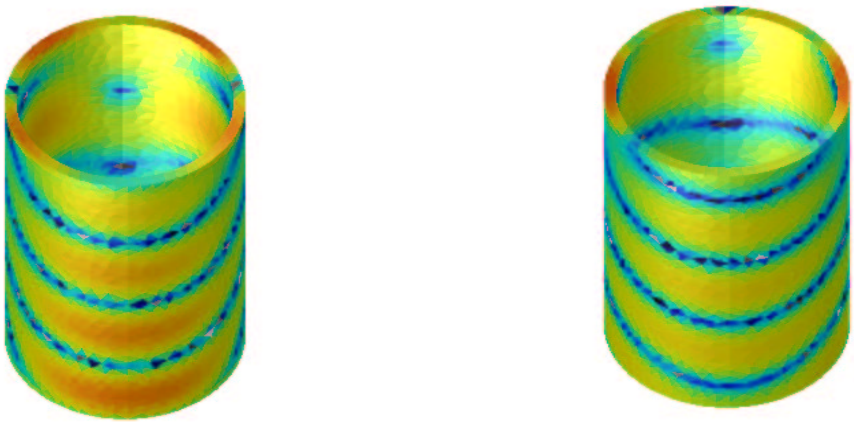


Figure 6.24: *Real part (left hand side) and imaginary part (right hand side) of the induced surface current for the cylindrical waveguide.*

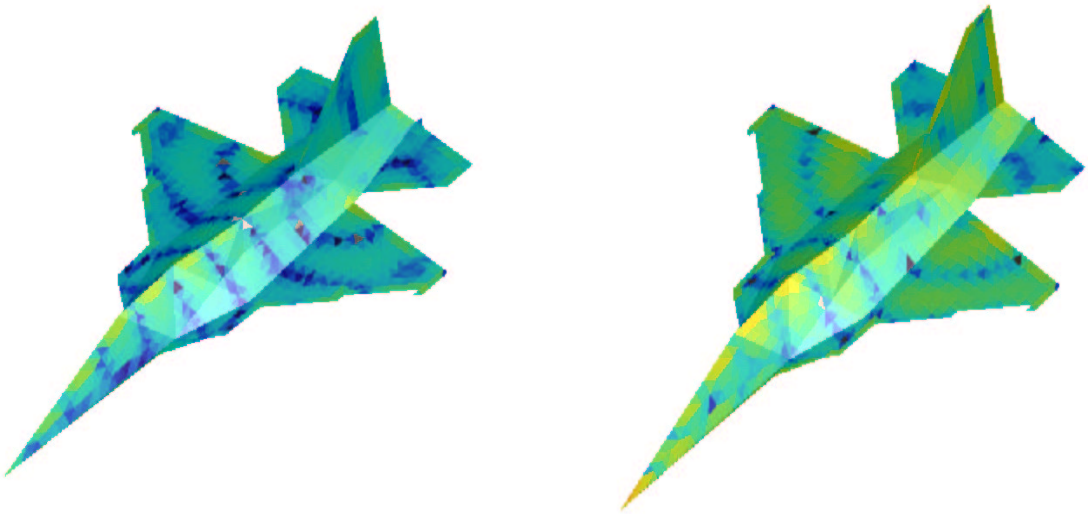


Figure 6.25: *Real part (left hand side) and imaginary part (right hand side) of the induced surface current for the small airplane.*

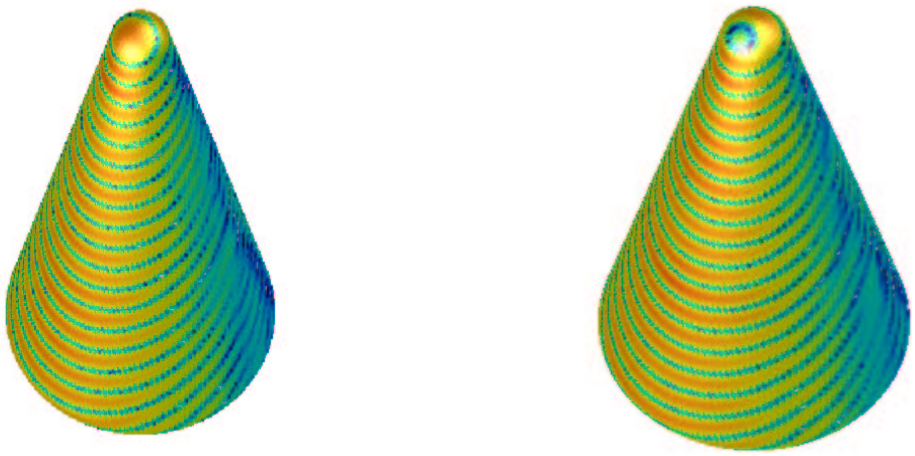


Figure 6.26: *Real part (left hand side) and imaginary part (right hand side) of the induced surface current for the last scatterer.*

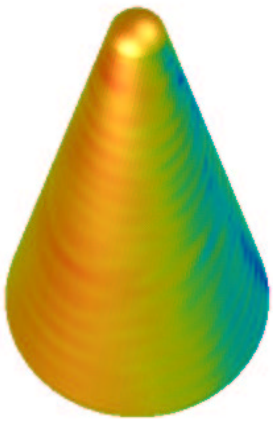


Figure 6.27: *Magnitude of the surface current for the last scatterer.*

6.3.3 Some Remarks Concerning the Numerical Results

Although we did not present any example from real industrial applications, the suite of scatterers presented above represents a wide variety of geometries, sizes and features.
The initial toy examples were, though small in size, not simple to solve regarding the number of iterations required by unpreconditioned solvers and even by SPAI preconditioning to achieve convergence.
An extremely unpleasant feature of the electromagnetic simulation that especially affects the search for a good preconditioner is the impossibility of extrapolating numerical results obtained for small examples to larger ones. As we have mentioned in numerous occasions in this thesis, the decreasing amount of information for constructing a preconditioner with increasing size impedes every kind of prediction concerning the performance of a would-be preconditioner with large scatterers. In particular, in an attempt to mitigate this situation and so as to obtain significant results, the amount of near-field entries in the impedance matrix of the considered scatterers started at approximately 33% for the idealized satellite and shrank to approximately 1% for the last example.
The numerical results have shown that the application of the HSS preconditioner based on a hierarchically semiseparable approximation to the near-field matrix leads to a substantial reduction in the number of iterations required to achieve convergence. Although only for the last example alternative Krylov solvers to the GMRES were mentioned, the situation occurring in the last example concerning the convergence history of them is representative and applies also to the other examples. In particular, we have seen that even for a small percentage of near-field matrix entries the HSS preconditioner was capable of reducing substantially the number of matrix-vector products necessary to achieve convergence against the CGS solver. Especially in the last example where the preconditioner had to cope with a relatively small near-field coupled with internal resonances, the HSS preconditioner was able to achieve convergence within a small number of iterations compared to the number of unknowns.
These numerical results have also shown that a relative small number of singular values may be sufficient to obtain a good preconditioner. This point was confirmed, for example, in the sort of barrier depicted in the evolution of the residual norm for the PEC sphere. An important achievement of the HSS preconditioner is the required storage. From the figures concerning this issue we see that the HSS preconditioner overcomes the most important drawback to the well established ILU preconditioner without forfeiting performance. We are aware that the major drawback to this preconditioner is the amount of time required for the set-up process of

the HSS preconditioner. We have already pointed out the culprit of this shortcoming and this point will constitute the first improvement of the SHSS package in the near future.

In order to emphasize the storage savings yielded by this preconditioner, let us compute the density the HSS preconditioner would have if it were a sparse matrix in the corresponding complex precision for the last numerical example. This computation with 3 levels MLFMM and nsv=500 leads to an approximate density of 4.42%, whereas for 4 levels MLFMM with nsv=400 it yields 1.31%. Hence, in the latter situation, the HSS preconditioner is in terms of storage equivalent to a sparse matrix with about 98% sparsity. These figures should be compared with the percentage of near-field matrix entries in the corresponding impedance matrices as mentioned above.

Chapter 7

Conclusions

The main objective of this work was to develop a preconditioner for the numerical solution of the EFIE in the context of Krylov methods. To this end, we have concentrated on the only part of the impedance matrix explicitly stored and available, namely, the near-field matrix.

The major contribution of this thesis has been the introduction of a new point of view in the construction of a preconditioner for the aforementioned problem. Most preconditioning techniques developed until now resulted in a preconditioner that considered only the magnitude of entries so as to select its density. This was the case of the most successful and extended preconditioners: ILU and SPAI. In the context of the former, the associated fill-in (needed for a reasonable performance) results in a preconditioner much denser than the near-field matrix itself with consequently large storage requirements. This density issue represents a storage bottleneck in electromagnetic applications and results in the search for less storage-intensive preconditioners which do not forfeit performance. The inclusion of information concerning the structure of the matrix is an attempt to alleviate this situation. The incapability of the SPAI preconditioner of clustering a significant part of the spectrum of the preconditioned impedance matrix around $(1, 0)$ is also well known to have a negative impact on the performance of Krylov solvers.

The novelty of this work consists in constructing a structured preconditioner by means of a hierarchically semiseparable matrix approximant to the near-field matrix.

The HSS preconditioner presented in the thesis addresses a new aspect of the impedance matrix, namely, its multilevel low-rank structure. So far, there has been a gap between this matrix and the preconditioners used to transform the coefficient matrix into an approximation of the identity matrix due to the neglected special matrix structure inherent to the impedance matrix. The part of the coefficient matrix used for the preconditioning, the near-field matrix, had never been the object of interest in this respect since it is a sparse matrix capturing only nearby interactions and therefore no low-rank phenomena had been addressed. By means of a suitable permutation we have shown that we can identify within this near-field matrix a near-diagonal part that, at least with respect to the absolute value of its entries, acts as an intermediary between the near and the far-field. From this part of the near-field matrix we were able to impose a HSS structure on it, thus promoting

the near-field matrix to a HSS matrix that approximates the rank structure of the impedance matrix.

We note that in this process an important property of the near-field matrix has been lost, namely the sparsity. Consequently, greater memory efficiency can be achieved by using a factorization that maintains sparsity in the construction of the HSS approximant to the near-field matrix. Nevertheless, as we emphasized throughout this work, the presented preconditioner has a data-sparse representation. Hence, this apparent drawback regarding sparsity has been overcome by means of an implicit sparsity, namely, its HSS representation.

The initial experiments with the HSS preconditioner presented in this thesis has shown that unlike SPAI, the HSS preconditioner achieves a shift of the complete spectrum to the vicinity of $(1, 0)$, thus overcoming the previously mentioned performance problems.

In particular, some of the figures concerning storage have shown that the storage cost associated to the HSS preconditioner is similar to that of storing the near-field matrix. In the intermediate stage of the set-up, however, there exists a peak in memory usage that has to be coped with. The usage of other possible factorizations in the set-up of the HSS preconditioner represents a future extension of the work.

The preconditioning is completed by utilizing the suitable fast solver for the HSS structure. This solver requires a matrix in HSS form and a right hand side thus avoiding a direct reference to the matrix resulting from the HSS preconditioner.

In this thesis we have shown that under some simplifying assumptions the computational complexity of the solver is of order $\mathcal{O}(N)$, with a relatively modest constant. The resulting preconditioner is obviously a full matrix. However, due to the multilevel low-rank structure, it it still inexpensive to store. Therefore, the storage requirements do not constitute a problem as far as our experience has shown.

The numerical results presented have shown the expected sensitivity of the impedance matrix towards the rank structure of the preconditioner. Furthermore, the storage constraint has been, at least in the examples considered, successfully fulfilled without forfeiting the performance. In particular, we have seen that in the last numerical example, the largest considered, the HSS preconditioner is in terms of storage equivalent to a sparse complex matrix with a degree of sparsity very close to that of the near-field matrix.

These numerical results have also shown that the application of the HSS preconditioner based on a hierarchically semiseparable approximation to the near-field matrix leads to a substantial reduction in the number of iterations required to achieve convergence to a given tolerance.

Summing up, the HSS preconditioner presented in the thesis represents a new type of structured preconditioning in the electromagnetic context. It overcomes the storage

problems which beset the popular ILU preconditioner thanks to a multilevel low-rank structure and as far as our experience has shown, it can cope with a very small percentage of near-field without compromising performance. However, in order to be competitive within industrial applications the time required by the set-up of the preconditioner has to be addressed as well as the search of new explicit factorizations that keep the sparsity of the near-field matrix.

Appendix A

The SHSS package

Now we turn our attention to the implementation of the preconditioner and, in particular, to some details concerning the memory usage that have to be addressed and constitute at the same time a future extension of this work.
The routines in the SHSS package implement the HSS matrix structure [20] and some utilities for these kind of matrices as a matrix-vector routine and the previously mentioned HSS solver. We have created so far a version for sparse matrices both for simple and double complex precision, namely `complex*8` and `complex*16` in FORTRAN 90, the language we have used for the implementation. The following routines from LAPACK (Linear Algebra Package) and BLAS (Basic Linear Algebra Subroutines) have been used

{z,c}gemv	Matrix-vector product.
{z,c}gemm	Matrix-matrix product.
{z,c}geqrf	Computation of a QR decomposition.
{z,c}ungqr	Generation of the Q matrix from the QR decomposition.
{z,c}unmqr	Multiplication with the the Q matrix from the QR decomposition.
{z,c}geqlf	Computation of a QL decomposition.
{z,c}ungql	Generation of the Q matrix from the QL decomposition.
{z,c}unmql	Multiplication with the the Q matrix from the QL decomposition.

In particular, we have used the Intel Math Kernel Library implementation of either LAPACK and BLAS. The majority of the numerical results were obtained using an IBM ThinkPad with 1 GByte memory and an 1.73 GHz Intel processor. The operating system is Linux, kernel 2.6.7. Only the last scatterer, the one with N=48690 in Chapter 6, was computed on a node of an IBM-Regatta computer with 32GByte memory and 1.7GHz processing core (Power4 chip). The BLAS implementation used in this case is ESSL (IBM Engineering and Scientific Subroutine Library).

A.1 Computation of the SVDs for the HSS Representation

The fundamental external library used in the package is called PROPACK [60] and is used for the sparse SVDs mentioned in Chapter 6 in the section devoted to the computation of the HSS preconditioner. PROPACK is a collection of Fortran 77 routines for computing the singular value decomposition of large and sparse or structured matrices. The SVD routines are based on the Lanczos bidiagonalization algorithm with partial reorthogonalization (BPRO).
The reason for this choice in lieu of ARPACK was that PROPACK generally outperforms the latter up to a factor of 2-3 on large examples [61] and as we have seen in the chapter devoted to the HSS preconditioner, the size of the block-rows may have a very large dimension though they are in the same extent extremely sparse for large examples. We will come back to this point after we introduce the structure used for the HSS matrices.
As far as the architecture of the SHSS package is concerned, features of F90 such as modules and derived data types have been extensively used. The former allows the modification of different parts of the program independently and enhances readability of the code. The latter has been used to model the principal components of the SHSS package such as the HSS structure.
The flexibility of this architecture has played a fundamental role in the aggregation process to the existing MLFMM code of EADS. The next picture shows the basic pillars comprising the SHSS package. The routine denoted as `solver` comprises in turn two routines that will be mentioned in the sequel. Same situation applies to the `merge` routine.

A.1.1 The HSS Structure

In order to implement the hierarchically semiseparable preconditioner already described we proceed to create a data type called `level`

```
type level
type(R_matrices),pointer::R(:)
type(D_matrices),pointer::D(:)
type(U_matrices),pointer::U(:)
type(B_matrices),pointer::B(:)
integer,pointer::I(:)
integer,pointer::Ib(:)  end type level
```

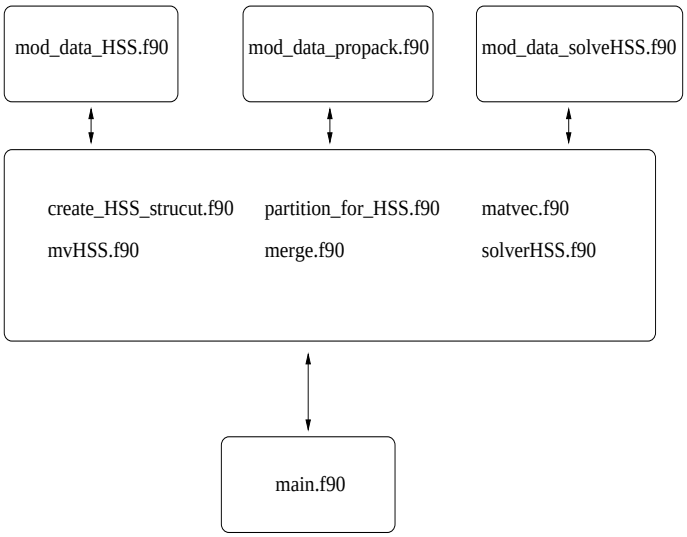


Figure A.1: *SHSS package's structure.*

This field encompasses the matrices already described together with the partition of the matrix indices and the MLFMM groups. In this fashion, the I accounts for the recursive partition of the indices on each level, whereas Ib holds the partition of the MLFMM groups in the binary tree.

As we mentioned before, the near-field matrix is partitioned recursively according to the MLFMM groups up to a certain HSS level and the SVDs of the block-rows without the diagonal blocks are computed.

Once the oct-tree in the MLFMM code has been computed, we obtain information about the distribution of the basis functions in the MLFMM groups, namely, the groups at the lowest level in the oct-tree. It is fundamental to know the smallest and largest block-rows for different values of the previously mentioned K. As the numerical examples have shown, the maximum number of levels K is chosen related to the number of MLFMM levels. The selection of K in the numerical examples has been made having in mind the complexity estimations of Chapter 6 and especially the trade-off between a fast solver and a moderate time to set up the HSS preconditioner.

In light of this information we can set the already introduced nsv, i.e., the number of largest singular triplets to be computed in the HSS representation of the near-field (See Remark 13).

As we previously mentioned in Chapter 6, the main sources of computational cost in terms of floating point operations are the sparse matrix-vector products, the vector operations and the reorthogonalization process. As far as the first is concerned, it is obvious that the sparsity of the near-field matrix and consequently the sparsity of

the block-rows represents a point in favor of this approach to build a preconditioner via sparse SVDs. The second issue, namely, the vector operations do not represent a problem for electrically larger objects, provided we are able to localize the nonzero entries in a submatrix of each block-row. This is unfortunately not the case for the SHSS package yet and represents thus the first and major computational bottleneck we have to remove in the future.
The estimation of the number of flops for the set up of the HSS preconditioner provided in Chapter 6 yielded, under some assumptions, an $\mathcal{O}(N^2)$ algorithm. The assumptions we made so as to obtain this computational complexity work not only in favor but also against our preconditioner. Recall that we did not provide a sharp estimation of the number of nonzero entries on the different block-rows along the levels but confined ourselves to a worst-case scenario due to the strong dependence on the geometry of the scatterer. Moreover, it is also clear that the computation of the HSS preconditioner based on the MLFMM groups rather than on a simple bisection procedure will have far-reaching effects on the computation of the nsv-largest triplets performed for each block-row on every level. The assumption on the maximum number of iterations to achieve convergence of the Lanczos vectors used in PROPACK tried to reflect somehow our numerical experience with the approach based on the MLFMM groups. A similar claim for the recursive bisection partitioning can not be made due to the lack of such a version in FORTRAN.
However, the comparison with the initial results obtained via the recursive bisection of the near-field matrix indices showed that the MLFMM-based approach leads to an important storage saving with respect to the former approach.

A.1.2 Time Complexity of PROPACK

Although the estimation of the number of flops involved in the construction of the HSS preconditioner provides certainly a measure of their performance, it is also important to assess the amount of time that this construction requires. In the part of Chapter 6 devoted to the numerical results these effective CPU times were provided for the examples considered.
In the numerical experiments performed so far, we have observed that due to the rapidly decreasing sparsity of the block-rows in lower levels, the time needed for computing the SVDs changes minimally on levels higher than K.
Figure A.2 shows the computation times for the cylindrical waveguide example computed in the last chapter with 4 levels HSS and 200 singular values for every block-row on each level. These figures are representative for the performance of PROPACK in the examples presented in the last chapter. Due to the high sparsity of the block-

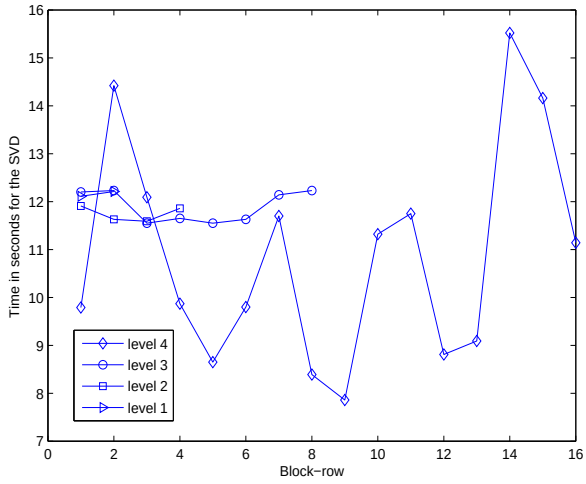


Figure A.2: *Computation time of the SVDs for every block-row on each HSS level for the cylindrical waveguide.*

rows on all levels the computation time for lower levels, where the block-rows become larger, does not increase significantly since the pertaining block-rows are referenced only in the matrix-vector product operation. At least for relatively small examples the time spent in vector operations and orthogonalizing does not seem to change the depicted situation.

A.1.3 An Important Remark Concerning the HSS Preconditioner

As far as the implementation of the preconditioner is concerned, we have to point out that the way we have implemented it, namely, on top of the MLFMM code, has the advantage that in the oct-tree we obtained, empty-groups have been already removed, i.e, the irregularities in the geometry have already been addressed. Nevertheless, the number of basis functions in each group is far from being uniform and even on the matrix level we have to address this issue unless we want to obtain a completely unbalanced tree.
Figure A.3 shows the irregular distribution of the number of basis functions which occurs even for an uncomplicated geometry as a cylinder.

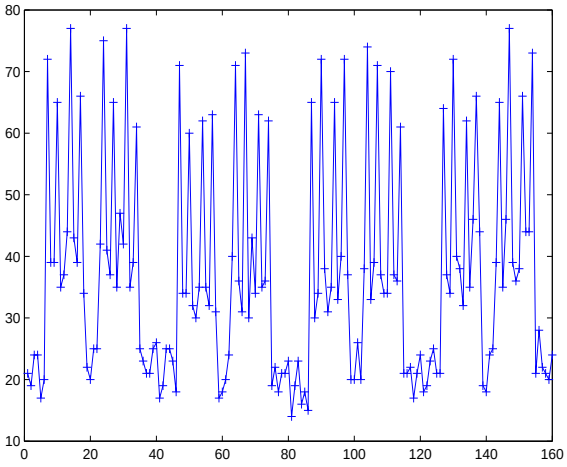


Figure A.3: *Distribution of the basis functions in the MLFMM groups for the cylindrical waveguide. The x-axis represents the MLFMM groups and the y-axis represents the number of basis functions in the corresponding MLFMM group.*

A.2 The HSS Solver

The implementation of the solver must be done carefully since it dominates the time required for the solution process. In the chapter devoted to the HSS framework we mentioned that the solver carries out an implicit ULV^H decomposition of the HSS representation. In fact, we note that several computations can be saved, provided we can cope with the extra storage required. The only vectors or matrices that need to be computed in every iteration are those involving the right hand side of the linear system. Those matrices which are based on the HSS representation of the HSS preconditioner are always the same as can be easily seen in Section 6.2.4.
In order to reflect this situation we perform the most expensive operations in terms of flops as described in Chapter 6 in a `solver_HSS_init`. This solver routine is executed only once at the beginning of the execution of the solver, i.e., in the first iteration. From the second iteration on, we call `solver_HSS_pos` which basically performs the operations related to the right hand side including the solving of the linear systems arising from the lower triangularized diagonals equal to the transformed right hand side (see Step 5 in Section 2.4 of Chapter 6).
A similar situation occurs in the merging process. With the exception of the merging of the subvectors conforming the partitioned right hand side, the rest of operations can be executed once for the whole solving process. Similar to the situation described

in the last paragraph, we distinguish between a `merge_init` and a `merge_pos`. This step makes the solver fast but at the same time it requires to store the information during the complete solving process.
Certainly, our implementation is far from being optimal in the sense of performance. This preliminary version of the SHSS package was written with the sole purpose of obtaining results without devoting too much attention to details. We are perfectly aware that there is obviously ample room for improvement.
In particular, concerning the solver, we provided in some detail the matrix operations to be performed. Especially the unitary matrices $Q_{K;i}$ and $S_{K;i}$ played a fundamental role in it. The way we store these matrices is of crucial importance since we need them at the end of the process to obtain the solution of the linear system we wanted to solve. The cheapest way to store them is to represent these matrices in terms of elementary reflectors in the fashion LAPACK does. In the implementation, we rather formed the matrices explicitly so as to reference them as matrices and not as a structure. Obviously, the cost incurred in this step is rather significant, not only in terms of storage but also in terms of time. Recall, however, that the time to compute these unitary matrices will afect only the call to `solver_HSS_init`. This point will be addressed in the future.
Similar situation applies to the merging process. In this step new matrices have to be formed and especially the storage of the new diagonal matrices resulting from the merging process (see Lemma 14) would be worth a revision.

A.3 A Declaration of Intent for the SHSS Package

The list below enumerates some of the intended improvements for the SHSS package

- Localization of the submatrix of each block-row which contains the nonzero entries in order to improve substantially the time required for the set-up process.
- Extend the set-up of the HSS preconditioner to a non-constant number of singular triplets and define criteria to determine this number on each level in the different block-rows (See also Remark 13).
- Consider other factorizations to create the HSS representation of the near-field matrix which maintain the sparsity of the block-rows described in the section concerning the set-up of the HSS preconditioner in Chapter 6.
- Revisit the storage of the new diagonal matrices arising in the merging process so as to improve the memory consume.

- Store the unitary matrices computed in the solving process in terms of elementary reflectors and create or apply the corresponding routines to operate with them.
- Possible parallelization with MPI.

The points listed above constitute a future extension of the SHSS package. Taking into account that the basic code is in place, these improvements should be feasible. The parallelization of both the set-up and the solver constitutes the next step in order to make this preconditioner competitive within industrial applications.

Bibliography

[1] G. Alleon, M. Benzi and L. Giraud, "Sparse Approximate Inverse Preconditioning for Dense Linear Systems Arising in Computational Electromagnetics", *Numerical Algorithms* 16, 1, pp. 1-15, 1997.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, A. M. S. Hammarling, S. Ostrouchov, and D. Sorensen, LAPACK Users' Guide, Second Edition, SIAM, Phildelphia, 1995.

[3] M. Arioli, V. Pták, Z. Strakos, "Krylov sequences of maximal length and convergence of GMRES", *BIT*, vol. 38, pp. 636-643, 1998.

[4] J. Barnes, P. Hut, "A Hierarchical $\mathcal{O}(N\log N)$ Force-calculated Algorithm ", *Nature*, vol. 324, pp. 446-449, Dezember, 1986.

[5] M. Bebendorf, "Why approximate LU decompositions of finite element discretizations of elliptic operators can be computed with almost linear complexity", Preprint 8/2005, Max-Planck-Institut MiS, Leipzig, submitted for publication.

[6] M. Benzi, M. Tuma, "A sparse approximate inverse preconditioner for nonsymmetric linear systems", *SIAM J. Sci. Comput.*, vol. 19, no. 3, pp. 968-994, 1998.

[7] S. Börm, L. Grasedyck, W. Hackbusch, "Hierarchical Matrices", Lecture note no. 21, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2003.

[8] M. Brüning, P. Benner, A. Bunse-Gerstner, R. Bunger, J. Reiter, J. Ritter, "A Sparse Approximate Inverse Preconditioner for the Method of Moments accelerated with the Multilevel Fast Multipole Method", *2002 IEEE Antennas and Propagation Society International Symposium and USNC/URSI National Radio Science Meeting*, San Antonio, USA, June 16 - 21, 2002, pp. 602-605.

[9] M. Brüning, A. Bunse-Gerstner, R. Bunger, J. Reiter, J. Ritter, "CSYM - An iterative solution method for systems of linear equations with complex symmetric coefficient matrix", *2002 IEEE Antennas and Propagation Society International Symposium and USNC/URSI National Radio Science Meeting*, San Antonio, USA, June 16 - 21, 2002, pp. 626 - 629.

[10] M. Brüning, A. Bunse-Gerstner, R. Bunger, J. Reiter, J. Ritter, "CSYM - An iterative solution method for systems of linear equations with complex symmetric coefficient matrix", *Tätigkeisbericht*, Zentrum für Technomathematik, Universität Bremen, 2001.

[11] M. Brüning, A. Bunse-Gerstner, "Schnelle Löser zur Berechnung elektromagnetischer Felder", Berichte aus der Technomathematik, 2002.

[12] A. Bunse-Gerstner, R. Stöver, "On a conjugate gradient type method for solving complex symmetric linear systems", *Linear Algebra and its Applications*, vol. 287, pp. 105-123, 1999.

[13] A. Bunse-Gerstner, I. Gutiérrez-Cañas, "A preconditioned GMRES for complex dense linear systems from electromagnetic wave scattering problems", to appear in *Linear Algebra and its Applications*.

[14] F. X. Canning, "Singular value decomposition of integral equations of EM and applications to the cavity resonance problem", *IEEE Transactions on Antennas and Propagation*, vol. 37, pp. 1156-1163, 1989.

[15] F. X. Canning, "Protecting EFIE-based scattering computations from effects of interior resonances", *IEEE Transactions on Antennas and Propagation*, vol. 39, pp. 1545-1552, 1991.

[16] B. Carpentieri, "Sparse Preconditioners for Dense Linear Systems from Electromagnetic Applications", Institut National Polytechnique de Toulouse, PhD Thesis, 2002.

[17] B. Carpentieri, I. S. Duff, and L. Giraud, "A class of spectral two-level preconditioners". *SIAM Journal on Scientific Computing* vol. 25 (2), pp. 749-765, 2003.

[18] T. F. Chan, W. L. Wan, "Analysis of projection methods for solving linear systems with multiple right hand sides", *SIAM J. Sci. Comput.*, vol. 18, no. 6, pp. 1698-1721, 1997.

[19] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, A.-J. van der Veen, " Fast stable solver for sequentially semi-separable linear systems of equations", Lectures Notes in Computer Science 2552, pp. 545-554, 2002.

[20] S. Chandrasekaran, M. Gu, T. Pals, "Fast and stable algorithms for hierarchically semi-separable representations", submitted for publication, 2004.

[21] K. Chen, "On a class of preconditioning methods for dense linear systems from boundary elements", *SIAM Journal on Scientific Computing*, vol. 20(2), pp. 684-698, 1998.

[22] W. C. Chew, J. M. Jin, E. Michielssen and J. M. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Artech House, Boston, 2001.

[23] S. H. Christiansen, J.-C. Nédélec, "A preconditioner for the electric field integral equation based on Calderon formulas", *SIAM Journal on Numerical Analysis*, vol. 40(3), pp. 1100-1135, 2002.

[24] R. Coifman, V. Rokhlin, S. Wandzura, "The Fast Multipole Method for the Wave Equation: A Pedestrian Prescription" *IEEE Antennas Propagat. Mag.*, vol. 35, no. 3, June 1993.

[25] E. Darrigrand, "Couplage Méthodes Multipôles-Discrétisation Microlocale pour les Équations Intégrales de l'Electromagnétisme", Université Bordeaux I, PhD Thesis, 2002.

[26] E. Darve, "Methódes Multipôles Rapides: Résolution des Équations de Maxwell par Formulations Intégrales", Université Paris 6, PhD Thesis, 1999.

[27] J. Demmel, "Applied Numerical Linear Algebra", SIAM textbook, 1997.

[28] P. Dewilde, A.-J. van der Veen, "Time-varying Systems and Computations", Kluwer academic publishers, Boston, June 1998.

[29] J. Dongarra, I. Duff, D. Sorensen, H. van der Vorst, "Numerical Linear Algebra for High-Performance Computers", SIAM book, 1998.

[30] I. Duff, L. Giraud, J. Langou, É. Martin, "Using spectral low rank preconditioners for large electromagnetic calculations", *International Journal for Numerical Methods in Engineering*, vol. 62(3), pp. 416-434, January 2005.

[31] Y. Eidelman, I. Gohberg, "Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices", *Computers and Mathematics with Applications*, vol. 33, pp. 69-79, 1997.

[32] Y. Eidelman, I. Gohberg, "A look-ahead block Shur algorithm for diagonal plus semiseparable matrices", *Computers and Mathematics with Applications*, vol. 35, pp. 25-34, 1998.

[33] Y. Eidelman, I. Gohberg, "A modification of the Dewilde and van der Veen method for inversion of finite structured matrices", *Linear Algebra and its Applications*, vol. 343-344, pp. 419-450, 2001.

[34] R. Freund, N. M. Nachtigal, "QMR: A quasi-minimal residual method for non-Hermitian linear systems", *Numer. Math.*, vol. 60, pp. 315-339, 1991.

[35] I. Gohberg, T. Kailath, I. Koltracht, "Linear complexity algorithms for semiseparable matrices", *Integral Equations Operator Theory*, vol. 8, 1985.

[36] G. Golub, C. van Loan, "Matrix Computations", The Johns Hopkins University Press, Baltimore, 1996.

[37] G. Golub, C. Reinsch. "Singular value decomposition and least squares solutions", *Handbook for Automatic Computation II, Linear Algebra*, Springer Verlag, New York, 1971.

[38] S. A. Goreinov, E. E. Tyrtyschnikov, A. Y. Yeremin, "Matrix-free iterative solution strategies for large dense linear systems", *Numer. Linear Algebra Appl.*, 4(4), pp. 273-294, 1997.

[39] S. A. Goreinov, E. E. Tyrtyschnikov, A. Zamarashkin, "A theory of pseudoskeleton approximations", *Linear Algebra Appl.*, vol. 261, pp. 1-21, 1997.

[40] A. Grama, V. Kumar, A. Sameh, "Parallel Hierarchical Solvers and Preconditioners for Boundary Element Methods", *Supercomputing*, Proceedings of the 1996 ACM/IEEE Conference, 1996.

[41] L. Grasedyck, "Theorie und Anwendungen Hierarchischer Matrizen", Universität Kiel, PhD Thesis 2001.

[42] A. Greenbaum, V. Pták, Z. Strakos, "Any nonincreasing convergence curve is possible for GMRES", *SIAM J. Matrix Analysis and Applications*, vol. 17, pp. 465-469, 1996.

[43] L. Greengard, "The rapid evaluation of potential fields in particle system", Cambridge, MA, MIT Press, 1988.

[44] L. Greengard, V. Rokhlin, "A fast algorithm for particle simulations", *J. Comput. Phys.*, vol. 73, pp. 325-349, 1987.

[45] M. Grote, T. Huckle, "Parallel preconditioning with sparse approximate inverses", *SIAM J. Sci. Comput.*, vol. 18, no. 3, pp. 838-853, 1997.

[46] M. Gyure, M. Stalzer, "A prescription for the multilevel Helmholtz FMM", *IEEE Computational Science and Engineering*, vol. 5, no. 3, pp. 39-47, 1998.

[47] M. Gu, S. C. Eisenstat, "A stable and fast algorithm for updating the SVD", Tech. Report YALEU/DCS/RR-966, Department of Computer Science, Yale University, New Haven, CT, 1993.

[48] W. Hackbusch, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I. Introduction to $\mathcal{H}$-matrices", *Computing*, vol. 62, pp. 89-108, 1999.

[49] W. Hackbusch, B. Khoromskij, " A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates". *Journal of Computational and Applied Mathematics*, vol. 125, pp. 479-501, 2000.

[50] P. C. Hansen, "Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion", *SIAM Monographs on Mathematical Modeling and Computation* 4, 1997.

[51] A. Heldring, J. M. Rius, L. P. Ligthart, A. Cardama , "Accurate numerical Modeling of the TARA Reflector System", *IEEE Transactions on Antennas and Propagation*, vol. 52 (7), pp. 1758-1766.

[52] M. R. Hestenes, E. Stiefel, "Methods of conjugate gradients for solving linear systems", *NBS J. Res.*, vol. 49, pp. 409-436, 1952.

[53] G. C. Hsiao, R. E. Kleinman, "Mathematical foundations of error estimation in numerical soultions of integral equation in electromagnetics", *IEEE Transactions on Antennas and Propagation*, vol. 45, pp. 316-328, 1997.

[54] International Electrotechnical Comission, "Electromagnetic compatibility, the role and contribution of IEC standards", 2000. Leaflet available at `http://www.iec.ch/news_centre/onlinepubs/pdf/emc_leaflet.pdf`.

[55] S. Kaniel, "Estimates for some computational techniques in linear algebra", *Math. Comp.*, vol. 20, pp. 369-379, 1966.

[56] M. Kilmer, E. Miller, C. Rappaport, "QMR-based projection techniques for the solution of non-hermitian systems with multiple right-hand sides", preprint.

[57] E. F. Knott, J. F. Shaeffer, T. Michael Tuley, "Radar Cross Section", *SciTech Publishing*, second edition, 2004.

[58] L. Y. Kolotilina, "Explicit preconditioning of systems of linear algebraic equations with dense matrices", *J. Soviet Math.*, vol. 43, pp. 2566-2573, 1988.

[59] L. Landesa, F. Obelleiro, J. L. Rodríguez, M. R. Pino, "Stable solution of the GMT-MOM method by Tikhonov regularization", *Progress in Electromagnetic Research PIER 20*, pp. 45-61, 1998.

[60] R. M. Larsen, " Lanczos bidiagonalization with partial reorthogonalization", Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357, September 1998.

[61] R. M. Larsen," Combining implicit restart and partial reorthogonalization in Lanczos bidiagnalization", SCCM, Stanford University, April 2001.

[62] J. F. Lee, R. Lee, R. J. Burkholder, "Loop Star Basis Functions and a Robust Preconditioner for EFIE Scattering Problems", *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 8, pp. 1855-1863, 2003.

[63] J. Lee, J. Zhang, C.-C. Lu , "Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems",*Journal of Computational Physics*, vol. 185, pp. 158-175, 2003.

[64] J. Liesen, Z. Strakos, "GMRES convergence analysis for a convection-diffusion model problem", *SIAM J. Sci. Comput.*, vol. 26, pp. 1989-2009, 2005.

[65] W. Lyons, "Fast Algorithms and Applications to PDEs", University of California, Santa Barbara, PhD thesis, 2005.

[66] A. Meister, "Numerik linearer Gleichungssysteme", Vieweg, Braunschweig/Wiesbaden, 1999.

[67] J.-C. Nédélec, "Acoustic and Electromagnetic Equations, Integral Representations for Harmonic Problems", Springer Verlag , 2001.

[68] M. Nilsson, "A fast multipole accelerated block quasi minimal residual method for solving scattering from perfectly conducting bodies", *2000 IEEE Antennas and Propagation Society International Symposium Proceedings*, vol. 4, pp. 1848-1851, 2000.

[69] C. C. Paige, "The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices", PhD thesis, University of London United Kingdom, 1971.

[70] A. Peterson, S. L. Ray, R. Mittra, "Computational Methods for Electromagnetics", *IEEE Press Series on Electromagnetic Waves*.

[71] C. Pommerell, "Solution of Large Unsymmetric Systems of Linear Equations", Hartung-Gorre, Konstanz, 1992.

[72] S. M. Rao, D. R. Wilton, A. W. Glisson, "Electromagnetic Scattering by Surfaces of Arbitrary Shape", *IEEE Trans. Antennas Propagat.*, vol. 30, no. 3, pp. 409-418, May 1982.

[73] V. Rokhlin, "Rapid solution of integral equations of classical potential theory", *Journal of Computational Physics*, vol. 60, pp. 187-207, 1985.

[74] I. Rullhusen, "Effizientes feldtheoretisch strenges Verfahren zur Berechnung der Strahlungseigenschaften gekoppelter Aperturstrahler bestehend aus Rechteck- und Rundhohlleiter-Elementen", Abschlussbericht DFG Projekt Ar 138/22-2, Universität Bremen, 2003.

[75] Y. Saad, M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856-869, 1986.

[76] Y. Saad, "Iterative Methods for Sparse Linear Systems", PWS Publishing Company, Boston, 1996.

[77] Y. Saad, A flexibel inner-outer preconditioned GMRES algorithm", *SIAM J. Scientific and Statistical Computing*, vol. 14, pp. 461-469, 1993.

[78] T. K. Sarkar, S. M. Rao, "A Simple Technique for Solving E-Field Integral Equations for Conducting Bodies at Internal Resonances", *IEEE Transactions on Antennas and Propagation*, vol. 30, No. 6, pp. 1250-1254, Nov. 1982.

[79] K. Sertel, "Multilevel Fast Multipole Method for modeling permeable Structures using conformal finite Elements", University of Michigan, PhD Thesis, 2003.

[80] K. Sertel, J. L. Volakis, "Incomplete LU preconditioner for FMM implementation", *Micro. Opt. Tech. Lett.*,26(7), pp.265-267, 2000.

[81] G. L. G. Sleijpen, D. R. Fokkema, "BICGstab(l) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum", *Electronic Transactions on Numerical Analysis*, vol. 1, pp. 11-32, September 1993.

[82] C. F. Smith, A. F. Peterson, R. Mittra, "A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields", *IEEE Trans. Antennas and Propagat.*, vol. 37, pp. 1490-1493, 1989.

[83] J. M. Song, W. C. Chew, "Multilevel fast-multipole algorithm for solving combined field integral equations for electromagnetic scattering", *Microwave Opt. Technol. Lett.*, vol. 10, no. 1, pp. 14-19, Sept. 1995.

[84] P. Sonneveld, "CGS: A fast Lanczos-Type Solver for Nonsymmetric Linear Systems", *SIAM J. Sci. Stat. Comput.*, vol. 10, no. 1, pp. 36-52, 1989.

[85] X. Sun, N. P. Pitsianis, "A Matrix Version of the Fast Multipole Method", *SIAM Review*, vol. 43, no. 2, pp. 289-300, 2001.

[86] R. Vandebril, "Semiseparable Matrices and the symmetric Eigenvalue Problem", Katholieke Universiteit Leuven, PhD thesis, 2004.

[87] H. A. van der Vorst, J. B. M. Melissen, "A Petrow-Galerkin Type Method for Solving Ax=b, where A is Symmetric Complex", *IEEE Trans. Mag.*, vol. 26, no. 2, pp. 706-708, 1990.

[88] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 631-644, 1992.

[89] N. Yarvin, V. Rokhlin, "A Generalized One-Dimensional Fast Multipole Method with Application to Filtering of Spherical Harmonics", *Journal of Computational Physics*, vol. 147, pp. 594-609, 1998.