IT Licentiate theses
2002-004

Iterative Solution of Maxwell's Equations in Frequency Domain

MARTIN NILSSON

UPPSALA UNIVERSITY
Department of Information Technology

UPPSALA
UNIVERSITET

Iterative Solution of Maxwell's Equations in Frequency Domain

BY
MARTIN NILSSON

June 2002

DEPARTMENT OF SCIENTIFIC COMPUTING
INFORMATION TECHNOLOGY
UPPSALA UNIVERSITY
UPPSALA
SWEDEN

Dissertation for the degree of Licentiate of Technology in Numerical Analysis
at Uppsala University 2002

Iterative Solution of Maxwell's Equations in Frequency Domain

Martin Nilsson
`Martin.Nilsson@tdb.uu.se`

Department of Scientific Computing
Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

`http://www.it.uu.se/`

ISSN 1404-5117

Printed by the University Printers, Uppsala University, Sweden

Abstract

We have developed an iterative solver for the Moment Method. It computes a matrix vector product with the multilevel Fast Multipole Method, which makes the method scale with the number of unknowns. The iterative solver is of Block Quasi-Minimum Residual type and can handle several right hand sides at once. The linear system is preconditioned with a Sparse Approximate Inverse, which is modified to handle dense matrices. The solver is parallelized on shared memory machines using OpenMP.

To verify the method some tests are conducted on varying geometries. We use simple geometries to show that the method works. We show that the method scales on several processors of a shared memory machine. To prove that the method works for real life problems, we do some tests on large scale aircrafts. The largest test is a one million unknown simulation on a full scale model of a fighter aircraft.

Keywords: Maxwell's equations, Moment Method, Fast Multipole Method, iterative solution, preconditioner.

Acknowledgments

I wish to thank my supervisor Per Lötstedt for all his valuable advice during this research. I would also like to thank Bo Strand at Saab Avionics who has helped me a lot over the years. I also thank my previous colleague Johan Edlund with whom I have been engaged in countless discussions on the topic of integral equations. He has also developed a large part of the Moment Method code which this solver is based on. Furthermore, I would like to thank Fredrik Edelvik and Gunnar Ledfelt for helping me out from time to time and providing time-domain results. I am thankful of several other people at TDB and at NADA, KTH, who have contributed to this work as well. Finally, I like to thank Jonas Gustafsson, Stefan Hagdahl and Erik Söderström at Saab Avionics for testing and using the code. A special thanks to Jonas Gustafsson who lets me use his results. I would also like to mention that the underlying Moment Method code uses integral kernels from a code developed at CERFACS, Toulouse, France.

This work was supported by the CEM program at the Parallel and Scientific Computing Institute (PSCI) through the General ElectroMagnetic Solvers project (GEMS) and the Signature Modelling And Reduction Tools project (SMART). Financial support was provided by the National Aeronautical Research Program (NFFP), Vinnova, SAAB Avionics AB and Ericsson Microwave Systems AB.

Contents

Chapter 1

Introduction

Products that depend on electromagnetics surround us. Everything from mobile telephones and televisions to large aircrafts and satellites use electricity. Many of these products have antennas that emit electromagnetic radiation when they are used. It is important to understand how the emitted radiation can affect other products and most of all how they can affect us. The development and construction of new devices that depend on electricity are expensive because a lot of time has to be spent on measuring the devices. This is because one wants to check that the device works within specified parameters and that it meets all requirements.

One way of reducing the development time is to use computers to predict the electromagnetic fields. From the predicted fields an engineer can see where a model should be changed in order to achieve a specified level of performance. Computer modeling can be used to give a first clue to which of the best models that should be measured more closely. One can then proceed by constructing the models to see which one performs best. Earlier the engineers had to construct a new model every time they came up with a new idea. This is not necessary with computers that test the models first.

Computer models are used for predicting the electromagnetic environment in problems varying from Electromagnetic Compatibility (EMC), which deals with the electromagnetic interference between different electric devices, to electromagnetic scattering by large objects, which deals with how much an object reflects from an incident plane wave.

The increased capacity of modern day computers means that real life problems can be simulated on a regular PC. However, it is important to note that the simulation time still is several days with standard methods for the type of problems we want to solve here. To address this problem one can either wait for a better computer or develop new methods that reduce the computational time.

There exist two main types of methods for solving electromagnetic problems. The first type is Time-Domain methods (TD) [7, 11, 12, 10, 25, 35, 41]. They solve Maxwell's equations in the time domain and are suitable for broadband applications. The solution is obtained by stepping forward in time. Solutions for all frequencies that can be resolved by the geometry are obtained at once. Time-domain methods usually suffer from dispersion errors which arise because different frequencies move at different speeds on the computational grid. The second type of methods is Frequency-Domain methods (FD) [7, 14, 26, 32, 33]. They solve the time-harmonic Maxwell equations for one frequency at a time. Therefore, they are better suited for narrow band applications.

Frequency-Domain methods are divided in low frequency, mid frequency and high frequency methods. The high-frequency methods are based on asymptotic expansions of the solution and are often of ray-tracing type. They include Physical Optics (PO) [14], Geometrical Theory of Diffraction (GTD) [26] and Uniform Theory of Diffraction (UTD) [32]. The expansions are usually valid when the object is much larger than the wavelength. The low frequency and mid frequency methods include the Moment Method (MM). It is a method that given an external field finds the currents on the surface of an electromagnetic scatterer and then computes the field from the known surface currents. The Moment Method defines an integral equation which is discretized and solved. The solution converges to the exact solution if the resolution is refined. A drawback with this method is that the memory requirements effectively prohibits the possibility of solving the equations for high frequencies. This is what makes the Moment Method a low frequency to mid frequency method. Between the high frequency methods and the mid frequency methods there is a gap which one can not handle with the Moment Method because of memory requirements. It is not possible to use the high frequency methods either because the asymptotic expansions are too inaccurate.

This thesis deals with ways of shortening the solution time for the Moment Method and reducing its memory requirements. Instead of solving the equations by explicitly generating the system of linear equations we can use approximations of them. The approximations are then solved with iterative methods. The advantages with this approach are that we do not require the same amount of memory and that the method is faster provided that the number of iterations in the iterative method is low. This implies that the frequency range in which the moment method can be used is increased.

The approximation to the linear system is achieved through the multilevel Fast Multipole Method for Helmholtz' equation. It can reduce the time of computing a matrix vector multiplication from $\mathcal{O}\left(N^2\right)$ for a standard method to $\mathcal{O}\left(N \log N\right)$, where N is the number of unknowns. It also reduces the memory requirement from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(N \log N\right)$. The reason is that the multilevel Fast Multipole Method computes the action of the matrix on a vector and therefore it does not need to store the matrix in memory.

The outline of this thesis is as follows. In Chapter 2 we discuss the Maxwell equations and derive the integral equations that are used in the Moment Method. Chapter 3 discusses discretization of the integral equations into systems of linear equations. We also mention the methods that are usually used to solve the equations. Next Chapter 4 presents a block iterative method that can be used to solve complex symmetric linear systems of equations with multiple right hand sides. The Fast Multipole Method is described in Chapter 5 followed by a preconditioner for the linear system of equations called the Sparse Approximate Inverse in Chapter 6. We conduct some experiments in Chapter 7 in order to verify the method and end with some conclusions in Chapter 8.

Chapter 2

Maxwell's equations

In this chapter the Maxwell equations are formulated. They are solved using integral equation formulations. The integral equations are used in Chapter 3 to formulate boundary element methods for Maxwell's equations. To achieve symmetric integral operators, the Rumsey reaction theory [37] is applied.

2.1 Equivalent currents

In this section the Maxwell equations are solved based on integral equation formulations. To obtain an integral equation, the concept of equivalent currents will be used. The theory is applicable to homogenous dielectric bodies and homogenous perfect electric conductors. Here, the steps follow quite closely those in [5]. The reader is referred to [24, 27, 33, 37] for more information.

2.1.1 The Maxwell equations in a homogenous dielectric

Let the electric and magnetic fields be time-harmonic in a homogenous dielectric with the time dependence $e^{-\imath\omega t}$. The dielectric domain $\Omega \in \mathbb{R}^3$ is either an interior or an exterior domain. It is characterized by the relative permittivity ϵ and relative permeability μ. The wavenumber is $\kappa = \omega c^{-1}$, where $c = c_0 n^{-1}$. Here, $c_0 = \sqrt{(\epsilon_0\mu_0)^{-1}}$ is the speed of light and $n = \sqrt{\epsilon\mu}$. The impedance $Z = Z_0\sqrt{\mu\epsilon^{-1}}$, where $Z_0 = 120\pi$ in SI-units. Sources are described by applied magnetic currents, denoted by $\mathbf{M}_a$, and applied electric currents, denoted by $\mathbf{J}_a$. The total electromagnetic field in domain Ω is then governed by the Maxwell equations

$$\begin{cases} \boldsymbol{\nabla} \times \mathbf{E} - \imath\kappa Z\mathbf{H} = -\mathbf{M}_a & \mathbf{x} \in \Omega \\ \boldsymbol{\nabla} \times \mathbf{H} + \imath\kappa Z^{-1}\mathbf{E} = \mathbf{J}_a & \mathbf{x} \in \Omega \end{cases} \tag{2.1}$$

Define the normal $\hat{\mathbf{n}}$ to be the one pointing outwards from region Ω. By introducing the equivalent currents $\mathbf{J} = -\hat{\mathbf{n}} \times \mathbf{H}$ and $\mathbf{M} = \hat{\mathbf{n}} \times \mathbf{E}$ on the boundary Γ as in Figure 2.1, the equations are extended to $\mathbb{R}^3$, assumed to be filled by the same dielectric, by the equation

$$\begin{cases} \boldsymbol{\nabla} \times \mathbf{E} - \imath\kappa Z\mathbf{H} = -\mathbf{M} - \mathbf{M}_a & \mathbf{x} \in \mathbb{R}^3 \\ \boldsymbol{\nabla} \times \mathbf{H} + \imath\kappa Z^{-1}\mathbf{E} = \mathbf{J} + \mathbf{J}_a & \mathbf{x} \in \mathbb{R}^3 \end{cases} \tag{2.2}$$

The surface currents replace the fields in $\mathbb{R}^3 \setminus \Omega$ by zero fields so that $\mathbf{E} = \mathbf{0}$ and $\mathbf{H} = \mathbf{0}$ there.

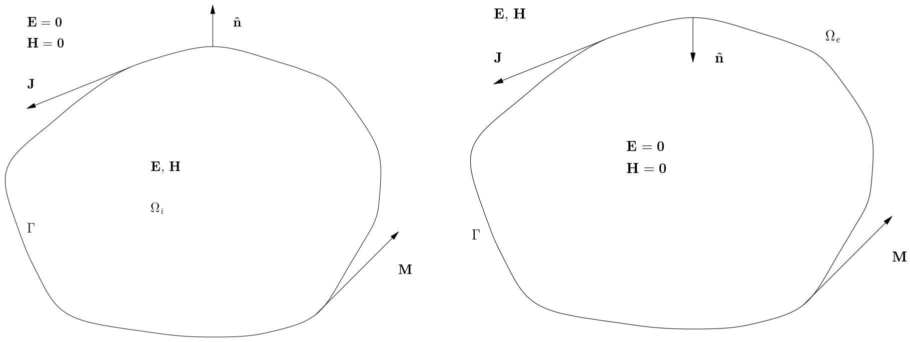

Figure 2.1: The principle of equivalent currents applied to an interior domain Ω_i (left) and an exterior domain Ω_e (right).

The currents that are defined on the boundary Γ of Ω are called the electric current $\mathbf{J}$ and magnetic current $\mathbf{M}$ respectively. In some cases, for instance a scattering problem, it is more convenient to use the normal pointing into region Ω. In this case the equations are adjusted appropriately.

Let $\mathbf{x}$ be a point in space. In the case of an exterior domain it is assumed that the field satisfies the Silver-Müller radiation condition [27]

$$\left|\sqrt{\epsilon}\mathbf{E} - \sqrt{\mu}\mathbf{H} \times \hat{\mathbf{n}}\right| \leq \frac{C}{|\mathbf{x}|^2} \quad |\mathbf{x}| \to \infty \tag{2.3}$$

This condition ensures that the following conditions hold [27]

$$\begin{cases} |\mathbf{E} \cdot \mathbf{x}| \leq \frac{C}{|\mathbf{x}|} \\ |\mathbf{H} \cdot \mathbf{x}| \leq \frac{C}{|\mathbf{x}|} \\ |\mathbf{H} \cdot \mathbf{E}| \leq \frac{C}{|\mathbf{x}|^3} \end{cases} \quad |\mathbf{x}| \to \infty \tag{2.4}$$

The Silver-Müller radiation condition will automatically be taken into account by the Green's function that is used [27].

2.1.2 Helmholtz' equation

The inhomogenous Helmholtz' equation is

$$\nabla^2 \Psi(\mathbf{x}) + \kappa^2 \Psi(\mathbf{x}) = -f(\mathbf{x}) \tag{2.5}$$

The function

$$G(\mathbf{x}, \mathbf{x}') = \frac{e^{\imath\kappa|\mathbf{x}-\mathbf{x}'|}}{4\pi|\mathbf{x}-\mathbf{x}'|} \tag{2.6}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two points in space, is a fundamental solution to Helmholtz' equation called Green's function. It represents outgoing solutions or solutions satisfying the radiation condition of Helmholtz' equation [27]

$$\left|\frac{\partial\Psi}{\partial r} - \imath\kappa\Psi\right| \leq \frac{C}{|\mathbf{x}|^2} \quad r = |\mathbf{x}| \to \infty \tag{2.7}$$

It satisfies

$$\nabla^2 G(\mathbf{x}, \mathbf{x}') + \kappa^2 G(\mathbf{x}, \mathbf{x}') = -\delta(\mathbf{x} - \mathbf{x}') \tag{2.8}$$

where δ is the Dirac measure. The solution to Helmholtz' equation can according to Huygens' principle be expressed as a superposition of waves from individual point sources. Thus, because of the property (2.8), it can be written as

$$\Psi(\mathbf{x}) = \int_{\mathbb{R}^3} f(\mathbf{x}') G(\mathbf{x}, \mathbf{x}') \, d\mathbf{x}' \tag{2.9}$$

The fundamental solution (2.6) is used later to construct fundamental solutions for the Maxwell equations.

2.1.3 The Stratton-Chu formula

Equation (2.2) enables us to express the electromagnetic field using the Green's function of Helmholtz' equation. To do this note that the divergence of (2.2) gives

$$\begin{cases} \boldsymbol{\nabla} \cdot \mathbf{H} = \frac{1}{\imath\kappa Z}(\boldsymbol{\nabla} \cdot \mathbf{M} + \boldsymbol{\nabla} \cdot \mathbf{M}_a) & \mathbf{x} \in \mathbb{R}^3 \\ \boldsymbol{\nabla} \cdot \mathbf{E} = \frac{1}{\imath\kappa Z^{-1}}(\boldsymbol{\nabla} \cdot \mathbf{J} + \boldsymbol{\nabla} \cdot \mathbf{J}_a) & \mathbf{x} \in \mathbb{R}^3 \end{cases} \tag{2.10}$$

which follows from the well-known vector relation $\boldsymbol{\nabla} \cdot \boldsymbol{\nabla} \times \mathbf{A} = 0$. The currents are related to charges through the assumption of conservation of charge. This assumption leads to a relationship between $\nabla \cdot \mathbf{M}$, $\nabla \cdot \mathbf{M}_a$, $\nabla \cdot \mathbf{J}$, $\nabla \cdot \mathbf{J}_a$ and the respective charges ρ_m, $(\rho_m)_a$, ρ_e, $(\rho_e)_a$ of the form

$$\boldsymbol{\nabla} \cdot \mathbf{J} + i\omega\rho_e = 0 \tag{2.11}$$

To decouple the electric field from the magnetic field, the curl is taken of both sides of equation (2.2) yielding

$$\left\{\begin{array}{ll}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\mathbf{E}-\imath\kappa Z\boldsymbol{\nabla}\times\mathbf{H}=-\boldsymbol{\nabla}\times\mathbf{M}-\boldsymbol{\nabla}\times\mathbf{M}_a & \mathbf{x}\in\mathbb{R}^3\\ \boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\mathbf{H}+\imath\kappa Z^{-1}\boldsymbol{\nabla}\times\mathbf{E}=\boldsymbol{\nabla}\times\mathbf{J}+\boldsymbol{\nabla}\times\mathbf{J}_a & \mathbf{x}\in\mathbb{R}^3\end{array}\right. \tag{2.12}$$

Inserting equation (2.2) into equation (2.12) implies

$$\left\{\begin{array}{ll}\begin{aligned}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\mathbf{E}-\kappa^2\mathbf{E}=&\imath\kappa Z\left(\mathbf{J}+\mathbf{J}_a\right)\\&-\left(\boldsymbol{\nabla}\times\mathbf{M}+\boldsymbol{\nabla}\times\mathbf{M}_a\right)\end{aligned} & \mathbf{x}\in\mathbb{R}^3\\ \begin{aligned}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\mathbf{H}-\kappa^2\mathbf{H}=&\imath\kappa Z^{-1}\left(\mathbf{M}+\mathbf{M}_a\right)\\&+\left(\boldsymbol{\nabla}\times\mathbf{J}+\boldsymbol{\nabla}\times\mathbf{J}_a\right)\end{aligned} & \mathbf{x}\in\mathbb{R}^3\end{array}\right. \tag{2.13}$$

Observe that charges are only present on the boundaries. The vector relation $\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\mathbf{A}=\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\mathbf{A}-\boldsymbol{\Delta}\mathbf{A}$ together with equation (2.10) give

$$\left\{\begin{array}{ll}\begin{aligned}-\left(\boldsymbol{\Delta}\mathbf{E}+\kappa^2\mathbf{E}\right)=&\imath\kappa Z\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\mathbf{J}+\mathbf{J}\right)\\&+\imath\kappa Z\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\mathbf{J}_a+\mathbf{J}_a\right)\\&-\boldsymbol{\nabla}\times\mathbf{M}-\boldsymbol{\nabla}\times\mathbf{M}_a\end{aligned} & \mathbf{x}\in\mathbb{R}^3\\ \begin{aligned}-\left(\boldsymbol{\Delta}\mathbf{H}+\kappa^2\mathbf{H}\right)=&\imath\kappa Z^{-1}\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\mathbf{M}+\mathbf{M}\right)\\&+\imath\kappa Z^{-1}\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\mathbf{M}_a+\mathbf{M}_a\right)\\&+\boldsymbol{\nabla}\times\mathbf{J}+\boldsymbol{\nabla}\times\mathbf{J}_a\end{aligned} & \mathbf{x}\in\mathbb{R}^3\end{array}\right. \tag{2.14}$$

Assuming that the applied sources do not intersect the boundary Γ, one can express the applied currents $\mathbf{J}_a$ and $\mathbf{M}_a$ in the fields $\mathbf{E}_a$ and $\mathbf{H}_a$. From the properties of the Green's function associated with Helmholtz' equation in (2.8) it follows that

$$\left\{\begin{array}{ll}\begin{aligned}\mathbf{E}\left(\mathbf{x}\right)=&\mathbf{E}_a\left(\mathbf{x}\right)\\&+\imath\kappa Z\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot+\mathbf{1}\right)\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{J}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\\&-\boldsymbol{\nabla}\times\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{M}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\end{aligned} & \mathbf{x}\notin\Gamma\\ \begin{aligned}\mathbf{H}\left(\mathbf{x}\right)=&\mathbf{H}_a\left(\mathbf{x}\right)\\&+\imath\kappa Z^{-1}\left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot+\mathbf{1}\right)\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{M}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\\&+\boldsymbol{\nabla}\times\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{J}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\end{aligned} & \mathbf{x}\notin\Gamma\end{array}\right. \tag{2.15}$$

Here, it should be mentioned that equation (2.15) can be expressed in the fields, yielding the Stratton-Chu formula [6]

$$\begin{cases} \mathbf{E}(\mathbf{x}) = \mathbf{E}_a(\mathbf{x}) + \int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\boldsymbol{\nabla}\times\mathbf{E}(\mathbf{x}')\times\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \\ \quad + \boldsymbol{\nabla}\int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\mathbf{E}(\mathbf{x}')\cdot\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \qquad \mathbf{x}\notin\Gamma \\ \quad + \boldsymbol{\nabla}\times\int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\mathbf{E}(\mathbf{x}')\times\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \\ \mathbf{H}(\mathbf{x}) = \mathbf{H}_a(\mathbf{x}) + \int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\boldsymbol{\nabla}\times\mathbf{H}(\mathbf{x}')\times\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \\ \quad + \boldsymbol{\nabla}\int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\mathbf{H}(\mathbf{x}')\cdot\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \qquad \mathbf{x}\notin\Gamma \\ \quad + \boldsymbol{\nabla}\times\int_\Gamma G(\mathbf{x},\mathbf{x}')\left(\mathbf{H}(\mathbf{x}')\times\hat{\mathbf{n}}(\mathbf{x}')\right)d\Gamma(\mathbf{x}') \end{cases} \tag{2.16}$$

For more details on these equations see [24].

Three operators are introduced, as in [24], in order to simplify the notation

$$\begin{cases} \mathcal{V}\mathbf{J} = \int_\Gamma G(\mathbf{x},\mathbf{x}')\,\mathbf{J}(\mathbf{x}')\,d\Gamma(\mathbf{x}') & \mathbf{x}\notin\Gamma \\ \mathcal{T}\mathbf{J} = \left(\frac{1}{\kappa^2}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot+\mathbf{1}\right)\mathcal{V}\mathbf{J} & \mathbf{x}\notin\Gamma \\ \mathcal{K}\mathbf{J} = -\boldsymbol{\nabla}\times\mathcal{V}\mathbf{J} & \mathbf{x}\notin\Gamma \end{cases} \tag{2.17}$$

With this notation equation (2.15) becomes

$$\begin{cases} \mathbf{E} = \mathbf{E}_a + \imath\kappa Z\mathcal{T}\mathbf{J} + \mathcal{K}\mathbf{M} & \mathbf{x}\notin\Gamma \\ \mathbf{H} = \mathbf{H}_a + \imath\kappa Z^{-1}\mathcal{T}\mathbf{M} - \mathcal{K}\mathbf{J} & \mathbf{x}\notin\Gamma \end{cases} \tag{2.18}$$

2.2 The Rumsey reaction concept

The contributions to the total electromagnetic field are divided into two parts

$$\begin{cases} \mathbf{E} = \mathbf{E}_a + \mathbf{E}(\mathbf{J},\mathbf{M}) & \mathbf{x}\notin\Gamma \\ \mathbf{H} = \mathbf{H}_a + \mathbf{H}(\mathbf{J},\mathbf{M}) & \mathbf{x}\notin\Gamma \end{cases} \tag{2.19}$$

where $\mathbf{E}_a$ and $\mathbf{H}_a$ are the fields created by the applied sources, while $\mathbf{E}(\mathbf{J},\mathbf{M})$ and $\mathbf{H}(\mathbf{J},\mathbf{M})$ are the fields generated by the equivalent currents. Using equation (2.18)

$$\begin{cases} \mathbf{E}(\mathbf{J},\mathbf{M}) = \imath\kappa Z\mathcal{T}\mathbf{J} + \mathcal{K}\mathbf{M} & \mathbf{x}\notin\Gamma \\ \mathbf{H}(\mathbf{J},\mathbf{M}) = \imath\kappa Z^{-1}\mathcal{T}\mathbf{M} - \mathcal{K}\mathbf{J} & \mathbf{x}\notin\Gamma \end{cases} \tag{2.20}$$

Let $\mathbf{J}'$ and $\mathbf{M}'$ be a set of testing currents that are tangential to Γ. Define the Rumsey reaction [37] of the currents $\mathbf{J}$ and $\mathbf{M}$ on $\mathbf{J}'$ and $\mathbf{M}'$ by

$$R_\Omega\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)=\int_\Gamma\left(\mathbf{E}\left(\mathbf{J},\mathbf{M}\right)\cdot\mathbf{J}'-\mathbf{H}\left(\mathbf{J},\mathbf{M}\right)\cdot\mathbf{M}'\right)d\Gamma \tag{2.21}$$

This reaction can be divided into two parts

$$\begin{aligned}R_\Omega\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)=&R_\Omega^s\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)\\&+R_\Omega^a\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)\end{aligned} \tag{2.22}$$

The R_Ω^s can be viewed as the symmetric part of the reaction and R_Ω^a can be viewed as the antisymmetric part.

The tangential values can be expressed in terms of explicit integral operators. By using jump relations one gets

$$\begin{cases}\begin{aligned}R_\Omega^s\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)=&\imath\kappa Z\int_\Gamma T\mathbf{J}\cdot\mathbf{J}'d\Gamma-\imath\kappa Z^{-1}\int_\Gamma T\mathbf{M}\cdot\mathbf{M}'d\Gamma\\&+\int_\Gamma K\mathbf{M}\cdot\mathbf{J}'d\Gamma+\int_\Gamma K\mathbf{J}\cdot\mathbf{M}'d\Gamma\\R_\Omega^a\left(\{\mathbf{J},\mathbf{M}\},\{\mathbf{J}',\mathbf{M}'\}\right)=&-\tfrac{1}{2}\textstyle\int_\Gamma\left(\hat{\mathbf{n}}\times\mathbf{M}\cdot\mathbf{J}'+\hat{\mathbf{n}}\times\mathbf{J}\cdot\mathbf{M}'\right)d\Gamma\end{aligned}\end{cases} \tag{2.23}$$

Here, T and K are the limits of $\mathcal{T}$ and $\mathcal{K}$ as the boundary is approached in Cauchy principal value sense. They are given by

$$\begin{cases}\begin{aligned}T\mathbf{J}=&\frac{1}{\kappa^2}\boldsymbol{\nabla}_\Gamma\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\boldsymbol{\nabla}_\Gamma\cdot\mathbf{J}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\\&+\left(\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{J}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\right)_t\\K\mathbf{J}=&\left(\textstyle\int_\Gamma\boldsymbol{\nabla}_{\mathbf{x}'}G\left(\mathbf{x},\mathbf{x}'\right)\times\mathbf{J}\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}'\right)\right)_t\end{aligned}\end{cases} \tag{2.24}$$

Here, $\boldsymbol{\nabla}_\Gamma$ denotes the surface divergence of a vector field tangent to Γ and $()_t$ denotes the tangential part. There is also a second contribution from $\mathcal{K}$ given by R_Ω^a. As we will see, the anti-symmetric part vanishes in formulations that are related to the reaction.

The hypersingularity in the first part of T is usually handled by moving one derivative from the $\mathbf{x}'$-variable to the $\mathbf{x}$-variable by using partial integration on the variational formulation yielding

$$\begin{aligned}\int_\Gamma T\mathbf{J}\cdot\mathbf{J}'d\Gamma=&-\int_\Gamma\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\frac{1}{\kappa^2}\boldsymbol{\nabla}_\Gamma\cdot\mathbf{J}\left(\mathbf{x}'\right)\boldsymbol{\nabla}_\Gamma\cdot\mathbf{J}'\left(\mathbf{x}\right)d\Gamma\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}\right)\\&+\int_\Gamma\int_\Gamma G\left(\mathbf{x},\mathbf{x}'\right)\mathbf{J}\left(\mathbf{x}'\right)\mathbf{J}'\left(\mathbf{x}\right)d\Gamma\left(\mathbf{x}'\right)d\Gamma\left(\mathbf{x}\right)\end{aligned} \tag{2.25}$$

The Rumsey reaction on the applied current gives the formula

$$R_\Omega\left(\{\mathbf{J}_a, \mathbf{M}_a\}, \{\mathbf{J}', \mathbf{M}'\}\right) = \int_\Gamma \left(\mathbf{E}_a \cdot \mathbf{J}' - \mathbf{H}_a \cdot \mathbf{M}'\right) d\Gamma \equiv V_\Omega\left(\{\mathbf{J}', \mathbf{M}'\}\right) \quad (2.26)$$

in the same way.

2.2.1 The Electric Field Integral Equation

Consider the case of a perfect electric conductor. The boundary condition is $\mathbf{E}_t = \hat{\mathbf{n}} \times \mathbf{E} = \mathbf{M} = \mathbf{0}$. The same condition is imposed on the magnetic testing current such that $\mathbf{M}' = \mathbf{0}$. The total field satisfies

$$\int_\Gamma \left(\mathbf{E} \cdot \mathbf{J}' - \mathbf{H} \cdot \mathbf{M}'\right) d\Gamma = 0 \quad (2.27)$$

for all $\mathbf{J}'$ and $\mathbf{M}'$ that satisfy the condition. Since $R^a_\Omega = 0$ in this case, it follows that

$$R^s_\Omega\left(\{\mathbf{J}, \mathbf{0}\}, \{\mathbf{J}', \mathbf{0}\}\right) = -V_\Omega\left(\{\mathbf{J}', \mathbf{0}\}\right) \quad (2.28)$$

This equation is called the Electric Field Integral Equation (EFIE). From the definition of R^s_Ω and V_Ω it follows that

$$\int_\Gamma T\mathbf{J} \cdot \mathbf{J}' d\Gamma = -\frac{1}{\imath\kappa Z} \int_\Gamma \mathbf{E}_a \cdot \mathbf{J}' d\Gamma \quad (2.29)$$

One can also derive the EFIE by multiplying equation (2.18) with the same testing current directly in a variational formulation.

2.2.2 The Magnetic Field Integral Equation

Another integral equation for the Perfect Electric Conductor case is the Magnetic Field Integral Equation (MFIE). In this case it is not straightforward to derive the equation from the Rumsey reaction concept. Instead recall that for $\mathbf{M} = \mathbf{0}$ equation (2.15) yields

$$\mathbf{H} = \mathbf{H}_a - \mathcal{K}\mathbf{J} \quad \mathbf{x} \in \Omega \quad (2.30)$$

In the case of a scattering problem the normal is defined as pointing into the region. The definition of the surface current is then $\mathbf{J} = \hat{\mathbf{n}} \times \mathbf{H}$. Equation (2.15) together with the boundary limit of $\mathcal{K}$ gives

$$\mathbf{J} = \hat{\mathbf{n}} \times \mathbf{H} = \hat{\mathbf{n}} \times \mathbf{H}_a - \left(-\frac{1}{2}\mathbf{J} + \hat{\mathbf{n}} \times K\mathbf{J}\right) \quad (2.31)$$

Multiplying with a test current $\mathbf{J}'$ defined as before and integrating yield the MFIE equation

$$\frac{1}{2}\int_{\Gamma} \mathbf{J} \cdot \mathbf{J}' d\Gamma + \int_{\Gamma} \hat{\mathbf{n}} \times K\mathbf{J} \cdot \mathbf{J}' d\Gamma = \int_{\Gamma} \hat{\mathbf{n}} \times \mathbf{H}_a \cdot \mathbf{J}' d\Gamma \tag{2.32}$$

2.2.3 The Combined Field Integral Equation

The EFIE (2.29) and the MFIE (2.32) both have different advantages and disadvantages. The EFIE has spurious solutions if κ^2 is a corresponding eigenvalue to the interior problem with perfect electric conducting walls. In case κ^2 is an eigenvalue, the spurious modes will not radiate in the exterior. Thus, the field is not corrupted outside of the object. The MFIE can also have nullspace solutions in the interior. However, the spurious solutions of the MFIE do radiate in the exterior domain and corrupt the field. The EFIE can also handle the case of an open object which the MFIE is not able to handle. Therefore, the EFIE would be the best choice, when solving the perfect electric conductor case. However, the EFIE suffers from ill-conditioning, while the MFIE is better conditioned thanks to the term $\frac{1}{2}\mathbf{J}$ in (2.31). The reason is that the EFIE is an Fredholm integral equation of the first kind, while the MFIE is a Fredholm integral equation of the second kind [24]. Thus, the MFIE is more appropriate for an iterative solution method.

The solution to these problems is to combine the two formulations into one called the Combined Field Integral Equation (CFIE). The CFIE is a linear combination of the EFIE (2.29) and the MFIE (2.32) defined as

$$\text{CFIE} = \alpha\text{EFIE} + (1-\alpha)\,\tfrac{\imath}{\kappa}\text{MFIE} \quad 0 < \alpha < 1 \tag{2.33}$$

With this choice there are no spurious modes. From experience it turns out that this equation has better conditioning than both EFIE and MFIE [7]. Since the formulation involves MFIE it can not be used for open objects. Thus, the CFIE is used for closed objects while EFIE is used for open objects involving perfect electric conductors.

2.2.4 Dielectric materials

Consider a dielectric material with wavenumber κ_+ in the interior domain and κ_- in the exterior domain. Define a set of equivalent currents in the interior domain $\mathbf{J}_+$ and $\mathbf{M}_+$. In the same way a set of exterior equivalent currents $\mathbf{J}_-$ and $\mathbf{M}_-$ are defined. From continuity of the tangential magnetic field over the boundary it follows that $\hat{\mathbf{n}}_+ \times \mathbf{H}_+ + \hat{\mathbf{n}}_- \times \mathbf{H}_- = \mathbf{J}_+ + \mathbf{J}_- = \mathbf{0}$, since the normals point in opposite directions. The condition for the magnetic currents, $\mathbf{M}_+ + \mathbf{M}_- = \mathbf{0}$, is obtained from the jump relation for the electric field in the same way. As in the case of EFIE a set of tangential test functions are defined. The reaction is defined as

$$\int_\Gamma \left(\mathbf{E}_+ \cdot \mathbf{J}'_+ - \mathbf{H}_+ \cdot \mathbf{M}'_+\right) d\Gamma + \int_\Gamma \left(\mathbf{E}_- \cdot \mathbf{J}'_- - \mathbf{H}_- \cdot \mathbf{M}'_-\right) d\Gamma = 0 \tag{2.34}$$

for all test functions satisfying $\mathbf{J}'_+ + \mathbf{J}'_- = \mathbf{0}$ and $\mathbf{M}'_+ + \mathbf{M}'_- = \mathbf{0}$. This is equivalent to

$$\begin{aligned}
& R^s_\Omega\left(\{\mathbf{J}_+, \mathbf{M}_+\}, \{\mathbf{J}'_+, \mathbf{M}'_+\}\right) + R^s_\Omega\left(\{\mathbf{J}_-, \mathbf{M}_-\}, \{\mathbf{J}'_-, \mathbf{M}'_-\}\right) \\
&= \int_\Gamma \left(\imath\kappa_+ Z_+ T_+ \mathbf{J}_+ \cdot \mathbf{J}'_+ + K_+ \mathbf{M}_+ \cdot \mathbf{J}'_+\right) d\Gamma \\
&\quad + \int_\Gamma \left(-\imath\kappa_+ Z_+^{-1} T_+ \mathbf{M}_+ \cdot \mathbf{M}'_+ + K_+ \mathbf{J}_+ \cdot \mathbf{M}'_+\right) d\Gamma \\
&\quad + \int_\Gamma \left(\imath\kappa_- Z_- T_- \mathbf{J}_- \cdot \mathbf{J}'_- + K_- \mathbf{M}_- \cdot \mathbf{J}'_-\right) d\Gamma \\
&\quad + \int_\Gamma \left(\imath\kappa_- Z_-^{-1} T_- \mathbf{M}_- \cdot \mathbf{M}'_- + K_- \mathbf{J}_- \cdot \mathbf{M}'_-\right) d\Gamma \\
&= -\int_\Gamma \left(\mathbf{E}_a \cdot \mathbf{J}'_- - \mathbf{H}_a \cdot \mathbf{M}'_-\right) d\Gamma = V_\Omega\left(\{\mathbf{J}'_-, \mathbf{M}'_-\}\right)
\end{aligned} \tag{2.35}$$

The anti-symmetric part of the reaction vanishes from the constraints on the currents and the test currents.

2.2.5 Extension to multilayered dielectrics and metals

The Rumsey reaction concept can quite easily be extended to problems involving metals and several layers of dielectrics. Let Ω_k be a dielectric domain for $1 \leq k \leq N$, where Ω_N is the exterior domain. In each domain, define κ_k and Z_k to be the wavenumber and impedance of the medium. Also, let $\mathbf{E}^k$ and $\mathbf{H}^k$ be the field produced by the applied sources in Ω_k. Let Ω_0 be a perfect electric conducting domain. The boundary between domain Ω_k and domain Ω_l is Γ_{kl}, as in Figure 2.2. The problem is now to find $\mathbf{E}$ and $\mathbf{H}$ such that

$$\left\{\begin{array}{lll}
\boldsymbol{\nabla} \times \mathbf{E} - \imath\kappa_k Z_k \mathbf{H} = -\mathbf{M}^k_a & \mathbf{x} \in \Omega_k & 1 \leq k \leq N \\
\boldsymbol{\nabla} \times \mathbf{H} + \imath\kappa_k Z_k^{-1} \mathbf{E} = \mathbf{J}^k_a & \mathbf{x} \in \Omega_k & 1 \leq k \leq N \\
\mathbf{E}^k_t = \mathbf{E}^l_t & \mathbf{x} \in \Gamma_{kl} & 1 \leq k, l \leq N \\
\mathbf{H}^k_t = \mathbf{H}^l_t & \mathbf{x} \in \Gamma_{kl} & 1 \leq k, l \leq N \\
\mathbf{E}^k_t = \mathbf{0} & \mathbf{x} \in \Gamma_{k0} & 1 \leq k \leq N
\end{array}\right. \tag{2.36}$$

The differences $\mathbf{E} - \mathbf{E}^N_a$ and $\mathbf{H} - \mathbf{H}^N_a$ are required to satisfy a radiation condition in Ω_N.

Let the equivalent currents in domain Ω_k be $\mathbf{J}^k = -\hat{\mathbf{n}}^k \times \mathbf{H}^k$ and $\mathbf{M}^k = \hat{\mathbf{n}}^k \times \mathbf{E}^k$. Then it follows that $\mathbf{M}^k = \mathbf{0}$ for $\mathbf{x} \in \Gamma_{k0}$ and that for $\mathbf{x} \in \Gamma_{kl}$ $\mathbf{J}^k + \mathbf{J}^l = \mathbf{0}$ and $\mathbf{M}^k + \mathbf{M}^l = \mathbf{0}$. Using the symmetric part of the Rumsey

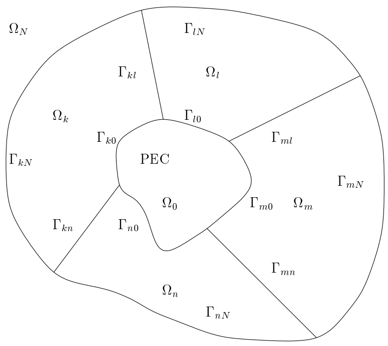

Figure 2.2: Several layers and the corresponding notation for the domains.

reaction in Section 2.2, the equation can be written as a sum of contributions from reactions in each domain

$$\sum_{k=1}^{N} R^s_{\Omega_k}\left(\left\{\mathbf{J}^k, \mathbf{M}^k\right\}, \left\{\left(\mathbf{J}^k\right)', \left(\mathbf{M}^k\right)'\right\}\right) = \sum_{k=1}^{N} V_{\Omega_k}\left(\left\{\left(\mathbf{J}^k\right)', \left(\mathbf{M}^k\right)'\right\}\right) \tag{2.37}$$

2.3 The far field

In electromagnetic problems it is often of interest to know how the electromagnetic field behaves for an observer far from the source. Let $\mathbf{x} = r\hat{\boldsymbol{\kappa}}$, where $\hat{\boldsymbol{\kappa}}$ is a unit vector, and $r \to \infty$. Then,

$$\begin{aligned} |\mathbf{x} - \mathbf{x}'| &= \sqrt{r^2 - 2r\hat{\boldsymbol{\kappa}} \cdot \mathbf{x}' + |\mathbf{x}'|^2} \\ &= r\sqrt{1 - \frac{2\hat{\boldsymbol{\kappa}} \cdot \mathbf{x}'}{r} + \frac{|\mathbf{x}'|^2}{r^2}} \\ &\approx r - \hat{\boldsymbol{\kappa}} \cdot \mathbf{x}' \end{aligned} \tag{2.38}$$

Applying this procedure to $\mathcal{V}\mathbf{J}$ yields

$$\begin{aligned} \mathcal{V}\mathbf{J} &= \int_\Gamma G\left(\mathbf{x}, \mathbf{x}'\right) \mathbf{J}\left(\mathbf{x}'\right) d\Gamma\left(\mathbf{x}'\right) \\ &\approx \frac{e^{\imath\kappa r}}{4\pi r} \int_\Gamma e^{-\imath\boldsymbol{\kappa}\cdot\mathbf{x}'} \mathbf{J}\left(\mathbf{x}'\right) d\Gamma\left(\mathbf{x}'\right) \\ &\equiv \frac{e^{\imath\kappa r}}{r} F\mathbf{J}\left(\boldsymbol{\kappa}\right) \end{aligned} \tag{2.39}$$

Here, κ is the wavenumber and $\boldsymbol{\kappa} = \kappa\hat{\boldsymbol{\kappa}}$ is a vector of length κ pointing in the direction of $\hat{\boldsymbol{\kappa}}$.

Using the relation $\boldsymbol{\nabla}_{\mathbf{x}'} e^{-\imath\boldsymbol{\kappa}\cdot\mathbf{x}'} = -\imath\boldsymbol{\kappa} e^{-\imath\boldsymbol{\kappa}\cdot\mathbf{x}'}$ together with the definition of $\mathcal{T}$ gives the far field for $\mathcal{T}$

$$\mathcal{T}\mathbf{J} \approx \frac{e^{\imath\kappa r}}{r}\left(F\mathbf{J}\left(\boldsymbol{\kappa}\right) - \hat{\boldsymbol{\kappa}}\left(\hat{\boldsymbol{\kappa}}\cdot F\mathbf{J}\left(\boldsymbol{\kappa}\right)\right)\right) = \frac{e^{\imath\kappa r}}{r}\hat{\boldsymbol{\kappa}} \times \left(F\mathbf{J}\left(\boldsymbol{\kappa}\right) \times \hat{\boldsymbol{\kappa}}\right) \tag{2.40}$$

The same relationship applied to $\mathcal{K}$ gives the far field generated by $\mathcal{K}$

$$\mathcal{K}\mathbf{M} \approx \frac{e^{\imath\kappa r}}{r}\left(-\imath\boldsymbol{\kappa} \times F\mathbf{M}\left(\boldsymbol{\kappa}\right)\right) \tag{2.41}$$

The equations (2.40) and (2.41) are used later to calculate the Radar Cross Section. They will also appear in the expressions for the Fast Multipole Method.

2.3.1 Plane wave scattering

Consider the case of electromagnetic scattering from a plane wave with wavenumber κ described by

$$\begin{cases} \mathbf{E}_a\left(\mathbf{x}, \hat{\boldsymbol{\kappa}}_a\right) = \mathbf{E}_0 e^{-\imath\kappa\hat{\boldsymbol{\kappa}}_a\cdot\mathbf{x}} \\ \mathbf{H}_a\left(\mathbf{x}, \hat{\boldsymbol{\kappa}}_a\right) = \mathbf{H}_0 e^{-\imath\kappa\hat{\boldsymbol{\kappa}}_a\cdot\mathbf{x}} \end{cases} \tag{2.42}$$

The wave is traveling in the direction given by the unit vector $-\hat{\boldsymbol{\kappa}}_a$. Let $\mathbf{E}_s\left(\mathbf{x}, \hat{\boldsymbol{\kappa}}_a\right)$ and $\mathbf{H}_s\left(\mathbf{x}, \hat{\boldsymbol{\kappa}}_a\right)$ be the corresponding scattered electromagnetic field. Then the bistatic Radar Cross Section (RCS) $\sigma\left(\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)$ is defined as

$$\sigma\left(\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right) = \lim_{r\to\infty} 4\pi r^2 \frac{\left|\mathbf{E}_s\left(r\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)\right|^2}{\left|\mathbf{E}_a\left(r\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)\right|^2} \tag{2.43}$$

The special case $\sigma\left(\hat{\boldsymbol{\kappa}}_a, \hat{\boldsymbol{\kappa}}_a\right)$ is called the monostatic RCS. The RCS is often computed in decibels (dB) by the relation $\sigma_{\mathrm{dB}}\left(\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right) = 10\log_{10}\sigma\left(\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)$.

From the definition of the far field it follows that one can express the scattered field as

$$\mathbf{E}_s\left(r\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right) \approx \tfrac{e^{\imath\kappa r}}{r}\mathbf{A}\left(\hat{\boldsymbol{\kappa}}, F\mathbf{J}\right) \quad r \to \infty \tag{2.44}$$

Here, $\mathbf{A}\left(\hat{\boldsymbol{\kappa}}, F\mathbf{J}\right)$ is defined by the behavior in the far field by the equations (2.18), (2.40) and (2.41). Thus, the RCS is given by

$$\sigma\left(\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right) = \lim_{r\to\infty} 4\pi r^2 \frac{\left|\mathbf{E}_s\left(r\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)\right|^2}{\left|\mathbf{E}_a\left(r\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{\kappa}}_a\right)\right|^2} = 4\pi \frac{\left|\mathbf{A}\left(\hat{\boldsymbol{\kappa}}, F\mathbf{J}\right)\right|^2}{\left|\mathbf{E}_0\right|^2} \tag{2.45}$$

The function $\mathbf{A}\left(\boldsymbol{\kappa}, F\mathbf{J}\right)$ is usually referred to as the far field pattern.

The Radar Cross Section is used later when measuring the numerical errors from different methods.

Chapter 3

Boundary Element Method

The integral equations from Chapter 2 are discretized using the Boundary Element Method. The result is a dense linear system of equations. Different solution methods for the integral equations are suggested. In Chapter 5 the Fast Multipole Method is used. The result is a fast iterative method for the solution of the dense linear system.

3.1 Methods for solving the Maxwell equations

Integral equations can be solved using Boundary Element Methods. Boundary Element Methods discretize the integral equations so that a dense linear systems of equations is obtained. The discretization is achieved by a Finite Element discretization of the surface integrals involved. In the case of the Maxwell equations, Boundary Element Methods are usually called Moment Methods (MM).

An advantage with moment methods is that because of their formulation they are free from dispersion errors, no outer artificial boundary is needed for exterior problems and only surfaces have to be discretized in $\mathbb{R}^3$.

Other methods for solving frequency domain problems include Physical Optics (PO) [14], Geometrical Optics (GO), the Geometrical Theory of Diffraction (GTD) [26] and the Uniform Theory of Diffraction (UTD) [32]. They are based on asymptotic high-frequency expansions of the equations and therefore also best suited for high-frequency problems. Physical Optics is based on the fact that the current on an infinite ground plane is known analytically. Therefore the approximation can be used on large surfaces with little curvature. Geometrical Optics is a method for deciding, which parts of an object that should be excited by a Physical Optics approximation. Volume integral methods [7, 33] and Finite Element methods [7, 33] are also used in frequency domain.

One can also solve the time-dependent Maxwell equations using time-domain methods. Methods in time-domain include the Finite-Difference Time-Domain method (FDTD) [41], Finite-Volume Time-Domain methods (FVTD) [11, 10, 35] and Finite-Element Time-Domain methods (FETD) [12, 25]. One can also use integral equation methods in time called Marching-On-in-Time methods (MOT) [7].

3.2 Discretization

In order to solve the integral equations EFIE (2.29), MFIE (2.32), CFIE (2.33) and for multilayered materials (2.37) in section 2.2, the equations will be approximated numerically. The numerical approximation gives a dense complex system of linear equations in the unknown currents.

3.2.1 Triangulation

In order to discretize the objects, a suitable representation of the geometry has to be found. In this case each medium has a separate description of its surfaces. The surfaces are partitioned into triangles, as in Figure 3.1. This gives a perfect approximation of flat surfaces, but introduces a geometrical error of $\mathcal{O}\left(h^2\right)$ on curved surfaces. Here, h is in the order of the length of the longest edge among all triangles.

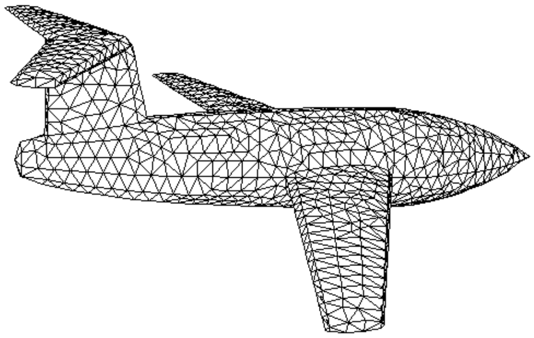

Figure 3.1: An example of an object surface represented by triangles. In this case it is a generic aircraft model called RUND.

3.2.2 The Rao-Wilton-Glisson basis functions

The so called Rao-Wilton-Glisson (RWG) basis functions [34] can be used to approximate the surface currents of the triangulated object. A RWG basis function is defined over the common edge of two adjacent triangles. If the two triangles are denoted by T_+ and T_-, then the RWG basis function is defined as

$$\mathbf{j}(\mathbf{x}) = \begin{cases} \frac{1}{2A_+}(\mathbf{x} - \mathbf{x}_+) & \mathbf{x} \in T_+ \\ -\frac{1}{2A_-}(\mathbf{x} - \mathbf{x}_-) & \mathbf{x} \in T_- \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{3.1}$$

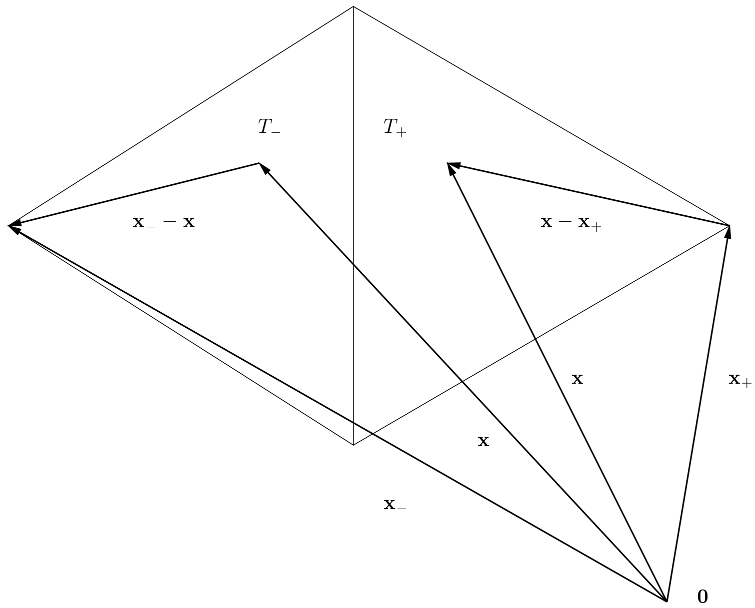

Figure 3.2: The Rao-Wilton-Glisson basis function (RWG).

Here, $\mathbf{x}_+$ and $\mathbf{x}_-$ are the two corners of the triangles not shared by the common edge, as in Figure 3.2. The RWG basis functions ensure that there is no accumulation of line-charges at the outer boundaries of the triangle pair. The scaling makes the normal continuous across the common edge. Also, the divergence on the triangle surface satisfies

$$\nabla_\Gamma \cdot \mathbf{j} = \begin{cases} \frac{1}{A_+} & \mathbf{x} \in T_+ \\ -\frac{1}{A_-} & \mathbf{x} \in T_- \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{3.2}$$

3.2.3 Discretizing the equations

To discretize the integral equations basis functions are used to expand the currents as

$$\begin{cases} \mathbf{J}(\mathbf{x}) = \sum_{l=1}^{N^e} I_l^e \mathbf{j}_l(\mathbf{x}) \\ \mathbf{M}(\mathbf{x}) = \sum_{l=N^e+1}^{N^e+N^m} I_l^m \mathbf{m}_l(\mathbf{x}) \end{cases} \tag{3.3}$$

Here, $\mathbf{j}_l(\mathbf{x})$ is an electric current basis function and $\mathbf{m}_l(\mathbf{x})$ is a magnetic one, N^e is the number of electric basis functions and N^m is the number of magnetic basis functions and $N = N^e + N^m$. The same basis functions are used in the testing procedure, resulting in a Galerkin method. If a triangular patch has both a magnetic current and an electric current, then its basis functions are of the same kind.

The expansion ends up in an equation $\mathbf{ZI} = \mathbf{V}$, where Z_{kl} is given by inserting (3.3) and letting $\mathbf{J}' = \mathbf{j}_l$ and $\mathbf{M}' = m_l$ in the appropriate equation EFIE (2.29), MFIE (2.32), CFIE (2.33) or multilayered materials (2.37). Thus, the element Z_{kl} is given by one of the integrals

$$Z_{kl} = \begin{cases} R^s_\Omega\left(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{j}'_k, \mathbf{0}\}\right) & 1 \le k \le N^e \quad 1 \le l \le N^e \\ R^s_\Omega\left(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{0}, \mathbf{m}'_k\}\right) & N^e < k \le N \quad 1 \le l \le N^e \\ R^s_\Omega\left(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{j}'_k, \mathbf{0}\}\right) & 1 \le k \le N^e \quad N^e < l \le N \\ R^s_\Omega\left(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{0}, \mathbf{m}'_k\}\right) & N^e < k \le N \quad N^e < l \le N \end{cases} \tag{3.4}$$

in the case of a dielectric. Notice, that the integrals must be calculated for both the interior and the exterior domain. In the case of EFIE, MFIE or CFIE the element is given by

$$Z_{kl} = \alpha \int_\Gamma T\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma + (1-\alpha)\frac{\imath}{\kappa}\int_\Gamma \hat{\mathbf{n}} \times K\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma \tag{3.5}$$

Here, $\alpha = 1$ is the special case EFIE and $\alpha = 0$ is the special case MFIE.

The same applies to the element V_k defined by the equations

$$V_k = \begin{cases} V_\Omega\left(\{\mathbf{j}'_k, \mathbf{0}\}\right) & 1 \le k \le N^e \\ V_\Omega\left(\{\mathbf{0}, \mathbf{m}'_k\}\right) & N^e < k \le N \end{cases} \tag{3.6}$$

in the case of a dielectric material. In the case of perfect electric conductors it is given by

$$V_k = -\alpha\frac{1}{\imath\kappa Z}\int_\Gamma \mathbf{E}_a \cdot \mathbf{j}'_k d\Gamma + (1-\alpha)\frac{\imath}{\kappa}\int_\Gamma \hat{\mathbf{n}} \times \mathbf{H}_a \cdot \mathbf{j}'_k d\Gamma \tag{3.7}$$

For different applied electromagnetic fields the only difference is the right hand side $\mathbf{V}$. To find out the currents the unknown I_l is sought. It is I^e_l for $1 \le l \le N^e$ and I^m_l for $N^e + 1 \le l \le N^e + N^m$ in I. The matrix $\mathbf{Z}$ an impedance matrix.

The assembly of the matrix $\mathbf{Z}$ is carried out by calculating the contributions from basis functions associated with each triangle and adding the contributions to the impedance matrix.

3.2.4 Numerical integration

The simplest way to calculate the individual contributions from the basis functions is by numerical integration. Because of the singular nature of the integrals associated with integral equations, numerical integration is difficult for triangles close to each other. This problem can be handled in several ways. One way is to use analytical treatment of the singular integrals to extract the singularity. Another more general way is to transform the singular integrals using Duffy transformations that remove the singularity [15]. Either way, the problem usually ends up in an integral over a triangle.

To simplify the integration over triangles, simplex coordinates are introduced. A point P in a triangle is described by three simplex coordinates ξ_1, ξ_2 and ξ_3. Each coordinate is defined by

$$\xi_i = \frac{A_i}{A} \quad i = 1, 2, 3 \tag{3.8}$$

where A_i is the area of the triangle with nodes $i-1$, $i+1$ and P, as in Figure 3.3, and $A = A_1 + A_2 + A_3$ is the area of the triangle. The indices should be interpreted in a cyclic sense, i.e. index 4 is index 1. It follows that $\xi_1+\xi_2+\xi_3 = 1$, implying that one coordinate is linearly dependent on the other two. One can interpret simplex coordinate ξ_i as the height from node i to the point P.

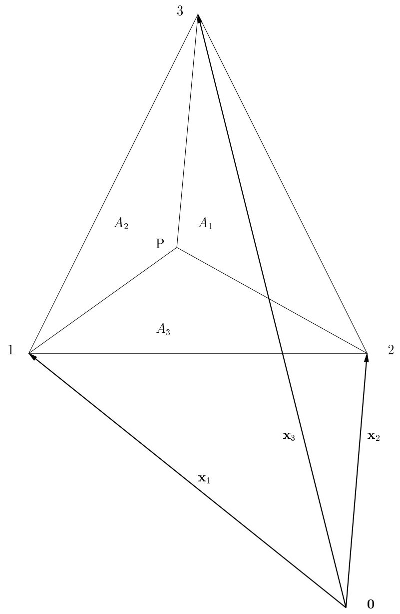

Figure 3.3: The simplex coordinates.

Simplex coordinates are used to transform the integration to a standard form. The global position vector $\mathbf{x}$ can be written in terms of ξ_1, ξ_2 and ξ_3 as

$$\mathbf{x} = \xi_1\mathbf{x}_1 + \xi_2\mathbf{x}_2 + \xi_3\mathbf{x}_3 \tag{3.9}$$

where $\mathbf{x}_i$ denotes the vector pointing from the origin to node i of the triangle.

When the simplex coordinates are used for integration, the integral is transformed using the formula

$$\int_{\triangle} I(\mathbf{x})\, d\Gamma(\mathbf{x}) = 2A \int_0^1 \int_0^{1-\xi_i} I(\xi_i, \xi_{i\pm1})\, d\xi_{i\pm} d\xi_i \tag{3.10}$$

A vector integral over a triangle is treated the same way using the fact that

$$\begin{aligned}\int_{\triangle} \mathbf{I}(\mathbf{x})\, d\Gamma(\mathbf{x}) = (\mathbf{x}_i - \mathbf{x}_{i-1}) \frac{1}{h_{i-1}} \int_{\triangle} \hat{\mathbf{h}}_{i-1} \cdot \mathbf{I}(\mathbf{x})\, d\Gamma(\mathbf{x}) \\ + (\mathbf{x}_i - \mathbf{x}_{i+1}) \frac{1}{h_{i+1}} \int_{\triangle} \hat{\mathbf{h}}_{i+1} \cdot \mathbf{I}(\mathbf{x})\, d\Gamma(\mathbf{x})\end{aligned} \tag{3.11}$$

Here, $\mathbf{h}_i$ is the height vector perpendicular to side i pointing from node i to side i, see Figure 3.4 and h_i is the length of $\mathbf{h}_i$. The two right-hand side integrals are scalar integrals and can be treated using the previous equation.

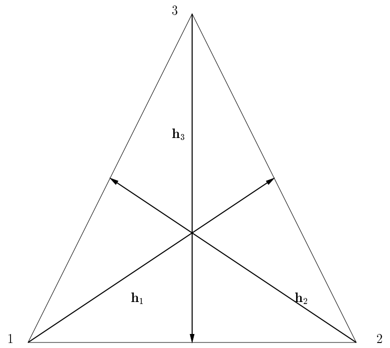

Figure 3.4: The height vectors used in integration of vector integrals.

The integrals are treated numerically using quadrature rules developed for triangles

$$\int_0^1 \int_0^{1-\xi_i} I(\xi_i, \xi_{i\pm1})\, d\xi_{i\pm} d\xi_i \approx \sum_{k=1}^{M} w_k I\left((\xi_i)_k, (\xi_{i\pm1})_k\right) \tag{3.12}$$

where w_k is the weight associated with point $\left((\xi_i)_k, (\xi_{i\pm1})_k\right)$ and M is the number of integration points.

3.3 Solution methods

The equation $\mathbf{ZI} = \mathbf{V}$ is an $N \times N$ dense system of linear equations with $\mathbf{Z}$ complex. In the case of Boundary Element Methods based on the Rumsey reaction the system is complex symmetric, that is $\mathbf{Z} = \mathbf{Z}^T$. To store the system in memory requires $\mathcal{O}\left(N^2\right)$ memory positions. The solution of an $N \times N$ system requires $\mathcal{O}\left(N^3\right)$ arithmethic operations for a direct method like Gaussian elimination. Instead one can use iterative methods like the Generalized Minimun RESidual method (GMRES) [22]. Iterative methods require $\mathcal{O}\left(N_{\text{iter}}N^2\right)$ arithmetic operations, where N_{iter} is the number of matrix vector multiplications required to meet some convergence criteria. The number of iterations depends on the condition number of the problem. Thus, for ill-conditioned problem an iterative method can be as slow as a direct method or worse. A good preconditioner can reduce the number of iterations.

For the integral equations it is usually the memory that limits the size of the problem one can solve. With about 10000 unknowns the storage requirement is larger than the memory size of most computers. One can resort to parallel computers to solve this problem, but with about 100000 unknowns the memory requirement is again prohibiting.

To solve larger problems one way is to use hybrid methods. In [14], a hybrid between a Physical Optics method and a Moment Method is used. With this method the memory scales as $\mathcal{O}\left(N_{\text{MM}}^2\right) + \mathcal{O}\left(N_{\text{PO}}\right)$, where N_{MM} is the number of unknowns in the Moment Method region and N_{PO} is the number of unknowns in the Physical Optics region. For Moment Method regions of small size, this approach works well.

The approach that is used here, is to use an iterative method together with an expansion of the Green's function resulting in a faster matrix vector multiplication needed in iterative Krylov methods. In this case the memory requirement is reduced to $\mathcal{O}\left(N \log N\right)$ and the work is reduced to $\mathcal{O}\left(N_{\text{iter}} N \log N\right)$ arithmetic operations. In order to reduce the number of iterations, a preconditioner that was introduced in [28] is employed together with a block iterative method for several right hand sides.

A parallel solver for boundary integral methods is described in [13]. It uses a dense block LDL^T-solver to factorize the impedance matrix. This means that it only works for Rumsey reaction formulations. A parallel iterative solver is described in [28]. It is based on a block symmetric matrix vector multiplication algorithm, which means that it only works for Rumsey reaction formulations. It uses the Quasi-Minimum Residual method (QMR) to solve the linear system and is preconditioned by a Sparse Approximate Inverse preconditioner (SPAI).

Chapter 4

Block Quasi-Minimum Residual method

In order to solve the discretized integral equations in Chapter 3 a block iterative method for several right hand sides is proposed. The reason for using a block method is that the change of excitation only changes the right-hand side for a given wavenumber.

Different versions of the Quasi-Minimum Residual method (QMR) have been studied extensively in [16, 17, 18, 19, 20]. Block versions of QMR are proposed for instance in [18, 38, 39]. In [18] the emphasis is on an update formula for one right hand side at a time to be able to use deflation efficiently. In [39] a block symmetric version of QMR is proposed and is extended in [38] to a version that requires the transpose of the matrix but can handle a preconditioner. The version here extends [39] to a QMR-method that can handle a symmetric preconditioner. The proof-technique is interestingly very similar to the one in [20] except that scalars are block-matrices and vectors are block vectors.

To construct the iterative method, a Block Lanczos method is proposed in Section 4.2. It requires the Gram-Schmidt method in Section 4.1 to work. The Block QMR method is constructed in Section 4.4. Besides the Block Lanczos method and Gram-Schmidt method it also uses a block QR-factorization method based on Givens transformations in Section 4.3.

4.1 A biorthogonal Gram-Schmidt procedure

In this section a modified Gram-Schmidt procedure is used to construct biorthogonal blocks of vectors, i.e. two sets of vectors $\mathbf{Q}$ and $\mathbf{V}$ such that $\mathbf{Q}^T\mathbf{V} = \mathbf{I}_s$. This procedure is used in Algorithm 4.2 for that purpose.

The modified Gram-Schmidt process is given in Algorithm 4.1. As input the algorithm takes a block vector $\tilde{\mathbf{V}}$ of size $n \times s$ and a complex symmetric matrix $\mathbf{M}$ of size $n \times n$. The result from the algorithm is $\mathbf{Q}$, $\mathbf{V}$, $\boldsymbol{\rho}$, $\boldsymbol{\psi}$ and $\boldsymbol{\beta}$. The vector $\mathbf{V}$ is an $n \times s$ vector that has the property $\mathbf{V}\boldsymbol{\rho}^T = \tilde{\mathbf{V}}$, where $\boldsymbol{\rho}$ is a lower triangular matrix. It is required that $\mathbf{Q}\boldsymbol{\psi} = \mathbf{M}\tilde{\mathbf{V}}$ and $\mathbf{Q}^T\mathbf{V} = \mathbf{I}_s$, where $\boldsymbol{\psi}$ is an upper triangular matrix. In order to satisfy $\mathbf{Q}^T\mathbf{V} = \mathbf{I}_s$ the vector $\mathbf{V}$ is required to be block $\mathbf{M}$-biorthogonal, which implies that $\mathbf{V}^T\mathbf{M}\mathbf{V} = \boldsymbol{\beta}^{-1}$, where $\boldsymbol{\beta}$ is diagonal. A Gram-Schmidt procedure modified for this purpose is used. The requirements imply that $\boldsymbol{\psi}^T = \boldsymbol{\psi}^T\mathbf{Q}^T\mathbf{V} = \boldsymbol{\rho}\mathbf{V}^T\mathbf{M}\mathbf{V} = \boldsymbol{\rho}\boldsymbol{\beta}^{-1}$. The proof of the validity of Algorithm 4.1 is by construction and is left out here.

Algorithm 4.1 A Gram-Schmidt process modified to handle the case of constructing $\mathbf{M}$-biorthogonal vectors.

Require: $\tilde{\mathbf{V}}$, $\mathbf{M}$
Ensure: $\mathbf{V}$, $\mathbf{Q}$, $\boldsymbol{\rho}$, $\boldsymbol{\psi}$, $\boldsymbol{\beta}$
  $\boldsymbol{\rho} = \mathbf{0}$, $\boldsymbol{\psi} = \mathbf{0}$, $\boldsymbol{\beta} = \mathbf{0}$
  $\mathbf{V} = \tilde{\mathbf{V}}$
  $\mathbf{Q} = \mathbf{M}\tilde{\mathbf{V}}$
  for $j = 1 : s - 1$ **do**
    $\delta_j = ||\mathbf{V}_{:,j}||_2$
    $\mathbf{V}_{:,j} = \frac{\mathbf{V}_{:,j}}{\delta_j}$
    $\mathbf{Q}_{:,j} = \frac{\mathbf{Q}_{:,j}}{\delta_j}$
    $\boldsymbol{\rho}_{j:s,j} = \mathbf{Q}^T_{:,j:s}\,\mathbf{V}_{:,j}$
    $\boldsymbol{\beta}_{j,j} = \boldsymbol{\rho}_{j,j}$
    $\boldsymbol{\psi}_{j,j+1:s} = \boldsymbol{\rho}_{j+1:s,j}$
    $\boldsymbol{\rho}_{j,j} = \delta_j\boldsymbol{\rho}_{j,j}$
    $\boldsymbol{\psi}_{j,j} = \delta_j$
    $\mathbf{V}_{:,j} = \frac{\mathbf{V}_{:,j}}{\boldsymbol{\beta}_{j,j}}$
    $\mathbf{V}_{:,j+1:s} = \mathbf{V}_{:,j+1:s} - \mathbf{V}_{:,j}\,\boldsymbol{\rho}^T_{j+1:s,j}$
    $\mathbf{Q}_{:,j+1:s} = \mathbf{Q}_{:,j+1:s} - \mathbf{Q}_{:,j}\,\boldsymbol{\psi}_{j,j+1:s}$
  end for
  $\delta_s = ||\mathbf{V}_{:,s}||_2$
  $\mathbf{V}_{:,s} = \frac{\mathbf{V}_{:,s}}{\delta_s}$
  $\mathbf{Q}_{:,s} = \frac{\mathbf{Q}_{:,s}}{\delta_s}$
  $\boldsymbol{\beta}_{s,s} = \mathbf{Q}^T_{:,s}\,\mathbf{V}_{:,s}$
  $\boldsymbol{\rho}_{s,s} = \delta_s\boldsymbol{\beta}_{s,s}$
  $\boldsymbol{\psi}_{s,s} = \delta_s$
  $\mathbf{V}_{:,s} = \frac{\mathbf{V}_{:,s}}{\boldsymbol{\beta}_{s,s}}$

4.2 The block Lanczos method

The idea of the block Lanczos method is to generate blocks of multi vectors of size $n \times s\,(k+1)$. Two blocks of multi vectors are generated from an $n \times n$ matrix $\mathbf{A}$. One of the blocks of multi vectors is denoted by $\hat{\mathbf{V}}_k = [\mathbf{V}_0\,\mathbf{V}_1\,\ldots\,\mathbf{V}_k]$ and the other one is denoted by $\hat{\mathbf{Q}}_k = [\mathbf{Q}_0\,\mathbf{Q}_1\,\ldots\,\mathbf{Q}_k]$. The blocks of multi vectors have the property $\mathbf{A}\hat{\mathbf{V}}_k = \hat{\mathbf{V}}_{k+1}\mathbf{T}_{k+1,k}$ and $\mathbf{A}^T\hat{\mathbf{Q}}_k = \hat{\mathbf{Q}}_{k+1}\hat{\boldsymbol{\beta}}_{k+1}^{-1}\mathbf{T}_{k+1,k}^T\hat{\boldsymbol{\beta}}_k$. The matrix $\mathbf{T}_{k+1,k}$ is a block tridiagonal $s\,(k+1) \times sk$-matrix, where the blocks are $s \times s$-matrices. The matrix $\hat{\boldsymbol{\beta}}_k$ is a diagonal $sk \times sk$ matrix. The multi vectors $\mathbf{V}_k$ and $\mathbf{Q}_j$ also have the property that they are mutually biorthogonal such that $\mathbf{Q}_j^T\mathbf{V}_k = \boldsymbol{\delta}_{jk}$. Here, $\boldsymbol{\delta}_{jk}$ denotes the $s \times s$ identity matrix $\mathbf{I}_s$ for $j = k$ and $\mathbf{0}$ otherwise. All the properties follow from the block Lanczos process.

The only demand on $\mathbf{Q}_k$ is that it fulfills the previously stated relationship. In some cases though it is fruitful to add extra demands on $\mathbf{Q}_k$, for instance when the Lanczos process is applied to the matrix $\mathbf{A} = \mathbf{ZM}$ where $\mathbf{Z}$ and $\mathbf{M}$ are complex symmetric. In this case a good choice for $\mathbf{Q}_k$ is $\mathbf{Q}_k = \mathbf{M}\mathbf{V}_k\boldsymbol{\beta}_k$, where $\boldsymbol{\beta}_k$ is a diagonal scaling factor. This choice allows $\hat{\mathbf{Q}}_k$ to be computed without computing $\mathbf{A}^T$.

An algorithm for the block Lanczos process is given in Algorithm 4.2. The function gram_M_sym performs a $\mathbf{M}$-biorthogonal Gram-Schmidt procedure according to Algorithm 4.1 and is explained in Section 4.1.

Algorithm 4.2 The Lanczos process for several right-hand sides.

Require: $\mathbf{B}$, $\mathbf{X}$, $\mathbf{Z}$ $\mathbf{M}$
Ensure: $\hat{\mathbf{Q}}_k\mathbf{ZM}\hat{\mathbf{V}}_k = \mathbf{T}_{k+1,k}$
  $\tilde{\mathbf{V}}_0 = \mathbf{B} - \mathbf{ZX}$
  $[\mathbf{V}_0, \boldsymbol{\rho}_0, \mathbf{Q}_0, \boldsymbol{\psi}_0, \boldsymbol{\beta}_0] = \text{gram\_M\_sym}(\tilde{\mathbf{V}}_0, \mathbf{M})$
  $\mathbf{P}_0 = \mathbf{Q}_0$
  for $k = 1 : maxiter$ **do**
    $\mathbf{L}_k = \mathbf{Z}\mathbf{P}_{k-1}$
    $\boldsymbol{\epsilon}_k = \mathbf{P}_{k-1}^T\mathbf{L}_k$
    $\tilde{\mathbf{V}}_k = \mathbf{L}_k - \mathbf{V}_{k-1}\boldsymbol{\epsilon}_k$
    $[\mathbf{V}_k, \boldsymbol{\rho}_k, \mathbf{Q}_k, \boldsymbol{\psi}_k, \boldsymbol{\beta}_k] = \text{gram\_M\_sym}(\tilde{\mathbf{V}}_k, \mathbf{M})$
    $\boldsymbol{\delta}_k = \boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k$
    $\mathbf{P}_k = \mathbf{Q}_k - \mathbf{P}_{k-1}\boldsymbol{\delta}_k$
  end for

Algorithm 4.2 allows one to transform the matrix $\mathbf{A}$ to a block tridiagonal matrix, since $\hat{\mathbf{Q}}_k\mathbf{ZM}\hat{\mathbf{V}}_k = \mathbf{T}_{k+1,k}$. This follows from the property $\mathbf{Q}_j^T\mathbf{V}_i = \boldsymbol{\delta}_{ji}$. This property is exploited in the Block QMR method in Section 4.4 to construct a short term recurrence formula for the solution of the equation $\mathbf{ZX} = \mathbf{B}$. To prove the property $\mathbf{Q}_j^T\mathbf{V}_i = \boldsymbol{\delta}_{ji}$ a few lemmas are needed. Note that $\mathbf{Q}_k^T\mathbf{V}_k = \mathbf{I}_s$ is assured by the Gram-Schmidt process. This is used in the proof of the following lemma

Lemma 4.2.1 *Suppose that* $\mathbf{Q}_j^T\mathbf{V}_i = \mathbf{0}$ *for all* $i \neq j \leq k$ *and that* $\mathbf{P}_j^T\mathbf{Z}\mathbf{P}_i = \mathbf{0}$ *for all* $i \neq j < k$. *Then it follows from Algorithm 4.2 that* $\mathbf{Q}_k^T\mathbf{Z}\mathbf{P}_{k-1} = \boldsymbol{\rho}_k^T$ *and that* $\mathbf{P}_k^T\mathbf{Z}\mathbf{P}_j = \mathbf{0}$ *for* $j < k$. *Thus* $\mathbf{P}_j^T\mathbf{Z}\mathbf{P}_i = \mathbf{0}$ *for all* $i \neq j \leq k$.

Proof The properties from Algorithm 4.1 and the construction of the vectors in Algorithm 4.2 are used together with the recursion formulas, yielding

$$\begin{aligned}
\mathbf{Q}_k^T\mathbf{Z}\mathbf{P}_{k-1} &= \mathbf{Q}_k^T\mathbf{L}_k = \mathbf{Q}_k^T\tilde{\mathbf{V}}_k + \mathbf{Q}_k^T\mathbf{V}_{k-1}\boldsymbol{\epsilon}_k && \\
&= \mathbf{Q}_k^T\mathbf{V}_k\boldsymbol{\rho}_k^T = \boldsymbol{\rho}_k^T && j = k-1 \\
\mathbf{P}_k^T\mathbf{Z}\mathbf{P}_{k-1} &= \mathbf{Q}_k^T\mathbf{Z}\mathbf{P}_{k-1} - \boldsymbol{\delta}_k^T\mathbf{P}_{k-1}^T\mathbf{Z}\mathbf{P}_{k-1} && \\
&= \boldsymbol{\rho}_k^T - \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\epsilon}_k = \mathbf{0} && j = k-1 \\
\mathbf{P}_k^T\mathbf{Z}\mathbf{P}_j &= \mathbf{Q}_k^T\mathbf{L}_{j+1} - \boldsymbol{\delta}_k^T\mathbf{P}_{k-1}^T\mathbf{Z}\mathbf{P}_j && \\
&= \mathbf{Q}_k^T\tilde{\mathbf{V}}_{j+1} + \mathbf{Q}_k^T\mathbf{V}_j\boldsymbol{\epsilon}_{j+1} = \mathbf{0} && j < k-1
\end{aligned}$$

Because of the choice of $\mathbf{Q}_0$ and $\mathbf{P}_0$ and the fact that $\mathbf{Z}$ is complex symmetric the lemma is proved.

Using lemma 4.2.1 it is now possible to prove the following properties for $\mathbf{V}_j$ and $\mathbf{Q}_i$.

Lemma 4.2.2 *Suppose that the assumptions in Lemma 4.2.1 holds. It follows that Algorithm 4.2 produces vectors* $\mathbf{V}_i$ *and* $\mathbf{Q}_j$ *such that* $\mathbf{V}_k^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k = \boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\beta}_k^{-1}\boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k$, $\mathbf{V}_{k-1}^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k = \boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k$ *and that* $\mathbf{V}_j^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k = \mathbf{0}$ *for* $j < k-1$.

Proof Again the properties follow from Algorithm 4.1 and the construction of the vectors in Algorithm 4.2. Together with Lemma 4.2.1 the use of recursion give

$$\begin{aligned}
\mathbf{V}_k^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k &= \boldsymbol{\beta}_k^{-1}\mathbf{Q}_k^T\mathbf{Z}\mathbf{Q}_k && \\
&= \boldsymbol{\beta}_k^{-1}\left(\mathbf{P}_k^T + \boldsymbol{\delta}_k^T\mathbf{P}_{k-1}^T\right)\mathbf{Z}\left(\mathbf{P}_k + \mathbf{P}_{k-1}\boldsymbol{\delta}_k\right) && j = k \\
&= \boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\beta}_k^{-1}\boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k && \\
\mathbf{V}_{k-1}^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k &= \boldsymbol{\beta}_{k-1}^{-1}\mathbf{Q}_{k-1}^T\mathbf{Z}\mathbf{Q}_k && \\
&= \boldsymbol{\beta}_{k-1}^{-1}\left(\mathbf{P}_{k-1}^T + \boldsymbol{\delta}_{k-1}^T\mathbf{P}_{k-2}^T\right)\mathbf{Z}\left(\mathbf{P}_k + \mathbf{P}_{k-1}\boldsymbol{\delta}_k\right) && j = k-1 \\
&= \boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k && \\
\mathbf{V}_j^T\mathbf{M}\mathbf{Z}\mathbf{Q}_k &= \boldsymbol{\beta}_j^{-1}\mathbf{Q}_j^T\mathbf{Z}\mathbf{Q}_k && \\
&= \boldsymbol{\beta}_j^{-1}\left(\mathbf{P}_j^T + \boldsymbol{\delta}_j^T\mathbf{P}_{j-1}^T\right)\mathbf{Z}\left(\mathbf{P}_k + \mathbf{P}_{k-1}\boldsymbol{\delta}_k\right) = \mathbf{0} && j < k-1
\end{aligned}$$

The choice of $\mathbf{V}_0$ and $\mathbf{Q}_0$ proves the lemma.

In order to simplify notation the vector $\tilde{\mathbf{Q}}_{k+1} = \mathbf{M}\tilde{\mathbf{V}}_{k+1}$ is introduced. It is used in the two following lemmas

Lemma 4.2.3 *Suppose the assumptions in Lemma 4.2.1 and Lemma 4.2.2 hold. Then Algorithm 4.2 give that* $\mathbf{V}_{k+1}\boldsymbol{\rho}_{k+1}^T = \mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) - \mathbf{V}_{k-1}\boldsymbol{\rho}_k$ *and the corresponding relation for* $\mathbf{Q}_{k+1}$ *is* $\mathbf{Q}_{k+1}\boldsymbol{\psi}_{k+1} = \mathbf{MZQ}_k - \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) - \mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k$

Proof Again using the properties of the Algorithm 4.1 and Algorithm 4.2 together with Lemma 4.2.1 and Lemma 4.2.2, it follows from recursion that

$$\begin{aligned}
\mathbf{V}_{k+1}\boldsymbol{\rho}_{k+1}^T &= \tilde{\mathbf{V}}_{k+1} = \mathbf{L}_{k+1} - \mathbf{V}_k\boldsymbol{\epsilon}_{k+1} \\
&= \mathbf{ZP}_k - \mathbf{V}_k\boldsymbol{\epsilon}_{k+1} \\
&= \mathbf{ZQ}_k - \mathbf{ZP}_{k-1}\boldsymbol{\delta}_k - \mathbf{V}_k\boldsymbol{\epsilon}_{k+1} \\
&= \mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{V}_k\boldsymbol{\epsilon}_{k+1} - \mathbf{L}_k\boldsymbol{\delta}_k \\
&= \mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{V}_k\boldsymbol{\epsilon}_{k+1} - \mathbf{V}_k\boldsymbol{\rho}_k^T\boldsymbol{\delta}_k - \mathbf{V}_{k-1}\boldsymbol{\epsilon}_k\boldsymbol{\delta}_k \\
&= \mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) - \mathbf{V}_{k-1}\boldsymbol{\rho}_k
\end{aligned}$$

In the same way one can prove that

$$\begin{aligned}
\mathbf{Q}_{k+1}\boldsymbol{\psi}_{k+1} &= \tilde{\mathbf{Q}}_{k+1} = \mathbf{M}\tilde{\mathbf{V}}_{k+1} \\
&= \mathbf{MZMV}_k\boldsymbol{\beta}_k - \mathbf{MV}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) - \mathbf{MV}_{k-1}\boldsymbol{\rho}_k \\
&= \mathbf{MZQ}_k - \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) - \mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k
\end{aligned}$$

Because of the choices of $\mathbf{V}_0$ and $\mathbf{Q}_0$ the lemma is proved.

Now the tools exist to prove Proposition 4.2.1. It implies that $\hat{\mathbf{Q}}_k\mathbf{ZM}\hat{\mathbf{V}}_k = \mathbf{T}_{k+1,k}$. The structure of the block tridiagonal matrix $\mathbf{T}_{k+1,k}$ is implied by the proof. It is found in Proposition 4.2.2.

Proposition 4.2.1 *Let the assumptions in Lemma 4.2.1, Lemma 4.2.2 and Lemma 4.2.3 hold. Furthermore, assume that* $\mathbf{V}_k$ *and* $\mathbf{Q}_k$ *are linearly independent and that they are biorthogonal. Then it follows from Algorithm 4.2 that* $\mathbf{Q}_j^T\mathbf{V}_i = \boldsymbol{\delta}_{ji}$ *for all* $i, j \leq k+1$.

Proof From the assumptions the matrices $\boldsymbol{\rho}_k^{-1}$ and $\boldsymbol{\beta}_k^{-1}$ exist. Algorithm 4.1 generates vectors such that $\mathbf{Q}_k^T\mathbf{V}_k = \mathbf{I}_s$. Using Algorithm 4.2 together with the lemmas it is proven that

$$\begin{aligned}
\mathbf{V}_k^T\tilde{\mathbf{Q}}_{k+1} =& \mathbf{V}_k^T\mathbf{MZQ}_k - \mathbf{V}_k^T\mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) \\
& - \mathbf{V}_k^T\mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k \\
=& \boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\beta}_k^{-1}\boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k - \boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1} && j = k \\
& - \boldsymbol{\beta}_k^{-1}\boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k = \mathbf{0} \\
\mathbf{Q}_k^T\tilde{\mathbf{V}}_{k+1} =& \mathbf{Q}_k^T\mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{Q}_k^T\mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) && j = k \\
& - \mathbf{Q}_k^T\mathbf{V}_{k-1}\boldsymbol{\rho}_k = \mathbf{0} \\
\mathbf{V}_{k-1}^T\tilde{\mathbf{Q}}_{k+1} =& \mathbf{V}_{k-1}^T\mathbf{MZQ}_k - \mathbf{V}_{k-1}^T\mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) \\
& - \mathbf{V}_{k-1}^T\mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k && j = k-1 \\
=& \boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k - \boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k = \mathbf{0} \\
\mathbf{Q}_{k-1}^T\tilde{\mathbf{V}}_{k+1} =& \mathbf{Q}_{k-1}^T\mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{Q}_{k-1}^T\mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) && j = k-1 \\
& - \mathbf{Q}_{k-1}^T\mathbf{V}_{k-1}\boldsymbol{\rho}_k = \mathbf{0} \\
\mathbf{V}_j^T\tilde{\mathbf{Q}}_{k+1} =& \mathbf{V}_j^T\mathbf{MZQ}_k - \mathbf{V}_j^T\mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) && j < k-1 \\
& - \mathbf{V}_j^T\mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k = \mathbf{0} \\
\mathbf{Q}_j^T\tilde{\mathbf{V}}_{k+1} =& \mathbf{Q}_j^T\mathbf{ZMV}_k\boldsymbol{\beta}_k - \mathbf{Q}_j^T\mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) && j < k-1 \\
& - \mathbf{Q}_j^T\mathbf{V}_{k-1}\boldsymbol{\rho}_k = \mathbf{0}
\end{aligned}$$

Since $\boldsymbol{\rho}_k^{-1}$ and $\boldsymbol{\beta}_k^{-1}$ exists and $\mathbf{Q}_j^T\mathbf{V}_i = \mathbf{0}$ for $i \neq j$ the proposition is proved.

In order to find the structure of $\mathbf{T}_{k+1,k}$ let $\hat{\mathbf{Q}}_k = [\mathbf{Q}_0\,\mathbf{Q}_1\,\ldots\,\mathbf{Q}_k]$ and $\hat{\mathbf{V}}_k = [\mathbf{V}_0\,\mathbf{V}_1\,\ldots\,\mathbf{V}_k]$. From the Gram-Schmidt procedure in Algorithm 4.1 $\boldsymbol{\rho}_k^T\boldsymbol{\psi}_k^{-1} = \boldsymbol{\beta}_k$. This equality together with Lemma 4.2.3 yields

Proposition 4.2.2 *The structure of the matrix* $\mathbf{T}_{k+1,k}$ *is*

$$\mathbf{T}_{k+1,k} = \begin{pmatrix}
\boldsymbol{\epsilon}_1\boldsymbol{\beta}_0^{-1} & \boldsymbol{\rho}_1\boldsymbol{\beta}_1^{-1} & \mathbf{0} & \cdots \\
\boldsymbol{\rho}_1^T\boldsymbol{\beta}_0^{-1} & \left(\boldsymbol{\epsilon}_2 + \boldsymbol{\rho}_1^T\boldsymbol{\epsilon}_1^{-1}\boldsymbol{\rho}_1\right)\boldsymbol{\beta}_1^{-1} & \boldsymbol{\rho}_2\boldsymbol{\beta}_2^{-1} & \ddots \\
\mathbf{0} & \boldsymbol{\rho}_2^T\boldsymbol{\beta}_1^{-1} & \left(\boldsymbol{\epsilon}_3 + \boldsymbol{\rho}_2^T\boldsymbol{\epsilon}_2^{-1}\boldsymbol{\rho}_2\right)\boldsymbol{\beta}_2^{-1} & \ddots \\
\mathbf{0} & \mathbf{0} & \boldsymbol{\rho}_3^T\boldsymbol{\beta}_2^{-1} & \ddots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots
\end{pmatrix} \tag{4.1}$$

It has the properties $\mathbf{ZM}\hat{\mathbf{V}}_k = \hat{\mathbf{V}}_{k+1}\mathbf{T}_{k+1,k}$ *and* $\mathbf{MZ}\hat{\mathbf{Q}}_k = \hat{\mathbf{Q}}_{k+1}\hat{\boldsymbol{\beta}}_{k+1}^{-1}\mathbf{T}_{k+1,k}\hat{\boldsymbol{\beta}}_k$.

Proof Lemma 4.2.3 give that

$$\begin{aligned}
\mathbf{ZMV}_k =& \mathbf{V}_{k-1}\boldsymbol{\rho}_k\boldsymbol{\beta}_k^{-1} + \mathbf{V}_k\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right)\boldsymbol{\beta}_k^{-1} + \mathbf{V}_{k+1}\boldsymbol{\rho}_{k+1}^T\boldsymbol{\beta}_k^{-1} \\
\mathbf{MZQ}_k =& \mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k + \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right) + \mathbf{Q}_{k+1}\boldsymbol{\psi}_{k+1} \\
=& \mathbf{Q}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\rho}_k\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k + \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\left(\boldsymbol{\epsilon}_{k+1} + \boldsymbol{\rho}_k^T\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\right)\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k \\
& + \mathbf{Q}_{k+1}\boldsymbol{\beta}_{k+1}^{-1}\boldsymbol{\rho}_{k+1}^T\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k
\end{aligned}$$

The proposition follows from the observation that $\hat{\mathbf{V}}_k = [\mathbf{V}_0\, \mathbf{V}_1 \ldots \mathbf{V}_k]$ and $\hat{\mathbf{Q}}_k = [\mathbf{Q}_0\, \mathbf{Q}_1 \ldots \mathbf{Q}_k]$ and carrying out the matrix vector multiplications.

The form of $\mathbf{T}_{k+1,k}$ can be used in an iterative scheme. It will generate a three-term recurrence formula. A more stable formula that is equivalent is two two-term recurrences. In this case they will generate an **LU**-factorization of **T**. For this purpose the vector $\mathbf{P}_k$ is examined together with an auxiliary vector $\mathbf{W}_k = \mathbf{M}^{-1}\mathbf{P}_k\boldsymbol{\beta}_k^{-1}$.

Proposition 4.2.3 *The vectors $\mathbf{P}_k$ and $\mathbf{W}_k$ are related to $\mathbf{Q}_k$ and $\mathbf{V}_k$ through the relations*

$$\begin{aligned} \hat{\mathbf{V}}_k &= \hat{\mathbf{W}}_k\mathbf{U}_{k,k} \\ \mathbf{ZM}\hat{\mathbf{W}}_k &= \hat{\mathbf{V}}_{k+1}\mathbf{L}_{k+1,k} \\ \hat{\mathbf{Q}}_k &= \hat{\mathbf{P}}_k\hat{\boldsymbol{\beta}}_k^{-1}\mathbf{U}_{k,k}\hat{\boldsymbol{\beta}}_k \\ \mathbf{MZ}\hat{\mathbf{P}}_k &= \hat{\mathbf{Q}}_{k+1}\hat{\boldsymbol{\beta}}_{k+1}^{-1}\mathbf{L}_{k+1,k}\hat{\boldsymbol{\beta}}_k \end{aligned} \tag{4.2}$$

where $\mathbf{U}_{k,k}$ is an upper bidiagonal matrix and has the structure

$$\mathbf{U}_{k,k} = \begin{pmatrix} \mathbf{I} & \boldsymbol{\beta}_0\boldsymbol{\epsilon}_1^{-1}\boldsymbol{\rho}_1\boldsymbol{\beta}_1^{-1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{I} & \boldsymbol{\beta}_1\boldsymbol{\epsilon}_2^{-1}\boldsymbol{\rho}_2\boldsymbol{\beta}_2^{-1} & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots \end{pmatrix} \tag{4.3}$$

$\mathbf{L}_{k+1,k}$ is a lower bidiagonal matrix with the structure

$$\mathbf{L}_{k+1,k} = \begin{pmatrix} \boldsymbol{\epsilon}_1\boldsymbol{\beta}_0^{-1} & \mathbf{0} & \mathbf{0} & \cdots \\ \boldsymbol{\rho}_1^T\boldsymbol{\beta}_0^{-1} & \boldsymbol{\epsilon}_2\boldsymbol{\beta}_1^{-1} & \mathbf{0} & \ddots \\ \mathbf{0} & \boldsymbol{\rho}_2^T\boldsymbol{\beta}_1^{-1} & \boldsymbol{\epsilon}_3\boldsymbol{\beta}_2^{-1} & \ddots \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\rho}_3^T\boldsymbol{\beta}_2^{-1} & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots \end{pmatrix} \tag{4.4}$$

*$\mathbf{L}_{k+1,k}$ and $\mathbf{U}_{k,k}$ are the **LU**-factors of $\mathbf{T}_{k+1,k}$, so that $\mathbf{T}_{k+1,k} = \mathbf{L}_{k+1,k}\mathbf{U}_{k,k}$.*

Proof Algorithm 4.2 gives

$$
\begin{aligned}
\mathbf{Q}_k &= \mathbf{P}_k + \mathbf{P}_{k-1}\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k = \mathbf{P}_k\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k + \mathbf{P}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\beta}_{k-1}\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k \\
\mathbf{MZP}_k &= \mathbf{ML}_{k+1} = \mathbf{M}\tilde{\mathbf{V}}_{k+1} + \mathbf{MV}_k\boldsymbol{\epsilon}_{k+1} = \mathbf{Q}_{k+1}\boldsymbol{\psi}_{k+1} + \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1} \\
&= \mathbf{Q}_{k+1}\boldsymbol{\beta}_{k+1}^{-1}\boldsymbol{\rho}_{k+1}^T\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k + \mathbf{Q}_k\boldsymbol{\beta}_k^{-1}\boldsymbol{\epsilon}_{k+1}\boldsymbol{\beta}_k^{-1}\boldsymbol{\beta}_k \\
\mathbf{V}_k &= \mathbf{M}^{-1}\mathbf{Q}_k\boldsymbol{\beta}_k^{-1} = \mathbf{M}^{-1}\mathbf{P}_k\boldsymbol{\beta}_k^{-1} + \mathbf{M}^{-1}\mathbf{P}_{k-1}\boldsymbol{\beta}_{k-1}^{-1}\boldsymbol{\beta}_{k-1}\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\boldsymbol{\beta}_k^{-1} \\
&= \mathbf{W}_k + \mathbf{W}_{k-1}\boldsymbol{\beta}_{k-1}\boldsymbol{\epsilon}_k^{-1}\boldsymbol{\rho}_k\boldsymbol{\beta}_k^{-1} \\
\mathbf{ZMW}_k &= \mathbf{ZMM}^{-1}\mathbf{P}_k\boldsymbol{\beta}_k^{-1} = \mathbf{ZP}_k\boldsymbol{\beta}_k^{-1} = \mathbf{L}_{k+1}\boldsymbol{\beta}_k^{-1} \\
&= \tilde{\mathbf{V}}_{k+1}\boldsymbol{\beta}_k^{-1} + \mathbf{V}_k\boldsymbol{\epsilon}_{k+1}\boldsymbol{\beta}_k^{-1} = \mathbf{V}_{k+1}\boldsymbol{\rho}_{k+1}^T\boldsymbol{\beta}_k^{-1} + \mathbf{V}_k\boldsymbol{\epsilon}_{k+1}\boldsymbol{\beta}_k^{-1}
\end{aligned}
$$

That $\mathbf{L}_{k+1,k}$ and $\mathbf{U}_{k,k}$ are the **LU**-factors of $\mathbf{T}_{k+1,k}$ is verified by multiplying them together proving the proposition.

In Section 4.4 the $\mathbf{L}_{k+1,k}$-matrix in the **LU**-factorization of $\mathbf{T}_{k+1,k}$ is used to find a simple recurrence formula for a minimization problem approximating the solution of $\mathbf{ZX} = \mathbf{B}$.

4.3 The block form of Givens rotations

For the block version of QMR, a method for solving minimization problems with small matrices is needed. Givens rotations can be used to calculate **QR**-factorizations of matrices with small bandwidths efficiently. A blocked version is described in [39]. It calculates

$$
\begin{pmatrix} \boldsymbol{\delta} \\ \boldsymbol{\chi} \end{pmatrix} = \mathbf{G}^H \begin{pmatrix} \tilde{\boldsymbol{\delta}} \\ 0 \end{pmatrix} \tag{4.5}
$$

and is repeated in Algorithm 4.3 for completeness. It is assumed that $\boldsymbol{\delta}$, $\tilde{\boldsymbol{\delta}}$ and $\boldsymbol{\chi}$ are upper triangular $s \times s$ matrices and $\mathbf{G}$ is a $2s \times 2s$ orthonormal matrix.

4.4 Block-QMR

The Block-QMR algorithm is an iterative algorithm for solving systems of equations with several right-hand sides. A complex symmetric version with right preconditioning is given in Algorithm 4.4. The problem $\mathbf{ZMM}^{-1}\mathbf{X} = \mathbf{B}$ is solved, where the matrices $\mathbf{Z}$ and $\mathbf{M}$ are complex symmetric.

The idea of the complex symmetric Block-QMR algorithm is to calculate new vectors of the form $\mathbf{X}_k = \mathbf{X}_0 + \hat{\mathbf{P}}_k\hat{\mathbf{y}}_k$ where $\hat{\mathbf{P}}_k = [\mathbf{P}_0\,\mathbf{P}_1\,\ldots\,\mathbf{P}_k]$ comes from Algorithm 4.2 and $\hat{\mathbf{y}} = [\mathbf{y}_0\,\mathbf{y}_1\,\ldots\,\mathbf{y}_k]^T$.

The **LU**-factorization of $\mathbf{T}_{k+1,k}$ is used to generate an update formula for $\mathbf{X}_k$. The update formula finds a minimum of the residual. Because of the fact

Algorithm 4.3 Givens rotations for blocked matrices

Require: $\boldsymbol{\delta}$, $\boldsymbol{\chi}$, $\boldsymbol{\tau}$

Ensure: $\mathbf{G}$, $\mathbf{t}$, $\tilde{\boldsymbol{\tau}}$, $\tilde{\boldsymbol{\delta}}$

$\mathbf{E}_{1:s,1:s} = \boldsymbol{\delta}$
$\mathbf{E}_{s+1:2s,1:s} = \boldsymbol{\chi}$
$\mathbf{T}_{1:s,1:s} = \boldsymbol{\tau}$
$\mathbf{G} = \mathbf{I}$
for $i = 1 : s-1$ **do**
  for $j = i+s : -1 : i+1$ **do**
    $[c, s] = \mathrm{rot}\left(\mathbf{E}_{j-1,i}, \mathbf{E}_{j,i}\right)$
    $\mathbf{y} = c\mathbf{E}_{j-1,i:s} + s\mathbf{E}_{j,i:s}$
    $\mathbf{E}_{j,i:s} = -s^*\mathbf{E}_{j-1,i:s} + c\mathbf{E}_{j,i:s}$
    $\mathbf{E}_{j-1,i:s} = \mathbf{y}$
    $\mathbf{y} = c\mathbf{T}_{j-1,1:s} + s\mathbf{T}_{j,1:s}$
    $\mathbf{T}_{j,1:s} = -s^*\mathbf{T}_{j-1,1:s} + c\mathbf{T}_{j,1:s}$
    $\mathbf{T}_{j-1:1:s} = \mathbf{y}$
    $\mathbf{y} = c\mathbf{G}_{j-1,1:2s} + s\mathbf{G}_{j,1:2s}$
    $\mathbf{G}_{j,1:2s} = -s^*\mathbf{G}_{j-1,1:2s} + c\mathbf{G}_{j,1:2s}$
    $\mathbf{G}_{j-1,1:2s} = \mathbf{y}$
  end for
end for
$\mathbf{t} = \mathbf{T}_{1:s,1:s}$
$\tilde{\boldsymbol{\tau}} = \mathbf{T}_{s+1:2s,1:s}$
$\tilde{\boldsymbol{\delta}} = \mathbf{E}_{1:s,1:s}$

that $\hat{\mathbf{V}}_k$ and $\hat{\mathbf{Q}}_k$ are biorthogonal and not orthogonal vectors, the minimum is a quasi-minimum. Hence, the name Block Quasi-Minimum Residual Method. The residual is computed from Proposition 4.4.1.

Proposition 4.4.1 *Let* $\mathbf{X}_k = \mathbf{X}_0 + \hat{\mathbf{P}}_k\hat{\mathbf{y}}_k$, *where* $\hat{\mathbf{P}}_k = [\mathbf{P}_0\,\mathbf{P}_1\,\ldots\,\mathbf{P}_k]$. *Here,* $\mathbf{P}_k$ *is defined by Algorithm 4.2. The residual* $\mathbf{R}_k = \mathbf{B} - \mathbf{Z}\mathbf{X}_k$ *is then given by* $\mathbf{R}_k = \hat{\mathbf{V}}_{k+1}\hat{\boldsymbol{\beta}}_{k+1}\left(\mathbf{E}_{k+1}\boldsymbol{\psi}_0 - \mathbf{L}'_{k+1,k}\hat{\mathbf{y}}_k\right)$, *where* $\mathbf{E}_{k+1} = [\mathbf{I}_s\,\mathbf{0}\,\ldots\,\mathbf{0}]^T$ *and the structure of* $\mathbf{L}'_{k+1,k}$ *is*

$$\mathbf{L}'_{k+1,k} = \begin{pmatrix} \boldsymbol{\beta}_0^{-1}\boldsymbol{\epsilon}_1 & \mathbf{0} & \mathbf{0} & \cdots \\ \boldsymbol{\psi}_1 & \boldsymbol{\beta}_1^{-1}\boldsymbol{\epsilon}_2 & \mathbf{0} & \ddots \\ \mathbf{0} & \boldsymbol{\psi}_2 & \boldsymbol{\beta}_2^{-1}\boldsymbol{\epsilon}_3 & \ddots \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\psi}_3 & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots \end{pmatrix} \tag{4.6}$$

Algorithm 4.4 The Block-QMR algorithm for several right-hand sides.

Require: $\mathbf{B}$, $\mathbf{X}$, $\mathbf{Z}$ $\mathbf{M}$
Ensure: $\mathbf{X} \approx \mathbf{Z}^{-1}\mathbf{B}$
  $\tilde{\mathbf{V}}_0 = \mathbf{B} - \mathbf{Z}\mathbf{X}$
  $\mathbf{X}_0 = \mathbf{X}$, $\mathbf{R}_0 = \tilde{\mathbf{V}}_0$
  $\mathbf{D}_0 = \mathbf{0}$, $\mathbf{U}_0 = \mathbf{0}$
  $\mathbf{G}_0 = \mathbf{I}$
  $[\mathbf{V}_0, \boldsymbol{\rho}_0, \mathbf{Q}_0, \boldsymbol{\psi}_0, \boldsymbol{\beta}_0] = \text{gram\_M\_sym}(\tilde{\mathbf{V}}_0, \mathbf{M})$
  $\mathbf{P}_0 = \mathbf{Q}_0$
  $\tilde{\boldsymbol{\tau}}_0 = \boldsymbol{\psi}_0$
  for $k = 1 : maxiter$ **do**
    $\mathbf{L}_k = \mathbf{Z}\mathbf{P}_{k-1}$
    $\boldsymbol{\epsilon}_k = \mathbf{P}_{k-1}^T \mathbf{L}_k$
    $\boldsymbol{\phi}_k = \boldsymbol{\beta}_{k-1}^{-1} \boldsymbol{\epsilon}_k$
    $\tilde{\mathbf{V}}_k = \mathbf{L}_k - \mathbf{V}_{k-1}\boldsymbol{\epsilon}_k$
    $[\mathbf{V}_k, \boldsymbol{\rho}_k, \mathbf{Q}_k, \boldsymbol{\psi}_k, \boldsymbol{\beta}_k] = \text{gram\_M\_sym}(\tilde{\mathbf{V}}_k, \mathbf{M})$
    $\boldsymbol{\zeta}_k = \mathbf{G}_{k-1}(1 : s, s+1 : 2s)\, \boldsymbol{\phi}_k$
    $\boldsymbol{\chi}_k = \mathbf{G}_{k-1}(s+1 : 2s, s+1 : 2s)\, \boldsymbol{\phi}_k$
    $[\mathbf{G}_k, \tilde{\boldsymbol{\tau}}_k, \boldsymbol{\tau}_{k-1}, \boldsymbol{\gamma}_k] = \text{givens}(\boldsymbol{\chi}_k, \boldsymbol{\psi}_k, \tilde{\boldsymbol{\tau}}_{k-1})$
    $\mathbf{D}_k = (\mathbf{P}_{k-1} - \mathbf{D}_{k-1}\boldsymbol{\zeta}_k)\, \boldsymbol{\gamma}_k^{-1}$
    $\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{D}_k \boldsymbol{\tau}_{k-1}$
    $\mathbf{U}_k = (\mathbf{L}_k - \mathbf{U}_{k-1}\boldsymbol{\zeta}_k)\, \boldsymbol{\gamma}_k^{-1}$
    $\mathbf{R}_k = \mathbf{R}_{k-1} - \mathbf{U}_k \boldsymbol{\tau}_{k-1}$
    if Convergence is achieved **then**
      $\mathbf{X} = \mathbf{X}_k$
      Break
    end if
    $\boldsymbol{\delta}_k = \boldsymbol{\epsilon}_k^{-1} \boldsymbol{\rho}_k$
    $\mathbf{P}_k = \mathbf{Q}_k - \mathbf{P}_{k-1}\boldsymbol{\delta}_k$
  end for

Proof From Algorithm 4.2 the residual is given by

$$\begin{aligned}
\mathbf{R}_k &= \mathbf{B} - \mathbf{Z}\mathbf{X}_k = \mathbf{B} - \mathbf{Z}\mathbf{X}_0 - \mathbf{Z}\hat{\mathbf{P}}_k \hat{\mathbf{y}}_k \\
&= \mathbf{R}_0 - \mathbf{Z}\hat{\mathbf{P}}_k \hat{\boldsymbol{\beta}}_k^{-1} \hat{\boldsymbol{\beta}}_k \hat{\mathbf{y}}_k \\
&= \mathbf{R}_0 - \hat{\mathbf{V}}_{k+1} \mathbf{L}_{k+1,k} \hat{\boldsymbol{\beta}}_k \hat{\mathbf{y}}_k \\
&= \hat{\mathbf{V}}_{k+1} \hat{\boldsymbol{\beta}}_{k+1} \left(\hat{\boldsymbol{\beta}}_{k+1}^{-1} \mathbf{E}_{k+1} \boldsymbol{\rho}_0^T - \hat{\boldsymbol{\beta}}_{k+1}^{-1} \mathbf{L}_{k+1,k} \hat{\boldsymbol{\beta}}_k \hat{\mathbf{y}}_k \right) \\
&= \hat{\mathbf{V}}_{k+1} \hat{\boldsymbol{\beta}}_{k+1} \left(\mathbf{E}_{k+1} \boldsymbol{\psi}_0 - \mathbf{L}'_{k+1,k} \hat{\mathbf{y}}_k \right)
\end{aligned}$$

which proves the proposition.

Notice that $\|\mathbf{R}_k\| \leq \left\|\hat{\mathbf{V}}_{k+1}\hat{\boldsymbol{\beta}}_{k+1}\right\| \left\|\left(\mathbf{E}_{k+1}\boldsymbol{\psi}_0 - \mathbf{L}'_{k+1,k}\hat{\mathbf{y}}_k\right)\right\|$. Since nothing can be done about the first part of this expression, $\hat{\mathbf{y}}_k$ is chosen to minimize $\left\|\left(\mathbf{E}_{k+1}\boldsymbol{\psi}_0 - \mathbf{L}'_{k+1,k}\hat{\mathbf{y}}_k\right)\right\|$. If $\mathbf{G}_k^*\tilde{\mathbf{R}}_k = \mathbf{L}'_{k+1,k}$ then $\hat{\mathbf{y}}_k = \tilde{\mathbf{R}}_k^{-1}\mathbf{G}_k\mathbf{E}_{k+1}\boldsymbol{\psi}_0$. From this, using the procedure described in [20, 39], a simple update formula of the form $\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{D}_k\boldsymbol{\tau}_{k-1}$ is found. One can also find a similar update formula for the residual $\mathbf{R}_k$ given by $\mathbf{R}_k = \mathbf{R}_{k-1} - \mathbf{U}_k\boldsymbol{\tau}_{k-1}$ [4].

The approach used to define the Block Quasi-Minimum Residual method first finds an appropriate Lanczos algorithm. The Lanczos algorithm generates a block tridiagonal matrix which one can compute an **LU**-decomposition of. The lower triangular matrix of the **LU**-decomposition is then used to find a simple minimization problem. The minimization problem is solved by a **QR**-decomposition which one can update by a recursion formula. In this way the approximate solution $\mathbf{X}_k$ is calculated.

Chapter 5

Fast Multipole Method

The Fast Multipole Method for electromagnetics is a method that reduces the solution time of Moment Method problems. It was first proposed in [36] for three dimensional problems and later it has been used in for instance [7, 9, 40].

In the Fast Multipole Method certain properties of the underlying integral equation is used to perform the matrix vector multiplication faster than $\mathcal{O}\left(N^2\right)$. Using a multilevel technique [40] the solution time of an iterative method is reduced to $\mathcal{O}\left(N_{iter}N\log N\right)$ and the memory requirement is reduced to $\mathcal{O}\left(N\log N\right)$.

First the Fast Multipole Method is presented and then extended to a multilevel version. Algorithms for both cases are presented. We then turn to implementation issues. Several methods for reducing the constant in front of the scaling are presented. The most important contributions by the author is the reduction of the memory requirements due to certain properties of plane wave vectors in Section 5.9.3. Finally, a simple parallelization strategy on shared memory machines is presented in Section 5.10. Part of this work was presented in [29, 30].

5.1 The Fast Multipole method

The Fast Multipole method for Helmholtz' equation is based on two key observations.

The first observation is that the Green's function for Helmholtz' equation can be expanded into an infinite series using Gegenbauer's addition theorem [3]. It states that

$$
\begin{aligned}
\frac{e^{\imath\kappa|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} &= \imath\kappa h_0^{(1)}\left(\kappa\left|\mathbf{X}+\mathbf{d}\right|\right) \\
&= \imath\kappa\sum_{l=0}^{\infty}(-1)^l\,(2l+1)\,j_l\left(\kappa d\right)h_l^{(1)}\left(\kappa X\right)P_l\left(\hat{\mathbf{d}}\cdot\hat{\mathbf{X}}\right)
\end{aligned}
\tag{5.1}
$$

where $j_l\left(x\right)$ is a spherical Bessel function, $h_l^{(1)}\left(x\right)$ is a spherical Hankel function and $P_l\left(x\right)$ is a Legendre polynomial.

The expansion is valid as long as $d < X$. It can be used to compute the field at a receiver point $\mathbf{x}$ from a source $\mathbf{x}'$. Notice that

$$
\begin{aligned}
\mathbf{x}-\mathbf{x}' &= \mathbf{x}-\mathbf{X}_m+\mathbf{X}_m-\mathbf{X}'_m+\mathbf{X}'_m-\mathbf{x}' \\
&= \left(\mathbf{X}_m-\mathbf{X}'_m\right)+\left(\mathbf{x}-\mathbf{X}_m+\mathbf{X}'_m-\mathbf{x}'\right)
\end{aligned}
\tag{5.2}
$$

as in Figure 5.1. If $\mathbf{X}_m$ is close enough to $\mathbf{x}$ and $\mathbf{X}'_m$ is close enough to $\mathbf{x}'$, then the choice $\mathbf{d} = \mathbf{x}-\mathbf{X}_m+\mathbf{X}'_m-\mathbf{x}'$ and $\mathbf{X} = \mathbf{X}_m-\mathbf{X}'_m$ implies that $d < X$ since $|\mathbf{x}-\mathbf{X}_m+\mathbf{X}'_m-\mathbf{x}'| < |\mathbf{X}_m-\mathbf{X}'_m|$. Thus, the expansion in equation (5.1) is valid.

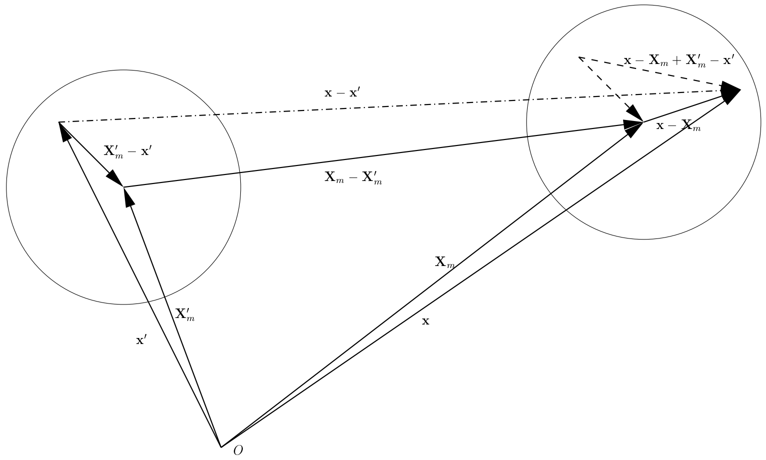

Figure 5.1: A graphical view of $\mathbf{x}-\mathbf{x}' = \left(\mathbf{X}_m-\mathbf{X}'_m\right)+\left(\mathbf{x}-\mathbf{X}_m+\mathbf{X}'_m-\mathbf{x}'\right)$.

The second key observation is that the product $j_l\left(\kappa d\right)P_l\left(\hat{\mathbf{d}}\cdot\hat{\mathbf{X}}\right)$ can be expanded into an integral over propagating plane waves [8]

$$
4\pi\imath^l j_l\left(\kappa d\right)P_l\left(\hat{\mathbf{d}}\cdot\hat{\mathbf{X}}\right) = \int_{\mathcal{S}} e^{\imath\boldsymbol{\kappa}\cdot\mathbf{d}}P_l\left(\hat{\boldsymbol{\kappa}}\cdot\hat{\mathbf{X}}\right)d^2\hat{\boldsymbol{\kappa}}
\tag{5.3}
$$

Here, the integral is taken over the unit sphere $\mathcal{S}$.

Inserting equation (5.3) into equation (5.1) yields

$$\frac{e^{\imath\kappa|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} = \frac{\imath\kappa}{4\pi}\sum_{l=0}^{\infty}\imath^{l}(2l+1)\,h_l^{(1)}(\kappa X)\int_{\mathcal{S}} e^{\imath\boldsymbol{\kappa}\cdot\mathbf{d}}P_l\left(\hat{\boldsymbol{\kappa}}\cdot\hat{\mathbf{X}}\right)d^2\hat{\boldsymbol{\kappa}} \tag{5.4}$$

Since the sum in (5.4) is infinite it is not possible to exchange the order of integration and summation. It turns out that only a finite number $L+1$ terms are needed in the sum to get a desired accuracy in an approximation of the Green's function. Once the number of terms is determined the order of summation and integration is changed

$$\frac{e^{\imath\kappa|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} \approx \frac{\imath\kappa}{4\pi}\int_{\mathcal{S}} e^{\imath\boldsymbol{\kappa}\cdot\mathbf{d}}\sum_{l=0}^{L}\imath^{l}(2l+1)\,h_l^{(1)}(\kappa X)\,P_l\left(\hat{\boldsymbol{\kappa}}\cdot\hat{\mathbf{X}}\right)d^2\hat{\boldsymbol{\kappa}} \tag{5.5}$$

The integral in equation (5.5) is possible to calculate with an appropriately chosen quadrature formula

$$\frac{e^{\imath\kappa|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} \approx \frac{\imath\kappa}{4\pi}\sum_{k=1}^{K} w_k e^{\imath\boldsymbol{\kappa}_k\cdot\mathbf{d}}\sum_{l=0}^{L}\imath^{l}(2l+1)\,h_l^{(1)}(\kappa X)\,P_l\left(\hat{\boldsymbol{\kappa}}_k\cdot\hat{\mathbf{X}}\right) \tag{5.6}$$

where w_k is the quadrature weight associated with direction $\hat{\boldsymbol{\kappa}}_k$.

In order to shorten notation the translation operator

$$\mathcal{T}_k^L(\kappa,\mathbf{X}) = \frac{\imath\kappa}{16\pi^2}w_k\sum_{l=0}^{L}\imath^{l}(2l+1)\,h_l^{(1)}(\kappa X)\,P_l\left(\hat{\boldsymbol{\kappa}}_k\cdot\hat{\mathbf{X}}\right) \tag{5.7}$$

is introduced. Then it follows that

$$\frac{e^{\imath\kappa|\mathbf{X}+\mathbf{d}|}}{4\pi|\mathbf{X}+\mathbf{d}|} \approx \sum_{k=1}^{K} e^{\imath\boldsymbol{\kappa}_k\cdot\mathbf{d}}\mathcal{T}_k^L(\kappa,\mathbf{X}) \tag{5.8}$$

From the previous discussion $\mathbf{X} = \mathbf{X}_m - \mathbf{X}'_m$ and $\mathbf{d} = \mathbf{x} - \mathbf{X}_m + \mathbf{X}'_m - \mathbf{x}'$. Since $e^{\imath\boldsymbol{\kappa}_k\cdot\mathbf{d}} = e^{\imath\boldsymbol{\kappa}_k\cdot\left(\mathbf{x}-\mathbf{X}_m+\mathbf{X}'_m-\mathbf{x}'\right)} = e^{\imath\boldsymbol{\kappa}_k\cdot(\mathbf{x}-\mathbf{X}_m)}e^{\imath\boldsymbol{\kappa}_k\cdot\left(\mathbf{X}'_m-\mathbf{x}'\right)}$, equation (5.8) yields

$$\frac{e^{\imath\kappa|\mathbf{x}-\mathbf{x}'|}}{4\pi|\mathbf{x}-\mathbf{x}'|} \approx \sum_{k=1}^{K} e^{\imath\boldsymbol{\kappa}_k\cdot(\mathbf{x}-\mathbf{X}_m)}\mathcal{T}_k^L(\kappa,\mathbf{X}_m-\mathbf{X}'_m)\,e^{\imath\boldsymbol{\kappa}_k\cdot\left(\mathbf{X}'_m-\mathbf{x}'\right)} \tag{5.9}$$

Notice that equation (5.9) is true for any source close to $\mathbf{X}'_m$ and any receiver close to $\mathbf{X}_m$ and that $\mathcal{T}_k^L(\kappa,\mathbf{X}_m-\mathbf{X}'_m)$ is independent of the locations of sources and receivers, since it depends on $\mathbf{X}_m$ and $\mathbf{X}'_m$ only.

5.2 Numerical integration on a sphere

In order to calculate the integrals in equation (5.5) a suitable quadrature rule has to be used. A simple way is to use a uniform distribution of points in ϕ and Gauss-Legendre points in θ.

The integrands can be written in terms of spherical harmonics, defined by

$$Y_l^m(\theta,\phi) = (-1)^m \sqrt{\tfrac{2l+1}{4\pi}\tfrac{(l-m)!}{(l+m)!}} e^{\imath m\phi} P_l^m(\cos\theta), \quad -l \le m \le l, \quad 0 \le l \tag{5.10}$$

Here, P_l^m are the associated Legendre functions defined by Rodrigues' formula [3]

$$P_l^m(x) = \tfrac{1}{2^l l!}\left(1-x^2\right)^{\frac{m}{2}} \tfrac{d^{m+l}}{dx^{m+l}}\left(x^2-1\right)^l, \quad -l \le m \le l, \quad 0 \le l \tag{5.11}$$

The special case $P_l^0(x) = P_l(x)$ is the Legendre function.

On the surface of the unit sphere S the integral is

$$\int_S C_{lm} e^{\imath m\phi} P_l^m(\cos\theta)\sin\theta d\theta d\phi = C_{lm}\int_{\phi=0}^{2\pi} e^{\imath m\phi}\int_{x=-1}^{1} P_l^m(x)\,dxd\phi \tag{5.12}$$

For $m \neq 0$ the integral over ϕ is calculated exactly by the periodic trapezoidal rule based on $L+1$ points for the first $Y_l^m(\theta,\phi)$ such that $-l \le m \le l$ and $0 \le l \le L$. The proof is simple. Choose $\phi_i = \frac{2\pi}{L+1}(i-1)$, then

$$\sum_{i=1}^{L+1}\frac{2\pi}{L+1}e^{\imath m\phi_i} = \frac{2\pi}{L+1}\sum_{i=0}^{L} e^{\imath m\frac{2\pi}{L+1}i} = \frac{2\pi}{L+1}\frac{e^{\imath m 2\pi}-1}{e^{\imath m\frac{2\pi}{L+1}}-1} \tag{5.13}$$

which is 0 for $0 < m < L+1$.

The special case $m = 0$ is handled by choosing Gauss-Legendre points in the θ-direction. Since $P_l(x)$ is a polynomial it can be integrated exactly with a suitable Gauss-Legendre rule. The first $2N-1$-order polynomials are integrated exactly by an N-point Gauss-Legendre rule. The Gauss-Legendre rule approximates an integral by

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{N}\tilde{w}_i f(x_i) \tag{5.14}$$

Here, $\tilde{w}_i$ is the weight associated with node x_i. The nodes x_i are the N zeros of the Legendre polynomial $P_N(x)$ and the weights are given by

$$\tilde{w}_i = \frac{2}{(1-x_i)\left(P_N'(x_i)\right)^2} \tag{5.15}$$

Thus, the nodes in the θ-direction are chosen as $\theta_i = \arccos x_i$, where x_i care defined by the Gauss-Legendre rule integrating the polynomials $0 \leq l \leq L$ exactly.

In the case of the Fast Multipole Method and equation (5.5) the first $2L$ spherical harmonics have to be integrated exactly [8, 36]. Thus, the choice of $2L+1$ nodes in the ϕ direction and $L+1$ nodes in the θ direction is sufficient. For reasons that will be clear later the choice of $2L+2$ nodes in ϕ is made. Notice that this choice implies that if ϕ is a quadrature point then $\phi+\pi$ is also a quadrature point. From the distribution of the Gauss-Legendre quadrature points it follows that if a θ between 0 and $\frac{\pi}{2}$ is a quadrature point, then $\pi-\theta$ is also a quadrature point.

The integral over the unit sphere is then approximated by

$$\int_S f(\theta,\phi)\, d\theta d\phi \approx \sum_{k=1}^{L+1} \sum_{j=1}^{2L+2} w_{jk} f(\theta_k, \phi_j) \tag{5.16}$$

where θ_k is derived from the Gauss-Legendre rule and ϕ_j from the trapezoidal rule. The weight $w_{j,k}$ is $\frac{2\pi}{2L+2}\tilde{w}_k$.

5.3 Error control

In order to control the error ϵ in equation (5.5) one must decide where to truncate the infinite series. There exist several criteria for that. One of them is a semi-empirical formula [7, 8] where $L = \kappa d + \alpha \log(\pi + \kappa d)$. Here, $\alpha = -\log_{10}(\epsilon)$ is the number of digits required. Another formula, which is the one mainly used here, is $L = \kappa d + \beta (\kappa d)^{\frac{1}{3}}$ [7], where $\beta = 1.8 (-\log_{10}(\epsilon))^{\frac{2}{3}}$ determines the number of digits.

The spherical Hankel functions $h_l^{(1)}(x)$ becomes highly oscillatory for $l \to \infty$ and x fixed. This means that the operator $\mathcal{T}_k^L(\kappa, \mathbf{X})$ becomes numerically unstable for large values of L. A solution to the problem is to use the exact expression for the Green's function for small values of $x = \kappa X$ and use the approximate expression for large values. Depending on the smallest size of x the size of L is also limited. Thus, there is a lower bound for the precision in equation (5.5) that depends on the smallest size of x.

5.4 The Fast Multipole Method applied to CFIE

The principle for the Fast Multipole Method can be applied to the EFIE, MFIE, and the CFIE. Since $\alpha = 0$ in CFIE is the MFIE and $\alpha = 1$ in CFIE is the EFIE, only the CFIE is analyzed here. Remember that expanding the current $\mathbf{J}$ in basis functions gives $\mathbf{J}(\mathbf{x}) = \sum_{l=1}^{N^e} I_l^e \mathbf{j}_l(\mathbf{x})$. Inserting this in the CFIE results in a dense linear system $\mathbf{ZI} = \mathbf{V}$ where

$$
\begin{cases}
\begin{aligned}
Z_{kl} =& \alpha \int_\Gamma T\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma \\
& + (1-\alpha)\frac{\imath}{\kappa}\left(\frac{1}{2}\int_\Gamma \mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma + \int_\Gamma \hat{\mathbf{n}} \times K\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma\right)
\end{aligned} \\
V_k = -\alpha \frac{1}{\imath\kappa Z}\int_\Gamma \mathbf{E}_a \cdot \mathbf{j}'_k d\Gamma + (1-\alpha)\frac{\imath}{\kappa}\int_\Gamma \hat{\mathbf{n}} \times \mathbf{H}_a \cdot \mathbf{j}'_k d\Gamma \\
I_l = I_l^e
\end{cases}
\tag{5.17}
$$

The part of the integrals where the Fast Multipole Method is used is the part where the support for $\mathbf{j}_k$ and $\mathbf{j}_l$ are different. This means that $\int_\Gamma \mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma = 0$.

Recalling the definitions of $T\mathbf{J}$ and $K\mathbf{J}$, together with equation (5.9), the result is a new expression for Z_{kl}

$$
\begin{aligned}
Z_{kl} =& \alpha \int_\Gamma T\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma + (1-\alpha)\frac{\imath}{\kappa}\int_\Gamma \hat{\mathbf{n}} \times K\mathbf{j}_l \cdot \mathbf{j}'_k d\Gamma \\
=& -\alpha \int_\Gamma G(\mathbf{x},\mathbf{x}')\frac{1}{\kappa^2}\boldsymbol{\nabla}_\Gamma \cdot \mathbf{j}_l(\mathbf{x}')\boldsymbol{\nabla}_\Gamma \cdot \mathbf{j}'_k(\mathbf{x})\, d\Gamma(\mathbf{x}')\, d\Gamma(\mathbf{x}) \\
& + \alpha \int_\Gamma G(\mathbf{x},\mathbf{x}')\mathbf{j}_l(\mathbf{x}')\mathbf{j}'_k(\mathbf{x})\, d\Gamma(\mathbf{x}')\, d\Gamma(\mathbf{x}) \\
& + (1-\alpha)\frac{\imath}{\kappa}\int_\Gamma \hat{\mathbf{n}} \times \int_\Gamma \boldsymbol{\nabla}_{\mathbf{x}'}G(\mathbf{x},\mathbf{x}') \times \mathbf{j}_l(\mathbf{x}') \cdot \mathbf{j}'_k d\Gamma(\mathbf{x}')\, d\Gamma\mathbf{x} \\
\approx& \sum_{j=1}^{K} \mathbf{R}_k(\hat{\boldsymbol{\kappa}}_j) \cdot \mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}'_m)\,\mathbf{F}_l(\hat{\boldsymbol{\kappa}}_j)
\end{aligned}
\tag{5.18}
$$

Here, $\mathbf{F}_l(\hat{\boldsymbol{\kappa}})$ and $\mathbf{R}_k(\hat{\boldsymbol{\kappa}})$ are defined by

$$
\begin{cases}
\mathbf{F}_l(\hat{\boldsymbol{\kappa}}) = \hat{\boldsymbol{\kappa}} \times \int_\Gamma e^{\imath\boldsymbol{\kappa}\cdot(\mathbf{X}'_m - \mathbf{x}')}\mathbf{j}_l d\Gamma(\mathbf{x}') \times \hat{\boldsymbol{\kappa}} \\
\begin{aligned}
\mathbf{R}_k(\hat{\boldsymbol{\kappa}}) =& \alpha\hat{\boldsymbol{\kappa}} \times \int_\Gamma e^{\imath\boldsymbol{\kappa}\cdot(\mathbf{x}-\mathbf{X}_m)}\mathbf{j}'_k d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}} \\
& + (1-\alpha)\int_\Gamma e^{\imath\boldsymbol{\kappa}\cdot(\mathbf{x}-\mathbf{X}_m)}\mathbf{j}'_k \times \hat{\mathbf{n}}(\mathbf{x})\, d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}}
\end{aligned}
\end{cases}
\tag{5.19}
$$

which should be compared with the expressions that are derived in Section 2.3. Because of the support of the basis functions the integrals are local, since they are only evaluated over two adjacent triangles. The integrals are calculated by numerical quadrature described in Section 3.2.4. An important point is that $\mathbf{F}_l$ depends on θ and ϕ only, since $\hat{\boldsymbol{\kappa}} \cdot \mathbf{F}_l = 0$. For reasons that will become clear later $\mathbf{F}$ is called the far field pattern and $\mathbf{R}$ is called the receiving pattern.

5.5 The Fast Multipole Method applied to dielectrics

The Fast Multipole Method can also be applied to dielectric materials. As before the currents are expanded in basis functions given by

$$\begin{cases} \mathbf{J}(\mathbf{x}) = \sum_{l=1}^{N^e} I_l^e \mathbf{j}_l(\mathbf{x}) \\ \mathbf{M}(\mathbf{x}) = \sum_{l=N^e+1}^{N^e+N^m} I_l^m \mathbf{m}_l(\mathbf{x}) \end{cases} \tag{5.20}$$

The element Z_{kl} in the dense linear system $\mathbf{ZI} = \mathbf{V}$ is then given by

$$Z_{kl} = \begin{cases} R_\Omega^s(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{j}'_k, \mathbf{0}\}) & 1 \le k \le N^e & 1 \le l \le N^e \\ R_\Omega^s(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{0}, \mathbf{m}'_k\}) & N^e < k \le N & 1 \le l \le N^e \\ R_\Omega^s(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{j}'_k, \mathbf{0}\}) & 1 \le k \le N^e & N^e < l \le N \\ R_\Omega^s(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{0}, \mathbf{m}'_k\}) & N^e < k \le N & N^e < l \le N \end{cases} \tag{5.21}$$

where the integral is taken in the interior and exterior domain see Section 3.2.3. Inserting equation (5.9) yields

$$\begin{cases} R_\Omega^s(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{j}'_k, \mathbf{0}\}) \approx \sum_{j=1}^K \mathbf{R}_{\mathbf{j}_k}^T(\hat{\boldsymbol{\kappa}}_j) \cdot \mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}'_m) \mathbf{F}_{\mathbf{j}_l}(\hat{\boldsymbol{\kappa}}_j) \\ R_\Omega^s(\{\mathbf{j}_l, \mathbf{0}\}, \{\mathbf{0}, \mathbf{m}'_k\}) \approx \sum_{j=1}^K \mathbf{R}_{\mathbf{m}_k}^K(\hat{\boldsymbol{\kappa}}_j) \cdot \mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}'_m) \mathbf{F}_{\mathbf{j}_l}(\hat{\boldsymbol{\kappa}}_j) \\ R_\Omega^s(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{j}'_k, \mathbf{0}\}) \approx \sum_{j=1}^K \mathbf{R}_{\mathbf{j}_k}^K(\hat{\boldsymbol{\kappa}}_j) \cdot \mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}'_m) \mathbf{F}_{\mathbf{m}_l}(\hat{\boldsymbol{\kappa}}_j) \\ R_\Omega^s(\{\mathbf{0}, \mathbf{m}_l\}, \{\mathbf{0}, \mathbf{m}'_k\}) \approx \sum_{j=1}^K \mathbf{R}_{\mathbf{m}_k}^T(\hat{\boldsymbol{\kappa}}_j) \cdot \mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}'_m) \mathbf{F}_{\mathbf{m}_l}(\hat{\boldsymbol{\kappa}}_j) \end{cases} \tag{5.22}$$

Here, $\mathbf{F}$ is the same operator as in Section 5.4. The operators $\mathbf{R}_{\mathbf{j}_k}^T$, $\mathbf{R}_{\mathbf{m}_k}^K$, $\mathbf{R}_{\mathbf{j}_k}^K$ and $\mathbf{R}_{\mathbf{m}_k}^T$ are given by

$$\begin{cases} \mathbf{R}_{\mathbf{j}_k}^T(\hat{\boldsymbol{\kappa}}) = \hat{\boldsymbol{\kappa}} \times \int_\Gamma e^{\imath \boldsymbol{\kappa} \cdot (\mathbf{x} - \mathbf{X}_m)} \mathbf{j}'_k d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}} \\ \mathbf{R}_{\mathbf{m}_k}^K(\hat{\boldsymbol{\kappa}}_j) = \imath\kappa \int_\Gamma e^{\imath \boldsymbol{\kappa} \cdot (\mathbf{x} - \mathbf{X}_m)} \mathbf{m}'_k(\mathbf{x}) d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}} \\ \mathbf{R}_{\mathbf{j}_k}^K(\hat{\boldsymbol{\kappa}}_j) = \imath\kappa \int_\Gamma e^{\imath \boldsymbol{\kappa} \cdot (\mathbf{x} - \mathbf{X}_m)} \mathbf{j}'_k(\mathbf{x}) d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}} \\ \mathbf{R}_{\mathbf{m}_k}^T(\hat{\boldsymbol{\kappa}}) = \hat{\boldsymbol{\kappa}} \times \int_\Gamma e^{\imath \boldsymbol{\kappa} \cdot (\mathbf{x} - \mathbf{X}_m)} \mathbf{m}'_k d\Gamma(\mathbf{x}) \times \hat{\boldsymbol{\kappa}} \end{cases} \tag{5.23}$$

To calculate the entries in $\mathbf{Z}$ the expressions are used on both the interior and the exterior of the object.

5.6 A fast matrix vector product

Only using the approximations in the Fast Multipole Method will not result in a faster matrix vector multiplication. This is easy to verify since it would mean that each entry involved an integral over the unit sphere in addition to the integrals on the triangles.

The acceleration of a matrix vector product is due to the fact that several basis functions close to each other can use the same translation operator $\mathcal{T}_k^L(\kappa, \mathbf{X})$ to transfer their contributions to another area holding other basis functions. In order to do this in a systematic way the object is divided into a number of boxes.

A simple way to cluster the basis functions together is to create a tree structure with boxes that are associated with the position of the basis functions in space. This can be achieved by enclosing the object in a box. In each direction x, y and z the sides are divided in two resulting in 8 smaller boxes. Each of the boxes are then divided again recursively until the size of the smallest boxes is the required size as in Figure 5.2. Only the boxes with basis functions inside them are kept, so the tree structure is sparse.

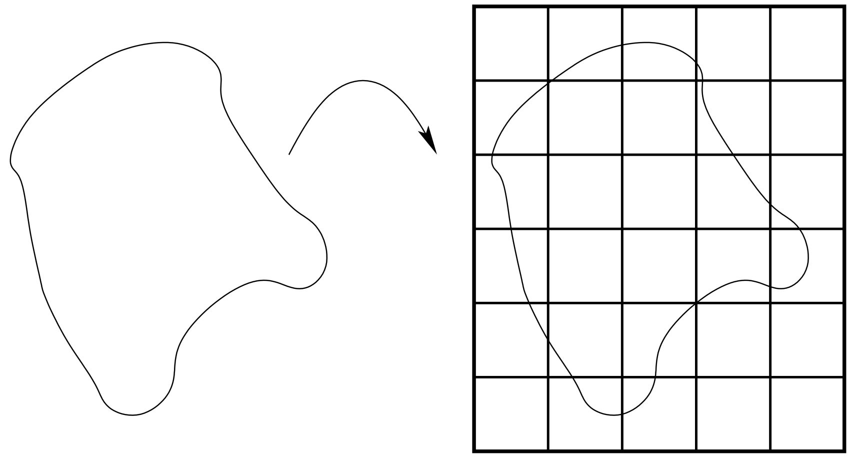

Figure 5.2: The object is enclosed in a box which is divided into boxes of required size.

Another way is to select a size of the smallest boxes and assign a number to each box based on its position. With a suitable numbering of the boxes, like Morton ordering [43], it is possible to calculate the father of a box. In this way the tree is constructed from the bottom and up. This is a better way of creating the boxes, because it is very easy to generate a list of pointers to the boxes at each level. An operation that involves all the boxes at a level is then performed by looping over the list of pointers and do the operation for each box as in Figure 5.3.

Assume that a list with all boxes containing basis functions exists. The boxes are denoted $\mathcal{B}_m$ for $1 \leq m \leq M$, where $\mathcal{B}_m$ is box number m in the list and M is the total number of boxes. When it is possible the interactions between boxes are calculated with the Fast Multipole Method. When the boxes are close it is

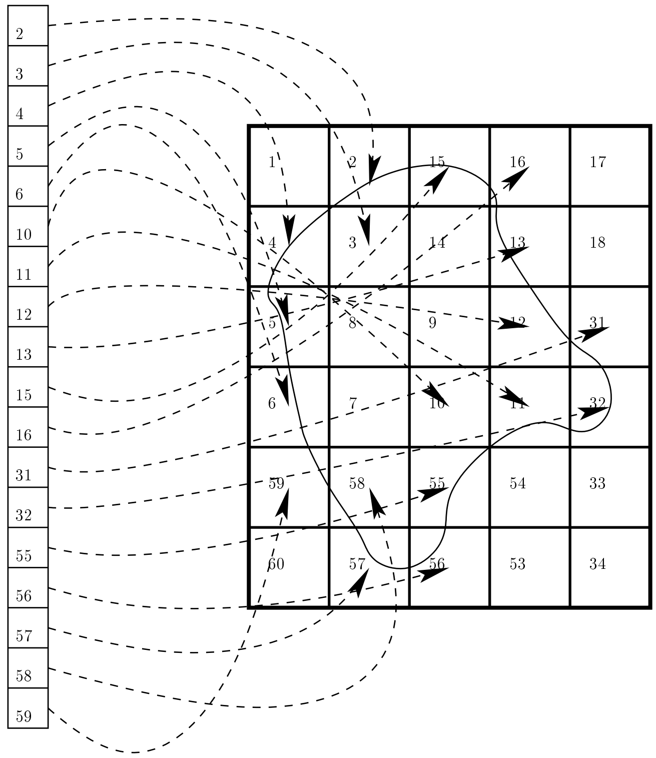

Figure 5.3: The boxes containing basis functions are stored in a list of pointers.

not possible to use the Fast Multipole Method and direct integration is used.

The procedure for computing the fast matrix vector product is divided into several steps. The first part handles initializations. For each box $\mathcal{B}_m$ two interaction lists are generated as in Figure 5.4. The first list $\mathcal{I}_m^{\mathcal{N}}$ contains the nearest neighbor boxes where direct integration is utilized. The second list $\mathcal{I}_m^{\mathcal{F}}$ contains the boxes that can use the Fast Multipole Method.

For box $\mathcal{B}_m$, the entries in the matrix $\mathbf{Z}$ corresponding to boxes in $\mathcal{I}_m^{\mathcal{N}}$ are computed and stored in memory. This part of the matrix is denoted $\mathbf{Z}^{near}$. The matrix $\mathbf{Z}^{near}$ is defined as

$$Z_{kl}^{near} = \begin{cases} Z_{kl} & k \in \mathcal{B}_m \quad \text{and} \quad l \in \mathcal{B}_n \quad \text{and} \quad \mathcal{B}_n \in \mathcal{I}_m^{\mathcal{N}} \\ 0 & \text{otherwise} \end{cases} \tag{5.24}$$

The Fast Multipole Method needs three initialization steps. In the first step the far field pattern of each box is generated. This is achieved by computing $\mathbf{F}_l(\hat{\boldsymbol{\kappa}}_j)$ for each basis function $\mathbf{j}_l$ and $\mathbf{m}_l$ in the box and each quadrature point $\hat{\boldsymbol{\kappa}}_j$ on the sphere. Since $\mathbf{F}_l(\hat{\boldsymbol{\kappa}}_j)$ has two components, a total of $2NK$ entries

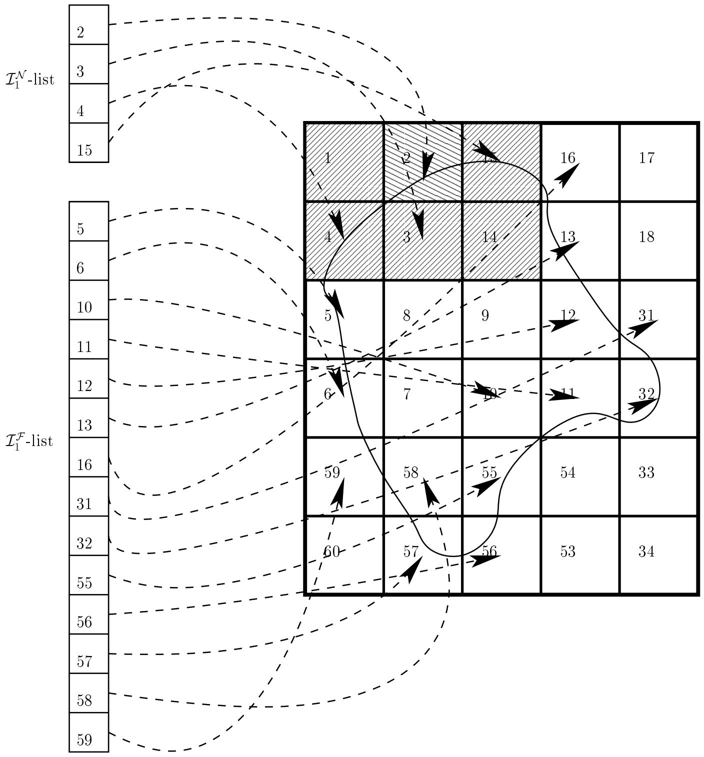

Figure 5.4: The two interaction lists $\mathcal{I}_1^{\mathcal{N}}$ and $\mathcal{I}_1^{\mathcal{F}}$ for box $\mathcal{B}_1$ which is box 2, the first box with basis functions.

have to be computed. Box $\mathcal{B}_m$ has $2M_mK$ entries, where M_m is the number of basis functions in that box. In the second step all the translation operators $\mathcal{T}_j^L(\kappa, \mathbf{X}_m - \mathbf{X}_n)$ needed by a box are computed and stored. Here, $\mathbf{X}_m$ denotes the geometrical center of box $\mathcal{B}_m$ and $\mathbf{X}_n$ the geometrical center of $\mathcal{B}_n$. Each box $\mathcal{B}_m$ has to know the translation operators for all boxes in $\mathcal{I}_m^{\mathcal{F}}$. The number of entries that have to be computed for box $\mathcal{B}_m$ is $N_{\mathcal{I}_m^{\mathcal{F}}}K$, where $N_{\mathcal{I}_m^{\mathcal{F}}}$ is the number of boxes in $\mathcal{I}_m^{\mathcal{F}}$. The third step is to compute the receiving patterns $\mathbf{R}_k(\hat{\boldsymbol{\kappa}}_j)$ for each basis function. This also requires a total of $2NK$ entries and, as for the far field pattern, box $\mathcal{B}_m$ has $2M_mK$ entries.

It is now possible to compute $\sum_{l=1}^{N} Z_{kl}I_l$, where basis function $k \in \mathcal{B}_m$, in an efficient manner since

$$\sum_{l=1}^{N} Z_{kl} I_l \approx \sum_{l \in \mathcal{B}_j \in \mathcal{I}_m^{\mathcal{N}}} Z_{kl}^{near} I_l + \sum_{j=1}^{K} \mathbf{R}_k\left(\hat{\boldsymbol{\kappa}}_j\right) \cdot \sum_{\mathcal{B}_n \in \mathcal{I}_m^{\mathcal{F}}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right) \sum_{l \in \mathcal{B}_n} \mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_j\right) I_l \tag{5.25}$$

The sum $\sum_{\mathcal{B}_n \in \mathcal{I}_m^{\mathcal{F}}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right) \sum_{l \in \mathcal{B}_n} \mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_j\right) I_l$ is the same for all basis functions in box $\mathcal{B}_m$ and $\sum_{l \in \mathcal{B}_n} \mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_j\right) I_l$ is independent of the receiving box $\mathcal{B}_m$. Thus, they are calculated only once. The resulting algorithm for the single-stage Fast Multipole Method is given in Algorithm 5.1.

Algorithm 5.1 The single-stage Fast Multipole Method matrix vector multiplication.

Require: The appropriate initializations and the vector $\mathbf{I}$
Ensure: $\mathbf{V} = \mathbf{ZI}$
  for $m = 1 : M$ **do**
    Find box $\mathcal{B}_m$
    for $k \in \mathcal{B}_m$ **do**
      Compute $V_k = \sum_{l \in \mathcal{B}_j \in \mathcal{I}_m^{\mathcal{N}}} Z_{kl}^{near} I_l$
    end for
  end for
  for $m = 1 : M$ **do**
    Find box $\mathcal{B}_m$
    for $j = 1 : K$ **do**
      Compute and store $\tilde{\mathbf{F}}_m\left(\hat{\boldsymbol{\kappa}}_j\right) = \sum_{l \in \mathcal{B}_m} \mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_j\right) I_l$
    end for
  end for
  for $m = 1 : M$ **do**
    Find box $\mathcal{B}_m$
    for $j = 1 : K$ **do**
      Compute $\mathbf{G}_m\left(\hat{\boldsymbol{\kappa}}_j\right) = \sum_{\mathcal{B}_n \in \mathcal{I}_m^{\mathcal{F}}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right) \tilde{\mathbf{F}}_n\left(\hat{\boldsymbol{\kappa}}_j\right)$
    end for
  end for
  for $m = 1 : M$ **do**
    Find box $\mathcal{B}_m$
    for $k \in \mathcal{B}_m$ **do**
      Compute $V_k = V_k + \sum_{j=1}^{K} \mathbf{R}_k\left(\hat{\boldsymbol{\kappa}}_j\right) \cdot \mathbf{G}_m\left(\hat{\boldsymbol{\kappa}}_j\right)$
    end for
  end for

Some analysis of Algorithm 5.1 [8, 36] shows that the computation of the matrix vector requires $aNM + bN^2M^{-1}$ arithmetic operations. The reason is that the first step requires $\mathcal{O}\left(N^2M^{-1}\right)$ operations, the second and the last step requires $\mathcal{O}\left(KN\right)$ operations and the third step requires $\mathcal{O}\left(KM^2\right)$ operations. Since, $K = \mathcal{O}\left(L^2\right)$ and $L = \mathcal{O}\left(\kappa d\right)$ and because of geometrical considerations $\kappa d = \mathcal{O}\left(\sqrt{NM^{-1}}\right)$, the conclusion is that $K = \mathcal{O}\left(NM^{-1}\right)$. The total operation count is minimized if $M = \sqrt{bNa^{-1}}$, where a and b are machine and implementation dependent parameters. The result is an $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ algorithm.

5.7 The multilevel Fast Multipole Method

The strategy that allows a fast matrix vector multiplication in the Fast Multipole Method can be extended to an $\mathcal{O}\left(N \log N\right)$ method. Consider the multilevel tree structure. For a box $\mathcal{B}_{m\lambda}$ at a given level λ there are some boxes that the box has to interact with directly. The other boxes could be interacted with using the Fast Multipole Method. Now consider the father of the box $\mathcal{B}_{m\lambda}$ denoted by $\mathcal{B}_{n(\lambda-1)}$ at a lower level $\lambda - 1$. That box also has some boxes which it has to interact with directly, while the other boxes can be interacted with using the Fast Multipole Method. Since, the boxes at the lower level are larger there are fewer boxes to interact with. Because the box is twice as large it requires about four times more integration points on the sphere to obtain the same precision in the approximation of the Green's function as the higher level. On the other hand, there are about four times fewer boxes since the body is a surface and only boxes with basis functions in them are considered. However, the fact that the box $\mathcal{B}_{n(\lambda-1)}$ has to interact directly with another box does not mean that the sons of the other box have to interact directly with the box $\mathcal{B}_{m\lambda}$.

The idea of the multilevel Fast Multipole Method is to compute only the far interactions at the level where they are considered to be near interactions at the level below as in Figure 5.5. As before a tree structure is constructed, but in this case the box size at the highest level is chosen as small as possible. At level λ the boxes are numbered $\mathcal{B}_{m\lambda}$ for $1 \leq m \leq M_\lambda$, where M_λ is the number of boxes on level λ. For each box $\mathcal{B}_{m\lambda}$, three lists are constructed. The first list is the nearest neighbor list $\mathcal{I}^{\mathcal{N}}_{m\lambda}$, which is only used at the highest level after construction of the tree. The second list $\mathcal{I}^{\mathcal{F}}_{m\lambda}$ contains the boxes which can use the Fast Multipole Method on level λ but are considered to be near neighbor interactions on level $\lambda - 1$. The third list $\mathcal{I}^{\mathcal{S}}_{m\lambda}$ contains the sons of the box and is not used on the highest level.

For each box $\mathcal{B}_{m\lambda_{max}}$ on the highest level λ_{max} the near interactions are computed and stored in $\mathbf{Z}^{near}$, which is defined as before. The far field pattern $\mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)$ for each basis function $\mathbf{j}_l$ and $\mathbf{m}_l$ is also computed for each quadrature point $\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}$ on the sphere, where $1 \leq j_{\lambda_{max}} \leq K_{\lambda_{max}}$ and $K_{\lambda_{max}}$ is the number of required integration points on the highest level. In the same way the receiving pattern $\mathbf{R}_k\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)$ is computed.

For all boxes $\mathcal{B}_{m\lambda}$ the translation operators $\mathcal{T}^L_j\left(\kappa, \mathbf{X}_{m\lambda} - \mathbf{X}_{n\lambda}\right)$ needed are

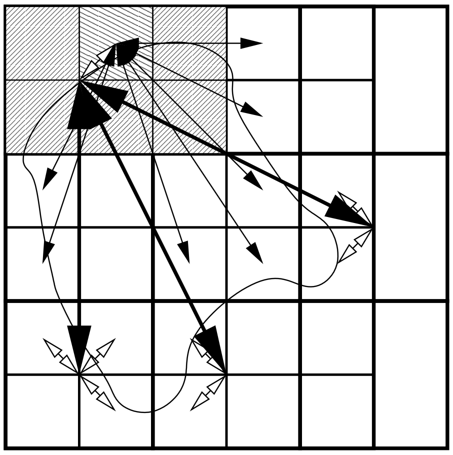

Figure 5.5: The principle behind the multilevel interactions. Here, the interactions are shown for one box. The nearest neighbors of the box have their interactions computed exactly.

computed and stored.

It is now possible to compute the far field patterns $\tilde{\mathbf{F}}_{m\lambda_{max}}\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)$ defined in Algorithm 5.1 of the boxes on the highest level. To compute the far field patterns on the lower levels in an efficient way, they should be computed from the information of the far field of the higher level. Since there are fewer samples on the higher level than are required on the lower levels, the low level far field patterns are computed through interpolation. In the same way the receiving pattern is computed. Consider the box $\mathcal{B}_{m\lambda}$ on level $\lambda < \lambda_{max}$. To compute

$$\sum_{j=1}^{K_\lambda} \mathbf{R}_{k\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) \cdot \sum_{\mathcal{B}_{n\lambda} \in \mathcal{I}^{\mathcal{F}}_{m\lambda}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right) \tilde{\mathbf{F}}_{n\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) \tag{5.26}$$

the far field patterns $\tilde{\mathbf{F}}_{n\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)$ of all the boxes in the far field list must be known. Once the far field patterns are known on the level $\lambda+1$ they can be interpolated and translated to the center of the father box on level λ.

The operator $\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)$ is used to denote an interpolation operator that interpolates the discrete far field values on level $\lambda+1$ to $\hat{\boldsymbol{\kappa}}_{j_\lambda}$ on level λ. The key observation is that $\tilde{\mathbf{F}}_{n\lambda}\left(\hat{\boldsymbol{\kappa}}\right) = e^{\imath\boldsymbol{\kappa}\cdot\left(\mathbf{X}_{n\lambda}-\mathbf{X}_{m(\lambda+1)}\right)}\tilde{\mathbf{F}}_{m(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)$ for the box $\mathcal{B}_{m(\lambda+1)}$. Thus, if $\tilde{\mathbf{F}}_{m(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)$ is known for $1 \le j_\lambda \le K_\lambda$ it is easy to compute the far field for that part of the father box, since the computation only involves a multiplication with an exponential, called a translation. Using the interpolation operator one can write $\tilde{\mathbf{F}}_{n\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) = e^{\imath\boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{n\lambda}-\mathbf{X}_{m(\lambda+1)}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\tilde{\mathbf{F}}_{m(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)$ for $1 \le j_\lambda \le K_\lambda$. The far field pattern of box $\mathcal{B}_{n\lambda}$ is then obtained by summing the 8 contributions from the son boxes.

In the same way the receiving pattern can be interpolated from the equivalent expression $\mathbf{R}_{k\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) = e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{m(\lambda+1)}-\mathbf{X}_{n\lambda}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\mathbf{R}_{k(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)$ for $1 \leq j_\lambda \leq K_\lambda$. However, this expression is not used to compute the expression (5.26). If it was computed in this way the receiving pattern would have to be known on each level. Instead the following observation is made

$$
\begin{aligned}
&\sum_{j=1}^{K_\lambda}\mathbf{R}_{k\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\cdot\sum_{\mathcal{B}_{n\lambda}\in\mathcal{I}^{\mathcal{F}}_{m\lambda}}\mathcal{T}^L_j\left(\kappa,\mathbf{X}_m-\mathbf{X}_n\right)\tilde{\mathbf{F}}_{n\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\\
&\quad=\sum_{j=1}^{K_\lambda}\left(e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{m(\lambda+1)}-\mathbf{X}_{n\lambda}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\mathbf{R}_{k(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)\right)\\
&\qquad\cdot\sum_{\mathcal{B}_{n\lambda}\in\mathcal{I}^{\mathcal{F}}_{m\lambda}}\mathcal{T}^L_j\left(\kappa,\mathbf{X}_m-\mathbf{X}_n\right)\\
&\qquad\left(\sum_{\mathcal{B}_{o(\lambda+1)}\in\mathcal{I}^{\mathcal{S}}_{n\lambda}}e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{n\lambda}-\mathbf{X}_{o(\lambda+1)}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\tilde{\mathbf{F}}_{o(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)\right) \qquad (5.27)\\
&\quad=\sum_{j=1}^{K_{(\lambda+1)}}\mathbf{R}_{k(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)\\
&\qquad\cdot\left(\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\right)^T e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{m(\lambda+1)}-\mathbf{X}_{n\lambda}\right)}\sum_{\mathcal{B}_{n\lambda}\in\mathcal{I}^{\mathcal{F}}_{m\lambda}}\mathcal{T}^L_j\left(\kappa,\mathbf{X}_m-\mathbf{X}_n\right)\\
&\qquad\left(\sum_{\mathcal{B}_{o(\lambda+1)}\in\mathcal{I}^{\mathcal{S}}_{n\lambda}}e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{n\lambda}-\mathbf{X}_{o(\lambda+1)}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\tilde{\mathbf{F}}_{o(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)\right)
\end{aligned}
$$

where the last equality follows from the fact that the interpolation operator is a matrix.

It is now possible to obtain all the far field interactions on the lowest level λ_{min} in a two step recursive way. The first step consists of a sweep from the highest level and down. The second sweep is from the lowest level and up. First all far field patterns on every level are computed from

$$
\tilde{\mathbf{F}}_{m\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)=\begin{cases}\displaystyle\sum_{\mathcal{B}_{n(\lambda+1)}\in\mathcal{I}^{\mathcal{S}}_{m\lambda}}e^{\imath \boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{m\lambda}-\mathbf{X}_{n(\lambda+1)}\right)}\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\tilde{\mathbf{F}}_{n(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right), & \\ \qquad\qquad\qquad\qquad\qquad\qquad\lambda_{min}\leq\lambda\leq\lambda_{max}-1 & \\ \displaystyle\sum_{l\in\mathcal{B}_{m\lambda_{max}}}\mathbf{F}_l\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)I_l, \qquad \lambda=\lambda_{max}\end{cases} \qquad (5.28)
$$

Secondly, the translations are computed and the contributions from the father to the box are added through

$$\mathbf{G}_{m\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) = \begin{cases} \displaystyle\sum_{\mathcal{B}_{n\lambda}\in\mathcal{I}^{\mathcal{F}}_{m\lambda}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right)\tilde{\mathbf{F}}_n\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) & \lambda = \lambda_{min} \\ \displaystyle\sum_{\mathcal{B}_{n\lambda}\in\mathcal{I}^{\mathcal{F}}_{m\lambda}} \mathcal{T}_j^L\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right)\tilde{\mathbf{F}}_n\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) & \\ \quad + \left(\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\right)^T e^{\imath\boldsymbol{\kappa}_{j_\lambda}\cdot\left(\mathbf{X}_{m\lambda}-\mathbf{X}_{o(\lambda-1)}\right)}\mathbf{G}_{o(\lambda-1)}\left(\hat{\boldsymbol{\kappa}}_{j_{(\lambda-1)}}\right) & \\ & \lambda_{min}+1 \le \lambda \le \lambda_{max} \end{cases} \tag{5.29}$$

For the boxes on level λ_{max} the contributions from the far field boxes are calculated from the receiving patterns and $\mathbf{G}_{m\lambda_{max}}\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)$ as before. An algorithm that uses the proposed method is given in Algorithm 5.2.

The interpolation can be performed by a sparse matrix. The number of quadrature points on one level is about four times greater than the number on the level one step higher. Since the object is a surface, it implies that there is an increase of about four times more boxes going from one level to the next higher. This implies that the number of levels is $\mathcal{O}\left(\log_4 N\right)$ if there are no clusterings of basis functions. Together all this implies that the work on each level is proportional to $\mathcal{O}\left(N\right)$ and the total work is of $\mathcal{O}\left(N\log N\right)$.

5.8 Interpolating the far field pattern

In order to achieve an $\mathcal{O}\left(N\log N\right)$ method the interpolation $\mathcal{P}^{\lambda}_{\lambda+1}$ between the levels must be fast. Several interpolation schemes exists. Here, the interpolation methods in [9, 40] are used.

The basic observation is that the values from a (θ, ϕ)-grid are interpolated to another grid with more nodes as in Figure 5.6. The ϕ-values are located on an equidistant grid, since the trapezoidal rule is used in that direction, while the θ-values are located on a non-equidistant grid, since Gauss-Legendre-quadrature is used in that direction. Since the values are located on a sphere the boundary conditions are periodic on the grid. For the ϕ-direction this is true since $0 \le \phi < 2\pi$. On the other hand θ is defined over $0 \le \theta \le \pi$. The boundary conditions for θ then becomes $(-\theta, \phi) = (\theta, \pi + \phi)$ and for $\theta = \pi + \Delta$ they become $(\pi + \Delta, \phi) = (\pi - \Delta, \pi + \phi)$, where the periodicity of ϕ is assumed.

5.8.1 Direct spectral interpolation

The use of quadrature points to sample the far field functions indicates that they are band limited. A function $f\left(\theta, \phi\right)$ that is band limited of degree L on a sphere is defined as

Algorithm 5.2 The multilevel Fast Multipole Method matrix vector multiplication.

Require: The appropriate initializations and the vector $\mathbf{I}$
Ensure: $\mathbf{V} = \mathbf{ZI}$
  for $m = 1 : M_{\lambda_{max}}$ **do**
    Find box $\mathcal{B}_{m\lambda_{max}}$
    for $k \in \mathcal{B}_{m\lambda_{max}}$ **do**
      Compute $V_k = \sum_{l \in \mathcal{B}_{j\lambda_{max}} \in \mathcal{I}^{\mathcal{N}}_{m\lambda_{max}}} Z^{near}_{kl} I_l$
    end for
  end for
  for $m = 1 : M_{\lambda_{max}}$ **do**
    Find box $\mathcal{B}_{m\lambda_{max}}$
    for $j = 1 : K_{\lambda_{max}}$ **do**
      Compute $\tilde{\mathbf{F}}_{m\lambda_{max}}\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right) = \sum_{l \in \mathcal{B}_{m\lambda_{max}}} \mathbf{F}_{l\lambda_{max}}\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right) I_l$
    end for
  end for
  for $\lambda = \lambda_{max} - 1 : -1 : \lambda_{min}$ **do**
    for $m = 1 : M_\lambda$ **do**
      Find box $\mathcal{B}_{m\lambda}$
      for $j = 1 : K_\lambda$ **do**
        $\tilde{\mathbf{F}}_{m\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) = \sum_{\mathcal{B}_{n(\lambda+1)} \in \mathcal{I}^{\mathcal{S}}_{m\lambda}} e^{\imath \boldsymbol{\kappa}_{j_\lambda} \cdot \left(\mathbf{X}_{m\lambda} - \mathbf{X}_{n(\lambda+1)}\right)} \mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) \tilde{\mathbf{F}}_{n(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}\right)$
      end for
    end for
  end for
  for $\lambda = \lambda_{min} : \lambda_{max}$ **do**
    for $m = 1 : M_\lambda$ **do**
      Find box $\mathcal{B}_{m\lambda}$
      for $j = 1 : K_\lambda$ **do**
        $\mathbf{G}_{m\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right) = \sum_{\mathcal{B}_{n\lambda} \in \mathcal{I}^{\mathcal{F}}_{m\lambda}} \mathcal{T}^L_j\left(\kappa, \mathbf{X}_m - \mathbf{X}_n\right) \tilde{\mathbf{F}}_n\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)$
      end for
    end for
  end for
  for $\lambda = \lambda_{min} : \lambda_{max} - 1$ **do**
    for $m = 1 : M_\lambda$ **do**
      Find box $\mathcal{B}_{m\lambda}$
      for $n = 1 : N_{\mathcal{I}^{\mathcal{S}}_{m\lambda}}$ **do**
        Find box $\mathcal{B}_{n(\lambda+1)}$
        for $j = 1 : K_{\lambda+1}$ **do**
          $\mathbf{G}_{n(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}_{j_{(\lambda+1)}}\right) = \mathbf{G}_{n(\lambda+1)}\left(\hat{\boldsymbol{\kappa}}_{j_{(\lambda+1)}}\right)$
          $+\left(\mathcal{P}^{\lambda}_{\lambda+1}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)\right)^T e^{\imath \boldsymbol{\kappa}_{j_\lambda} \cdot \left(\mathbf{X}_{n(\lambda+1)} - \mathbf{X}_{m\lambda}\right)} \mathbf{G}_{m\lambda}\left(\hat{\boldsymbol{\kappa}}_{j_\lambda}\right)$
        end for
      end for
    end for
  end for
  for $m = 1 : M_{\lambda_{max}}$ **do**
    Find box $\mathcal{B}_m$
    for $k \in \mathcal{B}_{m\lambda_{max}}$ **do**
      Compute $V_k = V_k + \sum_{j=1}^{K} \mathbf{R}_k\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right) \cdot \mathbf{G}_{m\lambda_{max}}\left(\hat{\boldsymbol{\kappa}}_{j_{\lambda_{max}}}\right)$
    end for
  end for

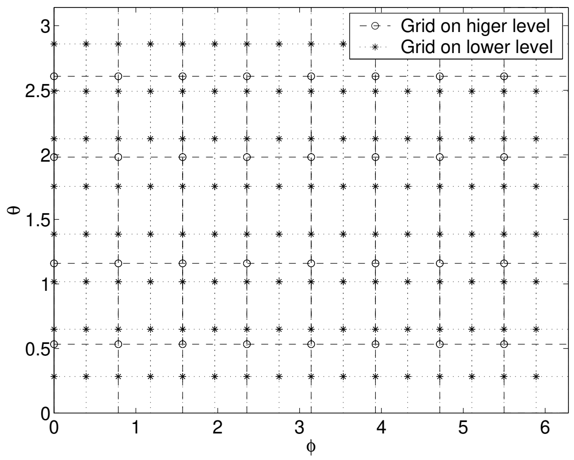

Figure 5.6: The values at the nodes on the grid at the higher level are interpolated to the nodes of the grid at the lower level.

$$f(\theta,\phi) = \sum_{l=1}^{L} \sum_{m=-l}^{l} f_{ml} Y_l^m(\theta,\phi) \tag{5.30}$$

Multiplying with $(Y_l^m(\theta,\phi))^*$ and using the orthogonality relation of spherical harmonics $\int_S Y_l^m (Y_k^n)^* \, dS = \delta_{kl}\delta_{mn}$ yield

$$f_{ml} = \int_S f(\theta,\phi) (Y_l^m)^* \, dS \tag{5.31}$$

Since, the function $f(\theta,\phi)$ can be expressed in terms of the spherical harmonics of order up to L and f_{ml} should be computed for $-l \le m \le l$ and $0 \le l \le L$, a quadrature rule that computes the $2L$ first order spherical harmonics is enough to compute all f_{ml} needed.

Once, all f_{ml} are known $f(\theta,\phi)$ can be evaluated at any point. For instance $f(\theta,\phi)$ can be computed at the quadrature points needed on a lower level yielding

$$\begin{aligned} f(\theta_{k'},\phi_{k'}) &= \sum_{l=0}^{L} \sum_{m=-l}^{l} f_{ml} Y_l^m(\theta_{k'},\phi_{k'}) \\ &= \sum_{l=0}^{L} \sum_{m=-l}^{l} Y_l^m(\theta_{k'},\phi_{k'}) \left(\sum_{k=1}^{K} w_k f(\theta_k,\phi_k) (Y_l^m(\theta_k,\phi_k))^* \right) \end{aligned} \tag{5.32}$$

where $f(\theta_{k'},\phi_{k'})$ can be computed for $1 \le k' \le K'$. Here, K' is the number of quadrature points on the lower level.

If equation (5.32) is used to compute the K' values at level $\lambda - 1$ from the K values at level λ, then it can be written as a dense matrix vector product as

$$f\left(\theta_{k'}, \phi_{k'}\right) = \sum_{k=1}^{K} A_{k'k} f\left(\theta_k, \phi_k\right) \tag{5.33}$$

where A is an $K' \times K$ matrix and the entry $A_{k'k}$ is given by

$$A_{k'k} = \sum_{l=0}^{L} \sum_{m=-l}^{l} w_k Y_l^m\left(\theta_{k'}, \phi_{k'}\right)\left(Y_l^m\left(\theta_k, \phi_k\right)\right)^* \tag{5.34}$$

The interpolation scheme described here does not result in any speedup compared to the ordinary Fast Multipole Method. The reason is that because of the dense matrix vector multiplication the number of arithmetic operations is $\mathcal{O}\left(K^2\right)$. In order to achieve any speedup, the number of arithmetic operations for the interpolation scheme must scale better than that.

5.8.2 Fast Fourier Transform based spectral interpolation

In this section we discuss how to improve on the direct spectral interpolation method in Section 5.8.1.

To shorten the notation in the following discussion the normalized associated Legendre functions are introduced

$$\bar{P}_l^m = \sqrt{\frac{2l+1}{2}\frac{(l-m)!}{(l+m)!}} e^{\imath m\phi} P_l^m\left(\cos\theta\right) \quad -l \le m \le l \tag{5.35}$$

Equation (5.32) together with the definition of the spherical harmonics in equation (5.10) and the definition of the normalized associated Legendre functions in equation (5.35) yield

$$\begin{aligned} f\left(\theta_{k'}, \phi_{k'}\right) &= \sum_{l=0}^{L} \sum_{m=-l}^{l} Y_l^m\left(\theta_{k'}, \phi_{k'}\right)\left(\sum_{k=1}^{K} w_k f\left(\theta_k, \phi_k\right)\left(Y_l^m\left(\theta_k, \phi_k\right)\right)^*\right) \\ &= \sum_{l=0}^{L} \sum_{m=-l}^{l} \sum_{k=1}^{K} \frac{w_k}{2\pi} e^{\imath m\left(\phi_{k'} - \phi_k\right)} \bar{P}_l^m\left(\cos\theta_{k'}\right) \bar{P}_l^m\left(\cos\theta_k\right) f\left(\theta_k, \phi_k\right) \\ &= \sum_{m=-L}^{L} \sum_{l=|m|}^{L} \sum_{k=1}^{K} \frac{w_k}{2\pi} e^{\imath m\left(\phi_{k'} - \phi_k\right)} \bar{P}_l^m\left(\cos\theta_{k'}\right) \bar{P}_l^m\left(\cos\theta_k\right) f\left(\theta_k, \phi_k\right) \end{aligned} \tag{5.36}$$

Since the weights are constant for the trapezoidal rule, choosing the trapezoidal rule as the quadrature rule for ϕ gives weights w_k that only depend on θ. If $\tilde{P}_{Lm}\left(\theta_{k'}, \theta_k\right) = \sum_{l=|m|}^{L} \bar{P}_l^m\left(\cos\theta_{k'}\right) \bar{P}_l^m\left(\cos\theta_k\right)$, it allows equation (5.36) to be rewritten as

$$f(\theta_{k'}, \phi_{k'}) = \sum_{m=-L}^{L} e^{\imath m\phi_{k'}} \sum_{k=1}^{L+1} \frac{w_{jk}}{2\pi} \tilde{P}_{Lm}(\theta_{k'}, \theta_k) \sum_{j=1}^{2L+2} e^{-\imath m\phi_j} f(\theta_k, \phi_j) \quad (5.37)$$

The first and last sum in equation (5.37) can be computed for all ϕ and a fixed θ by using a discrete Fast Fourier Transform. This means that they can be computed in $\mathcal{O}(K \log K)$ arithmetic operations. If the middle sum is computed by a direct method, it requires $\mathcal{O}\left(K^{\frac{3}{2}}\right)$ arithmetic operations, since it is computed for $2L+1$ matrices of size $\mathcal{O}(2L^2)$ and K is proportional to L^2. This method does not result in an $\mathcal{O}(N \log N)$ method either, but it can be used for small boxes yielding a spectrally accurate method for them.

5.8.3 Sparsifying the spectral interpolation operator

The entries in the matrix A in equation (5.34) show a decay as $(\theta_{k'}, \phi_{k'})$ moves away from (θ_k, ϕ_k). One possibility for a faster method is to sparsify the matrix A by removing the entries below some threshold. This approach is not so efficient however, which was shown in [9]. The reason is that the entries have a slow decay rate. The decay is explained by the fact that the higher modes in Fourier space are truncated abruptly. This problem is addressed in [9] where it is proposed that a smoother signature function is used instead, which redefines the entries in A as

$$A_{k'k} = \sum_{l=0}^{L+L'} \sum_{m=-l}^{l} C_l w_k Y_l^m(\theta_{k'}, \phi_{k'}) \left(Y_l^m(\theta_k, \phi_k)\right)^* \quad (5.38)$$

where the constant C_l is defined as

$$C_l = \begin{cases} 1 & 0 \le l \le L \\ \cos^2\left(\frac{(l-L)\pi}{2(L'-L)}\right) & L+1 \le l \le L' \end{cases} \quad (5.39)$$

The new entries show a much faster decay. The problem is that the number of spherical harmonics that has to be integrated is increased. This implies that more samples have to be used to represent the far field pattern, which in turn implies that more memory is required. Nevertheless it is possible to reduce the scaling of the algorithm to $\mathcal{O}(N \log N)$ with this approach, because A is sparse.

5.8.4 Lagrange interpolation

The simplest approach that results in an $\mathcal{O}(N \log N)$ method is to use ordinary Lagrange interpolation. To compute $f(\theta_{k'}, \phi_{k'})$ a number of neighboring points to (θ_k, ϕ_k) are defined. In each direction, m points are selected for the interpolation. In ϕ points are taken with the periodic boundary conditions and in θ the periodic conditions that were introduced in Section 5.8 are used.

The points close to $\theta_{k'}$ can be used to define a Lagrange polynomial in θ by

$$\tilde{f}(\theta, \phi_k) = \sum_{i=1}^{m} l_i(\theta) f(\theta_i, \phi_k) \tag{5.40}$$

Here, $l_i(\theta)$ is defined by

$$l_i(\theta) = \prod_{j \neq i} \frac{\theta - \theta_j}{\theta_i - \theta_j} = \begin{cases} 1 & \theta = \theta_j \quad j = i \\ 0 & \theta = \theta_j \quad j \neq i \end{cases} \tag{5.41}$$

An approximation to $f(\theta_{k'}, \phi_k)$ is then given by $\tilde{f}(\theta_{k'}, \phi_k)$. The coefficients given by $l_i(\theta_{k'})$ can be stored in a matrix and equation (5.40) used to define a matrix vector product.

Once the values in θ are known the same procedure can be used for ϕ. Using the approximation from θ a new Lagrange polynomial is defined in ϕ as

$$\hat{f}(\theta_{k'}, \phi) = \sum_{i=1}^{m} l_i(\phi) \tilde{f}(\theta_{k'}, \phi_i) \tag{5.42}$$

Here, $l_i(\phi)$ is defined by

$$l_i(\phi) = \prod_{j \neq i} \frac{\phi - \phi_j}{\phi_i - \phi_j} = \begin{cases} 1 & \phi = \phi_j \quad j = i \\ 0 & \phi = \phi_j \quad j \neq i \end{cases} \tag{5.43}$$

The function $\hat{f}(\theta_{k'}, \phi)$ results in an approximation to $f(\theta_{k'}, \phi_{k'})$ given by $\hat{f}(\theta_{k'}, \phi_{k'})$. As before the coefficients $l_i(\phi_{k'})$ are stored in a matrix.

The result is an approximation to $f(\theta_{k'}, \phi_{k'})$ which can be computed for all $(\theta_{k'}, \phi_{k'})$ using two sparse matrix vector multiplications. Since the matrices are sparse the scaling of the resulting multilevel method is $\mathcal{O}(N \log N)$.

5.9 Implementation issues

Although the multilevel Fast Multipole Method results in a method that scales as $\mathcal{O}(N \log N)$, the constant in front of the leading term in the complexity estimate is fairly large. Therefore it is very important to reduce the constant as much as possible. This is especially true for the constant in front of the memory consumption of $\mathcal{O}(N \log N)$, since it is usually the memory that is prohibiting factor when solving large problems.

Here, some issues are dealt with that reduce this constant. The most important contribution is the relation between different $\hat{\boldsymbol{\kappa}}$ directions.

5.9.1 Reducing the memory for the far field pattern

A simple method for reducing the memory required by the far field pattern was proposed in [9]. It is based on the observation that once the far field pattern on a level is transfered to all its far neighbors, it is not needed for any more computations. This means that the memory occupied by it can be reused to store something else. In our case the translated far field on the levels above occupy that memory space.

In Algorithm 5.2 the far field pattern $\tilde{\mathbf{F}}_{m\lambda}(\hat{\boldsymbol{\kappa}}_{j_\lambda})$ and the translated far field pattern $\mathbf{G}_{m\lambda}(\hat{\boldsymbol{\kappa}}_{j_\lambda})$ have to be stored. The simplest way is allocate one memory buffer for each of them. Instead, we need one memory buffer with the size of one of the previous ones plus the size of the memory for the level requiring the most memory.

First, all far field patterns are computed as before. Then starting from the lowest level, the far fields are translated to the unused part of memory. Once the far fields have been translated, a new piece of memory is free, which can be used to calculate the translations for the next level, and so on. This approach reduces the memory requirement for this part by almost half. A view is given in Figure 5.7.

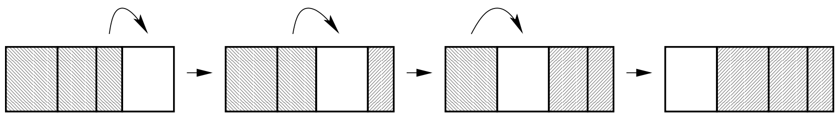

Figure 5.7: Efficient use of memory for storing the far field patterns.

To implement this one way is to allocate a large piece of memory and then decide where the pointers associated with $\tilde{\mathbf{F}}_{m\lambda}(\hat{\boldsymbol{\kappa}}_{j_\lambda})$ and $\mathbf{G}_{m\lambda}(\hat{\boldsymbol{\kappa}}_{j_\lambda})$ should point. In this way, the fact that several far field patterns share the same location is hidden for the higher level implementation of Algorithm 5.2.

A fast implementation of the Fast Multipole method would precompute all the far field patterns and receiving patterns for the basis functions on the highest level and store them. One way of reducing the memory requirement is to compute the far field patterns on the when they are needed. This gives a memory reduction of $\mathcal{O}(KN)$ but also gives an increase in the computing time of the same order. Thus, it is a trade-off between if it is computational time or memory that is the limiting factor.

5.9.2 Storing the translation operator

The most naive way of storing the translation operators is to compute all the translation operators required by a box and store them for the box. This approach misses a crucial point which is that the translation operators depend on $\hat{\boldsymbol{\kappa}}$ and $\mathbf{X}$. Since the boxes are inside a grid the same combination of the values of $\hat{\boldsymbol{\kappa}}$ and $\mathbf{X}$ will appear several times. An example of this is given in Figure 5.8. The best way to store the values is thus based on the possible values of $\hat{\boldsymbol{\kappa}}$ and $\mathbf{X}$. In a multilevel method with m boxes in any direction considered as near neighbors of a box, there is a total of $(2m+1)^3$ near neighbors of the box. These boxes are divided in eight smaller boxes on the level above, which means that the $(2m+1)^3$ boxes become $(4m+2)^3$ boxes on the higher level. The boxes that are handled using the far field method on the higher level are at most $(4m+2)^3 - (2m+1)^3$ boxes. With $m = 1$ there are 189 far field boxes. The fact that the boxes on the higher level have a different center compared to their father implies that there are $(4m+3)^3 - (2m+1)^3$ possible values on $\mathbf{X}$ each associated with a distance between two boxes. For $m = 1$ there are 316 possible values. This means that at most $\left((4m+3)^3 - (2m+1)^3\right) K_\lambda$ values have to be stored for the translation operator. It will be shown that it is actually enough with $\left((2m+2)^3 - (m+1)^3\right) K_\lambda$ values using certain properties of the plane wave vectors in Section 5.9.3. For $m = 1$ this means that $56K_\lambda$ values have to be stored for the translation operator on each level.

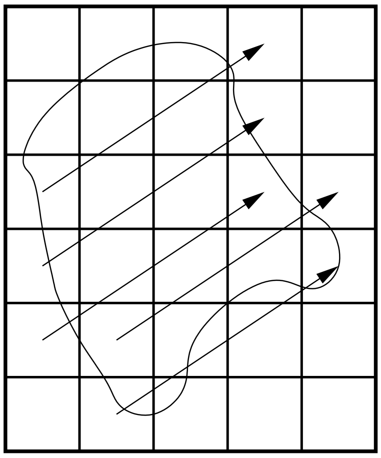

Figure 5.8: The translation operators in the picture are all the same. Only one copy of them should be stored.

5.9.3 Relationships between the plane wave vectors and its application

Some relationships between different $\hat{\boldsymbol{\kappa}}$ vectors are derived and applied to different parts of Algorithm 5.2. The result is a significant reduction in the memory storage requirements for the translation operators, the exponential translation operators in the interpolation and the far field and radiation patterns.

First the properties are derived. The directional plane wave vector $\hat{\boldsymbol{\kappa}}$ can be written as $\hat{\boldsymbol{\kappa}} = \sin\theta\cos\phi\hat{\mathbf{x}}_1 + \sin\theta\sin\phi\hat{\mathbf{x}}_2 + \cos\theta\hat{\mathbf{x}}_3$. An ordinary vector $\mathbf{x}$ can be written as $\mathbf{x} = x_1\hat{\mathbf{x}}_1 + x_2\hat{\mathbf{x}}_2 + x_3\hat{\mathbf{x}}_3$. Here, $\mathbf{x}_j$ for $j = 1, 2, 3$ are the three orthogonal unitary vectors in 3-dimensional space.

Using elementary trigonometric relations we have

$$\begin{cases} \begin{aligned} \hat{\boldsymbol{\kappa}} \cdot (x_1\hat{\mathbf{x}}_1 + x_2\hat{\mathbf{x}}_2 + x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi x_1 + \sin\theta\sin\phi x_2 + \cos\theta x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (-x_1\hat{\mathbf{x}}_1 + x_2\hat{\mathbf{x}}_2 + x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi\left(-x_1\right) + \sin\theta\sin\phi x_2 + \cos\theta x_3 \\ &= \sin\theta\cos\left(\pi - \phi\right) x_1 \\ &\quad + \sin\theta\sin\left(\pi - \phi\right) x_2 + \cos\theta x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (x_1\hat{\mathbf{x}}_1 - x_2\hat{\mathbf{x}}_2 + x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi x_1 + \sin\theta\sin\phi\left(-x_2\right) + \cos\theta x_3 \\ &= \sin\theta\cos\left(2\pi - \phi\right) x_1 \\ &\quad + \sin\theta\sin\left(2\pi - \phi\right) x_2 + \cos\theta x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (x_1\hat{\mathbf{x}}_1 + x_2\hat{\mathbf{x}}_2 - x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi x_1 + \sin\theta\sin\phi x_2 + \cos\theta\left(-x_3\right) \\ &= \sin\left(\pi - \theta\right)\cos\phi x_1 \\ &\quad + \sin\left(\pi - \theta\right)\sin\phi x_2 + \cos\left(\pi - \theta\right) x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (-x_1\hat{\mathbf{x}}_1 - x_2\hat{\mathbf{x}}_2 + x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi\left(-x_1\right) + \sin\theta\sin\phi\left(-x_2\right) + \cos\theta x_3 \\ &= \sin\theta\cos\left(\pi + \phi\right) x_1 \\ &\quad + \sin\theta\sin\left(\pi + \phi\right) x_2 + \cos\theta x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (-x_1\hat{\mathbf{x}}_1 + x_2\hat{\mathbf{x}}_2 - x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi\left(-x_1\right) + \sin\theta\sin\phi x_2 + \cos\theta\left(-x_3\right) \\ &= \sin\left(\pi - \theta\right)\cos\left(\pi - \phi\right) x_1 \\ &\quad + \sin\left(\pi - \theta\right)\sin\left(\pi - \phi\right) x_2 + \cos\left(\pi - \theta\right) x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (x_1\hat{\mathbf{x}}_1 - x_2\hat{\mathbf{x}}_2 - x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi x_1 + \sin\theta\sin\phi\left(-x_2\right) + \cos\theta\left(-x_3\right) \\ &= \sin\left(\pi - \theta\right)\cos\left(2\pi - \phi\right) x_1 \\ &\quad + \sin\left(\pi - \theta\right)\sin\left(2\pi - \phi\right) x_2 + \cos\left(\pi - \theta\right) x_3 \\ \hat{\boldsymbol{\kappa}} \cdot (-x_1\hat{\mathbf{x}}_1 - x_2\hat{\mathbf{x}}_2 - x_3\hat{\mathbf{x}}_3) &= \sin\theta\cos\phi\left(-x_1\right) \\ &\quad + \sin\theta\sin\phi\left(-x_2\right) + \cos\theta\left(-x_3\right) \\ &= \sin\left(\pi - \theta\right)\cos\left(\pi + \phi\right) x_1 \\ &\quad + \sin\left(\pi - \theta\right)\sin\left(\pi + \phi\right) x_2 + \cos\left(\pi - \theta\right) x_3 \end{aligned} \end{cases} \tag{5.44}$$

Suppose that a function only depends on a scalar product between $\hat{\boldsymbol{\kappa}}$ and $\mathbf{x}$ and that the function values are known for all possible positive values of x_1, x_2 and x_3 that we need. If the function values are distributed on a grid of θ and ϕ values for a given $\mathbf{x}$, as in our case, then equation (5.44) implies that the values in the seven other corners of an octant are known.

This is used in Algorithm 5.3. The assumption is that a function F is of the form $F(\hat{\boldsymbol{\kappa}}, \mathbf{x}) = F(\hat{\boldsymbol{\kappa}} \cdot \mathbf{x})$. The function F is sampled at the quadrature points (θ_i, ϕ_j) for $1 \leq j \leq L + 1$ and $1 \leq i \leq 2L + 2$ for a given $\mathbf{x}$.

Algorithm 5.3 Computes $V(\theta_i,\phi_j) = V(\theta_i,\phi_j) + F_{\mathbf{x}}(\theta_i,\phi_j)\,G(\theta_i,\phi_j)$ for all (θ_i,ϕ_j).

Require: $V(\theta_i,\phi_j)$, $F_{\mathbf{x}'}(\theta_i,\phi_j)$ for $x_i' = |x_i|$, $G(\theta_i,\phi_j)$ for all (θ_i,ϕ_j) and $\mathbf{x}$
Ensure: $V(\theta_i,\phi_j) = V(\theta_i,\phi_j) + F_{\mathbf{x}}(\theta_i,\phi_j)\,G(\theta_i,\phi_j)$ for all (θ_i,ϕ_j).
  if $x_3 < 0$ and $x_2 < 0$ and $x_1 < 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:L+1}) = V(\theta_i,\phi_{1:L+1}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{L+2:2L+2})\,G(\theta_i,\phi_{1:L+1})$
      $V(\theta_i,\phi_{L+1:2L+2}) = V(\theta_i,\phi_{L+1:2L+2}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{1:L+1})\,G(\theta_i,\phi_{L+1:2L+2})$
    end for
  end if
  if $x_3 < 0$ and $x_2 < 0$ and $x_1 > 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_1) = V(\theta_i,\phi_1) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_1)\,G(\theta_i,\phi_1)$
      $V(\theta_i,\phi_{2:2L+2}) = V(\theta_i,\phi_{2:2L+2}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{2L+2:-1:2})\,G(\theta_i,\phi_{2:2L+2})$
    end for
  end if
  if $x_3 < 0$ and $x_2 > 0$ and $x_1 < 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:L+2}) = V(\theta_i,\phi_{1:L+2}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{L+2:-1:1})\,G(\theta_i,\phi_{1:L+2})$
      $V(\theta_i,\phi_{L+2:2L+2}) = V(\theta_i,\phi_{L+2:2L+2}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{2L+2:-1:L+2})\,G(\theta_i,\phi_{L+2:2L+2})$
    end for
  end if
  if $x_3 < 0$ and $x_2 > 0$ and $x_1 > 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:2L+2}) = V(\theta_i,\phi_{1:2L+2}) + F_{\mathbf{x}'}(\theta_{L+2-i},\phi_{1:2L+2})\,G(\theta_i,\phi_{1:2L+2})$
    end for
  end if
  if $x_3 > 0$ and $x_2 < 0$ and $x_1 < 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:L+1}) = V(\theta_i,\phi_{1:L+1}) + F_{\mathbf{x}'}(\theta_i,\phi_{L+2:2L+2})\,G(\theta_i,\phi_{1:L+1})$
      $V(\theta_i,\phi_{L+1:2L+2}) = V(\theta_i,\phi_{L+1:2L+2}) + F_{\mathbf{x}'}(\theta_i,\phi_{1:L+1})\,G(\theta_i,\phi_{L+1:2L+2})$
    end for
  end if
  if $x_3 > 0$ and $x_2 < 0$ and $x_1 > 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_1) = V(\theta_i,\phi_1) + F_{\mathbf{x}'}(\theta_i,\phi_1)\,G(\theta_i,\phi_1)$
      $V(\theta_i,\phi_{2:2L+2}) = V(\theta_i,\phi_{2:2L+2}) + F_{\mathbf{x}'}(\theta_i,\phi_{2L+2:-1:2})\,G(\theta_i,\phi_{2:2L+2})$
    end for
  end if
  if $x_3 > 0$ and $x_2 > 0$ and $x_1 < 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:L+2}) = V(\theta_i,\phi_{1:L+2}) + F_{\mathbf{x}'}(\theta_i,\phi_{L+2:-1:1})\,G(\theta_i,\phi_{1:L+2})$
      $V(\theta_i,\phi_{L+2:2L+2}) = V(\theta_i,\phi_{L+2:2L+2}) + F_{\mathbf{x}'}(\theta_i,\phi_{2L+2:-1:L+2})\,G(\theta_i,\phi_{L+2:2L+2})$
    end for
  end if
  if $x_3 > 0$ and $x_2 > 0$ and $x_1 > 0$ **then**
    for $i = 1 : L+1$ **do**
      $V(\theta_i,\phi_{1:2L+2}) = V(\theta_i,\phi_{1:2L+2}) + F_{\mathbf{x}'}(\theta_i,\phi_{1:2L+2})\,G(\theta_i,\phi_{1:2L+2})$
    end for
  end if

This algorithm can help compress some of the memory requirements by eight times since only one function F has to be stored for a given $\mathbf{x}$ instead of eight functions. The two most important applications are compression of the translation operator and compression of the exponential translation values.

In the case of the exponential translation values the compression is a factor of eight. This is because the distance to the father from a box is the same for any box. In the case of the translation operator the factor is lower. If there are m neighboring boxes on a level there are $(2m+2)^3$ possible values for a positive $\mathbf{x}$. Since there are $(m+1)^3$ possible values on $\mathbf{x}$ for a near neighbor in the positive direction the number of possible translation operator values is reduced from $\left((4m+3)^3 - (2m+1)^3\right) K_\lambda$ to $\left((2m+2)^3 - (m+1)^3\right) K_\lambda$. For $m = 1$ the compression is about 5.6 times.

Another use of the algorithm is that only half of the far field and receiving patterns of the basis functions have to be stored. This is realized by comparing $\hat{\boldsymbol{\kappa}} \cdot \mathbf{x}$ and $\hat{\boldsymbol{\kappa}} \cdot (-\mathbf{x})$. For some of the receiving patterns a factor $C\hat{\boldsymbol{\kappa}}$ is in front, see (5.19) and (5.23). This factor is related to the K operator. The solution is to store the receiving patterns for T and K separately and use the fact the far field pattern is the same for both T and K and it takes the same form as the receiving pattern of T. If the far field and receiving patterns are computed on the fly to reduce the memory requirement a speedup factor of almost two is obtained if this method is used instead.

It should be noted that if the number of ϕ directions is divisible by four, further improvement is possible.

5.10 Shared memory parallelization

A simple parallelization of the Fast Multipole Method is achieved by using a shared memory machine. For this purpose OpenMP [31] can be used. A more difficult task is to use MPI as in [7]. Since all the operations for a box in Algorithm 5.2 are independent of the other boxes on the same level, it is easy to parallelize it using a shared memory paradigm. The work is divided between the processors by simply letting each processor do the work for a part of the boxes. The use of Morton ordering ensures that the boxes close to each other in the list are close to each other physically as well. This is favorable since boxes that are physically close communicate a lot. Since there is a big chance that neighboring boxes belong to the same processor this increases the chance of memory reusage.

The problem with this strategy is when the number of processors become larger than the number of boxes. If this is the case then there are idle processors, resulting in bad speedup. This can occur for the boxes on the lower levels, since they are quite few. A simple way of solving this problem is to increase λ_{min}, the lowest level on which any computations are made. The increase means that there are more boxes on the lowest level, which improves the parallelism. The drawback is that the number of translations is increased on that level. On the other hand there is no need to interpolate to a lower level. Furthermore, the number of samples on a given level is about four times more than on the level above. This indicates that an increase of λ_{min} with 1 or 2 should have a small impact on the scaling.

Chapter 6

Sparse Approximate Inverse

The efficiency of an iterative method depends on the number of iterations carried out before a convergence criterion is met. If the problem matrix is well-conditioned the number of iterations is low. For ill-conditioned matrices this is not the case. In this case the number of iterations can be very high and the iterative method looses its edge over Gaussian elimination type methods.

In order to address the problem of ill-conditioning one can use one of two strategies. The first strategy reformulates the mathematical problem into a new problem that results in a better conditioned matrix. Switching from EFIE to CFIE is one such example. The second strategy is to introduce a preconditioner. Suppose that we want to solve the problem $\mathbf{ZI} = \mathbf{V}$. If we introduce a preconditioner $\mathbf{M}$, we can solve the problem $(\mathbf{ZM})\left(\mathbf{M}^{-1}\mathbf{I}\right) = \mathbf{V}$ instead. If $(\mathbf{ZM})$ is better conditioned than $\mathbf{Z}$ the number of iterations is reduced. This also occurs if the spectrum of $(\mathbf{ZM})$ has better convergence properties than the spectrum of $\mathbf{Z}$. Since we want to use fast methods for the matrix vector multiplication part the construction of the preconditioner should be at least equally fast.

In this chapter we discuss how a Sparse Approximate Inverse preconditioner (SPAI) [23] can be constructed for the case of the integral equations in electromagnetics.

6.1 Sparse Approximate Inverse preconditioning

The goal of a Sparse Approximate Inverse preconditioner (SPAI) [23] is to construct a sparse matrix $\mathbf{M}$ that in some sense approximates the inverse of a matrix $\mathbf{Z}$.

In this case the goal will be to find a minimizer of the expression

$$\min_{\mathbf{M}\in\mathcal{A}} \|\mathbf{I} - \mathbf{ZM}\|_F \tag{6.1}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm and $\mathcal{A}$ is the space of admissible matrix solutions. The admissible solutions $\mathcal{A}$ in the case of a Sparse Approximate Inverse have some kind of sparsity constraints on them. If $\mathcal{A}$ includes all matrices then $\mathbf{M} = \mathbf{Z}^{-1}$. If the entries that contributes most to minimizing equation (6.1) are included in the space of admissible solutions $\mathcal{A}$ one can expect $\mathbf{M}$ to be a good approximation of $\mathbf{Z}^{-1}$. This is the reason why a Sparse Approximate Inverse preconditioner is attractive.

A drawback with this method is that it is difficult to predict what the sparsity constraints on $\mathbf{M}$ should be. The sparsity constraint should be such that the large entries in $\mathbf{Z}^{-1}$ are found in $\mathbf{M}$, since these entries are the ones that are expected to give the greatest reduction in equation (6.1). Usually only heuristic rules exist for choosing the sparsity constraints.

The minimization problem (6.1) can be restated as

$$\min_{\mathbf{M}\in\mathcal{A}} \|\mathbf{I} - \mathbf{ZM}\|_F^2 = \sum_{i=1}^{N} \min_{\mathbf{M}\in\mathcal{A}} \|(\mathbf{I} - \mathbf{ZM})\,\mathbf{e}_i\|_2^2 \tag{6.2}$$

where $\mathbf{e}_i$ is a unit vector. This means that the minimization of column i in $\mathbf{M}$ can be carried out independently of the other columns in $\mathbf{M}$. Thus, the minimization in (6.1) is equivalent to

$$\min_{\mathbf{m}_i\in\mathcal{A}_i} \|\mathbf{e}_i - \mathbf{Zm}_i\|_2 \quad 1 \le i \le N \tag{6.3}$$

where $\mathcal{A}_i$ describes the admissible solutions of column i.

Let $\mathcal{J}_i = \{j \in \mathcal{J}_i | m_{ji} \neq 0\}$ be the sparsity pattern that is prescribed for $\mathbf{m}_i$. Then the minimization problem in equation (6.3) is equivalent to

$$\min_{\mathbf{m}_i\in\mathcal{A}_i} \|\mathbf{e}_i - \mathbf{Zm}_i\|_2 = \min_{\mathbf{m}_i(\mathcal{J}_i)} \|\mathbf{e}_i - \mathbf{Z}\,(:,\mathcal{J}_i)\,\mathbf{m}_i\,(\mathcal{J}_i)\|_2 \tag{6.4}$$

If $\mathbf{Z}$ is a sparse matrix then it is probable that many rows in $\mathbf{Z}\,(:,\mathcal{J}_i)$ are zero. With the nonzero rows of $\mathbf{Z}\,(:,\mathcal{J}_i)$ denoted by $\mathcal{I}_i = \{i \in \mathcal{I}_i | \mathbf{Z}\,(i,\mathcal{J}_i) \neq 0\}$ the final problem to solve is

$$\min_{\mathbf{m}_i(\mathcal{J}_i)} \|\mathbf{e}_i - \mathbf{Z}\,(:,\mathcal{J}_i)\,\mathbf{m}_i\,(\mathcal{J}_i)\|_2 = \min_{\mathbf{m}_i(\mathcal{J}_i)} \|\mathbf{e}_i\,(\mathcal{I}_i) - \mathbf{Z}\,(\mathcal{I}_i,\mathcal{J}_i)\,\mathbf{m}_i\,(\mathcal{J}_i)\|_2 \tag{6.5}$$

Suppose that the number of nonzero rows of the columns in $\mathbf{M}$ are approximately the same. Let k be the average number of nonzero entries in the columns of $\mathbf{M}$. Assume that the distribution of nonzero entries is uniform and that the average number of elements in $\mathcal{I}_i$ are s. Then equation (6.5) can be solved for each column $\mathbf{m}_i$ in $\mathbf{M}$ by $\mathbf{QR}$-decomposing the matrix $\mathbf{Z}\,(\mathcal{I}_i,\mathcal{J}_i)$ and solving the normal equation

$$\mathbf{Z}\,(\mathcal{I}_i,\mathcal{J}_i)^H\,\mathbf{Z}\,(\mathcal{I}_i,\mathcal{J}_i)\,\mathbf{m}_i\,(\mathcal{J}_i) = \mathbf{Z}\,(\mathcal{I}_i,\mathcal{J}_i)^H\,\mathbf{e}_i\,(\mathcal{I}_i) \tag{6.6}$$

Therefore finding $\mathbf{M}$ requires $C_1Nsk^2 + C_2Nsk + C_3Nk^2$ arithmetic operations for some constants C_j. For sparse matrices $s = \mathcal{O}(k)$ so the number of arithmetic operations is $D_1Nk^3 + D_2Nk^2$ for some D_1 and D_2.

If the matrix $\mathbf{Z}$ is dense, then $s \approx N$ and the total number of arithmetic operations is $C_1N^2k^2 + C_2N^2k + C_3Nk^2$ which is too expensive to be used together with a fast solution method like the Fast Multipole Method where the matrix vector multiplication requires $\mathcal{O}(N \log N)$ operations. Since an impedance matrix is dense, some improvements are needed in order to quickly compute the approximate inverse.

6.2 Reducing complexity for impedance matrices

For dense matrices the number of arithmetic operations is unacceptable if the iterative method uses a fast matrix vector multiplication algorithm like the Fast Multipole Method. Thus, the complexity $C_1Nsk^2 + C_2Nsk + C_3Nk^2$ arithmetic operations must be reduced.

A Sparse Approximate Inverse preconditioner for the dense matrices arising in electromagnetics was proposed in [2]. The strategy used was to sparsify the matrix $\mathbf{Z}$ by removing all entries smaller than some threshold. Once the matrix was sparsified a Sparse Approximate Inverse preconditioner for the sparsified matrix was constructed. We notice that this strategy is not possible in the Fast Multipole Method since only a portion of the entries are stored.

One way to reduce the complexity is to block several columns together [29]. As for the Fast Multipole Method the blocking is based on boxes $\mathcal{B}_l$. For simplicity one can choose the same boxes as for the Fast Multipole Method. With this approach a number of columns share the same sparsity pattern denoted by $\mathcal{J}_{\mathcal{B}_l} = \{j \in \mathcal{J}_{\mathcal{B}_l} | m_{ji} \neq 0,\, i \in \mathcal{B}_l\}$. For each box $\mathcal{B}_l$ the following minimization problem should be solved

$$\min_{\mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})} \|\mathbf{e}_i - \mathbf{Z}(:, \mathcal{J}_{\mathcal{B}_l})\, \mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})\|_2 \quad i \in \mathcal{B}_l \tag{6.7}$$

With $M = N/l$ boxes the total number of arithmetic operations is reduced to $C_1Nsk^2/l + C_2Nsk + C_3Nk^2$. This is because the matrix $\mathbf{Z}(:, \mathcal{J}_{\mathcal{B}_l})$ is factorized once for each box.

To reduce the work further the fact that $s \approx N$ should be addressed. In the case of an impedance matrix, the entries become smaller for large distances, since $G(\mathbf{x}, \mathbf{x}') \to 0$ when $|\mathbf{x} - \mathbf{x}'| \to \infty$. This indicates that contributions to the minimization problem from the entries far away are small. With this in mind these entries are removed from the minimization problem by choosing the nonzero rows for each box $\mathcal{B}_l$ as $\mathcal{I}_{\mathcal{B}_l} = \{i \in \mathcal{I}_{\mathcal{B}_l} | \mathrm{dist}(i, \mathcal{B}_l) \leq \epsilon\}$ [28, 29]. Here, $\mathrm{dist}(i, \mathcal{B}_l)$ means the distance between the center of box $\mathcal{B}_l$ and basis function i. The distance is measured in max-norm for simplicity. The resulting minimization problem for box $\mathcal{B}_l$ is

$$\min_{\mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})} \|\mathbf{e}_i(\mathcal{I}_{\mathcal{B}_l}) - \mathbf{Z}(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})\, \mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})\|_2 \quad i \in \mathcal{B}_l \tag{6.8}$$

The choice of the parameter ϵ determines the number of rows in $\mathbf{Z}(:, \mathcal{J}_{\mathcal{B}_l})$ that are approximated to zero. Only the entries for the nearest neighbor boxes are stored in the Fast Multipole Method. Therefore, ϵ should not be larger than the distance that includes only the nearest neighbor boxes. Algorithm 6.1 shows how the values in $\mathbf{M}$ are computed.

Algorithm 6.1 Setup of the Sparse Approximate Inverse preconditioner.

Require: The appropriate initializations
Ensure: The preconditioner $\mathbf{M}$
  for $l = 1 : M$ **do**
    Find box $\mathcal{B}_l$
    Find $\mathcal{I}_{\mathcal{B}_l}$ and $\mathcal{J}_{\mathcal{B}_l}$
    Find the entries in $\mathbf{Z}(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})$
    for $i \in \mathcal{B}_l$ **do**
      Solve $\min_{\mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})} \|\mathbf{e}_i(\mathcal{I}_{\mathcal{B}_l}) - \mathbf{Z}(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})\, \mathbf{m}_i(\mathcal{J}_{\mathcal{B}_l})\|_2$
    end for
  end for

As was mentioned in Section 6.1 it is not clear how the entries in $\mathcal{J}_{\mathcal{B}_l}$ should be selected in general. In this case one can expect large entries in $\mathbf{Z}^{-1}$ when the distance is small. The choice for $\mathcal{J}_{\mathcal{B}_l}$ should be based on this assumption so it is $\mathcal{J}_{\mathcal{B}_l} = \{j \in \mathcal{J}_{\mathcal{B}_l} | \mathrm{dist}(j, \mathcal{B}_l) \leq \delta\}$ [28, 29]. The parameter δ is usually smaller than the parameter ϵ. Figure 6.1 illustrates how δ and ϵ can be chosen.

Only some parts of the impedance matrix $\mathbf{Z}$ is stored in the Fast Multipole Method. The rest are only available indirectly through the Fast Multipole expansion. Therefore, it is probable that some of the entries of $\mathbf{Z}(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})$ are unavailable. Thus they are not stored in memory and easily accessible. The entries that are not stored correspond to interactions between near neighbor boxes of box $\mathcal{B}_l$ that are not near neighbors to each other. One could compute these entries and store them in their positions which is quite costly. A simpler approach is to ignore these entries as well since they are expected to be small. A typical example of the structure of the matrices in the minimization problem is given in Figure 6.2.

The strategy of including only the nearest neighbors was proposed in [42]. It was however only used in the special case $\epsilon = \delta$. Another special case is the block Jacobi method in [7]. Here, $\epsilon = \delta$ and they are chosen so that they include the box itself and no neighbor boxes. In [1] it was proposed that the stored part of the matrix in the Fast Multipole Method should be **LU**-factorized in a block fashion. This resulted in a method with scaling $\mathcal{O}(Nk^3/l)$, but the method is difficult to parallelize.

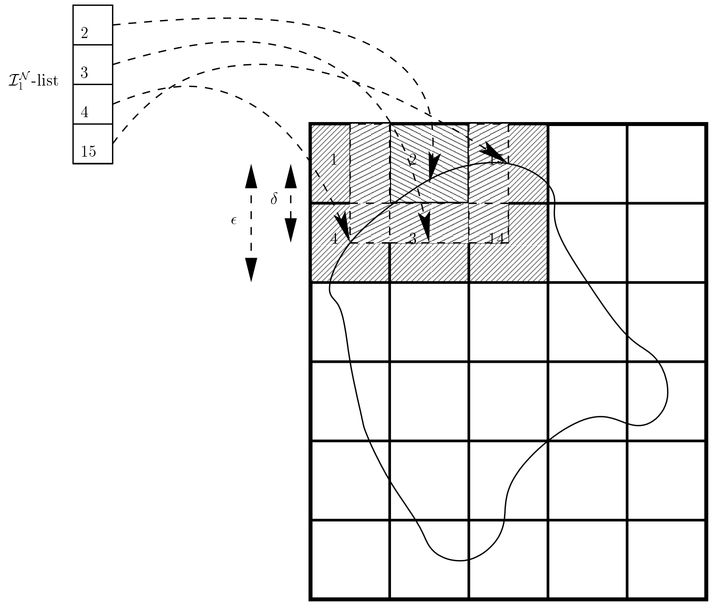

Figure 6.1: The parameters δ and ϵ for box $\mathcal{B}_1$.

Let k be the average number of nonzero rows in the columns of $\mathbf{M}$. With $\delta = \alpha\epsilon$, where $\alpha \leq 1$ is constant, the average number of rows in $\mathbf{Z}$ is $s = \mathcal{O}(k)$. Thus, the number of arithmetic operations for computing $\mathbf{M}$ is $C_1Nk^3/l + C_2Nk^2$. If δ is not much larger than the size of a box then $l = \mathcal{O}(k)$ and the total number of arithmetic operations is CNl^2. To be more precise $k \approx 4l$ and $s \approx 9l$, which can be realized from geometric considerations. This is faster than the estimate given in [42] which was $\mathcal{O}(Nk^3)$. That estimate comes from factorizing a matrix one column at a time.

It is clear from the previous discussion that the matrix $\mathbf{M}$ is sparse, since the number of nonzero entries in a column on average is k, which means that there is a total of $\mathcal{O}(Nk)$ nonzero entries. Thus, a matrix vector multiplication with $\mathbf{M}$ can be performed in $\mathcal{O}(Nk)$ arithmetic operations. One way of computing a matrix vector multiplication with $\mathbf{M}$ is to use Algorithm 6.2.

6.3 Algebraic analysis

We analyze how close the modified preconditioner suggested in Section 6.2 is to the minimizing preconditioner. We also consider how close the complex symmetric part of the Sparse Approximate Inverse preconditioner is to the minimizing preconditioner if the matrix $\mathbf{Z}$ is complex symmetric.

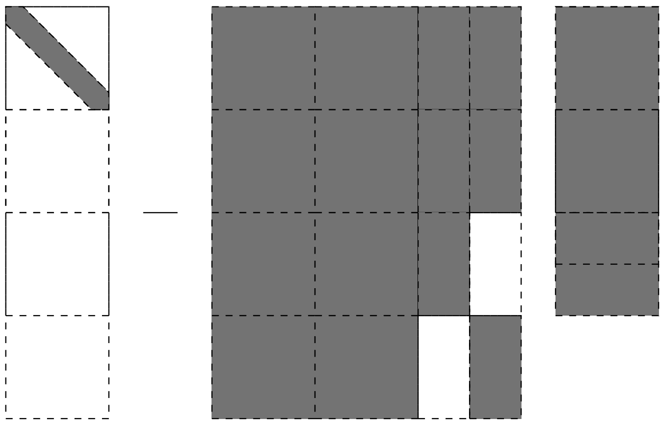

Figure 6.2: The structure of the matrices in the problem min $\|\mathbf{I} - \mathbf{ZM}\|$ for box $\mathcal{B}_1$. The empty parts of $\mathbf{Z}$ are ignored because these parts correspond to far field interactions computed with the Fast Multipole Method.

Algorithm 6.2 Applying the Sparse Approximate Inverse preconditioner.

Require: The appropriate initializations and the vector $\mathbf{I}$
Ensure: $\mathbf{V} = \mathbf{MI}$
  $V_j = 0$
  for $l = 1 : M$ **do**
    Find box $\mathcal{B}_l$
    for $j \in \mathcal{J}_{\mathcal{B}_l}$ **do**
      Compute $V_j = V_j + \sum_{i \in \mathcal{B}_l} M_{ji} I_i$
    end for
  end for

In the following analysis we will use norms $\|\cdot\|$ with the property that for any complex matrix $\mathbf{A}$ it holds that

$$\left\|\mathbf{A}^T\right\| = \|\mathbf{A}\| \tag{6.9}$$

We call this property the transpose independent matrix norm property. We also require that for any two matrices $\mathbf{A}$ and $\mathbf{B}$ the norm fulfills the submultiplicative property [21]

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\| \tag{6.10}$$

Both (6.9) and (6.10) hold in the Frobenius norm

In Section 3.3 it was pointed out that in equations derived from Rumsey's reaction principle, the impedance matrix is complex symmetric. The impact of using a non-symmetric matrix as preconditioner is analyzed in the following proposition that was previously stated in [28].

Proposition 6.3.1 *If a norm has the transpose independent matrix norm property (6.9) and the submultiplicative property (6.10) then for any complex symmetric matrix* $\mathbf{Z}$

$$\frac{\left\|\mathbf{M}^T - \mathbf{M}\right\|}{2\left\|\mathbf{M}\right\|} \leq \left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\| \tag{6.11}$$

Proof Suppose that $\mathbf{I} - \mathbf{Z}\mathbf{M} = \mathbf{E}$. It follows that

$$\mathbf{E}^T = (\mathbf{I} - \mathbf{Z}\mathbf{M})^T = \mathbf{I} - (\mathbf{Z}\mathbf{M})^T = \mathbf{I} - \mathbf{M}^T\mathbf{Z}$$

since $\mathbf{Z}$ is complex symmetric. This implies that

$$\mathbf{E}^T\mathbf{M} - \mathbf{M}^T\mathbf{E} = \left(\mathbf{I} - \mathbf{M}^T\mathbf{Z}\right)\mathbf{M} - \mathbf{M}^T(\mathbf{I} - \mathbf{Z}\mathbf{M}) = \mathbf{M} - \mathbf{M}^T$$

Taking the norm of this expression yields

$$\left\|\mathbf{M} - \mathbf{M}^T\right\| \leq \left\|\mathbf{M}\right\|\left\|\mathbf{I} - \mathbf{M}^T\mathbf{Z}\right\| + \left\|\mathbf{M}^T\right\|\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\|$$

Since the norm has the transpose independent matrix norm property the proposition follows.

We note that Proposition 6.3.1 is true in the Frobenius norm which indicates that there is a lower bound in (6.1) when one tries to minimize $\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\|_F$ and $\mathbf{Z}$ is symmetric. This is because the matrix $\mathbf{M}$ is in general not complex symmetric. There are other reasons for preferring a complex symmetric matrix $\mathbf{M}$ as well, e.g. if the iterative solver requires a complex symmetric preconditioner. One way to make the Sparse Approximate Inverse preconditioner symmetric is to use the complex symmetric part of the preconditioner. This choice is analyzed in Proposition 6.3.2. Proposition 6.3.2 indicates that it is possible to use the symmetric part of $\mathbf{M}$ as a preconditioner when $\mathbf{Z}$ is complex symmetric.

Proposition 6.3.2 *Assume that a norm has the transpose independent matrix norm property (6.9) and the submultiplicative property (6.10) and let* $\mathbf{M}$ *minimize* $\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\|$. *If* $\mathbf{Z}$ *is complex symmetric then the following properties hold*

$$\left\|\mathbf{Z}^{-1} - \mathbf{M}\right\| \leq \left\|\mathbf{Z}^{-1}\right\|\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\| \tag{6.12}$$

$$\left\|\mathbf{Z}^{-1} - \frac{\mathbf{M} + \mathbf{M}^T}{2}\right\| \leq \left\|\mathbf{Z}^{-1}\right\|\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\| \tag{6.13}$$

$$\left\|\mathbf{I} - \mathbf{Z}\left(\frac{\mathbf{M} + \mathbf{M}^T}{2}\right)\right\| \leq \frac{1 + cond(\mathbf{Z})}{2}\left\|\mathbf{I} - \mathbf{Z}\mathbf{M}\right\| \tag{6.14}$$

where $cond(\mathbf{Z}) = \left\|\mathbf{Z}\right\|\left\|\mathbf{Z}^{-1}\right\|$ *denotes the condition number of* $\mathbf{Z}$.

Proof The first property follows from

$$\left\|\mathbf{Z}^{-1}-\mathbf{M}\right\| = \left\|\mathbf{Z}^{-1}\left(\mathbf{I}-\mathbf{ZM}\right)\right\| \leq \left\|\mathbf{Z}^{-1}\right\| \left\|\mathbf{I}-\mathbf{ZM}\right\|$$

In the proof of the second property the transpose independent matrix norm property is used

$$\begin{aligned}
\left\|\mathbf{Z}^{-1}-\frac{\mathbf{M}+\mathbf{M}^T}{2}\right\| &= \left\|\mathbf{Z}^{-1}\left(\mathbf{I}-\frac{\mathbf{Z}}{2}\left(\mathbf{M}+\mathbf{M}^T\right)\right)\right\| \\
&= \left\|\frac{\mathbf{Z}^{-1}}{2}\left(\mathbf{I}-\mathbf{ZM}\right)+\frac{\mathbf{Z}^{-1}}{2}\left(\mathbf{I}-\mathbf{ZM}^T\right)\right\| \\
&= \left\|\frac{\mathbf{Z}^{-1}}{2}\left(\mathbf{I}-\mathbf{ZM}\right)+\left(\mathbf{Z}^{-1}-\mathbf{M}^T\right)\frac{\mathbf{ZZ}^{-1}}{2}\right\| \\
&= \left\|\frac{\mathbf{Z}^{-1}}{2}\left(\mathbf{I}-\mathbf{ZM}\right)+\left(\mathbf{I}-\mathbf{M}^T\mathbf{Z}\right)\frac{\mathbf{Z}^{-1}}{2}\right\| \\
&\leq \left\|\frac{\mathbf{Z}^{-1}}{2}\right\| \left\|\mathbf{I}-\mathbf{ZM}\right\| + \left\|\mathbf{I}-\mathbf{M}^T\mathbf{Z}\right\| \left\|\frac{\mathbf{Z}^{-1}}{2}\right\| \\
&= \left\|\mathbf{Z}^{-1}\right\| \left\|\mathbf{I}-\mathbf{ZM}\right\|
\end{aligned}$$

The proof of the third property is similar to the proof of the second property

$$\begin{aligned}
\left\|\mathbf{I}-\frac{\mathbf{Z}}{2}\left(\mathbf{M}+\mathbf{M}^T\right)\right\| &= \left\|\frac{1}{2}\left(\mathbf{I}-\mathbf{ZM}\right)+\frac{1}{2}\left(\mathbf{I}-\mathbf{ZM}^T\right)\right\| \\
&= \left\|\frac{1}{2}\left(\mathbf{I}-\mathbf{ZM}\right)+\frac{1}{2}\mathbf{Z}\left(\mathbf{I}-\mathbf{M}^T\mathbf{Z}\right)\mathbf{Z}^{-1}\right\| \\
&\leq \frac{1}{2}\left\|\mathbf{I}-\mathbf{ZM}\right\| + \frac{1}{2}\left\|\mathbf{Z}\right\| \left\|\mathbf{I}-\mathbf{M}^T\mathbf{Z}\right\| \left\|\mathbf{Z}^{-1}\right\| \\
&= \frac{1+\operatorname{cond}\left(\mathbf{Z}\right)}{2}\left\|\mathbf{I}-\mathbf{ZM}\right\|
\end{aligned}$$

We note that also Proposition 6.3.2 is true in the Frobenius norm. The first property is stated in [23] and is also true for non-symmetric matrices. The third property in Proposition 6.3.2 was proved in [28] and improves on a similar statement in [23].

The modification to the Sparse Approximate Inverse preconditioner in Section 6.2 that removed rows from $\mathbf{Z}$ changes the minimization problem. In the following proposition the impact of the change is analyzed. This proposition was previously stated in [28].

Proposition 6.3.3 *Suppose that $\mathbf{x}_{LS}$ is the least squares solution to*

$$\min_{\mathbf{x}_{LS}} \left\| \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \mathbf{x}_{LS} \right\|_2 \tag{6.15}$$

and let $\tilde{\mathbf{x}}_{LS} = \mathbf{x}_{LS} + \boldsymbol{\epsilon}$ be the least squares solution to

$$\min_{\tilde{\mathbf{x}}_{LS}} \|\mathbf{b} - \mathbf{A}_1 \tilde{\mathbf{x}}_{LS}\|_2 \tag{6.16}$$

Then the difference between $\tilde{\mathbf{x}}_{LS}$ and $\mathbf{x}_{LS}$ is given by

$$\boldsymbol{\epsilon} = \left(\mathbf{A}_1^H \mathbf{A}_1\right)^{-1} \mathbf{A}_2^H \mathbf{A}_2 \mathbf{x}_{LS} \tag{6.17}$$

The relative error between $\tilde{\mathbf{x}}_{LS}$ and $\mathbf{x}_{LS}$ can be estimated by

$$\frac{\|\tilde{\mathbf{x}}_{LS} - \mathbf{x}_{LS}\|_2}{\|\mathbf{x}_{LS}\|_2} \leq \left(cond_2\left(\mathbf{A}_1\right) \frac{\|\mathbf{A}_2\|}{\|\mathbf{A}_1\|} \right)^2 \tag{6.18}$$

Proof The solution of equation (6.15) is given by the normal equation

$$\begin{pmatrix} \mathbf{A}_1^H & \mathbf{A}_2^H \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \mathbf{x}_{LS} = \begin{pmatrix} \mathbf{A}_1^H & \mathbf{A}_2^H \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}$$

which is the same as

$$\mathbf{A}_1^H \mathbf{b} - \mathbf{A}_1^H \mathbf{A}_1 \mathbf{x}_{LS} = \mathbf{A}_2^H \mathbf{A}_2 \mathbf{x}_{LS}$$

The solution of equation (6.16) is given by the normal equation

$$\mathbf{A}_1^H \mathbf{A}_1 \tilde{\mathbf{x}}_{LS} = \mathbf{A}_1^H \mathbf{A}_1 \left(\mathbf{x}_{LS} + \boldsymbol{\epsilon}\right) = \mathbf{A}_1^H \mathbf{b}$$

This implies that $\boldsymbol{\epsilon}$ can be computed from the equation

$$\mathbf{A}_1^H \mathbf{A}_1 \boldsymbol{\epsilon} = \mathbf{A}_1^H \mathbf{b} - \mathbf{A}_1^H \mathbf{A}_1 \mathbf{x}_{LS} = \mathbf{A}_2^H \mathbf{A}_2 \mathbf{x}_{LS}$$

which proves the first part of the proposition. For the second part, we use the fact that $\left\|\mathbf{A}^H \mathbf{A}\right\|_2 = \sigma_1^2$, where σ_1 is the largest singular value of $\mathbf{A}$, and $\left\|\left(\mathbf{A}^H \mathbf{A}\right)^{-1}\right\|_2 = \sigma_n^{-2}$, where σ_n is the smallest singular value of $\mathbf{A}$. If $\sigma_i^{(j)}$ are the singular values of $\mathbf{A}_j$ then

$$\begin{aligned} \|\boldsymbol{\epsilon}\|_2 &\leq \left\|\left(\mathbf{A}_1^H \mathbf{A}_1\right)^{-1}\right\|_2 \left\|\mathbf{A}_2^H \mathbf{A}_2\right\|_2 \|\mathbf{x}_{LS}\|_2 \\ &= \left(\frac{\sigma_1^{(2)}}{\sigma_n^{(1)}}\right)^2 \|\mathbf{x}_{LS}\|_2 = \left(\frac{\sigma_1^{(1)} \sigma_1^{(2)}}{\sigma_n^{(1)} \sigma_1^{(1)}}\right)^2 \|\mathbf{x}_{LS}\|_2 \\ &= \left(\mathrm{cond}_2\left(\mathbf{A}_1\right) \frac{\|\mathbf{A}_2\|_2}{\|\mathbf{A}_1\|_2}\right)^2 \|\mathbf{x}_{LS}\|_2 \end{aligned}$$

where we have used that for any matrix $\mathbf{A}$ the 2-norm is $\|\mathbf{A}\|_2 = \sigma_1$ and the 2-norm condition number is $\mathrm{cond}_2\left(\mathbf{A}\right) = \sigma_1 \sigma_n^{-1}$. This proves the second part of the proposition.

Since, the entries in the ignored part of $\mathbf{Z}(:, \mathcal{J}_{\mathcal{B}_l})$ are small, it is expected that the error in the approximate solution will be small. To estimate the impact on the matrix $\mathbf{M}$ the next proposition is used

Proposition 6.3.4 *Suppose that $\tilde{\mathbf{M}}$ is the preconditioner when some of the rows in each minimization problem are replaced by $\mathbf{0}$ and that the matrix $\mathbf{M}$ is the result from* $\min \|\mathbf{I} - \mathbf{Z}\mathbf{M}\|_F$. *Assume that $\|\tilde{\mathbf{m}}_j - \mathbf{m}_j\|_2 \leq \gamma \|\mathbf{m}_j\|_2$ for all columns $1 \leq j \leq N$. Then the relative error in Frobenius norm is estimated by*

$$\frac{\left\|\tilde{\mathbf{M}} - \mathbf{M}\right\|_F}{\|\mathbf{M}\|_F} \leq \gamma \tag{6.19}$$

Proof From the definition of the Frobenius norm and the assumptions we find that

$$\left\|\tilde{\mathbf{M}} - \mathbf{M}\right\|_F^2 = \sum_{j=1}^{N} \|\tilde{\mathbf{m}}_j - \mathbf{m}_j\|_2^2 \leq \sum_{j=1}^{N} \gamma^2 \|\mathbf{m}_j\|_2^2 \leq \gamma^2 \|\mathbf{M}\|_F^2$$

The proposition follows by taking the square root of each side.

Proposition 6.3.3 together with Proposition 6.3.4 gives a simple estimate of the difference between the minimizing preconditioner and the modified preconditioner which we give in Corollary 6.3.1.

Corollary 6.3.1 *The difference between the minimizing preconditioner and the modified preconditioner can be estimated by choosing γ in Proposition 6.3.4 as*

$$\gamma = \max_{1 \leq l \leq M} \left(cond_2 \left(\mathbf{Z} \left(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l} \right) \right) \frac{\|\mathbf{Z}(: \backslash \mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})\|_2}{\|\mathbf{Z}(\mathcal{I}_{\mathcal{B}_l}, \mathcal{J}_{\mathcal{B}_l})\|_2} \right)^2 \tag{6.20}$$

The final estimate of the difference between the modified preconditioner and the minimizing preconditioner is given in Theorem 6.3.1. To prove it we need Lemma 6.3.1, which gives an estimate of $\|\mathbf{M}\|$.

Lemma 6.3.1 *In any norm with the submultiplicative property (6.10) one can estimate $\|\mathbf{M}\|$ by*

$$\|\mathbf{M}\| \leq \left\|\mathbf{Z}^{-1}\right\| (1 + \|\mathbf{I} - \mathbf{Z}\mathbf{M}\|) \tag{6.21}$$

Proof The submultiplicative property give

$$\|\mathbf{M}\| = \left\|\mathbf{Z}^{-1} + \mathbf{M} - \mathbf{Z}^{-1}\right\| \leq \left\|\mathbf{Z}^{-1}\right\| (1 + \|\mathbf{I} - \mathbf{Z}\mathbf{M}\|)$$

which proves the lemma.

We use Proposition 6.3.1 and Proposition 6.3.4 together with Lemma 6.3.1 to prove Theorem 6.3.1.

Theorem 6.3.1 *Suppose that* $\mathbf{M}$ *is the minimizer of* $\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F$ *and that* $\tilde{\mathbf{M}}$ *is a matrix such that* $\|\tilde{\mathbf{m}}_j-\mathbf{m}_j\|_2 \leq \gamma \|\mathbf{m}_j\|_2$ *for all columns* $1 \leq j \leq N$*. Then the following estimates hold in the Frobenius norm*

$$\left\|\mathbf{I}-\mathbf{Z}\tilde{\mathbf{M}}\right\|_F \leq (1+\gamma cond_F(\mathbf{Z}))\,\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F+\gamma cond_F(\mathbf{Z}) \tag{6.22}$$

$$\left\|\mathbf{Z}^{-1}-\tilde{\mathbf{M}}\right\|_F \leq (1+\gamma)\left\|\mathbf{Z}^{-1}\right\|_F\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F+\gamma\left\|\mathbf{Z}^{-1}\right\|_F \tag{6.23}$$

$$\left\|\mathbf{I}-\mathbf{Z}\left(\frac{\tilde{\mathbf{M}}+\tilde{\mathbf{M}}^T}{2}\right)\right\|_F \leq \frac{1+(1+2\gamma)\,cond_F(\mathbf{Z})}{2}\,\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F+\gamma cond_F(\mathbf{Z}) \tag{6.24}$$

$$\left\|\mathbf{Z}^{-1}-\frac{\tilde{\mathbf{M}}+\tilde{\mathbf{M}}^T}{2}\right\|_F \leq (1+\gamma)\left\|\mathbf{Z}^{-1}\right\|_F\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F+\gamma\left\|\mathbf{Z}^{-1}\right\|_F \tag{6.25}$$

Proof The submultiplicative property and the transpose independent matrix norm property of the Frobenius norm yield

$$\begin{aligned}
\left\|\mathbf{I}-\mathbf{Z}\tilde{\mathbf{M}}\right\|_F &\leq \|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F+\|\mathbf{Z}\|_F\left\|\tilde{\mathbf{M}}-\mathbf{M}\right\|_F \\
\left\|\mathbf{Z}^{-1}-\tilde{\mathbf{M}}\right\|_F &\leq \left\|\mathbf{Z}^{-1}-\mathbf{M}\right\|_F+\left\|\tilde{\mathbf{M}}-\mathbf{M}\right\|_F \\
\left\|\mathbf{I}-\mathbf{Z}\left(\frac{\tilde{\mathbf{M}}+\tilde{\mathbf{M}}^T}{2}\right)\right\|_F &\leq \left\|\mathbf{I}-\mathbf{Z}\left(\frac{\mathbf{M}+\mathbf{M}^T}{2}\right)\right\|_F+\|\mathbf{Z}\|_F\left\|\tilde{\mathbf{M}}-\mathbf{M}\right\|_F \\
\left\|\mathbf{Z}^{-1}-\frac{\tilde{\mathbf{M}}+\tilde{\mathbf{M}}^T}{2}\right\|_F &\leq \left\|\mathbf{Z}^{-1}-\frac{\mathbf{M}+\mathbf{M}^T}{2}\right\|_F+\left\|\tilde{\mathbf{M}}-\mathbf{M}\right\|_F
\end{aligned}$$

Proposition 6.3.4 together with Lemma 6.3.1 gives the estimate

$$\left\|\tilde{\mathbf{M}}-\mathbf{M}\right\|_F \leq \gamma\left\|\mathbf{Z}^{-1}\right\|_F(1+\|\mathbf{I}-\mathbf{Z}\mathbf{M}\|_F)$$

The other parts are estimated by Proposition 6.3.1.

We note that the first two estimates in Theorem 6.3.1 are true for all matrices $\mathbf{Z}$. Theorem 6.3.1 shows that it is possible to have control over the error caused by not solving the correct minimization problem if one has a good estimate of the error introduced in each column.

6.4 Shared memory parallelization of SPAI

To parallelize the Sparse Approximate Inverse preconditioner on a shared memory machine is quite easy. The same approach as for the Fast Multipole method is used. The boxes in Algorithm 6.1 and Algorithm 6.2 are split over the processors in the same way as for the Fast Multipole method. Each processor can solve its minimization problems in parallel with the other processors. The only problem occurs when the matrix vector multiplication in Algorithm 6.2 is performed. In this case it is possible that two processors want to write in the same location in memory. For that reason a memory position must be locked before it can be written to. The extra time that is required to lock a memory position reduces the speedup.

Chapter 7

Numerical experiments

We perform several numerical experiments in order to verify that the Fast Multipole Method works. Results obtained on the sphere are compared to an analytic solution. We also compare results obtained from other codes on aircrafts and on a thin cylinder. The scaling with increasing number of processors is investigated on some problems. We also show that there is a gain with using block iterative methods and the Sparse Approximate Inverse Preconditioner in Chapter 6. Finally, we show results from a full scale aircraft discretized with more than 1 million unknowns, which would require 16 Tb of memory to store the entire matrix.

For CFIE we use $\alpha = 0.2$ in (2.33) and for SPAI we use $\delta = 1.1$ in Figure 6.1. The experiments are mainly conducted on Ngorongoro, a SUN Fire 15k server with 900 MHz processors and shared memory at the Department of Scientific Computing at Uppsala University. Some of the experiments were carried out by Jonas Gustafsson at SAAB Avionics on an SGI 3800 with 500 MHz processors and shared memory at National Supercomputing Center in Linköping.

7.1 Verification

The first set of experiments are tests on a sphere with radius 1 m. It is discretized with different number of unknowns. The reason is that there exists an analytic solution for the sphere known as the Mie-series and it is a simple geometry to test because the equations are well-conditioned and the solution converges fast. In Figure 7.1 we compare the Radar Cross Section given by the Mie-series at 1 GHz compared to two different solutions obtained with the multilevel Fast Multipole Method. The first solution had a setting of $d_0 = 1.0$ digits required in the error tolerance, see Section 5.3. The second solution had d_0 set to 2.25. The sphere was discretized with 20280 edges which gave about 8.5 edges per wavelength. The integral equation used was CFIE with $\alpha = 0.2$. The solution had a maximum error of 0.8 dB located in the dip at around 170 degrees. The solution in Figure 7.1 shows that the Fast Multipole Method gives an accurate solution even at low error settings.

Next we turn to the scaling of the memory requirements and the CPU-time. The number of unknowns range from 3000 to 151230 and we solve for frequencies between 400 Mhz and 2400 MHz giving approximately $6-8$ edges per wavelength. This keeps the number of unknowns independent of wavelength. If this was not the case the memory consumption would increase rapidly, since there is a smallest size for a box. It is not a restriction, because the accuracy depends on the number of unknowns per wavelength. For the memory we only measure the parts that are explicitly used by the Fast Multipole Method and the CPU-time for one matrix vector multiplication is plotted. This means that memory required by the iterative method and the geometry are left out. We also test the theory from Section 5.10 about how the scaling is affected by an increase of λ_{min} in Algorithm 5.2. Three cases are tested: the lowest setting and 1 and 2 levels above the lowest setting. The results are shown in Figure 7.2.

The results from the scaling test indicate that it is not a good idea to increase the level by one or two. The memory requirement is about the same for all three cases, but the CPU time is decreased significantly when λ_{min} is increased. Some careful examination shows that the reduced speed is due to the application of the translation operator part which begins to scale poorly when the lowest level is increased. This could be explained by an increase of cache misses. If not the lowest setting on λ_{min} is used, it is probable that the boxes on the lowest level have to communicate with more boxes

The previous results show that we should not use a reduction in the number of levels, when we parallelize. In order to verify this we test the speedup with the number of processors increased from 1 to 30. Both the maximum number of levels and the case with one level less are tested. To get an idea of how competitive the second test case is we compared the scaling both with itself and with the first case. The test case is the sphere with 151230 unknowns at 2400 MHz. The results are shown in Figure 7.3.

The speedup in Figure 7.3 is quite good for a shared memory implementation. It is interesting that the speedup for the maximum level case stops scaling at the end while the second case continues to scale. This indicates that if a better strategy was devised for the translation part, the second case could be a better method.

In Figure 7.4 we show the Radar Cross Section for a thin open cylinder 10 wavelengths long and 5 wavelengths in diameter. The results obtained with the Fast Multipole Method is compared to results obtained with a different Moment Method. They seem to agree quite well. These results were computed by Jonas Gustafsson at SAAB Avionics using our code. Tomas Lundin at SAAB Avionics computed the Moment Method results.

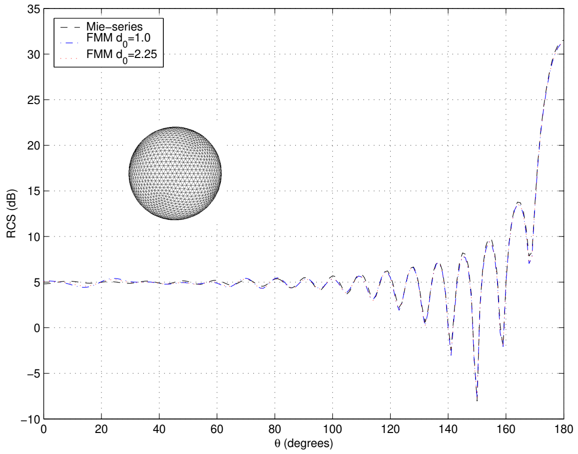

Figure 7.1: Scattering from a sphere at 1 GHz. The CFIE solution, accelerated with FMM, is compared to an analytic solution obtained from the Mie-series. The maximum error was 0.8 dB.

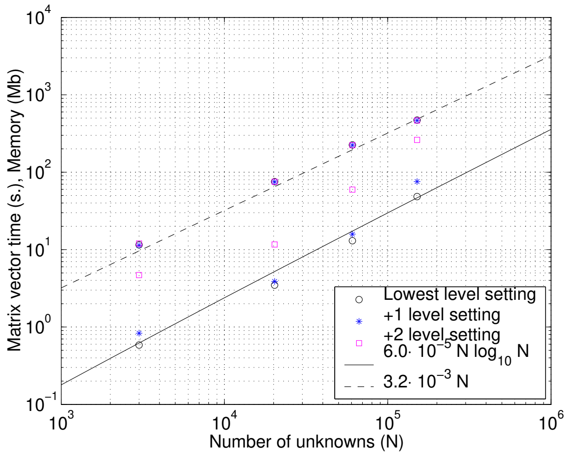

Figure 7.2: Scaling properties of the multilevel Fast Multipole Method on a sphere. The curve $6.0 \cdot 10^{-5} N \log_{10} N$ should be compared with the CPU time and the curve $3.2 \cdot 10^{-3} N$ should be compared with the memory requirement.

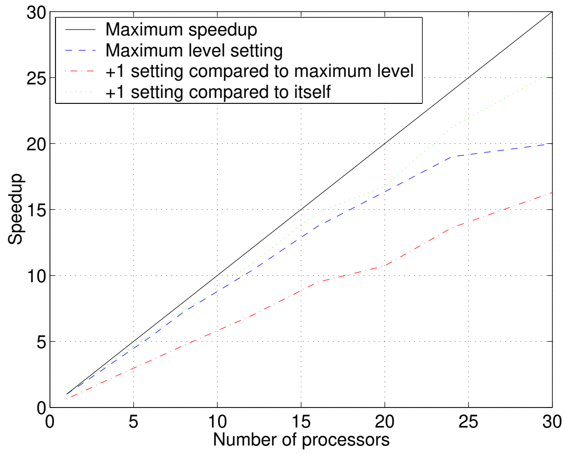

Figure 7.3: Speedup measured on the sphere. Comparison between using maximum number of levels and one less.

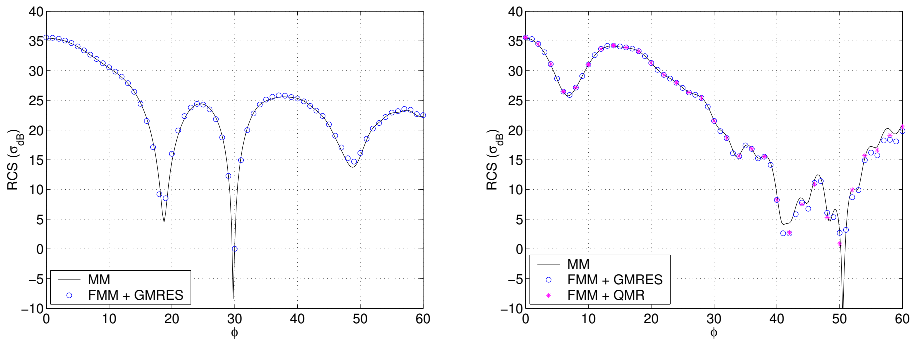

Figure 7.4: Radar Cross Section for a thin open cylinder 10 wavelengths long and 5 wavelengths in diameter. Results are obtained with the Fast Multipole Method and compared to a different Moment Method code. The polarization is vertical to the left and horizontal to the right.

7.2 Numerical experiments on aircraft models

To show that the Fast Multipole Method can handle more difficult geometries we use it to solve problems on different aircraft models. The first model is called RUND and is depicted in Figure 7.5. It is 1 m long. The second model is called SAAB Trainer and is depicted in Figure 7.9 and it is 11 m long.

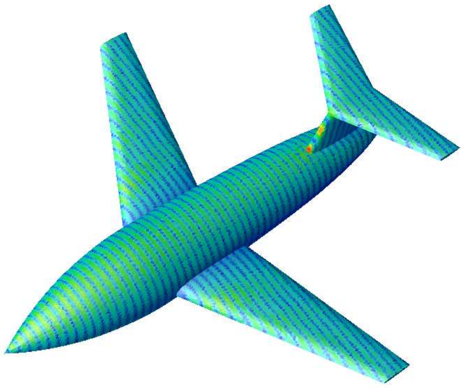

Figure 7.5: The aircraft model RUND. The induced surface currents at 10 GHz.

We compare the predicted Radar Cross Section for RUND at 1.5 GHz obtained with the Fast Multipole Method with several time domain solvers [12] in Figure 7.6. In the case of vertical polarization we see that FDTD with 10 mm resolution is not enough to give an accurate solution. In the case of horizontal polarization the solvers agree except for the nose of the aircraft where there is some discrepancy.

In Figure 7.7 we investigate how the solution time for the Fast Multipole Method depends on the number of processors for RUND. The aircraft is discretized with 64959 unknowns and solved at a frequency of 6 GHz using CFIE and GMRES. We compare solution times obtained on Ngorongoro, a SUN fire 15k system with 48 processors running at 900 MHz with solution times on an SGI 3800 running at 500 MHz. It is clear that the speedup is less for this problem than for the sphere. This is because there are fewer boxes on the lower levels in this case. The scaling is a little bit better on SUN than on SGI but about the same. Preconditioning with SPAI reduces the solution time with a factor of 2, despite the fact that CFIE is well-conditioned from the beginning. The results on SGI was computed by Jonas Gustafsson at SAAB Avionics.

Next we investigate the solution times on SUN when we solve for several right hand sides and use the Block QMR iterative solver in Figure 7.8. The aircraft is discretized with 64959 unknowns and solved at a frequency of 6 GHz using

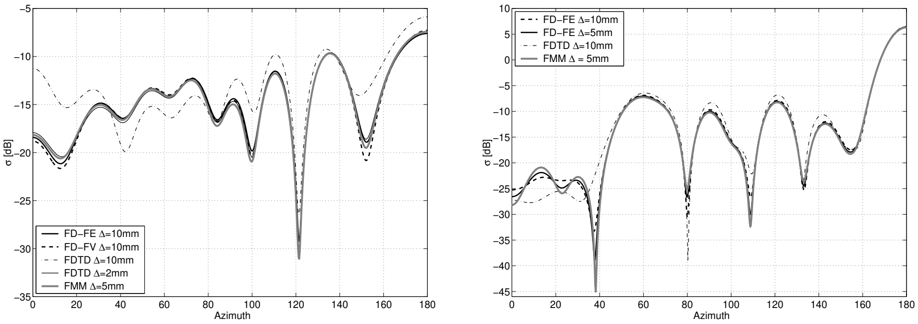

Figure 7.6: Radar Cross Section for RUND at 1.5 GHz. Results are obtained with the Fast Multipole Method and compared to different time domain solvers. The polarization is vertical to the left and horizontal to the right.

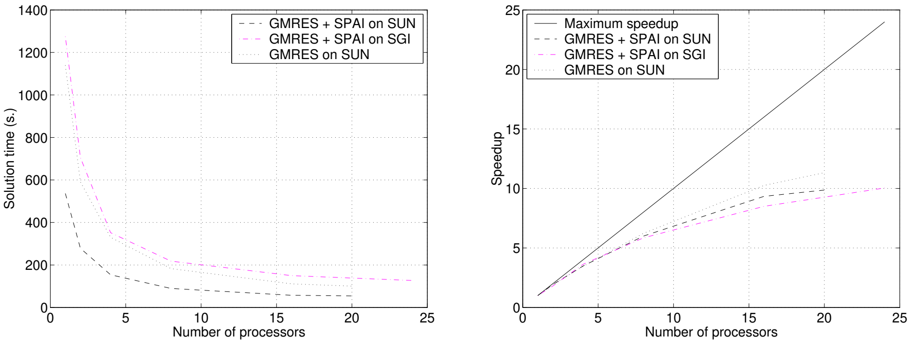

Figure 7.7: Scaling on RUND when the number of processors is increased. CPU time on the left and speedup on the right.

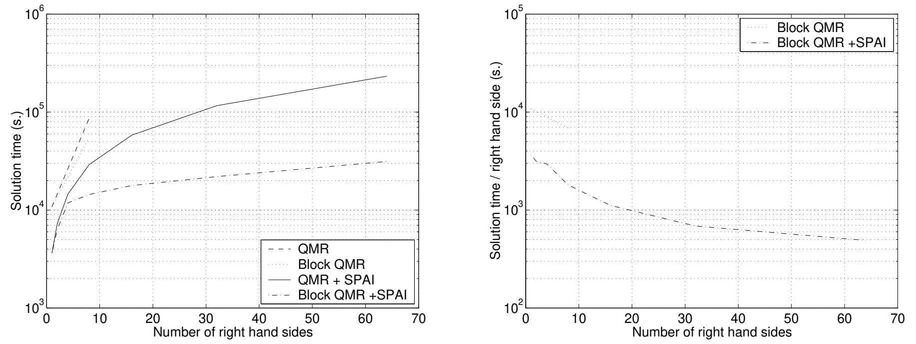

Figure 7.8: Scaling on RUND when the number of right hand sides is increased. CPU time on the left and solution time per right hand side on the right.

EFIE. We compare solution times when we increase the number of right hand sides with solving one right hand side at a time. There is a clear improvement in solution time when the number of right hand sides increases. Furthermore there is a gain when one uses SPAI as preconditioner.

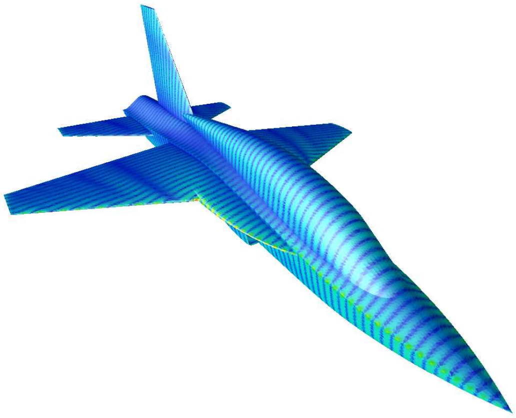

Figure 7.9: The aircraft model SAAB Trainer. The induced surface currents at 1 GHz.

Figure 7.10 compares the Radar Cross Section computed with EFIE and CFIE with the one obtained with a FETD solver [12] on the SAAB Trainer. Vertical polarization is used and we make the comparison at the two frequencies 375 MHz and 500 MHz. The aircraft is discretized with 191712 unknowns. The solutions agree except for the nose, where there is a visible difference between the results. In Figure 7.11 the same comparison is made between CFIE and FETD for horizontal polarization.

We investigate the speedup and scaling with several processors in Figure 7.12 on SGI for the SAAB Trainer. They are similar to the ones obtained on RUND. In this case there are fewer boxes on the lower levels compared to the sphere as well.

The largest simulation that we have done so far was to simulate a full scale jet fighter at 3 GHz with both vertical and horizontal polarization. The fighter was 11 m. long and the average length of a triangle edge was 17 mm. This resulted in a system with 1042644 unknowns. The problem was solved on an SGI 3800 with 500 MHz processors by Jonas Gustafsson at SAAB Avionics. It took 3 hours and 6 minutes on 24 processors and required 14.7 Gb of memory. The system converged in a total of 112 iterations when CFIE and GMRES with restart length 200 together with SPAI was used. The same system would have required 16 Tb of memory if it was solved with an explicitly generated system matrix. Figure 7.13 and Figure 7.14 show the surface currents on the aircraft for

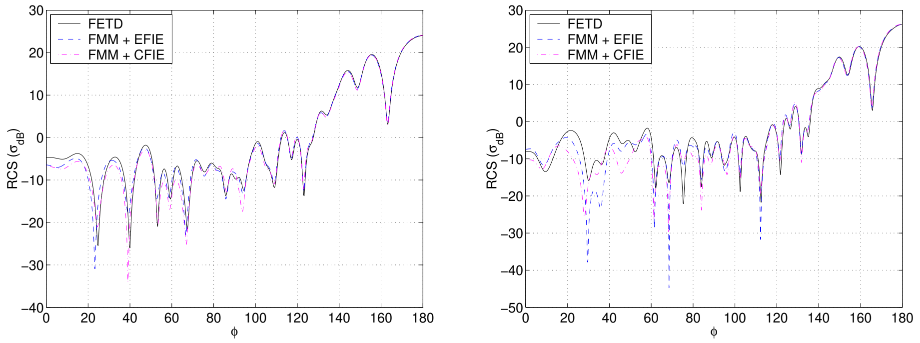

Figure 7.10: Radar Cross Section for SAAB Trainer vertical polarization. Results are obtained with the Fast Multipole Method and compared to a FETD code. The frequency is 375 MHz to the left and 500 MHz to the right.

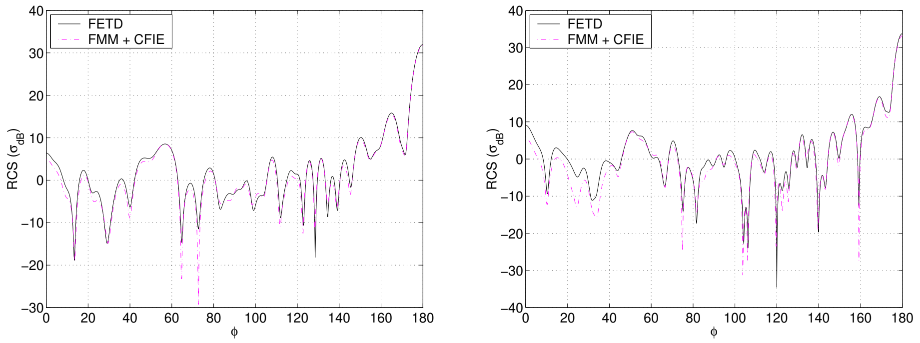

Figure 7.11: Radar Cross Section for SAAB Trainer horizontal polarization. The frequency is 375 MHz to the left and 500 MHz to the right.

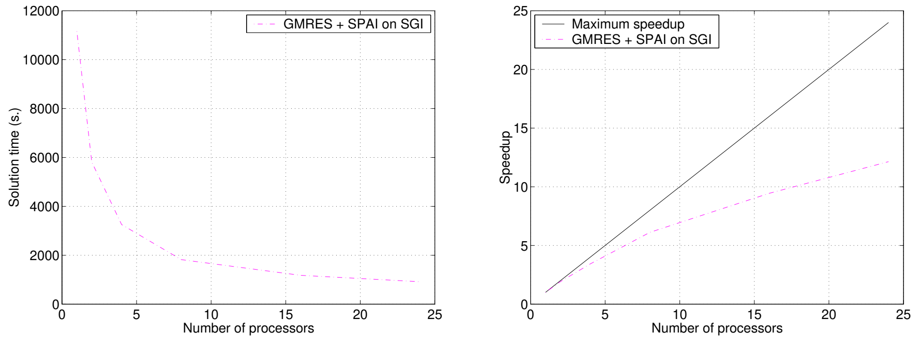

Figure 7.12: Scaling on SAAB Trainer when the number of processors is increased. CPU time on the left and speedup on the right.

the horizontal and vertical case. Figure 7.15 and Figure 7.16 shows the inside of the aircraft which has open air intakes.

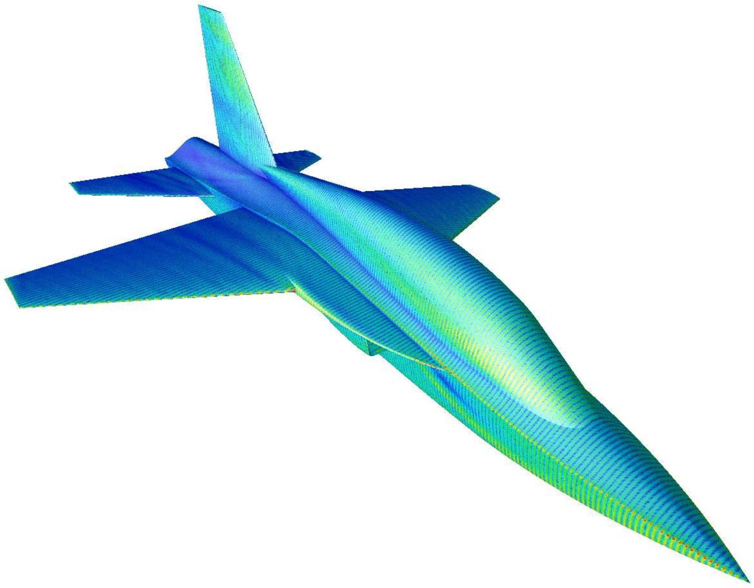

Figure 7.13: A full scale jet fighter at 3 GHz horizontal polarization.

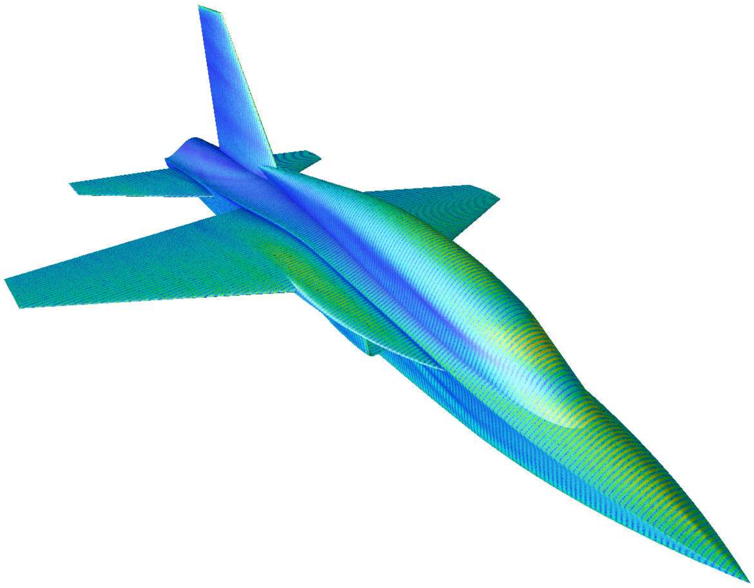

Figure 7.14: A full scale jet fighter at 3 GHz vertical polarization.

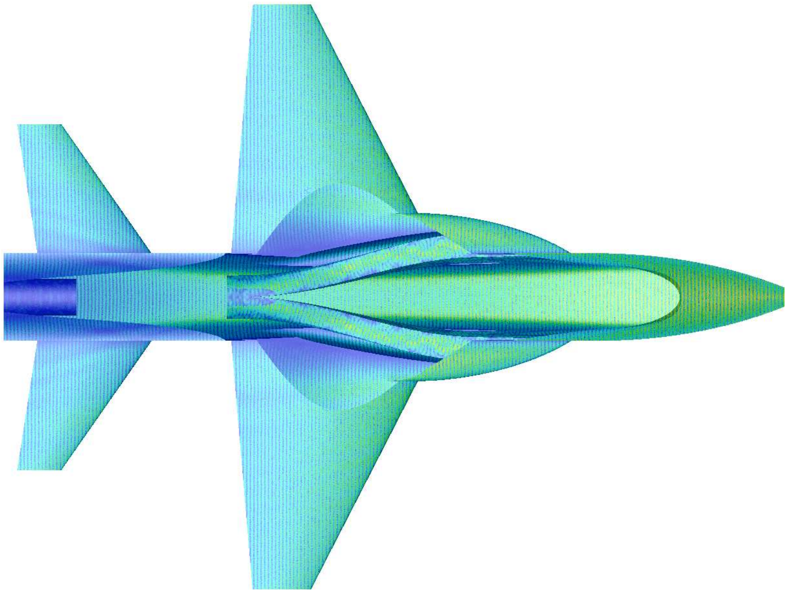

Figure 7.15: A view of the interior at 3 GHz vertical polarization.

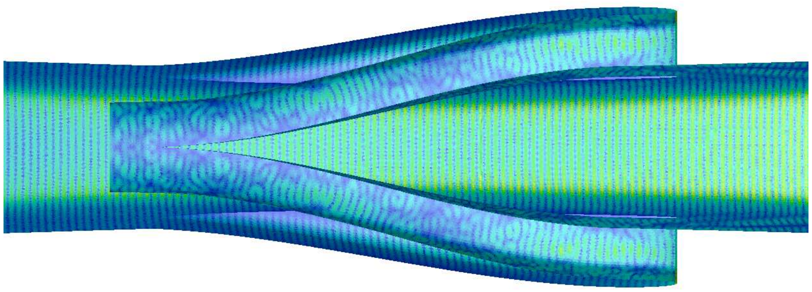

Figure 7.16: The air intakes at 3 GHz vertical polarization.

Chapter 8

Conclusions

An iterative solver for Maxwell's equations in the frequency domain has been developed. We use the multilevel Fast Multipole Method to reduce the time spent on matrix vector multiplications. The iterative method is preconditioned with a Sparse Approximate Inverse and for symmetric systems a block iterative solver was developed. The solver was parallelized on shared memory machines using OpenMP.

We have showed that the method works for several types of geometries, including a simulation with one million unknowns for a full scale aircraft. We gave examples where the preconditioner improved the convergence rate and also showed that the block iterative method improves the convergence rate for several right hand sides.

There are several issues that should be dealt with further. The parallelization could be improved. It would be interesting to show theoretical results on how the condition number improves with the preconditioner. We would also like to improve the strategy for solving several right hand sides.

Bibliography

[1] C. H. Ahn, W. C. Chew, J. S. Zhao, and E. Michielssen. Approximate inverse preconditioner for near resonant scattering problems. In *Antennas and Propagation Society International Symposium*, volume 3, pages 1546–1549, Atlanta, Georgia, USA, July 1998.

[2] Guillaume Alléon, Michele Benzi, and Luc Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numer. Algorithms*, 16(1):1–15, 1997.

[3] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists.* Academic Press, Inc., 4th edition, 1995.

[4] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[5] Abderrahmane Bendali. The Rumsey reaction principle. Lecture notes, February 2000.

[6] Abderrahmane Bendali, M'Barek Fares, and Jean Gay. A boundary–element solution of the Leontovitch problem. *IEEE Transactions on Antennas and Propagation*, 47(10):1597–1605, October 1999.

[7] Weng Cho Chew, Jian-Ming Jin, Eric Michielssen, and Jiming Song. *Fast and Efficient Algorithms in Computational Electromagnetics.* Artech House, Inc., Norwood, 2001.

[8] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for the wave equation: A pedestrian prescription. *IEEE Transactions on Antennas and Propagation*, 35(3):7–12, June 1993.

[9] Eric Darve. The fast multipole method: Numerical implementation. *Journal of Computational Physics*, 160(1):195–240, 2000.

[10] Fredrik Edelvik. *Finite Volume Solvers for the Maxwell Equations in Time Domain*. Licentiate thesis No. 2000-005, Department of Information Technology, Uppsala University, October 2000.

[11] Fredrik Edelvik and Gunnar Ledfelt. Explicit hybrid solver for the Maxwell equations in 3D. *J. Sci. Comput.*, 15(1), December 2000.

[12] Fredrik Edelvik and Gunnar Ledfelt. A comparison of time-domain hybrid solvers for complex scattering problems. *International Journal of Numerical Modeling*, 15(5), September/October 2002.

[13] Johan Edlund. Solving the electric field integral equation using a block LDLT method. MSc thesis UPTEC F99 007, Uppsala University, February 1999.

[14] Johan Edlund. *A Parallel, Iterative Method of Moments and Physical Optics Hybrid Solver for Arbitrary Surfaces*. Licentiate thesis No. 2001-010, Department of Information Technology, Uppsala University, August 2001.

[15] Stefan Erichsen and Stefan A. Sauter. Efficient automatic quadrature in 3-d Galerkin BEM. *Computer Methods in Applied Mechenics and Engineering*, 157(3–4):215–224, May 1998.

[16] Roland W. Freund. Conjugate gradient–type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comput.*, 13(1):425–448, 1992.

[17] Roland W. Freund. Transpose–free quasi–minimal residual methods for non-hermitian linear systems. In *Recent advances in iterative methods*, volume 60 of *IMA Vol. Math. Appl.*, 1994.

[18] Roland W. Freund and Manish Malhotra. A block–QMR algorithm for non–hermitian linear systems with multiple right–hand sides. *Linear Algebra Appl.*, 254:119–157, 1997.

[19] Roland W. Freund, Manish Malhotra, and Peter M. Pinsky. Iterative solution of multiple radiation and scattering problems in structural acoustics using a block quasi–minimal algorithm. *Comput. Methods Appl. Mech. Engrg.*, 146(1-2):173–196, 1997.

[20] Roland W. Freund and Noël M. Nachtigal. An implementation of the QMR method based on coupled two–term recurrences. *SIAM J. Sci. Stat. Comput.*, 15(2):313–337, 1994.

[21] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, London, 3rd edition, 1996.

[22] Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[23] Marcus J. Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, 1997.

[24] George C. Hsiao and Ralph E. Kleinman. Mathemathical foundations for error estimation in numerical solutions of integral equations in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 45(3):316–328, March 1997.

[25] J. Jin. *The Finite Element Method in Electromagnetics.* Wiley, New York, 1993.

[26] Joseph B. Keller. Geometrical theory of diffraction. *J. Opt. Soc. Amer.*, 52(2):116–130, February 1962.

[27] Jean-Claude Nédélec. *Acoustic and Electromagnetic Equations.* Springer-Verlag, New York, Berlin, Heidelberg, 2001.

[28] Martin Nilsson. Solving the electric field integral equation using a sparse approximate inverse preconditioned iterative method. MSc thesis UPTEC F99 004, Uppsala University, February 1999.

[29] Martin Nilsson. A fast multipole accelerated block quasi minimum residual method for solving scattering from perfectly conducting bodies. In *Antennas and Propagation Society International Symposium*, volume 4, pages 1848–1851, Salt Lake City, Utah, USA, July 2000.

[30] Martin Nilsson. A fast multipole method solver for large scale scattering problems. In Fredrik Edelvik et al., editors, *EMB 01 – Electromagnetic Computations – Methods and Applications*, pages 148–155, Uppsala, Sweden, November 2001. SNRV.

[31] OpenMP Architecture Review Board. *OpenMP Fortran Application Program Interface, Version 2.0*, November 2000. www.openmp.org (2002-04-05).

[32] P. H. Pathak and R.G. Kouyoumjian. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proceedings of the IEEE*, 62(11):1448–1461, November 1974.

[33] Andrew F. Peterson, Scott L. Ray, and Raj Mittra. *Computational Methods for Electromagnetics.* IEEE Press and Oxford University Press, New York, Oxford, Tokyo, Melbourne, 1998.

[34] S. M. Rao, D. R. Wilton, and A. W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagation*, 30(3):409–418, May 1982.

[35] D. J. Riley and C. D. Turner. VOLMAX: A solid-model-based, transient volumetric Maxwell solver using hybrid grids. *IEEE Antennas Propagat. Magazine*, 39(1):20–33, 1997.

[36] Vladimir Rokhlin. Diagonal forms of translation operators for the Helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1(1):82–93, 1993.

[37] V. H. Rumsey. Reaction concept in electromagnetic theory. *Physical Review*, 94(6):1483–1491, June 1954.

[38] V. Simoncini. A stabilized QMR version of Block BiCG. *SIAM J. Matrix Anal. Appl.*, 18(2):419–434, 1997.

[39] V. Simoncini and E. Gallopoulos. Iterative methods for complex symmetric systems with multiple right–hand sides. Technical Report 1322, Center for Supercomputing Research and Development, University of Illinois at Urbana–Champaign, Dec. 1993.

[40] Jiming Song and Weng Cho Chew. Multilevel fast multipole algorithm for solving combined field integral equation of electromagnetic scattering. *Microwave and Optical Technology Letters*, 10(1):14–19, September 1995.

[41] A. Taflove. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House, Boston, MA, 1995.

[42] Stephen A. Vavasis. Preconditioning for boundary integral equations. *SIAM J. Matrix Anal. Appl.*, 13(3):905–925, 1992.

[43] Michael S. Warren and John K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 87(1–2):266–290, May 1995.

Recent licentiate theses from the Department of Information Technology

2001-008 Markus Bylund: *Personal Service Environments — Openness and User Control in User-Service Interaction*

2001-009 Johan Bengtsson: *Efficient Symbolic State Exploration of Timed Systems: Theory and Implementation*

2001-010 Johan Edlund: *A Parallel, Iterative Method of Moments and Physical Optics Hybrid Solver for Arbitrary Surfaces*

2001-011 Pär Samuelsson: *Modelling and control of activated sludge processes with nitrogen removal*

2001-012 Per Åhgren: *Teleconferencing, System Identification and Array Processing*

2001-013 Alexandre David: *Practical Verification of Real-time Systems*

2001-014 Abraham Zemui: *Fourth Order Symmetric Finite Difference Schemes for the Wave Equation*

2001-015 Stefan Söderberg: *A Parallel Block-Based PDE Solver with Space-Time Adaptivity*

2001-016 Johan Furunäs Åkesson: *Interprocess Communication Utilising Special Purpose Hardware*

2002-001 Eva Mossberg: *Higher Order Finite Difference Methods for Wave Propagation Problems*

2002-002 Anders Berglund: *On the Understanding of Computer Network Protocols*

2002-003 Emad Abd-Elrady: *Harmonic Signal Modeling Based on the Wiener Model Structure*

2002-004 Martin Nilsson: *Iterative Solution of Maxwell's Equations in Frequency Domain*

UPPSALA
UNIVERSITET

University Printers, Uppsala University, Sweden